UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DEPARTAMENTO DE INGENIERÍA CARTOGRÁFICA, GEODESIA Y FOTOGRAMETRÍA

TESIS DOCTORAL

Desarrollo e implementación de algoritmos para clasificar datos LiDAR en áreas urbanas

Autor:

Sánchez Lopera, José

Director:

Lerma García, José Luis

Valencia, Septiembre 2015

Agradecimientos

En primer lugar, quiero expresar mi más sincero agradecimiento al director de la tesis, el Dr. José Luis Lerma García, no solo por su comprometida dedicación, paciencia, orientación científica y apoyo en la realización de la presente tesis, sino también por todos sus consejos, colaboración y el aporte invalorable de su experiencia que ha realizado como mi mentor y que ha sido una guía constante a lo largo de la última década.

A pesar de no conocerlo en persona, quisiera agradecer al Dr. John Burkardt toda la información aportada en el ámbito de la geometría computacional. Las librerías de libre distribución y la documentación adjunta en su web, han sido imprescindibles para el desarrollo de los algoritmos asociados a la organización de datos en estructuras espaciales propuestos en el presente trabajo. También quisiera reconocer la aportación de los datos test LiDAR por parte de las empresas *Dielmo 3D S.L* y *Stereocarto S.L* sin los cuales no habría sido posible evaluar la bondad de los algoritmos implementados.

Agradecer a mi familia todo el apoyo incondicional recibido desde que comencé mis estudios de tercer ciclo. En especial a mis padres, que siempre han creído en la importancia de la formación de sus hijos.

Por último dar las gracias de corazón a mi esposa por su apoyo y comprensión durante esta etapa. Sin su aliento constante no habría sido posible culminar este trabajo.

Resumen

Light Detection and Ranging (LiDAR) es una técnica fotogramétrica y de teledetección frecuentemente utilizada para obtener con rapidez y precisión densa información 3D de la superficie terrestre. La alta densidad de información espacial tridimensional que permite capturar el sistema se utiliza en una amplia gama de aplicaciones en áreas urbanas, rurales y forestales.

En los últimos años, la tecnología LiDAR ha sido ampliamente utilizada en diferentes aplicaciones dentro de entornos urbanos tales como la actualización cartográfica, el análisis de comunicaciones, el modelado de ciudades virtuales, la evaluación de daños producidos por catástrofes naturales y la simulación de inundaciones. En todas las aplicaciones mencionadas resulta necesario detectar las distintas entidades que conforman la escena como paso previo. En este sentido, conseguir una clasificación automática es uno de los aspectos que mayor dificultad presenta debido a la gran variedad de objetos naturales y artificiales, así como estructuras de diferentes tamaños, colores, texturas y materiales existentes en áreas urbanas.

La creciente demanda de un algoritmo rápido, eficiente y automático para extraer características urbanas en tres dimensiones motivó el desarrollo de la presente tesis. Concretamente, el objetivo de la investigación es el desarrollo e implementación de algoritmos eficientes en el ámbito de la clasificación de entidades en áreas urbanas complejas que posean una dependencia mínima con respecto a los parámetros umbrales preestablecidos, con el fin de conseguir así un mayor grado de automatización y una mayor homogeneidad en los resultados. Además, el diseño de todos los algoritmos tuvo en cuenta dos premisas: la utilización únicamente de datos LiDAR sin otras fuentes de datos adicionales, así como el uso de datos LiDAR brutos dispuestos de forma aleatoria sin realizar ningún tipo de interpolación a malla regular, evitando así la perdida de precisión asociada a la generalización de información ligada al proceso de interpolación.

Los algoritmos desarrollados abarcan la totalidad de fases propias del tratamiento y pre-procesado de datos LiDAR, desde la organización de la nube de puntos en estructuras espaciales de datos hasta la definición del vecindario, la segmentación en superficies continuas, la detección y filtrado de errores groseros y la clasificación de las distintas entidades que componen la escena. Asimismo, toda la metodología propuesta es totalmente automática y se ha implementado en C++.

El proceso de clasificación propuesto detecta un total de cuatro entidades: terreno, puentes, edificios y vegetación junto con pequeños objetos. Para identificar cada clase diferentes algoritmos han sido desarrollados y evaluados sobre dos zonas test urbanas de gran complejidad. Los datos test abarcan un área total de 1.231.095 m^2 y contienen edificios residenciales rodeados de vegetación, altas edificaciones, naves industriales, carreteras, puentes, canales, zonas verdes y árboles. Por lo que resultan idóneos para evaluar la dependencia de la precisión alcanzada con respecto a las características de la escena.

Para filtrar errores groseros se han desarrollado distintos algoritmos basados en el análisis del vecindario en cuanto a distribución de altitudes. En este sentido, el que obtiene mejores resultados utiliza un enfoque multiproceso con parámetros umbrales adaptativos, alcanzando una exactitud global de la clasificación del 99,9%.

El terreno es clasificado mediante un nuevo método de densificación que combina técnicas de segmentación y características propias de otros métodos como filtros morfológicos y filtros basados en interpolación de superficies. La inclusión de análisis adicionales centrados en el tamaño y posición relativa de las regiones obtenidas tras la segmentación mejora la clasificación, consiguiendo una media de errores para ambas zonas test menor al 0,5 %.

La clasificación de puentes se realiza mediante la localización de bordes y el posterior análisis direccional de la continuidad estructural. De nuevo, los mejores resultados se alcanzan al utilizar un algoritmo multiproceso con parámetros adaptativos que permite además clasificar puentes que discurren sobre agua y que por tanto son colindantes a zonas de sombra.

Para diferenciar los edificios de la vegetación y otros pequeños objetos se introduce el nuevo concepto de clasificador angular. Este clasificador analiza la distribución de los puntos de tierra en torno a pequeñas regiones todavía no clasificadas para determinar si la agrupación pertenece a un edificio o a vegetación. La clasificación más precisa se obtiene al combinar el clasificador angular con técnicas de análisis de texturas, consiguiendo una media de errores en la clase edificio que oscila entre 0,40 % y 1,52 % dependiendo de la zona test.

Los resultados experimentales confirman la alta precisión conseguida en la clasificación automática de entidades en zonas urbanas complejas. En este sentido, la presente tesis realiza distintos aportes individuales en cada uno de los ámbitos tratados. Además, cabe destacar que los resultados más precisos se obtienen al utilizar algoritmos multiproceso que ajustan automáticamente los parámetros umbrales, o bien al combinar algoritmos con enfoques distintos. Siendo este aspecto uno de los mayores aportes de la investigación.

Resum

Light Detection and Ranging (LiDAR) és una tècnica fotogramètrica i de teledetecció freqüentment utilitzada per obtenir amb rapidesa i precisió densa informació 3D de la superfície terrestre. L'alta densitat d'informació espacial tridimensional que permet capturar el sistema s'utilitza en una àmplia gamma d'aplicacions en àrees urbanes, rurals i forestals.

En els últims anys, la tecnologia LiDAR ha estat àmpliament utilitzada en diferents aplicacions dins d'entorns urbans com ara l'actualització cartogràfica, l'anàlisi de comunicacions, el modelat de ciutats virtuals, l'avaluació de danys produïts per catàstrofes naturals i la simulació d'inundacions. En totes les aplicacions esmentades resulta necessari detectar les diferents entitats que conformen l'escena com a pas previ. En aquest sentit, aconseguir una classificació automàtica és un dels aspectes que més dificultat presenta a causa de la gran variat d'objectes naturals i artificials, així com estructures de diferents mides, colors, textures i materials existents en les àrees urbanes.

La creixent demanda d'un algoritme ràpid, eficient i automàtic per extraure característiques urbanes en tres dimensions va motivar el desenvolupament d'aquesta tesi. Concretament, l'objectiu de la investigació és el desenvolupament i implementació d'algoritmes eficients en l'àmbit de la classificació d'entitats en àrees urbanes complexes que posseïxen una dependència mínima pel que fa als paràmetres umbrals preestablerts, per tal d'aconseguir així un major grau de automatització i una major homogeneïtat en els resultats. A més, el disseny de tots els algoritmes va tenir en compte dues premisses: la utilització únicament de dades LiDAR sense altres fonts de dades addicionals, així com l'ús de dades LiDAR bruts disposats de forma aleatòria sense realitzar cap tipus d'interpolació a malla regular, evitant així la pèrdua de precisió associada a la generalització d'informació lligada al procés d'interpolació.

Els algoritmes desenvolupats comprenen la totalitat de fases pròpies del tractament i pre-processat de dades LiDAR, des de l'organització del núvol de punts en estructures espacials de dades fins a la definició del veïnat, segmentació en superfícies contínues, detecció i filtrat d'errors grossers i la classificació de les diferents entitats que componen l'escena. Així mateix, tota la metodologia proposada és totalment automàtica i s'ha desenvolupat en C++.

El procés de classificació proposat detecta un total de quatre entitats: terreny, ponts, edificis i vegetació juntament amb petits objectes. Per identificar cada classe diferents algoritmes han estat desenvolupats i avaluats sobre dues zones test urbanes de gran complexitat. Les dades test comprenen un àrea total d'1.231.095 m^2 i contenen edificis residencials envoltats de vegetació, altes edificacions, naus industrials, carreteres, ponts, canals, zones verds i arbres. Per tant, resulten idonies per avaluar la dependència de la precisió aconseguida respecte a les característiques de l'escena.

Per filtrar errors grossers s'han desenvolupat diferents algoritmes basats en l'anàlisi del veïnat quant a distribució d'altituds. En aquest sentit, el que obté millors resultats utilitza un enfocament multiprocés amb paràmetres umbrals adaptatius, aconseguint una exactitud global de la classificació del 99,9%.

El terreny és classificat mitjançant un nou mètode de densificació que combina tècniques de segmentació i característiques pròpies d'altres mètodes com filtres morfològics i filtres basats en interpolació de superfícies. La inclusió d'anàlisis addicionals centrades en la grandària i posició relativa de les regions obtingudes després de la segmentació millora la classificació, aconseguint una mitjana d'errors per a ambdues zones test menor al 0,5 %.

La classificació de ponts es realitza mitjançant la localització de vores i la posterior anàlisi direccional de la continuïtat estructural. De nou, els millors resultats s'aconsegueixen amb la

utilització d'un algoritme multiprocés amb paràmetres adaptatius que permet a més classificar ponts que discorren sobre aigua i que per tant són adjacents a zones d'ombra.
Per diferenciar els edificis de la vegetació i altres petits objectes s'introdueix el nou concepte de classificador angular. Aquest classificador analitza la distribució dels punts de terra al voltant de petites regions encara no classificades per determinar si l'agrupació pertany a un edifici o a vegetació. La classificació més precisa s'obté combinant el classificador angular amb tècniques d'anàlisis de textures, aconseguint una mitjana d'errors a la classe edifici que oscil·la entre 0,40 % i 1,52 % depenent de la zona test.

Els resultats experimentals confirmen l'alta precisió aconseguida en la classificació automàtica d'entitats en zones urbanes complexes. En aquest sentit, la present tesi realitza diferents aportacions individuals en cadascun dels àmbits tractats. A més, cal destacar que els resultats més precisos s'obtenen utilitzant algoritmes multiprocés que ajusten automàticament els paràmetres umbrals, o bé combinant algoritmes amb enfocaments diferents. Aquest aspecte es una de les majors aportacions de la investigació.

Abstract

Light detection and ranging (LiDAR) is a remote-sensing technique used to obtain three-dimensional (3D) information of the Earth quickly and accurately. A high density of 3D spatial information can be acquired by lidar-based systems and used for a wide range of applications in urban, semi-urban, and rural areas.

In recent years, LiDAR technology systems have been intensively used in different urban applications such as map updating, communication analysis, virtual city modelling, natural disaster damage assessment and flood modelling. In all applications mentioned is a prerequisite to detect the different objects in the scene. In this regard, one of the most challenging topics is considered to be automatic object classification due to the large variety of natural and man-made objects, structures, and artefacts of different sizes, colours, textures, and materials available in urban areas.

The increasing demand for a fast, efficient and automatic algorithm to extract three-dimensional urban features was the motive behind this thesis. Specifically, the objective of the research is the development and implementation of efficient algorithms in the field of classification of objects in complex urban areas with minimal dependence on preset thresholds, in order to achieve a greater degree of automation and greater consistency in the results. Furthermore, the algorithm design took into account two assumptions: the employment of LiDAR data without additional sources of data and the use of raw LiDAR data without any type of regular grid interpolation, thereby averting the loss of precision associated with the generalization of information linked to the interpolation process.

The developed algorithms cover all phases of processing and pre-processing of LiDAR data, from the organization of the point cloud in spatial data structures to the neighbourhood definition, segmentation of data in continuous surfaces, filtering and outlier detection and the classification of different entities that make up the scene. Furthermore , the whole pipeline proposed herein is fully automatic and is developed in C++.

The proposed classification process detects four entities: bare earth, bridges, buildings and vegetation with small objects. To identify each class different algorithms have been developed and evaluated on two complex urban areas. The test data covers 1,231,095 m^2 and include residential buildings surrounded by vegetation, high buildings, industrial buildings, a network of local roads, bridges, canals, open and green areas, as well as trees. Making them suitable to evaluate the dependence of the accuracy achieved with respect to the characteristics of the scene.

To filter outliers different algorithms have been developed based on analysis of the neighborhood in terms of distribution of altitudes. In this regard, the best results are achieved by using a multiprocess approach with adaptive thresholds, achieving an overall classification accuracy of 99.9%.

Bare earth is classified using a new densification method that combines segmentation techniques and features of other methods such as morphological filters and interpolation-based Methods. The inclusion of additional analysis focused on the size and relative position of the clusters obtained after segmentation improves the classification, achieving an average of errors less than 0,5 % for both data test.

The classification of bridges is carried out by detecting edges and directional analysis of the structural continuity. Once again, the best results are achieved by using a multiprocess algorithm with adaptive thresholds, that also allows classifies bridges extending over the water and therefore are adjacent to shaded areas.

To differentiate the buildings from vegetation and other small objects is introduced the new concept of angular classification. This classifier analyses the distribution of bare-earth points around unclassified point clusters to determine whether a cluster can be classified either as building or as vegetation. The more precise classification is obtained by combining the angular classifier with texture analysis techniques, achieving an average errors in building class in the range of 0,40 % and 1,52 %, depending on the application zone.

The experimental results confirm the high accuracy achieved to automatically classify urban objects in complex areas. In this regard, the thesis reports different contributions in each of the topics discussed. Furthermore, it is noteworthy that the best results are obtained by multiprocess algorithms with adaptive thresholds or combining algorithms with different approaches. This latter aspect is one of the greatest contributions of the research proposed.

Índice:

Índice de figuras y tablas

ÍNDICE DE FIGURAS:

ÍNDICE DE TABLAS:

I. INTRODUCCIÓN

I. 1. Objetivo de la tesis y descripción general de la misma

Cumpliendo con la normativa vigente en el Departamento de Ingeniería Cartográfica, Geodesia y Fotogrametría, se ha realizado la presente tesis con el fin de obtener el título de doctor en Geodesia y Cartografía. La tesis doctoral se centra en la tecnología LiDAR ('*Light Detection and Ranging*'), concretamente, en la investigación y desarrollo de algoritmos para el tratamiento de datos LIDAR en cascos urbanos.

La tecnología LiDAR es una técnica fotogramétrica y de teledetección frecuentemente utilizada para obtener con rapidez y precisión densa información 3D de la superficie terrestre. La alta densidad de información espacial tridimensional se utiliza en una amplia gama de aplicaciones en áreas urbanas, rurales y forestales.

En entornos urbanos, disponer de cartografía tridimensional (3D) resulta de gran utilidad, ya que los distintos proyectos pueden calcularse, analizarse y mostrarse fielmente junto con su entorno. Además, disponer de amplia información 3D en áreas urbanas es esencial en muy diversos ámbitos tales como el planeamiento urbanístico, la gestión, el análisis territorial y medioambiental, el catastro, etc. En este sentido, en la última década la tecnología LiDAR se ha alzado como una potente herramienta para obtener de forma eficiente densas nubes de puntos 3D de alta precisión que sirven como punto de partida a multitud de aplicaciones asociadas tanto a la generación y la actualización de cartografía urbana 3D, así como a otras aplicaciones relacionadas con el medio ambiente y todo tipo de modelos de simulación, donde la información altimétrica cobra gran relevancia.

Una de las aplicaciones más extendidas de la tecnología LiDAR en entornos urbanos es la generación de MDT (Modelos Digitales del Terreno). En este aspecto, multitud de autores han desarrollado algoritmos para su obtención de forma directa a partir de datos LiDAR (Briese et al. 2002; Akel et al. 2003; Brovelli et al. 2004; Kim y Shan 2011; Susaki 2012; Chen et al. 2012). Estos precisos MDT obtenidos mediante tecnología LiDAR son utilizados en aplicaciones tan diversas como la simulación de inundaciones (Sole et al. 2012) o el cálculo de alturas de edificaciones para realizar reconstrucciones de edificios y modelos virtuales de ciudades (Arefi et al. 2008).

Por otro lado, disponer de cartografía urbana actualizada es muy útil tanto para los ciudadanos como, sobre todo, para la propia gestión municipal. Ésto ha propiciado la necesidad de actualizar periódicamente la cartografía existente, y si la actualización no se realiza de manera sistemática y mediante procedimientos automatizados puede suceder que la actualización de la cartografía no se efectúe por su complejidad, que no sea rentable y que, eventualmente, se opte por su renovación completa. Consecuentemente, con el fin de abaratar costes en la producción cartográfica urbana y catastral y mantener la cartografía lo más actualizada posible, muchos han sido los autores que han propuesto metodologías para

detectar cambios a partir de datos LiDAR y localizar así aquellas áreas urbanas donde es necesaria una actualización cartográfica (Vosselman et al. 2004; Rottensteiner 2008; Chaabouni-Chouayakh et al. 2010; Matikainen et al. 2010; Sanchez y Lerma 2012). La detección de cambios mediante el uso de datos LiDAR no solamente tiene como objetivo la actualización cartográfica, sino que gran parte de los estudios existentes utilizan la detección de cambios para evaluar el nivel de destrucción de catástrofes naturales, como terremotos o tsunamis en una determinada región (Vögtle y Steinle 2004; Vu et al. 2004; Rehor et al. 2008; Trinder y Salah 2012).

En los últimos años la densa información tridimensional proporcionada por los datos LiDAR está siendo utilizada para transformar cartografía 2D a 3D (Sanchez y Lerma 2012) y generar de forma automática modelos 3D de ciudades (Vosselman y Dijkman 2001; Alharthy y Bethel 2004; Forlani et al. 2006; Dorninger y Nothegger 2007; Lesparre y Gorte 2012; Kwak et al. 2012). Estos modelos tienen multitud de aplicaciones, tales como la planificación de redes de comunicaciones, análisis de visibilidad y simulación del clima urbano (Lesparre y Gorte 2012). Además, la tecnología LiDAR también puede ser utilizada para generar de forma automática o semiautomática ortofotografías en entornos urbanos consiguiendo así reducir costes de producción y aumentando la automatización del proceso (Keinan y Doytsher 2008; Moussa y Sheimy 2012).

En la totalidad de aplicaciones mencionadas resulta necesario detectar las distintas entidades que conforman la escena como paso previo. Las entidades a detectar varían en función de la utilidad final de los datos LiDAR, no obstante, para la gran mayoría de aplicaciones es imprescindible detectar tres entidades básicas: terreno, edificios y vegetación. En concreto, la detección de las tres entidades tiene carácter obligatorio en aquellos casos donde se precisa extraer información de los edificios o de la vegetación, ya que en estos casos, tras detectar y filtrar el terreno en una primera fase, se procede a diferenciar entre vegetación y edificios en el conjunto de datos restantes. Entre las aplicaciones que precisan detectar estas tres entidades básicas destacan aquellas relacionadas con la generación y actualización de cartografía en entornos urbanos, detección de cambios, modelado urbano 3D y generación automática de ortofotografías. Por el contrario, existen algunas aplicaciones donde únicamente se necesita diferenciar entre terreno y no terreno, sin entrar en detalle sobre las diferentes entidades que conforman la clase no terreno. En este caso, destacan aquellas aplicaciones relacionadas con la generación automática de modelos digitales del terreno. Por último, en aquellos casos donde la tecnología LiDAR es utilizada para generar modelos hidrológicos, también es necesario detectar los puentes y clasificarlos como entidad propia.

Dado que la tecnología LiDAR únicamente permite capturar nubes densas de puntos en 3D sin ningún tipo de información semántica, no es posible determinar de forma directa el tipo de objeto al que pertenece cada uno de los puntos capturados por el sensor. Por consiguiente, la discriminación entre las distintas entidades que conforman la escena en el conjunto de datos LiDAR es una tarea no exenta de dificultad. En este sentido, durante las dos últimas décadas multitud de algoritmos han sido desarrollados con diversos resultados. En la detección del terreno destacan metodologías tan diversas como la utilización de filtros morfológicos (Voselman 2000; Arefi y Hahn 2005; Kobler et al. 2007; Kim y Shan 2012; Li 2013), la aplicación de filtros de densificación progresiva de TIN (Axelsson 2000; Sohn y Dowman 2002; Ekhtari et al. 2008; Pérez et al. 2012), la utilización de filtros basados en interpolación de superficies (Pfeifer et al. 2001; Briese et al. 2002; Brovelli et al. 2004 y Evans y Hudak 2007) y el uso de filtros basados en segmentación (Tovari y Pfeifer 2005; Filin y Pfeifer 2006; Ghosh y Lohani 2011; Ural y Shan 2012). En cuanto a la diferenciación entre la clase vegetación y la clase edificio, como norma general la mayoría de los autores primero separan los puntos de tierra desnuda antes de extraer los objetos adicionales. Para detectar edificios y separarlos de la vegetación es muy frecuente la utilización de filtros de segmentación (Alharthy y Bethel 2004; Ekhtari et al. 2008). Además, también resulta de uso común la aplicación de análisis de texturas para diferenciar entre superficies lisas o

rugosas con el fin de separar los edificios de la vegetación (Maas 1999; Oude Elberink y Maas 2000; Kwak et al. 2012). Por último, una de las metodologías de uso cada vez más frecuente consiste en combinar los datos LiDAR con otras fuentes de datos como pueden ser imágenes aéreas (Lari et al. 2011; Moussa y Sheimy 2012) o imágenes multiespectrales (Haala y Brenner 1999; Vögtle y Steinle 2000; Matikainen et al. 2010; Moussa y Sheimy 2012; Wang y Li 2013).

La multitud y diversidad de algoritmos para diferenciar las distintas entidades que conforman la escena en el conjunto de datos LiDAR desarrollados en los últimos años, es una consecuencia directa de la dificultad a la hora de alcanzar un alto grado de automatización y homogeneidad en la precisión de la clasificación. Según señalan distintos investigadores (Sithole y Vosselman 2004; Meng et al. 2010; Chen et al. 2012) el resultado del proceso de filtrado y clasificación en datos LiDAR es fuertemente dependiente de las características y complejidad de la escena, donde la existencia de objetos complejos tales como puentes o edificaciones, así como la presencia de pendientes del terreno pronunciadas y vegetación variable condicionan la precisión del resultado. Esta variabilidad en los resultados en función de las características de la escena viene condicionada por la selección de los distintos parámetros (umbrales) utilizados por los algoritmos. No obstante, la selección de los mejores parámetros no siempre es sencilla y suele requerir múltiples ensayos, sobre todo, en aquellas zonas donde las características del terreno no son uniformes como puede ser en entornos urbanos (Kim y Shan 2011; Pérez et al. 2012; Li 2013). Además de la falta de automatización asociada a la dificultad en la selección de los parámetros umbrales en entornos urbanos, existen otros aspectos a mejorar en este ámbito, como es la eficiencia en la diferenciación entre árboles y edificaciones cuando son colindantes (Matikainen et al. 2004; Matikainen et al. 2009).

La presente tesis, trata de solucionar los aspectos anteriores mediante el desarrollo de nuevos algoritmos diseñados para actuar sobre datos LiDAR pertenecientes a entornos urbanos. En este sentido, el objetivo es la investigación y el desarrollo de algoritmos eficientes en el ámbito de la clasificación de entidades en zonas urbanas complejas que tengan una mínima dependencia con respecto a los parámetros umbrales preestablecidos, con el fin de conseguir así un mayor grado de automatización y una mayor homogeneidad en los resultados. Los algoritmos a desarrollar deberán actuar únicamente sobre datos LiDAR sin utilizar otras fuentes de datos adicionales. Las clases que los algoritmos a desarrollar deberán diferenciar serán cuatro: terreno, puentes, edificación y vegetación junto con pequeños objetos urbanos. Los algoritmos a desarrollar actuarán siempre sobre los datos brutos LiDAR dispuestos de forma aleatoria, sin realizar ningún tipo de interpolación a malla regular, evitando así la perdida de precisión asociada a la generalización de información ligada al proceso de interpolación. En consecuencia, será necesario desarrollar aplicaciones que comprendan la totalidad de las fases en el tratamiento y pre-procesado de datos LiDAR, desde la disposición y codificación de datos, pasando por el desarrollo de sistemas de detección del vecindario hasta la segmentación, el filtrado de valores atípicos y la clasificación de las distintas entidades que componen la escena.

El trabajo se organiza en diferentes apartados. Tras una breve descripción de la tecnología LiDAR se detallan algunos aspectos técnicos como la elección del lenguaje de programación y la descripción de los datos test empleados (apartado I), seguidamente se expone una exhaustiva revisión de los antecedentes que están relacionados con los diferentes aspectos a tratar (apartado II). La metodología y los algoritmos desarrollados aparecen en el apartado III, donde los algoritmos son presentados tras un breve análisis de su relación con los antecedentes. El apartado IV comprende el proceso de análisis de resultados y discusión. Las conclusiones y futuras líneas de investigación se presentan en apartado V, finalmente, tras las referencias bibliográficas (apartado VI) se adjuntan dos anexos. El primero de ellos (Anexo I) se corresponde con informes de precisión y errores; en él aparecen las matrices

de confusión así como diferentes gráficos de errores y precisión alcanzada para cada una de las clasificaciones obtenidas. El segundo (Anexo II) contiene los mapas de resultados.

I. 2. Descripción general de la tecnología LiDAR

La utilización del láser como instrumento de captura de datos cartográficos o como herramienta para la teledetección se remonta más de 40 años. Townes y Schawlow desarrollaron la teoría del láser óptico en el año 1958. Sin embargo, el concepto de LiDAR tal y como se conoce hoy, no aparece hasta la década de 1970 cuando la NASA utilizó los primeros prototipos aéreos orientados a obtener datos y propiedades de la atmósfera, del agua, el hielo y de los árboles (Flood 2001) . El uso y desarrollo de esta tecnología siguió evolucionando, y en torno a 1985 la utilidad del LiDAR como herramienta para obtener precisas mediciones de la superficie terrestre estaba ampliamente probada.

El LiDAR es un sensor activo láser que se sitúa sobre una plataforma aerotransportada, junto a un GPS y un sistema de navegación inercial (INS). El distanciómetro láser va colocado junto a un espejo oscilante que desvía el haz de forma perpendicular a la trayectoria del avión, de tal manera que el movimiento lateral combinado con el movimiento de avance del avión permite realizar un barrido del terreno.

La tecnología LIDAR se basa en la emisión continua de pulsos láser y en la medición del tiempo que tardan dichos pulsos en reflejarse en la superficie y ser devueltos hasta el sensor. Los datos LIDAR constituyen una nube de puntos tridimensional que cubren todo el terreno barrido por el láser. La metodología para obtener coordenadas absolutas de los puntos barridos por el láser, se basa en que conocemos las coordenadas del avión y su trayectoria, así como el ángulo del espejo oscilante. Todo ello es posible debido a que la plataforma aerotransportada lleva instalado un *Global Navigation Satellite System* (GNSS) () y un sistema de navegación inercial (INS) . El GNSS dota a los puntos de coordenadas absolutas mediante mediciones en modo cinemático diferencial, mientras que el INS permite interpolar dichas coordenadas a la totalidad de la trayectoria del avión (Figura 1).

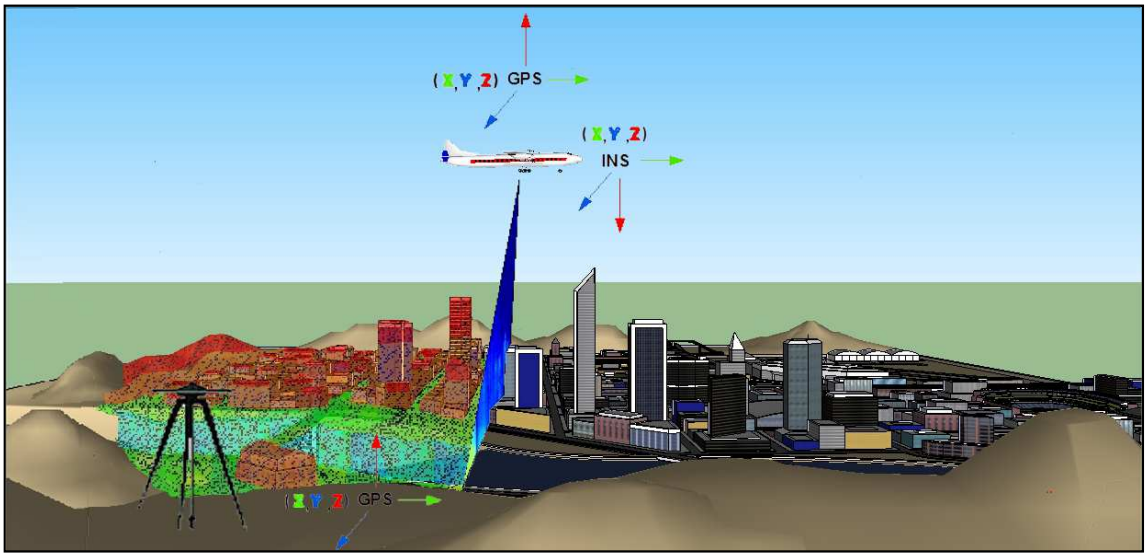


Figura 1. Metodología de medición láser.

Gracias a este sistema, es posible capturar nubes de puntos tridimensionales de alta densidad. La densidad de puntos es variable y se define como el número de puntos por metro cuadrado. La densidad depende de factores tan diversos como la altura de vuelo (Alharthy y Bethel 2004), la superficie sobre la que incide el haz láser, siendo mayor en

zonas con vegetación y nula en zonas de agua (Naus 2008) o la situación del punto dentro de la banda que define la pasada del avión, siendo en las zonas de solape entre pasadas consecutivas donde mayor densidad de puntos puede encontrarse (Moussa y El-Sheimy 2012). Asimismo, la densidad es un parámetro que varía dependiendo del programa que se utilice. Consecuentemente, debido a que la densidad de puntos depende de múltiples factores resulta un parámetro difícil de determinar. En este sentido, Balsa-Barreiro y Lerma (2012) proponen el cálculo de la densidad media como el mejor estimador. En este estudio, únicamente se consideran las bandas individuales, evitando analizar así conjuntos de puntos pertenecientes a varias pasadas del avión. Además, el trabajo pone de manifiesto que existen diferencias notables de densidad en función del ángulo de escaneado, alcanzando densidades mayores a medida que el ángulo aumenta. Como consecuencia, el estudio concluye que para obtener una estimación precisa de la densidad media debe de utilizarse la zona central asociada a las bandas o pasadas individuales, por lo que el 12,5% de la superficie más exterior de las mismas no debe ser utilizada en los cálculos.

Además de las coordenadas de cada punto, el sistema LiDAR es capaz de capturar información sobre el nivel de intensidad de la señal reflejada (Figura 2). La intensidad reflejada es diferente en función del tipo de superficie sobre la que incide el pulso láser y es utilizada frecuentemente en el proceso de clasificación de entidades (Elberink y Maas 2000; Song et al 2002; Hui et al. 2008; Shaker y El-Ashmawy 2012). En este sentido, el nivel de intensidad resulta efectivo para diferenciar entre objetos que absorben la energía del haz láser de forma distinta, tales como asfalto y vegetación o edificios y vegetación (Song et al. 2002).

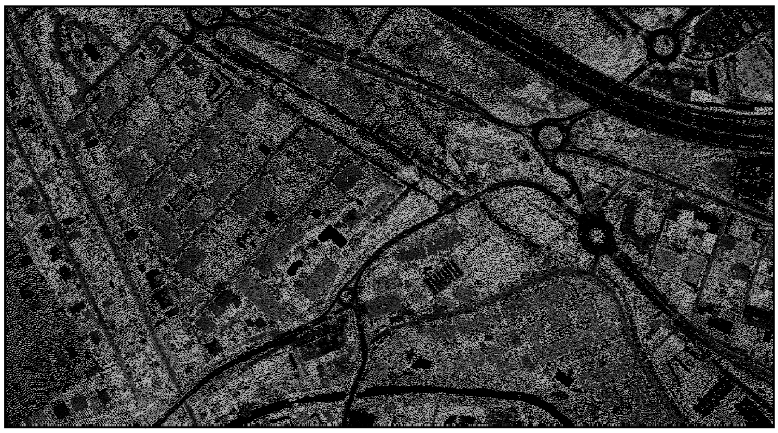


Figura 2. Nivel de intensidad visualizada en escala de grises. Datos test 2.

Otra de las características de mayor relevancia de los datos LiDAR es la capacidad de detectar pulsos con múltiples retornos. Cuando el pulso láser es reflejado por un objeto su comportamiento varía en función de la superficie de reflexión, así pues, si la señal incide sobre agua será mayoritariamente absorbida por completo y existirá poca reflexión, si incide sobre una superficie sólida será reflejada íntegramente como un único pulso y si la superficie no es completamente opaca o cubre diferentes objetos la señal podrá reflejar múltiples ecos. Esta capacidad multi-retorno del sistema láser cobra gran relevancia cuando se trata de detectar zonas de vegetación, ya que las ramas y hojas de los árboles constituyen una superficie no opaca muy propicia a reflejar múltiples retornos de la señal. En estos casos, en primer lugar el haz es reflejado por la copa de los árboles. Sin embargo, al tratarse de una superficie translúcida una parte del haz láser atraviesa las distintas capas de vegetación realizando múltiples reflexiones hasta llegar al terreno (Figura 3). El número de reflexiones es variable y depende del equipo láser utilizado, no obstante, como mínimo, los datos LíDAR suelen ofrecer información del primer y último pulso.

En la actualidad podemos encontrar sistemas que son capaces de capturar hasta cuatro o cinco retornos parciales antes de que la señal sea reflejada por el terreno. Además, nuevos sistemas están siendo desarrollados con el fin de procesar la información completa de la trayectoria de la señal (sistemas fullwaveform), consiguiendo así nuevas funcionalidades y aumentar considerablemente la información tridimensional asociada a cada pulso láser.

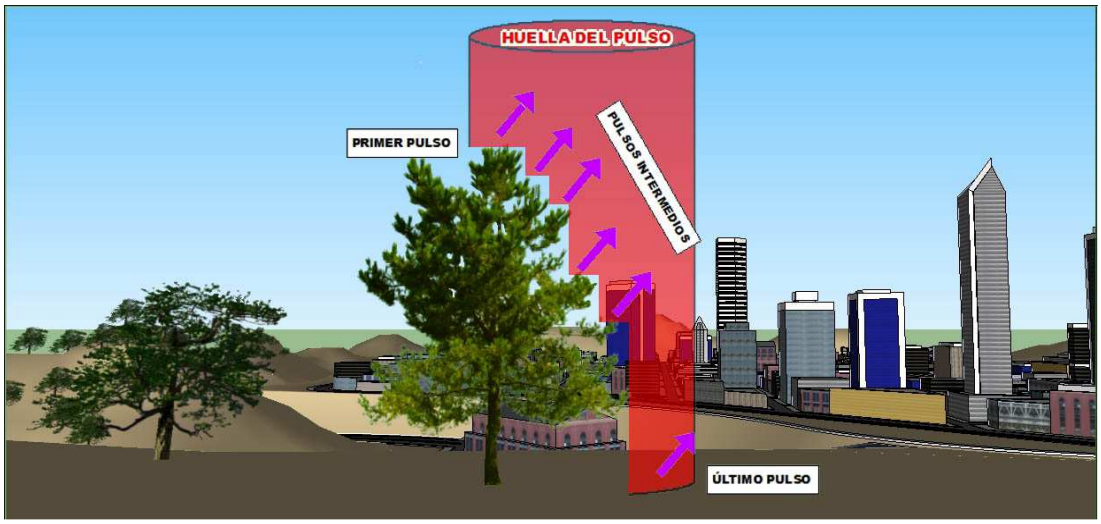


Figura 3. Multi-retorno de la señal láser sobre vegetación

Como se ha comentado con anterioridad, el uso de múltiples retornos de la señal láser es idóneo para detectar árboles y zonas de vegetación (Pfeifer et al. 2001; Alharthy y Bethel 2002; Tovari y Vögtle 2004; Meng, Wang y Currit 2009). Pfeifer et al. (2001) y Alharthy y Bethel (2002), utilizan información multi-retorno de la señal LiDAR para detectar vegetación penetrable. De manera similar, Tovari y Vögtle (2004) separan los edificios de la vegetación utilizando información multi-retorno. En este estudio, los edificios son detectados utilizando el primer retorno, posteriormente, tras filtrar los edificios del conjunto de datos la vegetación es identificada a partir de un estudio comparativo de la información asociada al primer y último retorno. El análisis estadístico de la diferencia de altura del entorno de cada uno de los puntos también puede ser útil para diferenciar la vegetación de los edificios (Alharthy y Bethel (2002), en este caso, se realiza un análisis de la variación estadística local de la diferencia de altura entre el primer y último pulso, considerando finalmente como edificios aquellos que presentan una baja variación (superficies lisas) y como vegetación aquellos donde la variación es alta (superficies rugosas).

Con respecto a los sistemas de onda-completa (fullwaveform), estos sistemas registran la totalidad de la forma del pulso láser recibido consiguiendo una señal resultante que presenta la suma de todas las reflexiones realizadas en las distintas superficies. De esta forma, se puede obtener información adicional como el tamaño de la huella del pulso LiDAR o la amplitud del pulso. Aunque su aplicación está todavía en desarrollo y bajo investigación, diversos estudios ponen de manifiesto su utilidad para clasificar las distintas coberturas (Chauve et al. 2007; Chehata et al. 2009, Wang et al. 2012).

I. 3. Elección del lenguaje de programación y breve descripción del mismo

I.3.1. Elección del lenguaje de programación

El primer paso en la implementación de algoritmos es la elección del lenguaje de programación, siendo esta fase de gran importancia en el desarrollo dl software. A la hora

de optar por un determinado lenguaje de programación debe de tenerse en cuenta el objetivo final del software a desarrollar y las ventajas que ofrece cada uno de los lenguajes al respecto. En el caso que nos ocupa, el objetivo es desarrollar algoritmos que actúen de forma eficiente sobre datos tridimensionales masivos dispuestos de forma aleatoria. Por tanto, el lenguaje a utilizar deberá ser capaz de operar con ficheros de gran tamaño con efectividad.

La primera elección que debe realizarse es si se opta por un lenguaje interpretado o compilado. Los lenguajes interpretados son aquellos donde no existe compilación, por lo que las diferentes proposiciones del programa fuente se traducen de forma secuencial y no se graban de forma permanente como código objeto, de tal forma que el código fuente se convierte en código máquina a medida que es necesario durante la ejecución. Uno de los lenguajes interpretados que tiene un uso ampliamente extendido en el ámbito de la ingeniería es el lenguaje de Matlab. Matlab es un software matemático muy versátil que ofrece un entorno de desarrollo integrado con lenguaje de programación propio, similar al lenguaje de programación C pero simplificado, donde no es necesaria la declaración de variables. La sencillez del lenguaje se ve reflejada en una optimización del rendimiento de programación. No obstante, al tratarse de un lenguaje interpretado, el código objeto no se graba en memoria para ser utilizado posteriormente y por tanto, cada vez que se utiliza una misma instrucción se debe interpretar de nuevo, como consecuencia, en procesamientos iterativos como bucles cada vez que se ejecuta el ciclo deberá realizarse la traducción, aumentando el tiempo de cómputo considerablemente con respecto a los lenguajes compilados.

Dado que el objetivo del presente trabajo es trabajar eficientemente con archivos de gran tamaño y hacer un gran número de operaciones con datos masivos 3D, el "lenguaje interpretado" no resulta la mejor opción, ya que aunque presenta la ventaja de una simplificación en el proceso de programación, la ejecución de las aplicaciones implementadas puede llegar a consumir un alto tiempo de computo e incluso resultar inviable.

Una vez descartada la programación en un lenguaje de programación interpretado, es necesario elegir entre los diversos lenguajes compilados existentes. En la actualidad, existen una gran cantidad de lenguajes de programación compilados que podrían ser válidos para el fin propuesto (Fortran, C, C++, Ada, Pascal, Algol...). La elección del lenguaje llegado a este punto, estará condicionada por las ventajas que ofrezca el lenguaje en cuanto a facilidad y rendimiento en el proceso de programación. Será necesario entonces, optar por un lenguaje de programación que disponga de gran variedad de librerías programadas, sobre todo librerías desarrolladas en el ámbito de la geometría computacional que puedan ser utilizadas para el desarrollo de los algoritmos.

Tras realizar una recopilación de las diferentes librerías geométricas de distribución libre que pueden ser utilizadas en el presente trabajo de investigación, puede concluirse que el lenguaje C++ es el que más posibilidades ofrece, ya que además de ser un lenguaje de programación compilado, dispone de una amplia gama de librerías libres asociadas a geometría computacional (CGAL, GEOMETRY, SUBPAK, OpenGL, GLUT,...). Estas librerías al ser de libre distribución pueden utilizarse en la implementación de software de investigación y constituyen un importante punto de partida para el desarrollo de algoritmos que actúan sobre entidades geométricas.

C++ es un lenguaje de programación que comenzó a desarrollarse en 1980, su autor fue B. Strouustrup. En un principio comenzó como una extensión de C que fue denominada C with classes. No obstante, pronto tuvo una gran difusión y éxito en el mundo de la programación. La ATT comenzó a estandarizarlo internamente en 1987, formándose 2 años después el comité ANSI y poco tiempo después el comité ISO. En la actualidad, C++ es un

lenguaje de alto nivel versátil y muy potente, de gran éxito entre los programadores profesionales.

A continuación se describen brevemente las librerías que serán utilizadas en el presente trabajo, así como aquellas que pueden ser útiles en futuros trabajos de desarrollo, a la hora de integrar los algoritmos desarrollados con otras aplicaciones, como puede ser software de visualización 3D.

- **CGAL (Computational Geometry Algorithm Library)**

CGAL es una librería de estructuras de datos y algoritmos geométricos robusta, eficiente y fácil de usar, en C++. La librería CGAL del proyecto de código abierto desarrollado a partir de la colaboración de diversas universidades y empresas europeas. En la actualidad, CGAL se ha convertido en un estándar universal utilizado tanto en el ámbito académico como en empresas privadas de desarrollo de software. Principalmente CGAL contiene:

a) Primitivas geométricas básicas, como puntos, vectores, rectas, predicados tales como posiciones relativas de puntos, y operaciones tales como intersecciones y cálculo de distancias.

b) Una colección de estructuras de datos y algoritmos estándar, tales como envolvente convexa, triangulación de Delaunay, poliedro, y estructuras de consulta multidimensional.

c) Interfaces con otros paquetes, en particular, para visualización, input/output o aritmética.

CGAL se distribuye bajo un esquema de doble licencia, de tal forma que se puede utilizar el código de forma gratuita en el desarrollo de software libre o comprar una licencia cuando su uso esté destinado al desarrollo de software comercial. En el caso que nos ocupa no será necesario utilizar la librería CGAL en el desarrollo de los algoritmos. N obstante, la posibilidad de aunar los algoritmos desarrollados en el presente trabajo con código de CGAL en futuras investigaciones, si ha sido un factor importante a tener en cuenta en la elección del lenguaje de programación.

- **Open GL (Open Graphics Library)**

Básicamente OpenGL es una especificación, es decir, un documento que describe un conjunto de funciones y el comportamiento exacto que deben tener. A partir de ella, se han creado e implementado distintas bibliotecas de funciones que actúan sobre entidades geométricas. Principalmente, OpenGL permite visualizar gráficos 3D en cualquier aplicación. Además dispone de diversas librerías adicionales como GULT, que contiene primitivas geométricas auxiliares y la posibilidad de construir aplicaciones de ventanas.

CGAL fue liberado en 2008 y puede utilizarse sin ningún problema en software libre y aunque no va ha utilizarse en el desarrollo de los algoritmos en el presente trabajo, si es un aspecto determinante a tener en cuenta, ya que en Open GL puede servir de motor de visualización 3D en futuros trabajos donde se pretenda integrar los algoritmos desarrollados en la presente tesis con un entorno de visualización de datos.

- **Funciones publicadas por John Burkardt**

John Burkardt es un profesor e investigador que ha trabajado en diferentes universidades de Norte América, sus investigaciones se centran en el campo de la matemática avanzada y

geometría computacional. En su página web pueden descargarse de forma gratuita una gran cantidad de librerías matemáticas y de geometría computacional, además de multitud de articulos, presentaciones y ponencias que ha realizado a lo largo de su carrera profesional (Burkardt s.f). Las librerías de libre distribución son muy diversas, y están programadas en diferentes lenguajes de programación entre los que se encuentran FORTRAN, C, C++, JAVA, PASCAL, PHYTHON scripts, PERL scripts, MATLAB y MATHEMATICA.

Las librerías que cobran especial relevancia para el presente trabajo son aquellas que se centran en el cálculo de la triangulación de Delaunay, ya que serán utilizadas como punto de partida en la codificación de datos. En concreto, algunas librerías y funciones desarrolladas por John Burkardt se utilizarán como base para el desarrollo del sistema de codificación así como en la definición del vecindario. Las librerías que contienen funciones útiles en dicho aspecto son las siguientes:

GEOMETRY

Es una librería de C++ con rutinas para cálculos geométricos en 2, 3 y n dimensiones. Estos cálculos incluyen ángulos, áreas de contención, distancias, intersecciones, longitudes y cálculo de volúmenes.

En esta librería también se contemplan diversas formas de describir objetos. Por ejemplo, la función que define una recta puede ser explicita, implícita o una representación paramétrica. Los nombres de las funciones o rutinas suelen especificar la representación utilizada, y existen funciones para convertir de una representación a otra.

Otras funciones útiles estarán relacionadas con el trazado de objetos geométricos, por ejemplo, la determinación de los vértices de un octaedro o definir puntos equidistantes en el interior de un círculo, elipse, esfera o triángulo.

GEOMPACK

Es una librería de rutinas programadas en C++ en doble precisión aritmética, que llevan asociados cálculos geométricos, entre ellos la triangulación de Delaunay y el cálculo del diagrama de Voronoi para una serie de puntos en el plano.

Las funciones de esta librería serán utilizadas para generar la triangulación de Delaunay en el presente trabajo.

TABLE DELAUNAY

No se trata de una librería en sí, simplemente se trata de código que utiliza funciones de la librería GEOMPACK para crear una triangulación de Delaunay, la ventaja del código es que concatena todas las funciones necesarias para generar una triangulación de Delaunay mediante un algoritmo de construcción incremental. En la presente investigación se utilizará el código y sus funciones asociadas para triangular los datos LiDAR.

I.3.2. Elección del compilador

Dado que finalmente se opta por un lenguaje de programación compilado (C++), será necesario elegir un compilador. Un compilador es un programa que es capaz de traducir el código fuente de una aplicación escrita en lenguaje de alto nivel, a otro lenguaje de nivel inferior (normalmente lenguaje máquina). Con esto, se consigue que la programación se realice en un lenguaje más cercano a la forma de pensar del ser humano, con el fin de compilarlo posteriormente a un lenguaje más manejable por la computadora.

Existen varias categorías de compiladores:

Compiladores cruzados: generan código para un sistema distinto del que están funcionando.
Compiladores optimizadores: realizan cambios en el código para mejorar su eficiencia, pero manteniendo la funcionalidad del programa original.
Compiladores de una sola pasada: generan el código máquina a partir de una única lectura del código fuente.
Compiladores de varias pasadas: necesitan leer el código fuente varias veces antes de poder producir el código máquina.
Compiladores JIT (Just In Time): forman parte de un interprete y compilan partes del código según se necesitan.

Hay una gran cantidad de compiladores de C++, una gran parte de ellos gratuitos (Dgwin, Mingw, ev-C 4.01, LCC-Win32, Borland C++ Builder,...). De entre todos los compiladores se ha optado por Visual C++ 2010.

Visual C++ es el nombre de una herramienta (IDE) producto de los lenguajes manejados por Microsoft C, C++ y C++/CLI. Está diseñado para el desarrollo y depuración de código escrito para las API's de Microsoft Windows, DiretX y la tecnología Microsoft. Net.

Visual C++ 2010 Express Edition, es un compilador de nivel profesional, que posee un entorno integrado de desarrollo que permite la programación orientada a objetos. Al tratarse de un entorno integrado incluye las siguientes herramientas de desarrollo:

- Editor de texto.
- Compilador/Enlazador
- Depurador.
- Visor de datos y dependencias (Browser).

Visual C++ 2010 permite diferentes modos de programación en función del objetivo del proyecto, en este sentido, el objetivo del trabajo es la implementación de algoritmos eficientes, por tanto, la interfaz de usuario quedará relevada a un segundo plano y se optará por una programación en modo consola que facilite la simplicidad a la hora de generar aplicaciones ejecutables.

I. 4. Descripción de los datos test

Para testear los diferentes algoritmos implementados se dispone de dos ficheros que contienen datos LiDAR de entornos urbanos. Cada conjunto de datos pertenece a un municipio diferente y define un entramado urbano con características propias, lo cual será ideal para probar los algoritmos en zonas urbanas con diferentes características (pendiente del terreno, masa de vegetación, altura de edificios, etc.) y comprobar así la capacidad de adaptación de los mismos. A continuación se describen ambas zonas test.

- **Datos test 1: Vall d'Uixó**

Los datos Lidar pertenecen al municipio de la Vall d'Uixó situado en la Comunidad Valencia, concretamente a la zona norte del casco urbano. Los datos test se proporcionan en un archivo ASCII que presenta los diferentes datos tabulados en columnas y abarcan una extensión de 221.070 m^2. Además, también se dispone de la ortofoto de la zona de estudio, así como de cartografía catastral, lo cual será muy útil para evaluar la precisión de los algoritmos. Las características de los datos LiDAR, ortofoto y cartografía catastral pueden observarse en la siguiente tabla.

Ortofoto	Resolución Fecha de adquisición	0,5 m 2005
Cartografía catastral	Fecha de adquisición	2002
LiDAR	Densidad Fecha de adquisición Número total de puntos LiDAR	1,83 pts/ m^2 2004 404.000 puntos

Tabla 1. Característica de los datos LiDAR y datos auxiliares. Datos test 1.

Tanto los datos LiDAR como los datos auxiliares (ortofoto y cartografía catastral) están disponibles en el sistema de referencia ED50, sistema de coordenadas UTM huso 30. El archivo de texto contiene la información tridimensional asociada a los puntos LiDAR así como la intensidad de la señal reflejada.

Los datos LiDAR describen un área urbana compuesta principalmente por edificios de viviendas de varias plantas, calles, vegetación aislada y otros objetos de menor tamaño como automóviles. En la zona de estudio, aproximadamente el 3,6% de la superficie se corresponde con vegetación, el 49,3% con edificaciones y el 47,1% con calles, carreteras y otro tipo de terreno. Cabe destacar que la zona tiene pendientes considerablemente altas para tratarse de un entorno urbano, con una pendiente media del terreno de 8,5% que llega a alcanzar en algunos casos hasta el 30% (Figura 4).

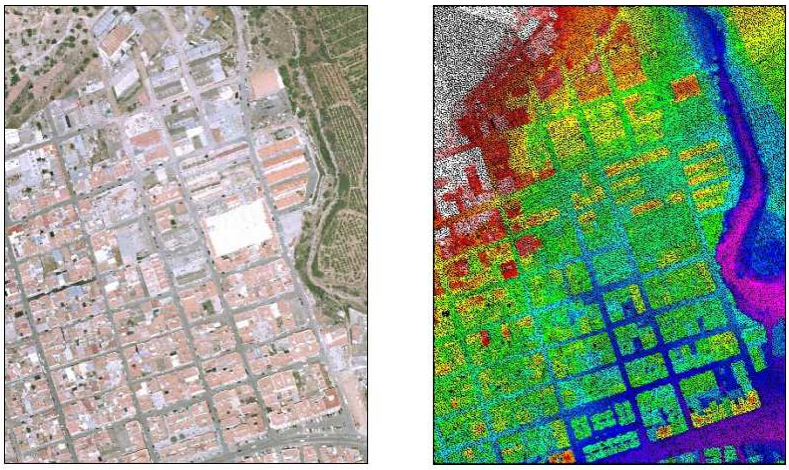


Figura 4. Ortofoto (a) y datos LiDAR coloreados en función de la altura (b).

Los datos test 1, tienen la ventaja de que comprenden una extensión de terreno relativamente pequeña (en torno a cinco veces menor que los datos test 2) y por tanto son ideales para realizar las primeras pruebas de los algoritmos donde se precisa que el tiempo de cómputo no sea excesivo. Además, la pendiente de las calles son considerables y presentan una alta variabilidad, aspecto que será esencial a la hora de determinar el grado de efectividad y adaptación del algoritmo de detección del terreno. Como contrapartida, no se trata de una zona urbana de características muy complejas, ya que los edificios son muy similares entre sí y nos encontramos con escasa presencia de árboles y vegetación. Por tanto, para testear los algoritmos que separan la clase edificio de la clase vegetación será más idóneo utilizar los datos test 2, donde existe una gran presencia de vegetación con características muy variables.

- **Datos test 2: Collado Villalba**

En este caso, los datos LiDAR pertenecen a la Comunidad de Madrid, concretamente al municipio de Collado de Villalba. La nube de puntos se proporciona en formato LAS 1.0 y abarca una extensión de 1.010.025 m^2. Además, también dispondremos de la ortofoto de la zona de estudio, lo cual será muy útil para evaluar la precisión de los algoritmos. Las características de los datos LiDAR y de la ortofoto pueden observare en la siguiente tabla.

Ortofoto	Resolución Fecha de adquisición	12,5 cm 2006
LiDAR	Densidad Fecha de adquisición Número total de puntos LiDAR	2.8 pts/ m^2 2007 2.841.938 puntos

Tabla 2.Característica de los datos LiDAR y datos auxiliares. Datos test 2.

Los datos LiDAR y la ortofoto están disponibles en el sistema de referencia ETRS89, sistema de coordenadas UTM huso 30. En cuanto a la información multi-retorno, el fichero contiene hasta 3 pulsos de retorno para una misma señal, y además, está disponible la intensidad de la señal reflejada.

Los datos test describen un área urbana de alta complejidad que incluye edificios residenciales, altos edificios de viviendas, edificaciones industriales, zonas de densa vegetación, árboles aislados, el cauce de un barranco, etc. En la zona de estudio, aproximadamente el 18% de la superficie se corresponde con vegetación, el 29% con edificaciones y el 53% con calles, carreteras y otro tipo de terreno. Cabe destacar que la zona considerada es relativamente plana, con una pendiente media del 2,3% (Figura 5).

Figura 5. Ortofoto (a) y datos LiDAR coloreados en función de la altura (b).

La principal ventaja de los datos test 2, es que se trata de una zona urbana de gran complejidad donde podemos encontrar edificios de tamaño, altura y forma muy varible (Figura 6). Además, la presencia de zonas de vegetación es inusualmente alta para un entorno urbano, encontrando en muchos casos árboles pegados a edificios, siendo ésto muy útil a la hora de testear los algoritmos cuyo objetivo es separar la vegetación de los edificios.

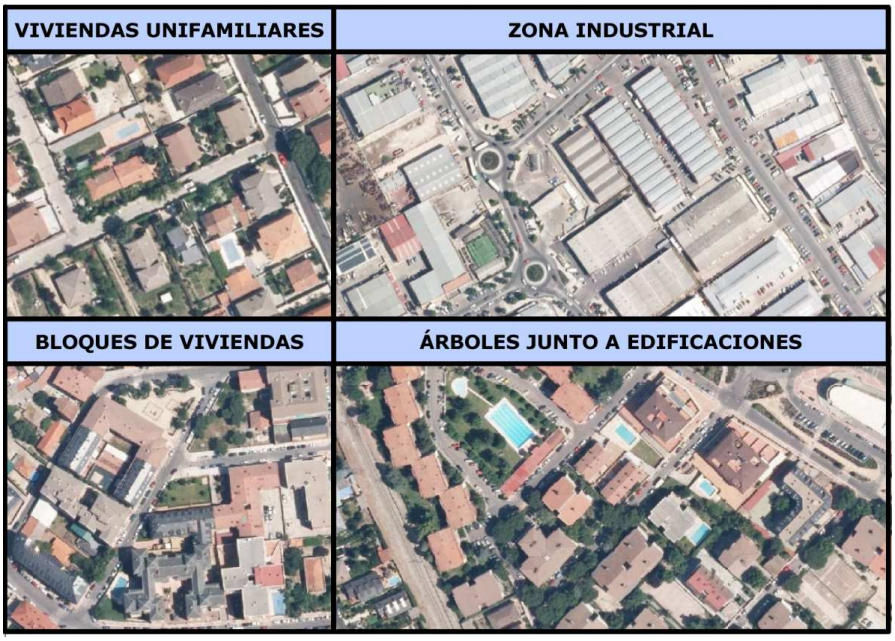


Figura 6. Características de la zona test 2. Collado Villalba.

II. ANTECEDENTES A LA PROGRAMACIÓN DESARROLLADA

II. 1. Disposición y organización de los datos LiDAR

Como norma general, los datos brutos LiDAR suelen presentarse en un archivo de texto ASCII tabulado en columnas o bien en un archivo binario formato LAS.

En los archivos de texto (ASCII), la información asociada a cada retorno de pulso LiDAR como son las coordenadas (XYZ), el nivel de intensidad, así como diferente información adicional (ángulo de giro del espejo oscilante, tiempo GPS correspondiente a la captura de cada pulso, dirección de vuelo,...) aparecen tabulados en diferentes columnas (Figura 7). Este formato tiene la ventaja de que los datos pueden visualizarse y editarse de forma sencilla mediante cualquier editor de texto. No obstante, tiene el inconveniente de que se tratan de ficheros de gran tamaño y por tanto poco manejables. Además no existe un estándar con respecto a la distribución de los datos que almacena y es posible encontrar la información dispuesta de forma diferente en función del fichero a tratar. Siendo este último aspecto un gran inconveniente en la utilización e intercambio de ficheros entre distintos usuarios y software ("LASer (LAS) File Format Exchange Activities", 2013).

NUM_PTO	TIEMPO	XP1	YP1	ZP1	XP2	YP2	ZP2	N1	N2
1.000	127913.156	737072.130	4412609.400	145.130	737072.120	4412609.390	145.180	25.000	25.000
2.000	127913.156	737073.240	4412610.170	145.040	737073.240	4412610.160	145.060	24.000	24.000
3.000	127913.156	737073.630	4412610.150	148.710	737073.630	4412610.160	148.680	10.000	10.000
4.000	127913.157	737073.510	4412610.210	148.220	737073.510	4412610.210	148.190	4.000	4.000
.....	.....	.....	.....	.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	.....	.....	.....	.....	.....	.....
1122.0	127913.158	737071.880	4412609.330	145.160	737071.870	4412609.330	145.160	9.000	9.000

Figura 7. Formato de los datos LIDAR brutos en ficheros ASCII.

Con el fin de solucionar los inconvenientes que presenta la utilización de ficheros ASCII para almacenar datos LiDAR, el formato binario desarrollado por la American Society for Photogrammetry & Remote Sensing se ha convertido en los últimos años en el formato estándar aceptado por la mayoría de empresas para almacenar la totalidad de atributos asociados a los datos LiDAR. Este formato es conocido como " LAS (LASer (LAS) File Format Exchange Activitie) y se trata de un archivo público de intercambio de datos láser que permite registrar grandes nubes de puntos 3D manteniendo la información específica de la naturaleza de los datos LiDAR originales. Además, al tratarse de un formato abierto, cualquier software, bien sea libre o comercial puede hacer uso de él y aprovechar las ventajas que proporciona.

El formato LAS inició su recorrido con la versión 1.0 desarrollada en mayo de 2003. Desde entonces, cuatro versiones han sido desarrolladas, siendo la versión 1.3 la última disponible

desde el 24 de octubre de 2010. El formato LAS tiene un archivo de cabecera que contiene una serie de datos genéricos tales como el número de puntos, área de estudio, datos de proyección, etc. Dicha cabecera viene seguida del registro de los datos asociados a cada pulso láser (Figura 8).

ARCHIVO DE CABECERA

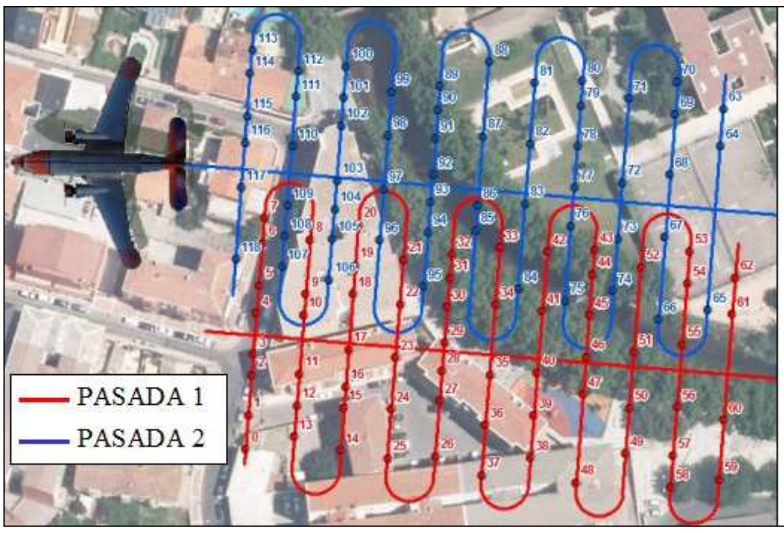
```
LAS Header
----------
GUID:                       00000000-0000-0000-0000
LAS Version:                1.0
System ID:                  <empty>
Generating Software:        TerraScan
Flight Date Julian:         0
Year:                       0

Header Size:                227
Point Data Offset:          229
VLR Count:                  0

Point Data Format:          Format 1
Point Data Record Length:   28
Number of Point Records:    494012
Points By Return
                Return 1:   0
                Return 2:   0
                Return 3:   0
                Return 4:   0
                Return 5:   0

X,Y,Z Scale Factors:         0.0100, 0.0100, 0.0100
X,Y,Z Offsets:               0.00, 0.00, 0.00
Min, Max X:                  508000.10, 509000.10
Min, Max Y:                  4066000.11, 4067000.07
Min, Max Z:                  -1.47, 40.56
```

REGISTRO DE INFORMACIÓN DE LOS PULSOS LIDAR

Item	Format	Size
X	long	4 bytes
Y	long	4 bytes
Z	long	4 bytes
Intensity	unsigned short	2 bytes
Return Number	3 bits (bits 0, 1, 2)	3 bits
Number of Returns (given pulse)	3 bits (bits 3, 4, 5)	3 bits
Scan Direction Flag	1 bit (bit 6)	1 bit
Edge of Flight Line	1 bit (bit 7)	1 bit
Classification	unsigned char	1 byte
Scan Angle Rank (-90 to +90) – Left side	char	1 byte
User Data	unsigned char	1 byte
Point Source ID	unsigned short	2 bytes
GPS Time	Double	8 bytes

Figura 8. Archivo de cabecera e información de registro. Formato LAS 3.1 (V1)

Independientemente del formato utilizado, el principal problema asociado al almacenamiento, consiste en que los puntos están dispuestos en el archivo en el orden en que fueron capturados por el sensor. Debido a que existen solapes entre las diferentes pasadas del avión, no existirá relación alguna entre la posición de un punto en el terreno y su correspondiente registro en el archivo de almacenamiento (Figura 9).




```
1.000  127913.156  737072.130
2.000  127913.156  737073.240
3.000  127913.156  737073.630
4.000  127913.157  737073.510
5.000  127913.157  737073.170
6.000  127913.158  737071.880
7.000  127913.191  737069.530
.....  ..........  ..........
.....  ..........  ..........
.....  ..........  ..........
.....  ..........  ..........
.....  ..........  ..........
.....  ..........  ..........

403990 129487.514 736925.310
403990 129487.548 736922.020
404000 129487.549 736921.610
```

Figura 9. Registro de puntos pertenecientes a distintas pasadas del avión.

Tomando como ejemplo la captura de datos LiDAR que muestra la figura anterior, un punto perteneciente a la primera pasada puede tener como punto más cercano un punto capturado en la segunda. Ésto ocurre por ejemplo para el punto 5, cuyo punto más cercano

es el punto 107. Así pues, dentro del fichero de almacenamiento no existirá forma posible de saber las relaciones de proximidad entre los distintos puntos capturados.

Como se ha comentado en la introducción, existen multitud de aplicaciones de los datos LiDAR, tales como la generación de modelos digitales del terreno (MDT) o la detección de entidades que conforman la escena con fines cartográficos (actualización de cartografía, detección de cambios, generación de modelos virtuales de construcciones, simulación de inundaciones,...). En todas ellas, es necesario determinar en post-proceso relaciones de proximidad entre puntos con el fin de aplicar filtros u otro tipo de algoritmos en el procesado de datos. Por tanto, teniendo en cuenta que la nube de puntos original registrada no proporciona información explícita sobre la distribución de los puntos, será necesario organizar los datos láser distribuidos de forma aleatoria en estructuras que permitan establecer relaciones de proximidad y faciliten la búsqueda del vecindario más próximo a un punto dado de forma eficaz en las operaciones de procesamiento (Lari et al. 2011).

La determinación del vecindario local de cada punto LiDAR es una tarea importante en cualquier proceso de segmentación, clasificación y procesado de datos 3D. Con el fin de facilitar esta tarea el enfoque más común consiste en interpolar los datos LiDAR originales en una rejilla o malla regular (GRID) para acelerar el procesamiento y facilitar cualquier tipo de análisis espacial (Alharthy y Bethel 2004; Arefi y Hahn 2005; Zhang y Whitman 2005; Ekhtari et al. 2008; Meng, Wang y Currit 2009; Matikainen et al. 2010; Kim y Shan 2011; Trinder y Salah 2012). Como norma general, el método la transformación de datos láser brutos dispuestos de forma de imagen o malla regular se realiza mediante una interpolación por el método del vecino más próximo (Arefi y Hahn 2005; Meng, Wang y Currit 2009), método con el que se consigue minimizar la pérdida de información asociada al proceso de generalización producido por la interpolación. Variaciones de este método han sido llevadas a cabo por otros autores. Elmqvist et al. (2001) genera una malla regular mediante la selección del punto de menor cota contenida en cada una de las celdas. De forma similar, otros autores (Kim y Shan 2011; Zhang y Whitman 2005) transforman los datos LiDAR brutos a malla regular asignando la altitud del punto LiDAR que contiene cada celda al centro de la misma. En caso de que exista más de un punto láser en una celda dada, la cota de la celda se obtiene tomando el punto de menor altitud. Por otro lado, si una celda no contiene ningún punto láser original, su valor se obtiene a partir de la cota del punto más cercano existente en los datos LiDAR brutos. Otro enfoque consiste en determinar en una primera fase el tamaño de las celdas que debe de tener la malla regular resultante mediante el cálculo del espaciado medio entre los puntos LiDAR originales dispuestos de forma aleatoria (Li 2013). En este estudio, para las celdas que contienen más de un punto láser se opta por una interpolación basada en el gradiante morfológico con un elemento de estructuración simétrica, con el fin de superar la influencia en la distribución irregular de la nube de puntos LiDAR. En este sentido, existen distintas variaciones del método anterior en los casos en los que una celda contiene más de un punto LiDAR. Los principales operaciones que se suelen realizar en estos casos son: la media de la altura de los puntos que contiene la celda, adjudicar la altitud del punto más bajo o tomar la cota del punto más cercano al centro de la celda (Pfeifer et al. 2001).

A pesar de que la transformación de los datos láser originales a malla regular es una práctica muy extendida, el proceso de interpolación asociado puede causar una pérdida significativa de información (Roggero 2001), sobre todo, en el cálculo de la altura de celdas situadas entre puntos de tierra y otras entidades como edificios o árboles, donde la cota final puede contener errores graves (Vosselman, 2000). Con el fin de solventar este problema, muchos investigadores optan por estructurar los datos mediante una triangulación (Vosselman, 2000; Vosselman y Dijkman 2001; Filin 2002; Akel, et al. 2003; Hofmann 2004; Rabbani et al. 2006; Pérez et al. 2012; Sánchez y Lerma, 2012). La triangulación, también conocida como red irregular de triángulos (TIN), establece conectividad entre los puntos más próximos, consiguiendo así establecer relaciones topológicas y facilitando la definición de vecindad. De entre los distintos tipos de

triangulación, la más utilizada es la triangulación de Delaunay (Vosselman y Dijkman 2001; Filin 2002; Hofmann 2004; Sánchez y Lerma 2012). La triangulación de Delaunay tiene la propiedad de conectar mediante aristas los puntos más próximos entre sí, consiguiendo reducir al máximo las operaciones asociadas a la determinación de los puntos más próximos a uno dado.

La triangulación de Delaunay no solamente es utilizada para establecer la conectividad entre puntos dispuestos de forma aleatoria, sino que además es ampliamente utilizada en el ámbito de visión por computador y generación de gráficos 3D con el fin de modelar objetos tridimensionales. En general se utilizan triángulos que ya son los polígonos más simples y tienen muchas propiedades favorables, como que representan una superficie coplanaria.

Dada una nube de puntos en el plano, la triangulación de Delaunay consiste en hallar la triangulación en la que los puntos más próximos entre sí están conectados por aristas, o dicho de otra forma, en la que los triángulos resultantes sean lo más regulares posibles. La triangulación de Delaunay es única y debe de cumplir dos propiedades fundamentales:

- **Propiedad 1**

Si tres puntos P1, P2 y P3 pertenecen a una triangulación de Delaunay "T", son vértices de una misma cara de la triangulación T, sí y solamente sí, el círculo que pasa por los puntos P1, P2 y P3 no contiene ningún otro punto de T en su interior.

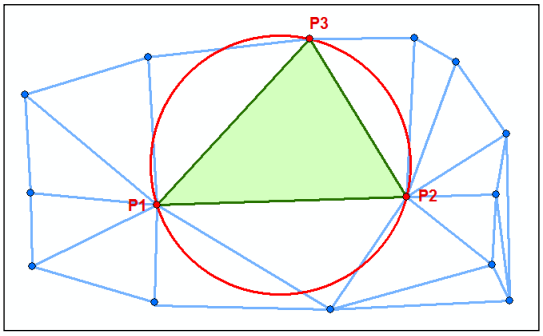


Figura 10. Propiedad 1. Triangulación de Delaunay.

- **Propiedad 2**

Dados dos puntos P1 y P2 pertenecientes a una triangulación de Delaunay "T". Dichos puntos pertenecen a un mismo lado de un triángulo de T, sí y solamente sí, existe un círculo que pasa por P1 y P2 que no contiene en su interior a ningún otro punto de T.

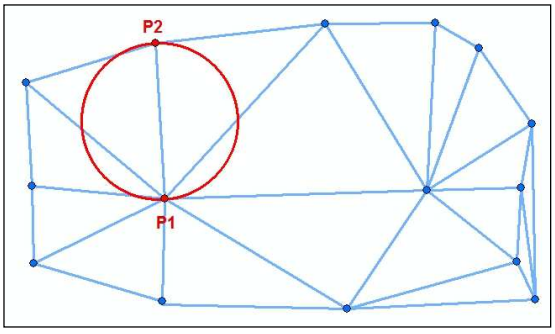


Figura 11. Propiedad 2. Triangulación de Delaunay.

Estas dos propiedades de la triangulación de Delaunay determinan que aquellos puntos que están conectados mediante aristas a un punto dado sean los más cercanos al mismo. De esta forma podrá obtenerse de forma directa el vecindario más próximo a un punto sin realizar búsqueda alguna: únicamente bastará con seleccionar los puntos contenidos en las aristas de la triangulación asociadas al punto base objeto de análisis.

A pesar de las grandes ventajas que ofrece la triangulación de Delaunay para estructurar datos LiDAR brutos, algunos autores describen varios inconvenientes que deben considerarse. Filin y Pfeifer (2006) indican que la estructuración de datos LiDAR mediante triangulación no tiene en cuenta factores como la densidad de puntos, y por tanto, si el propósito es segmentar los datos, el ruido aleatorio constituye un alto nivel de incertidumbre en el cálculo de vectores normales a las distintas superficies. Por otro lado, Filin y Pfeifer (2005) indican que a pesar de que la triangulación de Delaunay conecta mediante aristas los puntos más próximos entres sí, dichos puntos no siempre pertenecen a una misma superficie o a un mismo objeto, ya que puntos de tierra pueden estar conectados con puntos de vegetación o de edificio, constituyendo esta característica un problema a la hora de aplicar algunos algoritmos de segmentación. En consecuencia, algunos autores utilizan otros métodos para estructurar datos LiDAR brutos dispuestos de forma aleatoria. Estos métodos se basan en dividir el espacio tridimensional en subespacios de menor tamaño estructurando así los datos en forma de árbol. Esta disposición de los datos tiene la ventaja de que conserva la posición y el resto de información de los datos originales y permite obtener relaciones de proximidad de forma eficiente. Uno de los métodos más simples que dividen el espacio, es el descrito por Filin y Pfeifer (2005) y Sánchez (2006). Este método consiste en dividir el espacio en celdas de un tamaño determinado y asociar la información original de cada pulso LiDAR con la celda donde está contenida la proyección 2D de dicho punto. Sánchez (2006) subdivide el espacio en forma de malla regular con celdas cuadradas de 2,5 m de lado y altura infinita. Posteriormente los puntos LiDAR son codificados y almacenados en función del cuadrante o celda a la que pertenecen en una matriz tridimensional. Mediante esta metodología, es posible establecer relaciones de proximidad con un bajo coste computacional y mantener la información original de los datos LiDAR. Wang y Yi (2010) utilizan la estructura *octree* para dividir el espacio de forma recursiva en ocho octantes, constituyendo así una estructura en forma de árbol donde las conexiones entre los puntos LiDAR vienen determinadas por la relación entre los nodos de la propia estructura. En este estudio, tras disponer los datos LiDAR en una estructura *octree*, los datos son segmentados obteniendo los diferentes planos que constituyen la escena con buenos resultados. Por último, otros autores (Lari et al. 2011; Kwak et al. 2012; Lari y Habib 2012) utilizan la estructura kd-tree para organizar los datos LiDAR. Esta estructura dispone los puntos en un espacio euclídeo de k dimensiones, la división del espacio se produce mediante el corte de planos perpendiculares a los ejes de coordenadas. La principal ventaja de una estructura de este tipo es que optimiza el tiempo de cómputo en la búsqueda de puntos próximos.

Los distintos métodos comentados para organizar los datos mediante estructuras que dividen el espacio pueden observarse en la Figura 12.

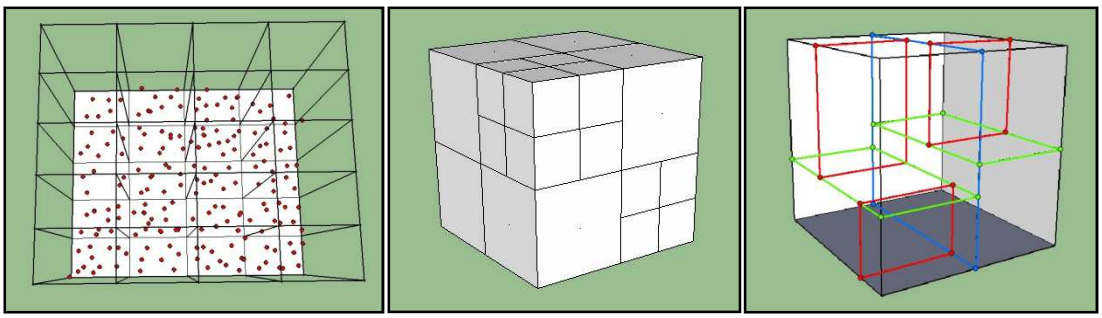


Figura 12. Estructura en matriz tridimensional (izquierda), octree (centro) y kd-tree (derecha).

II. 2. Definición de vecindad

La definición de vecindad es el primer paso en el tratamiento de datos láser 3D. Esta definición consiste en determinar los puntos más cercanos a un punto dado con el fin de aplicar distintos tipos de algoritmos de segmentación o clasificación. En cierto modo, el vecindario directo de un punto viene determinado por la estructura de organización de los datos que se haya aplicado con anterioridad. De esta forma, si los datos han sido interpolados a malla regular, el vecindario de una celda serán sus celdas colindantes. Por otro lado, si los datos han sido estructurados mediante una triangulación de Delaunay, los puntos más próximos a un punto dado serán aquellos que están conectados a dicho punto mediante aristas. No obstante, tal y como apuntan distintos investigadores (Lari et al. 2011; Filin y Pfeifer 2005), la utilización únicamente de los puntos más próximos a un punto dado como base para aplicar ciertos algoritmos de segmentación puede suponer un problema. El principal inconveniente de esta definición radica en que pueden ser incluidos en un mismo vecindario puntos que pertenecen a distintos objetos físicos, siendo ésto un inconveniente en algoritmos que segmentan datos LiDAR en los diferentes planos que conforman la escena, al producirse errores en el cálculo del vector normal utilizado para la segmentación.

Con el fin de aumentar la efectividad de los algoritmos que actúan sobre los datos LiDAR, diferentes definiciones de vecindario han sido propuestas en distintos trabajos (Lari y Schenk 2001 ; Lisen y Prautzsch 2001; Filin y Pfeier 2005; Filin y Pfeier 2006; Rabbani et al. 2006; Lari et al. 2011; Lari et al. 2012...). En cualquier caso, tal y como apunta Rabbani et al. (2006) existen dos definiciones de vecindario comúnmente utilizadas que ofrecen buenos resultados en aquellos casos en los que se pretende segmentar la escena en los diferentes planos que la componen mediante el cálculo del vector normal:

- **Cálculo del vecindario mediante los k vecinos más próximos**

Tal y como su propio nombre indica se trata de detectar el vecindario de un punto dado mediante la búsqueda de los k puntos más próximos al mismo. El método para calcular la distancia puede variar (distancia euclídea 2D o 3D, distancia Manhattan...). El número de puntos más próximo k es fijo, además, el método condiciona la zona del vecindario en función de la densidad de puntos del entorno, aumentando el tamaño en aquellas regiones donde la densidad de puntos es baja. Teniendo en cuenta que la densidad de puntos es un indicador de la medida del ruido en los datos LiDAR (debido a que la densidad de pulsos láser disminuye de forma inversamente proporcional a la distancia y el ángulo de incidencia) se puede concluir que el método consigue mejorar la estimación de los vectores normales a las superficies. Además, el procedimiento siempre utiliza un número de puntos fijos de vecindario con lo que nunca se dará el caso de un vecindario vacío como ocurre con otros métodos.

- **Cálculo del vecindario mediante distancia de búsqueda fija**

A diferencia del método anterior, en este caso el área asociada al vecindario es fija mientras que el número de puntos pertenecientes al mismo es variable. Para calcular el área del vecindario se suele utilizar la distancia euclídea y en este caso, el número de puntos del vecindario varía en función de la densidad de los datos LiDAR, de tal forma que una mayor densidad de puntos se traduce en un mayor número de puntos en el vecindario.

Para el cálculo del área del vecindario es usual utilizar la distancia euclídea entre los puntos (Lari et al. 2011). Esta definición de vecindad se denomina zona esférica, ya que se corresponde con una esfera de radio determinado cuyo centro coincide con el punto en cuestión (Lari y Schenk 2001), de tal forma, que todos los puntos que están dentro de la esfera se consideran como puntos vecinos (Figura 13b). Además de la distancia euclídea,

también es frecuente determinar el vecindario a partir de un cilindro (Filin y Pfeier 2005), donde destacan el método del cilindro vertical finito o el del cilindro inclinado finito. El método del cilindro vertical finito (Figura 13a) consiste en centrar un cilindro en el punto objeto de estudio del que queremos obtener su vecindario, de forma que el eje del cilindro pase por el punto de interés. El vecindario en este caso serán todos los puntos que están contenidos en el volumen del cilindro. Mediante este método, al igual que con la utilización de zona esférica, puntos pertenecientes a diferentes planos u objetos pueden pertenecer al mismo vecindario (Figura 13a). No obstante, este efecto se minimiza mediante la utilización del cilindro inclinado finito. Este método consiste en medir las distancias a lo largo del plano tangente a la zona local del punto de interés (Figura 13c). Resulta habitual aplicar esta metodología en dos pasos: en primer lugar se aplica la métrica del cilindro vertical finito o acotado para determinar un primer vecindario y seguidamente, se acepta como vecindario definitivo aquellos que están por debajo de una distancia umbral con respecto al plano tangente calculado a partir del vecindario provisional. Este tipo de vecindario minimiza el efecto de obtener puntos pertenecientes a diferentes objetos dentro del mismo vecindario (Figura 13c).

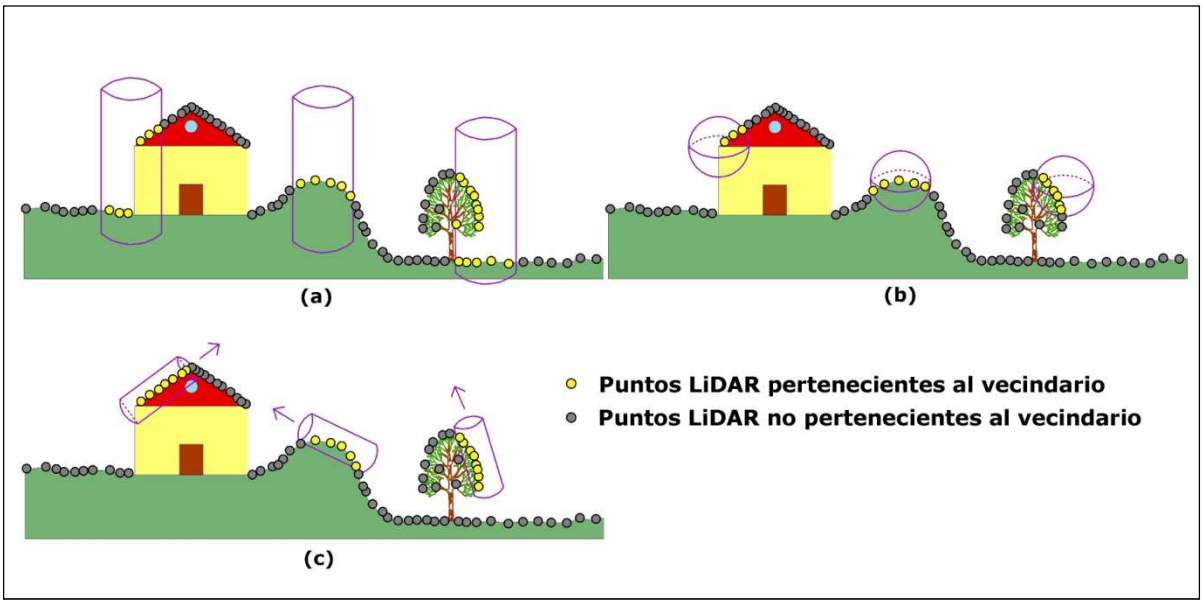


Figura 13. Vecindario cilindro vertical (a), zona esférica (b), cilindro inclinado finito (c).

Tanto el uso de la zona esférica como la utilización del cilindro vertical finito o acotado, no tienen en cuenta la forma física de los distintos objetos a la hora de determinar el área de vecindad, por lo que diferentes superficies pertenecientes a distintas entidades (terreno, árboles, edificios...) pueden incluirse en el mismo vecindario (Lari et al. 2011). En este sentido, con el propósito de segmentar la escena en los distintos planos que la conforman, varios trabajos ponen de manifiesto la necesidad de considerar otros atributos adicionales para definir el vecindario, tales como la irregular distribución de los puntos LiDAR, la variedad de objetos existentes en la nube de puntos o la densidad de los datos (Filin y Pfeier 2005; Lari et al. 2011). Algunos autores han propuesto algoritmos para modificar los métodos de triangulación y evitar así conexiones entre puntos que tienen grandes diferencias de altura pero que están muy próximos en cuanto a su distancia horizontal. Cabe destacar en este ámbito el trabajo realizado por Verbree y Oosterom (2001) donde se desarrolla un método para generar triangulaciones 3D de datos láser mediante la utilización de tetraedros en vez de triángulos. En este estudio, aquellos tetraedros que eran cortados por un haz láser (visual entre el punto de emisión láser y el punto de reflexión) eran eliminados, consiguiendo así eliminar las conexiones entre puntos con altitudes muy dispares. No obstante, el vecindario obtenido no tiene en cuenta la densidad de los puntos LiDAR en su definición.

Uno de los algoritmos que utiliza la triangulación para determinar el vecindario que sí tiene en cuenta la densidad de puntos es el propuesto por Lisen y Prautzsch (2001). En este trabajo proponen una triangulación local, donde cada punto está conectado a sus k vecinos más próximos. Para ello, tras localizar los k puntos más próximos se determina mediante mínimos cuadrados el plano que más se ajusta a la superficie definida por los mismos, posteriormente, la totalidad de los puntos son proyectados sobre dicho plano y los ángulos interiores entre los segmentos virtuales que conectan la proyección de cada punto del vecindario con el punto objeto de análisis son calculados. Si existe algún ángulo mayor a 90º se considera que el vecindario está mal definido ya que no recubre al punto en todas las direcciones, en cuyo caso nuevos puntos son incluidos al vecindario y el proceso es iterado hasta que se cumple la restricción angular (Figura 14). Tal y como muestra la Figura 14 , dado un punto pertenecientes al conjunto de los datos LiDAR cuyo vecindario quiere ser obtenido (Figura 14a), los puntos son proyectados sobre el plano que mejor se adapta a la forma local de la superficie contenida por dicho punto y los k puntos más próximos se detectan (en este caso k=5). Posteriormente, los ángulos interiores entre los segmentos virtuales que unen el punto en cuestión y los k puntos más próximos se calculan (Figura 14b), de forma que si como ocurre en el ejemplo alguno de los ángulos es mayor de 90º (α>90º), el vecindario aumenta y se itera el proceso hasta que la restricción angular se cumple en todos los casos (Figura 14c).

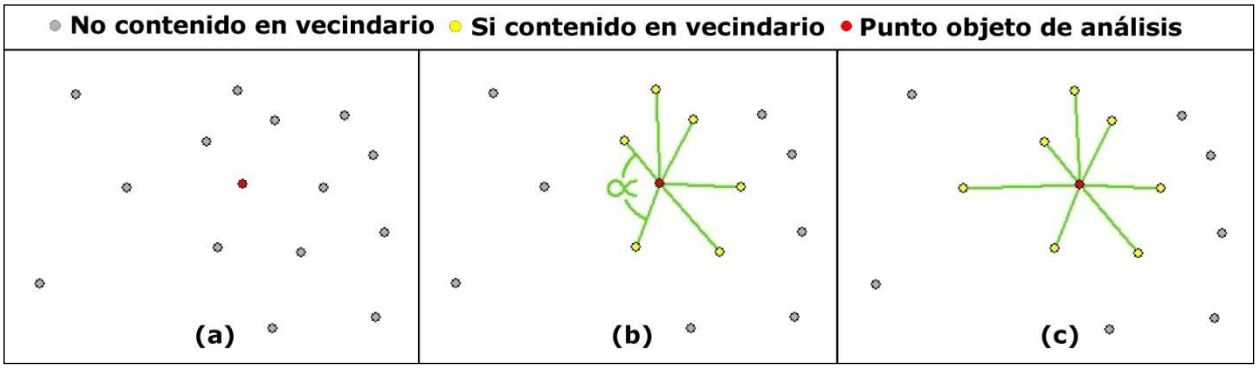


Figura 14. Proceso para determinar el vecindario mediante una triangulación local que tiene en cuenta la densidad de los datos LiDAR.

Lisen y Prautzsch (2001) también describe un método similar donde el vecindario de un punto LiDAR es detectado como puntos cercanos al mismo que están simétricamente distribuidos alrededor del punto objeto de análisis. Para ello se utiliza una región centrada en el punto que va aumentando de tamaño hasta que se seleccionan los k vecinos más próximos. La región utilizada para detectar el vecindario suele ser el volumen de un cilindro no finito o la zona esférica (distancia 3D).

Como se ha comentado con anterioridad, para evitar que un mismo vecindario contenga puntos pertenecientes a diferentes entidades, algunos autores calculan un vecindario adaptado a la pendiente local (Filin y Pfeier 2005; Filin y Pfeier 2006; Lari et al. 2011; Lari y Habib 2012). Filin y Pfeier (2005 y 2006) proponen un vecindario cilíndrico adaptativo a la pendiente. Básicamente el algoritmo consiste en determinar un primer vecindario provisional centrando un cilindro de radio y altura determinada en el punto objeto de análisis. Posteriormente el vecindario definitivo es aquel que se encuentra a una menor distancia umbral del plano tangente a la superficie local del punto. Para la elección de los parámetros como la altura del cilindro y su radio, diferentes aspectos deben de tenerse en consideración, tales como las características de los datos y del sistema (densidad de puntos, tamaño de la huella del pulso láser, efecto del ruido...). La altura del cilindro debe establecerse en función de la diferencia de altura mínima considerada para diferenciar un cambio brusco de altitud a la hora de distinguir superficies pertenecientes a diferentes

entidades u objetos. En cuanto a la distancia máxima a considerar entre los puntos del vecindario y el plano tangente a la superficie local, factores como el nivel de ruido esperado y las características de la superficie considerada deben de ser tenidos en cuenta. Lari et al. (2011) así como Lari y Habib (2012), utilizan un método similar al anterior, con la diferencia de que la densidad local de los puntos LiDAR es tenida en consideración para determinar el vecindario. Para ello, en una primera fase se detecta el vecindario de un punto dado mediante la distancia esférica y los k puntos más próximos al punto objeto de estudio. Posteriormente, la densidad de puntos de dicho vecindario se calcula y el radio del cilindro de adaptación se ajusta en función de dicha densidad, en este caso, el plano tangente a la superficie local del punto es obtenido mediante un ajuste iterativo por mínimos cuadrados. Finalmente, el vecindario definitivo se obtiene mediante la detección de los puntos cuyas distancias normales al plano son menores que la mitad de la altura del cilindro. Además, esta metodología puede ser utilizada para realizar una clasificación de los puntos en función de si pertenecen a una superficie plana o rugosa (Lari y Habib 2012), de tal forma que el punto objeto de análisis se puede considerar parte de una superficie plana, si la mayoría de los puntos de su vecindario esférico pertenecen a la zona cilíndrica de adaptación. Por el contrario, se considera que el punto en cuestión pertenece a una superficie rugosa en caso contrario. Siendo este proceso de utilidad en la diferenciación de vegetación y otros objetos como edificios.

Además del cálculo de un cilindro adaptativo a la pendiente para determinar el vecindario, otros algoritmos han sido utilizados en los últimos años con buenos resultados, en este sentido cabe mencionar aquellas metodologías que detectan de forma automática los bordes de las distintas superficies que conforman la escena y utilizan dichos bordes como zonas límite en el cálculo del vecindario, es el caso de Pauly et al. (2003) que determinan el tamaño del vecindario local mediante la búsqueda de saltos en la variación de la superficie a partir de un proceso iterativo donde el vecindario aumenta de tamaño en función del análisis de la matriz de covarianza asociada al vecindario considerado en cada iteración. De forma similar otros autores utilizan el cálculo de los valores propios de la matriz de covarianza del vecindario de un punto para obtener las propiedades geométricas y la estructura del vecindario local (Carlberg et al. 2009; Demantke et al. 2011; Niemeyer et al. 2012; Ural y Shan 2012).

Carlberg et al. (2009) clasifican las distintas entidades de la escena como planas o rugosas con el fin de diferenciar distintos objetos como árboles y edificios en zonas urbanas. Para ello utilizan un algoritmo de segmentación por crecimiento de regiones, donde el vecindario de cada punto objeto de estudio es determinado de forma automática mediante el estudio de la distribución de los puntos próximos aplicando análisis de componentes principales en la matriz de covarianza del vecindario local. Niemeyer et al. (2012) hacen uso también del cálculo de los valores propios de la matriz covarianza del vecindario, con el fin de determinar distintas características tales como esfericidad o planicidad y utilizarlas para clasificar los distintos objetos mediante un algoritmo que combina árboles de decisión ('*random forest*'). Demantke et al. (2011) presentan un método multiescala que calcula las características geométricas de una nube de puntos LiDAR con el fin de obtener el tamaño óptimo del vecindario de cada punto. Para ello utilizan un vecindario esférico de tamaño variable y determinan la geometría local a partir de los valores propios de la matriz covarianza del vecindario, consiguiendo determinar si el vecindario define una geometría lineal, plana o volumétrica y determinando finalmente el radio óptimo del vecindario esférico en cada caso. Finalmente Ural y Shan (2012) utilizan también los valores propios de la matriz covarianza para definir el vecindario óptimo de cada punto LiDAR. En este caso el vecindario aumenta paulatinamente de tamaño de manera iterativa, de forma que en cada iteración se analizan los valores propios de la matriz de covarianza con el fin de detectar cambios bruscos en la superficie definida por dicho vecindario. El proceso de crecimiento del vecindario finaliza cuando se detectan saltos en la superficie, consiguiendo así un vecindario continuo asociado a una misma superficie sin variaciones abruptas en altitud y pendiente.

II. 3. Detección de errores groseros ('*outliers*')

Debido a un mal funcionamiento del sistema de captura de datos, las nubes de puntos LiDAR suelen contener valores de elevación anómalos que están por encima o por debajo del valor real del terreno. Los puntos LiDAR que tienen una altura superior a la esperada suele deberse a la reflexión del pulsos láser sobre aves u otros objetos (Sithole y Vosselman 2004), mientras que aquellos que poseen una cota por debajo al terreno real se deben principalmente a problema de emisión y recepción del haz láser (Sithole y Vosselman 2004) así como a errores aleatorios causados por el ruido de los sensores (Meng et al. 2010).

La importancia de la detección de estos errores grosesor ('*outliers*') que proporcionan valores atípicos en los datos LiDAR, y su influencia en la calidad y la precisión de los productos derivados tales como modelos digitales del terreno ha sido objeto de diferentes trabajos (Höhle 2009; Aguilar y Mills 2008;Peng y Shih 2006; Akca et al. 2009).Así mismo, también son objeto de estudio cuando el objetivo es clasificar los puntos LiDAR en las distintas entidades que conforman la escena (Forlani et al. 2006; Chehata et al. 2008). Además, la detección de valores atípicos es un paso previo esencial en el modelado 3D de edificios y en el filtrado de datos LiDAR (Parian y Sargent 2007; Sotoodeh 2006 y 2007; Chen et al. 2007; Meng, et al. 2009; Silván-Cárdenas y Wang 2006; Kobler et al. 2007; Arefi et al. 2007; Sithole y Vosselman 2004; Höhle 2009).

Existen muy diversos métodos para detectar puntos de cota anómala en el conjunto de datos LiDAR. No obstante, en la mayoría de los casos se pueden clasificar en cinco grupos: métodos de distribución, métodos basados en la profundidad, métodos basados en segmentación, métodos basados en la distancia y métodos basados en la densidad de los datos.

En los datos LiDAR, los valores atípicos se presentan como puntos aislados o pequeñas regiones con una altura muy superior o inferior a la de los puntos de su vecindario. Tal y como demuestran algunos trabajos (Meng et al. 2009; Silván-Cárdenas y Wang 2006), resulta muy útil el análisis de la distribución de frecuencias de los valores de elevación para detectar los puntos de altitud atípica. Además, también son de uso común los métodos que utilizan filtros morfológicos (Chen et al. 2007; Kobler et al. 2007; Mongus y Zalik 2012) así como aquellos basados en realizar un análisis de la densidad de puntos (Sotoodeh 2006 y 2007).

Tanto el análisis de la distribución de frecuencias como la aplicación de los filtros morfológicos son métodos basados en comparar el atributo de elevación de forma local o global, de tal forma que mediante la definición de un umbral en elevación se analiza la altura de los puntos, considerando valores atípicos aquellos que superan el umbral previamente establecido. El principal problema de estos métodos es la dificultad de establecer el parámetro umbral en altitud de forma automática y una revisión y análisis debe de ser llevado a cabo por un operador en la mayoría de los casos. Existen números estudios donde se hace uso del análisis de la distribución de frecuencias para detectar valores atípicos (Shen et al. 2011, Lin y Zhang 2014; Meng et al. 2009; Silván-Cárdenas y Wang 2006). En estos trabajos primero se detectaron valores atípicos mediante la construcción y posterior análisis de un histograma de frecuencias obtenido a partir de la elevación de los datos LiDAR, seguidamente se realizó para cada punto LiDAR un análisis mediante un parámetro umbral en desnivel , detectando y eliminando aquellos puntos cuyo desnivel con respecto a sus vecinos era mayor que el umbral seleccionado. Para establecer el vecindario en estos casos suele utilizarse la relación de conectividad que aporta la triangulación de Delaunay (Meng et al. 2009; Silván-Cárdenas y Wang, 2006) o bien una estructura del tipo kd-tree (Lin y Zhang 2014). En cualquier caso, la completa automatización del proceso no resulta viable y es preciso una revisión de los errores groseros detectados (Lin y Zhang 2014). En estos métodos distintas variables estadísticas

son utilizadas para realizar el análisis del histograma de frecuencias, como la eliminación de puntos con elevación superior o inferior a un percentil calculado a partir de todos los datos (Wack y Wimmer,2002; Riaño et al. 2004).

Referente a la utilización de filtros morfológicos para eliminar errores groseros, diversos estudios (Kobler et al. 2007; Mongus y Zalik 2012) muestran la utilizad de aplicar operaciones morfológicas de apertura para eliminar valores atípicos de alta elevación, así como la eficacia del uso de operaciones de erosión para eliminar valores atípicos de baja altitud. No obstante, estos métodos no eliminan correctamente los puntos outliers cuando están agrupados en regiones.

En cualquier caso, debido a la variabilidad de los valores atípicos en función de las características de la escena, la completa automatización de la detección y eliminación de outliers resulta una tarea imposible (Lin y Zhang 2014), siendo todavía un campo de investigación abierto que precisa nuevos avances.

II. 4. Detección y extracción de entidades

Los datos registrados por el sistema LiDAR contienen una densa información 3D de la superficie terrestre. La principal ventaja de estos sistemas radica en que la medición de puntos se realiza de manera directa mediante un sistema de distanciometría láser, consiguiendo obtener así información tridimensional de forma rápida y precisa.

Durante la última década, numerosos investigadores han desarrollado metodologías para generar modelos digitales del terreno (Akel et al. 2003; Brovelli et al. 2004; Pfeifer y Mandlburger 2008, Chen et al. 2012) y modelos urbanos (Alharthy y Betel 2004; Dorninger y Nothegger 2007; Kwak et al. 2012) utilizando tecnología LiDAR. Estas metodologías tienen un papel importante en diversas aplicaciones, tales como la planificación urbana, los proyectos de ingeniería civil y la protección del medio ambiente.

En zonas urbanas, los datos LiDAR se utilizan en aplicaciones tan diversas como simulación de inundaciones (Sole et al. 2012), la detección automática de cambios en edificios (Matikainen 2010), actualización cartográfica (Sánchez y Lerma 2012), o la detección de cambios en evaluación de desastres (Trinder y Salah 2012). Además resulta frecuente la utilización de datos LiDAR en el modelado 3D de carreteras (Oude Elberink y Vosselman 2009), detección de árboles individuales (Koch et al. 2006 y Liu et al. 2013), cálculo de la altura de los árboles y la estimación de parámetros relacionados con la masa forestal tales como el volumen de vegetación arbustiva y la biomasa (de Sevilla Riaza et al. 2008, Wang et al. 2009; Estornell et al. 2011; Estornell et al. 2012; Landa et al. 2013), detección de líneas eléctricas y cartografía de infraestructuras (Lin et al. 2012), etc.

En la totalidad de aplicaciones anteriormente comentadas, como paso previo se precisa disponer de los puntos clasificados en las distintas entidades u objetos que componen la escena. Esta tarea es de vital importancia en el procesado de datos y los errores cometidos se transmiten directamente al producto final que se desea obtener. Así por ejemplo, si el objetivo es obtener modelos digitales del terreno, el error final del modelo estará condicionado por tres factores: error altimétrico asociado a la captura de datos LiDAR, error asociado al proceso de interpolación y error de clasificación. Mientras que los dos primeros factores no constituyen un error a tener en consideración en la mayoría de los casos (Haugerud y Harding 2001), el error asociado al proceso de clasificación puede ser potencialmente grande, sobre todo en zonas arboladas y altas pendientes que puede llegar al orden de metros (Haugerud y Harding 2001). En consecuencia, errores producidos en el proceso de clasificación se transmiten al producto final y sus aplicaciones, como la utilización de la clase terreno en modelos hidráulicos o la estimación de parámetros

asociados a la vegetación para el cálculo de biomasa (Hutton y Brazier 2012; Lin y Zhang 2014).

La clasificación de los puntos LiDAR en las diferentes entidades es una tarea no exenta de dificultad, ya que los datos LiDAR brutos carecen de información semántica que permita obtener de forma directa información sobre el objeto en el que cada pulso láser fue reflejado. Es por ello que en los últimos años se han desarrollado multitud de algoritmos para segmentar los datos en diferentes entidades básicas tales como terreno, puentes, edificios o vegetación. En los siguientes apartados se describen los trabajos más relevantes realizados al respecto.

II.4.1. Detección y extracción de puntos de tierra

La detección y extracción de puntos pertenecientes al terreno no solamente es un aspecto esencial para la generación de modelos digitales del terreno (Yunfei et al. 2008, Fagua et al. 2011, Pérez et al. 2012), sino que es el punto de partida para la gran mayoría de algoritmos cuyo objetivo es la extracción de otras entidades como edificios o vegetación. En este sentido, una gran variedad de algoritmos de filtrado han sido desarrollados en los últimos años. No obstante, la automatización completa del proceso resulta de gran dificultad (Baltsavias 1999). Los algoritmos de detección y filtrados de puntos pertenecientes al terreno los podemos agrupar en los siguientes tipos (Sithole y Vosselman 2004; Pérez et al 2012).

II.4.1.1. Filtrado morfológico

Este tipo de filtros utilizan operadores de morfología matemática que se aplican sobre un conjunto de datos situados dentro de un elemento estructural concreto que se desplaza sobre la zona de análisis. El elemento estructural puede ser de tamaño fijo o variable y suele tratarse de una ventana o cuadrícula. Las operaciones básicas que se aplican son la erosión y la dilatación. La dilatación consiste en la selección del punto de altitud máxima dentro del elemento estructural y la erosión en la selección del punto de altura mínima. Además, es posible combinar ambas operaciones dando lugar a una apertura o a un cierre. En el cierre se aplica primero la dilatación y posteriormente la erosión, mientras que en la apertura se invierte el orden de ambos operadores.

Linderberger (1993) propuso el primer filtro basado en el uso de la morfología matemática aplicado a perfiles capturados mediante un láser aerotransportado, para ello utilizó un elemento estructural lineal de tamaño constante donde fue aplicado un proceso de apertura utilizando un cierto umbral. Voselman (2000) desarrolló un filtro morfológico introduciendo un umbral en pendiente ligado a la distancia con un elemento estructural de tamaño constante. Básicamente el proceso consistía en situar un cono centrado en cada punto de los datos LiDAR, de tal forma que si no existe ningún punto dentro del cono significa que el punto considerado es el punto que tiene menor cota en todo su vecindario y que por tanto, puede ser considerado como terreno. Mediante este procedimiento, se puede modelar la variación de altura admisible en función de la distancia y diferenciar los puntos pertenecientes al terreno de los que no lo son. De forma similar, Sithole (2001) y Roggero (2001) incorporan también la pendiente local del terreno en cada punto para aplicarla dentro del elemento estructural en vez de trabajar con la pendiente global del área de estudio. Sithole (2001) sitúa un cono invertido en cada punto de cota menor siguiendo un proceso similar al descrito por Vosselman (2000), mientras que Roggero (2001) utiliza los datos para crear una malla regular y aplicar un operador local obteniendo así la pendiente del punto más bajo. De esta forma, es posible comparar la altitud de cada punto con la del punto más bajo de su vecindario. La distancia y la diferencia de altura con respecto al punto más bajo se utilizan para el cálculo de una regresión lineal mediante la asignación de pesos

en función del incremento de cota. Finalmente, tras el cálculo de la regresión lineal se obtiene una aproximación al terreno y es posible determinar así los puntos que pertenecen a él.

Considerando que el uso de un elemento estructural de tamaño fijo puede transmitir errores a los resultados, diferentes investigadores han propuesto métodos donde el tamaño es variable. Kilian et al. (1996) utilizan varios tamaños asociando distintos pesos para el cálculo de la altitud final del terreno. En este trabajo, un filtro de apertura se aplica mientras que el tamaño del elemento estructural aumenta iterativamente, aplicando un mayor peso a los puntos detectados como terreno a partir de pequeñas ventanas. Lohmann et al. (2000) aplican un filtro dual-Rank previamente a la aplicación de un filtro morfológico con el fin de eliminar que no pertenecen al terreno y Arefi y Hahn (2005) proponen un filtro morfológico de aplicación progresiva utilizando un operador de distancia geodésica. Además de variar el tamaño del elemento estructural, algunos autores proponen modificar la forma del mismo, como es el caso de Kobler et al. (2007), que propone la rotación del elemento estructural con el fin de adaptarlo a la pendiente local.

Otros autores, han aplicado el filtrado morfológico de forma adaptativa teniendo en cuenta las características locales del terreno. Es el caso de Chen et al. (2007) o Kim y Shan (2012) que aplican el filtro morfológico únicamente a puntos de discontinuidad en superficie previamente detectados mediante un proceso iterativo, consiguiendo así una metodología para extraer los puntos pertenecientes al terreno que presenta una baja dependencia de parámetros tales como la pendiente o el tamaño de los edificios, lo que permite a este enfoque ser aplicado a diferentes terrenos con un cambio mínimo de parámetros umbrales. Susaki (2012) también utiliza un filtro adaptativo basado en el concepto de filtrado morfológico. En este caso la metodología propuesta incorpora características propias de filtros basados en segmentación consiguiendo determinar de forma automática el parámetro umbral en pendiente considerando en cada caso la pendiente local. Por otro lado, la distribución aleatoria de los puntos LiDAR también es un factor que debe de ser considerado. En este sentido, Li (2013) propone un filtro morfológico basado en el análisis multi-gradiante en función de las características de distribución de los datos LiDAR. El filtro realiza el cálculo del multi-gradiante de cada punto a partir de sus puntos más próximos utilizando un elemento de estructuración en simetría. De esta forma, se puede superar la influencia de la distribución irregular de las nubes de puntos LiDAR y alcanzar resultados óptimos en escenas complejas.

En los filtros morfológicos también se incluyen aquellos que se basan en un proceso iterativo de selección de puntos de cota mínima dentro de una ventana de búsqueda, de tamaño fijo o variable, que se mueve por la totalidad de la zona de estudio. Una de las metodologías más utilizadas es la generación de un MDT provisional mediante la selección de los puntos de menor altitud en ventanas de un tamaño determinado y posteriormente, realizar una nueva selección de puntos de altitud mínima pero esta vez en ventanas de tamaño menor. Finalmente la diferencia de altitud entre los puntos seleccionados y el MDT provisional es calculada y los puntos próximos son incorporados al MDT. El proceso es iterativo hasta que ningún nuevo punto es clasificado como terreno, momento en el cual se considera que la clase terreno está completa y se genera el MDT definitivo. Mediante esta metodología se consigue eliminar del conjunto de datos aquellos puntos que no pertenecen al terreno como la vegetación o los edificios. Hyyppä et al. (2001) utilizaron este enfoque obteniendo un MDT provisional mediante la selección de puntos de cota mínima en ventanas cuadradas y posteriormente se procedió a comparar la altitud de la totalidad de puntos LiDAR con el MDT provisional clasificando los puntos como terreno o no terreno a partir del uso de varios umbrales. Popescu et al. (2002) utilizaron un procedimiento similar para generar un MDT provisional y posteriormente aplicar filtros a la imagen obtenida. Por otro lado, Wack y Wimmer (2002) calcularon un modelo digital de elevaciones a partir de la selección del punto más bajo en celdas de tamaño decreciente, aplicando en cada iteración un filtro laplaciano para eliminar los puntos no pertenecientes al terreno. Metodologías

similares a la anterior pero con modificaciones en el tamaño de las ventanas y en los parámetros umbrales utilizados son las empleadas por Yu et al. (2004) y Clark et al. (2004).

II.4.1.2. Algoritmos de densificación progresiva

Estos filtros tienen un enfoque progresivo y se basan en el hecho de que el relieve del terreno no suele experimentar grandes discontinuidades. El proceso comienza con la triangulación de un pequeño número de puntos pertenecientes al conjunto de puntos de altitud mínima de la zona de estudio, obteniendo así una primera aproximación de la superficie del terreno. Posteriormente y en sucesivas iteraciones, nuevos puntos son añadidos al terreno en el caso de que cumplan un determinado umbral en cuanto a incremento de altitud con respecto a la superficie de referencia. Axelsson (2000) fue el primero en proponer este tipo de filtros. En este caso la triangulación inicial fue realizada mediante la división del área de estudio en una cuadrícula regular y la selección de los puntos de menor cota de cada una de las cuadrículas. El resto de puntos pertenecientes al terreno eran añadidos a la triangulación en sucesivas iteraciones. Para ello, un análisis de cada punto con respecto a su triángulo asociado se llevó a cabo teniendo en cuenta la distancia desde el punto al triángulo y los ángulos formados entre dicho punto y los vértices del triángulo. En este sentido, otros autores como Von Hansen y Vogtle (1999) utilizan la altura del punto con respecto a su proyección en la triangulación en vez de la distancia.

Sohn y Dowman (2002) aplicaron un enfoque similar a los anteriores pero utilizaron una densificación descendiente inicial previa a la densificación ascendiente final. Así mismo, en los últimos años algunos autores han propuesto combinar métodos de densificación con filtros de segmentación. Es el caso de Pérez et al. (2012), que proponen realizar una segmentación de la totalidad de la escena mediante crecimiento de regiones y, por otro lado, aplicar un algoritmo de densificación progresiva de TIN. Finalmente ambos resultados se fusionan consiguiendo mejorar la clasificación de puntos pertenecientes al terreno. De forma similar, Ekhtari et al. (2008) combinan procesos de segmentación y densificación progresiva de TIN para extraer edificios en un conjunto de datos LiDAR pertenecientes a entornos urbanos. Este trabajo propone generar modelos digitales de superficie a partir del primer y último pulso reflejado para posteriormente segmentar dichos MDS diferenciando superficies lisas y rugosas y finalmente comparar ambas segmentaciones y combinarlas con el método de densificación progresiva de TIN con el fin de extraer los puntos pertenecientes al terreno así como los pertenecientes a edificios. Lin y Zhang (2014) también combinan segmentación con un proceso de densificación de TIN. Básicamente el método que proponen consiste en segmentar los datos LiDAR en superficies continuas, para ello aplican una segmentación mediante crecimiento de regiones que utiliza tres parámetros: un umbral en variación del vector normal a la superficie con respecto al punto semilla, una distancia vertical al plano de referencia y la longitud del radio para establecer el vecindario esférico. Seguidamente el algoritmo de densificación utiliza como unidad de procesamiento las agrupaciones obtenidas tras la segmentación en lugar de los puntos de manera individual, de forma que una agrupación es añadida al terreno siempre que más de la mitad de sus puntos estén asociados a pulsos de retorno único y además cumplan con los parámetros umbrales asociados al proceso de densificación de TIN (distancia ortogonal con respecto a la superficie de referencia, ángulos con respecto a los vértices del triángulo asociado....). El proceso de densificación de TIN propuesto en este trabajo utiliza cuatro parámetros más que el método clásico de densificación, lo que conlleva un aumento del tiempo de cómputo (18,3 veces mayor que el método clásico). No obstante, el estudio muestra una mejora significativa en los resultados experimentales, consiguiendo reducir significativamente errores de comisión asociados a la clasificación de puntos bajos de vegetación y otros objetos como terreno, además de eliminar parcialmente errores de omisión ligados al mal funcionamiento de los procesos de densificación en zonas con cambios bruscos de pendiente.

Este tipo de algoritmo sirvió de referencia para desarrollar el software TerraScan de 3D Laser Mapping de uso muy extendido en el tratamiento de datos LiDAR.

II.4.1.3. Filtros basados en la interpolación de superficies

Al igual que con el enfoque de la densificación progresiva, en este tipo de filtros se utiliza una reconstrucción inicial de la superficie del terreno a partir de la nube de puntos para posteriormente realizar un filtrado y extraer los puntos de tierra. No obstante, en este caso, se parte de la hipótesis de que la totalidad de puntos LiDAR pertenecen al terreno, para posteriormente y mediante un proceso iterativo reducir la influencia de aquellos que realmente no pertenecen a dicha clase.

Básicamente el objetivo de estos filtros es comparar la altitud de los puntos con los de una superficie estimada mediante interpolación, para posteriormente minimizar la influencia de puntos no pertenecientes al terreno mediante procesos iterativos. Uno de los métodos más utilizados es el propuesto por Kraus y Pfeifer (1998), conocido como método de interpolación robusto. Este método integra la extracción de los puntos de tierra y la generación del MDT del área de estudio en un mismo proceso. El algoritmo viene constituido por un proceso iterativo en el cual se calcula la superficie del terreno mediante una interpolación lineal utilizando mínimos cuadrados con un conjunto de funciones de peso. Los puntos pertenecientes al terreno tienen residuos negativos, mientras que aquellos puntos pertenecientes a otros objetos como la vegetación tienen residuos muy pequeños o positivos. A partir de los residuos se aplica un peso que oscila entro 0 y 1 para el cálculo de la interpolación. Este proceso se repite hasta obtener el Modelo Digital del Terreno definitivo, momento en el cual, los puntos pertenecientes al terreno han sido detectados por completo. Este método fue mejorado por en Pfeifer et al. (2001) y Briese et al. (2002). En las metodologías propuestas por estos autores primero se detecta una primera aproximación del terreno y posteriormente se asignan los pesos a cada punto para realizar el ajuste y obtener la superficie definitiva. Estos pesos se asignan en función de la distancia vertical de cada punto con la superficie aproximada, constituyendo así un proceso iterativo hasta obtener la superficie del terreno final. Por otro lado, podemos encontrar algoritmos como el propuesto por Elmqvist et al. (2001) basados en el concepto de fuerzas interiores y exteriores para determinar la superficie del terreno.

Estos algoritmos utilizan métodos de interpolación muy diversos. Brovelli et al. (2004) propusieron una metodología basada en interpolación de splines y en la detección y extracción de bordes para la clasificación de los puntos pertenecientes al terreno. Por otro lado, Evans y Hudak (2007) utilizaron un algoritmo de interpolación TPS y Zhang y Whitman (2005) emplearon métodos estadísticos.

Para finalizar, notar que el software SCOP++ desarrollado por el Instituto de Fotogrametría y Teledetección de Viena, muy utilizado en el tratamiento de datos LiDAR utiliza como base un filtro basado en interpolación, siendo uno de los programas en este ámbito de mayor renombre.

II.4.1.4. Filtros basados en segmentación

El objetivo de este tipo de filtros es agrupar en regiones homogéneas los puntos con atributos similares para posteriormente clasificar dichas regiones o bien utilizarlas en la reconstrucción virtual de edificios.

Con respecto al uso de técnicas de segmentación para detectar la clase terreno en nubes de puntos LiDAR, generalmente las metodologías existentes suponen que las agrupaciones o las regiones pertenecientes a objetos urbanos y vegetación están situadas a mayor altura

que las regiones de tierra. Habitualmente los filtros basados en segmentación funcionan en dos fases, siendo la primera el propio proceso de segmentación y la segunda el filtrado de las regiones obtenidas. Nardinocchi et al. (2003), aplicaron una segmentación mediante crecimiento de regiones utilizando las diferencias de altura para obtener las agrupaciones. La descripción geométrica y topográfica de las regiones obtenidas fueron analizadas y a partir de un conjunto de reglas se clasificaron en tres clases principales: terreno, edificios y vegetación. Sithole (2005) calculó y analizó la diferencias de altura entre agrupaciones vecinas para clasificarlas posteriormente como terreno u otros objetos. Lohmann (2002) analizó la compacidad de las agrupaciones y estudió la diferencia de altura entre agrupaciones vecinas para diferenciar el terreno y otras superficies. De forma análoga Sithole y Vosselman (2005) compararon la altura de regiones próximas en diferentes direcciones y a partir de un conjunto de reglas de decisión consiguieron clasificarlas como terreno y no terreno. Un enfoque similar es el propuesto por Shen et al. (2012), que supone que las agrupaciones pertenecientes al terreno son más horizontales y de menor altura que las pertenecientes a los objetos adyacentes.

Referente a las técnicas de segmentación utilizadas para obtener las regiones que conforman la escena, existen multitud de métodos de segmentación que han sido utilizados para segmentar datos LiDAR. Uno de los más utilizados es la segmentación por crecimiento de regiones. Este tipo de algoritmos fue propuesto por Besl y Jain (1988) y se basa en la selección de puntos semilla que constituyen el inicio de cada región, para posteriormente proceder a segmentar los datos mediante el crecimiento de dichas regiones a partir de criterios predefinidos tales como la proximidad entre puntos y la pendiente. Tovari y Pfeifer (2005) fueron los primeros en introducir el proceso de crecimiento de regiones para segmentar datos LiDAR. En este estudio, utilizaron el vector normal asociado a cada punto y un parámetro umbral en distancia para llevar a cabo la segmentación. Rabbani et al. (2006) propusieron un proceso similar pero utilizando un criterio de rugosidad o aspereza para agrupar puntos pertenecientes a una misma superficie. Pu y Vosselman (2006) propusieron un método de crecimiento de regiones para segmentar datos láser terrestres basándose en un umbral en proximidad y criterios globales de planicidad.

Otro tipo de algoritmos utilizados en procesos de segmentación son los llamados métodos de ajuste. Estos métodos realizan el ajuste de la forma de primitivas geométricas y son comúnmente utilizados para la representación matemática de superficies planas (Lari et al. 2011). Un algoritmo de uso muy común que tiene este enfoque es el método RANSAC (RANdom SAmple Consensus) propuesto por Fischer y Bolles (1981). Este método permite separar los puntos que no pertenecen a una superficie determinada y puede aplicarse sobre datos LiDAR con el fin de obtener modelos de edificios de forma automática (Tarsh-Kurdi et al. 2007).

Los métodos de segmentación basados en agrupación por atributos resultan también muy eficientes en la identificación de patrones homogéneos para segmentar datos. Este método comprende principalmente dos procesos: cálculo de los atributos y la agrupación de los datos en función de los atributos calculados. Filin y Pfeifer (2006) utilizaron este enfoque y presentaron un método de segmentación basado en los vectores normales calculados a partir de los puntos más próximos que fueron detectados mediante una función que se adaptaba a la pendiente. Para la segmentación, utilizaron la pendiente del vector normal en las direcciones X e Y así como la diferencia de altura entre cada punto y su vecindario. Además, el atributo en desnivel también fue utilizado para diferenciar planos paralelos que comparten las mismas pendientes asociadas al vector normal. Biosca y Lerma (2008) sugirieron un enfoque de agrupamiento difuso ('*fuzzy clustering*') en combinación con un proceso de fusión de regiones en una nube de datos láser terrestre. Kim et al. (2007) propusieron un método de segmentación en regiones planas para un conjunto de puntos LiDAR, para ello, utilizaron magnitudes que caracterizan el vector normal a las superficies como atributo para obtener las regiones. Estos métodos basados en agrupamiento por atributos son eficientes para identificar regiones homogéneas en los datos LiDAR. No

obstante, tienen como inconveniente un alto coste computacional cuando se trata de atributos multidireccionales asociados a una gran cantidad de puntos. Además, son altamente dependientes del proceso de definición de vecindario. Teniendo en cuenta estos inconvenientes, Lari et al. (2011) segmentaron los datos en superficies planas y rugosas organizando previamente los datos en una estructura kd-tree y utilizando una estructura cilíndrica para determinar correctamente el vecindario y mejorar el rendimiento computacional.

Otro de los algoritmos de agrupamiento que está siendo utilizado frecuentemente para segmentar datos LiDAR son los basados en la densidad de puntos, como el algoritmo de segmentación ADBSCAN. Ghosh y Lohani (2011) utilizan este algoritmo para extraer las diferentes agrupaciones pertenecientes a objetos naturales y artificiales. Posteriormente las distintas agrupaciones fueron procesadas y simplificadas para su visualización. Por otro lado, los parámetros estadísticos también pueden utilizarse para segmentar datos LiDAR. Es el caso de Costantino y Angelini (2011) que utilizan el coeficiente de curtosis y asimetría para extraer de forma automática características de un conjunto de datos LiDAR. En este trabajo la asimetría fue analizada mediante un filtro con el fin de determinar una primera clasificación provisional y posteriormente, se aplicó el coeficiente de curtosis para eliminar ruido y obtener una clasificación más precisa. De manera análoga, Crosilla et al (2001) desarrollaron un nuevo método de clasificación no supervisada de datos LiDAR mediante el análisis iterativo de la asimetría y el coeficiente de cortosis sobre la distribución de los puntos en cuanto a elevación y nivel de intensidad, consiguiendo una clasificación final con errores que oscilan entre el 1,2% y el 8,9%.

Además de los algoritmos de segmentación basados en densidad, también son frecuentemente utilizados los llamados algoritmos particionales como el algoritmo K-Means, CURE, o PSO. Samadzadegan y Saeedi (2009) demuestran las ventajas de este último (PSO) con respecto al resto en el tratamiento de datos LiDAR. En este estudio se describe un algoritmo de optimización de enjambre de partículas (PSO) que encuentra soluciones globales al problema de agrupamiento de datos LiDAR en zona urbana. El trabajo integra la simplicidad del algoritmo k-means con la capacidad del algoritmo PSO consiguiendo muy buenos resultados en cuanto a términos de precisión y tiempo de cómputo.

Además de los algoritmos expuestos anteriormente, en los últimos años se han desarrollado nuevas metodologías para segmentar datos láser aerotransportados con buenos resultados. Es el caso de métodos basados en "min-cut" que resuelven problemas de optimización mediante el corte de grafos. Golovinskiy y Funhouser (2009) utilizaron un método basado en min-cut para segmentar el plano de fondo de los objetos en una nube de puntos. De forma similar, Ural y Shan (2012) utilizan el algoritmo de minimización de energía min-cut propuesto por Boykov et al. (2001) para obtener mínimos locales mediante el recorte de grafos y utilizan el sistema para clasificar entidades en una nube de puntos. El proceso consiste en generar un grafo donde los puntos pertenecientes a un mismo vecindario constituyen nodos que están conectados mediante aristas. Seguidamente se utiliza el método min-cut para minimizar la función de coste considerando criterios de planicidad, consiguiendo así segmentar los datos en función del plano al que pertenecen.

Por último, notar las altas precisiones de distintos métodos propuestos que fusionan el proceso de segmentación con otras metodologías para detectar el terreno en datos láser aerotransportados como por ejemplo los desarrollados por Ekhtari et al. (2008), Pérez et al. (2012), Lin y Zhang (2014), que aplican un algoritmo de densificación de TIN tras haber segmentado los datos LiDAR mediante crecimiento de regiones.

II.4.1.5. Otros métodos

En este apartado se incluyen aquellos algoritmos que no pertenecen a los métodos anteriores como por ejemplo, el filtro de interpolación iterativa propuesto por Klober et al. (2007), el método de contornos activos propuesto por Elmqvist (2001) o el uso de información completa de la onda láser.

Elmqvist (2001) utiliza un algoritmo denominado "contornos activos" que funciona como una red para eliminar los puntos que no pertenecen al terreno. Para ello, calcula una superficie horizontal por debajo de todos los puntos y seguidamente ajusta dicha superficie hacia arriba hasta adaptarla al terreno consiguiendo descartar así los puntos pertenecientes a la vegetación y edificios. Un enfoque similar es utilizado por Elmqvist et al. (2001), donde una superficie 2.5 D que actúa como una superficie elástica se adapta a los puntos de tierra mediante un esquema de procesamiento jerárquico. Los puntos pertenecientes a árboles y edificios no son detectados como terreno, ya que el límite de elasticidad de la superficie impide alcanzarlos.

Desde 2004, el nuevo sistema ALS llamado LiDAR de onda completa (fullwaveform) está siendo utilizado para registrar la totalidad de la forma del pulso láser reflejado. La señal recibida resultante presenta la suma de todas las reflexiones de la onda en las distintas superficies interceptadas por el pulso láser. Gracias a este sistema, nueva información adicional como la anchura de la huella del pulso LiDAR o la amplitud del pulso son utilizadas para detectar y extraer entidades en el conjunto de datos LiDAR. En este sentido, el nuevo sistema es el objeto de varias investigaciones realizadas en los últimos años (Chauve et al. 2007; Lin et at. 2008; Chehata et al. 2009, Wang et al. 2012). Una de las mayores ventajas del sistema es la capacidad de detectar con mayor precisión la vegetación, especialmente zonas de matorral y sotobosque, lo que permite filtrar en áreas boscosas la vegetación con una mayor eficiencia y obtener MDTs más precisos (Doneus et al. 2008). Esta tecnología puede ser aplicada en campos tan novedosos como la detección automática de asentamientos arqueológicos ocultos bajo la cubierta vegetal (Doneus y Briese 2006; Doneus y Briese 2008). No obstante, el sistema aún requiere un esfuerzo de investigación considerable para demostrar su potencial (Lin y Mills 2009) y gran parte de sus aplicaciones están todavía bajo investigación.

II.4.2. Detección de puntos pertenecientes a puentes

La gran mayoría de los trabajos publicados referentes a la detección de entidades a partir de datos LiDAR se centran principalmente en dos aspectos. Por un lado, la detección de la clase terreno con el fin de generar precisos modelos digitales del terreno, y por otro, la detección de edificios con el objetivo de generar modelos 3D de las construcciones. Consecuentemente, existen escasas publicaciones cuyo objetivo sea la detección de puentes como entidad propia, principalmente, debido a que los algoritmos comúnmente utilizados para detectar la clase terreno, tales como el método clásico de densificación de TIN, filtros morfológicos o filtros basados en interpolación de superficies, son capaces de aislar el terreno y filtrar los puntos pertenecientes a puentes sin análisis adicionales. No obstante, el hecho de no diferenciar los puentes como una entidad propia suele ser una fuente de error común, principalmente en aquellos algoritmos que combinan técnicas de segmentación con procesos de densificación para detectar el terreno (Pérez et al. 2012, Lin y Zhang 2014). Además, la detección y extracción de puentes tiene especial relevancia en la reconstrucción digital de ciudades en 3D (Zheng et al. 2013).

Referente a los algoritmos que detectan puentes mediante el uso de tecnología LiDAR, una de las características más relevantes que tienen en consideración es que los edificios se elevan verticalmente por encima del terreno en todas las direcciones, mientras que los puentes solamente en algunas partes (Sithole y Vosselman 2003). Bajo esta premisa,

Sithole y Vosselman (2006) presentan un algoritmo geométrico que detecta puentes mediante el uso de perfiles transversales y el análisis de la información topológica contenida en los mismos. El trabajo pone de relevancia la precisión de los resultados y la capacidad del algoritmo para detectar puentes que no poseen anchura uniforme ni bordes paralelos. Un enfoque similar es el presentado por Zheng et al. (2013). En este trabajo se analizan las características estructurales y geométricas de los puentes para su detección. Para ello, tras detectar los bordes de los puentes sobre un MDT obtenido a partir de datos LiDAR, se estudian las características de los mismos en cuanto al paralelismo de las líneas que los definen. Finalmente, un análisis de las direcciones principales y de la continuidad estructural entre los puntos del vecindario se lleva a cabo para detectar la totalidad del conjunto de puntos que pertenecen a los puentes. De forma análoga, Evans (2008) utiliza un MDT obtenido a partir de datos LiDAR para detectar puentes. En este caso, también se detectan los bordes de los puentes como primera instancia, para posteriormente analizar distintas características geométricas para su detección, tales como la simetría existente entre el cambio de aspecto de bordes opuestos o cambios en la anchura del puente.

La fusión de datos LiDAR con imágenes aéreas para detectar puentes también ha sido objeto de estudio. Ju (2012) propone una metodología que consiste en detectar el grueso de los puentes en el conjunto de datos LiDAR y posteriormente orientar una imagen aérea de alta resolución mediante la correlación existente entre la imagen y la intensidad de la señal reflejada del conjunto de datos LiDAR. Finalmente, la transformada de Hough es utilizada para extraer el contorno de los puentes en la imagen aérea y fusionarlo con los datos LiDAR para la detección precisa de los mismos.

II.4.3. Detección de puntos pertenecientes a edificios y vegetación

Detectar edificios y separarlos de la vegetación y otras entidades es una importante y difícil tarea. La mayoría de los autores primero separan los puntos de tierra desnuda antes de extraer los objetos adicionales. Como norma general, para detectar edificios se utilizan filtros basados en segmentación. Además de los algoritmos de segmentación mencionados con anterioridad que también son utilizados para detectar edificios y planos de techo, otros autores han propuesto diversos algoritmos con el mismo propósito. Alharthy y Bethel (2004) extraen edificios utilizando segmentación por crecimiento de regiones para detectar los distintos planos que constituyen los tejados mediante un ajuste mínimo cuadrático. Kwak et al. (2012) utilizan un algoritmo de segmentación similar que es aplicado para extraer segmentos planos y rugosos. Ekhtari et al. (2008) combinan una segmentación en regiones rugosas y planas con un proceso progresivo de densificación de TIN para detectar puntos de tierra y obtener la clase edificio en zonas urbanas. Chehata et al. (2009) analizan datos LiDAR urbanos y extraen vegetación y edificios utilizando información completa de la onda y un algoritmo de segmentación basado en un árbol aleatorio. Finalmente, Samadzadegan y Saeedi (2009) segmentan datos LiDAR urbanos pertenecientes a zonas residenciales e industriales mediante los algoritmos de k-media y PSO obteniendo buenos resultados en la detección de edificios.

El análisis de las texturas obtenidas en la escena también es una práctica común para diferenciar edificios de vegetación, como por ejemplo, el análisis de texturas obtenidas a partir de la altura (Hug 1997; Maas 1999; Oude Elberink y Maas 2000) o rugosidad (Brunn y Weidner 1998). Así mismo, la utilización de datos LiDAR en combinación con otras fuentes de datos también muestran resultados más que aceptables en la detección y extracción de las diferentes entidades que conforman la escena. Es el caso de Kwak et al. (2012) que basándose en la metodología propuesta por Lari et al. (2011) segmentan una nube de datos LiDAR en superficies lisas y rugosas en combinación con imágenes aéreas con el fin de extraer los distintos objetos de la escena. De forma similar Moussa y Sheimy (2012) tras generar un modelo digital de superficies utilizando la información altimétrica de los datos LiDAR proceden a segmentar los datos en planos a la vez que segmentan una ortofoto de la

zona de estudio. Finalmente ambos resultados se fusionan consiguiendo detectar las distintas entidades en zonas urbanas. Un proceso análogo al anterior es el llevado a cabo por Nakano y Chikatsu (2012) que extraen el terreno y el resto de objetos segmentando los datos en superficies continuas utilizando el vector normal y posteriormente detectan los bordes de los objetos a partir del tratamiento de ortofotos digitales del área de estudio, fusionando finalmente ambos procesos.

Otro enfoque es la fusión de datos LiDAR con imágenes multiespectrales. Haala y Brenner (1999) combinaron los datos LiDAR y una fotografía aérea multiespectral en la extracción edificios, consiguiendo mejorar sustancialmente los resultados de la clasificación. Por otro lado, la utilización del índice de vegetación normalizado (NDVI) obtenido a partir de imágenes multiespectrales en combinación con datos LiDAR con el fin de diferenciar edificios de vegetación, es el objeto de multitud de investigaciones (Vögtle y Steinle 2003; Hermosilla y Ruiz 2009; Hermosilla y Ruiz 2011; Matikainen et al. 2010; Hartfield et al. 2011; Moussa y Sheimy 2012; Wang y Li 2013). Vögtle y Steinle (2000) diferenciaron edificios de vegetación mediante el cálculo del NDVI en combinación con el estudio de otras características como el tamaño y forma de los objetos, consiguiendo diferenciar vegetación, edificios y otros objetos de menor tamaño tales como automóviles. Hermosilla y Ruiz (2009) y (2011) evalúan dos métodos para detectar edificios que utilizan imágenes satélites multiespectrales y datos altimétricos obtenidos mediante tecnología LiDAR. En ambos métodos se utiliza como base un modelo de superficies normalizado (nDSM) derivado de los datos LiDAR. El primer enfoque utilizado es un método de clasificación orientada a objetos, donde los diferentes objetos a clasificar son detectados a partir de una segmentación por crecimiento de regiones aplicada sobre el nDSM, posteriormente, diferentes características descriptoras son extraídas (espectrales, de textura, de forma, de altura y de contexto) con el fin de identificar los edificios y separarlos de la vegetación mediante la aplicación de árboles de decisión. El segundo enfoque es un algoritmo basado en establecimiento de umbrales. En este caso, un umbral en altura se aplica sobre el nDSM para detectar los objetos que se alzan sobre el terreno y con el fin de enmascarar la vegetación otro umbral se aplica sobre la imagen de NDVI. Además, para aumentar la precisión de los métodos propuestos un análisis adicional es incluido donde los edificios finalmente clasificados deben de ser colindantes a zonas de sombra. Los resultados muestran como la utilización del método de umbrales sencillos proporciona en general una mayor precisión en las diferentes zonas test, especialmente en zonas periurbanas. Matikainen et al. (2010) utilizan la banda del infrarrojo cercano de una imagen aérea multiespectral en combinación con datos LiDAR para diferenciar edificios de vegetación mediante la aplicación de un árbol de decisión donde el índice de vegetación normalizado y la pendiente son los parámetros fundamentales considerados. Posteriormente, utilizan los resultados para detectar cambios en edificios con el objetivo de actualizar cartografía existente. De manera análoga, Hartfield et al. (2011) utilizan el índice (NDVI) junto con datos LiDAR para diferenciar edificios de vegetación mediante un árbol de decisión. Concretamente aplican un algoritmo basado en árboles de clasificación y regresión (modelo CART) para realizar un mapa de usos del suelo y detectar zonas con vegetación susceptibles de albergar colonias de mosquitos en zonas urbanas o próximas a las mismas. Moussa y Sheimy (2012) generan un modelo digital de superficies a partir la información altimétrica proporcionada por datos LiDAR en zona urbana y posteriormente aplican un algoritmo de segmentación basado en crecimiento de regiones para obtener las distintas superficies continuas que constituyen la escena. Posteriormente, diferencian entre edificios y vegetación a partir del tamaño de los objetos detectados, considerando como vegetación aquellos objetos de un tamaño menor a 10 m^2. Tras obtener esta primera clasificación preliminar, los resultados son mejorados eliminando errores en la clase edificios mediante la aplicación de reglas de decisión a partir de un análisis del índice NDVI. Por último, Wang y Li (2013) aplican un método jerárquico multinivel para detectar la vegetación en casco urbano mediante la información altimétrica proporcionada por los datos láser y el cálculo del índice NDVI a partir de imágenes aéreas multiespectrales. Posteriormente la vegetación detectada es enmascarada en imágenes multitemporales con el fin de realizar un análisis de construcciones demolidas.

Para diferenciar la vegetación de los edificios, otros autores han utilizado información adicional existente en los datos LiDAR tales como la información multi-retorno de los pulsos láser o el nivel de intensidad de la señal LiDAR reflejada.

Pfeifer et al. (2001) analizan la información multi-retorno de la señal LiDAR reflejada para detectar la vegetación penetrable. De forma análoga, Meng, Wang y Currit (2009) utilizan la diferencia de altura entre el primer y último pulso láser, así como otros indicadores de forma, textura y elevación para separar la vegetación de los edificios. Además, un análisis de la variación estadística local de la diferencia de altura entre el primer y último pulso puede ser útil para separar los árboles de las construcciones, ya que una baja variación indica que se trata de una superficie lisa, y por el contrario, una variación alta es un indicador de que estamos sobre una superficie no homogénea como vegetación (Alharthy y Bethel 2002). Otros autores utilizan la altitud asociada al primer y ultimo pulso láser para aplicar un enfoque jerárquico de segmentación y clasificación. Es el caso de Tóvari y Vögtle (2004) que segmentan la escena a partir de la información asociada al primer pulso consiguiendo detectar las edificaciones, y posteriormente, tras enmascarar las construcciones proceden a detectar la vegetación mediante análisis comparativo de la información asociada al primer y último pulso.

El análisis del nivel de intensidad de la señal reflejada también es frecuentemente utilizado para separar la vegetación de las edificaciones (Oude Elberink y Maas 2000; Song et al. 2002; Hui et al. 2008; Shaker y El-Ashmawy 2012). Oude Elberink y Maas (2000) calculan un modelo digital de superficies normalizado a partir de la diferencia entre el modelo digital del terreno (MDT) y modelo digital de superficie (MDS) en una zona urbana. Posteriormente un análisis de texturas es aplicado al MDS resultante consiguiendo clasificar las diferentes entidades que conforman la escena. Por último, la intensidad de la señal reflejada es estudiada para mejorar la clasificación y poder discernir entre distintos tipos de vegetación, así como diferenciar entre caminos y cultivos. Song et al. (2002) realizan un estudio para determinar la viabilidad de la utilización del nivel de intensidad de la señal LiDAR como único dato para clasificar los distintos objetos de la escena, evaluando para ello la separabilidad existente entre las diferentes clases en cuanto al rango de reflectancia se refiere. En este estudio, el nivel de intensidad es interpolado a malla regular (GRID) y un filtro de mediana es aplicado para eliminar el ruido producido por errores sistemáticos (ángulo de giro del espejo, errores en la captura de datos,...) así como errores derivados del propio proceso de interpolación. Finalmente, para evaluar la separabilidad de las intensidades, un método de divergencia transformado es utilizado, consiguiendo así permitir fácilmente interpretar la separabilidad de las diferentes clases. En este trabajo, se concluye que es fácil de diferenciar el asfalto y la vegetación, así como edificios y vegetación mediante la intensidad de la señal reflejada. No obstante, diferenciar edificios de asfalto o zonas de hierba con otro tipo de vegetación utilizando únicamente el nivel de intensidad no es viable, siendo necesario para ello otra información adicional. Hui et al. (2008) utilizan el nivel de intensidad así como la altura asociada a los datos LiDAR para realizar una clasificación supervisada obteniendo cuatro clases diferenciadas: árboles, edificios, tierra desnuda y vegetación baja. En este estudio se observó que el análisis del nivel de intensidad combinado con la altimetría resulta un método eficaz para clasificar datos LiDAR. No obstante, la evaluación cuantitativa de la exactitud de los resultados no se incluyó en el trabajo de investigación. Shaker y El-Ashmawy (2012) generan tres imágenes a partir de datos LiDAR pertenecientes a un entorno urbano: un GRID obtenido a partir del nivel de intensidad de la señal reflejada, un MDS y un MDT. A partir de estas tres capas básicas se derivan otras nuevas como textura del nivel de intensidad, alturas normales (diferencia entre MDS y MDT) o pendiente. Finalmente, tras combinar los distintos datos se aplica un análisis de componentes principales así como una clasificación supervisada. Los resultados de este estudio muestran una precisión global en la clasificación del 43,7% con el uso únicamente del nivel de intensidad, que puede mejorar hasta el 70,9% mediante la combinación del nivel de intensidad, textura, pendiente y alturas normales.

III. DESARROLLO DE ALGORÍTMOS

III. 1. Sistema de importación de datos y desarrollo de una doble estructura de organización espacial de datos LiDAR

III.1.1. Motivación para el desarrollo del algoritmo y relación con los antecedentes

Como fase previa a la clasificación de puntos LiDAR será necesario desarrollar un sistema de importación de datos con el fin de poder almacenar la información asociada a densas nubes de puntos en la memoria del ordenador. Además, con el objetivo de optimizar la memoria de la computadora y evitar que se almacene información sin utilidad del fichero original, el sistema de importación deberá permitir al usuario cargar en memoria únicamente la información necesaria para el posterior procesamiento. Por otro lado, debido a errores en la captura de datos y a la superposición de huellas de pulsos láser cuando la densidad de captura es alta (Filin y Pfeifer 2005), en ocasiones es posible encontrar registros duplicados en el archivo de datos LiDAR o puntos capturados más de una vez que poseen coordenadas idénticas. Esto puede ocasionar errores de cómputo en la ejecución de algoritmos tales como la triangulación de Delaunay, por lo que la aplicación a desarrollar deberá tener en cuenta este aspecto.

Con respecto al sistema de organización y disposición de datos, apuntar que dado que los datos brutos LiDAR no proporcionan información sobre la distribución de los puntos en el terreno y ni sus relaciones de proximidad, será necesario desarrollar un sistema de organización que sirva como base al resto de algoritmos para la búsqueda del vecindario de forma eficaz (Lari et al. 2011). En este sentido, como el objetivo es clasificar los puntos LiDAR en función de la entidad u objeto al que pertenecen de la forma más precisa posible, se trabajará en todo momento con los datos LiDAR brutos dispuestos de forma aleatoria, quedando descartados todos los sistemas basados en interpolar los datos a malla regular (GRID) evitando así la perdida de información asociada al proceso de generalización que ello conlleva (Roggero 2001).

El sistema de organización a desarrollar tendrá que resolver dos aspectos fundamentales. Por una parte, deberá permitir de forma eficiente establecer relaciones de proximidad entre datos LiDAR. Por otro, deberá ser capaz de detectar los puntos LiDAR más próximos a un punto dado que no pertenezca al conjunto de los datos LiDAR. La primera de las cuestiones se soluciona de forma eficaz mediante una organización de datos que utilice la triangulación de Delaunay. Este sistema establece eficientemente la conectividad entre los puntos más próximos facilitando la definición del vecindario dentro de un conjunto de datos tridimensional y es frecuentemente utilizado para organizar datos LiDAR (Vosselman y Dijkman 2001; Filin 2002; Hofmann 2004; Sánchez y Lerma 2012). No obstante, tal y

como se describirá en posteriores apartados, en ocasiones es necesario establecer relaciones topológicas entre los datos LiDAR y un punto que no pertenece al conjunto de los mismos. En este caso, las coordenadas de los puntos externos a los datos LiDAR se obtienen en tiempo de ejecución, y por tanto la utilización de la triangulación como base para la organización supondría recalcular la triangulación en cada una de las operaciones de procesamiento, aumentando el tiempo de cómputo de los algoritmos hasta límites inadmisibles. Por tanto, en estos casos será necesario utilizar un sistema de codificación alternativo que proporcione la posición relativa de un punto con respecto al área total de la zona de estudio de forma eficiente. Entre estos sistemas destacan aquellos cuya base es dividir el espacio en subespacios de menor tamaño estructurando así los datos en forma de árbol, como por ejemplo, dividir el espacio en celdas de un tamaño determinado y altura infinita (Filin y Pfeifer 2005; Sánchez 2006), la división del espacio de forma recursiva en ocho octantes como con el uso de la estructura "octree" (Wang y Yi 2010), o la disposición de los puntos en un espacio euclídio de k dimensiones, tal y como consigue el sistema de estructuración kd-tree (Lari et al. 2011; Kwak et al. 2012; Lari y Habib 2012). Dado que las situaciones en las que no va a ser posible utilizar la triangulación de Delaunay van a ser muy escasas, finalmente se opta por utilizar la división del espacio en celdas de igual tamaño por tratarse del sistema más simple de los tres propuestos. Por consiguiente, se desarrollará un sistema de organización doble, implementando la triangulación de Delaunay para ser utilizada cuando es necesario obtener los puntos LiDAR más próximos a uno dado de forma directa, y el sistema de celdas cuando la triangulación como base para la organización no resulte efectiva.

Con respecto a la triangulación de Delaunay, existen varios algoritmos para generar la triangulación. De entre todos ellos se opta por un algoritmo de construcción incremental, tomando como base para la implementación la librería de distribución libre *Geompack* que puede descargarse de la página web de John Burkardt (Burkardt s.f). Además, el algoritmo de triangulación que contiene dicha librería deberá modificarse para adaptarlo a las características de los datos, y solucionar algunos problemas como el almacenamiento de ficheros de gran tamaño y la generación de triángulos con lados de excesiva longitud en los límites de la zona de estudio. Este último problema ya fue mencionado por Filin y Pfeifer (2005), en cuyo estudio señalan el inconveniente que tiene la triangulación en los límites de la nube de puntos LiDAR, ya que en estas zonas se generan triángulos con lados de gran longitud que poseen un área relativamente pequeña. En estos casos, como el límite de la triangulación de Delaunay define un polígono convexo, puntos muy alejados entre sí pueden estar conectados mediante aristas y por tanto, la organización de datos en triangulación no determina de forma eficiente el vecindario más cercano a un punto dado. Todo ello puede observarse en la Figura 15.

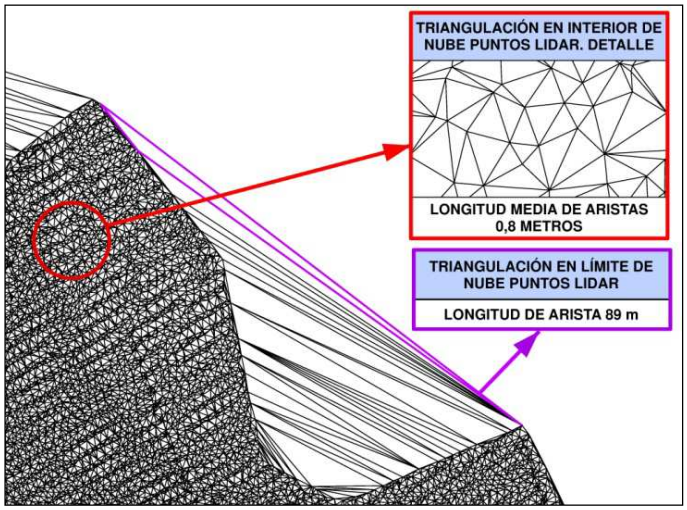


Figura 15. Longitud de las aristas en el límite de la nube de puntos

Además, algunos algoritmos que serán desarrollados en posteriores apartados (Apartado III. 4, Apartado III. 5 y Algoritmo auxiliar. Apartado III.10.2.) precisan el polígono de la zona límite de estudio. Por tanto, en esta fase de codificación de datos será necesario desarrollar un método para detectar y extraer de forma automática el contorno de la zona de estudio en forma de polígono, donde los diferentes vértices del polígono estén ordenados en el sentido de las agujas del reloj.

Como ya se ha comentado, existen distintos algoritmos para calcular la triangulación de Delaunay, los más relevantes son los que se describen a continuación.

- **Fuerza bruta**

Es el algoritmo más simple para calcular la triangulación de Delaunay de una nube de puntos. No obstante, es uno de los que mayor tiempo de cómputo consume. Básicamente consiste en calcular todas las posibles triangulaciones descartando aquellas que no cumplen los criterios de la triangulación de Delaunay. En cualquier caso, al tener que calcular todas las posibles triangulaciones, el tiempo de cómputo es demasiado alto cuando se trata de grandes nubes de puntos, siendo inviable este algoritmo para calcular la triangulación de densas nubes de datos LiDAR.

- **Algoritmo Flipping**

El algoritmo se basa en la sucesiva comprobación de la propiedad de Delaunay del círculo vacío, que especifica que la circunferencia circunscrita a cada uno de los triángulos no puede contener en su interior a ningún otro punto de la nube. Básicamente el algoritmo genera una triangulación arbitraria y posteriormente la convierte en una triangulación de Delaunay mediante la modificación de las aristas de los triángulos. La modificación o intercambio de aristas (legalización de aristas) se produce cuando existen aristas de triángulos que no cumplen con la propiedad de Delaunay (aristas ilegales). Esta operación de intercambio de aristas se denomina flip, de ahí el nombre del algoritmo.

Básicamente, el algoritmo tras generar una primera triangulación aleatoria recorre la totalidad de los vértices comprobando para cada uno de ellos si está contenido en una arista ilegal. En caso afirmativo se modifica la aristas para legalizarla y que cumpla la propiedad de Delaunay. Gráficamente el proceso de legalización de aristas puede observarse en la Figura 16. Para cada arista se comprobarán las circunferencias circunscritas a los dos triángulos adyacentes (triángulo A y B), de forma que si alguno de los vértices está dentro de dichas circunferencias tendremos una arista ilegal. En la Figura 16 la circunferencia circunscrita al triángulo A contiene un vértice perteneciente al triángulo B. Por tanto esta arista se considera ilegal y se legaliza modificando sus vértices extremos (flip).

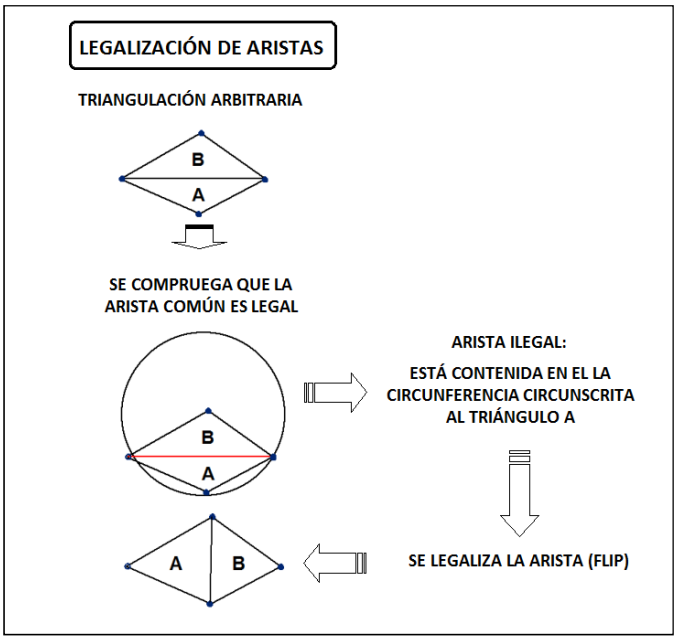


Figura 16. Proceso de legalización de aristas

- **Algoritmo de construcción incremental**

Este algoritmo no parte de una triangulación previa como en el caso del algoritmo Flipping, si no que la triangulación se construye de forma secuencial mediante la adición de nuevos vértices. Cada vez que un nuevo vértice es añadido a la triangulación, la red entera se testea y corrige si es necesario para que la totalidad de los vértices cumplan la condición de Delaunay.

Existen varias posibilidades para seleccionar el vértice siguiente: incidentalmente, ordenado por una coordenada o usando un árbol. La selección del vértice siguiente tiene gran influencia en el tiempo de ejecución del algoritmo.

Los pasos que realiza el algoritmo son:

Paso 1: Seleccionar un punto de la nube aún no triangulado.

Paso 2: Averiguar dentro de qué triángulo (ficticio o no) se encuentra ese punto.

Paso 3: Trazar una arista desde el punto a triangular a cada uno de los tres vértices del triángulo que lo contiene (dado que las aristas del triángulo son los puntos más cercanos al introducido).

Paso 4: Averiguar si es legal cada una de las tres aristas del triángulo (ficticio o no) que contiene al punto.

Paso 5: Si alguna arista es ilegal, realizar un flip y comprobar la legalidad de las otras dos aristas del triangulo.

Paso 6: Descartar las aristas de la triangulación en cuyos extremos se encuentren vértices del triángulo ficticio inicial.

El proceso se puede observar gráficamente en la siguiente Figura 17. Una vez es añadido un nuevo vértice a la triangulación se averigua en que triángulo está contenido (Figura17a),

seguidamente se trazan aristas a dicho punto desde los vértices del triángulo que lo contiene generando tres nuevos triángulos (Figura17b), por último se comprueba si las nuevas aristas de la triangulación son legales (Figura17c).

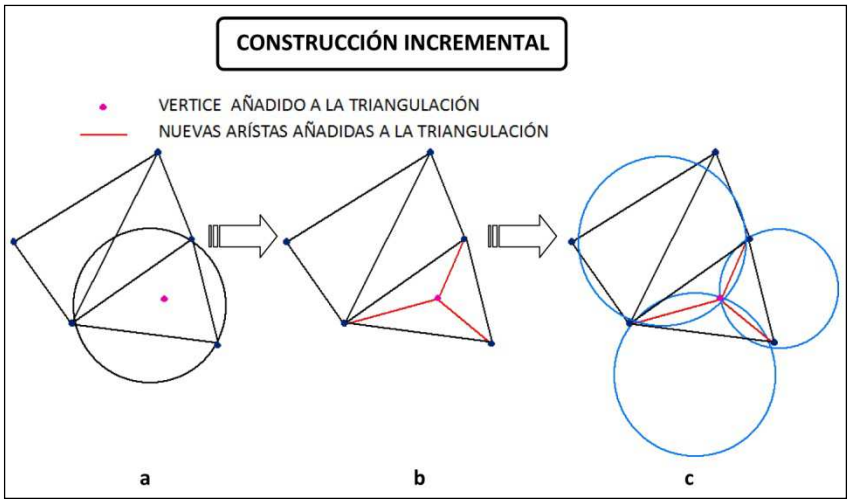


Figura 17. Proceso de legalización de aristas

Este será el algoritmo que se utilizará, para ello se tomará como base la librería de distribución libre *Geompack* que puede descargarse de la página web de John Burkardt (Burkardt s.f).

- **Sweepline**

El algoritmo *sweepline* (recorra la línea) utiliza un enfoque similar al de la construcción incremental, el algoritmo construye una pequeña parte de la triangulación final y posteriormente añade vértices paulatinamente hasta completar la triangulación. La diferencia radica en que no hay que corregir error alguno.

- **Divide y vencerás**

El algoritmo resuelve el problema de la triangulación de forma recursiva, dividiendo la nube de puntos y calculando triangulaciones independientes que finalmente son fusionadas para obtener la triangulación de Delaunay definitiva.

Generalmente, los puntos son ordenados lexicográficamente, primero a partir de la coordenada X y posteriormente a partir de la coordenada Y en caso de haber puntos con idéntica coordenada X. Seguidamente se divide la zona de estudio en partes iguales y se calcula la triangulación de Delaunay de cada una de ellas. Finalmente, las distintas triangulaciones se fusionan obteniendo la triangulación de todo el conjunto.

III.1.2. Descripción del algoritmo programado

III.1.2.1. Importación de datos

Los datos LiDAR pueden presentarse en muy diversos formatos tales como ficheros ASCII o archivos .LAS. La aplicación desarrollada permitirá la carga y posterior análisis de datos LiDAR en fichero de texto (ASCII), donde la diferente información esté tabulada en columnas. Dado que la mayoría de algoritmos que se desarrollarán en posteriores apartados únicamente utilizan la información tridimensional (X,Y,Z) asociada a los pulsos LiDAR, existirá en los archivos originales información que no será utilizada en los cálculos como por ejemplo el ángulo de giro del espejo oscilante, el número de pasada del avión, la hora de

captura del pulso, etc. Por consiguiente, con el fin de no almacenar información sin utilidad en la memoria del computador, el software desarrollado permite al usuario cargar únicamente la información necesaria, de tal forma, que al iniciar la importación de datos la aplicación preguntará al usuario el número de columnas correspondientes a las coordenadas X,Y,Z de los datos LiDAR (Figura 18). Las coordenadas de los puntos LiDAR se almacenarán en una matriz de tres columnas (X,Y,Z) e igual al número de filas que el número total de puntos LiDAR.

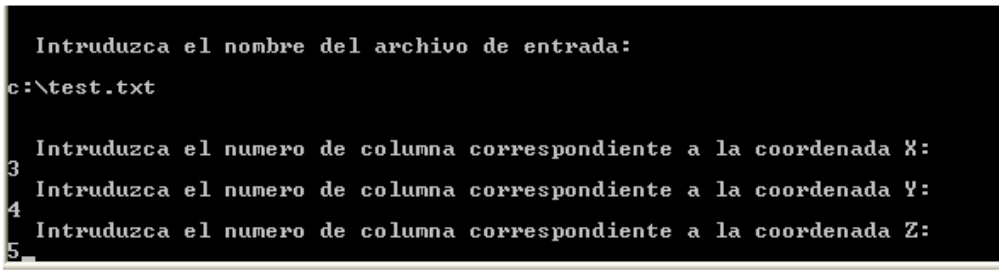


Figura 18. Importación de coordenadas de puntos LiDAR.

Debido a la alta densidad de las nubes de puntos y a la superposición de huellas de pulsos láser, en ocasiones es posible encontrar registros duplicados o puntos con idénticas coordenadas en el fichero de datos LiDAR original. En esta parte del proceso, será necesario detectar los puntos que están duplicados, ya que de lo contrario pueden darse errores en tiempo de ejecución en otros algoritmos, como es el caso de la triangulación de Delaunay.

Para eliminar los puntos duplicados, la aplicación ordena la matriz de coordenadas primero en función del valor de la coordenada X y después en función de la coordenada Y. De esta forma, dos puntos que poseen coordenadas idénticas quedarán dispuestos en la matriz resultante en filas consecutivas. A la vez que la información asociada a los puntos LiDAR es ordenada y los registros cambian de posición, un vector de igual dimensión que el número total de puntos LiDAR se irá actualizando. Este vector consiste en un vector de apuntadores. Se trata de un sistema para poder acceder a la información original en cualquier momento del procesado de datos. El problema consiste en que tras ordenar la matriz de coordenadas ya no existirá una relación entre el registro de un punto en el fichero original y su posición en la matriz. Con el fin de poder acceder a cualquier información del archivo original que no ha sido cargado en memoria, como por ejemplo el nivel de intensidad de la señal reflejada o información multi-retorno, la aplicación genera un índice que consiste en un vector con punteros que correlaciona los registros de la matriz de datos y el archivo origen.

Para ordenar la matriz de coordenadas se utilizan las funciones de libre distribución *r82vec_sort_heap_index_a* y *r82vec_permute* obtenidas de la página de John Burkardt y que están contenidas en la librería *GEOMETRY*. El algoritmo de ordenación sigue el método de la burbuja. Este método consiste en recorrer la lista iterativamente comparando elementos adyacentes de dos en dos. Si un elemento es mayor que el que está en la siguiente posición se intercambian. El proceso se repite hasta que ninguna posición es modificada, momento en el que la ordenación se ha completado. El algoritmo no modifica por sí mismo la matriz de coordenadas, si no que genera un vector de apuntadores (índice 1) que posteriormente es utilizado para la ordenación. Una vez el algoritmo ha ordenado las coordenadas de los puntos, se recorre la matriz de coordenadas comparando los elementos adyacentes de dos en dos, de tal forma que si las coordenadas de dos registros consecutivos son idénticos es debido a que un punto LiDAR está duplicado, y por tanto se procede a la eliminación de uno de ellos. Todo el proceso puede apreciarse en la Figura 19.

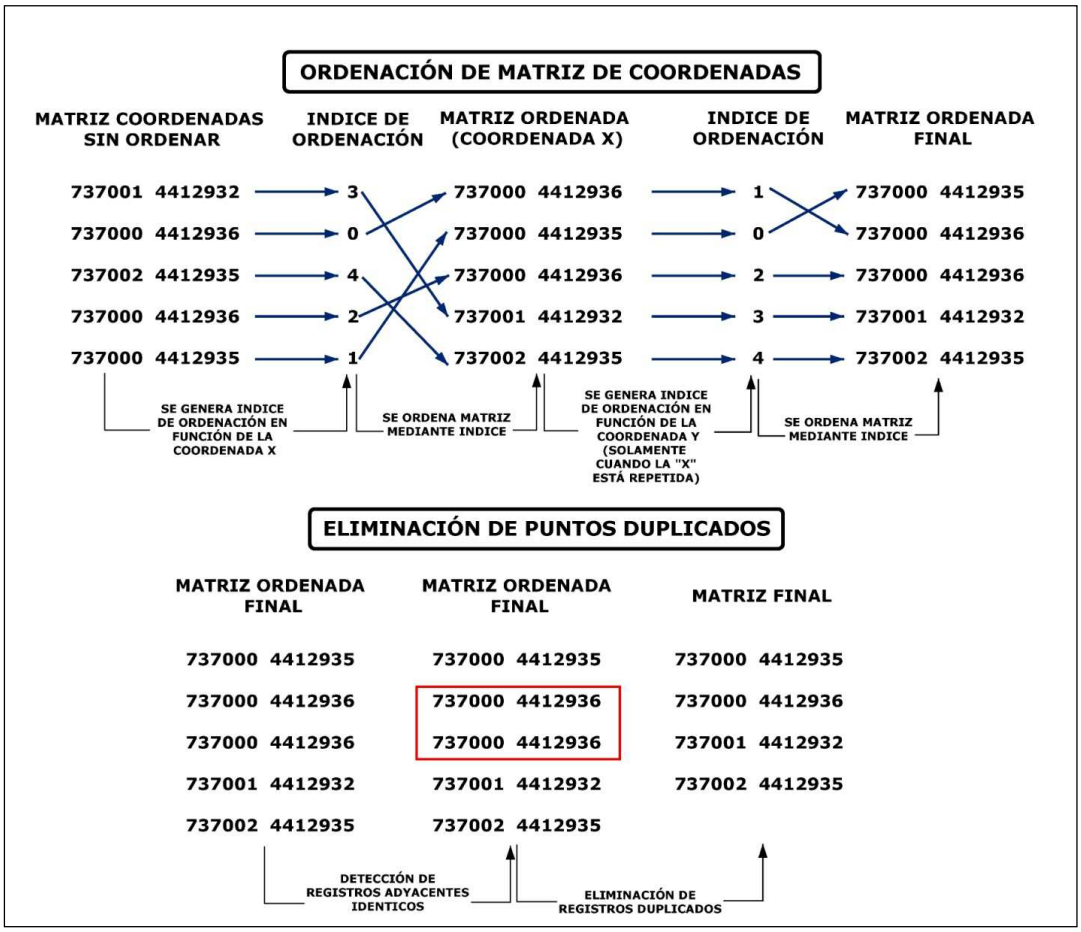


Figura 19. Eliminación de información duplicada. Importación de datos

En la figura anterior se aprecia el proceso para identificar puntos duplicados entre un total de cinco puntos almacenados en una matriz de coordenadas. Tras generar un índice de ordenación a partir de la coordenada X de la matriz inicial, dicho índice es utilizado para reordenar la matriz. Seguidamente se genera un segundo índice de ordenación, esta vez a partir de la coordenada Y considerando solamente aquellos puntos donde la coordenada X es idéntica, finalmente se ordena la matriz obtenida en el paso previo con este segundo índice obteniendo la matriz de coordenadas final ordenada. El proceso de eliminación de puntos duplicados continúa mediante la detección de registros idénticos que estarán dispuestos de forma consecutiva en la matriz. Finalmente se elimina la información redundante detectada, obteniendo la matriz final de coordenadas.

La importación de datos finaliza mediante la impresión de tres ficheros diferentes: un fichero con la información tridimensional de los puntos (coordenadas X,Y,Z), un fichero con la información planimétrica de los mismos (coordenadas X,Y) y un índice donde sus elementos relacionarán las coordenadas de cada punto de los dos archivos anteriores con su registro en el fichero original de datos LiDAR, que servirá para poder acceder de forma directa a cualquier información adicional contenida en el fichero origen en cualquier momento del procesado de datos. El esquema completo del proceso de importación de datos puede observarse a continuación (Figura 20).

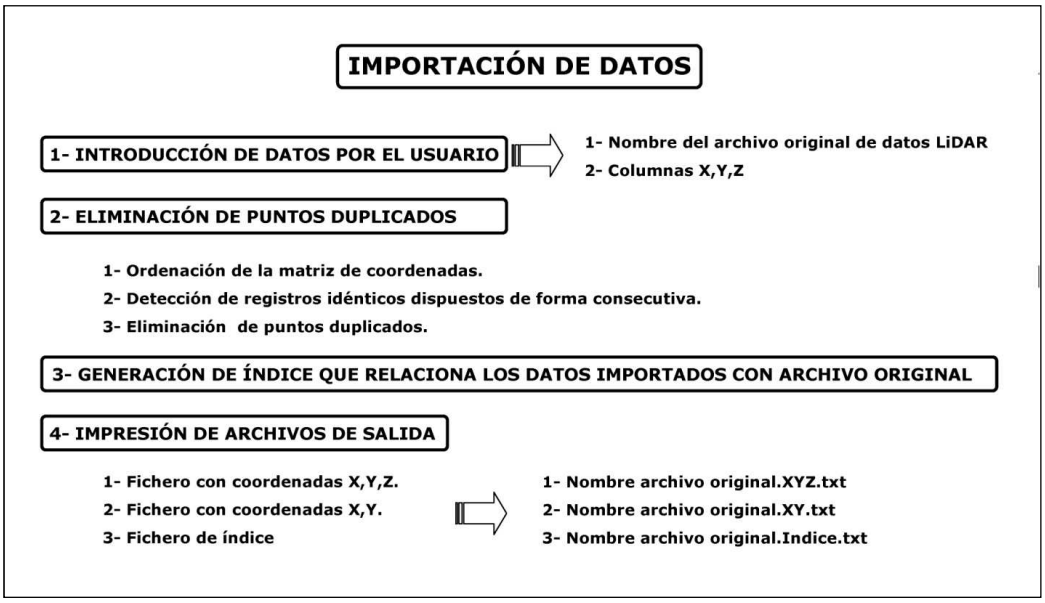


Figura 20. Esquema del proceso de importación.

III.1.2.2. Organización de datos

La organización de los datos brutos LiDAR en estructuras que permitan obtener el vecindario de forma eficiente es un aspecto fundamental en el post-procesado de datos. En este sentido, tal y como se ha comentado en apartados anteriores, se desarrollará un sistema de organización doble que permita obtener el vecindario de un punto perteneciente al conjunto de datos LiDAR, así como de cualquier otro punto externo no contenido en dicho conjunto. Esta doble codificación se resuelve mediante la implementación de la triangulación de Delaunay y la partición del espacio en subconjuntos de menor tamaño estructurando los datos en celdas de tamaño fijo y altura infinita.

Mientras que la organización de los datos a partir de la triangulación de Delaunay requiere un coste computacional alto, la división del espacio en celdas es un proceso sencillo y rápido. Por este motivo, la codificación en forma de triangulación se realizará en el proceso de importación y los resultados obtenidos serán almacenados en ficheros para ser utilizados con posterioridad por el resto de algoritmos. Por el contrario, la organización de los datos en celdas será realizada por el algoritmo que así lo requiera en tiempo de ejecución, sin necesidad de cargar o almacenar ningún fichero de datos adicional. A continuación se describen ambos algoritmos implementados.

- **Organización mediante partición del espacio en celdas de igual tamaño y altura infinita**

Este sistema se basa en dividir la zona de estudio en celdas de igual tamaño y clasificar los puntos en función de la celda a la que pertenecen, sin tener en cuenta en ningún caso la información altimétrica. Se trata de relacionar el identificador de cada punto con su celda asociada, de forma que cada celda tenga asociado un vector con los identificadores de los puntos LiDAR que contiene.

Tal y como se puede observar en la imagen siguiente (Figura 21), la zona de estudio se dividirá en celdas de igual dimensión y altura infinita. Las celdas constituirán una matriz donde se almacenarán los identificadores de los puntos LiDAR. En el ejemplo, la zona de estudio se ha dividido en un total de 20 celdas donde cada celda se identifica por dos índices, el primero hace referencia a la fila y el segundo a la columna. Mediante este

sistema por ejemplo, la celda "C 2-3" estará situada en la fila 2 y columna 3. La primera de las filas será la fila cero, y lo mismo ocurre con las columnas.

En la imagen (figura 21a) puede apreciarse la división de la zona de estudio en las diferentes celdas que constituirán la matriz de organización. En la imagen de detalle (Figura 21b) se aprecian los identificadores de puntos que están contenidos en cuatro de las celdas. Finalmente, el sistema de organización asocia cada celda con un vector compuesto por los identificadores de los puntos contenidos en cada una de ellas (Figura 21c), donde el tamaño de cada vector será variable y vendrá indicado en el primer elemento del vector.

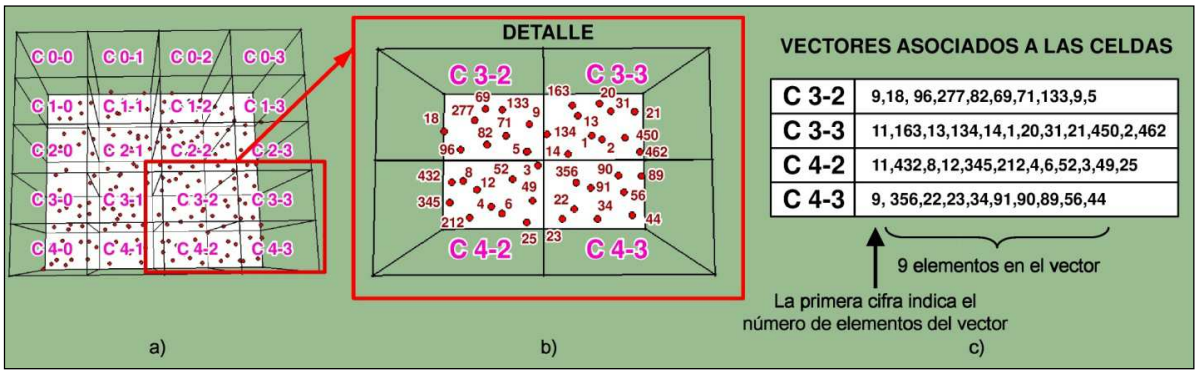


Figura 21. Organización de datos mediante la partición del espacio en celdas.

El sistema de organización se desarrolla como un arreglo bidimensional que queda definido a partir de un doble puntero. El primer puntero apunta a la dirección de memoria donde está la celda o cuadrante en cuestión, mientras que el segundo apunta a la posición del vector que almacena los diferentes puntos contenidos en dicha celda. En programación, un arreglo se define como un conjunto de datos que son almacenados en memoria de manera consecutiva con un mismo nombre. Para diferenciar los diferentes elementos del arreglo se utilizan índices.

De forma detallada, los diferentes pasos y operaciones que realiza el algoritmo para crear el sistema de organización son los siguientes:

Cálculos previos:

Para generar la codificación será necesario que el usuario introduzca la longitud de los lados de las celdas que dividirán el área de estudio. Esta información será almacenada en una variable denominada *Tamaño_Celda*. Si el usuario no introduce ningún valor por defecto se dividirá el espacio en celdas cuadradas de 2,5 m de lado, al considerarse esta dimensión la más adecuada para trabajar con datos LiDAR tras realizar diferentes pruebas.

En esta primera fase el algoritmo calculará los parámetros necesarios para poder definir el arreglo que constituirá el sistema de organización. Los cálculos previos que realizará son los siguientes:

Cálculo de coordenadas máximas y mínimas de la zona de estudio:

Se recorrerá la totalidad de puntos LiDAR almacenando las coordenadas máximas y mínimas de la zona de estudio en cuatro variables (Xmin, Ymin, Xmax, Ymax).

- Cálculo de la longitud del área de estudio en el sentido de abscisas y ordenadas:

- Para determinar el número de filas y columnas de la matriz de organización será necesario calcular la distancia horizontal (Ix) y vertical(IY) total del área de estudio,

para ello se utilizarán las coordenadas máximas y mínimas obtenidas en el paso anterior.

$$Ix=Xmax-Xmin$$
$$Iy=Ymax-Ymin$$

- Cálculo del número total de filas y de columnas de la matriz de organización: A partir del tamaño de celda y de las dimensiones del área de estudio será posible establecer el número total de filas y columnas que poseerá la matriz.

$$Columnas\_Total=(Ix/Tamaño\_Celda)+1$$
$$Filas\_Total=(Iy/Tamaño\_Celda)+1$$

Declaración del arreglo de organización:

Como se ha comentado con anterioridad, el sistema vendrá constituido por un arreglo bidimensional que se definirá en tiempo de ejecución mediante reserva dinámica de memoria. El primer índice del arreglo apuntará a la localización de cada celda de la matriz dentro de la memoria, mientras que el segundo apuntará al inicio de los vectores asociados a cada celda que contienen los identificadores de puntos LiDAR. Para definir el arreglo será necesario disponer de dos datos, el número total de celdas y el tamaño de cada vector. El número total de celdas se obtiene de forma directa ya que conocemos el número total de filas y de columnas. No obstante el tamaño de cada vector es una incógnita que deberá resolverse previamente a la definición del arreglo.

Para obtener el tamaño de cada vector el algoritmo genera un vector auxiliar de contadores de igual tamaño que el número total de celdas, el vector se reiniciará con el valor cero en todas sus posiciones y posteriormente se recorrerán todos los puntos LiDAR detectando la celda a la que pertenece cada uno de ellos e incrementando el registro asociado a dicha celda en el vector auxiliar. De esta forma al finalizar el proceso dispondremos de un vector donde cada elemento indicará el número de puntos LiDAR que contiene cada celda. Para determinar la celda a la que pertenece cada punto LiDAR se realizarán los siguientes cálculos:

$$Ix\_Punto=X\_Punto-Xmin;$$
$$Iy\_Punto=Ymax-Y\_Punto$$
$$Columna\_Punto=Ix\_Punto/\ Tamaño\_Celda;$$
$$Fila\_Punto=Iy\_Punto/Tamaño\_Celda;$$

A partir de la fila y columna de cada punto se actualizará el vector auxiliar de contadores mediante el acceso a la posición de la celda utilizando la siguiente expresión:

$$Posicion=\ Columna\_Punto+Fila\_Punto*\ Columnas\_Total$$
$$Vector\_Auxiliar[Posicion]=Vector\ Vector\_Auxiliar[Posicion]+1$$

Tras finalizar el proceso ya se dispondrá de la información necesaria para definir el arreglo bidimensional. Además, antes de destruir el vector auxiliar se copiará su información en el primer elemento del vector de puntos asociado a cada celda.

Generación del arreglo de organización:

Una vez definido el arreglo deberá rellenarse el vector asociado a cada celda con los identificadores de los puntos que contiene cada una de ellas. El proceso será análogo al realizado en el paso anterior, se recorrerán la totalidad de puntos LiDAR calculando la fila y la columna de la celda a la que pertenecen, posteriormente mediante dicha información se accederá a su vector correspondiente añadiendo al mismo el identificador asociado al punto LiDAR. Finalmente dispondremos de tantos vectores como celdas dividen el espacio. Cada

vector tendrá una longitud igual al número de puntos contenidos en cada celda siendo el primero de los elementos el número total de elementos que constituyen el vector (Figura 21c).

- **Organización mediante triangulación de Delaunay**

La generación de la triangulación de Delaunay en densas nubes de puntos lleva asociada un coste computacional alto. Es por ello que la triangulación se generará en el proceso de importación y el resultado será impreso en un fichero que será utilizado por el resto de algoritmos cuando sea necesario.

Para desarrollar esta aplicación se han utilizado diversas funciones de distribución libres contenidas en la librería *Geompack* que pueden descargarse desde la página web de John Burkardt (Burkardt s.f). El programa, tras cargar el archivo con las coordenadas de los puntos LIDAR genera la triangulación de Delaunay mediante construcción incremental e imprime un archivo de texto que contiene los vértices pertenecientes a cada triangulo.

El fichero de texto con la triangulación consistirá en una matriz de igual número de filas que el número total de triángulos existentes en la red y tres columnas que se corresponderán con los identificadores de puntos LiDAR que constituyen los vértices de cada triángulo. De esta forma se podrá acceder de forma directa a las coordenadas de los vértices de cada triángulo que se encontrarán en el fichero de coordenadas de puntos LiDAR.

Como ejemplo, tal y como se puede observar en la siguiente imagen (Figura 22), el triángulo nº 3 de la red de triángulos estará definido en la tercera fila del fichero de triangulación, donde vendrán los identificadores de los puntos LiDAR que constituyen los vértices de dicho triángulo (puntos 4, 6 y 8). Por último, para obtener las coordenadas de dichos vértices únicamente habrá que acceder a las filas correspondientes dentro del fichero de coordenadas de puntos LiDAR.

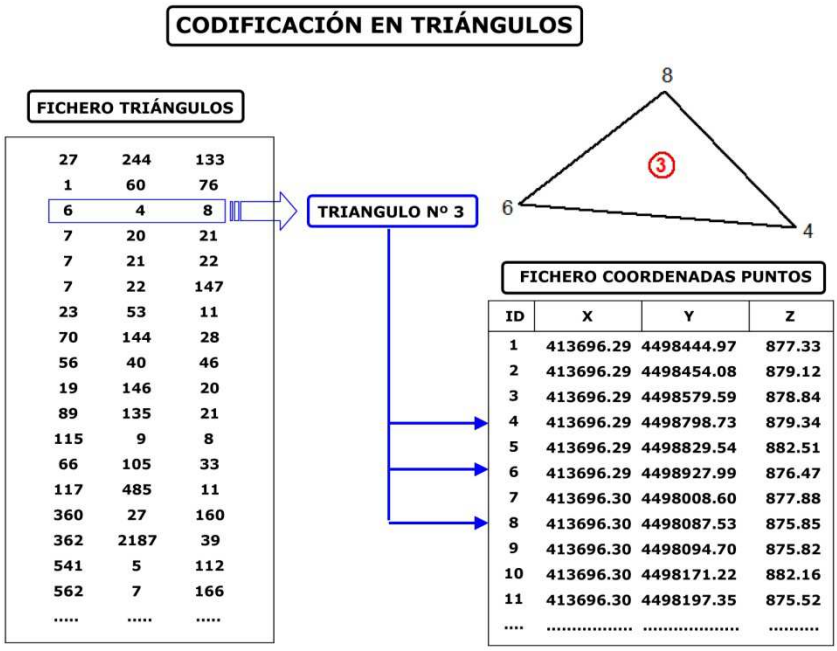


Figura 22. Organización de datos mediante triangulación de Delaunay

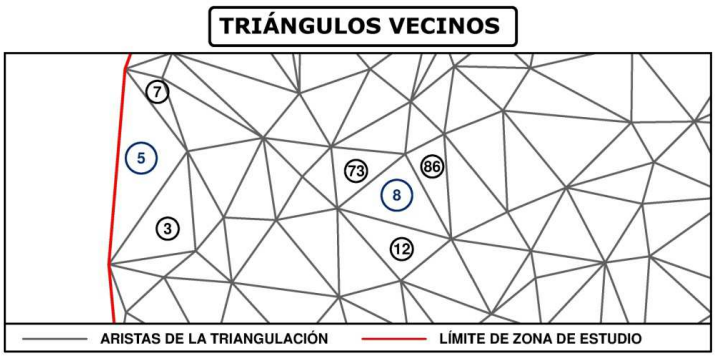
Además de generar el fichero con los vértices correspondientes a cada triángulo, se generará un archivo que contendrá los identificadores de triángulo colindantes a cada uno de los triángulos de la triangulación. Esto será de gran utilizada para detectar los triángulos

que pertenecen al límite del área de estudio así como para extraer de forma automática el polígono convexo que contiene la totalidad de puntos LiDAR.

El archivo con los triángulos vecinos se generará mediante el uso de la función de libre distribución *dtris2* que puede descargarse de la página web de John Burkardt (Burkardt s.f). El resultado es una matriz de igual número de filas que el número total de triángulos existentes en la red de triángulos y tres columnas, donde vendrán especificados los tres triángulos colindantes a cada triángulo dado.

Los identificadores de los triángulos colindantes a uno dado aparecerán en las tres columnas asociadas a dicho triángulo. En el caso de que el triángulo en cuestión pertenezca al límite de la zona de estudio no tendrá tres triángulos colindantes y por tanto podrá detectarse fácilmente. En estos casos, el triángulo tendrá uno o dos triángulos colindantes cuyos identificadores aparecerán en las distintas columnas de su registro asociado. El resto de columnas se aparecerán con el valor -1.

Según el ejemplo que puede observarse en la Figura 23, el triángulo nº 8 pertenece al interior del área de estudio y tiene tres triángulos colindantes que comparten aristas con él. Los identificadores de los tres triángulos están registrados en las columnas asociadas en el fichero de triángulos vecinos. Por otro lado, el triángulo nº 5 está situado en el límite de la zona de estudio y por tanto solamente tiene dos triángulos colindantes, que están registrados de igual modo en dos de las tres columnas asociadas a dicho triángulo en el archivo de triángulos vecinos. La columna restante registrará el valor "-1", de tal forma que para detectar los triángulos que pertenecen al límite de la zona de estudio, únicamente deberán detectarse aquellos triángulos que contienen un "-1" en alguno de sus registros asociados en el fichero de triángulos vecinos.



FICHERO DE TRIÁNGULOS VECINOS

	ID	ADYACENTE 1	ADYACENTE 2	ADYACENTE 3
	1	32323	6	3456
	2	1234	399	5
	3	2488	2	5552
	4	9832	544	221
TRIÁNGULO 5	5	-1	7	3
	6	4439	669	1559
	7	18	47	322
TRIÁNGULO 8	8	73	86	12
	....	.....	.....	.....

Figura 23. Organización de datos mediante triangulación de Delaunay

La detección de los triángulos pertenecientes al límite de la zona de estudio es esencial para conseguir extraer de forma automática el contorno límite del área de estudio y para solucionar el problema de triángulos anómalos con lados de longitud excesiva.

Tal y como se ha comentado con anterioridad, en los límites de la triangulación de Delaunay se generan triángulos con lados que poseen una gran longitud y que por tanto conectan puntos que están muy alejados entre sí. Ésto influye directamente en el sistema de organización de los datos impidiendo la detección del vecindario de un punto de forma eficiente. Para solucionar este problema, tras generar la triangulación de Delaunay el algoritmo busca todos los triángulos que pertenecen al límite de estudio, eliminando de la triangulación aquellos que tienen lados con longitud excesiva. La longitud límite para eliminar un triángulo será de 4 m por defecto aunque podrá ser modificada por el usuario. El proceso para eliminar los triángulos anómalos se describe a continuación.

Detección de triángulos pertenecientes al límite de la zona de estudio:

Se recorre la matriz de triángulos vecinos detectando aquellos triángulos que contengan el valor "-1" en alguna de sus columnas asociadas.

Detección y eliminación de triángulos anómalos:

Para cada uno de los triángulos colindantes al límite del área de estudio se calcularán las longitudes de sus lados, de forma que si alguno de ellos tiene una longitud mayor al parámetro umbral en longitud previamente establecido será eliminado de la matriz de triángulos y de la matriz de triángulos vecinos.

Actualización de la información asociada a los triángulos colindantes:

Cuando un triángulo es eliminado, los triángulos adyacentes que no eran colindantes al área de estudio pasarán automáticamente a serlo. Por tanto, deberán actualizarse sus registros asociados en la matriz de triángulos vecinos cambiando el identificador del triángulo eliminado por el valor "-1".

El proceso completo de eliminación de triángulos anómalos se aprecia en la Figura 24. Tras detectar un triángulo que tiene el valor "-1" en su matriz de triángulos vecinos y que por tanto pertenece al límite de la zona de estudio (triángulo 234), se comprueba que alguno de sus lados es mayor al parámetro umbral establecido, momento en el cual el triángulo es eliminado y sus triángulos adyacentes son marcados con "-1" en la matriz de triángulos vecinos. El proceso es iterativo, de forma que se comprueba si los lados del triángulo adyacente al anterior (triángulo 236) cumplen con el parámetro umbral en longitud. Dado que este no es el caso, el triángulo también es eliminado y su triángulo adyacente (triángulo 237) es marcado con "-1" en la matriz de triángulos vecinos. En este último caso los lados del triángulo también son mayores al parámetro umbral en longitud, por lo que de nuevo es eliminado y sus triángulos adyacentes marcados con "-1". Llegado a este punto el proceso iterativo finaliza, ya que los nuevos triángulos marcados en la matriz de triángulos vecinos tienen lados que cumplen con el parámetro umbral y por tanto no son eliminados.

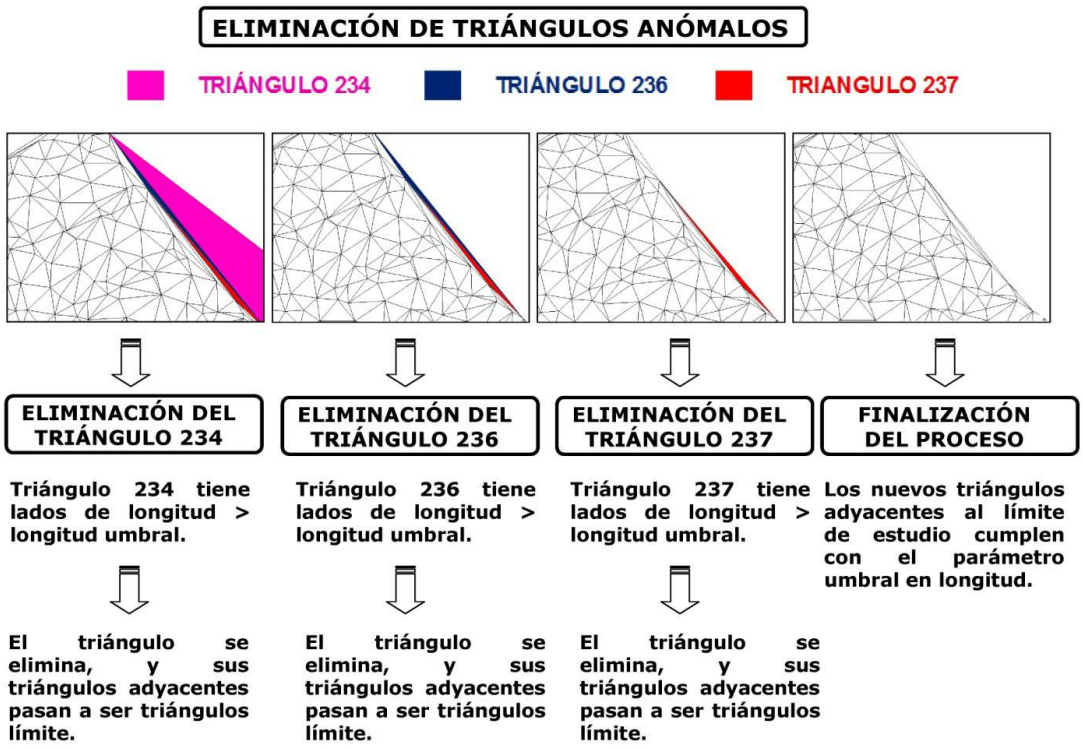


Figura 24. Eliminación de triángulos anómalos en el límite de zona de estudio.

Mediante este sistema se consigue eliminar de forma eficiente los triángulos anómalos que la triangulación de Delaunay genera en la zona límite de la nube de puntos (Figura 25), obteniendo así una triangulación refinada donde cada punto está conectado mediante aristas a sus puntos más próximos, constituyendo un sistema de organización idóneo para determinar de forma eficiente el vecindario próximo de un punto.

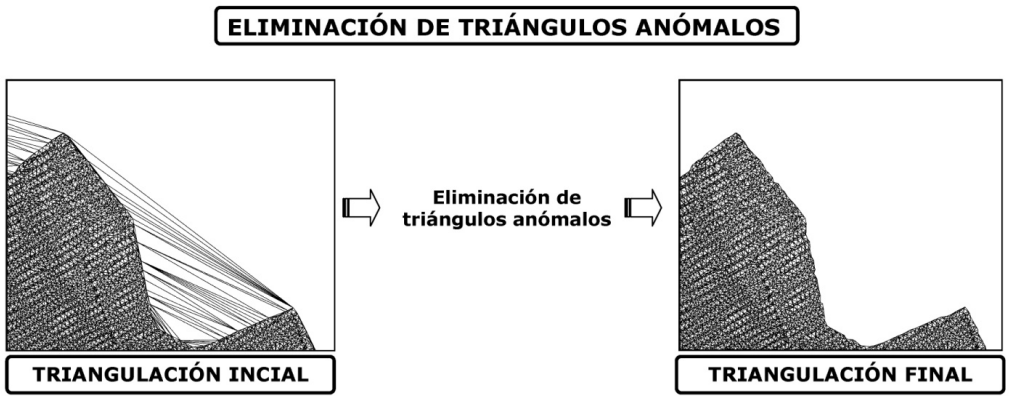


Figura 25. Eliminación de triángulos anómalos. Datos test 1.

- **Extracción automática del polígono límite que contiene el área de estudio**

Los diferentes algoritmos que serán desarrollados en apartados posteriores, necesitan como dato de entrada el polígono límite que contiene a la totalidad de puntos LiDAR pertenecientes al área de estudio. Es por ello, que en este proceso de importación y organización de datos será necesario desarrollar un algoritmo que extraiga de forma automática el contorno límite de la zona de estudio. Este algoritmo toma como punto de

partida la triangulación de Delaunay refinada obtenida tras eliminar los triángulos anómalos (tal y como se ha expuesto con anterioridad).

Principalmente, el algoritmo se basa en detectar las aristas de la triangulación pertenecientes al límite de la nube de puntos. Para ello tiene en cuenta que cualquier arista que no pertenece al contorno límite es una arista que está definida en dos triángulos adyacentes y por tanto está duplicada en la triangulación, mientras que las aristas que pertenecen al contorno límite únicamente están definidas en un triángulo y por tanto no están duplicadas. Todo ello puede observarse en la Figura 26. En este ejemplo (Figura 26a), una de las aristas de la triangulación tiene como vértices adyacentes los puntos LiDAR con identificador 123 y 128. Esta arista es común a dos triángulos distintos, los triángulos con identificadores 4 y 6, por tanto será una arista duplicada que estará definida dentro de la matriz de triángulos en dos filas diferentes. Por el contrario, la arista que tiene como vértices adyacentes los puntos LiDAR 436 y 512 pertenece al límite de la zona de estudio y por tanto solamente está asociada a un triángulo (triángulo 8). El proceso para detectar el contorno límite pasará por detectar aquellas aristas que solamente pertenecen a un triángulo y que por tanto no están duplicadas (Figura 26b).

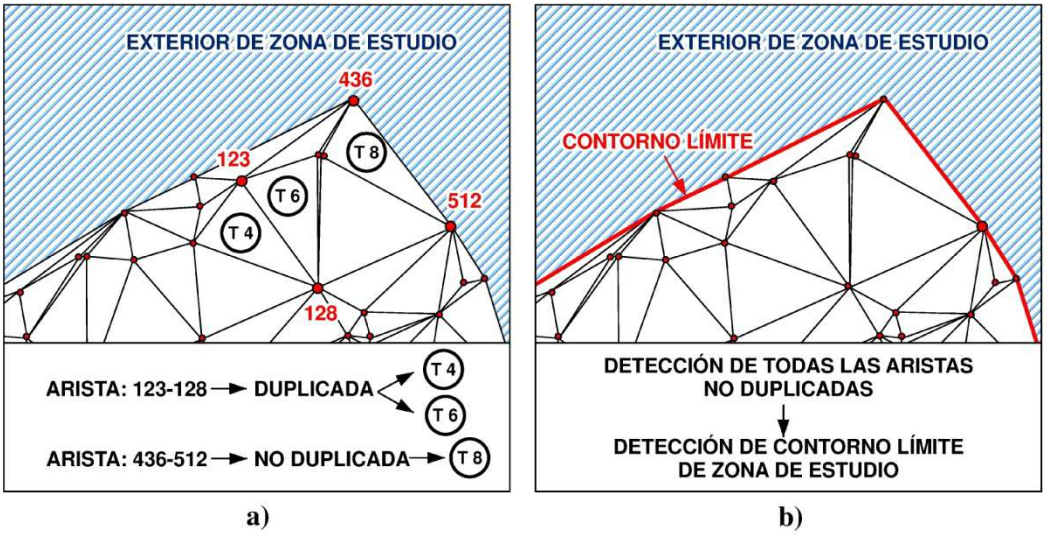


Figura 26. Detección de contorno límite

El contorno final se imprimirá en un fichero de texto donde los distintos vértices que conforman el polígono estarán ordenados en el sentido de las agujas del reloj. El proceso para detectar las aristas no duplicadas y extraer el contorno de forma correcta es el descrito a continuación.

Generación de matriz de aristas:

Para detectar las aristas que no están duplicadas, el primer paso es generar una matriz con la totalidad de aristas existentes en la triangulación. Para ello se partirá de la matriz de triángulos, donde para cada triángulo vienen los identificadores de los puntos LiDAR que constituyen cada uno de sus tres vértices. En la matriz de triángulos aparecen de forma implícita las aristas que constituyen cada uno de ellos. Por tanto, únicamente deberá utilizarse esa información para generar una matriz de aristas de dos columnas, una para cada vértice. Tal y como se observa en el ejemplo de la Figura 27, el triángulo nº 7 está definido por los puntos LiDAR 234, 27 y 698, estos vértices definen tres aristas distintas (A1,A2 y A3) que se reflejan en tres filas diferentes de la matriz de aristas, donde cada fila contendrá los vértices adyacentes a cada una de ellas. Para generar la matriz de aristas se define una matriz de igual numero de filas que el triple del número de triángulos existentes

en la triangulación de Delaunay y con dos columnas. Posteriormente se recorre la matriz de triángulos combinando los vértices de dos en dos, definiendo así las tres aristas de cada triángulo que serán registradas de forma automática en la matriz de aristas.

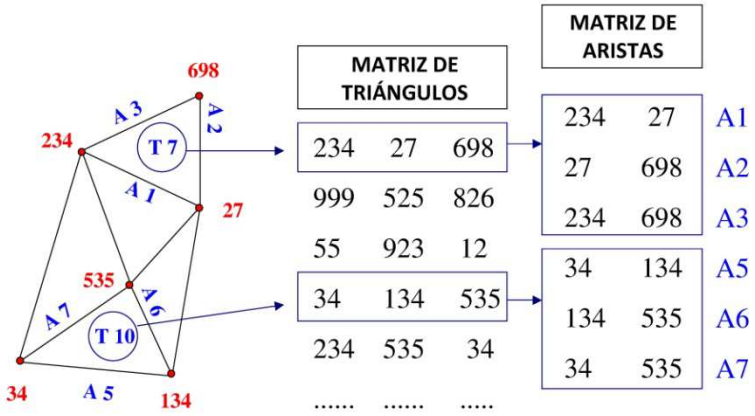


Figura 27. Generación de matriz de aristas.

Permutación de registros dentro de la matriz de aristas:

A la hora de generar la matriz de aristas puede ocurrir que dos aristas con los mismos vértices adyacentes que se corresponden con una arista duplicada queden registradas en sus dos filas correspondientes de la matriz de aristas con los vértices en diferente orden. Tomando el ejemplo de la siguiente imagen (Figura 28), el triángulo nº 7 y nº 8 tienen una arista común y por tanto se registrará por duplicado en la matriz de aristas (A1 y A6). El problema reside en que los vértices de las aristas A1 y A6 a pesar de ser los mismos se han registrado con diferente orden en la matriz de aristas, esto dificultará la búsqueda de aristas no duplicadas para la extracción automática del contorno. Dado que la posición de un vértice en una u otra columna dentro de la matriz de aristas no afecta a la definición de la misma, se procederá a reordenar los registros colocando en la primera columna siempre el vértice con mayor identificador, permutando los registros de ambas columnas cuando sea necesario. Observando de nuevo el ejemplo, se aprecia como tras realizar la permutación de las columnas, las aristas A1 y A6 ya poseen sus vértices en el mismo orden.

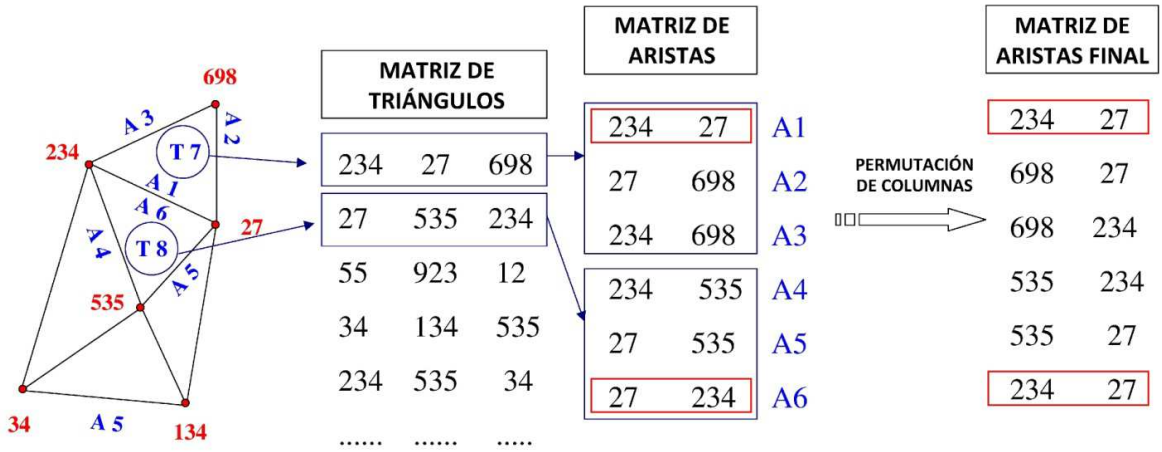


Figura 28. Permutación de registros de la matriz de aristas.

Detección de aristas no duplicadas:

Tras permutar los valores de la matriz de aristas en los casos en que sea necesario, se procederá a detectar las aristas que no están duplicadas dentro de dicha matriz, obteniendo así las aristas de la triangulación que conforman el polígono límite que envuelve a la zona de estudio. Para determinar las aristas duplicadas la aplicación ordena la matriz de aristas primero en función de la primera columna y posteriormente en función de la segunda, de forma que una arista duplicada vendrá definida en dos filas consecutivas de la matriz de forma idéntica. Finalmente el algoritmo detectará las aristas que no están duplicadas recorriendo la matriz y comparando sus filas dos a dos. La comparación se realizará comparando cada fila con su fila posterior y anterior, de forma que si no existe coincidencia de registros es debido a que la arista no está repetida y por tanto pertenece a un triángulo límite de la triangulación. Las aristas no duplicadas serán almacenadas en una nueva matriz (*Aristas_Contorno)* que contendrá los identificadores de puntos LiDAR que constituyen los vértices extremos de todas las aristas que definen el polígono límite. La detección de aristas de contorno límite y su extracción puede apreciarse en la Figura 29.

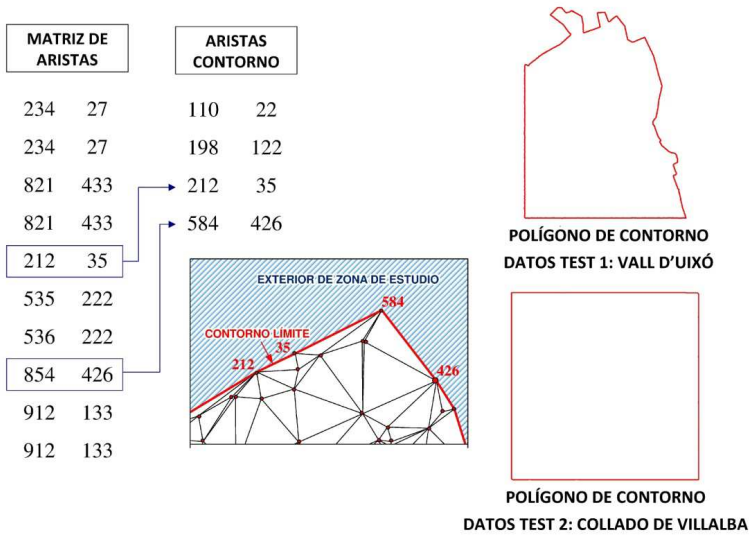


Figura 29. Detección de aristas pertenecientes al contorno límite.

Extracción del contorno en el sentido de las agujas del reloj:

La extracción del contorno límite de la nube de puntos debe de realizarse en forma de cadena mediante la impresión de los vértices que lo conforman en el sentido de las agujas del reloj. Ésto servirá para generar topología y poder utilizar el polígono en operaciones GIS tales como la intersección y recorte de polígonos que utiliza el algoritmo auxiliar de extracción de contorno de entidades (Apartado III.10.2.). Por tanto, un aspecto previo y fundamental será determinar el sentido de extracción antes de comenzar con la misma. Para ello, se tendrá en cuenta que el ángulo interior que forman dos aristas consecutivas de un polígono, siempre será mayor al que forma la segunda de las aristas con un punto interior del polígono, considerando la segunda arista la de orden posterior en el sentido de las agujas del reloj (Figura 30).

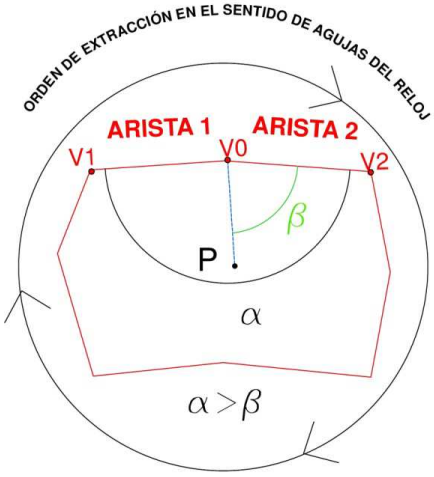


Figura 30. Propiedad para determinar sentido de extracción de vértices en el sentido de las agujas del reloj.

Tal y como se aprecia en la figura anterior, el ángulo interior que forman la arista 2 y la arista 1 (α) será siempre mayor al ángulo formado por la arista 2 y un punto cualquiera del interior del polígono (β). Esta propiedad será la utilizada por el algoritmo para extraer los vértices del contorno en el sentido de las agujas del reloj. Para ello, tras seleccionar un vértice del contorno cualquiera (V0) se accederá a partir de la matriz de aristas a las dos aristas consecutivas de las que el vértice forma parte. Seguidamente se almacenarán los vértices adyacentes a ambas aristas (V1, V2) y se calculará el ángulo (α) que forman ambas aristas mediante la diferencia de acimutes, así como el ángulo (β) que forma la arista 2 con un punto LiDAR cualquiera que no pertenezca al contorno límite (P). Llegado a este punto pueden darse dos casos, que el ángulo α calculado sea mayor que β, lo cual significará que se cumple el supuesto expresado en la Figura 30 y que por tanto el siguiente vértice del polígono a extraer es el V2 por ser el vértice adyacente a la arista 2, o bien puede suceder que el ángulo β sea mayor que α, lo que significará que la arista 2 no es la posterior a la primera en el sentido de las agujas del reloj, y por tanto el siguiente vértice a extraer en el sentido de las agujas del reloj será el V1.

Para la selección del primer vértice de extracción (V0) se tomará el primer vértice de la matriz de de aristas de contorno, mientras que el acceso a los vértices adyacentes (V1,V2) será directo mediante la utilización de dicha matriz. El principal problema radica en diferenciar el siguiente vértice del polígono en el sentido de las agujas del reloj entre los vértices V1 y V2. Para ello, tal y como se ha comentado, el algoritmo aplicará la siguiente regla de decisión tras el cálculo de ángulos:

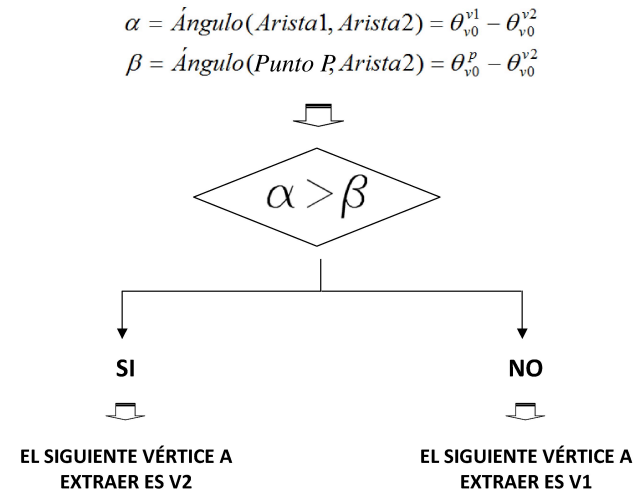


Figura 31. Regla de decisión para la extracción de vértices en el sentido de las agujas del reloj.

Esta regla de decisión puede aplicarse independientemente de la orientación de la arista de contorno, obteniendo de forma sencilla y eficaz el segundo vértice de extracción. En la Figura 32 se observa la aplicación de dicha regla de decisión para distintos aristas con diferente orientación con respecto a la nube de puntos LiDAR.

En cualquier caso, la utilización de la regla de decisión solamente es necesaria para la extracción del segundo vértice del polígono, ya que tras extraer los dos primeros vértices, dichos vértices son marcados en un vector de marcadores permitiendo así la extracción de forma directa del resto de vértices en el sentido de las agujas del reloj sin necesidad de hacer cálculos adicionales. Para ello, únicamente deberá accederse a las aristas que contienen el último vértice extraído, tomando seguidamente como siguiente vértice aquél que no haya sido marcado con anterioridad en el vector de marcadores. El proceso de extracción y marcado es iterativo, finalizando el proceso cuando todos los vértices pertenecientes a las aristas de contorno han sido marcados. Este proceso permite extraer el contorno en el sentido de las agujas del reloj de forma directa y sin realizar cálculos ni búsquedas (a excepción de la extracción del segundo vértice). Como consecuencia el algoritmo obtiene el resultado de forma inmediata incluso cuando se trata de polígonos con más de 2.000 vértices.

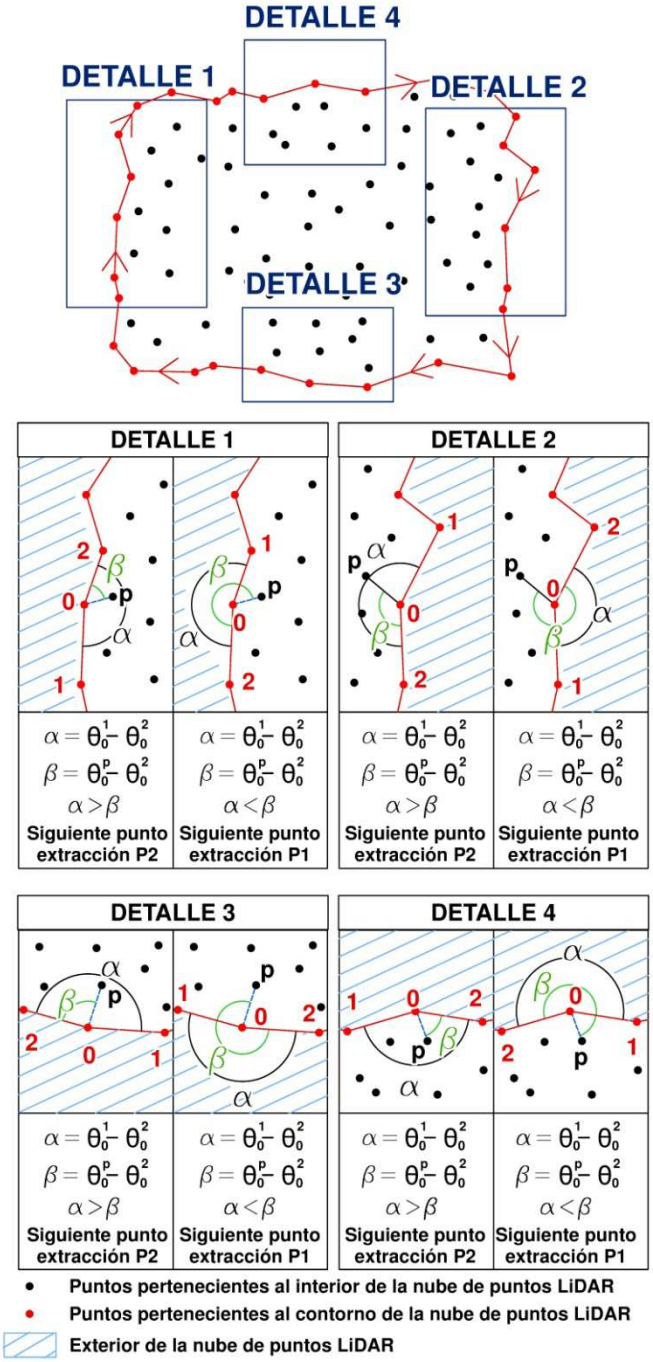


Figura 32. Extracción del segundo vértice en sentido de las agujas del reloj. Varios supuestos.

III. 2. Definición de vecindad

III.2.1. Motivación para el desarrollo del algoritmo y relación con los antecedentes

El doble sistema de organización de datos desarrollado (partición del espacio y triangulación de Delaunay) tienen como objetivo obtener el vecindario de cualquier punto LiDAR sin necesidad de realizar cálculos ni búsquedas adicionales dentro de la nube de puntos. En este sentido, tal y como se ha comentado en el apartado de antecedentes, existen multitud de modelos para definir el vecindario de un punto. Una de los enfoques más utilizado en los

últimos años es la adaptación del vecindario a la pendiente local para evitar que un mismo vecindario contenga puntos pertenecientes a diferentes entidades (Filin y Pfeier 2005; Filin y Pfeier 2006; Lari et al. 2011; Lari y Habib 2012). Este tipo de vecindario es muy útil cuando se pretende generar modelos virtuales de ciudades a partir de puntos LiDAR y es necesario generar los planos de techo que mejor se ajustan a los datos en el proceso de reconstrucción de los edificios. No obstante, éste no es el caso que nos ocupa, ya que el objetivo es detectar de forma eficiente las distintas entidades que conforman la escena y por tanto, será necesario que distintas entidades estén incluidas en la misma definición de vecindario para poder extraer sus características con el fin de diferenciarlas. Por consiguiente, se decide utilizar una definición de vecindario basado en un cilindro vertical de altura no finita similar a la utilizada por otros autores como Lisen y Prautzsch (2001), donde el radio del cilindro será fijo o variable en función de la aplicación que se esté ejecutando. Esta definición del vecindario permitirá seleccionar los puntos más próximos a uno dado sin tener en cuenta la altitud ni la entidad a la que pertenece cada uno de ellos. Todo ello puede apreciarse en la Figura 33, donde para una zona concreta de los datos test 2(Collado de Villalba) se observa como puntos pertenecientes al terreno, edificios y árboles quedan incluidos en un mismo vecindario.

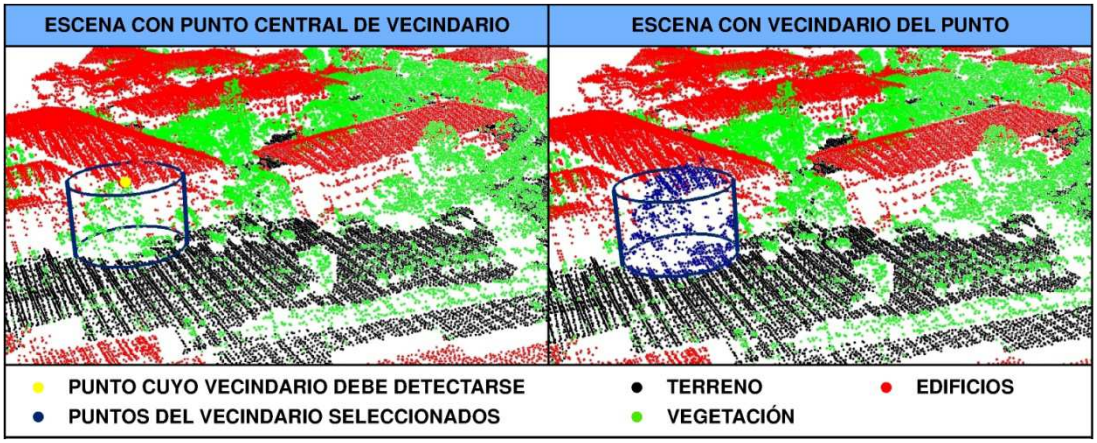


Figura 33. Definición de vecindario basado en cilíndrico vertical. Zona test 2 (Collado Villalba).

III.2.2. Descripción del algoritmo programado

Para definir el vecindario se genera un arreglo de codificación que permite obtener de forma directa los puntos más próximos a uno dado. Eesta estructura se determina de forma diferente en función del sistema de organización que la aplicación esté utilizando en cada momento.

- **Definición de vecindario en un sistema de organización basado en partición del espacio**

El proceso comienza con la propia organización mediante partición del espacio en celdas de altura no finita, ya que esta organización consume escasos recursos y se realiza en tiempo de ejecución. El tamaño de las celdas vendrá determinado por el radio del cilindro utilizado para definir el vecindario.

Una vez que el espacio ha sido dividido en celdas de igual tamaño y que la totalidad de puntos LiDAR han sido clasificados en función de la celda a la que pertenecen, la detección del vecindario de un punto *P* se realiza mediante la búsqueda de los puntos contenidos en la misma celda que dicho punto, así como en las celdas colindantes. De tal forma que para todos los puntos pertenecientes a dichas celdas se calcula la distancia horizontal con respecto a *P* , incluyendo en el vecindario aquellos puntos cuya distancia es menor que el

radio del cilindro elegido. Tal y como se aprecia en la Figura 34, dado un punto *P* del que queremos obtener su vecindario cilíndrico, tras organizar los datos en celdas cuadradas de igual lado que el radio del cilindro, se accede de forma automática a la celda en la que está contenida dicho punto, así como a las celdas colindantes, seleccionando finalmente aquellos puntos contenidos en dichas celdas que están a una menor distancia horizontal de *P* que el radio del cilindro vertical. Mediante este sistema el número de cálculos se reduce al número de puntos contenidos en las celdas testeadas, consiguiendo eliminar búsquedas y cálculos adicionales con respecto al resto de datos contenidos en la nube de puntos.

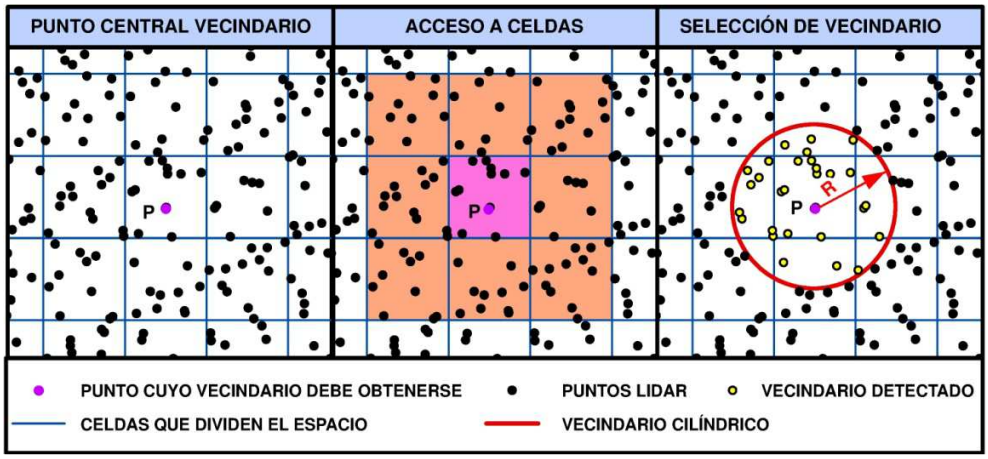


Figura 34. Vecindario cilíndrico mediante partición del espacio.

- **Definición de vecindario en un sistema de organización basado en triangulación de Delaunay**

La organización de datos mediante triangulación se realiza en el proceso de importación ya descrito en apartados anteriores (Apartado III.1.2.2) y no en tiempo de ejecución, tal y como ocurre con la organización mediante partición del espacio. Para definir el vecindario utilizando como base la red de triángulos, el algoritmo lleva a cabo los siguientes procesos:

Carga de datos de entrada y codificación de datos:

Será necesario almacenar en memoria tanto las coordenadas de los puntos LiDAR como los triángulos que definen la red de triángulos de Delaunay. Para ello se cargarán los siguientes ficheros:

Nombre_Proyecto.delaunay.txt: Fichero de texto con la triangulación de Delaunay generada en el proceso de importación. Se trata de una matriz de tres columnas donde cada fila se corresponde con un identificador del triángulo de la triangulación. En las diferentes columnas se definen los tres vértices de cada triángulo, que se corresponderán con identificadores de puntos LiDAR de la nube de puntos. Este fichero se almacenará en una matriz denominada "Tabla_triangulos".

Nombre_Proyecto.XYZ.txt: Fichero de texto con tres columnas correspondientes a las coordenadas X,Y,Z de la nube de puntos. Para cada punto, el número de registro se corresponde con el identificador del punto LiDAR.

Generación de codificación en triángulos:

Para facilitar la definición del vecindario se generará un sistema de codificación que permita acceder de forma directa a la información asociada a la totalidad de triángulos en los que está contenido cada uno de los puntos.

Dado un punto LiDAR cualquiera, dicho punto será el vértice adyacente de una serie de aristas pertenecientes a diferentes triángulos, de lo que se trata es de asociar cada punto LiDAR con los triángulos a los que pertenecen dichas aristas. Tal y como se observa en la triangulación de la Figura 35, el punto LiDAR número "5" forma parte de los triángulos de identificadores 3,5,6,7 y 8. De lo que se trata es de plasmar esta información para poder acceder directamente a los triángulos en los que está contenido cada uno de los puntos. Para ello se generará un vector para cada uno de los puntos LiDAR que contendrá los identificadores de triángulo de los que forma parte. El sistema se desarrolla a partir de un arreglo bidimensional mediante la definición de un doble puntero. El primer puntero apuntará a la dirección de memoria de cada punto LiDAR, mientras que el segundo apuntará a la posición del vector asociado que almacena los identificadores de triángulos en los que está contenido. Dado que el número de triángulos en los que pertenece cada punto LiDAR es variable, el tamaño de los vectores será diferente, por tanto, para facilitar la lectura de cada vector el primer elemento del mismo indicará su tamaño. Todo ello puede observarse en el ejemplo de la Figura 35.

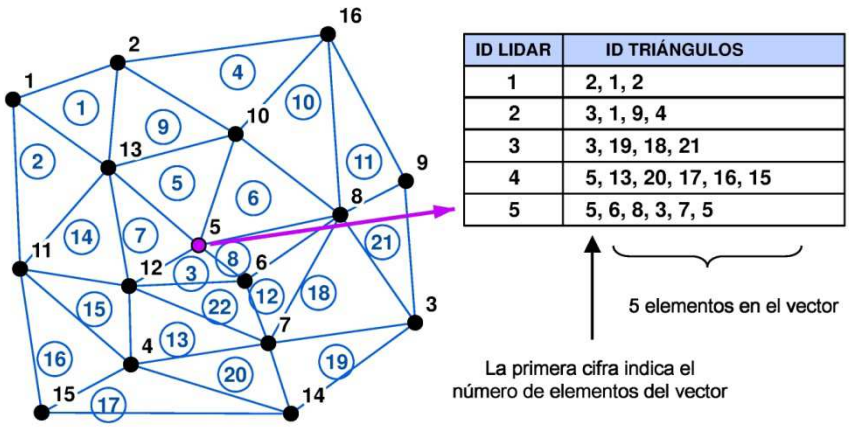


Figura 35. Sistema de codificación en triángulos.

A priori no se conocerá el tamaño de cada uno de los vectores que componen el arreglo, por tanto, el primer paso será determinar el tamaño de los mismos. Para ello, se genera un vector de contadores de igual tamaño que el número total de puntos LiDAR y se inicializa con el valor 0. Posteriormente se recorre la matriz de triángulos leyendo los identificadores de puntos existentes en cada una de las tres columnas a la vez que se incrementa el vector de contadores en la posición correspondiente. El resultado constituirá un vector donde para cada punto LiDAR aparecerá el número de triángulos en los que está contenido y por tanto, el tamaño del vector asociado dentro del arreglo. Finalmente, mediante la utilización de este vector de contadores se define el arreglo bidimensional de codificación.

Tras definir el arreglo deberán rellenarse los vectores asociados a cada uno de los puntos. Para ello se recorrerá de nuevo la matriz de triángulos leyendo los identificadores de los puntos LiDAR asociados a cada triángulo y registrando los identificadores de triángulos en los registros correspondientes del arreglo.

Definición de vecindario cilíndrico a partir de codificación en triángulos

Una vez se dispone del arreglo de codificación en triángulos será posible obtener el vecindario de un punto LiDAR de forma eficiente sin cálculos adicionales. Para ello, se implementa una metodología que utiliza como base el crecimiento de regiones.

Dado un punto *P* perteneciente al conjunto de datos LiDAR del que se pretende obtener el vecindario determinado por un cilindro vertical no finito de radio *R*, la aplicación accede directamente a los identificadores de los triángulos en los que el punto *P* está contenido

mediante la utilización del arreglo de codificación de triángulos y comprueba la distancia que existe entre sus vértices y el punto *P*. Si la distancia es menor al radio del cilindro, los identificadores LiDAR de los vértices se añaden a una lista enlazada que se genera utilizando la clase *Vector* disponible en C++. De esta forma se consigue un vector *V1* que aumenta su tamaño en tiempo de ejecución y que contiene los puntos que poseen una distancia menor a R con respecto a *P*. En adelante, a este vector lo denominaremos vector dinámico.

El proceso es iterativo y cada vez que un triángulo es computado se marca en un vector de marcadores de triángulos para no volver a ser comprobado de nuevo en iteraciones posteriores. Así mismo, ocurre lo mismo con los puntos que son añadidos al vector dinámico *V1* que serán marcados en un vector de marcadores de puntos.

Para detectar la totalidad de puntos pertenecientes al vecindario se recorre el vector dinámico *V1* computando los triángulos asociados a los puntos que contiene. Dicho vector aumentará de tamaño a la vez que es recorrido cuando un nuevo punto cumple con la condición umbral en distancia (Dist<*R*), de tal forma que el proceso finalizará cuando el vector es recorrido por completo.

El proceso completo puede entenderse más fácilmente mediante un ejemplo práctico. Dado el punto LiDAR con idenficador 5 del que se pretende obtener el vecindario, en primer lugar se genera un vector *V1* con todos los puntos que están a menor distancia que el radio *R* del cilindro. Para ello se computan los triángulos en los que está contenido el punto, de tal forma que si un punto perteneciente a estos triángulos está a menor distancia que el parámetro umbral, se registra en un vector dinámico *V1*. Tomando, una distancia *R* de 2,5 m y el punto 5 como origen de los cálculos, el proceso en la generación del vector *V1* puede observarse en la siguiente imagen (Figura 36).

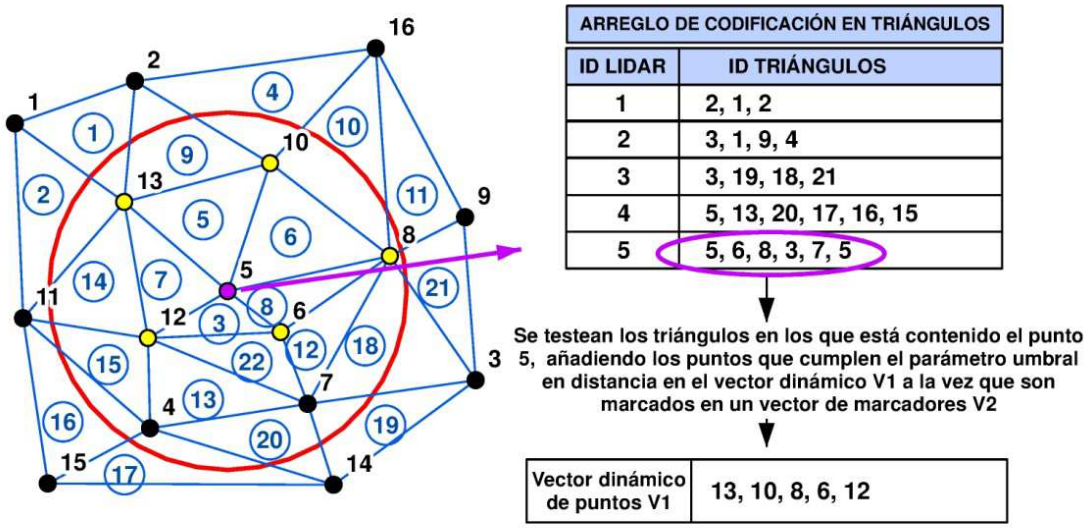
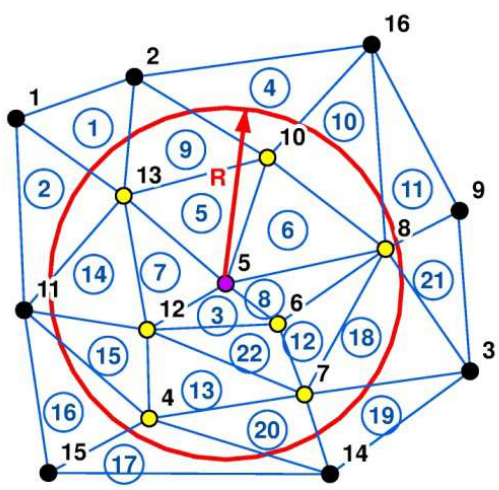


Figura 36. Generación inicial del vector dinámico de puntos.

Seguidamente se recorre el vector *V1* computando los triángulos asociados a los nuevos puntos registrados a partir del arreglo de codificación en triángulos. Así mismo, cada vez que se detecta un nuevo punto que está a menor distancia de 2,5 m del punto 5, se registra en *V1* siempre y cuando no haya sido marcado con anterioridad en el vector de marcadores, lo cual significaría que el punto ya ha sido introducido en el vector dinámico *V1* con anterioridad. De esta forma, el vector *V1* irá aumentando de tamaño a medida que se efectúan los cálculos. En el ejemplo que nos ocupa, se computarán los triángulos asociados

a los puntos 13, 10, 8, 6 y 12 y dos nuevos puntos son detectados dentro del vecindario (puntos 4 y 7) y añadidos automáticamente al vector *V1*.



Vector dinámico de puntos V1	13, 10, 8, 6, 12

Se testean los triángulos asociados a los puntos añadidos al vector dinámico en la iteración anterior

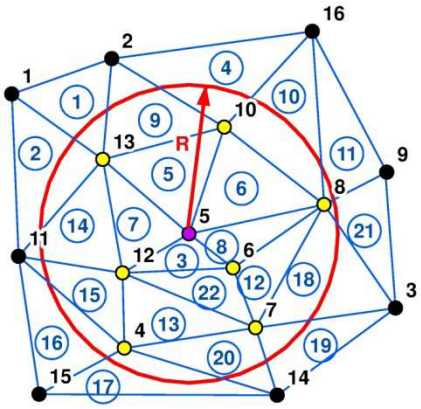
ARREGLO DE CODIFICACIÓN EN TRIÁNGULOS	
ID LIDAR	**ID TRIÁNGULOS**
13	7, 1, 9, 5, 7, 14, 2, 1
10	5, 4, 10, 6, 5, 9
8	6, 11, 21, 18, 8, 6, 10
6	4, 3, 8, 12, 22
12	6, 7, 3, 22, 13, 15, 14

Se añaden los puntos que cumplen con respecto al parámetro umbral en distancia (R)

Vector dinámico de puntos V1	13, 10, 8, 6, 12, 4, 7

Figura 37. Actualización del vector dinámico de puntos. Vecindario cilíndrico.

El proceso continúa recorriendo las dos últimas posiciones del vector dinámico *V1* y computando los triángulos asociados a los puntos LiDAR 4 y 7. En este caso, ningún nuevo punto perteneciente a sus triángulos asociados cumple con respecto al parámetro umbral en distancia *R* y en consecuencia el vector dinámico *V1* no aumenta de tamaño, lo que significa que el vecindario ha sido detectado por completo y el proceso finaliza.



Vector dinámico de puntos V1	13, 10, 8, 6, 12, 4, 7

Se testean los triángulos asociados a los puntos añadidos al vector dinámico en la iteración anterior

ARREGLO DE CODIFICACIÓN EN TRIÁNGULOS	
ID LIDAR	**ID TRIÁNGULOS**
4	5, 13, 20, 17, 16, 15
7	7, 12, 18, 19, 20, 13, 22

Ningún nuevo punto cumple con respecto al parámetro umbral en distancia (R)

El proceso de detección del vecindario finaliza

Vector dinámico de puntos V1	13, 10, 8, 6, 12, 4, 7

Figura 38. Finalización de cálculo del vecindario cilíndrico con Triangulación de Delaunay.

III. 3. Detección de errores grosesor ('*outliers*')

III.3.1. Motivación para el desarrollo del algoritmo y relación con los antecedentes

La existencia de puntos de altitud atípica (errores groseros) en el conjunto de datos LiDAR puede producirse por dos motivos fundamentales: por un lado, la reflexión de pulsos láser sobre aves u otros objetos que producen puntos de cota superior a la esperada (Sithole y Vosselman 2004); por otro, un mal funcionamiento del sistema de emisión y recepción del haz láser así como de los errores aleatorios que generan ocasionalmente puntos LiDAR por debajo de la superficie real del terreno (Meng et al. 2010).

En el tratamiento de datos LiDAR donde el objetivo es clasificar las diferentes entidades que conforman la escena, es fundamental detectar y filtrar todos los errores groseros antes de comenzar con el procesado de datos (Forlani et al. 2006; Chehata et al. 2008). Dado que este es el objetivo del presente trabajo, es necesario desarrollar una aplicación para detectar y filtrar este tipo de puntos.

Existen numerosos enfoques para detectar pulsos con valores atípicos en los datos LiDAR tales como análisis de la distribución de frecuencias en la altitud (Shen et al. 2011, Lin y Zhang 2014; Meng et al. 2009; Silván-Cárdenas y Wang 2006), la aplicación de filtros morfológicos (Kobler et al. 2007; Mongus y Zalik 2012,...), o bien aquellos basados en realizar un análisis de la densidad de puntos (Sotoodeh 2006 y 2007). Además, otra de las metodologías que ofrece buenos resultados consiste en analizar el vecindario de los distintos puntos LiDAR en búsqueda de una distribución inusual de los datos en cuanto a altitud se refiere (Lin et al. 2011, Lin y Zhang 2014; Meng et al. 2009; Silván-Cárdenas y Wang 2006)

El algoritmo desarrollado en el presente estudio utiliza este último enfoque debido a la facilidad de implementación y a la posibilidad de ejecución sobre datos LiDAR brutos dispuestos de forma aleatoria. La aplicación implementada combina técnicas de segmentación mediante crecimiento de regiones con un análisis del vecindario para detectar distribuciones anómalas de los datos de forma local. Todo ello se expone con detalle en el siguiente apartado.

III.3.2. Descripción del algoritmo programado

Tal y como se ha comentado con anterioridad, los puntos de cota anómala son generados por diversos motivos tales como errores de medición del sensor o reflexiones inusuales de los pulsos láser (reflexiones sobre aves, personas, etc.). Dado que con los distintos algoritmos diseñados en el presente trabajo se pretende clasificar la escena en cuatro clases (vegetación y pequeñas entidades, terreno, puentes y edificios), será necesario detectar también como errores groseros aquellos puntos pertenecientes a líneas eléctricas, grúas y otras estructuras artificiales que no pertenecen a la clase edificio. Los distintos tipos de puntos con altitud atípica que deben de ser detectados se muestran en la Figura 39. La imagen pertenece a dos zonas distintas de los datos test 2 (Collado de Villalba). En ella se pueden observar puntos groseros ocasionados por errores en el sistema de medición que poseen una altitud muy distinta a la del terreno (errores de medición tipo 1). Estos puntos pueden identificarse fácilmente debido a que su altitud varía mucho con respecto al resto de los datos LiDAR. Otros errores registrados por el sistema tienen una altura más próxima al resto de puntos e identificarlos conlleva mayor dificultad (errores de medición tipo 2). Además en el ejemplo propuesto, existen otro tipo de puntos ocasionados por la reflexión de los pulsos láser en diferentes objetos que deberán también detectarse (reflexión sobre grúas, líneas eléctricas, etc.).

Al analizar las características de los diferentes puntos que deben de ser identificados encontramos: puntos aislados, puntos de altitud anómala agrupados (gruas y líneas eléctricas), puntos cuya altura difiere en gran medida con respecto al resto de información tridimensional (errores de medición tipo 1), así como puntos cuya altitud no difiere en exceso con respecto al resto (errores de medición tipo 2). En este sentido, el algoritmo a desarrollar deberá tener en cuenta todas estas características para conseguir una mayor eficiencia.

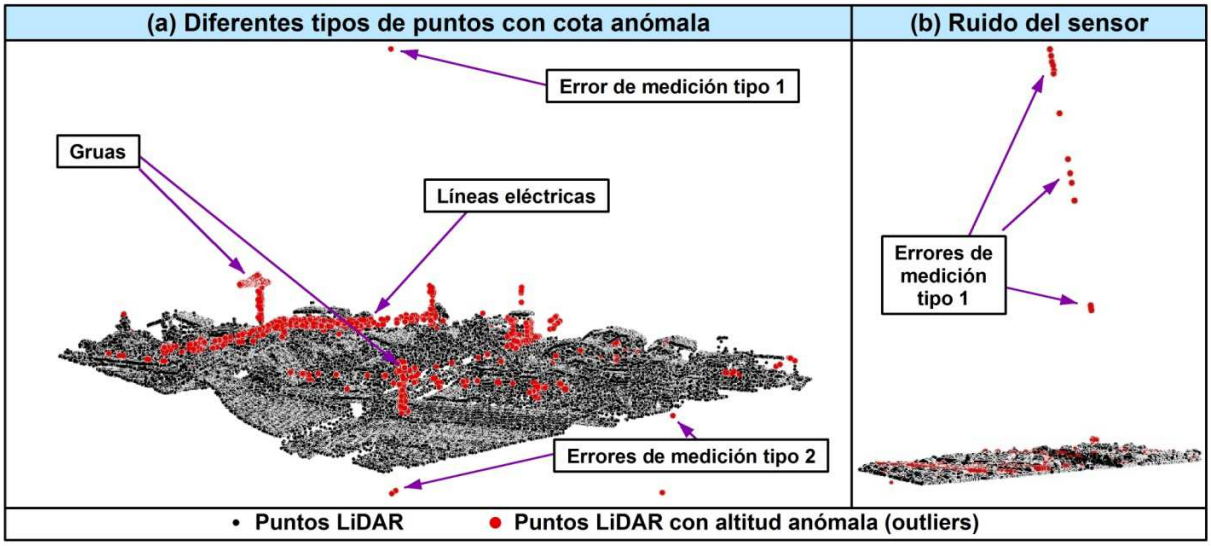


Figura 39. Diferentes puntos groseros ('*outliers*') en dos zonas de los datos 2.

Para empezar, dado que este tipo de errores pueden darse como una entidad puntual aislada o como regiones compuestas por una escasa cantidad de puntos, el algoritmo desarrollado segmenta la escena en superficies continuas, de tal forma que los puntos susceptibles de poseer una altitud anómala se localizarán en aquellas regiones que poseen un menor tamaño. No obstante, no todos los puntos pertenecientes a regiones pequeñas se corresponderán con outliers, ya que por ejemplo, pequeños planos de techo y otras estructuras en edificaciones pertenecen a regiones de pequeño tamaño y no por ello deben de considerarse como valores atípicos. Por consiguiente, para identificar los puntos outliers entre las regiones de menor tamaño será necesario realizar un análisis adicional. Este segundo análisis, se fundamenta en el hecho de que la mayoría de puntos LiDAR incluidos en el vecindario de un punto grosero tiene una altitud muy diferente.

De forma general, los dos procesos que sigue el algoritmo implementado pueden observarse en la Figura 40. En primer lugar los datos LiDAR son segmentados en superficies continuas (Algoritmo auxiliar. Apartado III.10.1). Seguidamente, se identifican aquellas regiones cuya superficie es menor que un cierto umbral. En el ejemplo propuesto se localizan dos pequeñas regiones susceptibles de estar compuestas por puntos de altitud anómala (Figura 40a). La primera de ellas (R-1) está constituida por los pulsos láser reflejados sobre la chimenea de un edificio, la segunda (R-2), se corresponde con un punto de altitud atípica. En ambos casos se genera el vecindario cilíndrico vertical asociado al posible punto grosero (Figura 40b) y posteriormente se comprueba si el desnivel con respecto a los puntos incluidos en su vecindario es elevado. En caso afirmativo estaremos frente a un punto de cota anómala. Para determinar si el desnivel es elevado o no se establece un parámetro umbral en desnivel.

En el ejemplo propuesto, para la primera de las regiones (R-1) únicamente un punto del vecindario supera el umbral en desnivel y consecuentemente los pulsos reflejados sobre la chimenea no son clasificados como errores groseros. En el segundo caso (R-2) la totalidad

de los desniveles calculados superan el parámetro umbral establecido, por lo que el punto LiDAR es clasificado como error grosero (Figura 40b).

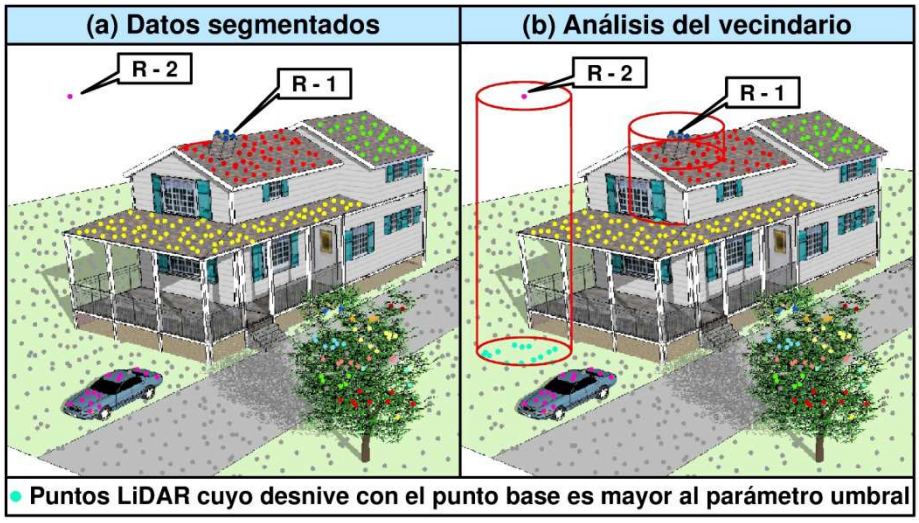


Figura 40. Procedimiento para detectar errores groseros.

Para detectar puntos de cota anómala se han implementado las dos aplicaciones que se describen con detalle a continuación.

III.3.2.1. Detección de errores groseros. Algoritmos A y B: análisis de variaciones locales de desnivel. Único proceso

Tal y como se aprecia en la Figura 39, se deben identificar dos tipos de anomalías en los datos. La primera de ellas es debida a errores en el proceso de medición o a la reflexión de pulsos láser sobre aves. En estos casos, se generan puntos LiDAR con una altitud muy diferente a la del resto de datos y por tanto fáciles de identificar. Por el contrario, también se producen errores groseros cuya altitud es similar al terreno y consecuentemente detectarlos correctamente conlleva mayor dificultad. El algoritmo desarrollado es capaz de detectar ambos tipos de anomalías pero mediante el uso de distintos parámetros umbrales. En lo sucesivo, se hará referencia al algoritmo A cuando el objetivo sea detectar puntos con errores groseros cuya altitud difiere en gran medida al resto datos, y al algoritmo B cuando la finalidad sea detectar puntos anómalos próximos al terreno.

El diagrama de flujo que sigue la aplicación se detalla en la Figura 41. En primer lugar, en función del tipo de anomalías que se pretende detectar deberán adecuarse los parámetros umbrales. Si el objetivo es detectar puntos de cota anómala con una altitud muy diferente al resto de datos (Algoritmo A) deberá optarse por un parámetro umbral en desnivel (*Umbral_IZ*) alto. Además, el parámetro umbral en porcentaje utilizado para analizar las variaciones locales en desnivel (*Umbral_Porcentajes*) debe de ser más relajado. Por otro lado, en caso de que la finalidad sea detectar puntos anómalos con una altitud próxima al terreno, el parámetro umbral en desnivel deberá ser menor y el umbral en porcentajes más restrictivo.

Una vez definidos los parámetros umbrales, la aplicación segmenta la escena en superficies continuas delimitadas por cambios bruscos en altitud (Algoritmo auxiliar. Apartado III.10.1). Los puntos de cota anómala pertenecerán a regiones de pequeño tamaño, por lo que la aplicación únicamente analizará las regiones menores que un cierto parámetro umbral con el fin de identificar anomalías. Para cada uno de los puntos pertenecientes a estas regiones se analizan las variaciones locales en desnivel. Para ello, se genera el vecindario cilíndrico

vertical asociado y el desnivel existente entre los diferentes puntos incluidos en él con respecto al punto base calculado. Para determinar si la altitud del punto base considerado es anómala, se contabiliza el número de desniveles mayores al parámetro umbral establecido (*Umbral_IZ*), de forma que si la mayoría de los desniveles calculados superan el umbral podrá afirmarse que la altitud del punto base difiere en gran medida con respecto a su vecindario y por tanto la altitud del punto será considerada anómala. Con el fin de determinar si la mayoría de puntos del vecindario tienen una altitud diferente con respecto al punto base objeto de análisis, la aplicación contabiliza el número de desniveles que sobrepasan el parámetro umbral y calcula el porcentaje con respecto al total, de tal forma que si el porcentaje es superior al umbral establecido (*Umbral_Porcentajes),* el punto es clasificado como error grosero.

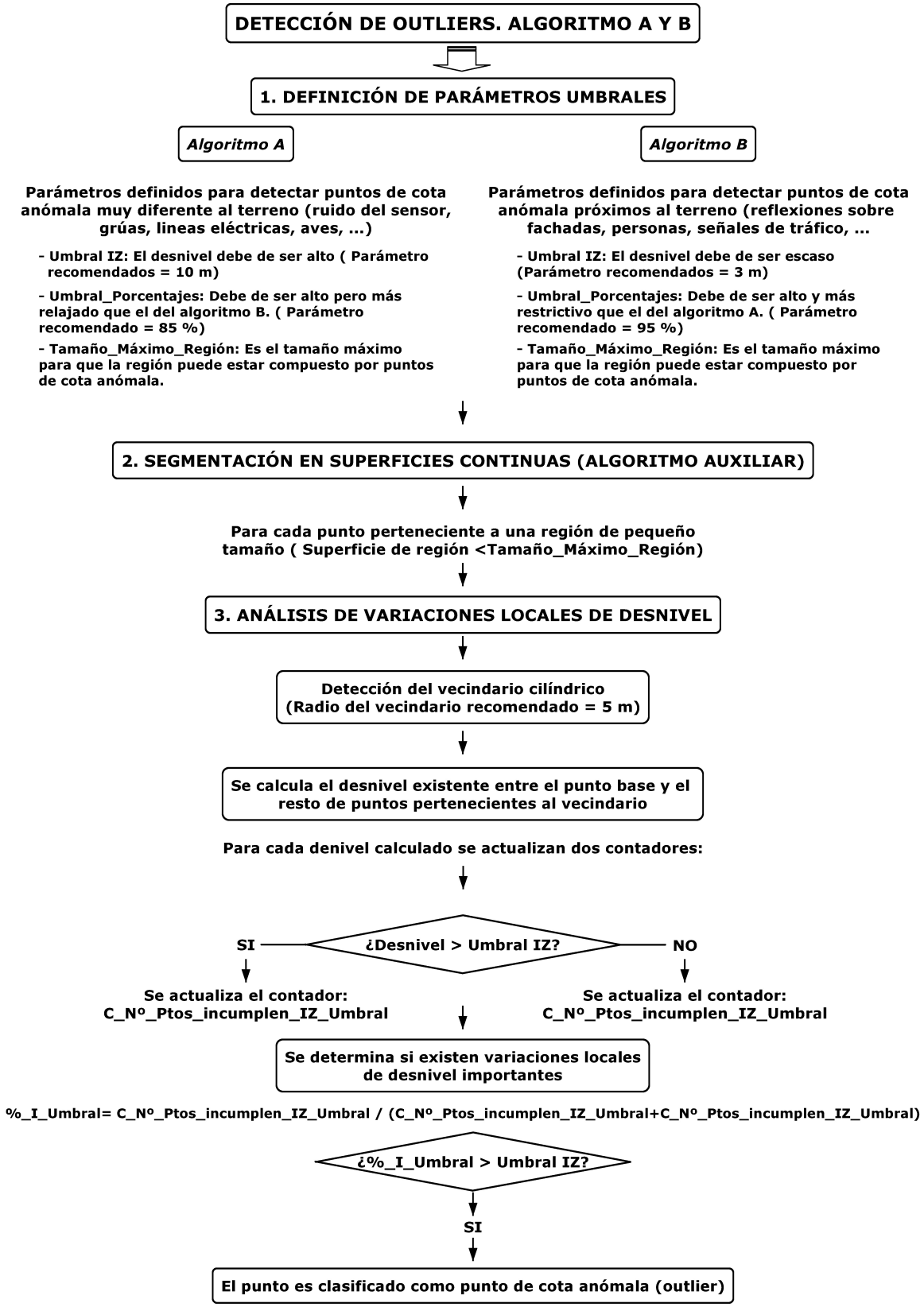


Figura 41. Diagrama de flujo. Detección de outliers. Algoritmo A y B.

III.3.2.2. Detección de errores groseros. Algoritmo C: análisis de variaciones locales de desnivel. Multiproceso con parámetros adaptativos

El algoritmo descrito en el apartado anterior es capaz de detectar puntos anómalos cuya altitud difiere en gran medida con respecto a la esperada (algoritmo A) así como aquellos cuya altitud está más próxima a la superficie real del terreno (algoritmo B). El principal problema que presenta la metodología propuesta es la necesidad de ajustar los parámetros umbrales para identificar los dos tipos de anomalías, ya que en consecuencia no es posible detectar la totalidad de errores groseros mediante una única ejecución del proceso. Para solucionar el problema, se implementa una nueva aplicación que ejecuta el algoritmo A y B de forma independiente y posteriormente fusiona los resultados. Con ello se consigue detectar y filtrar los dos tipos de valores atípicos.

Los diferentes procesos que ejecuta el algoritmo se detallan en el diagrama de flujo (Figura 42). En primer lugar, la aplicación segmenta la escena en superficies continuas, seguidamente, se ejecuta de forma individualizada el algoritmo descrito en el apartado anterior con los parámetros ajustados para identificar los dos tipos de puntos anómalos (algoritmo A y B). Los errores groseros detectados tras ejecutar ambos procesos son marcados en un vector de marcadores de igual dimensión que el número total de puntos LiDAR (*V_M_Ptos_Anómalos*). Finalmente, utilizando dicho vector es posible eliminar de los datos LiDAR la totalidad de anomalías detectas e imprimir el resultado. Además se imprime un fichero con los identificadores de los puntos con valores anómalos.

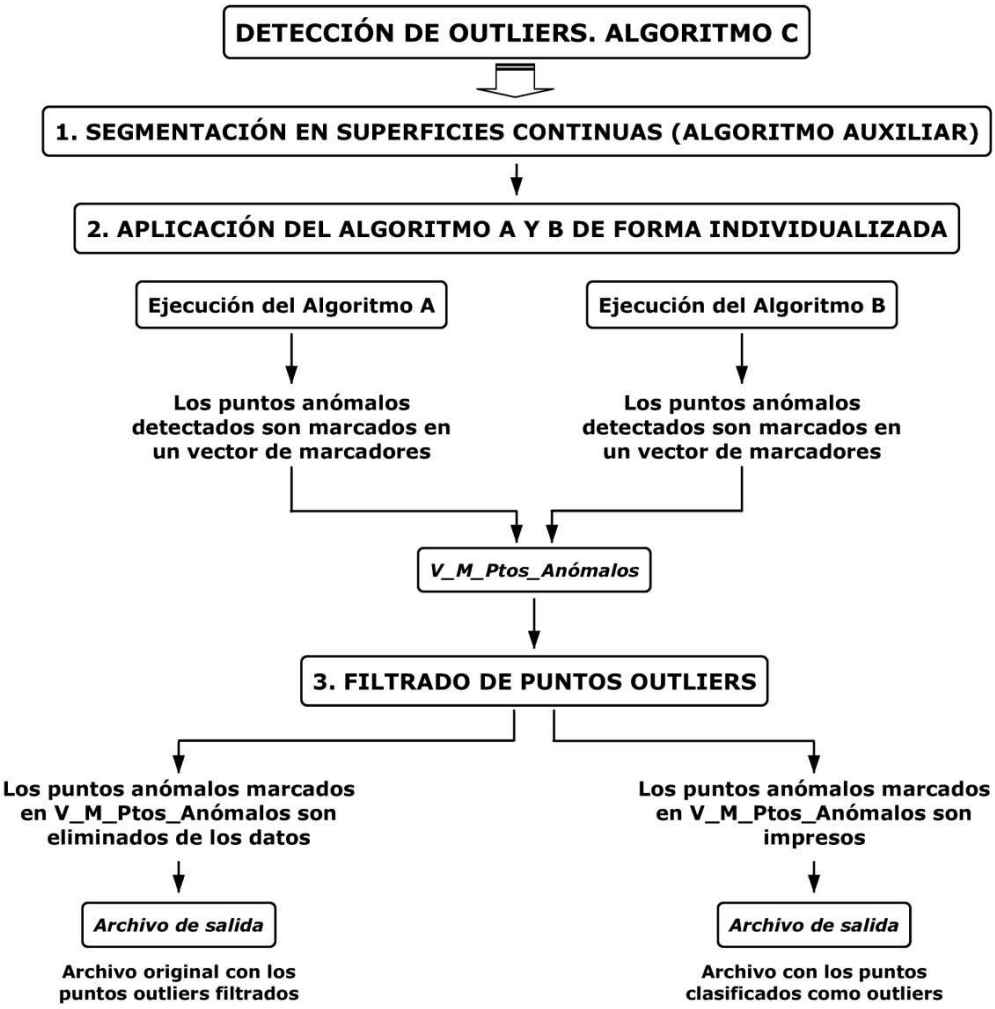


Figura 42. Diagrama de flujo. Detección de outliers. Algoritmo C.

III. 4. Detección automática de puntos pertenecientes al terreno. Algoritmo A: densificación progresiva mediante interpolación selectiva combinada con segmentación

III.4.1. Motivación para el desarrollo del algoritmo y relación con los antecedentes

Un aspecto fundamental en el tratamiento de datos LiDAR es la detección de los puntos pertenecientes al terreno con el fin de generar modelos digitales del terreno y utilizar el resultado como punto de partida en la detección del resto de entidades (árboles, edificios, automóviles, etc.).

Tal y como se ha comentado en el apartado de antecedentes existen numerosos métodos con este propósito, donde uno de los problemas más importantes son las grandes dificultades encontradas a la hora de desarrollar algoritmos completamente automáticos que actúen con efectividad independientemente de las características de la escena (Baltsavias 1999). En este sentido, el presente trabajo pretende dar solución a este problema en entornos urbanos. Concretamente, el objetivo es desarrollar un algoritmo de precisión que actúe de forma eficiente en entramados urbanos complejos con características altamente variables.

Uno de los algoritmos más utilizados para detectar el terreno son los filtros morfológicos (Voselman 2000; Sithole 2001; Roggero 2001; Chen et al. 2007; Kim y Shan 2012). Una de las ventajas que posee este método es que a la vez que se detectan los puntos pertenecientes al terreno se genera el modelo digital del terreno de forma automática, por lo que es altamente eficiente computacionalmente cuando el objetivo es la generación de modelos digitales del terreno. No obstante, resulta un método de filtrado cuyos parámetros son muy variables en función de las características de la escena, tales como la pendiente y la existencia de discontinuidades en altura (edificios, vegetación...). Con el fin de solucionar este problema, multitud de autores han propuesto distintas variaciones del algoritmo de filtrado morfológico propuesto inicialmente por Linderberger (1993). Es el caso de Sithole (2001) y Roggero (2001) que incorporan la pendiente local de cada punto para aplicarla al elemento estructural o Chen et al. (2007) y Kim y Shan (2012) que aplican el filtrado morfológico de forma adaptativa a la pendiente teniendo así en cuenta las características locales del terreno. Además, con el fin de minimizar la dependencia del filtro morfológico a la variabilidad de las características de la escena, otros autores han aplicado el filtro con diferentes tamaños y la forma del elemento estructural (Kilian et al. 1996; Arefi y Hahn 2005; Kobler et al. 2007). Por otro lado, la distribución aleatoria de los puntos LiDAR también es un factor que influye directamente en la eficacia de este tipo de filtros (Li, 2013).

Dada la dificultad de encontrar parámetros que actúen eficientemente de forma independiente a las características de la escena tales como la forma y el tamaño del elemento estructural, así como la dificultad de adaptar el filtro a terrenos con diferentes pendientes y grandes discontinuidades en altura, se opta en el presente estudio por no utilizar como base el filtro morfológico en el desarrollo del algoritmo de detección del terreno. No obstante, algunas características del mismo sí se utilizarán tal y como se describirá en posteriores apartados.

Otro clase de algoritmos muy utilizados es la segmentación de los datos LiDAR con el fin de agruparlos en regiones homogéneas con atributos similares. En este sentido, la segmentación por crecimiento de regiones es uno de las metodologías más utilizadas para segmentar los datos en las diferentes superficies que conforman la escena (Tovari y Pfeifer 2005; Rabbani et al. 2006; Kim et al. 2007; Lari et al 2011). La segmentación de datos

suele utilizarse para clasificar los puntos en los diferentes planos que conforman la escena en procesos de reconstrucción de edificios y generación de modelos virtuales de ciudades. No obstante, algunos estudios muestran también su eficacia en la detección de la clase terreno (Lohmann 2002; Nardinocchi et al. 2003; Sithole 2005; Sithole y Vosselman 2005; Shen et al. 2012). El principal problema que muestra la aplicación de este tipo de filtros en la detección del terreno es que no utiliza ningún tipo de superficie de referencia adicional como el método de densificación o los filtros basados en interpolación de superficies. Ello conlleva a que en muchos casos el análisis de las características de las agrupaciones no sea suficiente para obtener buenos resultados, sobre todo en zonas boscosas donde se generan multitud de pequeñas agrupaciones tras la segmentación (Lin y Zhang 2014). Con el fin de solucionar este problemas, es frecuente combinar técnicas de segmentación con otras metodologías como la densificación de TIN (Ekhtari et al. 2008; Pérez et al. 2012; Lin y Zhang 2014). En el presente trabajo, tal y como se describirá en posteriores apartados, la segmentación de datos LiDAR será un proceso fundamental de la metodología propuesta, ya que además de permitir una alta optimización del tiempo de cómputo en procesos iterativos, servirá para detectar altas discontinuidades en altura y poder realizar una clasificación previa de los puntos que pueden pertenecer al terreno, consiguiendo así detectar correctamente el terreno y diferenciarlo de los edificios en entramados urbanos con alta pendiente y con tejados que poseen la misma altura que calles colindantes.

La densificación progresiva es otro de los métodos comúnmente utilizados para detectar el terreno en una nube de puntos LiDAR. Se trata de un proceso progresivo, donde a partir de una superficie inicial de referencia se van incorporando nuevos puntos pertenecientes al terreno de forma iterativa. Normalmente, se parte de una triangulación generada a partir de los puntos de menor cota del área de estudio y posteriormente los puntos que no difieren de altitud en exceso con respecto a la superficie de referencia (triangulación) se incorporan a la clase terreno. El proceso es iterativo de forma que la superficie de referencia (triangulación) es actualizada en cada iteración. Este algoritmo ha sido utilizado con frecuencia en multitud de estudios (Axelsson 2000; Sohn y Dowman 2002; Ekhtari et al. 2008) y es el algoritmo que utiliza el software comercial TerraScan. Este método tiene la ventaja de poseer una menor dependencia con respecto a las características del terreno que otras metodologías. No obstante, como inconveniente, lleva asociado un alto coste computacional, ya que la adición de cada nuevo punto a la clase terreno conlleva recalcular la triangulación de Delaunay para obtener la nueva superficie de referencia.

Dado que el objetivo del presente estudio es desarrollar un algoritmo eficiente para la detección del terreno en entramados urbanos, entendiendo como eficiencia una alta automatización del proceso, un bajo coste computacional y una óptima precisión en los resultados, se opta por desarrollar un nuevo algoritmo que utilice las características más ventajosas de los diferentes métodos existentes, combinándolas y aunándolas en un único algoritmo capaz de obtener óptimos resultados en entornos urbanos complejos.

En la elección de la metodología básica que utilizará el algoritmo a desarrollar se tiene en cuenta el estudio realizado por Sithole y Vosselman (2004). Este trabajo constituye un análisis comparativo experimental de los diferentes algoritmos de filtrado existentes, llegando a la conclusión de que los algoritmos que mejores resultados ofrecen son aquellos que utilizan superficies de referencia aproximadas tales como los métodos basados en interpolación de superficies o el método de densificación progresiva de TIN. Es por ello que finalmente se opta por desarrollar un algoritmo que utilice como base un método de densificación. No obstante, tal y como se ha comentado anteriormente, los métodos de densificación de TIN llevan asociado un alto coste computacional debido a la necesidad de recalcular la triangulación en cada iteración. Para evitar este problema se combinará el método de densificación con la segmentación de datos LiDAR, tal y como han propuesto otros autores como por ejemplo Pérez et al. (2012). En dicho estudio, previamente a la aplicación del método de densificación de TIN se procede a segmentar los datos en superficies continuas, de tal forma que posteriormente en la aplicación del método de

densificación de TIN cuando un punto es detectado como terreno, la totalidad de su grupo o segmento asociado es clasificado como tal. Este proceso minimiza considerablemente el número de iteraciones en el proceso de densificación y por tanto reduce el tiempo de cómputo asociado a la actualización de la red de triángulos. En este sentido, el algoritmo a desarrollar utilizará una premisa similar y segmentará los datos LiDAR en superficies continuas. No obstante, se pretende ir un paso más allá y evitar la actualización de la triangulación por ser la operación que mayor tiempo de cómputo conlleva. Para ello, se partirá de una triangulación de Delaunay inicial que será utilizada para establecer relaciones de proximidad y como base para la detección de los contornos de los distintos objetos que conforman la escena. No obstante, el terreno se extenderá bajo los objetos mediante una interpolación selectiva, proceso que conlleva un menor coste computacional y que consigue minimizar la transmisión de errores cuando un punto que no pertenece al terreno es clasificado erróneamente como tal. Además, para determinar los puntos pertenecientes al terreno se partirá de una clasificación inicial donde las agrupaciones que poseen menor altitud del área de estudio serán clasificadas como clase terreno. Para ello se dividirá la zona de estudios en bloques cuadrados y se tomará como terreno aquellas agrupaciones que poseen una menor cota de cada uno de ellos, siendo ésta una característica propia de los métodos de filtrado morfológico. La división de la zona de estudio en bloques con el fin de generar la superficie de terreno provisional inicial es una metodología que se ha utilizado con anterioridad en algoritmos de densificación (Axelsson 2000; Lin y Zhang 2014). En el presente estudio se tomará una anchura de bloque superior a la máxima longitud de fachada de edificio existente en la zona de estudio, de igual forma que en el estudio realizado por Lin y Zhang (2014).

A partir de estas premisas y mediante la combinación de características propias de diferentes métodos, así como la adición de nuevos aspectos tales como la interpolación selectiva, se han desarrollado dos nuevos algoritmos para la detección del terreno en entornos urbanos complejos. El primero de ellos parte de la propuesta realizada por Pérez et al. (2012) que combina la densificación de TIN con la segmentación, realizando las modificaciones oportunas para conseguir un proceso de densificación que utiliza interpolación selectiva para extender el terreno bajo las superficies sin necesidad de recalcular iterativamente la triangulación de Delaunay, consiguiendo así reducir el tiempo de cómputo y minimizar la transmisión de errores en la clasificación. Por otro lado, el segundo algoritmo desarrollado es una evolución del primero, donde la segmentación no solamente se utiliza para minimizar el tiempo de cómputo, sino que forma parte activa del propio proceso de clasificación de la clase terreno mediante el análisis individualizado de cada segmento o grupo obtenido tras la segmentación, consiguiendo una clasificación preliminar de las regiones que tienen mayor posibilidad de pertenecer al terreno. Con este segundo algoritmo se consiguen eliminar diferentes errores de comisión y disminuir la dependencia del resultado con respecto a los diferentes parámetros umbrales. Todo ello se expone con detalle en los siguientes apartados.

III.4.2. Descripción del algoritmo programado

Como se ha comentado con anterioridad, este algoritmo combina aspectos de las distintas metodologías existentes e incluye nuevas características con el fin de detectar de forma eficiente puntos LiDAR pertenecientes al terreno en entornos urbanos complejos. Para ello usa como base el método de densificación combinado con una segmentación de los datos mediante crecimiento de regiones. A continuación se describe de forma general el algoritmo desarrollado para posteriormente pasar a un análisis detallado del mismo.

El proceso comienza con la segmentación de los puntos LiDAR en las diferentes superficies continuas que conforman la escena (Algoritmo auxiliar. Apartado III.10.1), para ello se emplea un método de segmentación basado en crecimiento de regiones. La segmentación se inicia con la selección de un punto semilla al azar de entre el conjunto de los datos LiDAR

y posteriormente el crecimiento de la región se lleva a cabo a partir de dos parámetros umbrales: un parámetro umbral en distancia (*Id*); y un parámetro umbral en incremento de cota (*Iz*). Los puntos que cumplan ambos parámetros serán automáticamente incluidos en el grupo y seleccionados como puntos semilla para la siguiente iteración. El proceso de formación de un grupo o segmento finalizará cuando no se añada ningún nuevo punto al mismo. La operación se repite hasta que todos los puntos han sido clasificados en su grupo correspondiente, momento en el cual la segmentación de la escena en las distintas superficies continúas que la conforman la escena ha finalizado (Figura 43).

Figura 43. Superficies continuas segmentadas cada una en un color. Datos test 2.

Con el proceso de segmentación descrito, los puntos LiDAR son clasificados en superficies homogéneas delimitadas por cambios bruscos en altura. El siguiente paso consiste en clasificar de forma provisional las regiones obtenidas en la segmentación en dos clases: puntos pertenecientes al terreno y puntos no pertenecientes al mismo. Con ese fin, el área de estudio se divide en bloques de 150 metros de lado y posteriormente, una primera aproximación de la clase terreno se obtiene tras la selección de la agrupación de menor cota perteneciente a cada uno de los bloques. Para ello, se detecta el punto de menor cota de cada cuadrante y seguidamente se selecciona el grupo al que pertenece (Figura 44).

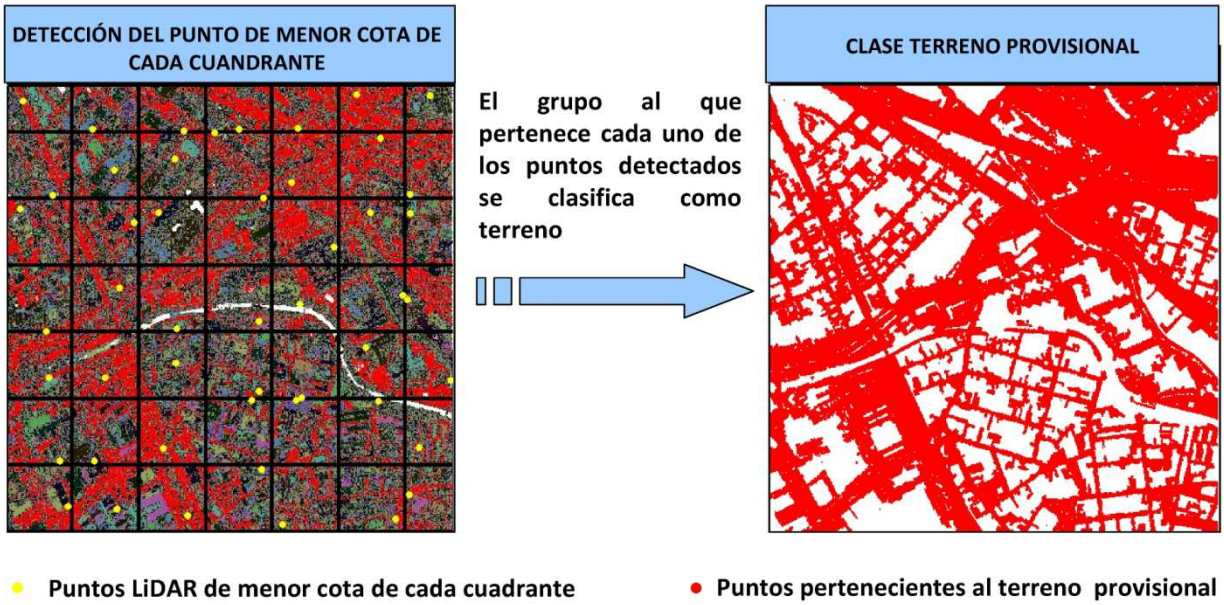


Figura 44. Detección de clase terreno provisional. Datos test 2 (Collado Villalba).

Llegado a este punto el resultado está compuesto por un gran número de regiones o segmentos interconectados que definen la clase terreno provisional. Sin embargo, en las grandes regiones conectadas pueden existir algunas pequeñas zonas pertenecientes al terreno que no han sido detectadas por no contener el punto de cota mínima del bloque al que pertenecen. Estas agrupaciones están asociadas a áreas de terreno aisladas rodeadas de vegetación o edificios. Un ejemplo de una región perteneciente al terreno que no ha sido detectado como tal puede observarse en la Figura 45.

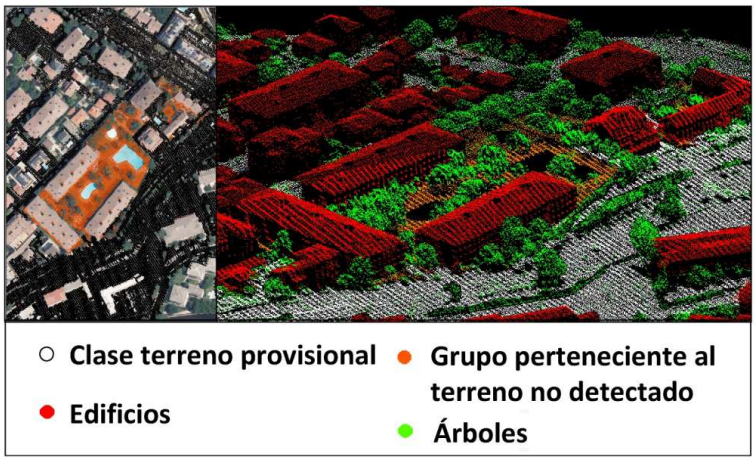


Figura 45. Región de terreno no detectada. Planta y vista 3D. Datos test 2.

Los grupos pertenecientes al terreno que no han sido detectados pueden ser identificados mediante un proceso iterativo. En primer lugar, se detectan y extraen los contornos límite de cada objeto que no pertenece al terreno. Seguidamente, los puntos de tierra asociados a dichos límites se detectan y el suelo se prolonga mediante interpolación generando una superficie continua. Por último, los puntos que pertenecen al terreno y que no fueron detectados inicialmente, se identifican mediante la comparación de su altura con respecto a la superficie del terreno obtenida mediante interpolación. Este proceso es iterativo y los puntos cuyo desnivel es menor a un cierto umbral con respecto a la superficie interpolada de referencia se añaden a la clase terreno, de tal forma que el proceso iterativo finaliza cuando ningún nuevo punto se añade a dicha clase.

El algoritmo programado se centra en el hecho de que el terreno no suele presentar grandes discontinuidades y tiene el enfoque de los métodos de densificación, ya que de forma iterativa, nuevos puntos se añaden a la clase terreno. No obstante, a diferencia de los métodos de densificación de TIN, no utiliza la red de triángulos para prolongar el terreno por debajo de los objetos, sino una interpolación selectiva de ponderación inversa a la distancia, donde los puntos pertenecientes al terreno y que son circundantes a cada entidad no detectada como tierra, son aquellos que intervienen en el proceso de interpolación. Gracias a esta enfoque se consigue minimizar considerablemente el tiempo de cómputo al no ser necesario recalcular la triangulación de Delaunay tras la adición de cada nuevo punto a la clase terreno. Además, el hecho de partir de una segmentación en superficies continuas de la totalidad de los puntos LiDAR, permite añadir la totalidad de la región o agrupación a la clase terreno cuando un solo punto perteneciente al mismo se detecta como tal, consiguiendo así minimizar el número de iteraciones y por tanto reducir el tiempo de cómputo considerablemente. Por otro lado, la utilización de la interpolación selectiva minimiza la transmisión de errores en altura asociada a la utilización de triángulos para extender el terreno por debajo de los objetos existentes en la escena.

Este último aspecto puede apreciarse en la siguiente imagen (Figura 46). Dado un conjunto de puntos pertenecientes al terreno, si se dispone de una zona no clasificada como tal donde el terreno debe de ser prolongado, el primer paso consiste en detectar los puntos de

borde del terreno asociados al límite de dicha zona (Figura 46a) para posteriormente utilizarlos como base de cálculo en la obtención de la superficie extendida. No obstante, al trabajar con datos LiDAR, en ocasiones pueden producirse errores en la clasificación del terreno asociados por ejemplo a pulsos láser reflejados por las fachadas de los edificios o a zonas con vegetación baja. En estos casos, un punto de borde del terreno podría pertenecer a otra entidad con el consiguiente error en cota (E(Z)) que ello conlleva (Figura 46b). En este sentido, el método elegido para extender el terreno influye directamente en la transmisión de dicho error a la nueva superficie calculada. Así por ejemplo, si se opta por utilizar una triangulación (Figura 46c), el error se transmitirá a todos los triángulos de la red donde el punto anómalo *P* constituye un vértice (Figura 46d). Sin embargo, si la superficie extendida se obtiene mediante una interpolación ponderada de forma inversa a la distancia (Figura 46e) el área de la superficie extendida donde el error se transmite será menor (Figura 46f). La diferencia fundamental radica en que la altura de cualquier punto de la superficie obtenida mediante triangulación se determina únicamente mediante tres puntos (vértices del triángulo asociado) y por tanto un error en cota existente en alguno de ellos se transmite de forma directa. Por el contrario, para el cálculo de la altura de cualquier celda de la superficie obtenida por interpolación participan la totalidad de puntos de borde (16 en el presente ejemplo), con lo que la influencia del error es menor y se hace patente únicamente en la zona más próxima al punto erróneo *P*.

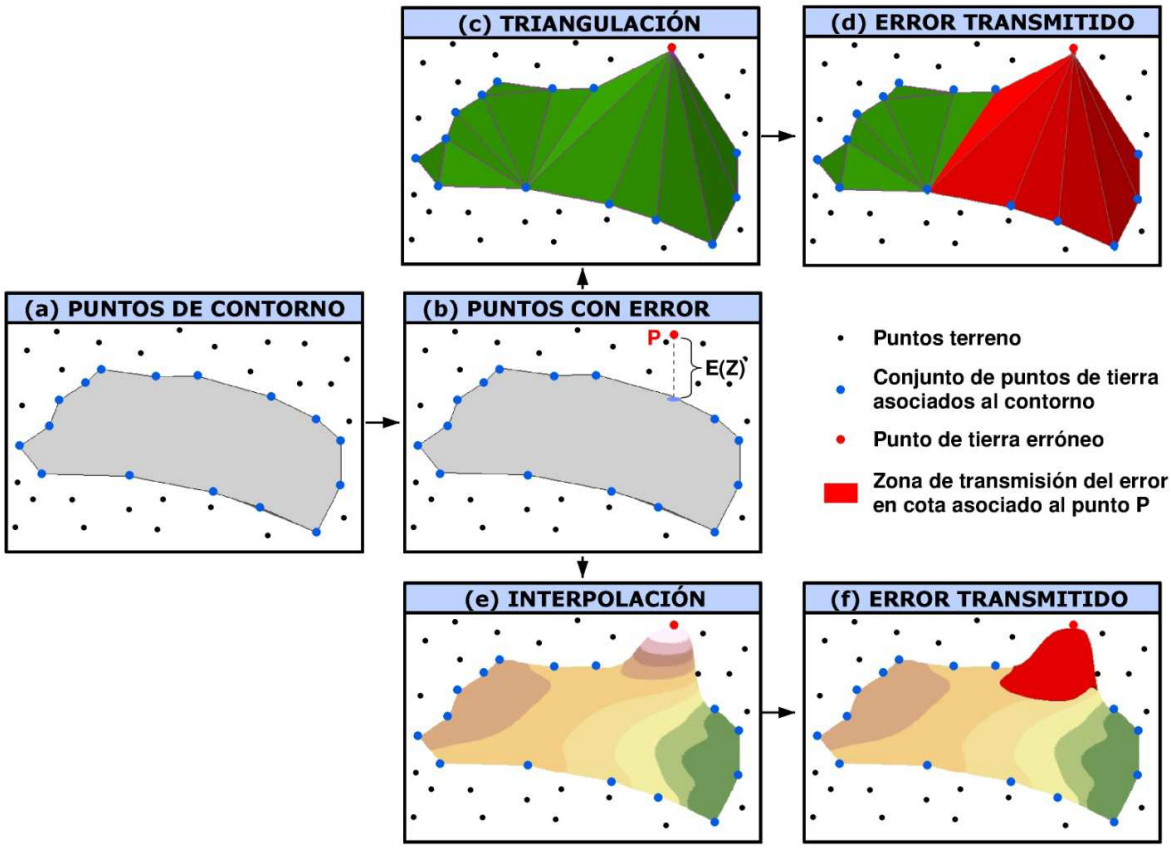


Figura 46. Comparación en la transmisión de errores al extender el terreno mediante interpolación y triangulación.

En el ejemplo propuesto el terreno debe de extenderse sobre una superficie de 32 m^2. Tras prolongar la superficie utilizando ambos métodos (triangulación e interpolación) y realizar un análisis de la superficie afectada por la altitud anómala del punto *P* puede concluirse que la porción de superficie donde se ha transmitido el error es más de cinco veces mayor al utilizar una red de triángulos que una interpolación selectiva mediante ponderación inversa

a la distancia. Concretamente el área afectada de la superficie obtenida por triangulación es de 19,75 m^2, mientras que al utilizar interpolación se reduce a tan solo 3,6 m^2.

Esta minimización de la transmisión de errores en altitud que presenta la interpolación mediante distancia inversa ponderada frente a la triangulación, junto con una disminución del tiempo de cómputo, son los dos aspectos fundamentales por los que se opta por utilizar la interpolación en el proceso de densificación. Siendo precisamente este aspecto, uno de los más novedosos de la metodología propuesta.

Los rasgos principales de la totalidad del proceso descrito pueden observarse en el siguiente diagrama de flujo (Figura 47).

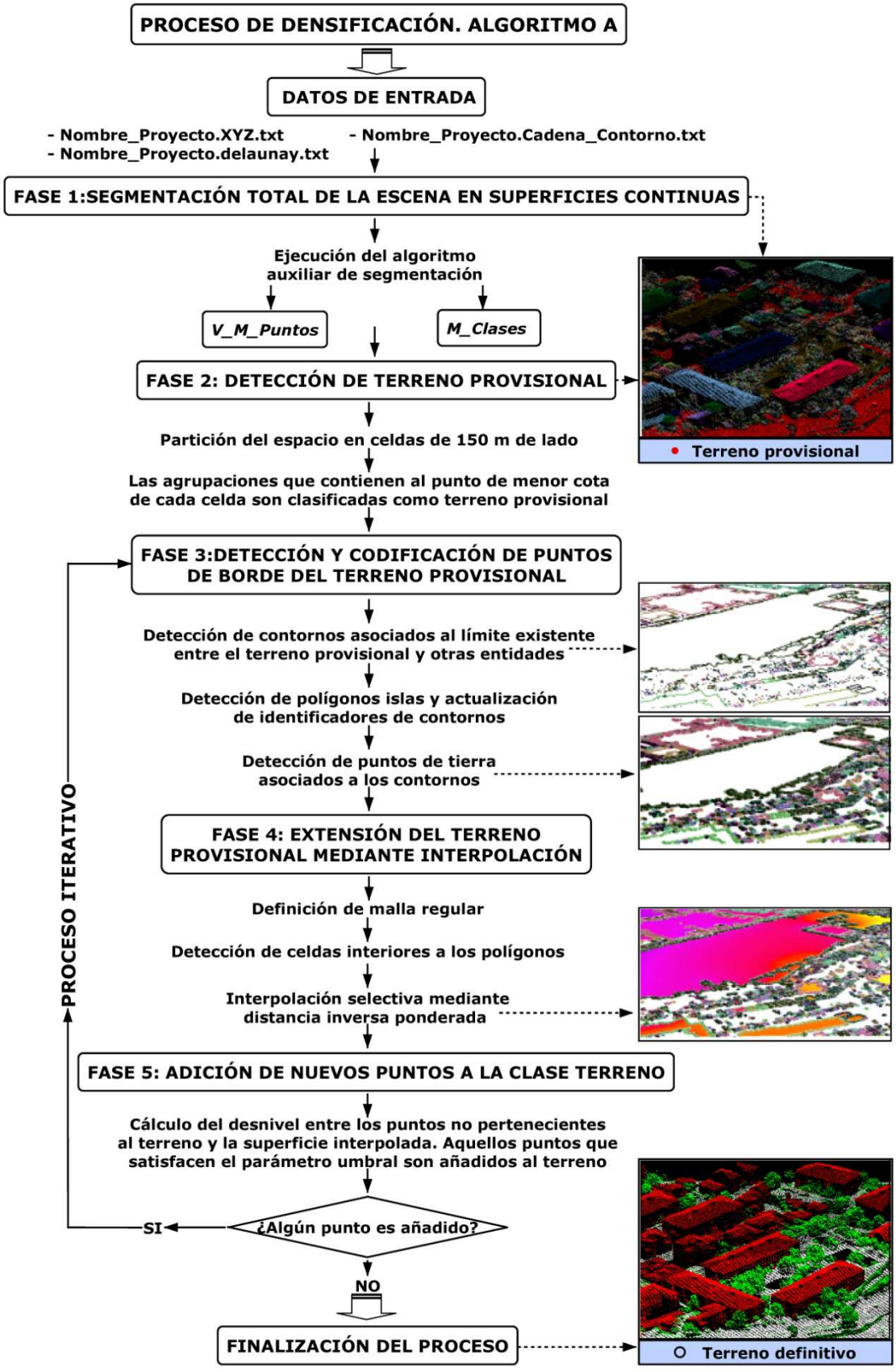


Figura 47. Diagrama de flujo para detectar la clase terreno. Algoritmo A.

A continuación se describe de forma detallada las distintas fases de las que consta el algoritmo desarrollado.

III.4.2.1. Carga de datos de entrada y codificación de datos

La función programada realiza en primer lugar la apertura y la carga de los datos de entrada. Estos datos se corresponden con los ficheros impresos tras el proceso de importación descrito con anterioridad, en concreto, los datos almacenados son los siguientes:

Nombre_Proyecto.XYZ.txt: Fichero de texto con tres columnas correspondientes a las coordenadas X,Y,Z de la nube de puntos. Para cada punto, el número de registro se corresponde con el identificador del punto LiDAR.

Nombre_Proyecto.delaunay.txt: Fichero de texto con la triangulación de Delaunay generada en el proceso de importación. Se trata de una matriz de tres columnas donde cada fila se corresponde con un identificador del triángulo de la triangulación. En las diferentes columnas se definen los tres vértices de cada triángulo, que se corresponderán con identificadores de puntos LiDAR de la nube de puntos.

Nombre_Proyecto.Cadena_Contorno.txt: Fichero de texto con los puntos LiDAR pertenecientes al contorno límite de la zona de estudio extraídos en forma de cadena poligonal en el sentido de las agujas del reloj. Se trata de un vector con los identificadores de los puntos LiDAR.

Además de cargar los datos de entrada, en esta parte del proceso el algoritmo realiza la codificación de datos en tiempo de ejecución. Se trata de la codificación en triángulos ya descrita (Apartado III.1.2.2), donde se genera un arreglo bidimensional definido mediante un doble puntero (*A_Code_Triangulos*). De forma que cada punto LiDAR tendrá un vector asociado con los identificadores de los triángulos de la triangulación de Delaunay en los que está contenido. El sistema de codificación será de gran utilidad en el proceso de detección de los bordes de las diferentes entidades que conforman la escena así como en el proceso de extracción de contornos.

III.4.2.2. Segmentación total de la escena en superficies continuas

Un aspecto fundamental del algoritmo programado consiste en segmentar la escena en superficies continuas delimitadas por cambios bruscos en pendiente, con ello se consigue segmentar los datos en superficies continuas que pertenecen a un mismo objeto. Esto servirá para agilizar el proceso de densificación optimizando el tiempo de cómputo. Esta optimización viene asociada al hecho de que cuando un punto es clasificado como terreno, la totalidad del grupo al que pertenece se clasifica como tal, consiguiendo así reducir considerablemente el número de iteraciones en el proceso de densificación.

Como valores de retorno, el algoritmo auxiliar de segmentación (Algoritmo auxiliar. Apartado III.10.1) devuelve el vector *V_M_Puntos* que contiene a la totalidad de puntos LiDAR marcados con el identificador de región al que pertenecen, así como la matriz *M_Clases* donde los puntos pertenecientes a una misma agrupación vienen registrados de forma consecutiva. El resultado de la segmentación obtenida para las diferentes zonas test utilizadas se observa en la Figura 48.

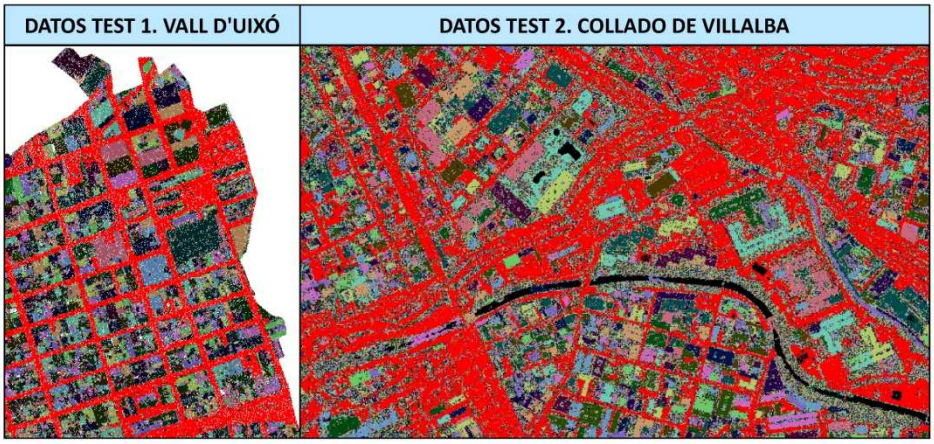


Figura 48. Segmentación de la escena. Datos test 1 y 2.

III.4.2.3. Detección del terreno provisional

Los métodos de densificación comienzan con la selección de un pequeño número de puntos pertenecientes al terreno obteniendo así una primera aproximación del mismo. Posteriormente, nuevos puntos son añadidos al terreno en sucesivas iteraciones siempre y cuando cumplan con un determinado umbral en diferencia de altitud con respecto a la superficie de referencia.

La selección del pequeño grupo de puntos pertenecientes al terreno que es utilizado para generar la superficie de referencia provisional en el inicio del proceso es arbitraria y depende de la metodología empleada. Una buena práctica consiste en dividir la zona de estudio en bloques o cuadrantes de igual tamaño y seleccionar el punto de menor cota como terreno provisional (Axelsson 2000). Esta metodología funciona correctamente cuando el terreno no tiene cambios bruscos en la pendiente del terreno, como es el caso de zonas urbanas. En este sentido, la metodología propuesta sigue el mismo principio y la zona de estudio automáticamente se divide en una cuadrícula regular de celdas de 150 metros de lado. Posteriormente el punto de menor cota de cada cuadrícula se detecta y la región a la que pertenece cada uno de ellos se añade al terreno provisional.

El algoritmo realiza una partición del espacio en celdas cuadradas de 150 metros de lado y altura infinita generando un arreglo bidimensional donde cada celda tendrá asociado un vector con los identificadores de puntos LiDAR que contiene (Apartado III.1.2.2). Posteriormente, el algoritmo recorre dichos vectores y marca el punto de menor altura de cada uno de ellos en un vector temporal. Finalmente, mediante la utilización de dicho vector temporal, los grupos a los que dichos puntos pertenecen son seleccionados como terreno provisional (Figura 44) y marcados con el valor "1" en un vector de marcadores (*V_Terreno_Provisional*) de igual dimensión que el número total de puntos LiDAR. Dicho vector previamente habrá sido reiniciado con el valor "-1", de esta forma los puntos pertenecientes al terreno tendrán registrado el valor "1" y el resto el valor "-1".

III.4.2.4. Proceso de densificación progresiva

El terreno provisional estará constituidos por puntos LiDAR pertenecientes a multitud de regiones conectadas. No obstante no podrá considerarse como terreno definitivo, ya que existirán algunas regiones aisladas rodeadas por vegetación o edificios que no habrán sido detectadas a pesar de pertenecer a la clase terreno (Figura 45).

Para solucionar este problema se implementa un método de densificación progresiva que añade de forma iterativa nuevos puntos pertenecientes al terreno. El método se basa en utilizar dicho terreno provisional para generar una superficie de referencia que se extiende bajo los diferentes objetos urbanos, añadiendo como nuevos puntos terreno aquellos cuya diferencia de altitud con respecto a la superficie de referencia es menor que un cierto umbral. El proceso es iterativo y en cada iteración los nuevos puntos añadidos al terreno servirán para actualizar la superficie de referencia, de forma que el proceso finaliza cuando ningún nuevo punto es añadido a la clase terreno.

La gran mayoría de métodos de densificación utilizan la triangulación como superficie de referencia, no obstante, con el fin de optimizar el tiempo de cómputo se opta por utilizar una interpolación selectiva mediante ponderación inversa a la distancia, siendo este punto, uno de los aspectos más novedosos del método propuesto.

Los diferentes pasos que realiza el algoritmo en el proceso de densificación se detallan a continuación.

- **Detección de puntos pertenecientes al borde o límite del terreno provisional**

La superficie de referencia se extenderá únicamente sobre los huecos asociados a los segmentos o grupos que no han sido previamente clasificados como terreno. Para ello será necesario extraer los contornos de dichas zonas en forma de polígono cerrado. Este paso será necesario para poder detectar posteriormente los puntos que pertenecen a cada uno de los huecos y poder aplicar así una interpolación selectiva de forma óptima.

Los pasos que realiza el algoritmo para la extracción de contornos son los siguientes:

a) Detección y extracción de contornos asociados al límite existente entre el terreno provisional y otras entidades

Tal y como se explicará con mayor detalle, resulta esencial detectar los puntos LiDAR que pertenecen a cada hueco que contiene regiones que no han sido clasificados como terreno, en adelante.

El primer paso consiste en detectar los bordes de las diferentes entidades, es decir, detectar aquellos puntos LiDAR que son el límite entre el terreno provisional y las regiones que no han sido clasificados como tal. Para ello se partirá del terreno provisional detectado con anterioridad y se comprobarán las conexiones de las aristas de la triangulación para cada uno de estos puntos.

El enfoque consiste en determinar que puntos pertenecen al límite del terreno provisional así como al límite de otras entidades. Para ello, se comprobarán las conexiones de las aristas de la triangulación de Delaunay para cada uno de los puntos del terreno provisional, de tal forma que si un punto perteneciente al terreno está conectado mediante una arista de la triangulación a un punto no clasificado como tal se considerará punto límite del terreno, siendo este procedimiento análogo para detectar los puntos límite de las entidades no pertenecientes al terreno.

Para registrar los puntos pertenecientes al límite del terreno y otras entidades se generará un vector de marcadores de igual dimensión que el número total de puntos LiDAR existentes en la nube de puntos (*V_Marca_Bordes*). Este vector se reiniciará con el valor "-1". Posteriormente, se comprobará para cada punto clasificado como terreno provisional las conexiones de sus aristas de la triangulación utilizando el arreglo de codificación en triángulos (*A_Code_Triangulos*). En caso de que alguna de las aristas conecte el punto con uno no clasificado como terreno, el punto del terreno será marcado con el valor "1",

mientras que el punto no perteneciente al terreno será marcado con "2". Finalmente, se marcarán todos los puntos pertenecientes al terreno provisional con el valor "0", siempre y cuando no pertenezcan al borde.

Con este procedimiento se consigue un vector de igual dimensión que la totalidad de puntos LiDAR (*V_Marca_Bordes*). Donde los puntos pertenecientes al límite del terreno estarán marcados con "1", los puntos pertenecientes a los límites de otras entidades con "2", los puntos pertenecientes al terreno provisional con "0" y el resto de puntos con el valor "-1". En la Figura 49 se aprecian los diferentes puntos contenidos en el vector *V_Marca_Bordes* representados en diferente color en función del tipo de borde y la entidad a la que pertenecen (Figura 49a). Además, una imagen de detalle (Figura 49b) muestra las conexiones entre los puntos LiDAR asociadas a las aristas de los triángulos pertenecientes a la triangulación de Delaunay. Estas conexiones son utilizadas para detectar los bordes o límites, basta con detectar aquellas aristas que conectan puntos del terreno provisional con puntos no clasificados como tal (aristas verdes). Para facilitar la notación, a este tipo de aristas las denominaremos a partir de ahora aristas mixtas.

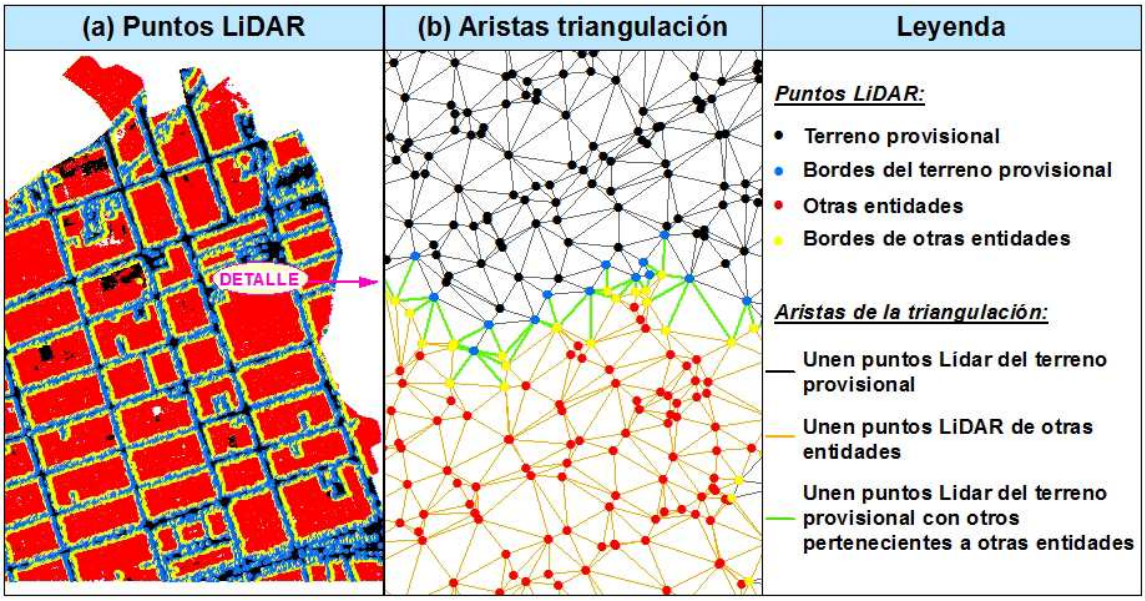


Figura 49. Detección de bordes. Clasificación del terreno.

b) Extracción de los contornos asociados al terreno provisional

Para extender la superficie de referencia bajo aquellas entidades que no han sido detectadas como terreno se utilizará una interpolación selectiva. Esta interpolación obtendrá como resultado un GRID (malla regular interpolada). Para ello será necesario determinar que celdas de dicha malla están contenidas en cada uno de los huecos a interpolar. En este sentido, se hace necesario extraer los contornos de los diferentes objetos urbanos que constituyen la zona de estudio. Concretamente serán extraídos en polígonos cerrados donde los vértices estarán ordenados en el sentido de las agujas del reloj. Este último aspecto, será esencial para poder establecer posteriormente propiedades y relaciones topológicas que serán utilizadas por otros algoritmos. En el caso que nos ocupa, se tratará de determinar si existe la propiedad de inclusión entre cada una de las celdas a interpolar del GRID los diferentes polígonos que determinan el límite de las entidades no clasificadas como terreno provisional.

La extracción de contornos se realizará partiendo del vector *V_Marca_Bordes*, donde estarán clasificados los diferentes puntos LiDAR en función de si pertenecen al terreno, a otras entidades, o al borde o límite de cualquiera de ellas (Figura 49). Este vector será

utilizado como dato de entrada por el algoritmo de extracción de contornos (Algoritmo auxiliar. Apartado III.10.2) obteniendo así un arreglo (*M_Coord_Contor*) definido mediante la clase *Vector* disponible en C++. Este arreglo constituirá una matriz de tres columnas, donde en la primera columna vendrá el identificador de contorno y en las dos restantes las coordenadas de cada uno de los vértices del polígono.

El algoritmo desarrollado extrae el contorno a partir del cálculo de las coordenadas de los puntos medios de las aristas mixtas, entendiendo por aristas mixtas aquellas aristas de la triangulación que conectan puntos de borde de distintas entidades. Además, la aplicación sigue los principios del algoritmo de Weiler-Atherton para el recorte de polígonos y es capaz de detectar la intersección de cada contorno con el polígono límite de la zona de estudio, consiguiendo así obtener contornos poligonales cerrados incluso para aquellos objetos de los que solamente se dispone de datos LiDAR de una parte de los mismos. La descripción detallada del algoritmo de detección y extracción de contorno se encuentra en el Apartado III.10.2. A continuación se puede observar una imagen (Figura 50) del resultado obtenido tras aplicar el algoritmo al terreno provisional. Se trata de una zona perteneciente a los datos Test 2 (Collado de Villalba) donde los diferentes contornos poligonales cerrados están dibujados en distinto color dependiendo de su identificador.

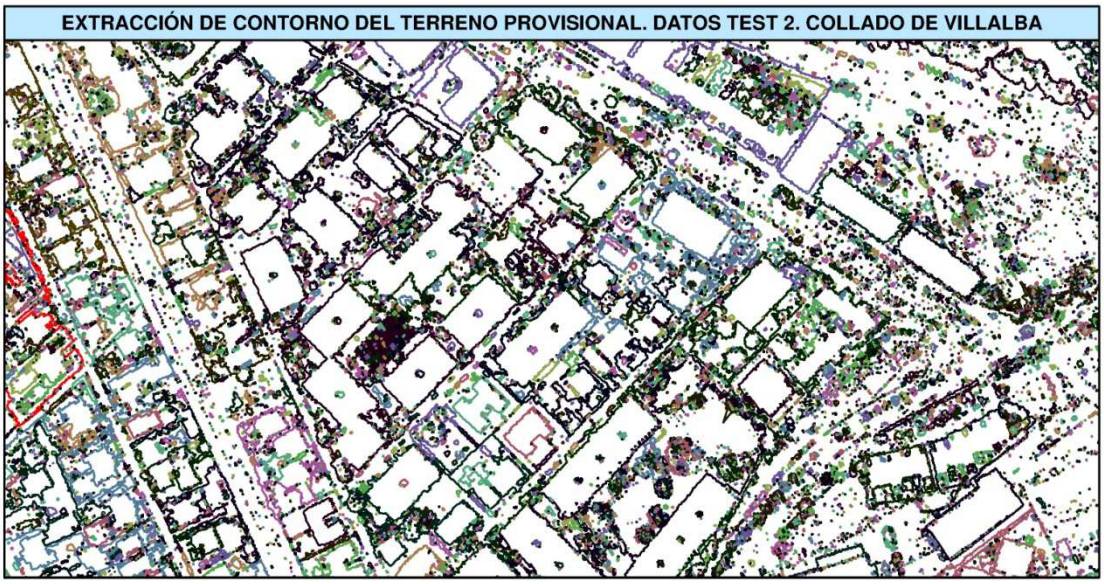


Figura 50. Contorno del terreno provisional para una zona de los datos test 2.

Además, el algoritmo de extracción de contornos (Algoritmo auxiliar. Apartado III.10.2) también generará un vector (*V_Id_Bordes_Entidad_A*) con los puntos de borde de los diferentes objetos que constituyen la entidad cuyo contorno desea extraerse. En el caso que nos ocupa, se tratará de un vector de igual dimensión que el número total de puntos LiDAR donde los puntos de borde del terreno tendrán el mismo identificador que su contorno poligonal asociado.

En la Figura 51 aparecen los puntos de borde de la entidad terreno en diferente color en función de su identificador. Dicho identificador coincidirá con el del polígono de contorno asociado. Ésto se aprecia en la figura de detalle, donde los contornos poligonales y los puntos de borde asociado están representados con el mismo color al poseer el mismo identificador.

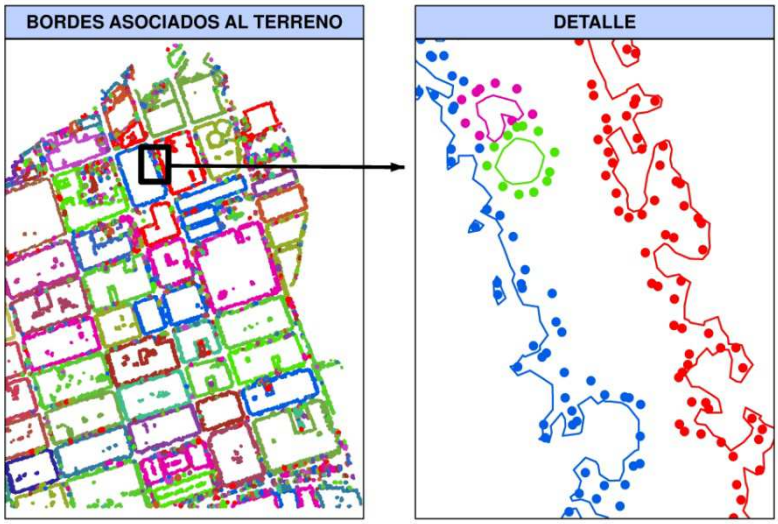


Figura 51. Asociación entre puntos de borde y contornos. Detección del terreno.

c) Detección de polígonos isla (polígonos interiores a uno dado)

Con la metodología propuesta en el apartado anterior dispondremos del contorno del terreno provisional, donde cada contorno tendrá asociado un identificador diferente. Este último aspecto ocasionará problemas en el posterior proceso de interpolación selectiva, ya que pueden existir contornos contenidos completamente dentro de otro polígono exterior asociados a patios interiores de edificios, así como a terreno rodeado por árboles y demás entidades. En estos casos, un mismo hueco estará definido por distintos contornos que poseen identificadores diferentes. Dado que la interpolación selectiva se basará en determinar los puntos de bordes asociados a un mismo hueco mediante el análisis de los identificadores de contorno, será necesario en estos casos utilizar un mismo identificador de polígono.

En la Figura 52 aparecen los contornos asociados al terreno representados en diferente color en función de su identificador. En este sentido, en los casos donde existen patios interiores, los polígonos asociados a dichos contornos tienen un identificador diferente al del contorno exterior del edificio a pesar de pertenecer al mismo objeto y por tanto, aparecen con color diferente (Figura 52a). Por ello, se hace necesario desarrollar una metodología que detecte estos polígonos interiores (polígonos isla) y que modifique sus identificadores para asignarles el mismo que posee su contorno exterior asociado (Figura 52b).

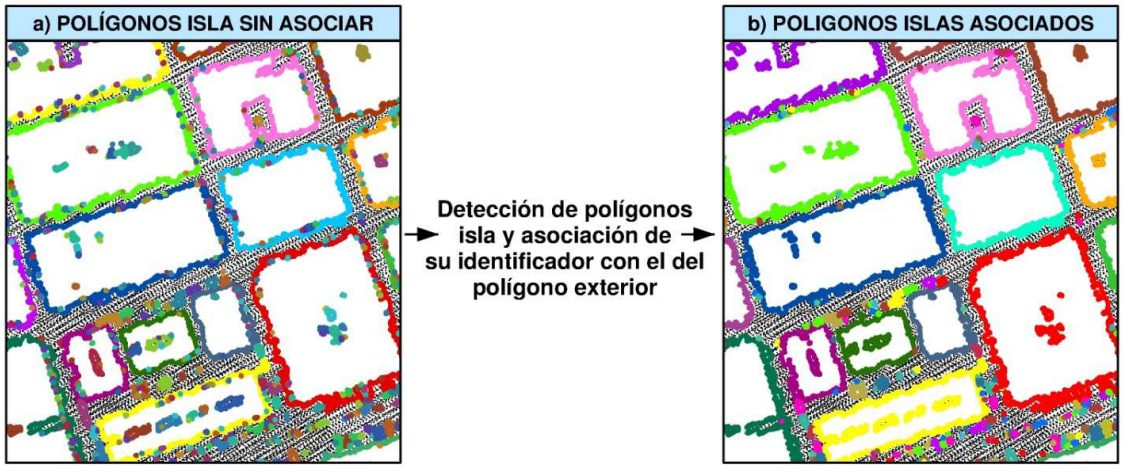


Figura 52. Fusión de identificadores en polígonos isla. Detección del terreno. Datos test 1.

Para determinar la propiedad de inclusión entre polígonos, el algoritmo se basa en que no existe posibilidad de intersección o solape entre ellos, es decir, dados dos polígonos cualquiera pertenecientes al contorno de dos objetos de la zona estudio, dichos polígonos únicamente pueden ser exteriores o interiores entre sí, pero en ningún caso experimentar intersección alguna.

Gracias a esta premisa, el problema de determinar si un polígono es interior a otro dado se reduce a determinar la propiedad de inclusión entre un punto y un polígono, siendo el punto cualquier vértice del polígono cuya propiedad de inclusión desea analizarse.

Dado un polígono *Z2* cuya propiedad de inclusión desea analizarse con respecto a un segundo polígono *Z1*, y sea *V* un vértice cualquiera del polígono *Z2*. Para averiguar si *Z2* es interior a *Z1* bastará con determinar si el vértice *V* es interior a *Z1*, ya que si esta premisa se cumple, entonces obligatoriamente *Z2* está contenido por *Z1*. Todo ello puede apreciarse en la Figura 53.

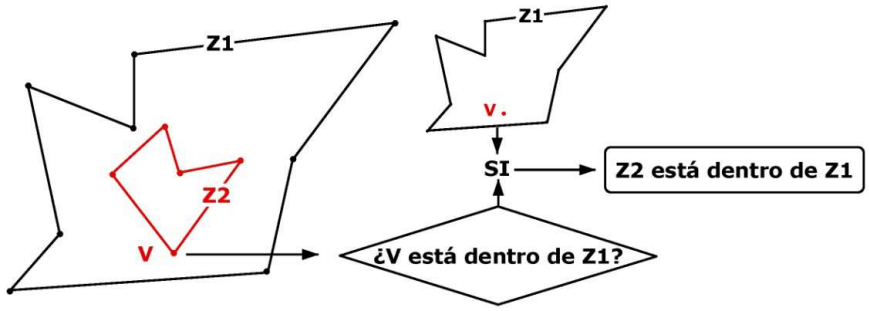


Figura 53. Detección de un polígono interior a otro dado. Detección del terreno.

Para verificar la propiedad de inclusión de un punto con respecto a un polígono dado existen numerosos algoritmos en el ámbito de la geometría computacional (algoritmos basados en el teorema de la curvatura de Jordan, algoritmo radial...). De entre todos ellos se opta por utilizar un algoritmo basado en el teorema de curvatura de Jordan combinado con una subdivisión plana del espacio que permite codificar las aristas de los polígonos y reducir así considerablemente el número de cálculos (Algoritmo auxiliar. Apartado III.10.3). Básicamente, para determinar si un punto *P* está dentro de un polígono *Z*, el algoritmo define la semirrecta horizontal que tiene como origen el punto *P* y que se extiende hacia $+\infty$ en la dirección del eje de abscisas. Seguidamente se contabiliza el número de intersecciones que tiene dicha semirrecta con el polígono *Z* en cuestión. Finalmente, si el número de intersecciones es par significará que el punto está fuera del polígono, mientras que si es impar significará que está dentro. La metodología completa del algoritmo, así como las distintas fases de las que consta aparecen descritas de forma detallada en el Apartado III.10.3.

El algoritmo (Algoritmo auxiliar. Apartado III.10.3) determina la propiedad de inclusión entre una serie de puntos y un conjunto de polígonos. En este caso, como deseamos verificar si existen polígonos islas que son interiores a otros, bastará con seleccionar un vértice de cada polígono y verificar si el vértice está contenido dentro de algún otro polígono de la zona de estudio. Concretamente la aplicación almacenará las coordenadas del primer vértice perteneciente a cada polígono en una matriz denominada *M_Puntos*. Dicha matriz se constituirá mediante la localización del primer vértice de cada polígono dentro de la matriz *M_Coord_Contor* donde vienen definidos la totalidad de los contornos de la zona de estudio. Además, se generará un vector de apuntadores (*V_Apuntadores_Contor*) de igual dimensión que el número de registros de *M_Puntos* donde para cada punto se almacenará un apuntador a su localización dentro de la matriz *M_Coord_Contor*.

El algoritmo continua mediante el envío de la matrices *M_Puntos* y *M_Coord_Contor* a la aplicación auxiliar que determina la propiedad de inclusión entre puntos y polígonos (Algoritmo auxiliar. Apartado III.10.3), obteniendo finalmente un arreglo bidimensional denominado *A_Puntos_Interiores_Pol*. En este arreglo vendrán indicados los puntos LiDAR del vector *M_Puntos* que pertenecen al interior de un polígono o contorno y estarán marcados con el identificador de contorno que los contiene. Finalmente, el algoritmo recorrerá dicho arreglo (*A_Puntos_Interiores_Pol*) y detectará los vértices que están contenidos dentro de un polígono, de tal forma que a partir del vector de apuntadores (*V_Apuntadores_Contor*) será posible acceder a la totalidad de vértices pertenecientes al polígono isla (polígono interior) y actualizar el identificador dentro de la matriz de contornos (*M_Coord_Contor*).

Además, una vez actualizados los polígonos isla de forma que posean el mismo identificador que su polígono exterior asociado (Figura 52b), se procederá a actualizar el vector V_Id_Bordes_Entidad_A donde vienen marcados los puntos LiDAR pertenecientes al terreno con el identificador de contorno asociado. Esta actualización será esencial para poder realizar una correcta interpolación selectiva y extender el terreno provisional por debajo de los objetos no clasificados como terreno en el proceso de densificación.

El diagrama de flujo que comprende las diferentes fases que ejecuta el algoritmo para detectar puntos de borde asociados al terreno provisional es el siguiente (Figura 54).

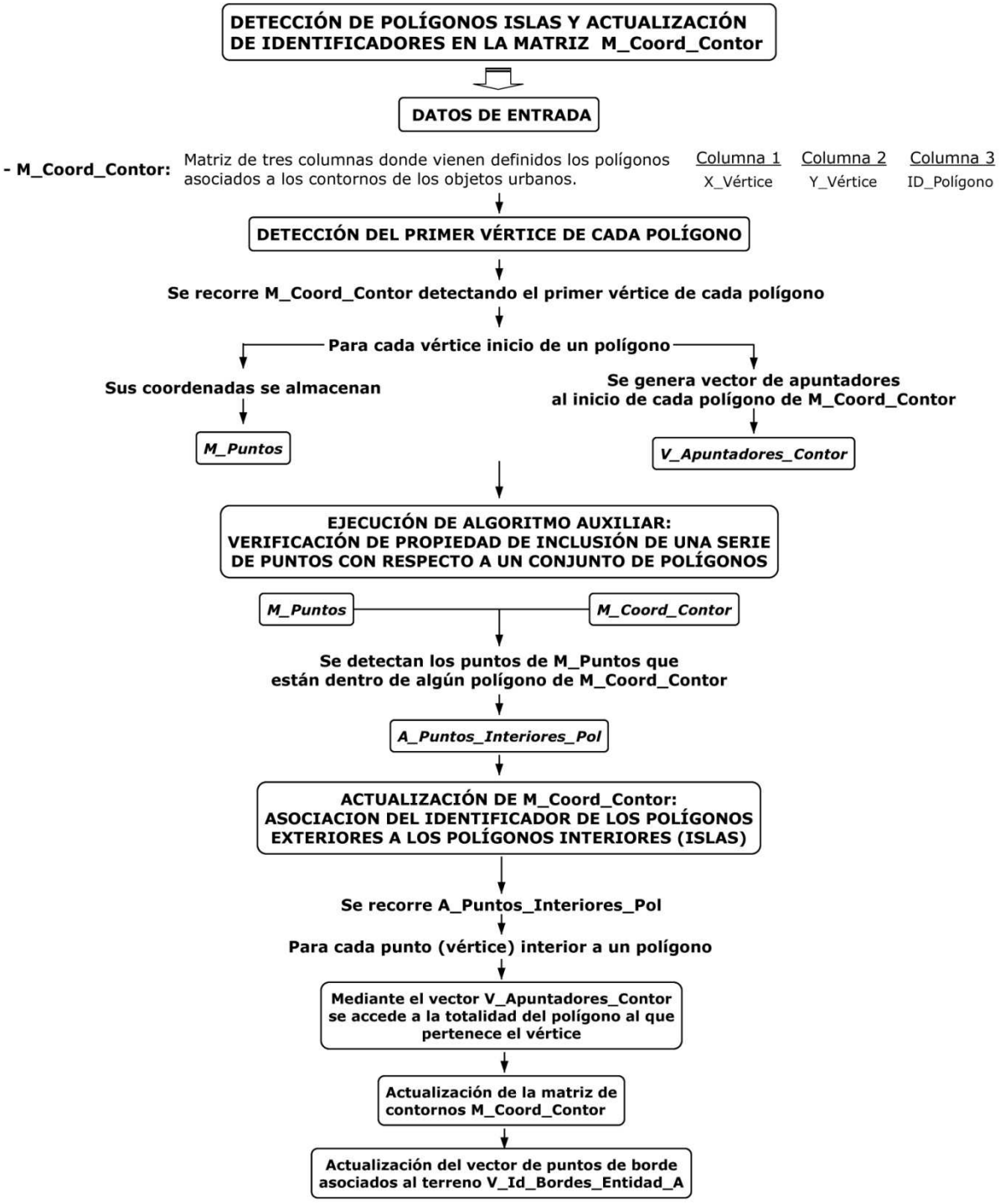


Figura 54. Diagrama de flujo del proceso para detectar polígonos islas.

- **Extensión del terreno provisional bajo los grupos no detectados como tal mediante interpolación selectiva**

Llegado a este punto dispondremos de los contornos que delimitan el terreno provisional de cualquier otro segmento o agrupación que no ha sido clasificado como tal. Además, dispondremos de los puntos de borde del terreno provisional asociados a cada uno de dichos contornos. Estos datos servirán para prolongar el terreno provisional bajo las agrupaciones no clasificadas como tierra mediante el uso de una interpolación selectiva.

a) Definición de la malla regular (GRID)

La malla vendrá constituida por celdas cuadradas de igual tamaño y recubrirá la totalidad de la zona de estudio. La anchura de las celdas la definirá el usurio; en caso contrario el algoritmo tomará por defecto un tamaño de celda de 0,5 m de lado.

Dado que la malla debe de contener por completo a la zona de estudio, el primer paso será redimensionar la malla para que cumpla esta premisa, de forma que el algoritmo calculará el número de filas y columnas a partir de la siguientes expresiones:

$$N\_Filas=((X\_max-X\_min)/ (Tamaño\_Celda))+1;$$
$$N\_Columnas=((Y\_max-Y\_min)/ (Tamaño\_Celda))+1;$$

Donde:

- N_Filas: Número total de filas de la malla.
- N_Columnas: Número total de columnas de la malla.
- X_max: Coordenada X máxima de la zona de estudio.
- X_min: Coordenada X mínima de la zona de estudio.
- Y_max: Coordenada Y máxima de la zona de estudio.
- Y_min: Coordenada Y mínima de la zona de estudio.
- Tamaño_Celda: Longitud del lado de las celdas (0,5 metros por defecto).

La malla se almacenará en una matriz donde todas las celdas se reiniciarán tras su definición con el valor -1000. Este valor será tomado como valor nulo en los posteriores cálculos.

b) Proceso de interpolación selectiva

La interpolación se llevará a cabo solamente en el interior de los polígonos que definen el contorno de las entidades que no pertenecen al terreno. Para ello, se realizará la interpolación de cada polígono contenido en *M_Coord_Contor* de forma independiente.

Se tomará como ejemplo una zona perteneciente a los datos test 2, donde la primera iteración del proceso de densificación en la detección del terreno ha sido ejecutada (Figura 55). Tras detectar el terreno provisional y extraer el borde y contornos asociados se extiende el terreno sobre dichos contornos mediante una interpolación de distancia inversa ponderada (IDW). Dado el conjunto *Z* de polígonos que delimitan el terreno provisional de otras agrupaciones no clasificadas como tal (Figura 55b) y sea *Z1* un polígono concreto del conjunto *Z* (Figura 51c). El algoritmo genera una malla regular auxiliar *M1* que contiene el polígono *Z1* (Figura 51d) siendo el proceso de definición análogo al descrito en el apartado anterior. Posteriormente se determinan las celdas que están contenidas en el interior del polígono *Z1*. Para ello, se recorre la totalidad de las celdas existentes en *M1* calculando las coordenadas del centro de cada una de ellas y almacenándolas en una nueva matriz denominada *M_Puntos_M***. S**eguidamente la información asociada al polígono *Z1* que está contenida en la matriz *M_Coord_Contor* se almacena en una matriz auxiliar *M_Coord_Contor_Z1*. Finalmente tanto la matriz con las coordenadas del centro de las celdas *M_Puntos_M* como la matriz *M_Coord_Contor_Z1* que contiene las coordenadas de los vértices de los polígonos *Z1* se envían al algoritmo auxiliar que determina la propiedad de inclusión de una serie de puntos con respecto a un conjunto de polígonos (Algoritmo auxiliar. Apartado III.10.3), consiguiendo así verificar el conjunto de celdas cuyo centro se encuentra en el interior del polígono *Z1* (Figura 55e).

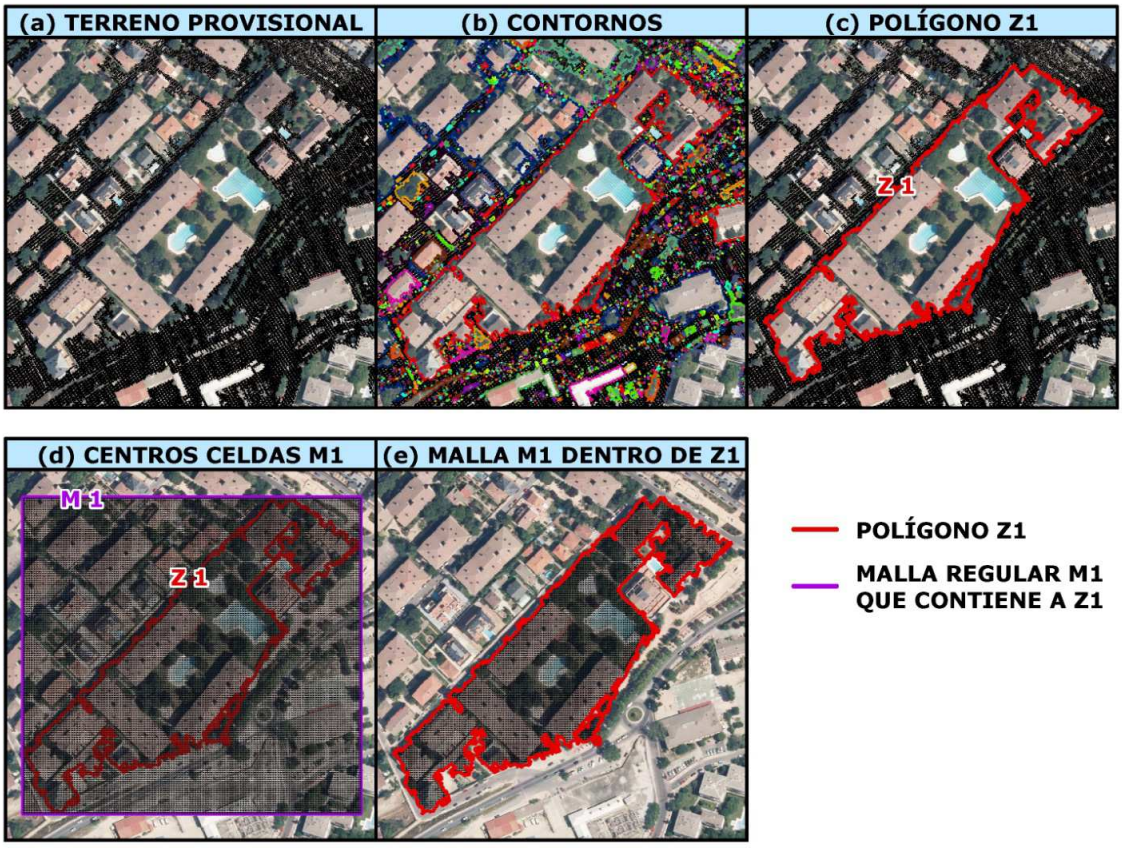


Figura 55. Generación de malla auxiliar para un contorno concreto. Proceso de prolongación del terreno mediante interpolación selectiva. Detección del terreno.

El siguiente paso consiste en interpolar las celdas de *M1* contenidas en *Z1*. Para este fin, una interpolación mediante distancia inversa ponderada (IDW) se ejecuta tomando como datos de entrada la totalidad de puntos de tierra asociados al borde del polígono *Z1*. La interpolación inversa a la distancia es un método de interpolación computacionalmente efectivo y resulta eficaz al utilizar un número muy alto de puntos distribuidos en todas las direcciones asociados al contorno del terreno que envuelve a los polígonos. Dado que la interpolación utiliza un alto número de puntos distribuidos en todas las direcciones para un área relativamente pequeña (superficie de cada polígono), se consigue evitar el típico efecto de "ojos de buey" asociado a este tipo de interpolación. La altura de cada celda se obtiene a partir de la siguiente expresión:

$$z_j = \frac{\sum_i z_i / D_{ij}^B}{\sum_i 1 / D_{ij}^B}$$

Donde:

- i: Número de puntos del terreno asociados al contorno o polígono.
- Zi: Altura del conjunto de puntos del terreno asociados al polígono.
- Dij: Distancia existente entre el centro de cada celda y los distintos puntos de tierra asociados.

La interpolación mediante distancia inversa ponderada determina los valores de celda a través de una combinación ponderada linealmente de un conjunto de puntos de muestra.

Este método presupone que los puntos pertenecientes a la muestra tienen menor influencia cuanto mayor es la distancia a la celda cuyo valor desea obtenerse. En este sentido, el exponente de ponderación *B* controla la región de influencia. Un valor alto de dicho parámetro tiene como consecuencia una mayor influencia de los puntos próximos y viceversa. Cuando se interpolan datos LiDAR, los puntos más próximos a cada celda deben tener una mayor influencia en el cálculo de la cota final de la misma, especialmente en regiones con altas pendientes. Un valor de *B* igual a 3 se ha utilizado en el presente estudio. Sin embargo, en regiones montañosas este parámetro debería ser incluso mayor.

Tal y como se observa en la Figura 56 una vez detectadas las celdas de *M1* contenidas en el interior de *Z1* se obtiene el valor de altitud de dichas celdas mediante interpolación selectiva de ponderación inversa a la distancia utilizando los puntos de borde de terreno asociados al polígono *Z1* (Figura 56a). El proceso se repite para todos los polígonos del conjunto *Z*, consiguiendo así extender el terreno bajo las agrupaciones no detectadas como tal (Figura 56b y Figrua 56c) y obtener una malla interpolada *M* que contiene la extensión del terreno bajo la totalidad de agrupaciones no detectadas.

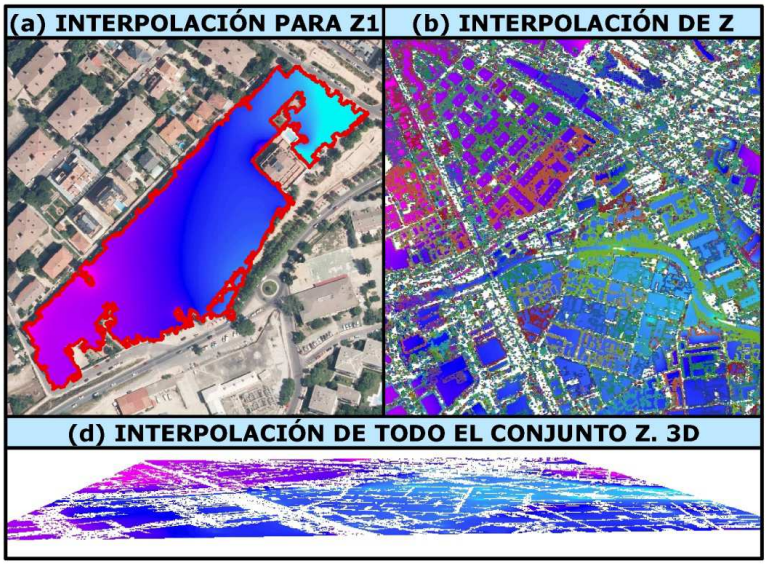


Figura 56. Interpolación selectiva en el proceso de densificación. Detección del terreno. Datos test 2 (Collado Villaba).

- **Adición de nuevos puntos a la clase terreno**

Los métodos de densificación se basan en añadir iterativamente nuevos puntos pertenecientes al terreno a la clase de terreno provisional hasta obtener el terreno definitivo. La adición de nuevos puntos al terreno provisional depende de la diferencia de altura entre el punto y la superficie interpolada auxiliar previamente calculada. En este sentido, el algoritmo implementado calcula la diferencia de altura de los puntos no clasificados como terreno y la malla regular previamente obtenida mediante interpolación, de forma que si la diferencia de altura es menor que el umbral establecido (Δh) el punto es marcado en un vector de marcadores como nuevo punto perteneciente a la clase terreno provisional. Finalmente el algoritmo recorrerá la totalidad del vector de marcadores almacenando el identificador de los nuevos puntos añadidos y clasificando como terreno la región a la que pertenecen cada uno de ellos.

A continuación se describe de forma detallada las distintas fases de las que consta el proceso.

a) Detección de nuevos puntos que cumplen el parámetro umbral en altitud (Δh)

En esta parte del proceso, el algoritmo utiliza como base distintas matrices y vectores generados en fases previas. Básicamente partirá del vector de marcadores *V_Terreno_Provisional* que contiene los puntos pertenecientes al terreno provisional marcados con el valor "1", así como de la matriz *M* donde viene definida la malla interpolada que constituye la extensión del terreno bajo la totalidad de regiones no clasificadas.

El algoritmo recorre la totalidad de los registros de *V_Terreno_Provisional* de tal forma que cuando detecta un punto *P* que no pertenece al terreno provisional (marcado con "-1") determina la celda sobre la que el punto se proyecta a partir de las siguientes expresiones:

$$N\_Fila\_Proyec=((X\_Punto-X\_min)/ (Tamaño\_Celda))$$
$$N\_Columnas\_Proyec=((Y\_max-Y\_Punto)/ (Tamaño\_Celda))$$

Donde:

- N_Fila_Proyec: Número de fila de la matriz *M* donde se encuentra la proyección del punto *P*.
- N_Columnas_Proyec: Número de columna de la matriz *M* donde se encuentra la proyección del punto *P*.
- X_min: Coordenada X mínima de la zona de estudio.
- Y_max: Coordenada Y máxima de la zona de estudio.
- Tamaño_Celda: Longitud del lado de las celdas de la malla definida en la matriz *M*.

Una vez calculada la celda donde se proyecta el punto *P* el algoritmo almacena la altura de la celda contenida en la matriz *M* en una variable denominada *Cota_Celda*. Seguidamente se verifica si el incremento de altura entre el punto y la celda es menor que el umbral *Δh*. En caso afirmativo el punto es clasificado como nuevo punto perteneciente al terreno provisional y marcado con el valor "2" en el vector *V_Terreno_Provisional*. El parámetro umbral *Δh* tiene un valor por defecto de 0,2 m; no obstante, puede ser modificado por el usuario. En el presente trabajo, tal y como se detalla en el posterior análisis de resultados, se ha ejecutado el algoritmo utilizando diferentes parámetros umbrales con el fin de evaluar la dependencia de la precisión alcanzada con respecto a los parámetros empleados.

El proceso se repite para la totalidad de puntos LiDAR marcados con "-1" en el vector *V_Terreno_Provisional*, de forma que cuando se ejecuta una iteración donde ningún nuevo punto es añadido, el terreno provisional pasa a considerarse como terreno definitivo y el proceso iterativo finaliza. Por el contrario, en caso de que algún punto se añada al terreno provisional el proceso continua tal y como se describe en el siguiente apartado.

b) Adición de nuevos puntos clasificados como terreno en el vector V_Terreno_Provisional

En este momento del proceso, se dispondrá de aquellos puntos cuya diferencia de altura respecto a la superficie interpolada es menor que el parámetro *Δh* marcados con el valor *"2"* en el vector *V_Terreno_Provisional*. Estos son los nuevos puntos detectados como terreno provisional. No obstante, dado que la escena ha sido segmentada en superficies continuas, la totalidad de la agrupación a la que pertenecen dichos puntos pertenecerá también al terreno y por tanto deberán clasificarse como tal. Para ello, el algoritmo recorre el vector *V_Terreno_Provisional* detectando aquellos puntos que han sido marcados con el valor "2" y posteriormente accede a los identificadores de los puntos que están contenidos en las mismas agrupaciones a partir de la utilización del vector *V_M_Puntos* y la matriz *M_Clases*, marcando finalmente la totalidad de los puntos pertenecientes a dichas regiones con "1" en el vector *V_Terreno_Provisional* consiguiendo así actualización del mismo.

Una vez añadidos las nuevas regiones detectadas a la clase terreno provisional, el proceso de densificación se repite desde su inicio y únicamente finaliza cuando ningún nuevo punto es añadido a la clase terreno, momento en el cual el terreno provisional se convierte en definitivo.

En la Figura 57 puede apreciarse el terreno provisional obtenido en la primera iteración (Figura 57a) así como los puntos añadidos al terreno en la segunda iteración (Figura 57b). Con un parámetro umbral en desnivel de 0,2 m, la clase terreno definitiva se obtiene para los datos test 1 (Vall d'Uixó) en seis iteraciones, mientras que para los datos tes 2 (Collado de Villalba) el algoritmo precisa de un total de nueve iteraciones.

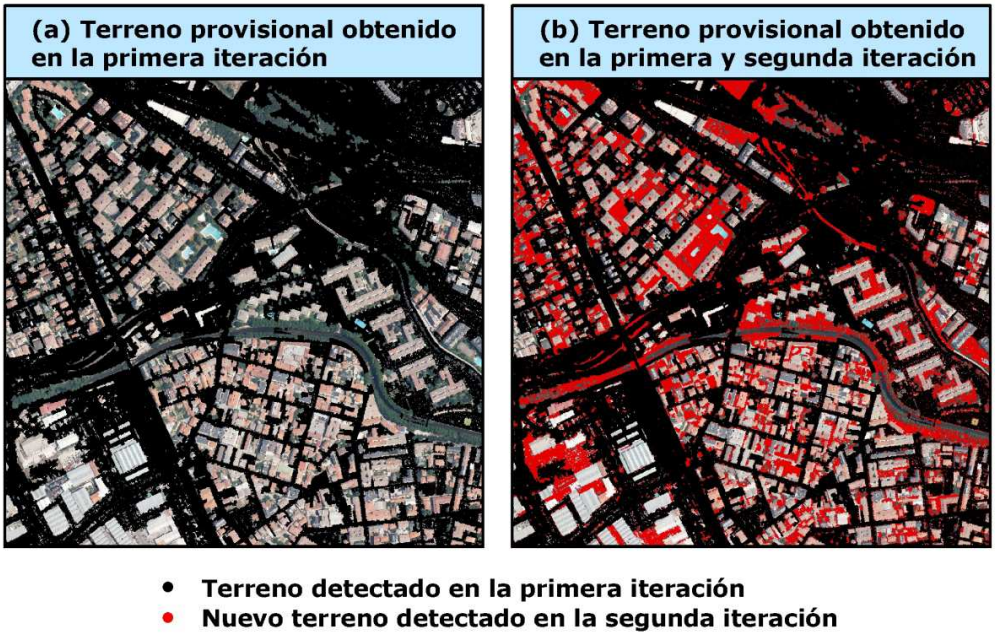


Figura 57. Terreno detectado en la primera y segunda iteración del proceso de densificación. Datos test 2.

III. 5. Detección automática de puntos pertenecientes al terreno. Algoritmo B: densificación progresiva mediante interpolación selectiva combinada con segmentación y aplicación de reglas de decisión

III.5.1. Motivación para el desarrollo del algoritmo y relación con los antecedentes

Tras ejecutar el algoritmo A para la detección del terreno en ambas zonas test, y con el fin de detectar posibles errores groseros en los resultados, la clase terreno es visualizada en su conjunto. Además, la clase obtenida se divide en cuadrantes de 150 m de lado que son visualizados en 3D junto con las ortofotos en ArcGIS 9.3. En esta primera fase de evaluación de resultados se localizan diferentes errores de clasificación. Principalmente, se trata de errores de clasificación por exceso donde puntos pertenecientes a edificios u otras entidades son clasificados erróneamente como terreno. Este tipo de errores es frecuente cuando se trata de zonas urbanas complejas y se hace uso de métodos basados en densificación progresiva. Los errores detectados pueden clasificarse en tres tipos.

- **Error tipo 1: selección errónea de agrupaciones como terreno provisional inicial**

Previo al método iterativo de densificación es necesario detectar el terreno de forma provisional con el fin de definir una superficie de referencia para iniciar el proceso. Para ello, resulta frecuente dividir las zonas de estudio en bloques y seleccionar como terreno el punto de menor cota de cada uno de ellos (Axelsson 2000; Lin y Zhang 2014). No obstante, esta práctica no está exenta de errores. Para empezar, el método precisa de un parámetro umbral asociado al tamaño de cada bloque. En este sentido, el tamaño del bloque debe de ser siempre mayor al edificio más grande de la zona de estudio. Consecuentemente, un error en la selección del valor de dicho parámetro se transmite en la inclusión de puntos pertenecientes a edificios dentro de la clase de terreno provisional. Por otro lado, puede darse el caso de que en zonas con altas pendientes, el punto de menor cota del bloque no pertenezca al terreno sino a un edificio (Figura 48).

Como ejemplo, dada una nube de puntos LiDAR a partir de la cual se genera un modelo virtual 3D (Figura 58a), tras dividir la zona en bloques o celdas de igual tamaño y altura infinita (Figura 58b), el punto de menor cota de cada uno de los bloques es seleccionado para definir el terreno provisional (Figura 58c). No obstante, al observar el bloque B 1-2 con detalle (Figura 58c) se aprecia que el punto de menor cota del mismo pertenece a un edificio en vez de al terreno. Ello se debe a que el edificio en cuestión está próximo a un cambio abrupto del terreno y por tanto resulta la parte más baja del bloque. La detección de un punto de edificio como terreno provisional llevará a la selección de la totalidad de puntos pertenecientes a la misma agrupación como terreno, con el error que ello conlleva (Figura 58e).

La existencia de grandes desniveles entre calles adyacentes no suele ser frecuente en entramados urbanos. Por tanto la selección de puntos de edificio como terreno por este motivo no constituye una fuente de error común en el proceso de clasificación. No obstante, la selección del parámetro umbral con respecto al tamaño de los bloques es realizada por un operador y puede ocasionar importantes errores en el resultado si la elección no es la adecuada.

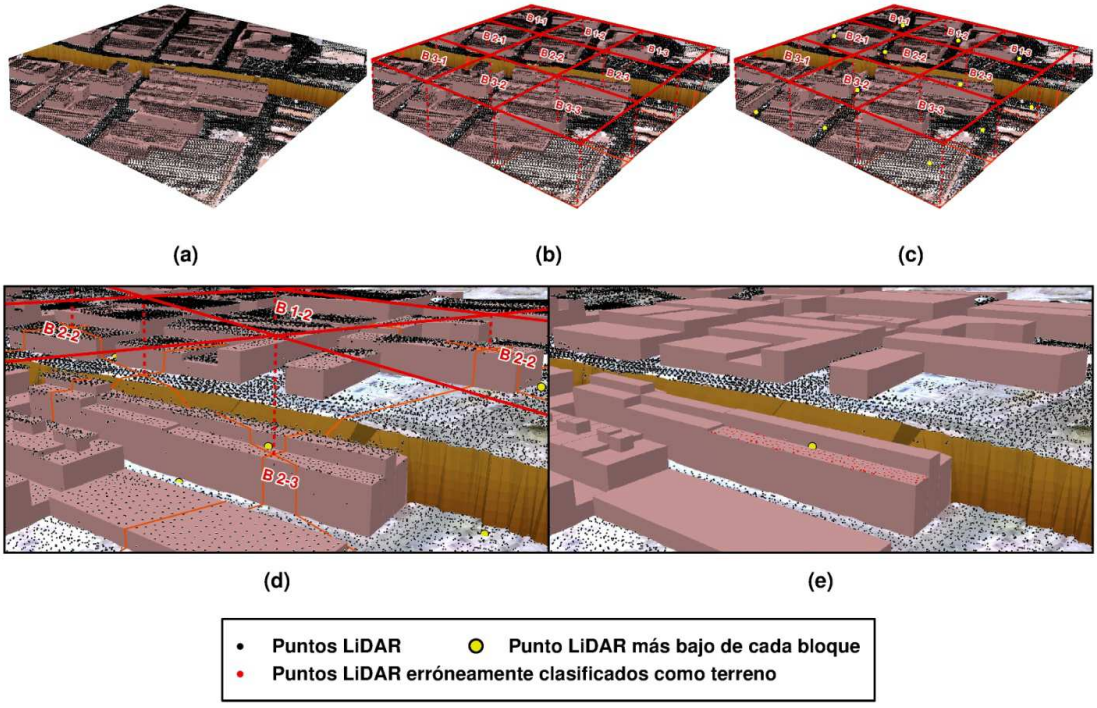


Figura 58. Error tipo 1. Elección errónea de edificios como terreno provisional.

- **Error tipo 2: puntos pertenecientes a objetos urbanos con baja altura son clasificados erróneamente dentro de la clase terreno**

El láser aerotransportado además de capturar puntos de edificios que pertenecen a planos de techo, también captura con frecuencia puntos pertenecientes a las fachadas de los mismos. Estos puntos en ocasiones están próximos al terreno y en consecuencia pueden ser incluidos en la clase terreno en el proceso de densificación debido a que la diferencia de altura con respecto a la superficie de referencia es menor que el umbral preestablecido. Todo ello se muestra en la Figura 59 donde se aprecia una zona concreta de los datos test 1 (Vall d'Uixó). En la primera de las imágenes (Figura 59a) se muestran los puntos capturados por el láser que pertenecen al terreno y a las fachadas de los edificios sobre un modelo 3D generado a partir de la nube de puntos LiDAR, mientras que en la segunda (Figura 59b) se observan los puntos pertenecientes a las fachada que han sido erróneamente clasificados como terreno tras aplicar el algoritmo de densificación. Además de puntos de fachada, vegetación de baja altura así como pequeños objetos urbanos pueden ser clasificados erróneamente como terreno.

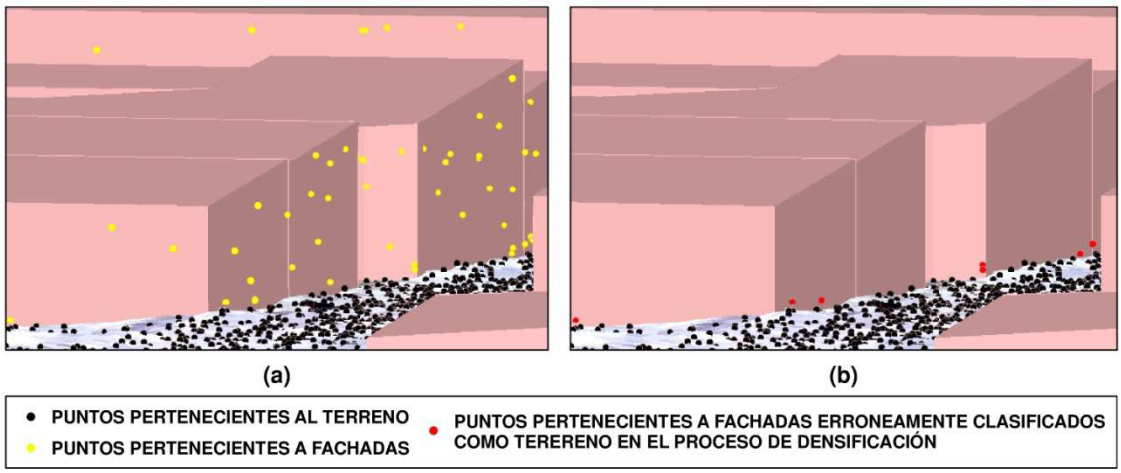


Figura 59. Error tipo 2. Datos test 1 (Vall d'Uixó)

- **Error tipo 3: edificios y planos de techo erróneamente clasificados como terreno en zonas con alta pendiente**

En ocasiones, cuando la pendiente de las calles es elevada, así como en solares que presentan terraplenes y taludes pronunciados, puede ocurrir que la cota del terreno sea mayor que la del plano de techo de un edificio próximo. En estos casos, el algoritmo de densificación prolonga la superficie de referencia (terreno provisional) por encima de los planos de techo de dichos edificios y consecuentemente los puntos LiDAR pertenecientes a estos edificios son erróneamente clasificados como terreno. En la Figura 60 se muestra un caso real de los datos test 1 (Vall d'Uixó). En la primera de las imágenes (Figura 60a) se observan los puntos capturados por el láser y el modelo 3D generado a partir de ellos. Se trata de una zona que presenta una parcela sin construir colindante a una calle que posee una pendiente pronunciada. En este caso, el solar tiene dos alturas conectadas mediante un terraplén y la altura superior posee una cota media de 158 m. Por otro lado, la altura del plano de techo de un edificio colindante a la parcela es más de 3 m inferior (Figura 60b). Esto provoca que tras detectar los puntos pertenecientes al terreno provisional en una primera iteración del proceso de densificación (Figura 60c), el algoritmo prolongue el terreno por encima del tejado del edificio (Figura 60d), detectando erróneamente los puntos del tejado como puntos pertenecientes al terreno (Figura 60e).

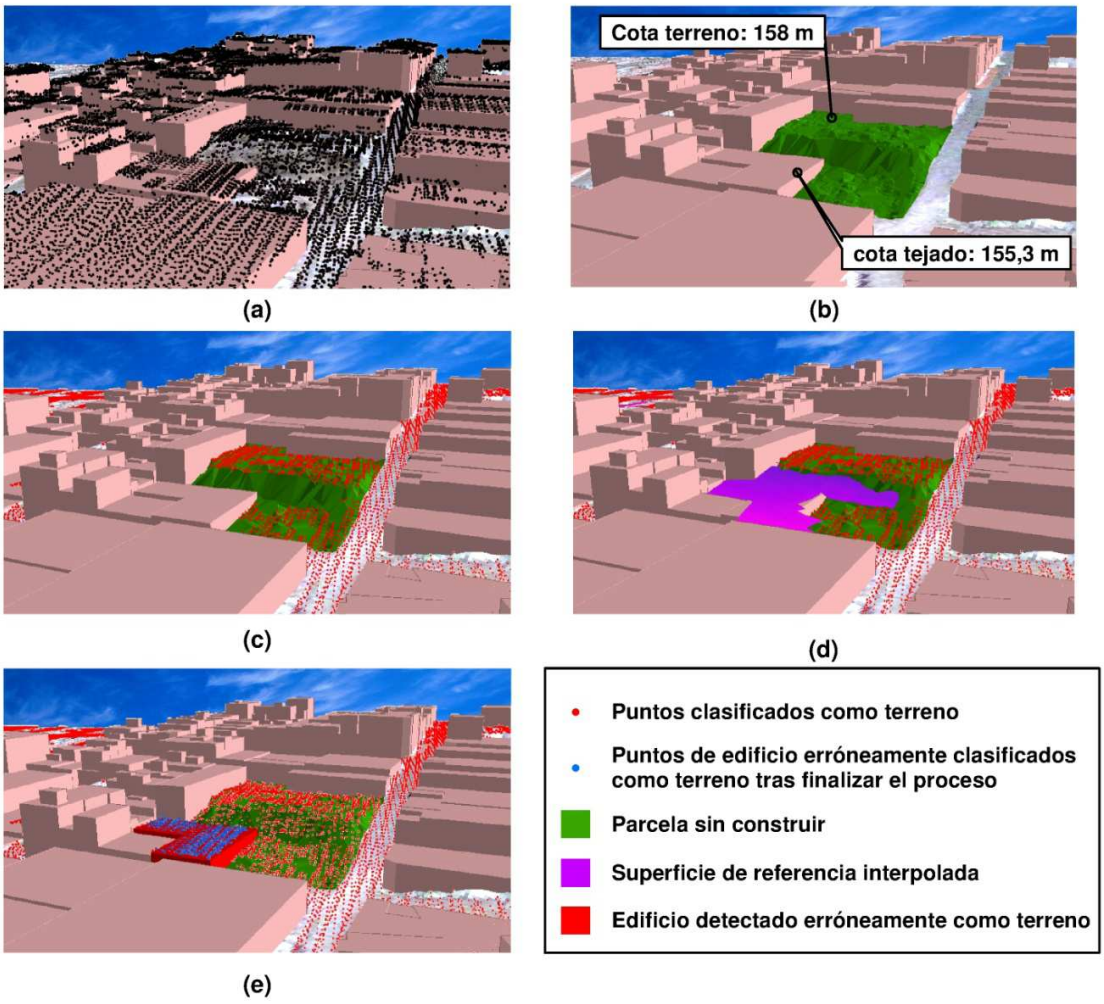


Figura 60. Error tipo3. Datos test 1 (Vall d'Uixó)

Alguno de los errores detectados han sido objeto de análisis en otros estudios, como la clasificación errónea de puntos pertenecientes a objetos de escasa altura como terreno (Error tipo 2). Con el fin de minimizar este tipo de error, Tovari y Pfeifer (2005) utilizaron un filtro basado en interpolación de superficies combinado con técnicas de segmentación. En primer lugar segmentaron la nube de puntos LiDAR y, posteriormente, los residuos se calcularon teniendo en cuenta cada segmento en vez de cada punto. Los residuos fueron utilizados para calcular los diferentes pesos a aplicar en el ajuste, donde la totalidad de puntos pertenecientes a una misma agrupación tenían asociado un mismo peso. Los resultados mostraron buenos resultados tanto para zonas urbanas como de montaña.

Un enfoque similar es el utilizado por Lin y Zhang (2014), que combinan la segmentación con un enfoque de densificación progresiva de TIN con el fin de minimizar este tipo de errores. En este caso, tras realizar la segmentación el proceso de densificación se llevaba a cabo, la diferencia básica con respecto al enfoque clásico de densificación es que en este caso el objeto de análisis no es el punto, sino las diferentes agrupaciones obtenidas tras la segmentación, de tal forma que una agrupación es añadida por completo a la clase terreno siempre y cuando la mayoría de los puntos que la componen son clasificados como tal en el proceso de densificación. Las conclusiones de este estudio muestran una clara reducción de los errores asociados a la clasificación de puntos pertenecientes a objetos de escasa altura como terreno (Error tipo 2).

En cualquier caso no se ha presentado una solución definitiva que permita eliminar por completo los errores de comisión asociados a puntos pertenecientes a objetos de baja altura clasificados como terreno (Errores tipo 2). Además, no existen investigaciones centradas en eliminar los errores groseros que produce el método de densificación en zonas urbanas con grandes pendientes (Errores tipo 3) así como trabajos que ofrezcan una solución a la clasificación de puntos pertenecientes a objetos urbanos como terreno provisional previamente a la aplicación del proceso de densificación (Error tipo 1). En este sentido, con el fin de dar solución a estos errores típicos de metodologías de densificación progresiva, se opta por desarrollar un nuevo algoritmo que combine el método de densificación con técnicas de segmentación tal y como realizan otros autores (Ekhtari et al. 2008 ; Pérez et al. 2012 ; Lin y Zhang 2014). No obstante, en este caso se pretende ir más allá y extraer la máxima información posible de las agrupaciones obtenidas tras la segmentación con el fin de descartar aquellos puntos que no pertenecen con certeza al terreno antes de comenzar con la densificación progresiva. Para ello, se ha diseñado un análisis con respecto al tamaño de las agrupaciones y la posición relativa existente entre ellas.

Dado que las diferentes agrupaciones obtenidas tras la segmentación representan las diferentes superficies continuas existentes en la escena, las agrupaciones con un gran tamaño estarán asociadas a grandes superficies. En consecuencia, las agrupaciones de mayor tamaño se corresponderán con calles interconectadas, mientras que pequeñas agrupaciones pertenecerán a vegetación y a otras entidades como planos de techo, automóviles, etc.

Además del tamaño de las regiones, la posición relativa existente entre las diferentes agrupaciones será objeto de análisis con el fin de determinar si una región es susceptible de pertenecer al terreno. Para ello, se partirá del supuesto de que las agrupaciones pertenecientes al terreno están situadas a menor altura que aquellas que pertenecen a otros objetos (edificios, árboles...). En este sentido, el algoritmo desarrollado calcula la diferencia de altura entre agrupaciones adyacentes con el fin de diferenciar aquellas que obligatoriamente pertenecen a objetos urbanos de aquellas que pueden pertenecer al terreno, tal y como realizan algunos filtros basados en segmentación (Lohmann 2002; Sithole 2005; Sithole y Vosselman 2005; Shen et al. 2012).

III.5.2. Descripción del algoritmo programado

El nuevo método desarrollado se basa en el algoritmo de densificación progresiva descrito en el apartado anterior (Algoritmo A. Apartado III.4), únicamente se añaden nuevas fases de procesamiento para analizar el tamaño y la posición relativa de las agrupaciones obtenidas tras la segmentación, con el fin de realizar una clasificación previa que evite errores de comisión en el posterior proceso de densificación. El objetivo del nuevo algoritmo es reducir o incluso eliminar los errores de tipo 1, 2 y 3 anteriormente comentados y que están asociados al método clásico de densificación.

El análisis se realizará y calculará de forma independiente para cada región o agrupación obtenida tras segmentar la escena en superficies continuas. . Además, se detectan las aristas mixtas de la triangulación que conectan puntos pertenecientes a cada agrupación con las agrupaciones adyacentes. En la Figura 61 se aprecia el proceso de detección de aristas mixtas, entendiendo por aristas mixtas aquellas aristas de la triangulación que conectan puntos pertenecientes a dos regiones diferentes.

Dada una nube de puntos LiDAR perteneciente a un entorno urbano (Figura 61a), la segmentación en superficies continuas se lleva a cabo (Figura 61b) (Algoritmo auxiliar. Apartado III.10.1). Seguidamente, para cada región se detectan aquellas aristas de la triangulación que conectan puntos pertenecientes a dicha región con las adyacentes (aristas mixtas) (Figura 61c). Estas aristas están asociadas al límite de las diferentes regiones. Por

tanto, conectan las distintas superficies continuas existentes en la escena. Ello conlleva que estén asociadas a discontinuidades en altitud y cambios bruscos de pendiente. Consecuentemente, resultan de gran utilidad para analizar la posición relativa de las distintas regiones con respecto a sus regiones adyacentes en cuanto a altitud se refiere.

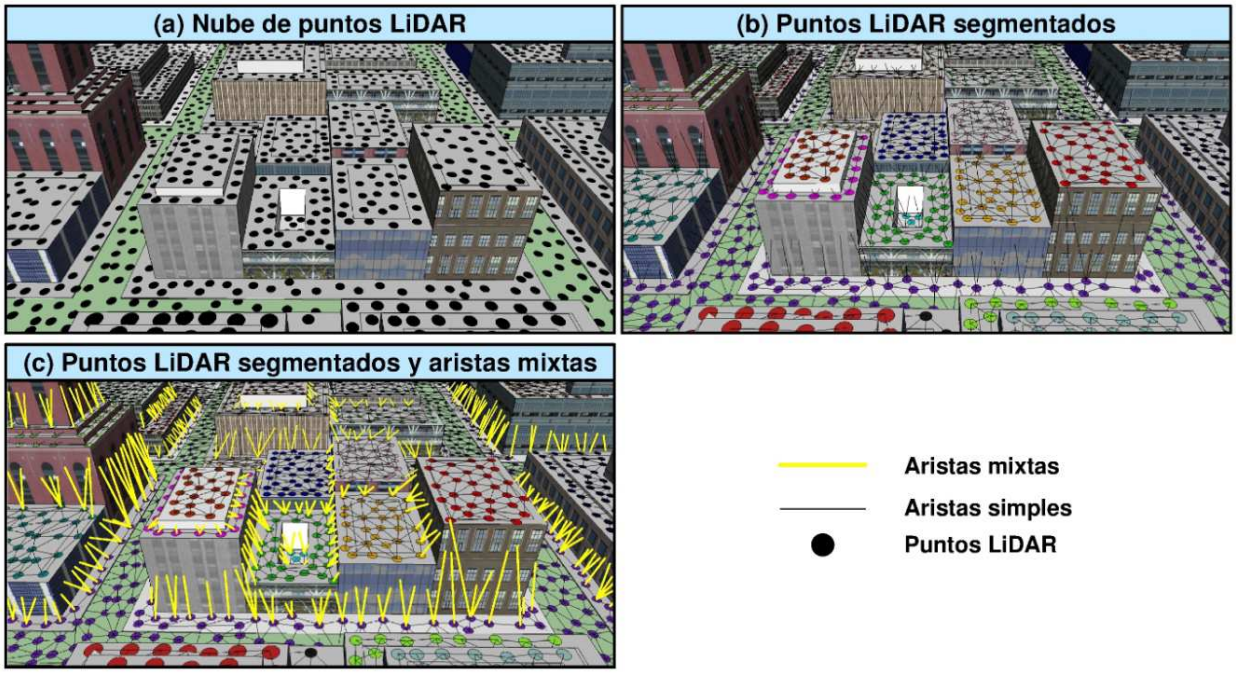


Figura 61. Detección de aristas mixtas para cada región segmentada. Modelo 3D.

Una vez detectadas las aristas mixtas de cada región o agrupación, dos parámetros umbrales entran a formar parte del proceso de clasificación de regiones: un parámetro umbral en desnivel (*UZ*) y un parámetro umbral en cuanto al porcentaje de aristas mixtas que sobrepasan dicho desnivel (*U%*). Considerando el ejemplo propuesto en la Figura 62, el análisis se realiza para tres regiones o agrupaciones distintas (agrupación 1, agrupación 2 y agrupación 3) (Figura 62a). Tras la detección de las aristas mixtas, el desnivel que definen los vértices adyacentes a cada una de ellas se calcula mediante la siguiente expresión:

$$Iz = Z(V\_Adyacente) - Z(V\_Objeto)$$

Donde:

- Z(V_Adyacente): es la coordenada Z del vértice constituido por el punto LiDAR perteneciente a la región adyacente.
- Z(V_Objeto): es la coordenada Z del vértice constituido por el punto LiDAR perteneciente a la región objeto de análisis.

Seguidamente, las aristas mixtas se clasifican en función de si su desnivel es mayor que el umbral en desnivel previamente establecido (*UZ*). En el ejemplo propuesto se ha utilizado un parámetro umbral en desnivel de 1,5 m. En este supuesto, la totalidad de las aristas mixtas pertenecientes a la primera de las agrupaciones (agrupación 1) sobrepasan dicho desnivel (aristas tipo 1), lo que significa que la totalidad de las agrupaciones adyacentes están situadas a una altura superior a 1,5 metros (Figura 62.b) con respecto a dicha región. El caso contrario se aprecia en la agrupación 3, donde la totalidad de los desniveles calculados son negativos y por tanto las agrupaciones adyacentes están situadas a una altitud inferior a 1,5 metros con respecto a la región objeto de análisis (Figura 62b) (aristas tipo 2). Por último en la agrupación 2 existen aristas que sobrepasan el parámetro umbral

(*UZ*) (aristas tipo 1) y aristas que no (aristas tipo 2), lo que significa que se trata de una agrupación cuyas superficies adyacentes están por encima y por debajo en cuanto a altitud. Así mismo, podría darse el caso de que alguna de las agrupaciones adyacentes estuviera a mayor altitud pero que no sobrepasará el parámetro umbral previamente establecido (Figura 62b).

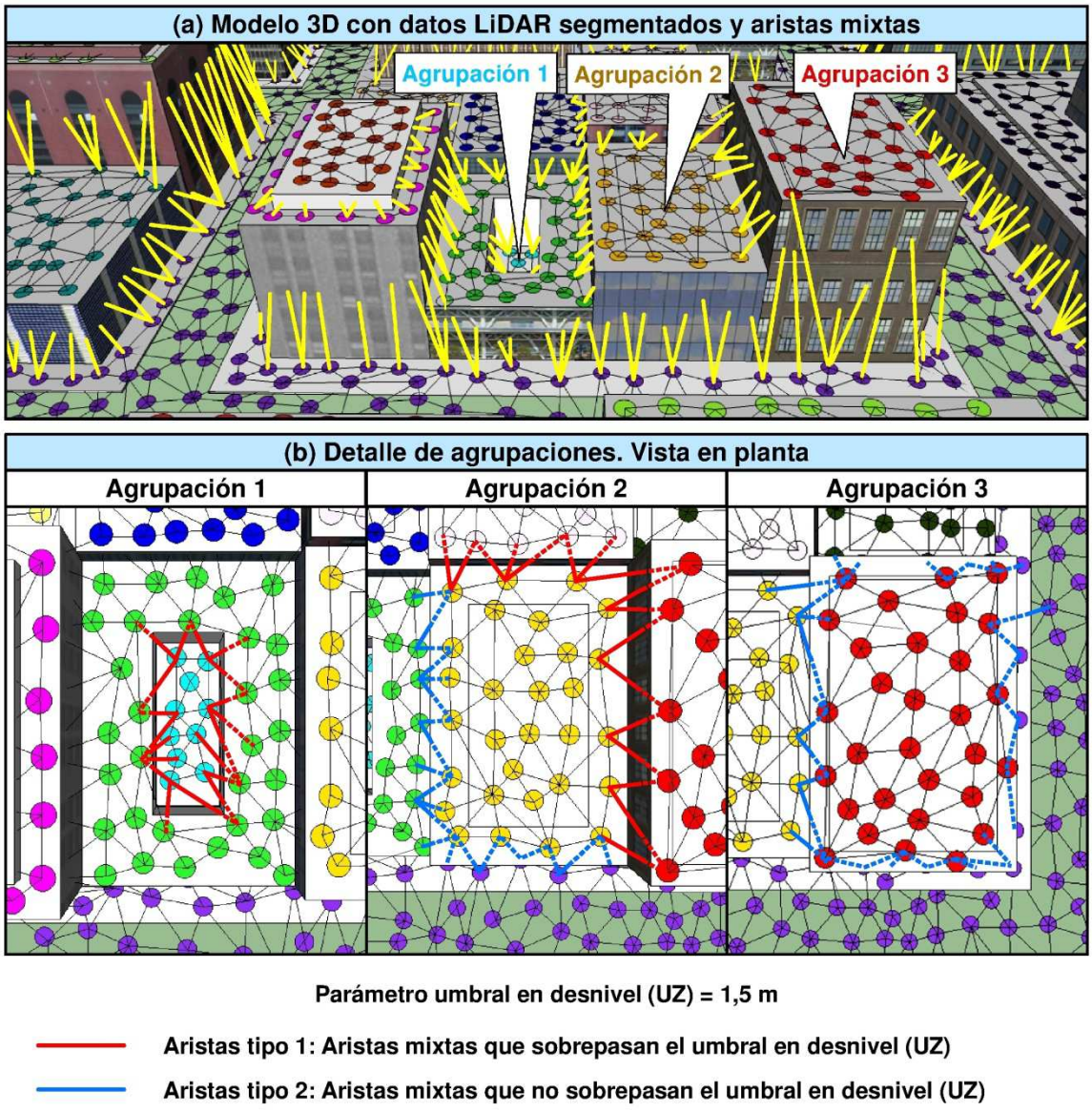


Figura 62. Clasificación de aristas mixtas en función de parémtro umbral en desnivel (UZ). Detección del terreno. Algoritmo B.

El objetivo del análisis de las artistas mixtas en cuanto al desnivel resulta de utilidad para generar una clasificación preliminar del terreno, ya que las agrupaciones pertenecientes a la clase terreno estarán rodeadas en su mayor parte por agrupaciones de mayor altitud pertenecientes a otras entidades (árboles, planos de techo, automóviles, farolas...) y por tanto, la mayor parte de sus aristas mixtas sobrepasarán el parámetro umbral en desnivel establecido (*UZ*) (aristas tipo 1). En este sentido, para realizar una clasificación provisional del terreno deberá establecerse un porcentaje límite en cuanto al número de aristas que sobrepasan el umbral en desnivel asociadas a cada región. Este umbral en porcentaje (*U%)* definirá el límite a partir del cual una agrupación debe de considerarse terreno u otro objeto

en el proceso de clasificación. Para el ejemplo propuesto se ha establecido un umbral en porcentaje del 80% que servirá como regla de decisión para la clasificación de regiones, donde una región podrá pertenecer a la clase terreno en el caso de que más del 80% de sus aristas mixtas sean del tipo 1. Para el cálculo de porcentajes de aristas mixtas del tipo 1 asociadas a cada región se utilizará la siguiente expresión:

$$\%\_Aristas\_Tipo\_1 = Nº\ Aristas\_Tipo\_1 \cdot 100 / (Nº\ Aristas\_Tipo\_1 + Nº\ Aristas\_Tipo\_2)$$

Donde:

- %_Aristas_Tipo_1: Porcentaje de aristas tipo 1 asociadas a la región objeto de análisis.
- Nº Aristas_Tipo_1: Número de aristas del tipo 1 asociadas a la región objeto de análisis.
- Nº Aristas_Tipo_2: Número de aristas del tipo 2 asociadas a la región objeto de análisis.

El proceso de clasificación mediante el umbral en porcentaje para el ejemplo propuesto puede apreciarse en la Figura 63:

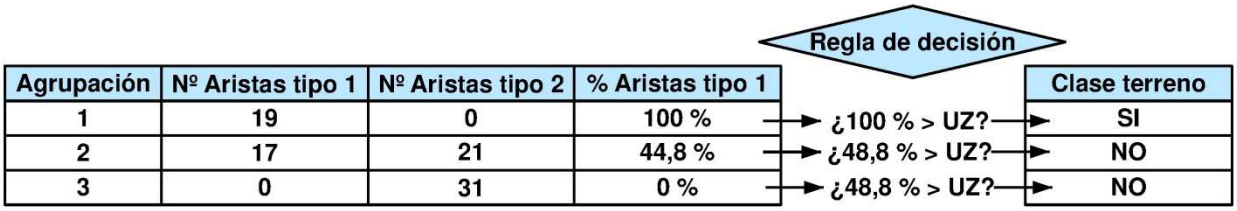


Figura 63. Clasificación de puntos como terreno a partir de umbral en porcentaje de aristas mixtas (tipo 1). Detección del terreno. Algoritmo B.

En el ejemplo propuesto, un parámetro umbral en desnivel (*UZ*) de 1,5 y un parámetro umbral en porcentaje (*U%*) del 80% permiten eliminar del posterior proceso de densificación dos planos de techo pertenecientes a edificios y únicamente se clasifica como terreno provisional una región perteneciente a un patio interior. Con esta operación se reducen principalmente errores de tipo 1 y 3. No obstante, la metodología propuesta debe de hacer frente a los tres tipos de errores detectados tras aplicar el algoritmo A. En este sentido, es necesario tener en cuenta una nueva variable para realizar el análisis. Esta nueva variable es el tamaño de cada una de las regiones, mientras que grandes regiones suelen pertenecer al entramado de calles urbanas y como consecuencia a la clase terreno, otras regiones de pequeño tamaño están asociadas a pulsos láser reflejados en las fachadas de edificios, vegetación de escasa altura, automóviles u otros objetos. En este sentido, los parámetros umbrales para regiones de escaso tamaño deben de ser más restrictivos con el fin de evitar errores de tipo 2. Así mismo, dado que a mayor dimensión de la región, mayor es la probabilidad de que pertenezca a la clase terreno los parámetros umbrales también deberán relajarse a medida que el tamaño de las agrupaciones aumenta.

Para calcular el área de las regiones obtenidas tras segmentar la escena se clasifican los triángulos existentes en la triangulación de Delaunay en función de la región a la que pertenecen sus vértices. Posteriormente, para cada región se realiza el sumatorio del área de sus triángulos asociados obteniendo así el área de cada región en un vector de igual dimensión que el número total de regiones denominado *V_Superficie_Regiones*. El proceso detallado del cálculo de áreas para las distintas regiones viene descrito en el apartado de algoritmos auxiliares (Algoritmo auxiliar. Apartado III.10.5).

El análisis individualizado en cuanto al tamaño y la posición relativa de cada una de las regiones se lleva a cabo en dos partes diferentes del proceso. Previamente al inicio del proceso de densificación la zona de estudio se divide en bloques o celdas de igual tamaño y

la región que contiene al punto de menor cota es seleccionada como terreno provisional. Posteriormente este terreno provisional es utilizado como base para iniciar el proceso de densificación que concluye con la completa clasificación de la clase terreno. Uno de los problemas asociados a esta parte del proceso es que el usuario debe de seleccionar el tamaño de las bloques, cuya superficie debe de ser siempre superior al mayor edificio existente en la zona de estudio para evitar errores en la generación del terreno provisional. Una selección errónea del tamaño de bloque tendrá como consecuencia la clasificación de planos de techo u otras entidades como terreno (error de tipo 1). Para evitar errores de este tipo, un análisis individualizado de cada región se lleva a cabo y las regiones son clasificadas como posible terreno o no mediante el análisis de su tamaño y la posición relativa con respecto a sus regiones colindantes. Las variables asociadas al análisis (tamaño de agrupación, umbral en desnivel y umbral en porcentajes) deben de ser altamente restrictivas en esta parte del proceso, ya que de lo contrario cualquier error de comisión se transmite directamente a la generación del terreno provisional y consecuentemente al posterior proceso de densificación.

Posteriormente, dentro del propio proceso de densificación se realiza otro análisis individualizado para cada región en cuanto a tamaño y posición relativa. En este caso los parámetros no serán tan restrictivos como en el primer análisis. No obstante, los umbrales deben de ser más restrictivos a medida que el tamaño de las regiones es menor, consiguiendo reducir así errores de tipo 2.

Tal y como se aprecia en el diagrama de flujo (Figura 64), gran parte del proceso es idéntico al realizado por el algoritmo A, si bien se incluyen dos análisis adicionales. El primer análisis (análisis 1) se ejecuta previamente a la detección del terreno provisional. Dicho análisis consiste en identificar aquellas regiones que tienen una mayor probabilidad de pertenecer al terreno y marcarlas en un vector de igual dimensión que el número total de regiones obtenidas tras segmentar la escena (*V_M_Clases_Terreno*), de tal forma que las regiones con menor altura local serán marcadas en el vector y serán las únicas que participarán en la determinación del terreno provisional inicial. Con ello se consigue reducir la influencia del tamaño de los bloques utilizados en el proceso de generación del terreno provisional y consecuentemente disminuyen los errores de comisión a la vez que aumenta el grado de automatización del proceso.

El segundo análisis (análisis 2) se ejecuta dentro del proceso iterativo de densificación. En este análisis se utilizan unos parámetros umbrales menos restrictivos que en el análisis 1. Básicamente, consiste en actualizar el vector *V_M_Clases_Terreno* marcando aquellas agrupaciones que pueden pertenecer al terreno. Para ello se realiza un análisis individualizado para cada una de las regiones en cuanto a tamaño y la posición relativa entre agrupaciones colindantes. En este sentido, las regiones de gran tamaño tendrán una mayor probabilidad de pertenecer al terreno. Además, las regiones pertenecientes al terreno estarán en contacto con otras cuyos puntos tendrán una mayor altitud (edificios, árboles, automóviles, etc.). Gracias a este doble análisis (tamaño y altitud relativa de agrupaciones colindantes) será posible identificar las agrupaciones que no pueden pertenecer a la clase terreno y eliminarlas del proceso de densificación.

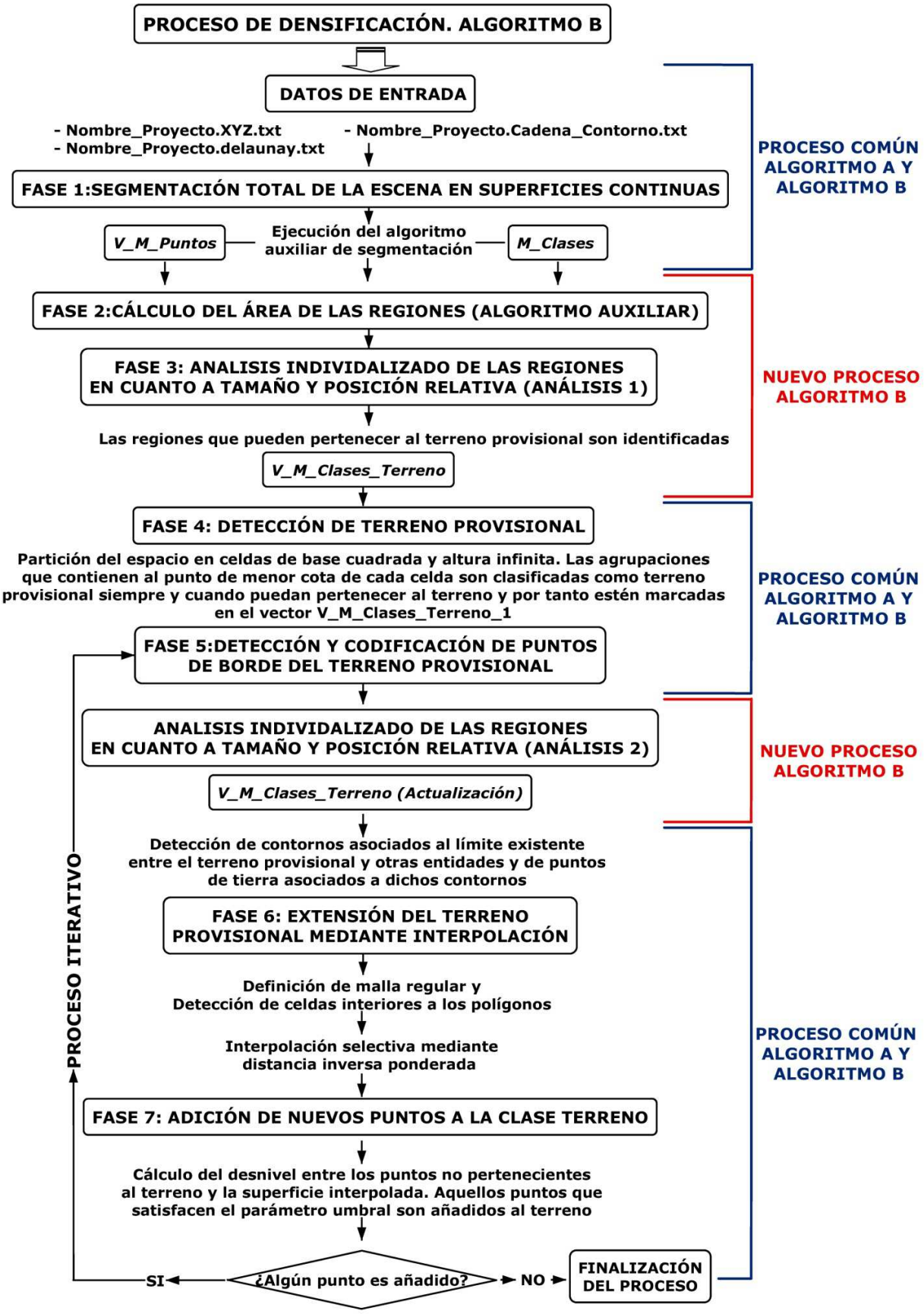


Figura 64. Diagrama de flujo del algoritmo B para detectar la clase terreno.

Los procesos que tienen en común el algoritmo A y B ya han sido descritos con anterioridad al definir el Algoritmo A (Apartado III.4), por lo que a continuación únicamente se describen los dos nuevos análisis que se incorporan al algoritmo y que consiguen reducir notablemente los errores de comisión asociados (errores de tipo 1, 2 y 3).

- **Análisis 1: análisis previo a la detección del terreno provisional**

La detección del terreno provisional inicial es esencial para obtener una clasificación precisa del terreno, ya que constituye el punto de partida del proceso de densificación. Para conseguir una menor dependencia del tamaño de bloque utilizado en el proceso de detección del terreno provisional y evitar errores del tipo 1, se incluye un análisis previo donde las regiones que tienen una mayor probabilidad de pertenecer a la clase terreno son marcadas en un vector de marcadores (*V_M_Clases_Terreno*). Para ello se aplica una regla de decisión que tiene en cuenta dos características de las regiones, su tamaño y la posición relativa existente entre regiones adyacentes en cuanto a altitud se refiere.

Dado que incluir una región que pertenece a un edificio, vegetación u otra entidad como terreno provisional constituye una fuente de error inicial considerable, es necesario establecer parámetros umbrales restrictivos en cuanto a los dos variables a tener en cuenta.

Para determinar la superficie de cada región se hace uso del algoritmo auxiliar que calcula el área de las distintas regiones obtenidas tras la segmentación (Algoritmo auxiliar. Apartado III.10.5), mientras que para determinar la posición relativa de las agrupaciones colindantes en cuanto a altitud se refiere, se recurre al cálculo del desnivel asociado a las aristas de la triangulación que unen puntos de la región objeto de análisis con sus colindantes (aristas mixtas). Para empezar, un parámetro umbral en desnivel es definido (*UZ*). Seguidamente se genera una matriz de dos columnas e igual número de filas que el número total de regiones obtenidas tras segmentar la escena (*M_Aristas_mixtas*). Posteriormente, para cada región se analizan sus aristas mixtas clasificándolas en dos tipos:

Aristas de tipo 1: número de Aristas que sobrepasan el parámetro umbral en desnivel (*UZ*). Son aristas asociadas a puntos de la región objeto de estudio que están conectados con regiones de mayor altitud, cuyo desnivel es superior al parámetro umbral (*UZ*).

Aristas de tipo 2: número de Aristas que no sobrepasan el parámetro umbral en desnivel (*UZ*). En estos casos se engloban aquellas aristas que conectan con puntos de menor altitud o las que conectan con puntos de similar altura.

Dado que los puntos contenidos en la clase terreno son colindantes a puntos pertenecientes a otras entidades que poseen una mayor altitud (edificios, árboles, automóviles...), se considera que los puntos que definen el límite de la clase terreno deben de estar conectados con puntos de otras entidades que tienen una mayor altitud y por tanto, las regiones de la clase terreno necesariamente deben tener un mayor porcentaje de aristas del tipo 1 que aquellas que pertenecen a otras entidades. Consecuentemente, con el fin de evitar errores por exceso en la definición del terreno provisional y minimizar la influencia del tamaño de los bloques seleccionado por el usuario, para cada región se analiza su tamaño y la posición relativa, para finalmente aplicar las reglas de decisión que aparecen estructuradas en el siguiente árbol de decisión (Figura 65):

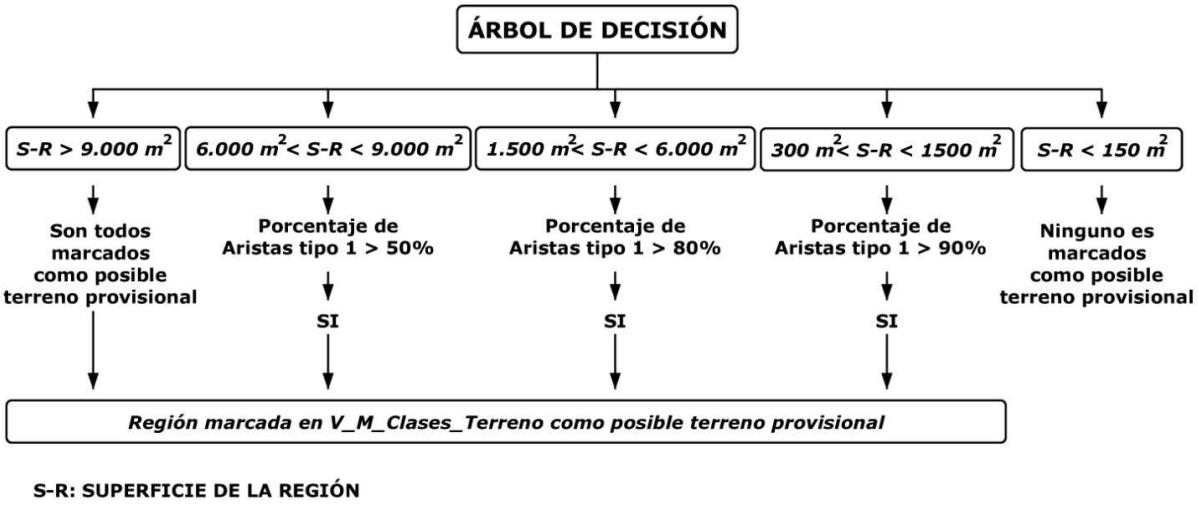


Figura 65. Detección del terreno. Árbol de decisión análisis 1. Algoritmo B.

A partir del árbol de decisión el algoritmo marca en el vector *V_M_Clases_Terreno* aquellas regiones que tienen mayor probabilidad de pertenecer a la clase terreno. Para ello, la aplicación tiene en cuenta que cuanto mayor es el tamaño de la región mayor es la probabilidad de que pertenezca al terreno. En este sentido, las regiones cuya superficie son mayores a 9.000 m^2 son seleccionadas como posible terreno sin análisis adicionales. Para el resto de las agrupaciones de menor tamaño, se calcula el porcentaje de aristas mixtas que conectan el borde de las regiones con regiones de mayor altitud, de forma que cuanto menor es el tamaño de la región mayor debe de ser el porcentaje de aristas del tipo 1 para ser considerado posible terreno provisional.

El hecho de que una región sea marcada como posible terreno provisional en el vector *V_M_Clases_Terreno* no implica que finalmente sea clasificado como tal; el árbol de decisión únicamente restringe las posibles regiones consiguiendo eliminar así errores de comisión en la posterior identificación del terreno provisional. Para generar el terreno provisional inicial que sirve de base al proceso de densificación, la zona de estudio se dividirá en bloques iguales y la región con el punto de menor altura que además esté marcada en el vector *V_M_Clases_Terreno* será añadida al terreno provisional inicial.

En la Figura 66 se muestran las regiones marcadas en el vector *V_M_Clases_Terreno* tras realizar el análisis 1 así como el terreno provisional inicial finalmente detectado tras dividir la zona de estudio en bloques de igual tamaño.

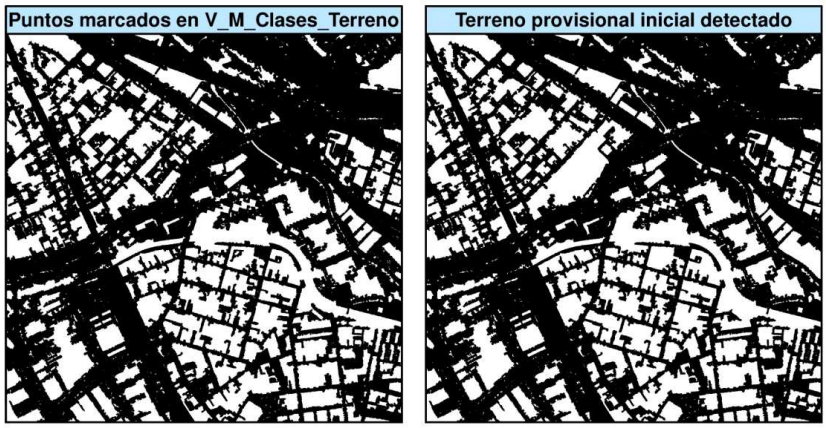


Figura 66. Comparación entre puntos del vector *V_M_Clases terreno* y el finalmente detectado como terreno provisional inicial. Detección del terreno. Algoritmo B.

Para detectar el terreno provisional la aplicación utiliza la misma metodología que el algoritmo A. En primer lugar, la zona de estudio se devide en bloques de igual tamaño y posteriormente la región de menor altitud de cada una de ellos se selecciona para generar la superficie de referencia. No obstante, a diferencia del algoritmo A, la región de menor altura de cada celda debe de haber sido marcada previamente en el vector *V_M_Clases_Terreno*, en caso contrario la región queda descartada y no se añade al terreno provisional.

- **Análisis 2: análisis en el proceso de densificación**

Una vez definido el terreno provisional inicial, el proceso de densificación comienza y la superficie de referencia se actualiza en cada iteración añadiendo los nuevos puntos de terreno detectados. Con el fin de eliminar errores del tipo 2 y 3, un nuevo análisis análogo al análisis 1 se lleva a cabo. En este caso los parámetros umbrales son menos restrictivos para evitar errores de omisión. Antes de ejecutar el análisis, el vector de marcadores *V_M_Clases_Terreno* se reinicia. Seguidamente se aplica el siguiente árbol de decisión marcando las clases que tienen una alta probabilidad de pertenecer al terreno.

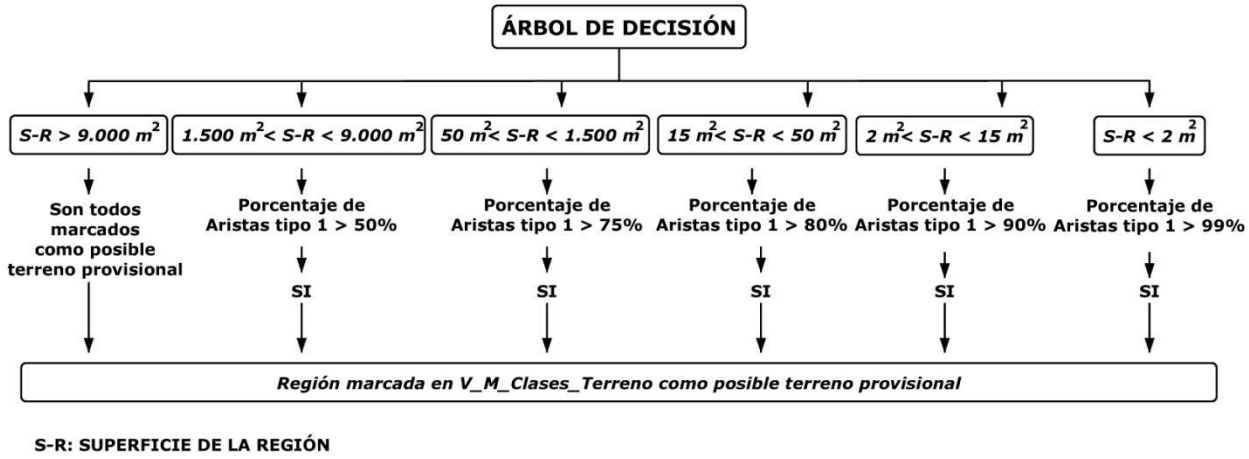


Figura 67. Detección del terreno. Árbol de decisión análisis 2. Algoritmo B.

El objetivo del análisis 1 es minimizar errores en la selección del terreno provisional inicial, estos errores (errores de tipo 1) se dan debido principalmente a una inadecuada selección del tamaño de los bloques por parte del operador, como consecuencia el análisis es más restrictivo con respecto a las regiones asociadas a medias y pequeñas superficies. No obstante, el análisis 2 está diseñado para reducir errores del tipo 2 y 3, por tanto, el análisis establece nuevos parámetros para las regiones con escasa superficie, diferenciando un mayor número de casos. Los parámetros umbrales en cuanto a las superficies umbrales y porcentajes de aristas del tipo 1 se han establecido tras realizar numerosas pruebas experimentales y analizar la precisión mediante una revisión visual de los resultados.

III. 6. Detección de puentes. Algoritmo A: localización de bordes y análisis direccional de la continuidad estructural

III.6.1. Motivación para el desarrollo del algoritmo y relación con los antecedentes

Una gran parte de los métodos existentes para detectar el terreno tales como el método de densificación de TIN, filtros morfológicos o filtros basados en interpolación de superficie, son capaces de aislar el terreno y filtrar los puntos pertenecientes a puentes sin análisis adicionales. No obstante, aquellos algoritmos que segmentan la escena y utilizan como objeto de análisis las regiones obtenidas, clasifican los puentes dentro de la clase terreno. En este sentido, el algoritmo diseñado para detectar el terreno en el presente trabajo combina la segmentación de la escena en superficies continuas con un proceso de densificación y por consiguiente, los puentes son incluidos automáticamente dentro en la clase terreno. Esta inclusión se debe a que la superficie de los puentes está conectada con el terreno sin presentar discontinuidades en el inicio y fin de los mismos. Existen algunos estudios que ya combinan el método de densificación con un proceso de segmentación (Pérez et al. 2012, Lin y Zhang 2014,...). Lin y Zhang (2014) identifican como una fuente de error importante, la inclusión de los puentes en la clase terreno. Por otro lado, en el presente estudio se han utilizado datos test que presentan puentes y se ha hecho especial hincapié en cuantificar la influencia de este tipo de error.

La inclusión de puentes dentro de la clase terreno es una fuente de error que tiene especial relevancia cuando el objetivo es utilizar el MDT derivado para analizar riesgos de inundaciones. Ello se debe a que los puentes generan en el MDT presas ficticias y por tanto el análisis es erróneo. En estos casos, conviene detectar los puentes en el conjunto de datos LiDAR y filtrarlos.

En la zona test 2 (Collado Villalba) existen un total de seis puentes, estos puentes son detectados erróneamente como terreno tras ejecutar el algoritmo de detección del terreno que combina la segmentación en superficies continuas con un proceso de densificación. Consecuentemente 6.057 puntos pertenecientes a dichos puentes son clasificados por error como terreno, lo que se transmite en un incremento del error de comisión en la clase terreno en torno al 0,5%. Con el fin de eliminar esta fuente de error se opta por desarrollar un algoritmo que sea capaz de detectar los puentes incluidos en la clase terreno y clasificarlos como una entidad propia.

En cuanto a los estudios existentes cuyo objetivo es diferenciar los puentes del terreno en una nube tridimensional de datos LiDAR, los enfoque que mejores resultados obtienen son aquellos basados en identificar características geométricas y estructurales propias de los puentes. Sithole y Vosselman (2006) presentan un algoritmo que detecta los puentes mediante la utilización de perfiles transversales y el posterior análisis de la información topológica asociada. De forma análoga Zheng et al. (2013) detectan los bordes de los puentes sobre un MDT generado a partir de datos LiDAR para posteriormente analizar distintas características como la dirección principal del puente o la continuidad estructural entre los puntos del vecindario. Análogamente, Evans (2008) utiliza un MDT obtenido a partir de datos LiDAR para detectar los bordes de los puentes y seguidamente analizar características geométricas como la simetría existente en los mismos.

A la hora de diseñar el nuevo algoritmo para detectar puentes, dos nuevos aspectos se introducen con respecto a los trabajos existentes anteriormente mencionados. Por un lado, la totalidad de algoritmos implementados en el presente estudio actúan directamente sobre los datos LiDAR brutos sin interpolar a malla regular (GRID). Por tanto, siguiendo con la misma línea de investigación el algoritmo a desarrollar deberá actuar sobre los datos LiDAR

dispuestos de forma aleatoria sin utilizar en ningún caso MDTs derivados. Además, los algoritmos existentes analizan las características geométricas de los puentes (paralelismo de los bordes, simetría, continuidad estructural, etc.) mediante procesos encadenados, y consecuentemente el tiempo de cómputo es considerable. Con el fin dar solución a ambas cuestiones, en el presente estudio se expone el nuevo concepto de clasificador direccional por sectores.

III.6.2. Descripción del algoritmo programado

El algoritmo desarrollado se basa en detectar los bordes de los puentes y posteriormente clasificar los puntos LiDAR comprendidos entre dichos bordes que pertenecen a los puentes. Para ello, la aplicación ejecuta dos procesos concatenados. Una primera fase consiste en detectar los puntos de borde de los puentes en el conjunto de los datos LiDAR y posteriormente, un segundo proceso analiza las características geométricas de los puntos comprendidos entre ellos con el fin de determinar si pertenecen a un puente.

Para detectar los bordes de los puentes la aplicación implementada detecta cambios bruscos de pendiente dentro de la clase terreno. Debe de tenerse en cuenta que el algoritmo se ejecuta sobre la clase terreno donde los edificios, vegetación y otros objetos ya han sido filtrados. Consecuentemente, para identificar los bordes de los puentes bastará con localizar pendientes verticales en el terreno. En este sentido, el algoritmo implementado analiza el vecindario cilíndrico vertical de cada uno de los puntos pertenecientes al terreno detectando cambios bruscos de altitud con respecto a los puntos del terreno incluidos en su vecindario.

Una vez localizados los bordes, la siguiente fase consiste en identificar los puntos LiDAR que están situados entre dos puntos de borde opuestos y que por tanto son susceptibles de pertenecer a un puente. Seguidamente se comprueba que la altitud del supuesto punto de puente es similar a la de los bordes y finalmente se determina si existe continuidad estructural en la dirección principal del puente. En caso afirmativo, el punto es clasificado dentro de la clase puentes.

A continuación se describen detalladamente los dos fases principales que componen el algoritmo.

- **Detección de bordes de puentes**

Para detectar los puntos LiDAR que constituyen el límite o borde de los puentes, el algoritmo recorre cada uno de los puntos pertenecientes a la clase terreno y determina su vecindario cilíndrico vertical. Seguidamente la aplicación comprueba si existe un alto número de puntos del vecindario que además de pertenecer a la clase terreno tienen un desnivel superior a un cierto parámetro umbral. En caso afirmativo significará que existe una pendiente vertical del terreno en torno al punto y por tanto, será clasificado como borde.

Tres parámetros umbrales son utilizados en este primer proceso. Por un lado será necesario establecer el radio del vecindario del cilíndrico vertical donde se buscarán pendientes verticales. En este sentido, dado que los bordes de los puentes son colindantes a cambios bruscos de altitud, un radio de 2 m de longitud es suficiente. El siguiente parámetro consiste en el desnivel umbral (*UIZ_Borde*) necesario para determinar que existe un desnivel importante colindante al posible punto de borde, tal y como se analizará en apartados posteriores (Apartado IV. 3). La correcta elección de este parámetro tendrá una importante influencia en la precisión de los resultados obtenidos. Por último, se aplicará un umbral en porcentaje (*U%_IZB*) que determinará el porcentaje mínimo de puntos del

vecindario que sobrepasan el umbral en desnivel para que el punto sea clasificado como borde de puente.

El proceso para detectar el borde de los puentes puede apreciarse en la Figura 68. Supongamos que tres puntos del terreno son seleccionados al azar con el fin de comprobar si son bordes de una línea de rotura que define una pendiente vertical (Figura 68a). Para los tres puntos (puntos base) se detectan los puntos del terreno que pertenecen al vecindario cilíndrico (Figura 68b), seguidamente se contabiliza el número de puntos de borde cuyo desnivel con respecto al punto base es mayor que el parámetro umbral *UIZ_Borde* establecido (Figura 68c). Finalmente si el número de puntos del vecindario que incumplen el parámetro umbral *UIZ_Borde* es mayor que un cierto parámetro umbral en porcentaje (*U%_IZB*), el punto es clasificado como borde (Figura 68d y Figura 68e). Aplicando este proceso a la totalidad de puntos pertenecientes a la clase terreno se consigue detectar los puntos de borde adyacentes a pendientes verticales del terreno (Figura 68f).

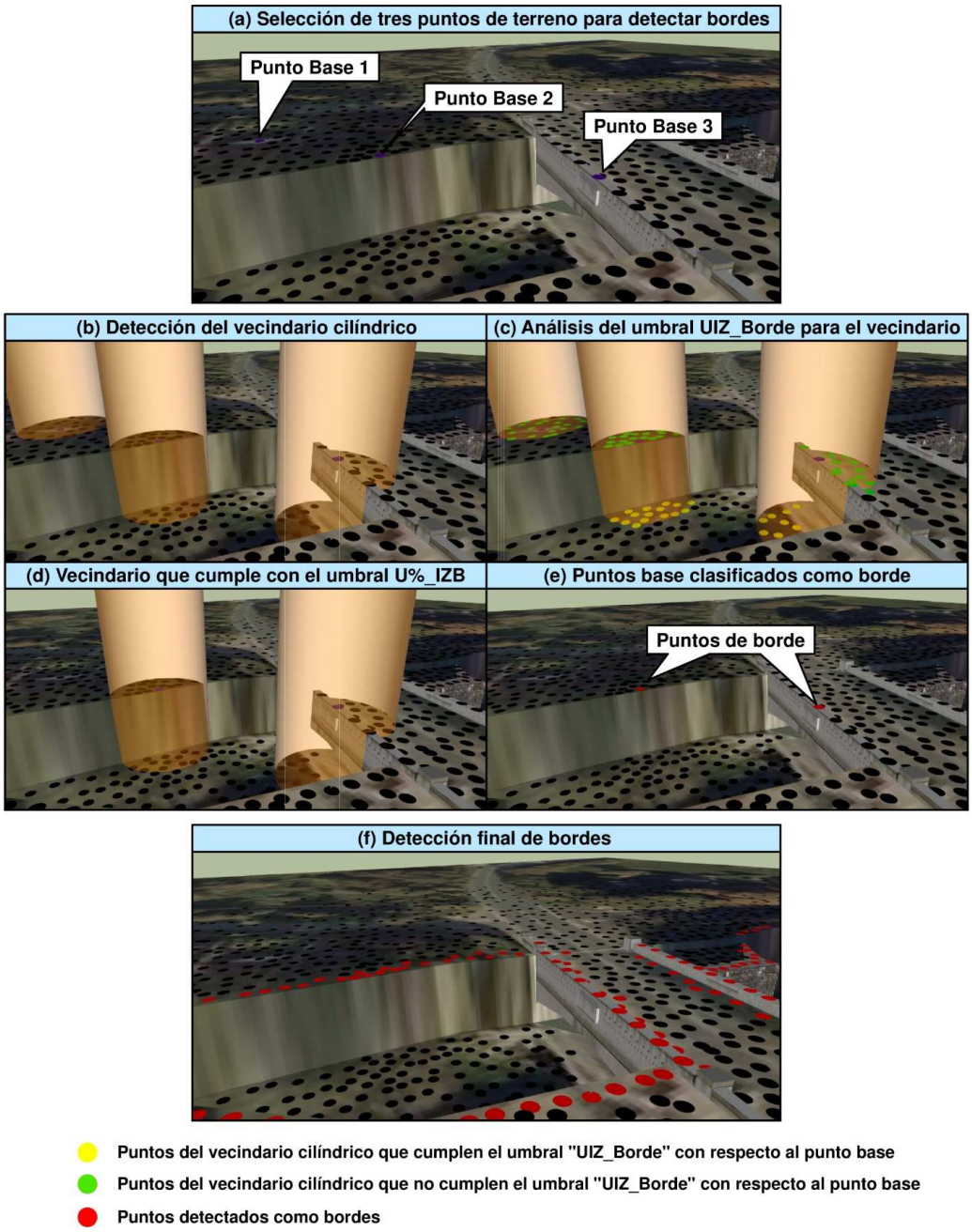


Figura 68. Detección de bordes asociados a pendientes verticales del terreno.

Tras realizar multitud de pruebas experimentales y posteriormente analizar visualmente los resultados, se determina que el parámetro umbral en porcentajes (*U%_IZB*) que mejores resultados obtiene es del 20%. No obstante, cabe resaltar que los bordes detectados pertenecen a todas las pendientes verticales definidas dentro de la clase terreno. Por tanto, se clasifican como bordes tanto los límites de los puentes como los de los canales, barrancos u otros tipos de terreno adyacente a pendientes verticales (Figura 68f). Consecuentemente un nuevo análisis deberá realizarse para diferenciar los puntos de borde que pertenecen a los puentes de los restantes. Para ello, en la siguiente fase, el algoritmo analiza distintas características de la estructura de los puentes tales como la continuidad estructural entre bordes opuestos, así como en la dirección principal del puente.

- **Detección de puntos pertenecientes a puentes**

Para detectar aquellos puntos que pertenecen a los puentes dentro del conjunto de datos LiDAR, el algoritmo tiene en cuenta distintas características de los puentes tales como su anchura, paralelismo de los bordes y continuidad estructural en la dirección principal del mismo.

El algoritmo toma como datos de entrada la clase terreno donde los bordes adyacentes a pendientes verticales han sido previamente detectados. Seguidamente, dado que los puntos pertenecientes a los puentes están situados entre puntos de bordes que pertenecen a lados opuestos del puente, el algoritmo recorre la totalidad de puntos del terreno y detecta el vecindario cilíndrico de cada uno de ellos. El objetivo es identificar puntos de borde pertenecientes a lados opuestos del puente, por lo que el radio del vecindario deberá ser mayor a la anchura de los puentes existentes en la zona de estudio.

Con el fin de detectar bordes que pertenecen a lados opuestos del puente, el nuevo concepto de clasificador direccional por sectores es utilizado. Este clasificador consiste en dividir el vecindario cilíndrico asociado a cada punto de tierra en sectores de igual amplitud y los puntos del terreno contenidos en el vecindario son clasificados en función del sector al que pertenecen. Seguidamente, la aplicación recorre los diferentes sectores comprobando si existe algún punto de borde. En caso afirmativo, se comprueba si su sector opuesto (sector que difiere en 180º) también contiene al menos un punto de borde. Si es así, el punto de tierra objeto de estudio estará situado entre dos puntos de bordes diametralmente opuestos y por tanto, podrá pertenecer a un puente.

En cualquier caso, no todos los puntos del terreno situados entre dos puntos de borde pertenecen a un puente, por ejemplo, los puntos del interior de un canal estarán también situados entre los puntos de borde que define la parte superior del talud asociada a la propia estructura del canal. Por tanto, es necesario diferenciar los posibles puntos que pertenecen a un puente de aquellos que pertenecen al interior de canales, barrancos u otras zonas del terreno situadas entre pendientes verticales. A tal efecto, se define un nuevo parámetro (*UMBRAL_IZ_PT_BASE*) que establecerá el desnivel máximo que debe de existir entre un posible punto de puente y sus bordes adyacentes. Cabe destacar, que los puntos pertenecientes a los puentes tienen una altitud similar a la de los bordes de los mismos, debido a la propia estructura del puente. Esta característica no se cumple con respecto a puntos pertenecientes al interior de barrancos o canales, ya que los bordes asociados a estos últimos poseen una altura mucho mayor. Consecuentemente, para diferenciar ambos casos, bastará con establecer un parámetro umbral en incremento de altitud con respecto a los puntos de borde adyacentes (*UMBRAL_IZ_PT_BASE*), de tal forma que si el desnivel entre el punto del terreno objeto de estudio y los puntos de borde asociados es mayor a dicho parámetro, el punto deberá ser descartado y pertenecerá a la clase terreno.

La Figura 69 refleja todo el proceso de identificación de un posible punto de puente. El algoritmo recorre la totalidad de puntos clasificados como terreno. Supongamos que se están analizando dos puntos del terreno para comprobar su posible inclusión en la clase

puente. En el ejemplo propuesto se testea un punto perteneciente a un puente y otro que pertenece al interior de un canal (Figura 69a). El primer paso consiste en detectar el vecindario cilíndrico vertical asociado a cada uno de los puntos bases y dividirlo en sectores de igual amplitud (Figura 69b). La amplitud de los sectores es un umbral a seleccionar por el usuario y sirve para establecer tanto la dirección de los puntos de bordes diametralmente opuestos así como la dirección principal del puente. Tras realizar distintos análisis se establece que un parámetro umbral menor o igual a 20^g de amplitud ofrece óptimos resultados. Seguidamente, el algoritmo recorre los diferentes sectores buscando aquellos que contienen puntos de borde, en caso de que existan sectores diametralmente opuestos que contienen puntos de borde el punto podrá pertenecer a un puente. Sin embargo, esta no es una condición suficiente, ya que en el ejemplo propuesto ambos puntos base tienen puntos de borde situados en sectores que difieren 180º (Figura 69c). Con el fin de descartar puntos del terreno que no pertenecen a la clase puente pero están situados entre puntos de bordes opuestos, el algoritmo calcula el desnivel entre el punto base y sus bordes asociados. Si el desnivel es mayor a un cierto parámetro umbral (*UMBRAL_IZ_PT_BASE*), el punto queda descartado y directamente se clasifica como terreno. En el caso que nos ocupa, el punto base 2 pertenece al interior de un canal e incumple el parámetro umbral establecido (Figura 69d). Consecuentemente, el punto base 2 es descartado y únicamente es seleccionado como posible punto de puente el punto base 1 (Figura 69e).

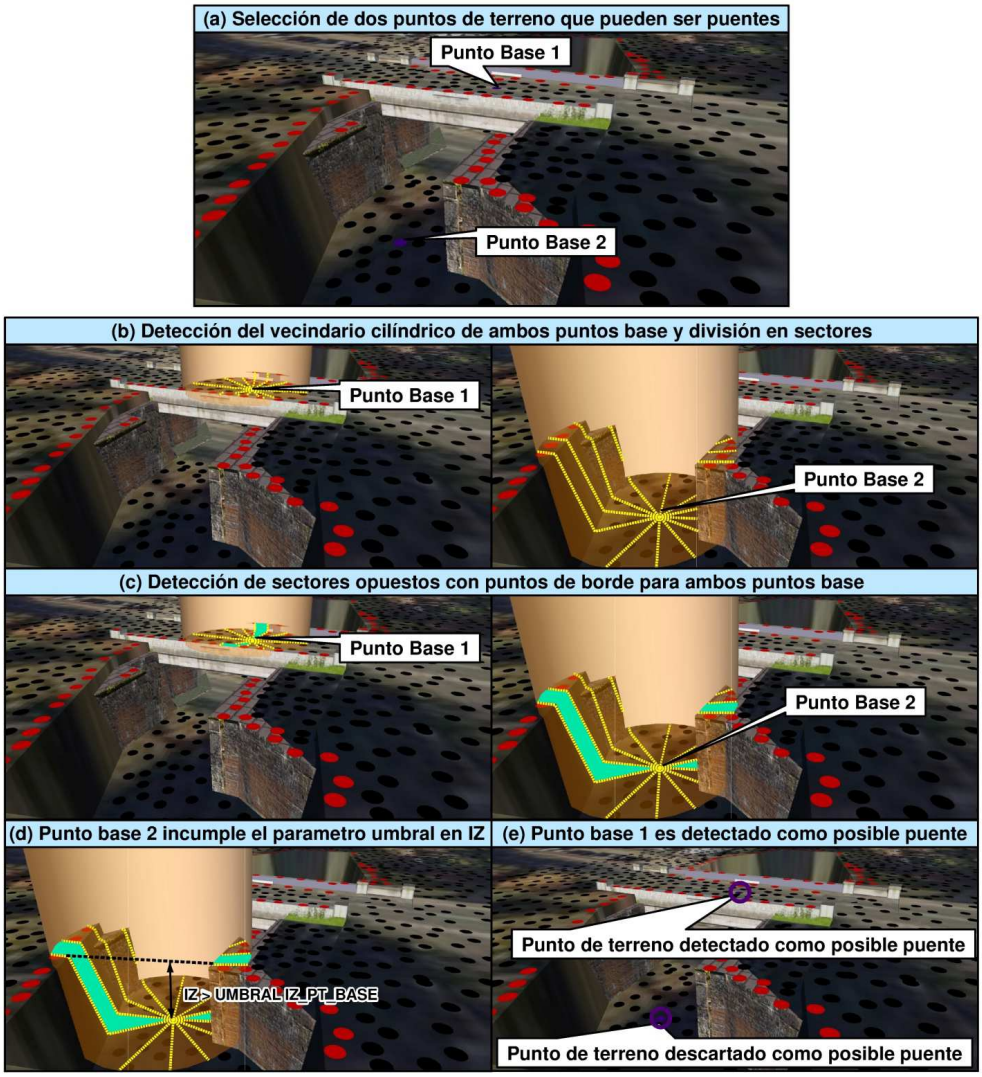


Figura 69. Análisis direccional mediante sectores para detectar puntos de bordes opuestos. Detección de puentes (Algoritmo A).

A pesar de que el análisis anteriormente descrito identifica la mayor parte de puntos LiDAR que pertenecen a los puentes, puede darse el caso de puntos del terreno que tienen puntos de borde diametralmente opuestos y que además cumplen con el parámetro umbral en desnivel. Estos casos puntuales se dan en aquellos puntos de tierra que son colindantes a la parte superior de un desnivel vertical en la clase terreno, tales como las bordes superiores del talud asociado a un canal o un barranco. Para descartar estos falsos positivos es necesario analizar otra característica básica de los puentes, su dirección principal.

Los puentes son un elemento artificial que lleva asociada una continuidad estructural tanto entre bordes opuestos como en la dirección principal del mismo. Esta característica no se cumple en otras entidades como los barrancos o los canales. Por tanto, con el fin de eliminar esta fuente de error un nuevo análisis se realiza en la dirección principal del puente.

Básicamente, el algoritmo detecta si existen puntos de bordes en sectores opuestos que difieren en 180º. En caso afirmativo, se comprueba que el punto base tiene un desnivel inferior a un cierto umbral (*UMBRAL_IZ_PT_BASE*) con respecto a los bordes detectados, consiguiendo así descartar los puntos del terreno que pertenecen al interior de un canal o barranco (Figura 69). Llegado a este punto, si el punto pertenece a un puente debe de existir continuidad estructural a lo largo de la dirección principal del puente. Esta dirección es paralela a los bordes del puente y por tanto, basta con analizar los sectores que son perpendiculares a aquellos que contienen puntos de borde diametralmente opuestos. Para tal fin, el algoritmo analiza los sectores que difieren en 90º con respecto a los sectores que contienen puntos de borde previamente detectados. En caso de que los puntos que pertenecen a estos sectores cumplan el parámetro umbral en desnivel con respecto al punto base significa que existe continuidad estructural en la dirección principal y por tanto el punto es clasificado como puente, en caso contrario el punto base será descartado.

La totalidad del proceso puede apreciarse en la Figura 70. En el ejemplo propuesto dos puntos base pertenecientes al terreno son seleccionados como objeto de análisis. Por un lado un punto perteneciente a un puente (punto base 1), por otro un punto del terreno que pertenece a la parte superior de un talud asociados a un barranco (punto base 2) (Figura 70a). En primer lugar, el vecindario cilíndrico de cada punto base es detectado y dividido en sectores de igual amplitud angular (Figura 70b). Posteriormente son detectados aquellos sectores diametralmente opuestos que contienen puntos de bordes (Figura 70c). En el ejemplo propuesto ambos puntos base cumplen con el parámetro en desnivel (*UMBRAL_IZ_PT_BASE*) con respecto a los puntos de borde existentes en los sectores diametralmente opuestos detectados. Por tanto, ambos puntos base son válidos y pueden pertenecer a un puente. Con el fin de descartar falsos positivos, el algoritmo comprueba los sectores que difieren en 90º con respecto a los sectores que contienen puntos de borde (Figura 70d). En el caso del *punto base 1* que pertenece a un puente, dichos sectores están asociados a la dirección principal del puente y por tanto, en ellos existe una continuidad estructural que se transmite en un escaso desnivel entre el punto base y los puntos asociados a dichos sectores. No obstante, dicha continuidad estructural no existe en el *punto base 2* dentro de los sectores que difieren en 90º con respecto a aquellos que contienen bordes, quedando así descartado el *punto base 2* como puente (Figura 70e). Finalmente el algoritmo clasifica la totalidad de los puntos existentes en los sectores asociados al *punto base 1* y que contienen bordes diametralmente opuestos (Figura 70f). Ejecutando el mismo procedimiento para la totalidad de puntos situados entre bordes, la totalidad de los puntos pertenecientes al puente son clasificados (Figura 70g).

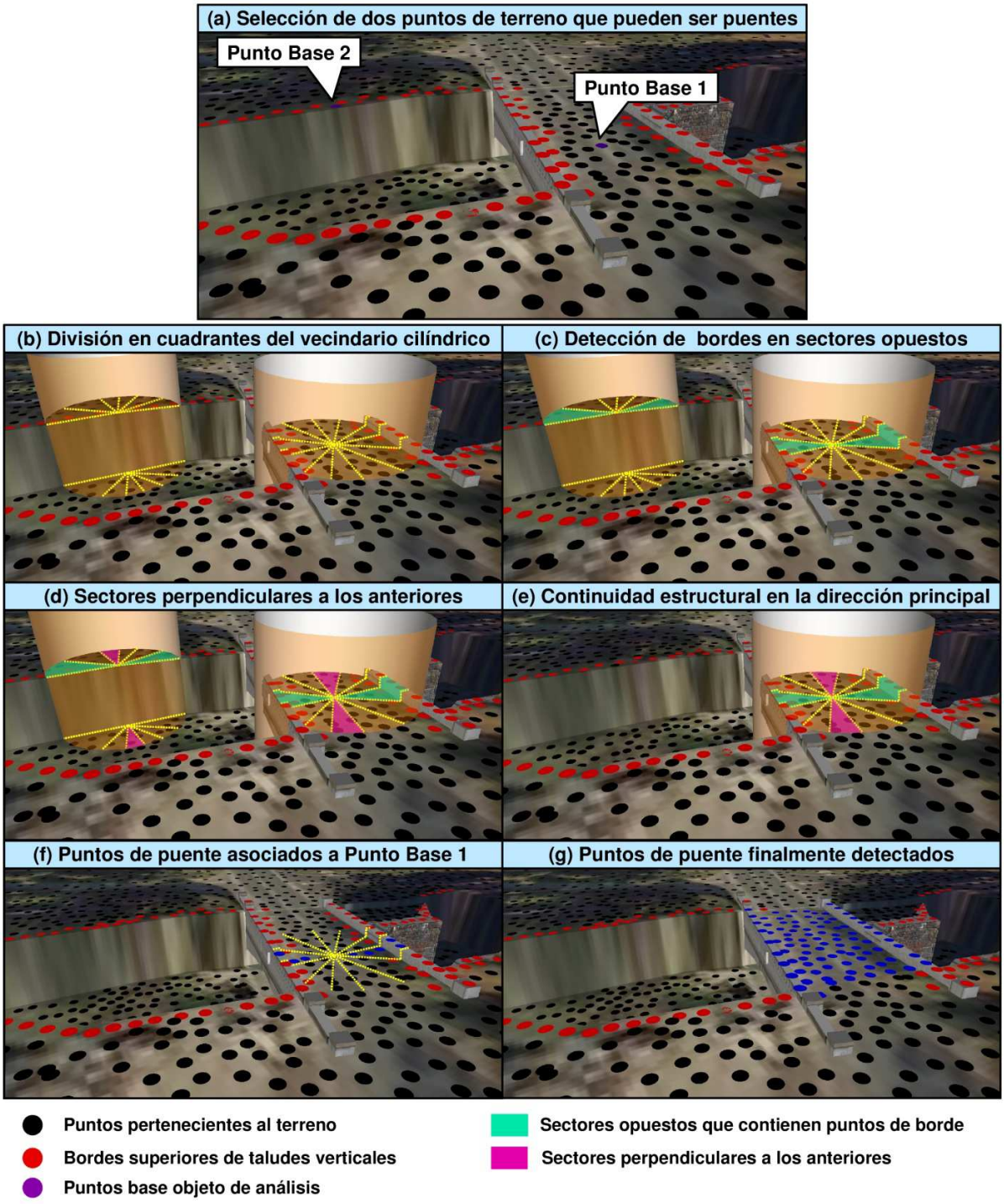


Figura 70. Análisis direccional mediante sectores para detectar puentes.

El diagrama de flujo que sigue la totalidad del proceso puede observarse en la siguiente imagen (Figura 71):

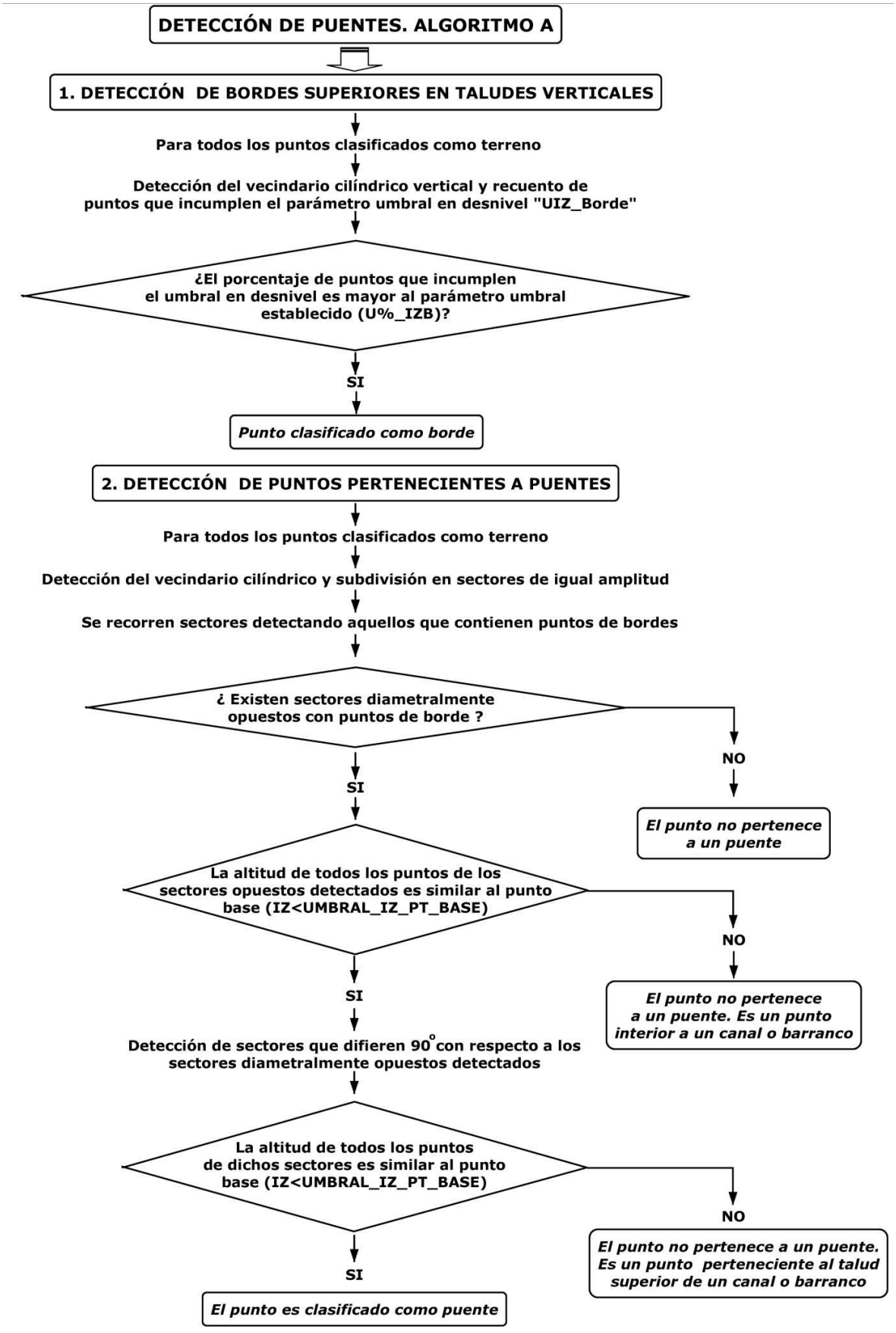


Figura 71. Diagrama de flujo. Detección de puentes. Algoritmo A.

Una de las ventajas del método planteado es que la detección de puntos situados entre los bordes de los puentes y el análisis de la continuidad estructural con respecto a la dirección principal del puente se integran en un único proceso. Para ello, tras generar el vecindario cilíndrico vertical, el algoritmo calcula automáticamente el número de sectores en el que se dividirá el vecindario en función de la amplitud angular introducida por el usuario. De esta forma, la aplicación define en tiempo de ejecución una matriz de igual número de filas que el número de sectores y los puntos del terreno, y de borde son clasificados dentro de la matriz en su sector correspondiente. Por tanto, para realizar el análisis basta con recorrer dicha matriz y aplicar las reglas de decisión correspondientes.

III. 7. Detección de puentes. Algoritmo B: localización de bordes y análisis direccional de la continuidad estructural. Multiproceso con parámetros umbrales adaptativos

III.7.1. Motivación para el desarrollo del algoritmo y relación con los antecedentes

El algoritmo propuesto en el apartado anterior (algoritmo A) consigue detectar los puentes con buenos resultados. No obstante, tal y como se profundizará en posteriores aparatados (Apartado IV. 3) para obtener óptimos resultados es necesario una precisa detección previa de los puntos pertenecientes a los bordes de los puentes. En consecuencia, existe una fuerte dependencia entre la precisión de los resultados obtenidos y el parámetro umbral en desnivel (*UIZ_Borde*) utilizado para detectar los bordes. En este sentido, la elección de un pequeño desnivel como parámetro umbral se transmitirá en una detección de bordes pertenecientes a otras estructuras (entradas de metros o aparcamientos, barandillas, quitamiedos, medianas de autovías o autopistas...) como bordes de puentes, dando lugar a errores por exceso en la detección de los mismos. Por otro lado, si el desnivel elegido como parámetro umbral es demasiado alto, se eliminarán los errores de comisión pero se generarán nuevos errores de omisión en los extremos de los puentes. Ello es debido a que en el inicio y fin de los puentes su estructura conecta con el terreno natural y por tanto el desnivel entre el puente y el terreno disminuye hasta la franja de conexión donde es nulo. En consecuencia, la utilización de un desnivel alto como parámetro umbral tiene como resultados errores por defecto en los extremos del puente.

El error de omisión asociado a la elección de un desnivel alto como parámetro umbral (*UIZ_Borde*) en la detección de bordes puede apreciarse en la Figura 72. El ejemplo propuesto se corresponde con un caso real de los datos test 2 (Collado Villalba), concretamente la zona norte del área de estudio donde existe una autovía con un puente que la cruza. En la primera imagen se muestra el modelo 3D generado con los datos LiDAR sobre el que se ha proyectado la ortofoto. Además se aprecian los puntos LiDAR que constituyen el puente (Figura 72a). El parámetro umbral en desnivel utilizado es de 4 m y la mayoría de puntos asociados al borde del puente han sido detectados correctamente. No obstante, cuando nos acercamos al inicio del puente el desnivel entre el puente y el terreno disminuye hasta que el puente conecta con el propio terreno, momento en el cual el desnivel es cero. Por este motivo, en los tramos cercanos al inicio o al fin del puente el desnivel es menor al parámetro umbral y existen puntos de borde que no son detectados. En el ejemplo propuesto se aprecia claramente como la zona donde el desnivel entre el puente y el terreno coincide con el parámetro umbral (*UIZ_Borde*) marca el límite a partir del cual los bordes son detectados (Figura 72b). Finalmente este error por defecto en la detección de bordes se transmite directamente a un error de omisión en la clasificación de la entidad (Figura 72c).

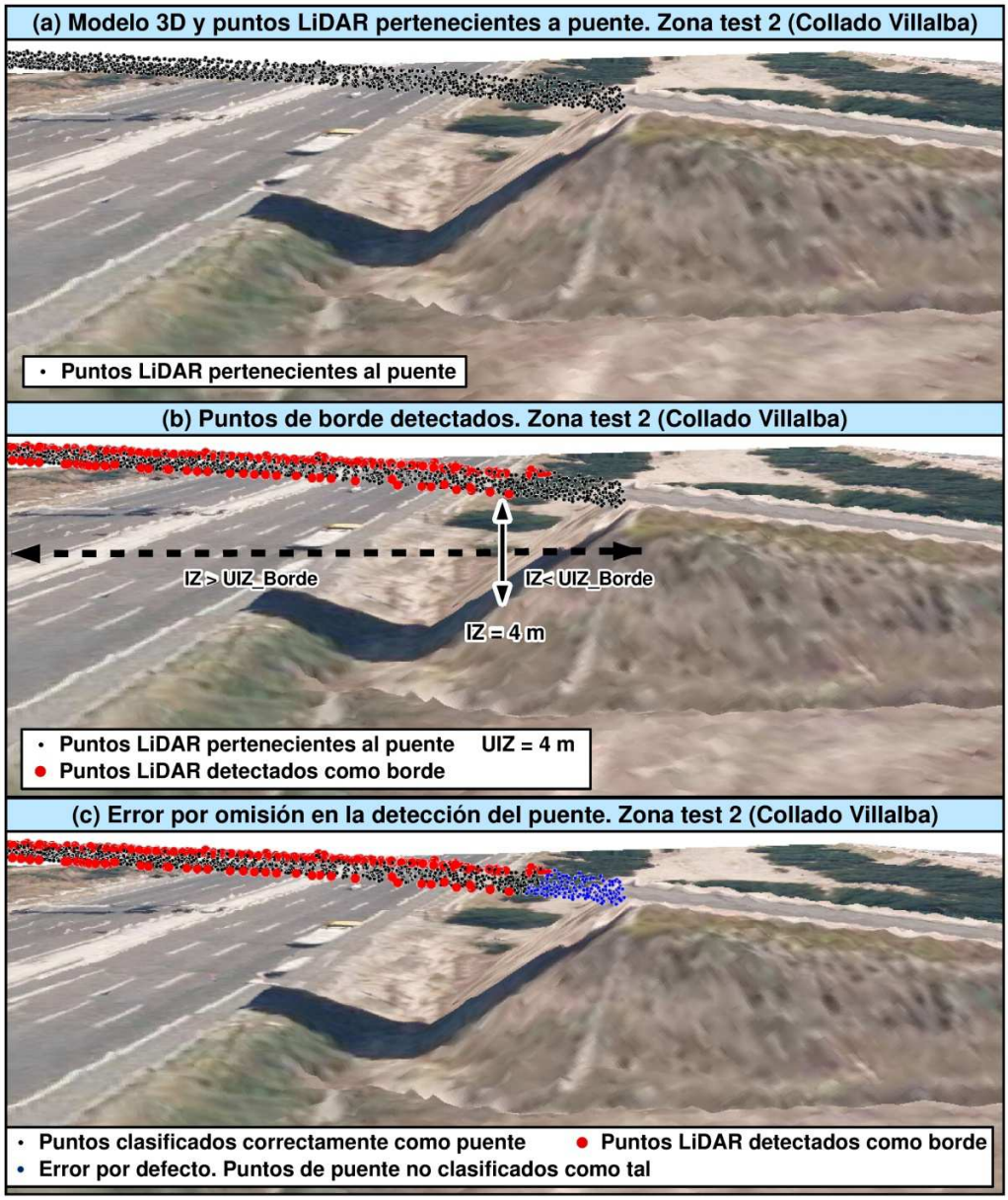


Figura 72. Errores por defecto en la detección de puentes (Algoritmo A)

Existe además otra fuente de error importante en la detección de bordes que afecta directamente a la precisión del resultado. Este error es debido a una de las características de la tecnología LiDAR, concretamente a la absorción del haz láser por parte del agua.

El sistema de medición LiDAR se basa en la emisión de pulsos láser que son reflejados por las superficies sólidas y devueltos de nuevo hacia el emisor. No obstante, cuando el rayo láser infrarrojo incide sobre una lámina de agua el haz es absorbido y no se obtiene información tridimensional. Ello conlleva a que en zonas de agua tales como piscinas o ríos no existan puntos LiDAR y constituyan zonas de sombra en el conjunto de datos LiDAR.

En los dados test 2 (Collado Villalba) existen numerosas piscinas así como un canal que discurre por el municipio de este a oeste en su zona central. Además este canal es cruzado por un puente (Figura 73). Por tanto, tal y como se observa en la siguiente figura, la escena contiene numerosas zonas de sombra donde no existe información tridimensional.

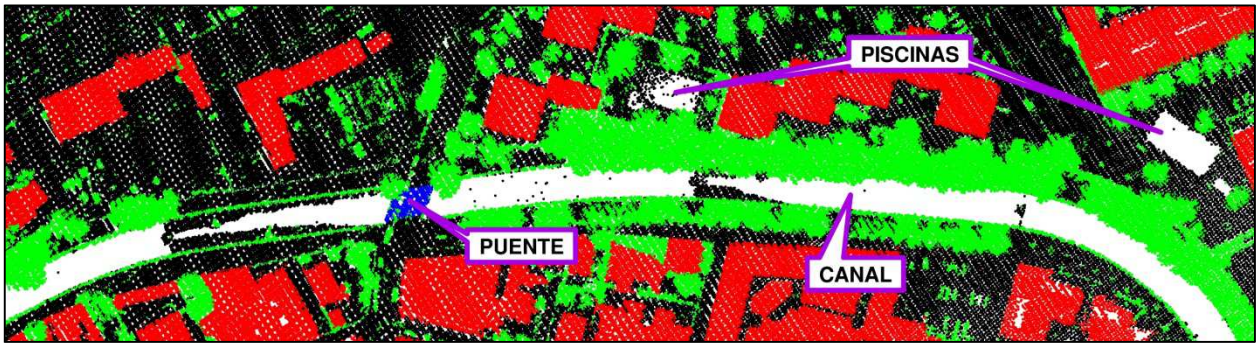


Figura 73. Zonas de sombra debido a la absorción de la señal por el agua. Datos test 2.

Estas zonas de sombra afectan directamente a la detección de puentes, ya que frecuentemente los puentes cruzan barrancos, ríos, canales u otras estructuras que contienen agua.

El algoritmo propuesto en el apartado anterior (Algoritmo A), detecta los bordes o límites de los puentes para posteriormente clasificar la totalidad de la estructura. Para identificar los bordes el algoritmo localiza desniveles verticales existentes dentro de la clase terreno. No obstante, si el puente cruza una zona de sombra no existe información altimétrica sobre el lecho del canal o río adyacente y por tanto, no es posible detectar dichos desniveles verticales.

Tomando como ejemplo el puente que cruza el canal en los datos test 2 (Collado Villalba) (Figura 74), puede apreciarse que existe una zona de sombra sin datos LiDAR colindantes al puente (Figura 74a), debido a ello solamente tres puntos del límite del puente son clasificados como borde y por tanto un importante error de omisión se transmite a la clasificación final del puente (Figura 74.b).

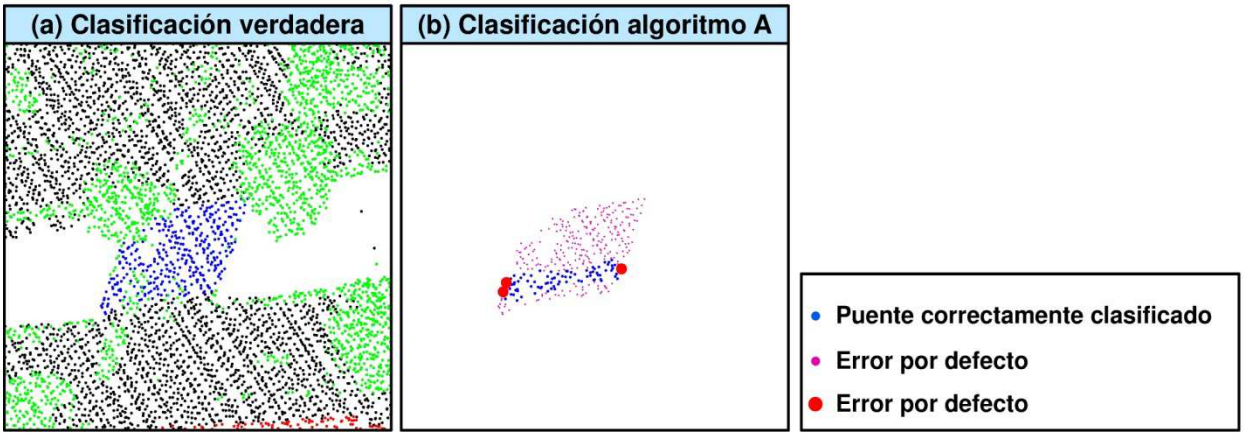


Figura 74. Error de omisión en la clasificación de puentes. Zonas de sombra. Zona test 2.

Los dos tipos de errores de omisión deben de ser evitados. Para ello, se desarrolla un nuevo algoritmo que tiene en cuenta los aspectos anteriormente descritos y que minimiza los errores en la detección de puentes.

III.7.2. Descripción del algoritmo programado

El nuevo algoritmo es una modificación del anterior que incluye análisis adicionales para minimizar los dos tipos de errores detectados tras ejecutar el algoritmo A. En consecuencia,

la nueva aplicación debe reducir el error de omisión que existe en el inicio y fin de los puentes (error de tipo 1) así como detectar los bordes de los puentes que son colindantes al agua y por tanto adyacentes a zonas de sombra (error de tipo 2). Para ello el algoritmo incluye dos nuevos procesos en la detección de bordes, mientras que la detección de los puntos de puente existentes entre los bordes no sufre modificaciones y no es descrita al incluirse ya en la descripción del algoritmo A. Seguidamente se exponen los dos análisis adicionales que incorpora el nuevo algoritmo.

- **Eliminación del error por defecto en los extremos del puente (error de tipo 1)**

Tras analizar detalladamente los resultados obtenidos al ejecutar el algoritmo A, se observa que la utilización de un pequeño desnivel como parámetro umbral (*UIZ_Borde*) en la detección de los bordes de los puentes produce errores de comisión. Como contrapartida, la utilización de un desnivel alto ocasiona errores de omisión en el inicio y el fin de los puentes. Dado que el error de comisión es difícil de eliminar se opta por desarrollar un nuevo algoritmo que detecte los puntos de borde mediante dos procesos concatenados. El nuevo algoritmo se basa en aplicar un parámetro umbral en desnivel restrictivo para detectar los bordes de los puentes sin introducir errores de comisión para posteriormente relajar el parámetro umbral e iterar el proceso de detección de bordes en los puntos del terreno que están próximos a los bordes anteriormente detectados, consiguiendo así reducir o incluso eliminar los errores de omisión existentes en los extremos del puente sin introducir nuevos errores por exceso en la clasificación.

El proceso puede apreciarse en la siguiente imagen (Figura 75). Se trata del mismo ejemplo propuesto con anterioridad que se corresponde con un puente de los datos test 2 (Collado Villalba). El algoritmo ejecuta en primer lugar el proceso para detectar bordes de puentes con un desnivel alto como parámetro umbral (*UIZ_Borde=4m*). Con ello se consigue detectar la mayor parte de los bordes del puente sin introducir errores de comisión (Figura 75a). Seguidamente, la aplicación selecciona los puntos del terreno que están próximos a los bordes anteriormente detectados. Para ello se determina el vecindario cilíndrico vertical de cada uno de los puntos de borde y los puntos pertenecientes al terreno incluidos en él son marcados en un vector auxiliar (Figura 75b). En este proceso, un radio de 8 m para el vecindario cilíndrico ofrece buenos resultados. Por último, se aplica de nuevo el proceso para detectar bordes pero únicamente sobre los puntos del terreno seleccionados utilizando un parámetro umbral en desnivel menos restrictivo (*UIZ_Borde=1m*), consiguiendo así clasificar los puntos de borde próximos a los extremos del puente no detectados con anterioridad (Figura 75c). Mediante este doble proceso la totalidad de los puntos pertenecientes al borde del puente son detectados sin introducir errores de comisión (Figura 75d).

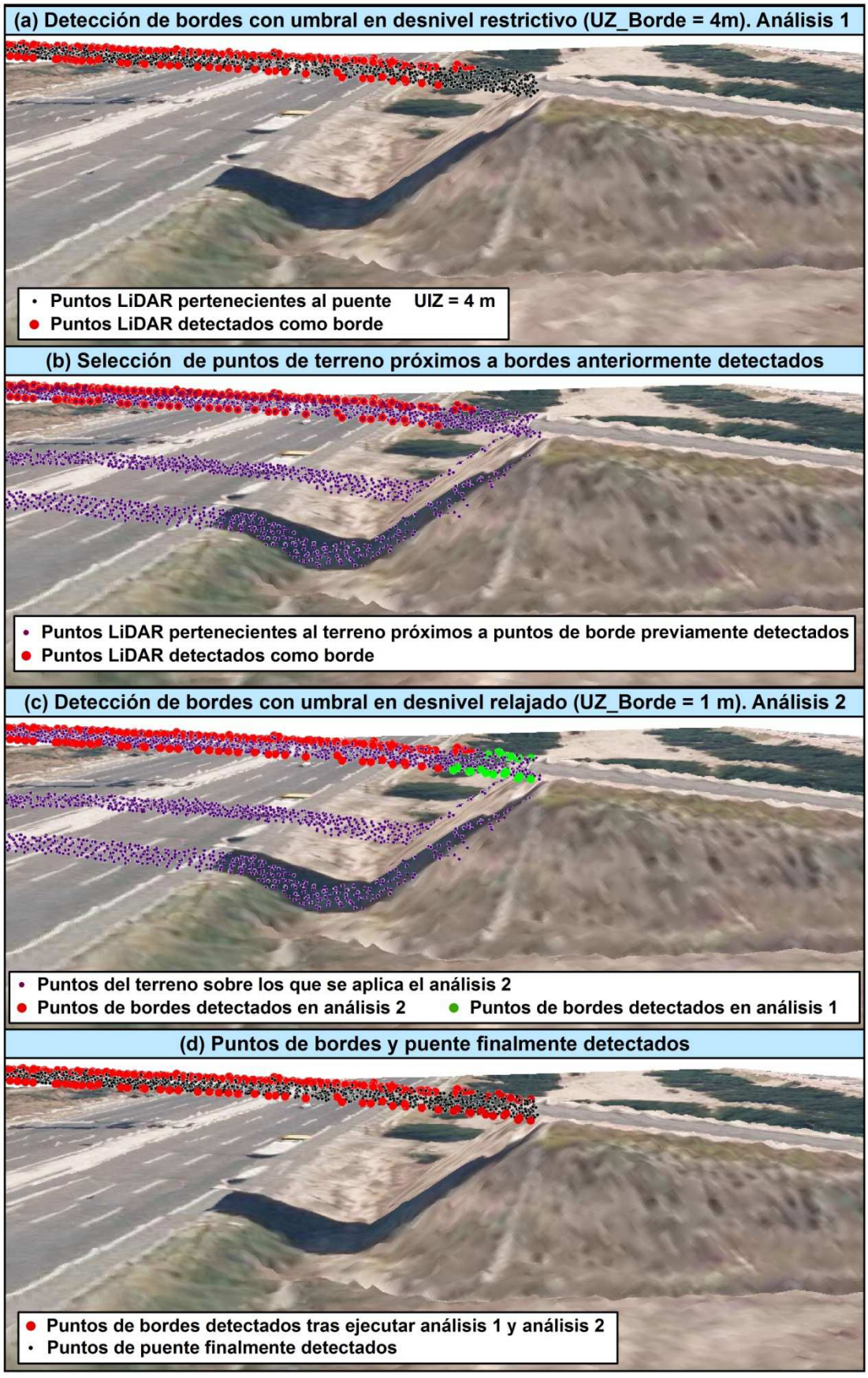


Figura 75. Eliminación de errores de omisión en extremos de los puentes. Algoritmo B.

- **Detección de bordes colindantes al agua y por tanto adyacentes a zonas de sombra (error de tipo 2)**

En la detección de bordes de puente se producen errores de omisión en aquellas estructuras que discurren sobre torrentes de agua. Ésto es debido a que los pulsos láser infrarrojos son absorbidos por el agua y por tanto se generan zonas de sombra donde no existe información tridimensional. Para solucionar este problema, se incluye un nuevo análisis donde para cada punto del terreno se analiza su vecindario con el fin de detectar zonas donde no existen datos LiDAR. El proceso sigue una metodología similar a la utilizada para detectar los puntos de puentes que están situados entre los bordes y la usada en el análisis de la continuidad estructural en la dirección principal del puente. Básicamente, el algoritmo genera el vecindario cilíndrico vertical para cada punto del terreno y lo divide en sectores. Posteriormente se contabiliza el número de puntos LiDAR que contiene cada uno de los sectores, de tal forma que si un sector no contiene ningún pulso LiDAR significa que es colindante a una zona de sombra.

Tras realizar diferentes pruebas experimentales y analizar los resultados se opta por dividir el vecindario en sectores iguales de 50^g de amplitud. La totalidad del proceso se muestra en la Figura 76. En ella se aprecia un puente que discurre sobre un canal donde no existen datos debido a la absorción de los pulsos LiDAR por parte del agua. Se trata de un puente perteneciente a los datos test 2 (Collado Villalba). El proceso se realiza para la totalidad de puntos clasificados como terreno. No obstante, en el ejemplo se aplica únicamente a dos puntos base (Figura 76a). En primer lugar, el algoritmo genera el vecindario cilíndrico para cada punto base y lo divide en sectores de igual amplitud (50^g) (Figura 76b). Seguidamente, la aplicación clasifica cada uno de los puntos LiDAR del vecindario en función del sector al que pertenece para posteriormente recorrer los distintos sectores y localizar aquellos que no contienen puntos LiDAR. En el ejemplo propuesto, el punto base 2 no pertenece a un punto de borde y por tanto todos sus sectores contienen información tridimensional, esto no ocurre en el punto base 1, donde existen dos sectores que no contienen información (Figura 76c). Esta ausencia de información es debida a que el punto pertenece al borde de un puente cuyo límite es adyacente a una zona de sombra y en consecuencia el punto es clasificado finalmente como borde (Figura 76d). Este procedimiento se aplica a la totalidad de puntos de la clase terreno, consiguiendo así detectar los bordes de los puentes que discurren sobre el agua (Figura 76e).

Para clasificar los puntos en los diferentes sectores se genera una matriz de igual dimensión que el número de sectores existente en el vecindario. Posteriormente se recorren la totalidad de puntos del vecindario clasificándolos a partir de su acimut en su sector asociado. Por último, basta recorrer la matriz detectando aquellos sectores que no contienen ningún punto. En caso de que algún sector no contenga punto alguno, el punto base es clasificado automáticamente como punto de borde.

La utilización de una matriz cuyos registros son contadores, establece un sistema de codificación en sectores muy eficiente computacionalmente, consiguiendo obtener precisos resultados en datos LiDAR dispuestos de forma aleatoria en un tiempo de cómputo aceptable.

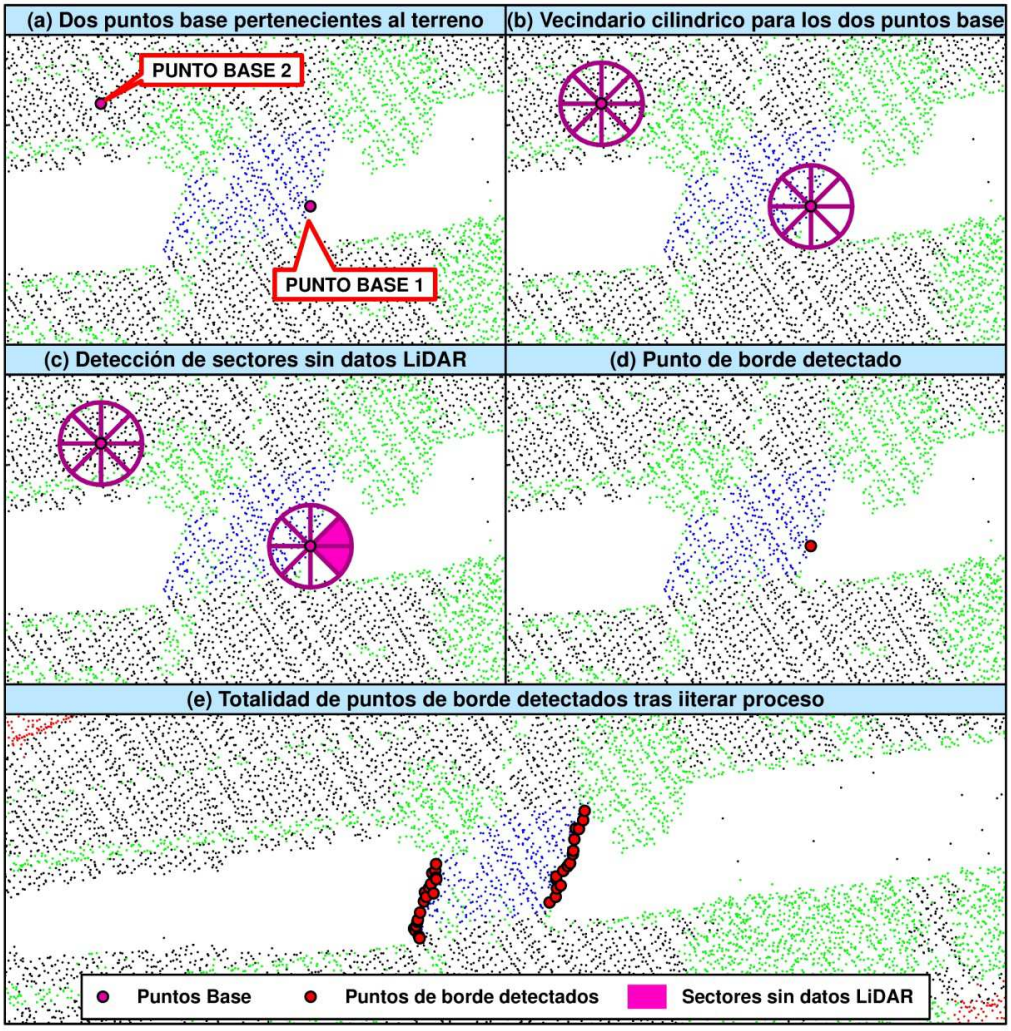


Figura 76. Detección de bordes de puentes que discurren sobre agua. Algoritmo B.

La totalidad de los análisis adicionales que incorpora el algoritmo forman parte del proceso de detección de bordes, ya que la clasificación de puntos de puente situados entre bordes no sufre modificación alguna con respecto al algoritmo A. La Figura 77 muestra el diagrama de flujo que sigue el algoritmo. Para detectar los bordes el algoritmo aplica el mismo proceso definido en el algoritmo A con un parámetro umbral en desnivel restrictivo (análisis 1). Con ello se consigue detectar el grueso de los bordes de puentes. Seguidamente se seleccionan los puntos del terreno que están próximos a los bordes previamente detectados y se aplican dos análisis adicionales. El primer análisis (análisis 2) consiste en ejecutar de nuevo sobre el vecindario el proceso de detección de bordes pero esta vez con un parámetro umbral en desnivel más relajado, consiguiendo con ello eliminar los errores de omisión que se dan en los extremos de la estructura (errores de tipo 1). El segundo análisis adicional (análisis 3) está diseñado para detectar bordes de puentes adyacentes a cauces de agua y por tanto a zonas de sombra. Básicamente este proceso consiste en dividir en sectores el vecindario de los puntos pertenecientes al terreno y buscar sectores que no contienen ningún punto LiDAR, consiguiendo con ello reducir el error de tipo 2. Finalmente la totalidad de bordes son marcados en un vector de marcadores y los puentes son detectados mediante la detección de puntos válidos situados entre bordes y el análisis de la continuidad estructural en la dirección principal del puente. Este último proceso no sufre modificación alguna con respecto al algoritmo A.

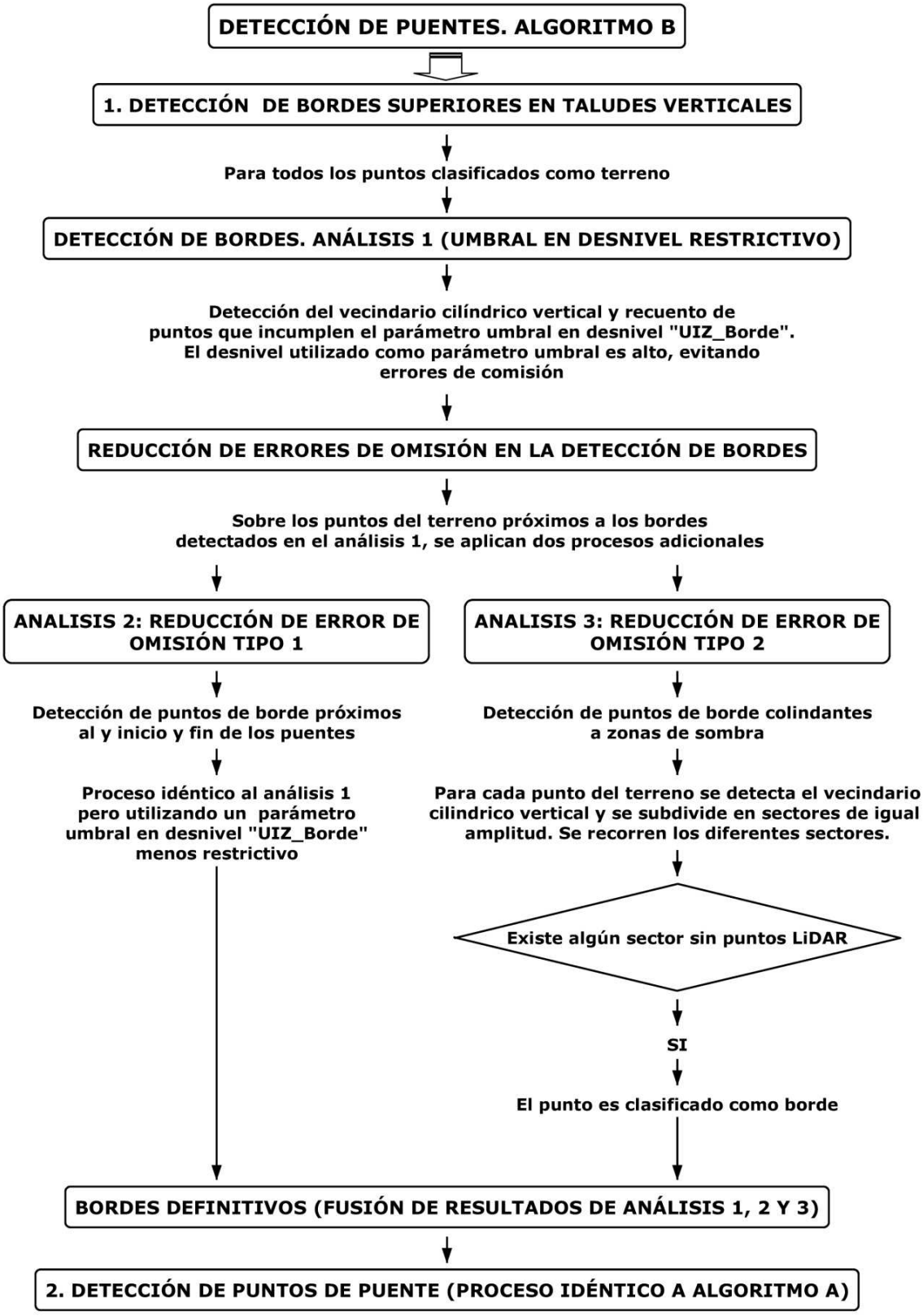


Figura 77. Diagrama de flujo. Detección de puentes (Algoritmo B)

III. 8. Detección de edificios así como vegetación y otros pequeños objetos. Algoritmo A: Análisis de textura

III.8.1. Motivación para el desarrollo del algoritmo y relación con los antecedentes

El procesado de datos LiDAR en entornos urbanos tiene multitud de aplicaciones tales como modelos hidrológicos urbanos (Sole et al. 2012), detección automática de cambios en edificaciones (Matikainen et al. 2010), actualización de mapas urbanos (Sánchez y Lerma 2012) o detección y evaluación de daños producidos por catástrofes naturales (Trinder y Salah 2012). En todos los casos, identificar las edificaciones y diferenciarlas de la vegetación y otros objetos urbanos resulta un proceso esencial. Con este propósito, una práctica común es combinar los datos LiDAR con otras fuentes de datos como fotografías aéreas (Kwak et al. 2012, Moussa y Sheimy 2012, Nakano y Chikatsu 2012) o imágenes multiespectrales (Vögtle y Steinle 2000; Matikainen et al. 2010; Hartfield et al. 2011; Moussa y Sheimy 2012; Wang y Li 2013). No obstante, el objetivo del presente trabajo es desarrollar algoritmos que actúen directamente sobre los datos LiDAR brutos, sin utilizar otras fuentes de datos adicionales, por lo que otros enfoques deben de tenerse en consideración.

En este sentido, cabe destacar los procesos que utilizan análisis de textura para detectar edificios y separarlos de la vegetación y otros pequeños objetos. Para realizar el análisis suele utilizarse la altitud de los puntos LiDAR (Hug 1997; Maas 1999; Oude Elberink y Maas 2000) o bien la rugosidad (Brunn y Weidner 1998). Cabe mencionar que la gran mayoría de las trabajos existentes aplican el análisis de textura sobre los datos LiDAR interpolados en una malla regular (GRID). No obstante todos los algoritmos implementados en el presente estudio trabajan con los datos LiDAR brutos dispuestos de forma aleatoria para no perder precisión espacial, por lo que deberá desarrollarse una nueva metodología que salve este obstáculo.

Por último, otra característica muy útil para diferenciar los edificios de la vegetación es la capacidad de detectar pulsos con múltiples retornos. Cuando un pulso láser se refleja sobre una superficie que no es completamente opaca (como la vegetación) es posible recibir distintos ecos de la señal, el número de ecos o reflexiones es variable aunque como norma general suele estar disponible al menos la información asociada al primer y último pulso. En este sentido diferentes trabajos muestran las ventajas de utilizar la información multi-retorno para detectar la vegetación (Pfeifer et al. 2001; Alharthy y Bethel 2002; Tovari y Vögtle 2004; Meng, Wang y Currit 2009).

Para detectar edificios y separarlos de la vegetación en el presente apartado se expone un nuevo algoritmo que segmenta los datos LiDAR en superficies continuas para posteriormente analizar las características de las agrupaciones obtenidas y aplicar reglas de decisión a fin de obtener la clasificación. Para ello, se implementa un proceso análogo al análisis de textura que tiene en cuenta el tamaño de las regiones y que permite detectar vegetación y otros pequeños objetos con una elevada precisión.

III.8.2. Descripción del algoritmo programado

Llegado a este punto, se dispone de los datos LiDAR clasificados en tres clases: terreno, puentes y resto de entidades. Dentro de la tercera clase se incluyen los edificios así como la vegetación y otros pequeños objetos tales como automóviles, bancos, juegos infantiles en los parques, etc.

En el presente apartado, se presenta un algoritmo para diferenciar los edificios de la vegetación, obteniendo finalmente un total de cuatro clases: terreno, puentes, edificios y vegetación.

La clase vegetación, además de contener la cobertura vegetal incluirá otras pequeñas entidades próximas al terreno tales como automóviles, bancos, personas, perros, etc. En cualquier caso, la mayor parte de puntos serán de vegetación ya que mediante la detección de errores groseros la mayoría de objetos urbanos de pequeño tamaño habrán sido previamente filtrados.

Uno de los enfoques que mejores resultados obtiene para diferenciar la vegetación y otros pequeños objetos de los edificios es aplicar técnicas de análisis de texturas. El algoritmo que se presenta en este apartado aplica este enfoque sobre los datos LiDAR brutos dispuestos de forma aleatoria. La aplicación utiliza como objeto de análisis las regiones obtenidas tras segmentar la escena en superficies continuas. Básicamente, la metodología propuesta tiene en cuenta que la vegetación está constituida por un conglomerado de regiones de pequeño tamaño, mientras que esto no ocurre en los edificios donde existen grandes planos de techo. Este aspecto puede apreciarse en la Figura 78. En ella se muestran las regiones que tienen una superficie menor a 10 m^2 para la zona central de los datos test 2 (Collado de Villalba). Como puede apreciarse, las zonas de vegetación están constituidas por un conglomerado de regiones menores a 10 m^2 y por tanto, para detectar la mayor parte de la vegetación bastará con identificar las zonas donde se agrupan este tipo de regiones.

Figura 78. Regiones con superficie menor a 10 m^2. Datos test 2.

Para localizar las zonas donde se agrupan pequeñas regiones, el algoritmo recorre los puntos LiDAR que pertenecen a regiones de pequeño tamaño y para cada uno de ellos determina el vecindario cilíndrico vertical. Seguidamente, la aplicación comprueba si existe una densidad alta de regiones de pequeño tamaño en el vecindario y analiza su distribución espacial, de forma que un punto es clasificado como vegetación siempre y cuando exista una alta densidad de regiones pequeñas distribuidas de forma uniforme en todo el vecindario. Para analizar la distribución espacial de las regiones, el vecindario se divide en sectores de igual amplitud angular. Así mismo, dichos sectores son divididos en semisectores, tal y como muestra la siguiente imagen (Figura 79).

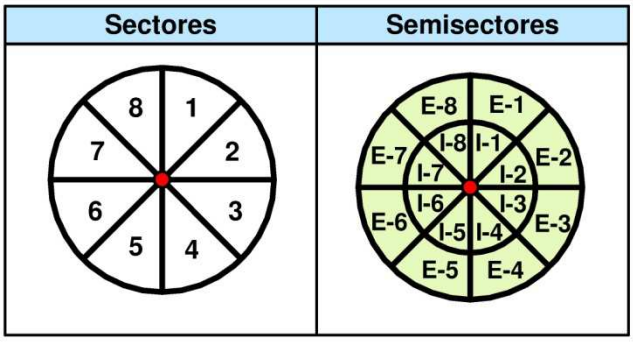


Figura 79. División del vecindario en sectores y semisectores

Tras realizar distintas pruebas y analizar los resultados puede afirmarse que la división del vecindario en sectores de 50^g ofrece precisos resultados. En este caso, tal y como se aprecia en la imagen anterior (Figura79) el vecindario queda dividido en 8 sectores iguales que a su vez se subdividen en 16 semisectores. Esta división del espacio que ocupa el vecindario sirve para determinar si las regiones de poco tamaño están distribuidas de forma uniforme en todo el vecindario.

A la hora de determinar si existe una densidad suficientes de puntos pertenecientes a regiones pequeñas dentro del conglomerado de regiones del vecindario, así como si la distribución de dichas regiones es uniforme, el algoritmo precisa los siguientes parámetros umbrales:

- *Area_Maxima_Region*: Superficie máxima de una región para ser considerada como vegetación o un pequeño objeto. El tamaño recomendado es de 10 m^2.

- *Umbral_Porcentaje*: Porcentaje de puntos mínimo pertenecientes a regiones pequeñas en cada sector para determinar si el punto base es susceptible de pertenecer a vegetación.

- *Umbral_Puntos_Semisectores*: Número de puntos mínimos pertenecientes a regiones pequeñas en los semisectores. Sirve para asegurar que el conglomerado de regiones pequeñas se distribuye de forma uniforme dentro de cada sector.

Para determinar si existe una alta densidad de regiones pequeñas en el vecindario y analizar su distribución espacial el algoritmo genera para cada punto objeto de análisis una matriz de contadores de igual número de filas que el número total de sectores donde se almacenará la información asociada a cada sector. La matriz tendrá siete columnas que registrarán la siguiente información:

Columna 1: Número de puntos de regiones pequeñas que no pertenecen al terreno (sector).
Columna 2: Número de puntos de regiones grandes que no pertenecen al terreno (sector).
Columna 3: Número de puntos de regiones pequeñas que no pertenecen al terreno (semisector interior).
Columna 4: Número de puntos de regiones pequeñas que no pertenecen al terreno (semisector exterior).
Columna 5: Número de puntos de regiones grandes incluyendo terreno (sector).
Columna 6: Número de puntos pertenecientes al terreno (sector).
Columna 7: Número de puntos LiDAR independientemente de su clasificación (sector).

Una vez generada la matriz el algoritmo la recorrerá analizando la información contenida en las siete columnas para cada sector y será capaz de determinar si existe un conglomerado de regiones de pequeño tamaño distribuidas de forma uniforme por todo el vecindario.

Antes de detallar los distintos análisis que realiza el algoritmo es necesario describir los distintos casos prácticos con los que la aplicación puede encontrarse.

Una gran parte de las regiones que deben de clasificarse como vegetación o pequeños objetos se corresponden con arbustos o árboles de pequeño tamaño, así como vehículos u otros pequeños objetos (bancos, señales de tráfico...). En estos casos el conglomerado de regiones de pequeño tamaño abarcará una escasa superficie y el punto base para los cálculos estará rodeado por puntos de tierra en todas las direcciones. Por otra parte, el algoritmo deberá identificar zonas de árboles o vegetación densa donde no existen pulsos LiDAR que hayan podido penetrar hasta el terreno y como consecuencia, el vecindario no contendrá puntos de tierra en todas las direcciones. En estas zonas el conglomerado de regiones de pequeño tamaño abarcará superficies más amplias. La aplicación es capaz de diferenciar ambos casos y aplicar un análisis distinto a cada uno de ellos. Los dos casos que el algoritmo debe detectar así como el tipo de análisis que aplicará se muestran en la Figura 80. El ejemplo muestra la zona central de los datos test 2 (Collado de Villalba), donde se localizan dos puntos bases pertenecientes a cada uno de los dos casos posibles (Figura 80a). En las imágenes de detalle (Figura 80b y Figura 80c) aparecen las regiones que tienen una superficie pequeña (menores de 10 m^2) cada una de un color así como los puntos de tierra. El caso 1 (Figura 80b) se corresponde con un pequeño arbusto y por tanto está definido por un conglomerado de pequeñas regiones que abarca una escasa superficie. Además, existen puntos de tierra en todos los sectores. Sin embargo, el caso 2 (Figura 80c), muestra una zona de vegetación densa que está definida por un conglomerado de regiones pequeñas que abarca una gran superficie. En este caso, no existen puntos de tierra en todos los sectores ya que la densa vegetación impide que el haz láser llegue hasta el terreno. El algoritmo implementado es capaz de identificar los dos casos y aplicar un análisis diferenciado (análisis 1 y análisis 2) en función del caso concreto objeto de análisis. Para diferenciar ambos casos, una vez definido el vecindario del punto base y dividido en sectores y semisectores, el algoritmo recorre la matriz de sectores y comprueba que la columna 6 es mayor que cero para todos los sectores. En caso afirmativo significará que existen puntos de tierra en todos los sectores y por tanto estaremos en el caso 1; en el caso contrario, no existirán puntos de tierra en todos los sectores y nos encontraremos en el caso 2.

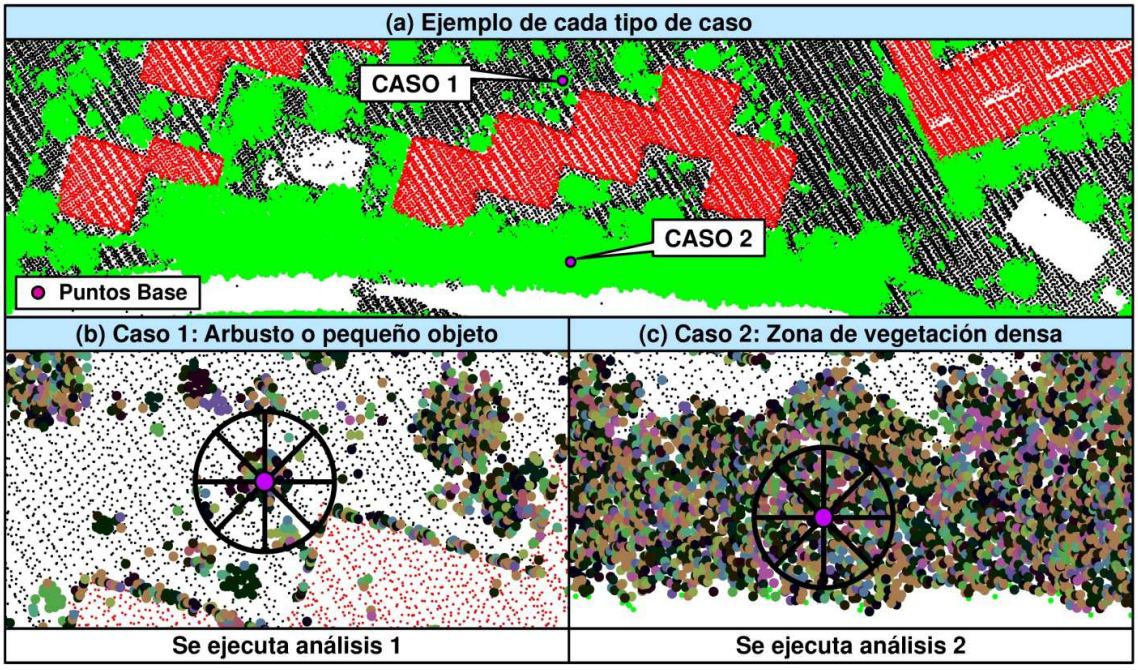


Figura 80. Distintos casos en los que se aplica análisis de textura

Antes de pasar a describir los dos tipos de análisis que ejecuta el algoritmo (análisis 1 y análisis 2) en función de la posición relativa de los puntos del terreno con respecto al punto base, debe de tenerse en cuenta cómo actúa el algoritmo cuando el punto base es colindante a una zona de sombra. Tal y como se describe en el siguiente apartado (Apartado III. 9), uno de los problemas del clasificador angular es que para su correcto funcionamiento precisa la existencia de datos LiDAR en todas las direcciones y, en consecuencia, el algoritmo falla cuando los puntos objeto de análisis están en el límite de la zona de estudio o son adyacentes a una lámina de agua que ha absorbido los pulsos láser. En ambos casos se trata de puntos que son colindantes a zonas de sombra donde no existe información tridimensional. En este sentido, el presente algoritmo consigue salvar este problema mediante el análisis del número de puntos LiDAR que contiene cada uno de los sectores.

En la Figura 81 se muestran dos casos concretos de puntos pertenecientes a vegetación que son adyacentes a zonas de sombra. Se trata de dos puntos de árboles situados en la zona central de los datos test 2 (Collado de Villalba). El primer punto base es colindante a la zona de estudio y en su parte este no existen datos, mientras que el segundo punto base pertenece a una zona de vegetación densa colindante a un canal de agua y, por tanto, a una zona de sombra (Figura 81a). En las imágenes de detalle aparecen las regiones pequeñas (menores a 10 m^2) cada una en diferentes colores. Para el primer punto base (Figura 81b), puede apreciarse cómo el límite de la zona de estudio constituye una barrera donde su zona este no contiene datos. Consecuentemente, los sectores situados en esta parte (sectores 1, 2 y 3) no tienen ningún punto LiDAR. De forma análoga, el canal no contiene ningún punto LiDAR y por tanto los sectores situados más al norte para el punto base 2 tampoco contienen información (sectores 1 y 8) (Figura 81c). Esta información queda registrada en la matriz de sectores en su columna 7. Por tanto, la localización de puntos adyacentes a zonas de sombra es inmediata y basta con comprobar si la columna 7 contiene un número mayor que cero.

Gracias al sistema de codificación del vecindario en sectores es posible eliminar automáticamente del proceso las zonas de sombra. Para ello, al ejecutarse el análisis correspondiente aquellos sectores que no contienen datos (columna 7=0) no serán incluidos en los cálculos. En lo sucesivo, los sectores que contienen datos y por tanto no están asociados a una zona de sombra serán denominados "sectores válidos".

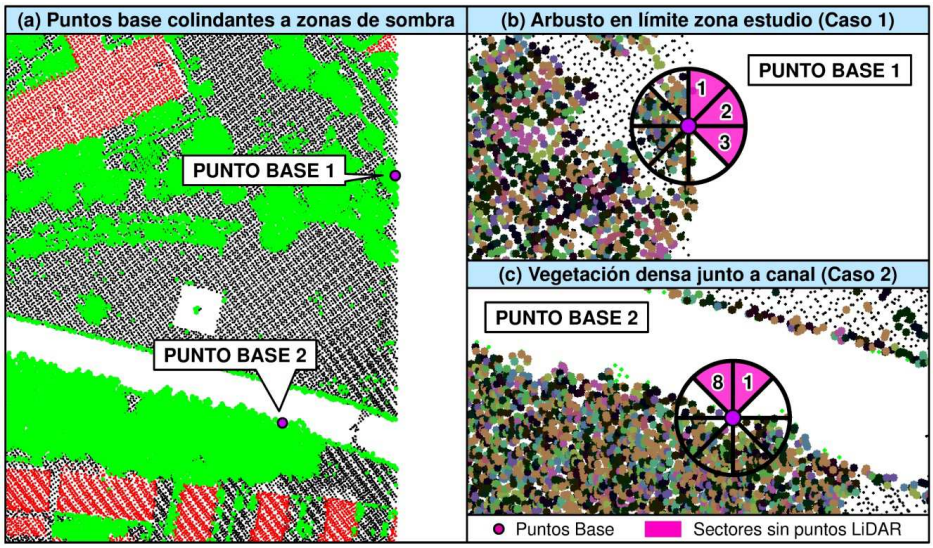


Figura 81. Aplicación de análisis de textura junto a zonas de sombra

En el ejemplo propuesto (Figura 81), para el primer punto base los sectores 1,2,3 no pasarán a formar parte del análisis (Figura 81b). Una vez eliminados del proceso el

algoritmo comprobará si existen puntos de tierra en el resto de sectores (sectores válidos), dado que en este caso concreto existen puntos de tierra en todos los sectores válidos se tratará de un pequeño arbusto rodeado en todas las direcciones por puntos de tierra y se aplicará el análisis 1. Por el contrario, tras eliminar los sectores 1 y 8 en el procesado del punto base 2 (Figura 81c), todos los sectores válidos no contienen puntos de tierra, por lo que se tratará de una densa cobertura de vegetación y se aplicará el análisis 2. A continuación se describen con detalle los dos tipos de análisis.

- **Análisis 1: puntos pertenecientes a pequeños arboles, arbustos u otros objetos de escaso tamaño**

Como se ha comentado con anterioridad, este análisis se realizará para aquellas regiones de pequeño tamaño (menores a 10 m^2) que tienen puntos de tierra en todas las direcciones dentro de su vecindario próximo. Debido a la proximidad de puntos del terreno en todas las direcciones este proceso analiza puntos susceptibles de pertenecer a arbustos o pequeños árboles, así como objetos de pequeño tamaño tales como automóviles, farolas, señales de tráfico, etc. Este análisis se ejecutará únicamente en el caso de que la columna 6 sea mayor que cero para todos los sectores válidos. En este análisis los puntos pertenecientes al terreno no son considerados en los cálculos. Básicamente, la aplicación calcula el porcentaje de puntos pertenecientes a regiones pequeñas con respecto al total de puntos para cada uno de los sectores, de forma que si para todos los sectores el porcentaje calculado es mayor a un cierto parámetro umbral (*Umbral_Porcentaje*), el punto es clasificado como vegetación o pequeño objeto.

Para realizar este análisis bastará con recorrer la matriz de sectores y para cada sector válido calcular el siguiente porcentaje:

Porcentaje= Columna 1/(Columna 1+ Columna 2)

Si en todos los casos se cumple que el porcentaje es mayor al umbral establecido (*Umbral_Porcentaje*) el punto será clasificado como vegetación o pequeño objeto.

- **Análisis 2: puntos pertenecientes a zonas de vegetación densa**

En este caso se incluyen aquellos puntos que pertenecen a regiones pequeñas que no están rodeadas por puntos de tierra en todas sus direcciones dentro de su vecindario próximo. El objetivo de este análisis es diferenciar los puntos que pertenecen a zonas de vegetación densa de aquellos que pertenecen a pequeñas regiones incluidas dentro de la clase edificio tales como chimeneas, pequeños planos de techo, balcones o puntos de fachada. En este sentido, a diferencia del análisis 1, los puntos pertenecientes al terreno formarán parte de los cálculos realizados.

El análisis que realiza el algoritmo consiste en determinar si existe en el vecindario un conglomerado de regiones de pequeño tamaño denso y distribuido de forma uniforme por todo el vecindario. Para ello realiza dos comprobaciones concatenadas, la primera de ellas consiste en determinar si el porcentaje de puntos pertenecientes a regiones pequeñas en cada sector es grande. El cálculo de porcentajes se realiza mediante el uso de la matriz de sectores a partir de la siguiente expresión:

Porcentaje= Columna 1/(Columna 1+ Columna 5)

En el caso de que para todos los sectores válidos el porcentaje calculado sea mayor que el parámetro umbral establecido (*Umbral_Porcentaje*), podrá asegurarse que existe en el vecindario del punto base una alta densidad de regiones pequeñas distribuidas en todos los sectores. Además, para asegurar que las regiones pequeñas se distribuyen de forma

uniforme dentro de cada sector, cada semisector deberá contener un número mínimo de puntos pertenecientes a regiones de pequeño tamaño. Para ello, se comprueba para cada sector válido que la columna 3 y 4 de la matriz de sectores es mayor al parámetro umbral establecido (*Umbral_Puntos_Semisectores*).

En el siguiente ejemplo (Figura 82) se muestra el proceso realizado para tres puntos base pertenecientes a los datos test 2 (Collado de Villalba) (Figura 82a). En las imágenes de detalle aparecen los puntos pertenecientes a regiones de gran tamaño, a regiones de pequeño tamaño, así como el terreno.

Para los tres puntos base propuestos, el algoritmo divide el vecindario en sectores y comprueba si existe terreno en todos ellos, como este no es el caso, se ejecuta para todos ellos el análisis 2.

El primero de los puntos base se corresponde con una pequeña agrupación perteneciente a la fachada de un edificio (Figura 82b). Tras determinar el vecindario y dividirlo en sectores y semisectores se aprecia como prácticamente la totalidad de los puntos incluidos en los diferentes sectores pertenecen a la clase terreno o a regiones de gran tamaño (tejado del edificio). Consecuentemente, los porcentajes calculados no cumplen con el mínimo exigido (*Umbral_Porcentaje*) y el punto base es finalmente clasificado como edificio.

El segundo de los puntos base pertenece a una región asociada a la chimenea de un edificio (Figura 82c). En este caso todos los sectores contienen mayoritariamente puntos pertenecientes a grandes regiones que se corresponden con planos de techo. Por lo que los porcentajes calculados tampoco superan el parámetro umbral establecido y el punto es clasificado como edificio.

El tercer y último punto base del ejemplo pertenece a una zona densa de árboles donde los pulsos LiDAR no han sido capaces de llegar hasta el terreno (Figura 82d). En este caso, la totalidad de puntos existentes en los sectores pertenecen a regiones de pequeño tamaño por lo que el porcentaje calculado para cada sector supera el mínimo establecido. Además. La distribución de las pequeñas regiones es uniforme a lo largo de cada sector y los semisectores cumplen con el número mínimo de puntos pertenecientes a pequeñas regiones (*Umbral_Puntos_Semisectores*). Por consiguiente, el punto es clasificado dentro de la clase Vegetación y pequeños objetos.

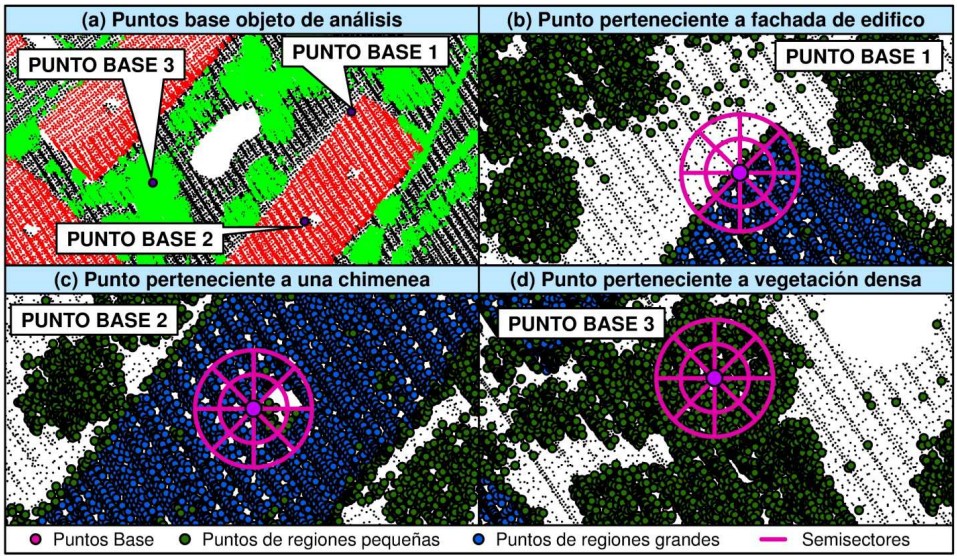


Figura 82. Análisis de textura. Ejemplos de análisis 2.

El diagrama de flujo donde se aprecian todas las fases que componen el algoritmo es el que se muestra a continuación (Figura 83).

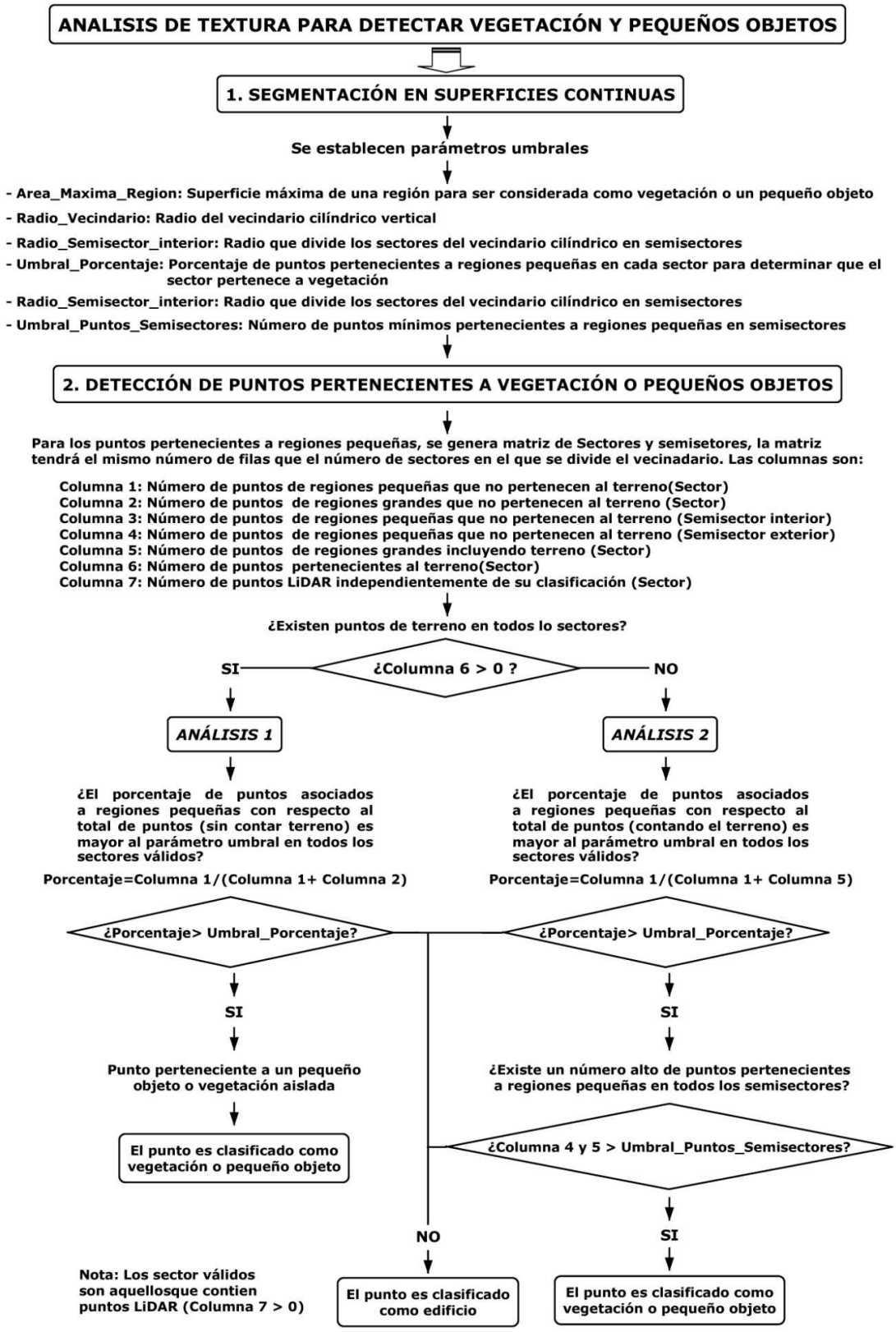


Figura 83. Diagrama de flujo. Análisis de textura para diferenciar edificios de vegetación y pequeños objetos.

III. 9. Detección de edificios así como vegetación y otros pequeños objetos. Algoritmo B: análisis de textura y clasificador angular

III.9.1. Motivación para el desarrollo del algoritmo y relación con los antecedentes

El análisis de texturas (Algoritmo A) no funciona correctamente en los límites exteriores de las zonas de vegetación densa (Figura 88c). Ello se debe a la inclusión de un alto número de puntos pertenecientes al terreno en los sectores más alejados de la parte central de la vegetación. Consecuentemente, solamente algunos sectores contendrán puntos pertenecientes al terreno y el algoritmo ejecuta el análisis 2. Finalmente, dada la gran presencia de puntos del terreno en algunos sectores del vecindario, las pequeñas regiones objeto de análisis no serán clasificadas como vegetación por ser susceptibles de pertenecer a otro tipo de estructuras propias de edificios (chimeneas, balcones, pequeños planos de techo, etc.).

Con el fin de subsanar los errores producidos por el análisis de texturas, otros enfoques deben de considerarse. En este sentido la utilización de información asociada a múltiples ecos de la señal resulta de gran utilidad para localizar vegetación y diferenciarla de los edificios (Pfeifer et al. 2001; Alharthy y Bethel 2002; Tovari y Vögtle 2004; Meng, Wang y Currit 2009). Además, los haces láser consiguen penetrar en gran parte dentro de la cobertura vegetal y capturar puntos del terreno, por lo que la distribución y posición relativa de los puntos del terreno con respecto a las regiones de menor tamaño también constituye una fuente de información adicional de gran utilidad. En este sentido, en el presente apartado se introduce el concepto de clasificador angular. Este clasificador utiliza dos características asociadas a los datos LiDAR: el tamaño de las regiones obtenidas tras segmentar la escena y la distribución de los puntos del terreno con respecto a las regiones de menor tamaño. Finalmente, se presenta un nuevo algoritmo que aúna técnicas de análisis de texturas con el clasificador angular, consiguiendo así realizar un análisis exhaustivo de las distintas características que poseen los datos LiDAR segmentados tales como la información multi-retorno, distribución espacial de puntos pertenecientes al terreno, el tamaño de las regiones obtenidas tras la segmentación, densidad del conglomerado de regiones de pequeño tamaño y distribución espacial de las mismas.

III.9.2. Descripción del algoritmo programado

En cuanto al tamaño de las regiones obtenidas tras segmentar la escena, la vegetación y otros pequeños objetos (automóviles, bancos...) vienen definidos por pequeñas regiones o agrupaciones, mientras que los tejados y terrazas de los edificios suelen estar definidos por planos de techo de gran tamaño. Por tanto, el análisis para detectar vegetación y otros pequeños objetos se aplicará únicamente a regiones de pequeño tamaño. A tal efecto, tal y como señala Moussa y El-Sheimy (2012), un parámetro umbral de 10 m^2 en cuanto a superficie máxima de las regiones ofrece buenos resultados.

En cualquier caso, un tamaño de región pequeña es una condición necesaria pero no suficiente para detectar la vegetación y otros pequeños objetos. Esto se debe a que en los edificios existen también estructuras de escaso tamaño (chimeneas, pequeños planos de techo, balcones...) asociadas a regiones pequeñas. Por tanto, es necesario introducir un análisis adicional donde toma especial relevancia el concepto de clasificador angular. Este nuevo clasificador analiza la distribución de los puntos de tierra próximos a las regiones de pequeño tamaño.

Una de las características de la tecnología LiDAR es que el pulso láser es capaz de penetrar la cobertura vegetal y detectar la tierra situada bajo ella. Estos puntos de tierra están distribuidos de forma aleatoria en todas las direcciones. En consecuencia las regiones que pertenecen a la clase vegetación o a pequeños objetos suelen estar rodeados en todas las direcciones por puntos de tierra, característica que no cumplen los edificios (Figura 84).

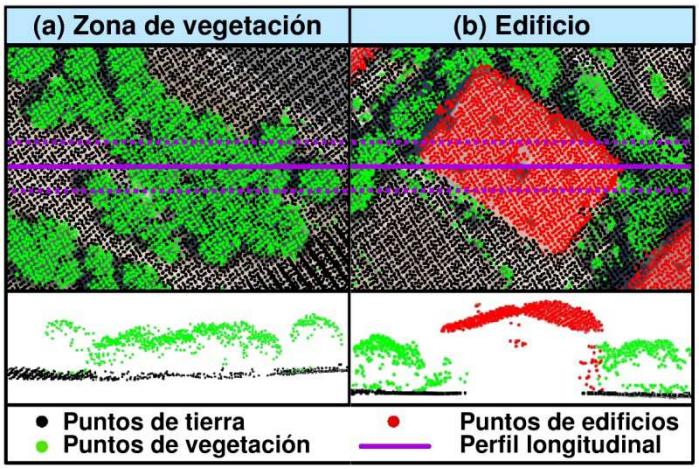


Figura 84. Distribución de los puntos de tierra en torno a vegetación y edificios.

Las regiones de pequeña superficie que pertenecen a la vegetación o a pequeños objetos suelen estar rodeados de puntos de tierra en todas las direcciones. Esta premisa no se cumple en las edificaciones debido a que las líneas de fachada actúan como barreras que impiden la ubicación de puntos de tierra dentro del área de edificación. Estas líneas de fachada definen un ángulo de 180° que puede llegar a 90° en las esquinas de los edificios. Desde esta perspectiva, para detectar regiones que pertenecen a la clase vegetación u otros pequeños objetos bastará con recorrer las agrupaciones de pequeño tamaño y analizar el vecindario cilíndrico vertical de cada uno de los puntos que las componen. El análisis del vecindario consiste en generar segmentos virtuales que unen el punto base objeto de estudio con los puntos del terreno detectados dentro del vecindario. Posteriormente los ángulos internos que forman los segmentos son calculados, en el caso de que todos los ángulos sean menores a 90° significará que no existen líneas de fachada y por tanto la región o agrupación no pertenecerá a un edificio.

El clasificador angular tiene en cuenta dos parámetros umbrales: primero, un umbral en distancia para determinar la proximidad de los puntos de tierra (radio del vecindario cilíndrico); y segundo, un umbral angular para considerar el ángulo mínimo definido por las líneas de fachada. El umbral en distancia debe de ser menor a la longitud de la fachada más pequeña existente en la zona de estudio; en la práctica, un umbral de 6 m ofrece buenos resultados. En cuanto al umbral angular, éste debe de ser menor a 90º por ser el ángulo mínimo que suele existir en las fachadas de los edificios (esquinas). En la Figura 85 se observa el funcionamiento del clasificador angular para diferentes entidades. En primer lugar se aplica el algoritmo sobre dos puntos pertenecientes a una edificación, en ambos casos, los segmentos virtuales que unen el punto base con los puntos de terreno existentes en el vecindario forman un ángulo mayor a 90º (concretamente de 145º y 230º), por tanto no pertenecen a regiones rodeadas por el terreno en todas las direcciones y ambas regiones son clasificadas como edificio. Además, el clasificador angular se ejecuta sobre un punto perteneciente a un automóvil y a un árbol. En estos dos últimos casos la totalidad de ángulos interiores que forman los segmentos son menores de 90º, por lo ningún punto es clasificado como edificio y automáticamente sus regiones asociadas son incluidas en la clase "Vegetación y pequeños objetos".

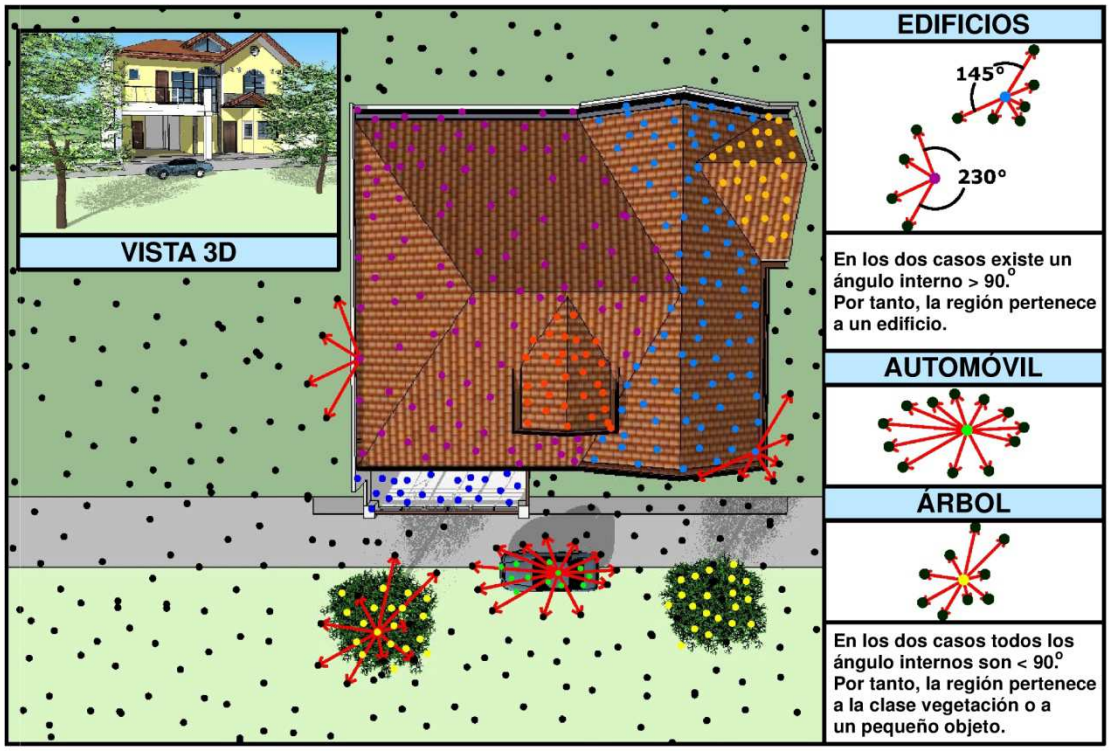


Figura 85. Funcionamiento del clasificador angular

Además de la distribución espacial de los puntos pertenecientes al terreno y de la capacidad de penetración de los pulsos LiDAR a través de la cubierta vegetal, en ocasiones los datos LiDAR incluyen información multi-retorno. Esta información es muy útil ya que los ecos múltiples se producen cuando el haz láser es reflejado por superficies que no son completamente opacas como la vegetación. Consecuentemente los haces LiDAR que contienen información multi-retorno se agrupan en zonas de vegetación. En la siguiente imagen (Figura 86) aparecen los pulsos lidar que contienen información multi-retorno para los datos test 2 (Collado Villalba). Tal y como puede apreciarse los ecos múltiples se agrupan en zonas arboladas donde la vegetación es densa.

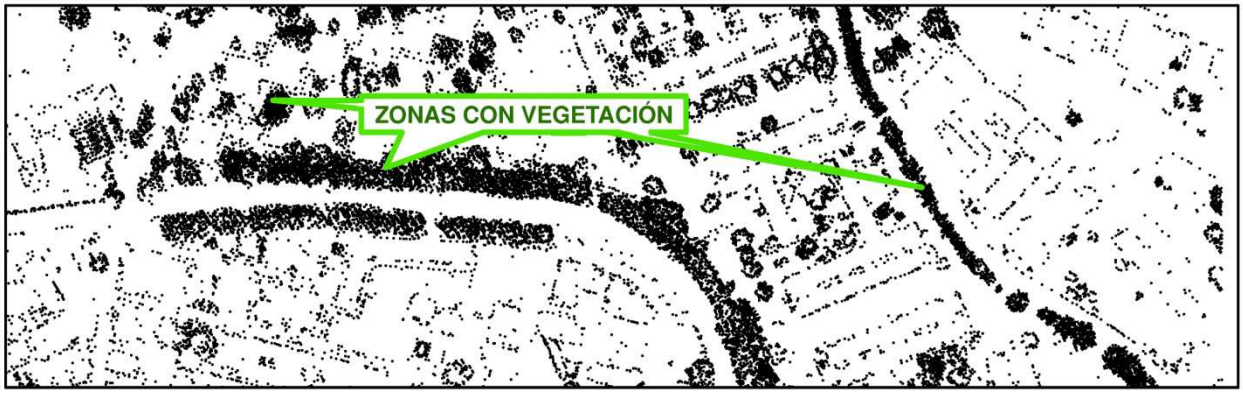


Figura 86. Información multi-retorno. Datos test 2 (Collado Villalba)

Esta fuente de información se incluye para mejorar lo resultados obtenidos por el clasificador angular. Para ello, los segmentos virtuales se generarán desde el punto base a los puntos de tierra próximos así como a aquellos que contienen información multi-retorno. Los diferentes procesos que realiza el algoritmo vienen detallados en el correspondiente diagrama de flujo (Figura 87). El algoritmo toma como datos de entrada los puntos LiDAR

segmentados en las diferentes superficies continuas que conforman la escena así como los identificadores de puntos que contienen información multi-retorno. Seguidamente se recorre la totalidad de puntos LiDAR que pertenecen a regiones de pequeño tamaño y para cada uno se genera su vecindario cilíndrico vertical. Del vecindario se extraen los puntos pertenecientes al terreno y aquellos que contienen información multi-retorno. Con los puntos extraídos se genera el vector de segmentos virtuales. Este vector constituye la base para calcular los ángulos interiores. Para ello, se recorre el vector y se calculan los acimutes desde el punto base a cada uno de los puntos contenidos en el mismo. Estos acimutes se almacenan en un vector (*Vector de acimutes)* que posteriormente es ordenado en orden creciente. Finalmente para calcular los ángulos interiores se recorre el *vector de acimutes ordenados* restando los acimutes dos a dos. Los ángulos son almacenados en un vector (*Vector de ángulos interiores*). El último paso consiste en recorrer el *vector de ángulos interiores* y detectar si alguno de ellos es mayor al umbral establecido (*Umbral_Angular*): en caso afirmativo el punto será clasificado como edificio; en caso contrario, será clasificado como vegetación o pequeño objeto.

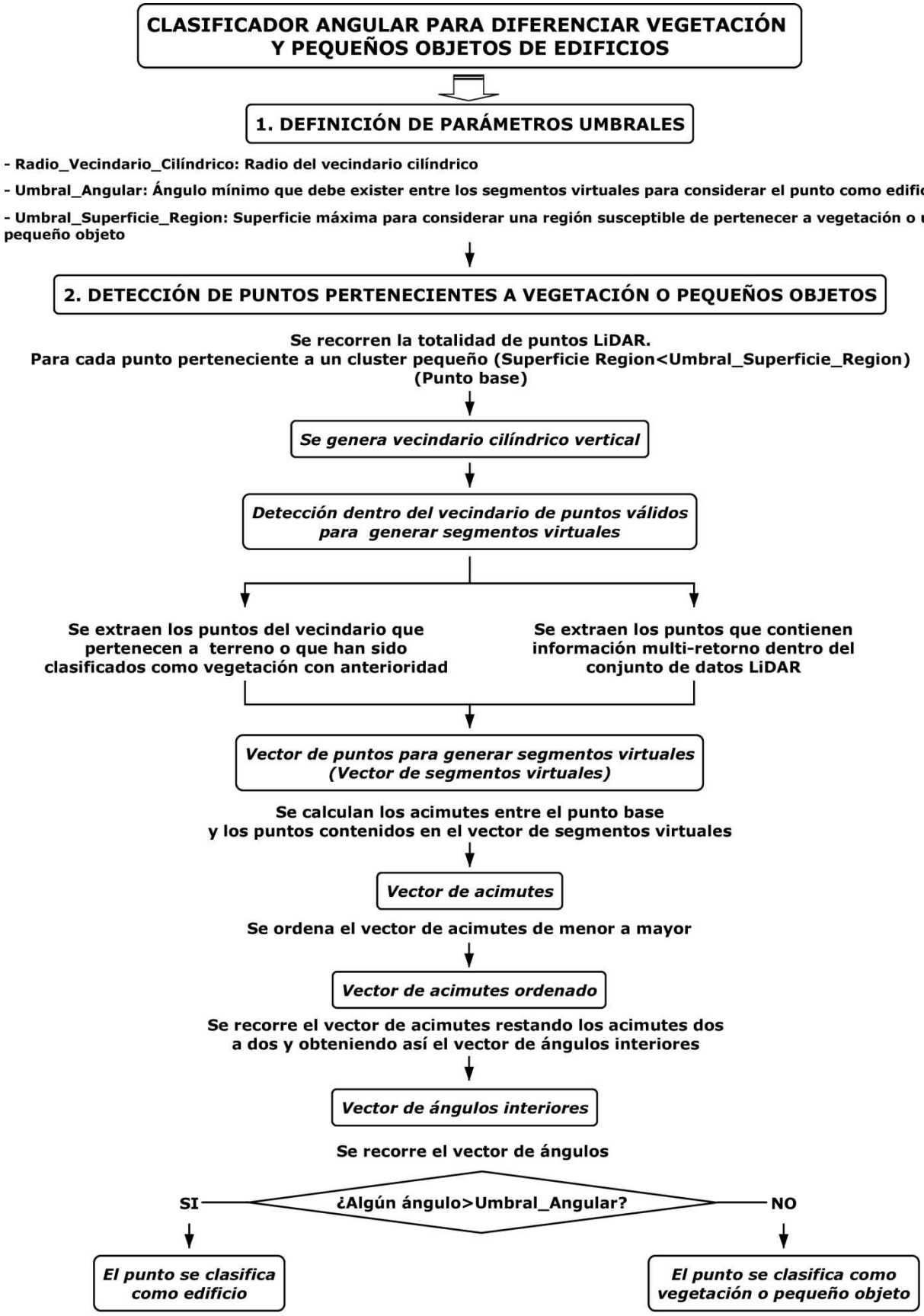


Figura 87. Diagrama de flujo del clasificador angular.

Tal y como se profundizará en posteriores apartados, tanto el análisis de texturas como el filtro angular, producen en mayor o menor medida errores de omisión en la clasificación de vegetación.

El clasificador angular necesita que los pulsos láser lleguen hasta el terreno para analizar su posición relativa a partir del cálculo de ángulos interiores asociados a los segmentos virtuales. No obstante, a medida que aumenta la densidad de la cobertura vegetal disminuye el número de haces láser que consiguen penetrar hasta el terreno. Como consecuencia, en zonas de vegetación muy densas es frecuente que existan áreas relativamente amplias donde ningún punto LiDAR pertenece al terreno. En estas zonas concretas, el clasificador angular no funciona correctamente y se producen errores por defecto en la clasificación. Además, el clasificador angular no detecta zonas de sombra y precisa que exista información tridimensional en todas las direcciones. Como consecuencia, falla en aquellos puntos base que están próximos al límite de la escena o cuando existen zonas de sombra asociadas a la absorción de los pulsos LiDAR por parte del agua.

La Figura 88 muestra los dos tipos de errores que se producen tras ejecutar el clasificador angular y el análisis de texturas. El ejemplo propuesto pertenece a dos zonas concretas de los datos test 2 (Collado de Villalba). Con respecto al clasificador angular, en el primero de ellos se aprecia como en áreas de densa vegetación los pulsos LiDAR no penetran hasta llegar al terreno y por tanto se producen errores de omisión importantes (Figura 88a). El segundo ejemplo se corresponde con una zona de vegetación situada junto a un canal donde no existen puntos LiDAR debido a la absorción de los pulsos láser por parte del agua. En este caso, el clasificador angular falla en los puntos próximos a las zonas de sombra produciendo numerosos errores por defecto (Figura 88b).

Estos dos tipos de errores no se dan al aplicar análisis de texturas (algoritmo A). No obstante, este algoritmo produce errores por omisión en los límites de las zonas de vegetación densa, donde la clase terreno actúa como una barrera e impide la distribución uniforme de regiones pequeñas dentro del conglomerado asociado al vecindario de cada punto base. En la figura puede apreciarse cómo el análisis de texturas consigue detectar las zonas interiores de áreas de vegetación densa (Figura 88.c) mejorando los resultados del clasificador angular. Además el análisis de texturas diseñado detecta la existencia de zonas de sombra y se adapta en consecuencia para obtener óptimos resultados (Figura 88d).

En cualquier caso, ambos algoritmos se complementan si se ejecutan de forma concatenada, ya que el análisis de texturas detecta el grueso de la vegetación incluida en la parte central de áreas con vegetación densa, y el clasificador angular clasifica el resto. Por tanto, se implementa un nuevo algoritmo que ejecuta el análisis de textura y el clasificador angular de forma concatenada.

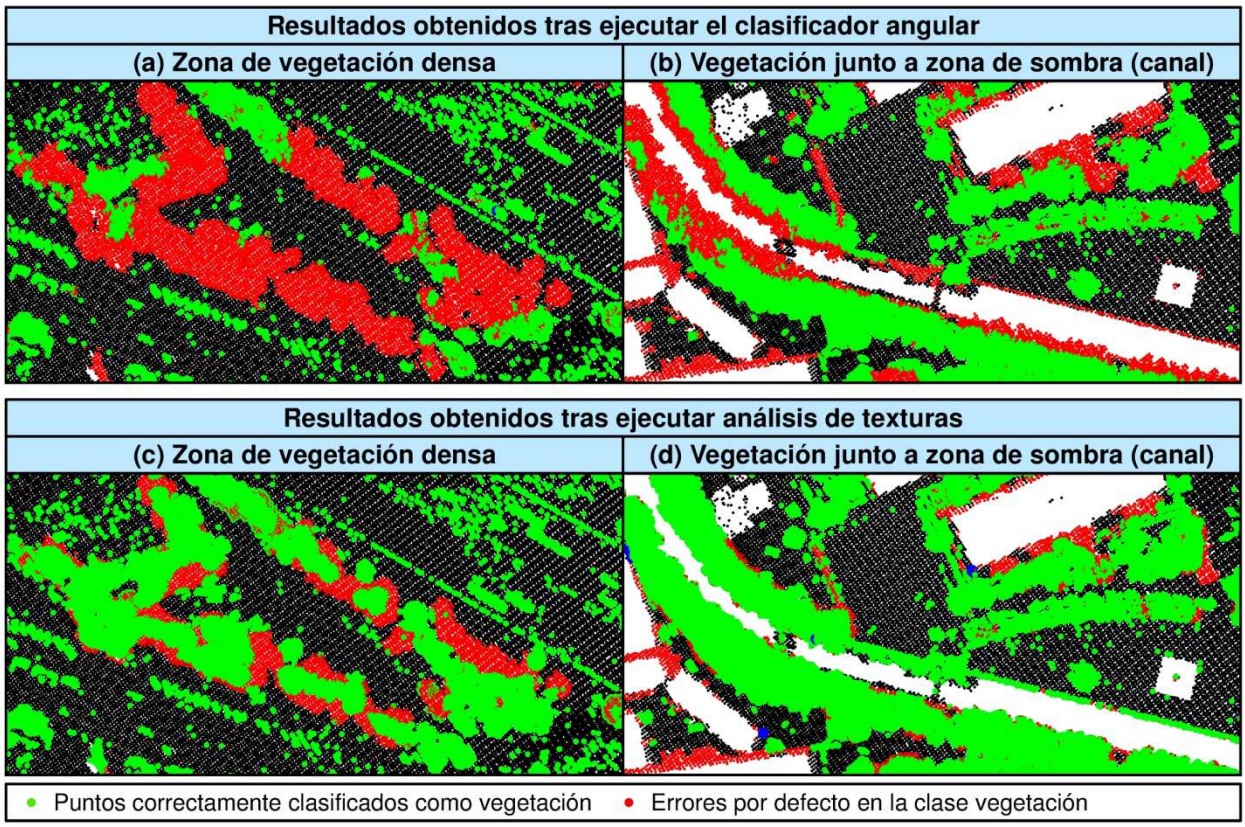


Figura 88. Errores de omisión en la vegetación. Clasificador angular y análisis de textura.

En primer lugar, la aplicación segmenta la escena en superficies continuas (Algoritmo auxiliar. Apartado III.10.1)). Seguidamente, detecta gran parte de la vegetación mediante análisis de texturas (algoritmo A). El resultado obtenido son los datos de entrada que el clasificador angular precisa para un correcto funcionamiento, la única diferencia es que a la hora de generar los segmentos virtuales que dan pie al cálculo de ángulos interiores no solamente serán válidos los puntos de tierra y los que contienen información multi-retorno, sino que además serán válidos aquellos puntos que han sido clasificados con anterioridad como vegetación. El resultado final será el obtenido tras ejecutar el clasificador angular. El diagrama de flujo que sigue el algoritmo es el siguiente (Figura 89).

Figura 89. Diagrama de flujo para detectar vegetación mediante análisis de texturas y clasificador angular (Algoritmo B)

III. 10. Algoritmos auxiliares

III.10.1. Segmentación total de la escena en superficies continuas

III.10.1.1. Motivación para el desarrollo del algoritmo y relación con los antecedentes

Un aspecto fundamental de los diferentes algoritmos programados consiste en segmentar la escena en superficies continuas delimitadas por cambios bruscos en pendiente. Con ello se consigue agrupar los datos LiDAR en conjuntos que pertenecen a un mismo objeto, lo que facilita el proceso de clasificación y reduce el tiempo de cómputo considerablemente.

En cuanto al proceso de clasificación será de gran utilidad el análisis de las características asociadas a las regiones obtenidas tras la segmentación, tales como el tamaño o la posición relativa existente entre ellas. Así por ejemplo, en entornos urbanos las agrupaciones de mayor tamaño pertenecen al entramado de calles y en consecuencia a la clase terreno. Por otro lado, el análisis de la posición relativa existente entre los segmentos será utilizado para diferenciar los puntos pertenecientes a edificios de aquellos que pertenecen a la vegetación u otros pequeños objetos. De forma análoga, para la detección de la clase terreno provisional, la metodología propuesta (algoritmo B) tiene en cuenta el hecho de que en entornos urbanos las agrupaciones pertenecientes al terreno son colindantes a agrupaciones que poseen una altitud mayor ya que están asociadas a edificios u otros objetos urbanos.

Además, de igual forma que han hecho otros autores como Ekhtari et al. (2008) y Pérez et al. (2012), la metodología propuesta para detectar el terreno combina un proceso de densificación con técnicas de segmentación consiguiendo así optimizar considerable el tiempo de cómputo.

La mayoría de los autores que aplican métodos de segmentación en datos LiDAR utilizan los resultados para identificar las diferentes superficies planas que conforman la escena con el fin de realizar modelos de edificaciones de forma automática. En este sentido, la segmentación de datos LiDAR en superficies planas puede realizarse mediante algoritmos de segmentación muy diversos tales como los métodos de ajuste (Lari et al. 2011) o métodos basados en agrupación por atributos (Vosselman y Dijkman 2001; Filin y Pfeifer 2006; Kim et al. 2007). No obstante, el objetivo de la metodología propuesta no consiste en la detección de superficies planas, si no en superficies continuas que pertenezcan a un mismo objeto. Es por ello que de entre la totalidad de métodos de segmentación existentes se opta por la segmentación mediante crecimiento de regiones. Este método fue propuesto por Besl y Jain (1988) y ha sido utilizado en multitud de ocasiones para segmentar datos LiDAR (Tovari y Pfeifer 2005, Rabbani et al. 2006, Pu y Vosselman 2006, Ekhtari et al. 2008 y Pérez et al. 2012).

La segmentación mediante crecimiento de regiones se inicia mediante la selección de un conjunto de semillas que marcan cada uno de los objetos que tienen que ser segmentados. Estas semillas serán el origen de cada región a la que se añadirán de forma iterativa nuevos puntos del vecindario que tengan atributos similares. De esta forma, las regiones aumentarán de tamaño hasta que los diferentes objetos hayan sido segmentados por completo, momento en el cual el proceso iterativo finaliza.

III.10.1.2. Descripción del algoritmo programado

El algoritmo desarrollado utiliza dos parámetros para segmentar los datos en las diferentes superficies continuas que conforman la escena: un parámetro umbral en distancia de búsqueda (Δd) y un parámetro umbral en desnivel (Δz). El parámetro umbral en distancia no es más que el radio del cilindro vertical utilizado para determinar el vecindario, mientras

que el parámetro en desnivel consiste en el Δz mínimo para determinar si dos puntos pertenecen a una misma superficie continua.

La aplicación implementada no requiere semillas explícitas y comienza seleccionando un punto cualquiera del conjunto de datos LiDAR como punto semilla del primer grupo a segmentar. Posteriormente selecciona los puntos LiDAR que pertenecen a su vecindario cilíndrico y comprueba si cumplen con el parámetro umbral en desnivel (Δz), de forma que los puntos que cumplen con dicho parámetro son añadidos al grupo y actúan como semilla en la iteración posterior. El proceso finaliza cuando ningún nuevo punto es añadido al grupo, momento en el cual la superficie continua ha quedado completamente definida. El proceso prosigue seleccionando un nuevo punto semilla entre el resto de datos LiDAR que todavía no han sido segmentados, iniciándose de nuevo el proceso de segmentación. Todo el proceso es iterativo y finaliza cuando la totalidad de puntos LiDAR han sido segmentados.

Tal y como se puede observar en la Figura 90, el proceso de segmentación comienza con la selección al azar de un punto semilla origen del primer grupo o región. En este caso, se trata de un punto perteneciente al plano de techo de un edificio residencial. Seguidamente el vecindario cilindro del punto semilla es detectado (Figura 90a) y aquellos puntos cuyo desnivel es menor que el parámetro umbral (Δz) son añadidos automáticamente a la región (Figura 90b). Este proceso es iterativo, de forma que en cada iteración actúan como semilla los puntos añadidos al grupo en la iteración anterior (Figura 90b, Figura 90c y Figura 90d). El proceso de formación del grupo o región finaliza cuando ningún nuevo punto es añadido al mismo, obteniendo como resultado una región constituida por la totalidad de los puntos pertenecientes al plano del techo segmentado (Figura 90e). Una vez finaliza la formación de una región, otro punto semilla es seleccionado de entre el conjunto de datos LiDAR restante y el proceso de segmentación se repite hasta que la totalidad de datos LiDAR han sido segmentados (Figura 90f).

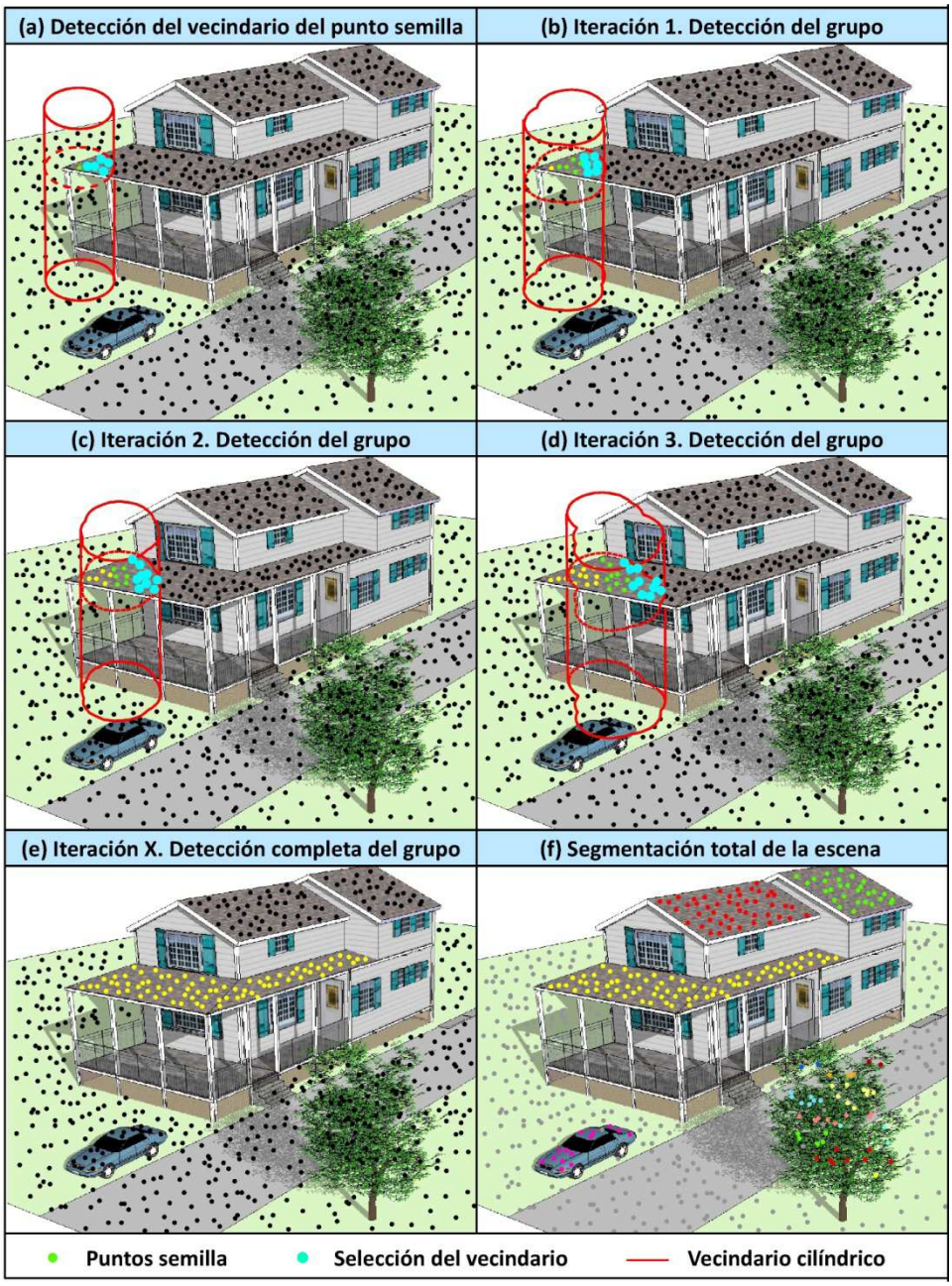


Figura 90. Proceso de segmentación.

Las diferentes fases de las que consta la aplicación implementada se detallan a continuación.

- **Carga de datos de entrada**

La aplicación implementada es una función que recibe los siguientes parámetros para su funcionamiento:

Tabla_XYZ: matriz de tres columnas con las coordenadas X,Y,Z de la nube de puntos. Para cada punto, el número de registro se corresponde con el identificador del punto LiDAR.

Tabla_Triangulos: se trata de una matriz de tres columnas donde cada fila se corresponde con un identificador de triángulo de la triangulación. En las diferentes columnas se definen

los tres vértices de cada triángulo que se corresponderán con identificadores de puntos LiDAR.

A_Code_Triangulos: arreglo bidimensional definido mediante un doble puntero, donde cada punto LiDAR tendrá un vector asociado con los identificadores de los triángulos de la triangulación de Delaunay en los que está contenido. Es de gran utilidad para determinar el vecindario de un punto.

- **Definición de vectores de marcadores y vectores dinámicos**

Se define un vector de marcadores de igual tamaño que el número total de puntos LiDAR (*V_M_Puntos*). Este vector se reinicia con el valor "-1" y a lo largo del proceso de segmentación se actualizarán sus registros mediante la impresión del identificador del grupo o segmento al que pertenece cada punto.

Se define además un vector dinámico que aumentará de tamaño en tiempo de ejecución (*V_D_Puntos*). Se trata de una lista enlazada donde se irán añadiendo los diferentes puntos detectados que pertenezcan a un determinado grupo o región.

- **Proceso iterativo de segmentación**

La segmentación de una región consiste en detectar la totalidad de puntos que pertenecen a una misma superficie continua, para ello se parte de un punto semilla que no debe de haber sido segmentado con anterioridad, por tanto, el inicio del proceso de segmentación consiste en seleccionar un punto origen del grupo marcado con el valor "-1" en el vector *V_M_Puntos*.

Una vez detectado el punto origen del grupo, se genera de forma automática un identificador de grupo y el punto semilla es marcado con dicho identificador en su registro correspondiente dentro del vector *V_M_Puntos*. Además, se añade a un vector dinámico que aumentará de tamaño a la vez que nuevos puntos son añadidos a la región y que denominaremos *V_D_Puntos*. Seguidamente se detecta el vecindario del punto. El algoritmo admite dos formas distintas de detectar el vecindario: por un lado permite la detección del vecindario directo de un punto, es decir, la selección de los puntos más próximos a uno dado utilizando únicamente las conexiones asociadas a las aristas de la triangulación; por otro, permite la detección del vecindario mediante la utilización de un cilindro vertical finito (Apartado II. 2). Por defecto el algoritmo detecta el vecindario directo a no ser que se especifique el radio del cilindro vertical (Δd), momento en el que la aplicación detectará el vecindario mediante el uso de un cilindro vertical no finito centrado en el punto origen del que queremos obtener su vecindario.

Seguidamente, el incremento de cota de cada uno de los puntos del vecindario obtenido con respecto al punto origen se calcula y aquellos cuyo desnivel es menor que el parámetro umbral (Δz) son añadidos a la región, momento en que son marcados en el vector *V_M_Puntos* con el identificador de región correspondiente y además son añadidos al vector dinámico *V_D_Puntos*. El proceso continúa tomando ahora como puntos origen del cálculo los nuevos puntos añadidos al grupo en la iteración anterior. Para ello se recorre el vector dinámico *V_D_Puntos*. Este vector aumentará de tamaño a la vez que es recorrido cuando nuevos puntos son añadidos al grupo, de tal forma que en el momento en que haya sido recorrido por completo la totalidad de la región habrá sido detectada.

Cuando una superficie u objeto ha sido segmentada, todos los registros del vector dinámico *V_D_Puntos* se eliminan para poder ser reutilizados en la segmentación del siguiente objeto.

El proceso es iterativo y un nuevo punto semilla se detecta de entre los datos LiDAR que todavía no hayan sido segmentados. De tal forma, que cuando no exista ningún punto LiDAR marcado con "-1" en el vector de marcadores *V_M_Puntos* significará que todos los puntos LiDAR han sido ya segmentados y por tanto el proceso de segmentación finaliza. La totalidad del proceso puede apreciarse en el siguiente diagrama de flujo (Figura 91):

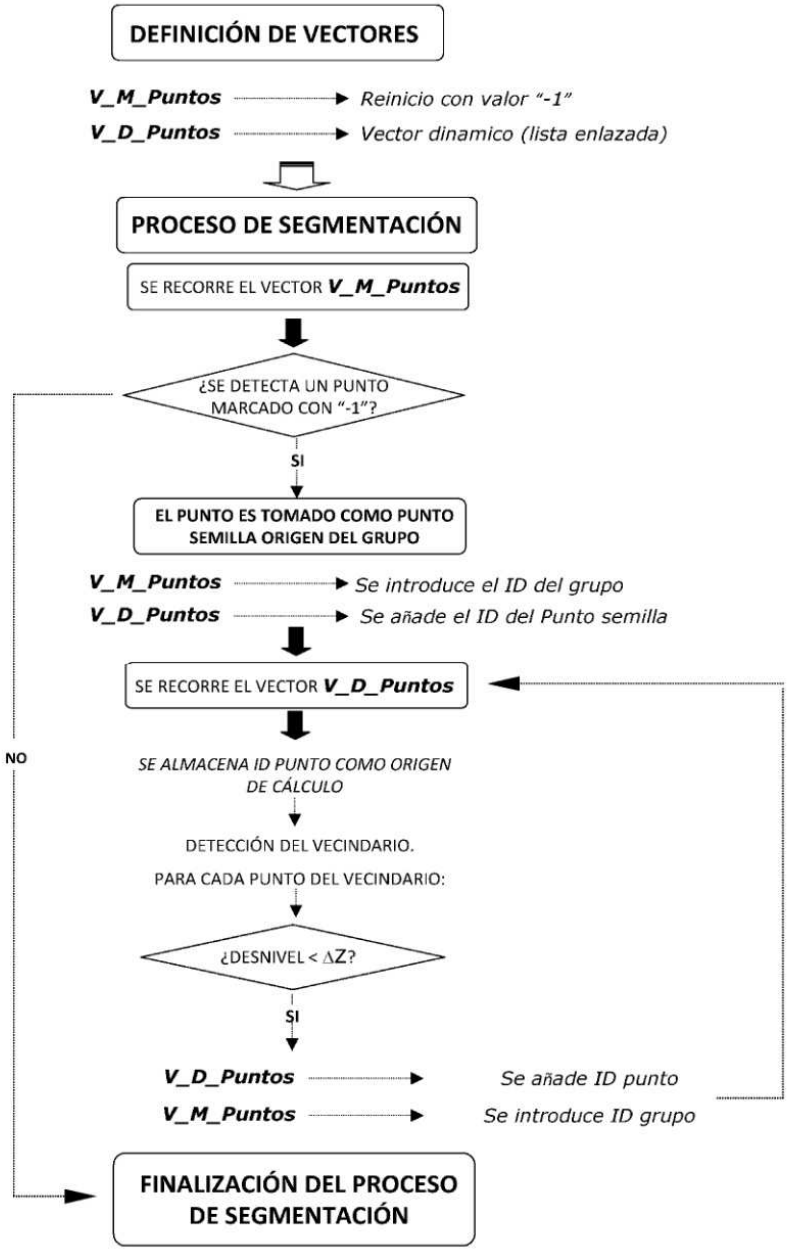


Figura 91. Diagrama de flujo del algoritmo de segmentación en superficies continuas.

- **Devolución de valores de retorno e impresión de archivos de salida**

La función devuelve a la aplicación principal las siguientes matrices y vectores:

V_M_Puntos: vector de igual dimensión que el número total de datos LiDAR. De forma que para cada punto LiDAR vendrá registrado el identificador de grupo o segmento al que pertenece.

M_Clases: matriz de dos columnas. En la primera columna aparece el identificador del grupo y en la segunda el identificador de un punto LiDAR que pertenece al mismo. Todos los puntos pertenecientes a un mismo grupo aparecerán registrados de forma consecutiva.

La aplicación implementada imprime los siguientes archivos:

Nombre del Proyecto.Clases.scr: script de Autocad donde los puntos se visualizarán con diferente color en función de la región a la que pertenecen. En la Figura 92 se observan la segmentación obtenida tras aplicar el algoritmo en los datos test 2 (Collado Villalba).

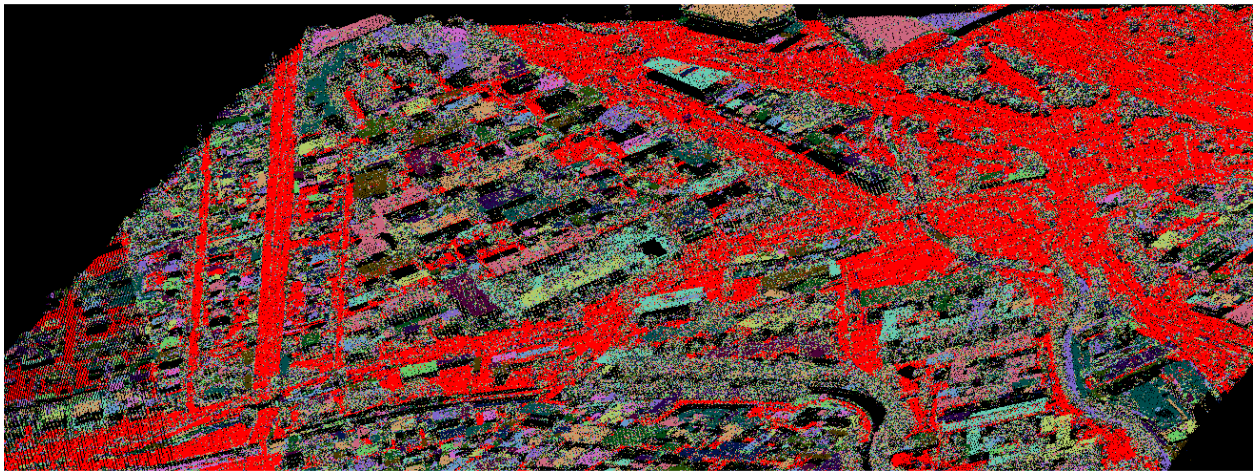


Figura 92. Segmentación de la escena en superficies continuas. Datos test 2.

III.10.2. Detección y extracción del contorno de las entidades

III.10.2.1. Motivación para el desarrollo del algoritmo y relación con los antecedentes

En el proceso de densificación necesario para detectar el terreno es necesario detectar y extraer el contorno de las distintas entidades que conforman la escena en forma poligonal. Además, los contornos deben de estar constituidos por polígonos cerrados cuyos vértices estén ordenados en el sentido de las agujas del reloj, siendo esta característica de vital importancia para poder establecer con posterioridad propiedades y relaciones topológicas.

En este sentido, se hace necesario desarrollar un algoritmo para extraer de forma automática los contornos de las distintas entidades que definen los puntos LiDAR. Para ello, tal y como se verá con mayor detalle en el siguiente apartado se ha implementado un nuevo algoritmo que aplica los principios básicos del algoritmo Weiler-Atherto (Weiler 1977). Este algoritmo es utilizado en visión por computador para recortar polígonos. No obstante, en el presente trabajo se adaptan sus bases para desarrollar un nuevo algoritmo que es capaz de detectar los bordes de los distintos objetos urbanos que conforman la escena y que están definidos por nubes de puntos, para posteriormente vectorizar dichos bordes obteniendo el contorno límite del objeto en cuestión en forma de polígono cerrado.

III.10.2.2. Descripción del algoritmo programado

El algoritmo desarrollado extrae el contorno de las diferentes entidades de la escena en forma poligonal. Cada contorno será extraído como un polígono cerrado cuyos vértices estarán siempre ordenados en el sentido de las agujas del reloj. De esta forma, el resultado podrá ser utilizado por otros algoritmos para determinar propiedades topológicas como inclusión o vecindad, así como para realizar operaciones propias de análisis SIG tales como el recorte de polígonos.

Básicamente, el algoritmo utiliza los bordes previamente detectados de las entidades así como el contorno límite de la zona de estudio para extraer en forma de cadena poligonal su límite.

De entre la totalidad de contornos que el algoritmo permite extraer, supongamos que se precisa extraer el contorno de un edificio. En este caso, al igual que en el resto de entidades, el contorno límite estará situado entre los puntos de bordes pertenecientes al edificio y los asociados a sus entidades adyacentes. De esta forma, tal y como se observa en la Figura 93, la línea de fachada del edificio estará siempre situada entre los puntos de borde del edificio y los puntos de borde del terreno.

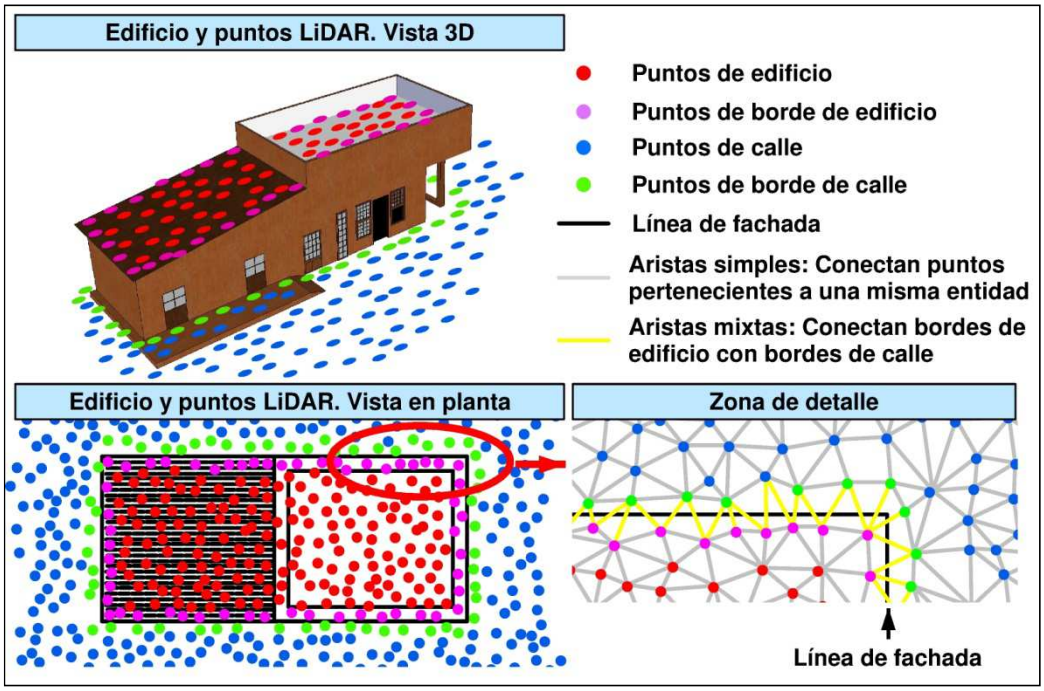


Figura 93. Posición de la fachada del edificio con respecto a los puntos LiDAR

En la imagen de detalle de la figura anterior (Figura 93), se aprecian las aristas de los diferentes triángulos que conforman la triangulación de Delaunay. Concretamente es posible diferenciar dos tipos de aristas: aquellas que conectan puntos LiDAR pertenecientes a una misma entidad (aristas simples) y las que conectan puntos de entidades diferentes (aristas mixtas). En este sentido, se puede observar cómo la línea de fachada no coincidirá con los puntos LiDAR detectados como bordes, si no que discurrirá siempre cortando las aristas mixtas de la triangulación. El enfoque propuesto para extraer el contorno se basa en esta premisa y supone que la línea de fachada siempre pasará entre los puntos de borde pertenecientes a ambas entidades, cortando en todos los casos las aristas mixtas. En concreto, el algoritmo calcula las coordenadas de los puntos medios de las aristas mixtas y los conecta mediante una cadena poligonal cerrada, registrando los vértices del polígono siempre en el sentido de las agujas del reloj (Figura 94).

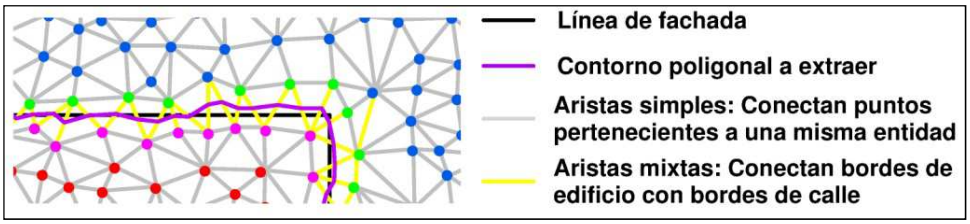


Figura 94. Extracción del contorno en forma de polígono

Esta metodología funciona correctamente en las entidades que están incluidas por completo en el interior de la zona de estudio. No obstante, aquellas entidades que no cumplen esta premisa y de las que solamente se disponen datos LiDAR de una zona parcial de las mismas, estarán cortadas por el contorno límite de la zona de estudio y en una de sus partes no existirán aristas mixtas. Esta ausencia de aristas mixtas en una parte del límite de la entidad tiene como consecuencia la no extracción de la totalidad del contorno, si no una parte del mismo, obteniendo en estos casos como resultado un polígono abierto.

Tal y como se observa en el ejemplo de la Figura 95, se dispone de un edificio rodeado por la clase calle. En este caso, si la nube de puntos engloba por completo al edificio (Figura 95a), el algoritmo detecta los bordes de las entidades adyacentes (edificio y calle) y calcula los puntos medios de las aristas de la triangulación que unen ambos bordes (aristas mixtas) extrayéndolos en forma de polígono cerrado. No obstante, si el edificio no está incluido por completo en la nube de puntos LiDAR (Figura 95b), estará cortado por el polígono límite que engloba la zona de estudio y en esta zona no existirán aristas mixtas, con lo que el algoritmo únicamente extraerá el contorno de forma parcial y no constituirá un polígono cerrado, sino un polígono abierto en la zona donde discurre el límite del área de estudio.

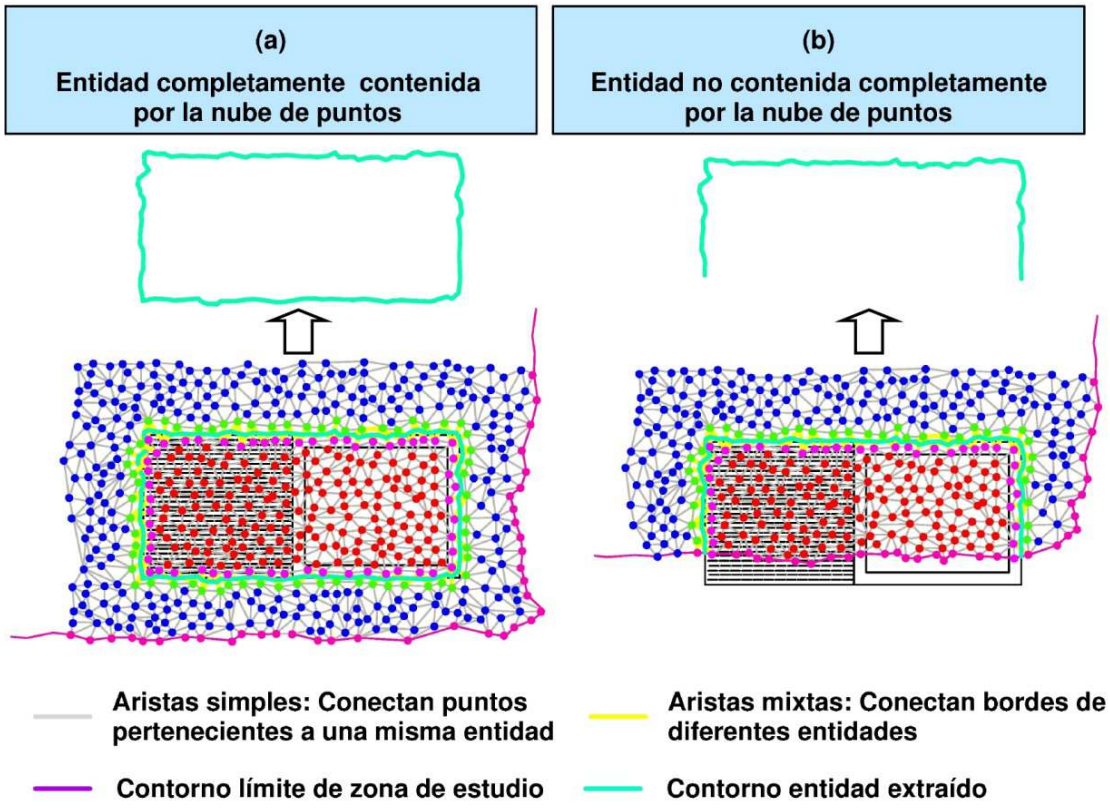


Figura 95. Problemas en la extracción de contornos de entidades no contenidas por completo en la nube de datos LiDAR

Para solucionar este problema el algoritmo desarrollado sigue una metodología similar a la del algoritmo de Weiler-Atherton utilizado para el recorte de polígonos en visión por computador y análisis SIG. Este algoritmo se basa en recorrer los vértices y aristas del polígono a recortar que sean interiores al polígono de recorte, de forma que si en algún momento del recorrido un vértice está fuera del polígono de recorte, se cambia el recorrido al polígono de recorte, para continuar con su lista de vértices y aristas. Este algoritmo requiere por tanto el cálculo previo de los puntos de intersección entre ambos polígonos, ya que serán los puntos entrantes y salientes para cambiar el recorrido de recorte y de extracción.

El proceso de extracción se muestra en el ejemplo de la Figura 96. Dados dos polígonos, un polígono de recorte y un polígono base que será recortado por el anterior. El algoritmo de Weiler-Atherton comienza con el cálculo de los puntos de intersección entre ambos polígonos (Figura 96a). Seguidamente, inicia la extracción del polígono resultante tomando como origen cualquier vértice del polígono base que esté contenido en el interior del polígono de recorte, extrayendo la totalidad de los vértices del polígono base hasta llegar al primer punto de intersección previamente calculado (Figura 96b). Llegar a un punto de intersección siempre tendrá como consecuencia cambiar el polígono que se está recorriendo. En este caso se está recorriendo el polígono base y por tanto comienza a recorrerse el polígono de recorte extrayendo sus vértices hasta llegar al segundo punto de intersección (Figura 96c). Análogamente, al llegar a este nuevo punto de intersección se deja de recorrer el polígono de recorte y se extraen la totalidad de los vértices del polígono base hasta llegar al tercer punto de intersección (Figura 96d), momento en el que pasa a recorrerse nuevamente el polígono de recorte hasta el último punto de intersección (Figura 96e). Finalmente se recorre el polígono base hasta llegar de nuevo al primer vértice del polígono que fue extraído (Figura 96f).

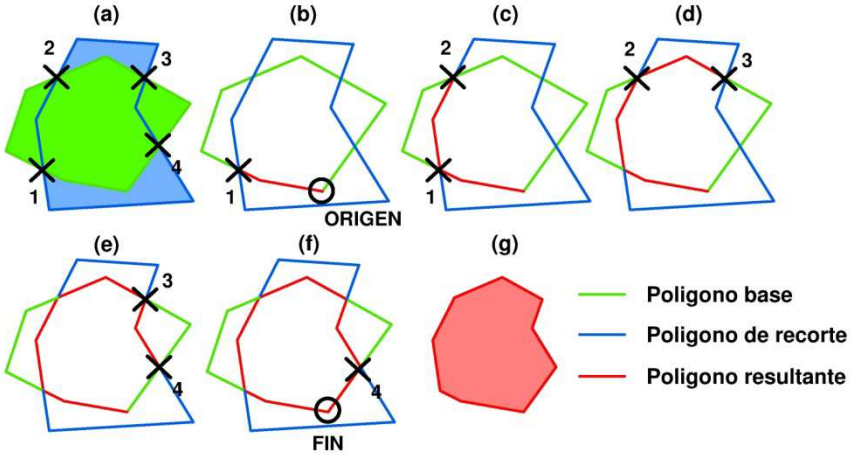


Figura 96. Algoritmo de Weiler-Atherton para el recorte de polígonos.

La aplicación implementada para la extracción de contornos sigue un procedimiento análogo al algoritmo de Weiler-Atherton cuando la entidad no está completamente incluida en la nube de puntos LiDAR y únicamente se dispone de datos de una parte de la misma. En estos casos el polígono límite de la zona de estudio detectado y extraído en el proceso de importación de datos (Apartado III.1.2.2) es utilizado como si se tratara del polígono de recorte.

Retomando el ejemplo de la Figura 95 donde el contorno de un edificio debe extraerse, el proceso de extracción del contorno comenzará en una arista mixta cualquiera de las asociadas al borde de edificio, detectando su punto medio como vértice origen de extracción del contorno (Figura 97a). Seguidamente, partiendo de este primer vértice se extraerán el

resto de puntos medios de las aristas mixtas asociadas al edificio en el sentido de las agujas del reloj hasta detectar una intersección con el polígono límite de la zona de estudio (Figura 97b). Llegado a este punto, ya no existirán más aristas mixtas que recorrer. Por tanto, siguiendo una metodología análoga al algoritmo de Weiler-Atherton se pasará a recorrer el contorno límite de la zona de estudio que actuará como si se tratara del polígono de recorte. El polígono límite de recorte se recorrerá extrayendo sus vértices hasta detectar cualquier conexión con una arista mixta, momento en el cual se habrá llegado a otra intersección y se dejará de recorrer el polígono límite de la zona de estudio para pasar a extraer de nuevo los puntos medios de las aristas mixtas (Figura 97c). Finalmente, el algoritmo extraerá los puntos medios de las aristas mixtas en el sentido de las agujas del reloj hasta llegar de nuevo al punto origen de extracción, momento en el que finalizará el proceso y se dispondrá del contorno del edificio extraído en forma de polígono cerrado (Figura 97c)

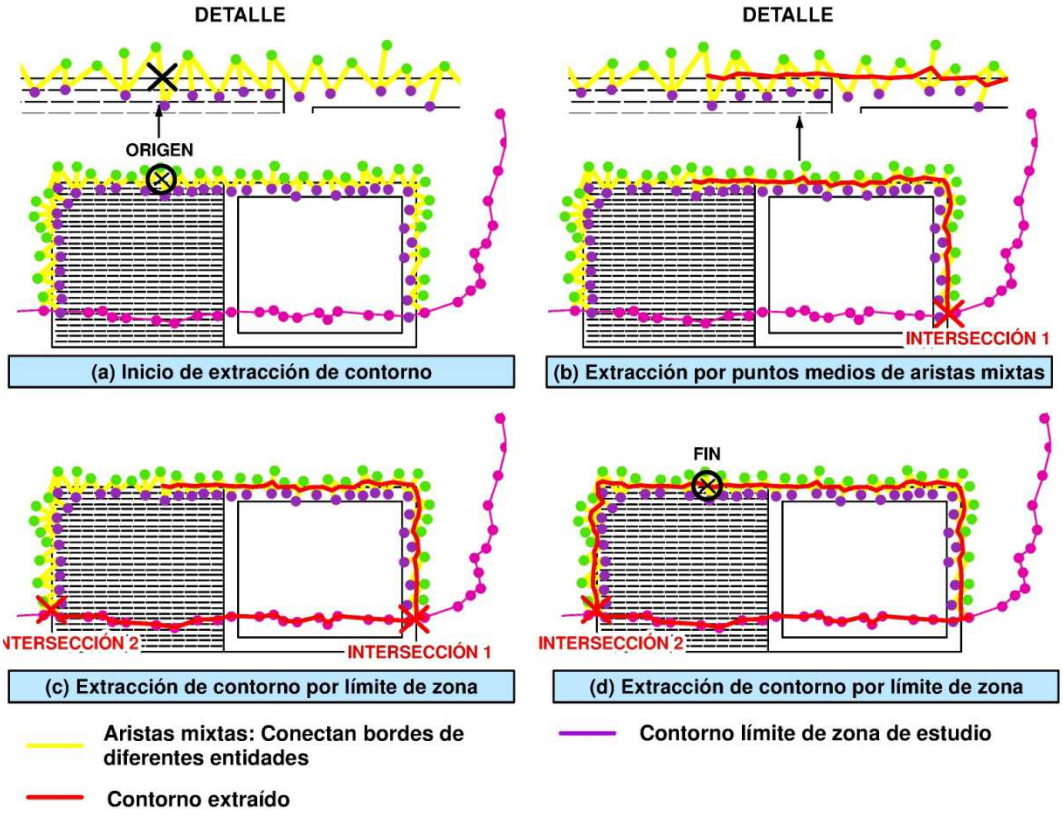


Figura 97. Extracción de contorno de entidades mediante metodología basada en el algoritmo de Weiler-Atherton.

Supongamos que desea extraerse el contorno de la totalidad de objetos que pertenecen a una misma entidad A, si llamamos entidad B al resto de entidades cuyo contorno no desea ser extraído el algoritmo ejecutará las fases que aparecen el siguiente diagrama de flujo (Figura 98).

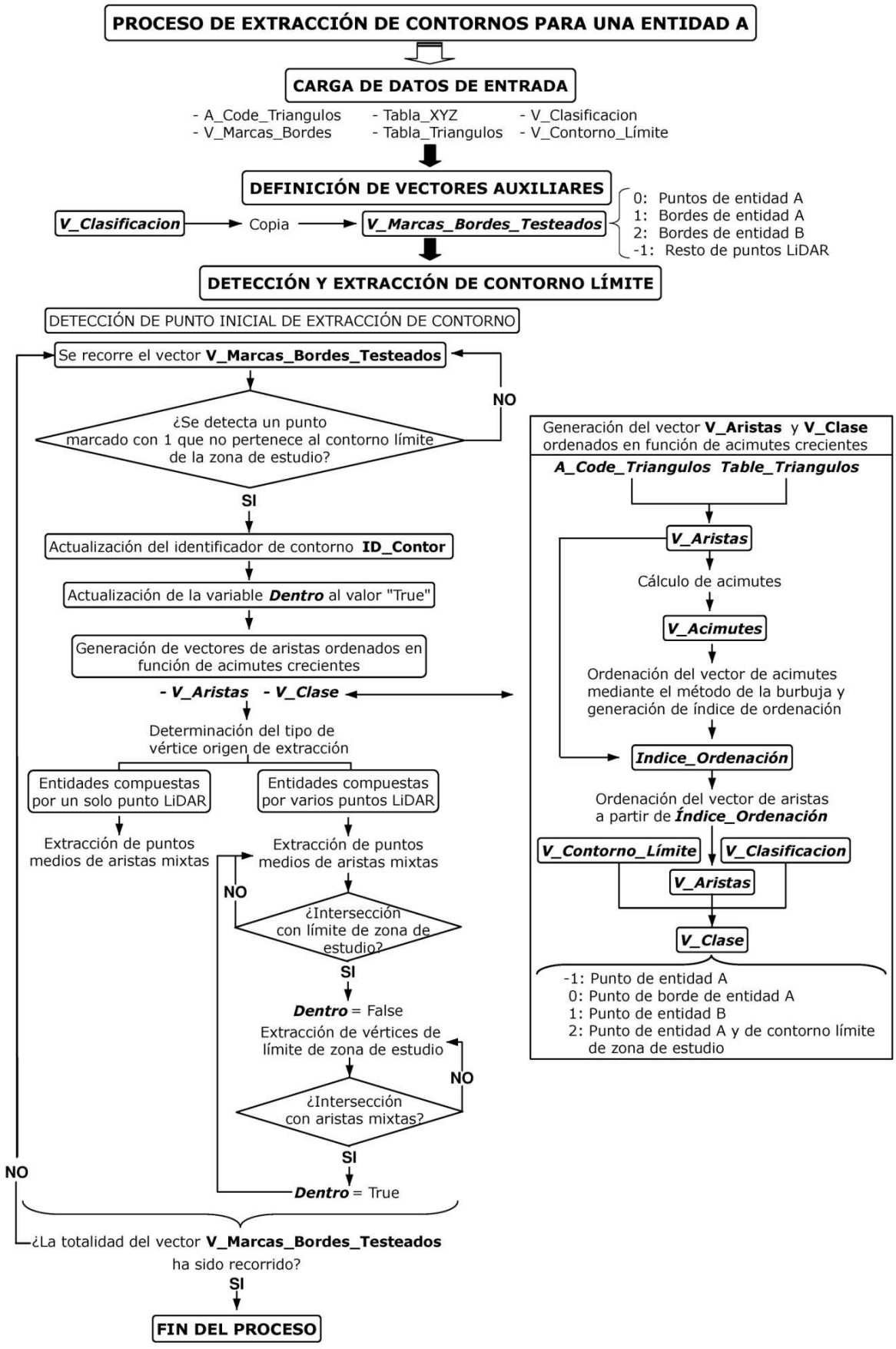


Figura 98. Diagrama de flujo del proceso de extracción de contornos

Las distintas fases que ejecuta el algoritmo se detallan a continuación:

- **Carga de datos de entrada**

La aplicación implementada es una función que recibe los siguientes parámetros para su funcionamiento:

Tabla_XYZ: matriz de tres columnas con las coordenadas X,Y,Z de la nube de puntos. Para cada punto, el número de registro se corresponde con el identificador del punto LiDAR.

Tabla_Triangulos: se trata de una matriz de tres columnas donde cada fila se corresponde con un identificador del triángulo de la triangulación. En las diferentes columnas se definen los tres vértices de cada triángulo, que se corresponderán con identificadores de puntos LiDAR.

A_Code_Triangulos: arreglo bidimensional definido mediante un doble puntero, donde cada punto LiDAR tendrá un vector asociado con los identificadores de los triángulos de la triangulación de Delaunay en los que está contenido. Es de gran utilidad para determinar el vecindario de un punto.

V_Marcas_Bordes: vector de igual dimensión que el número total de puntos LiDAR donde vendrán marcados los bordes de las diferentes entidades cuyo contorno debe de ser detectado. El algoritmo extrae el contorno para dos entidades distintas que serán adyacentes (entidad A y entidad B). En este vector estarán marcados los puntos que pertenecen a la entidad A con "0", los que pertenecen al borde o límite de dicha entidad con el valor "1", los puntos pertenecientes a bordes de la entidad B con el valor "2" y finalmente, el resto de puntos con "-1".

V_Clasificacion: vector de igual dimensión que el número total de puntos LiDAR donde vendrán marcados las puntos pertenecientes a cada una de las entidades a detectar. Los puntos pertenecientes a la entidad A irán marcados con el valor "3" y los puntos pertenecientes a la entidad B marcados con el valor "4".

V_Contorno_Límite: vector donde vendrán los vértices que definen el contorno límite de la zona de estudio. Los vértices se corresponderán con puntos LiDAR de la nube de datos tridimensional y estarán ordenados en el sentido de las agujas del reloj. Además, se trata de un polígono cerrado y por tanto el primer y último vértice serán el mismo.

V_Id_Bordes_Entidad_A: vector de igual dimensión que el número total de puntos LiDAR existentes en la zona de estudio. El algoritmo marcará cada punto de borde de la entidad A con el identificador del polígono de contorno que lleve asociado. El vector es recibido por la aplicación reiniciado con el valor "-1" en la totalidad de sus registros.

- **Generación de vectores auxiliares**

Se generan dos vectores de marcadores que serán utilizados en el proceso iterativo de detección y extracción de contornos:

V_Marcas_Bordes_Testeados: vector de igual dimensión que el número total de puntos LiDAR. Este vector será una copia del vector ***V_Marcas_Bordes***, donde estarán marcados los puntos pertenecientes a la entidad A con "0", los que pertenecen al borde o límite de dicha entidad con el valor "1", los puntos pertenecientes a bordes de la entidad B con el valor "2" y el resto de puntos con "-1". Este vector servirá para marcar en tiempo de ejecución los puntos de borde que ya han sido procesados en el procedimiento de la extracción de contornos.

- **Proceso de detección y extracción de contornos**

Los diferentes procesos que realiza el algoritmo para extraer los contornos de la totalidad de entidades son los siguientes:

a) Detección de posible punto válido para la extracción de contorno.

El proceso de extracción de contornos comienza detectando un punto válido que será tomado como punto origen de extracción. Para ello se recorrerá el vector ***V_Marcas_Bordes_Testeados*** hasta encontrar un punto LiDAR marcado con el valor "1", lo que significará que dicho punto pertenece al borde de la entidad A. El identificador del punto origen es almacenado y el punto es marcado en el vector ***V_Marcas_Bordes_Testeados*** con el valor "-2". Ésto evitará seleccionar con posterioridad de nuevo el punto para realizar la extracción de contornos, evitando así obtener contornos duplicados en el resultado.

b) Generación de identificador de contorno.

El contorno de cada objeto urbano extraído llevará asociado un identificador propio, en este momento y antes de comenzar con la extracción se genera el identificador de contorno (*ID_Contor*).

c) Reinicio de variable asociada al procedimiento análogo al algoritmo Weiler-Atherton.

Para poder utilizar la metodología propuesta que sigue las bases del algoritmo Weiler-Atherton, la aplicación deberá saber en cada momento si se está recorriendo el límite de la entidad en el interior de la nube de puntos o bien si se está recorriendo el polígono límite de la zona de estudio (caso de entidades incompletas). Para ello se define una variable tipo *booleana* que denominaremos *Dentro*. Este tipo de variable admite dos valores: True o False. Siempre que el algoritmo esté recorriendo aristas mixtas en el interior de la zona de estudio la variable *Dentro* tendrá el valor True; por el contrario cuando se esté recorriendo el contorno límite del conjunto de datos, su valor será False. Dado que el algoritmo comienza a extraer el contorno de las entidades siempre a partir de una arista mixta, la variable se reinicia en este momento con el valor True.

d) Detección de vértices conectados al punto origen de extracción y ordenación en función de acimutes crecientes.

Llegado a este punto, se trata de detectar las aristas mixtas asociadas al borde para comenzar la extracción mediante el cálculo de sus puntos medios. Para ello se crea un vector de aristas asociado al vértice origen de extracción (*V_Aristas*). Este vector contendrá los identificadores de puntos LiDAR que constituyen los vértices adyacentes de las aristas de la triangulación en las que está contenido el punto origen de cálculo. Para ello se utiliza el arreglo de codificación en triángulos (*A_Code_Triangulos*), con lo que se consigue determinar automáticamente los puntos LiDAR que están conectados con el punto origen de forma directa y sin realizar cálculos ni búsquedas adicionales.

En la Figura 99 se puede observar un ejemplo del proceso para generar el vector de aristas. Dado un punto origen de extracción de contorno seleccionado de forma automática, el algoritmo determina las aristas de la triangulación de Delaunay que contienen a dicho punto como vértice. Seguidamente los identificadores de puntos LiDAR que constituyen los vértices adyacentes a dichas aristas son almacenados en el vector de aristas (*V_Aristas*).

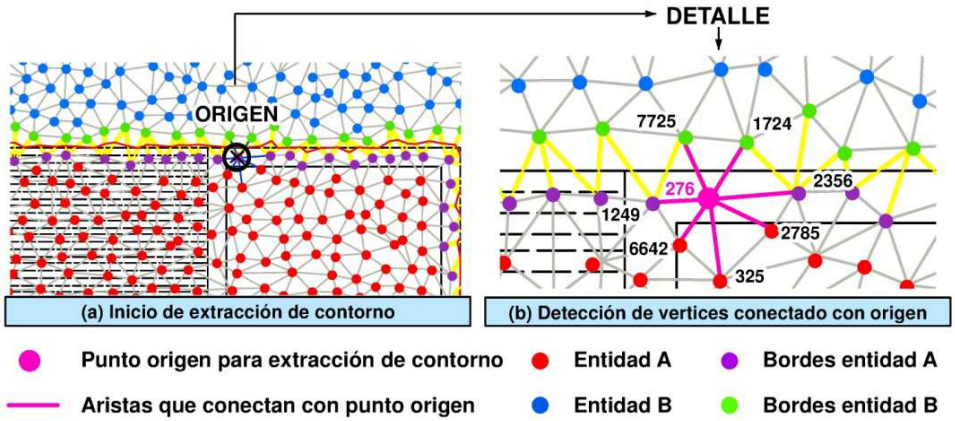
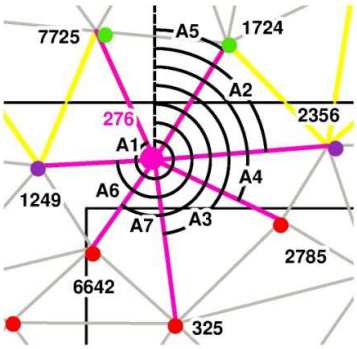


Figura 99. Generación automática del vector de aristas.

Como se ha comentado con anterioridad resulta necesario extraer los vértices del contorno en el sentido de las agujas del reloj. En este sentido, detectar la dirección correcta de extracción en el punto origen es esencial para un correcto funcionamiento del proceso. Para ello, el primer paso consiste en calcular los acimutes desde el punto origen a la totalidad de vértices contenidos en el vector de aristas, con el fin de ordenar posteriormente el vector de aristas en función de la magnitud de los acimutes calculados.

Siguiendo con el mismo ejemplo, en la siguiente Figura 100 se muestra el proceso que realiza el algoritmo para generar el vector de acimutes. Primero toma como punto base para el cálculo de acimutes el punto origen de extracción y posteriormente recorre el vector de aristas (*V_Aristas*) almacenando los identificadores de los puntos LiDAR que contiene y calculando el acimut asociado a cada uno de ellos. Finalmente los acimutes son almacenados en el vector de acimutes en el mismo orden en que fueron calculados (*V_Acimutes*).



V_Aristas	Cálculo de acimutes	V_Acimutes
7725	A1= ACIMUT (276 - 7725) =372,89 g	372,89
2356	A2= ACIMUT (276 - 2356) = 94,84 g	94,84
325	A3= ACIMUT (276 - 325) =191,68 g	191,68
2785	A4= ACIMUT (276 - 2785) = 128,37 g	128,37
1724	A5= ACIMUT (276 - 1724) = 35,40 g	35,40
1249	A6= ACIMUT (276 - 1249) =296,66 g	296,66
6642	A7= ACIMUT (276 - 6642) =239,26 g	239,26

Figura 100. Cálculo del vector de acimutes.

Una vez que el vector de acimutes (*V_Acimutes*) ha sido generado, el algoritmo ordena el vector de acimutes en función de la magnitud de los mismos (de menor a mayor) y posteriormente genera un índice de ordenación. Este índice es utilizado para ordenar de la misma forma el vector de aristas (*V_Aristas*), consiguiendo así tener los vértices ordenados dentro del vector de aristas en función de la magnitud de sus acimutes.

Todo ello puede observarse en la Figura 101, tras calcular el vector de acimutes se ordenan sus valores de menor a mayor, para ello se utiliza el método de la burbuja. Este algoritmo de ordenación consiste en comparar pares de elementos adyacentes e intercambiarlos en caso de estar desordenados. Además, el algoritmo genera un índice de ordenación a la vez que ordena el vector. Este índice de ordenación es un vector de igual dimensión que el vector de aristas donde cada registro indica la posición que ocupaba cada elemento del vector de acimutes ordenados (*V_Acimutes_Ordenados*) en el vector original previo a la ordenación (*V_Acimutes*). Finalmente el algoritmo utiliza el índice de ordenación para ordenar el vector de aristas en función de los acimutes crecientes.

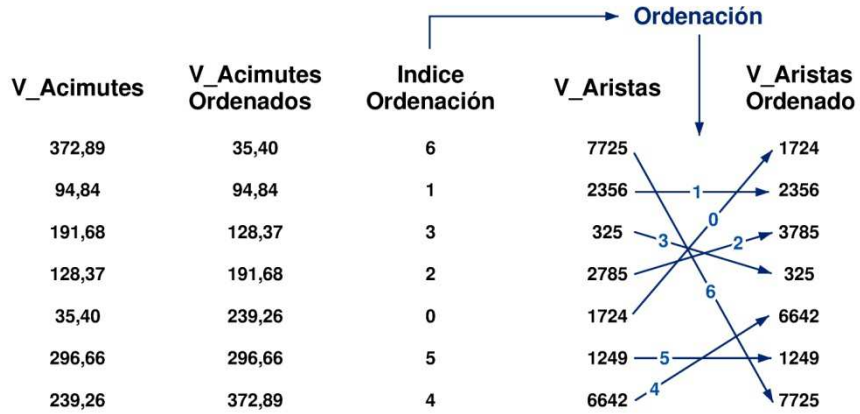


Figura 101. Ordenación del vector de aristas en función de los acimutes crecientes.

e) Clasificación de los elementos del vector de aristas en función de la entidad a la que pertenece cada vértice.

En este momento del proceso se dispone del vector aristas ordenados en función de la magnitud de sus acimutes calculados con respecto al punto origen de extracción. El vector viene constituido por los identificadores de los puntos LiDAR asociados a los vértices extremos de cada una de las aristas. No obstante, dado que la extracción se realiza extrayendo los puntos medios de las aristas mixtas, es necesario clasificar los vértices que constituyen el vector de aristas en función de la entidad a la que pertenecen. La clasificación de cada vértice se almacenará en un vector de igual dimensión que la matriz de aristas denominado *V_Clase*.

El procedimiento consiste en recorrer el vector de aristas y para cada identificador de punto LiDAR contenido en el mismo comprobar en el vector *V_Marcas_Bordes* la entidad a la que pertenece dicho punto. De esta forma el vector *V_Clase* se rellena con el siguiente código numérico:

Valor "-1": En caso de pertenecer a la entidad A.
Valor "0": En caso de pertenecer a un borde de la entidad A.
Valor "1": En caso de pertenecer a la entidad B.
Valor "2": En caso de pertenecer a la entidad A y además ser un punto de contorno límite.

Para el ejemplo propuesto, el vector que determina la clase de cada uno de los puntos del vector de aristas es el que se muestra en la siguiente imagen (Figura 102).

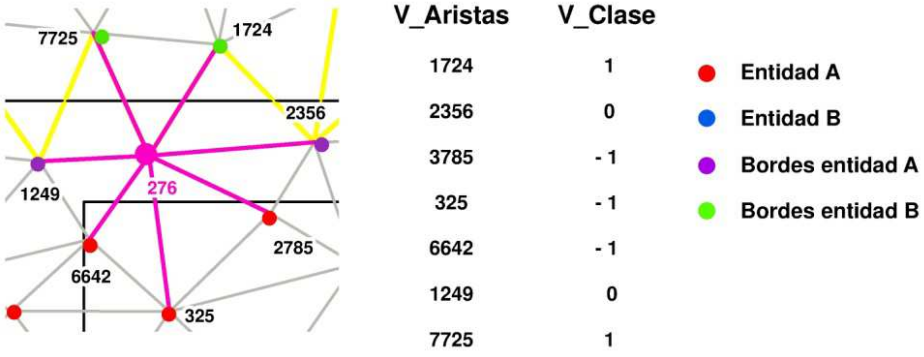


Figura 102. Generación del vector de clasificación de aristas (V_Clase).

f) Determinación del tipo de vértice origen de extracción de contorno.

A la hora de comenzar a extraer el contorno, el algoritmo debe de tener en cuenta las distintas posibilidades en cuanto a los tipos de puntos contenidos en el vector de aristas (*V_Aristas*), ya que en función de ello se procederá de una forma u otra en el procedimiento de extracción. Para ello, el algoritmo define una variable denominada *Tipo_Vertice_Origen* y posteriormente recorre el vector *V_Clase* comprobando a que entidades pertenece cada uno de los puntos LiDAR registrados y clasificando el punto origen teniendo en cuenta los casos que se detallan a continuación.

Existen pequeñas entidades tales como farolas o semáforos que están definidas en la nube de puntos LiDAR por una escasa cantidad de puntos. De hecho, en ocasiones están constituidas por un único punto LiDAR. En estos casos la totalidad de aristas que están conectadas con el punto origen son aristas mixtas, y por tanto, al recorrer el vector *V_Clase* no existirán puntos de borde, si no que todos los puntos pertenecerán a una entidad distinta a la del punto origen. En este caso, la variable *Tipo_Vertice_Origen* tomará el valor "0".

Cuando la entidad esté definida por varios puntos LiDAR y el punto origen de extracción no pertenezca al contorno límite de la zona de estudio, la variable ***Dentro*** tomará el valor "True" y comenzará el proceso de extracción con total normalidad. La variable *Tipo_Vertice_Origen* toma el valor "1" en estos casos.

En el caso de que existan puntos pertenecientes al contorno límite de la zona de estudio en el vector *V_Clase* estaremos en un vértice donde la variable *Dentro* cambia su valor. Estos casos se darán en contadas ocasiones y puede producir errores en el proceso de identificar el sentido de avance en la extracción de contornos. Por tanto, nunca se seleccionará un punto origen de extracción con estas características, sino que será descartado y se buscará un punto origen alternativo en el vector *V_Marcas_Bordes_Testeados*. La variable *Tipo_Vertice_Origen* tomará el valor "2" en estos casos.

g) Extracción del contorno de la entidad.

En función del tipo de vértice origen el algoritmo utlizará un procedimiento de extracción u otro. A continuación se detallan los diferentes casos:

CASO1: entidades constituidas por un único punto LiDAR (Tipo_Vertice_Origen = "0")

Se tratará de pequeñas entidades como farolas, semáforos, etc. En estos casos todas las aristas definidas en el vector *V_Aristas* serán aristas mixtas y bastará con extraer los puntos medios de dichas aristas en el sentido de las agujas del reloj. Para ello, dado que en el vector de aristas los vértices ya están ordenados en el sentido de las agujas del reloj,

únicamente deberá recorrerse dicho vector calculando las coordenadas del punto medio entre los puntos LiDAR que contiene y el punto origen de extracción.

El contorno se extraerá en un arreglo constituido por la clase *Vector* disponible en C++. Este arreglo se denominará *M_Coord_Contor* y se tratará de una matriz de tres columnas: en la primera columna vendrá el identificador de contorno, y en las dos siguientes las coordenadas de cada vértice del polígono, registrando siempre los vértices ordenados en el sentido de las agujas del reloj. Además el primer vértice extraído y el último siempre serán el mismo, ya que se trata de polígonos cerrados.

La Figura 103 muestra el proceso de extracción de contornos para estos casos. Dado un punto origen de extracción cuyas aristas de la triangulación están todas conectadas con puntos pertenecientes a otra entidad (entidad B), tras ordenar la matriz de aristas en función de acimutes crecientes se calculan las coordenadas medias de las aristas recorriendo la matriz *V_Aristas* y las coordenadas se registran en la matriz *M_Coord_Contor*. Esta matriz tendrá tres columnas, la primera con el identificador de contorno (335 en este caso) y las dos restantes con las coordenadas de los puntos medios en las aristas mixtas. Dado que existen ocho aristas mixtas el polígono que define el contorno estará constituido por nueve vértices, siendo idéntico el primero y el último al tratarse de un polígono cerrado.

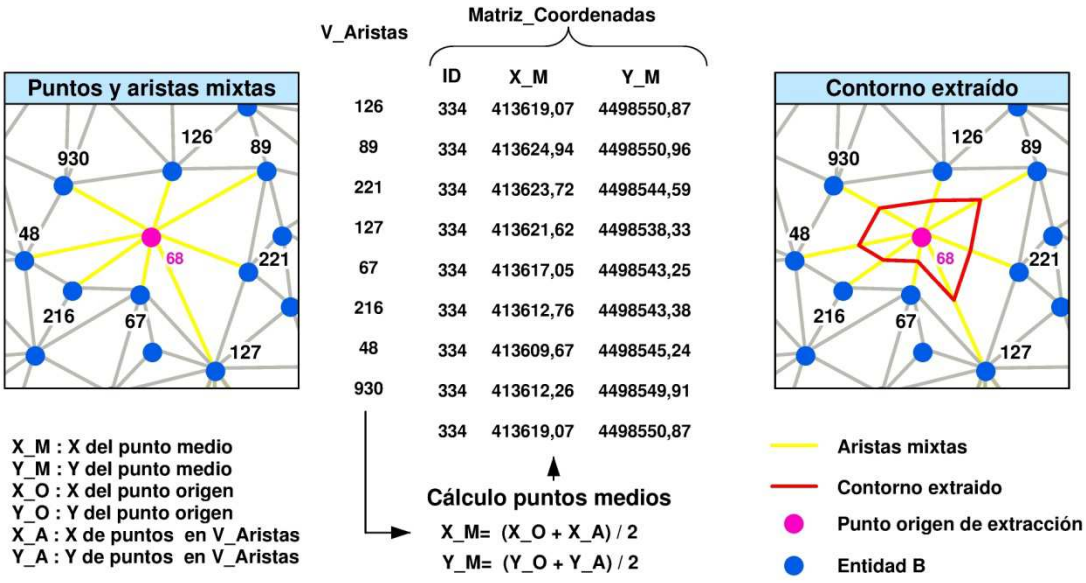


Figura 103. Extracción de contornos de entidades definidas por un único punto LiDAR.

CASO2: entidades constituidas por varios puntos LiDAR donde el punto origen de extracción no pertenece al contorno límite de la zona de estudio (Tipo_Vertice_Origen = "1")

La extracción del contorno se inicia mediante el cálculo de los puntos medios de las aristas mixtas. Por tanto, la variable *Dentro* siempre tendrá el valor "True" en el inicio del proceso.

La extracción comienza mediante el cálculo de la parte del contorno asociado al punto origen. Para ello debe detectarse una arista mixta a partir de la cual se iniciará el proceso. En este sentido, el algoritmo recorre el vector de aristas ordenado en función de acimutes crecientes hasta detectar un vértice adyacente que pertenezca a un borde de la entidad B. Esta será la arista mixta que dará comienzo a la extracción del polígono, siendo su punto medio el primer vértice del polígono a determinar.

Seguidamente se extraerán los puntos medios de las aristas mixtas que estén definidas en el vector de aristas de forma consecutiva a la arista mixta inicial. No obstante, llegará un momento en el que el algoritmo deje de recorrer aristas mixtas y detecte una arista de borde que conecta el punto origen con un punto de borde de la entidad A. En este momento el punto origen de extracción pasa a ser el punto de borde detectado y el proceso se itera tomando la arista inicial de extracción la última arista mixta detectada.

Tal y como muestra el ejemplo de la Figura 104, una vez detectado un punto origen de extracción de contorno (Punto 91) se generan sus vectores asociados (*V_aristas* y *V_Clase*) siempre ordenados en función de la magnitud de sus acimutes. Seguidamente se recorre la matriz *V_Clase* desde su primer elemento hasta detectar una arista mixta. En este caso, el primer registro del vector tiene el valor "1", lo que significa que su vértice asociado (punto 326) pertenece a la entidad B y por tanto la arista que conecta al punto origen (entidad A) y dicho vértice es una arista mixta. En este sentido, siempre que aparezca el valor "1" en el vector *V_Clase* significará que el vértice asociado del vector *V_Aristas* pertenece a una arista mixta. En el ejemplo propuesto, los dos primeros registros pertenecen a aristas mixtas y las coordenadas de sus puntos medios se extraerán de forma consecutiva.

Tras extraer los puntos medios de las dos aristas mixtas el algoritmo detectará que el tercer registro del vector *V_Clase* no pertenece a una arista mixta, ya que tiene el valor "0" y por tanto estará asociado a una arista de bordes que conectará el punto origen (entidad A) con otro punto de borde de la entidad A (Punto 228). En este momento el proceso de extracción de contorno desde el punto origen (Punto 91) finaliza y continua siguiendo el mismo procedimiento, pero tomando como nuevo punto origen el punto de borde que ha sido detectado (Punto 228).

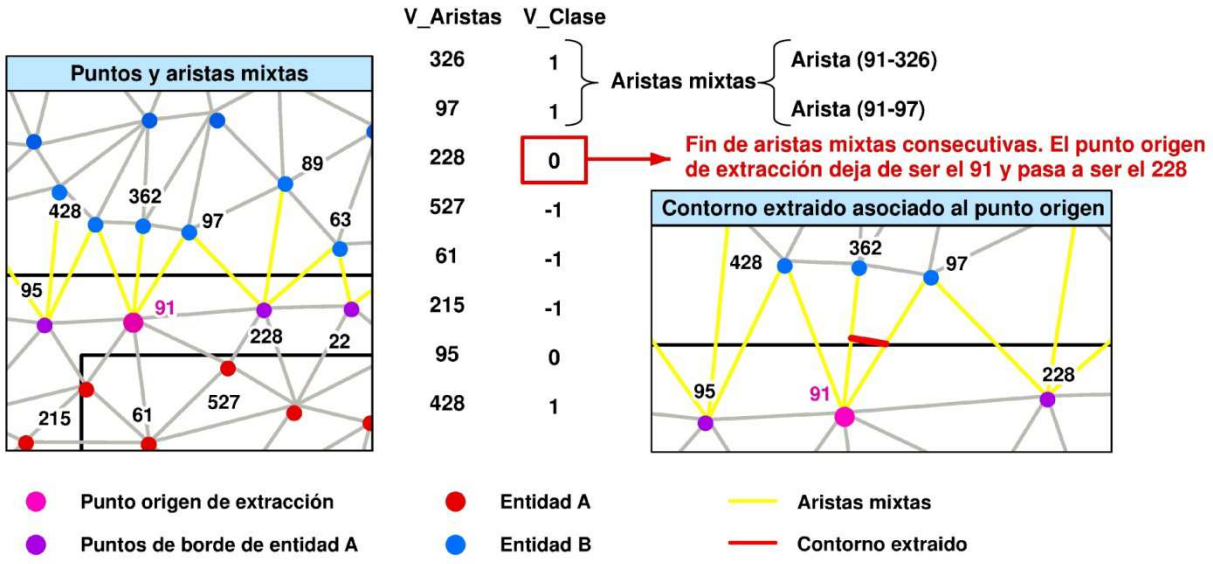


Figura 104. Extracción de contorno de entidades definidas por varios puntos LiDAR. Contorno asociado al punto origen.

El proceso se repite, pero esta vez tomando como punto origen de extracción el punto de borde detectado (Punto 228). Además el identificador del punto origen anterior (Punto 91) es almacenado dentro de la variable *Pto_Origen_Ant*.

El procedimiento será análogo al realizado con anterioridad y para el nuevo punto origen (Punto 228) se generará el vector de aristas (*V_Aristas*) y el vector donde vienen clasificados los vértices de las mismas (*V_Clase*), ambos ordenados en función de la magnitud de los acimutes asociados.

Todo ello viene detallado en la Figura 105. Una vez definidos los vectores se localizará el *Pto_Origen_Ant* (Punto 91) y se extraerán los puntos medios de las aristas mixtas que estén dispuestos de forma consecutiva a dicho punto dentro del vector de aristas. El vector de aristas hay que entenderlo como una lista circular, es decir una lista donde el último nodo apunta al primero. De esta forma y según el ejemplo, la primera arista mixta cuyo punto medio debe de extraerse es la que está definida en el registro posterior al Punto 91 (arista 228-97), dado que este es el último elemento del vector de aristas. Comenzará a recorrerse el vector desde el principio extrayendo las puntos medios de las aristas mixtas que se detecten de forma consecutiva (arista 228-106 y arista 228-48). El proceso de extracción finalizará en el Punto 113 marcado con "0" en el vector *V_Clase*. Este punto ya no define una arista mixta si no una arista de borde, por tanto el proceso de extracción desde el punto origen 228 finaliza y continúa de igual forma, pero tomando como punto origen el nuevo punto de borde detectado (Punto 113).

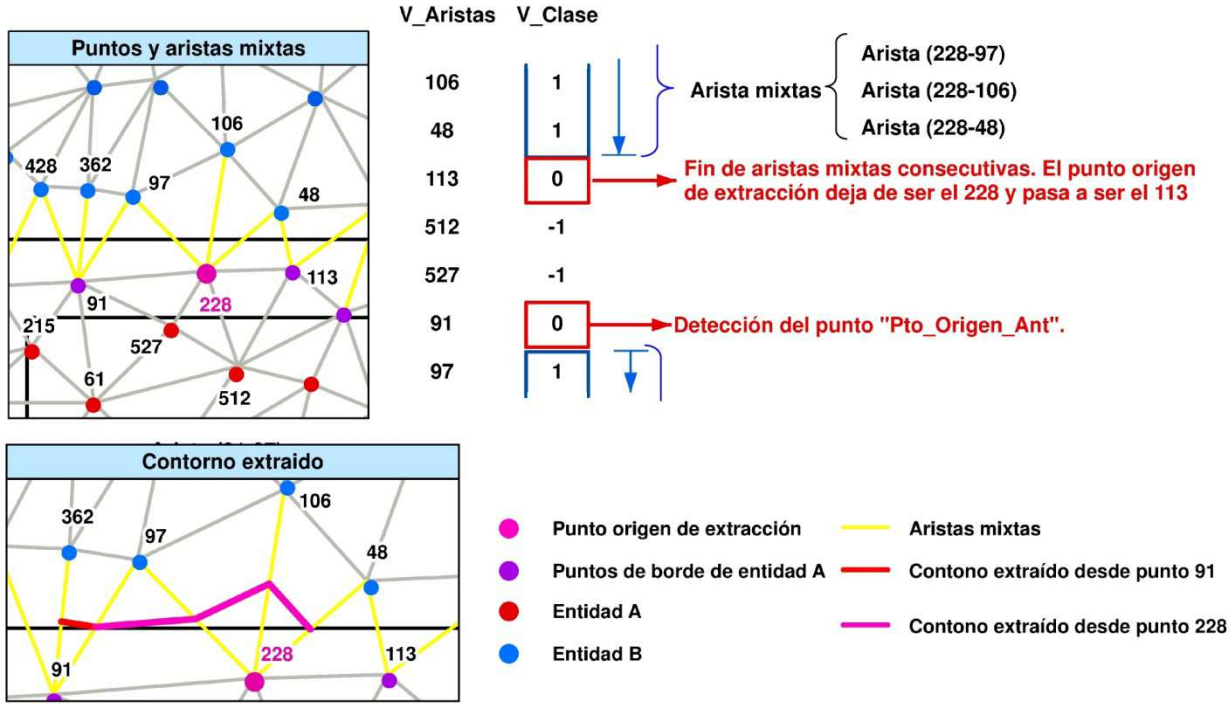


Figura 105. Extracción del contorno de entidades definidas por varios puntos LiDAR. Contorno asociado al segundo punto origen

El proceso descrito continua ininterrumpidamente hasta que la totalidad del polígono de contorno es extraído o hasta que se produce una intersección con el polígono límite de la zona de estudio.

En el caso de existir alguna intersección con el polígono límite de la zona de estudio, el algoritmo pasará a extraer sus vértices para delimitar el contorno de la entidad. Todo ello puede observarse en la Figura 106. El algoritmo detectará la intersección con el polígono límite de la zona de estudio mediante el vector de aristas (*V_Aristas*) y el vector de clasificación de vértices (*V_Clase*), ambos ordenados a partir de los acimutes asociados. Tal y como se ha comentado con anterioridad, los puntos pertenecientes al borde de la entidad cuyo contorno desea extraerse y que además pertenecen al límite de la zona estudio, estarán marcados con el valor "2" en el vector *V_Clase*. Por tanto una vez identificado este valor en el vector, el algoritmo modificará la variable *Dentro* introduciendo el valor False y comenzará a recorrer el polígono de contorno límite de la zona de estudio extrayendo sus vértices.

Continuando con el ejemplo propuesto, tras extraer el contorno a partir del cálculo del punto medio de las aristas mixtas, en el momento en que se toma como punto origen de extracción el Punto 732 (Figura 106a), la aplicacion extraerá los puntos medios de las tres aristas mixtas asociadas (arista 732-736, arista 732-755 y arista 732-787) utilizando el vector *V_Clase* (vértices marcados con el valor "1"). Seguidamente detectará el valor "2" que está asociado al Punto 743 que pertenece tanto al borde de la entidad A como al contorno límite de la zona de estudio, momento en el cual la variable *Dentro* cambiará su valor de True a False y comenzará a recorrerse el contorno límite de la zona de estudio.

La extracción del contorno asociado al límite de la zona de estudio tomará como origen de extracción el Punto 743 anteriormente detectado. Seguidamente, se realizará la búsqueda de dicho punto en el vector *V_Contorno_Límite* que contiene los puntos LiDAR que definen el polígono límite de la zona de estudio ordenados en el sentido de las agujas del reloj (Apartado III.1.2.2) y se extraerá el punto medio de la arista que define dicho punto (Punto 473) con el punto anterior del contorno (Punto 787) (Figura 106b). Finalmente se extraerán las coordenadas de los puntos LiDAR contenidos en el vector *V_Contorno_Límite* que están dispuestos de forma consecutiva hasta llegar a un punto cuyo vector de aristas tenga asociada una arista mixta (marcada con 1 en *V_Clase*), momento en el cual la variable Dentro pasará de nuevo a tomar el valor True y volverán a extraerse los puntos medios de las aristas mixtas siguiendo el proceso detallado con anterioridad (Figura 106c).

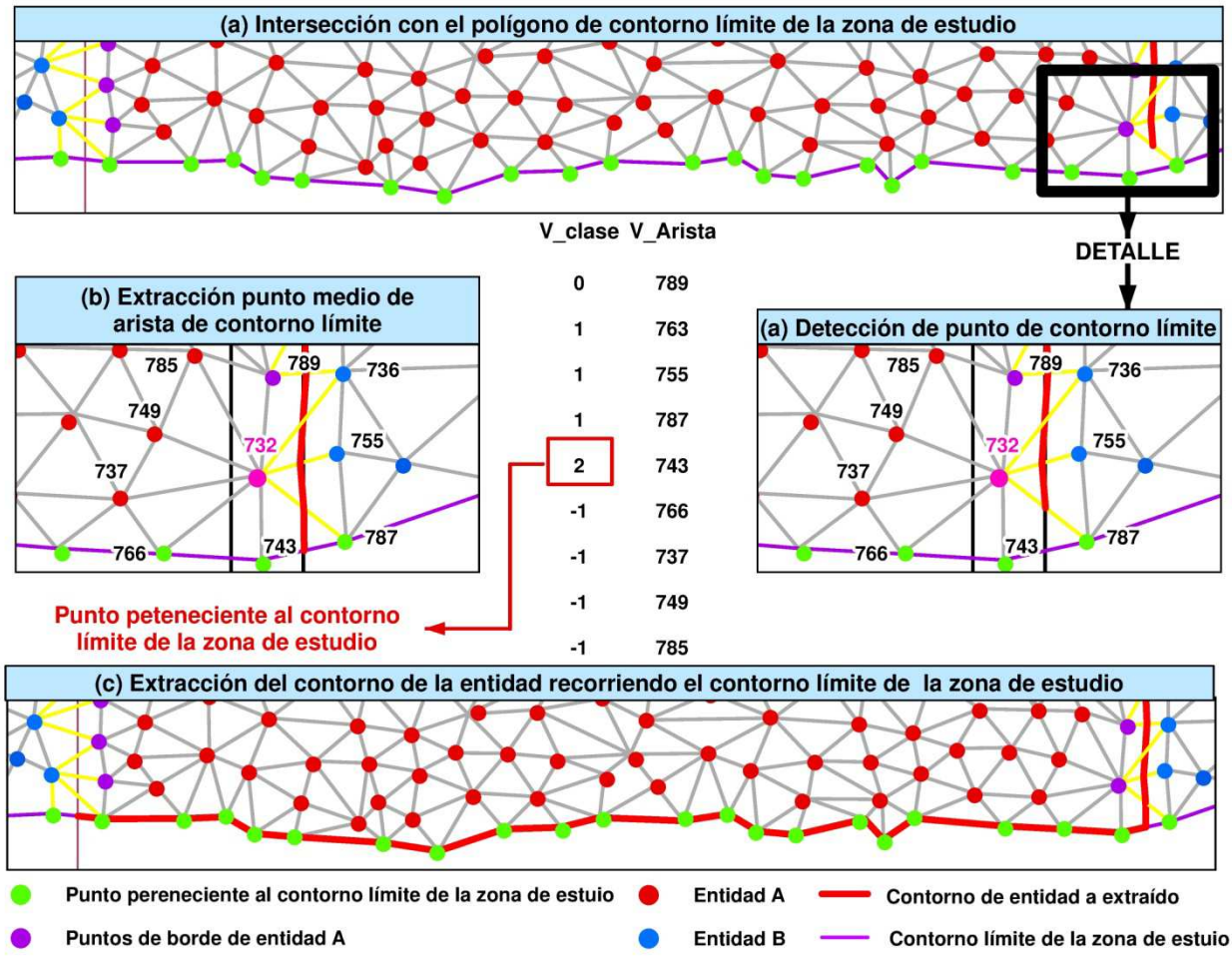


Figura 106. Extracción del contorno de una entidad en la zona límite de la nube de puntos.

h) Iteración del proceso hasta la extracción de la totalidad de contornos.

Una vez se ha extraído por completo el contorno de una entidad, se continuará recorriendo la matriz *V_Marcas_Bordes_Testeados* hasta encontrar el punto origen de extracción de una entidad cuyo contorno todavía no se haya extraído (marcado con el valor "1" en el vector *V_Marcas_Bordes_Testeados*) y se repetirá la totalidad del proceso desde el apartado b). En el momento en que el vector *V_Marcas_Bordes_Testeados* se recorra por completo significará que todos los contornos de la totalidad de entidades que conforman la escena han sido extraídos y por tanto el proceso de extracción finalizará.

Un aspecto de especial relevancia a lo largo de la totalidad del proceso es la actualización del vector de marcadores *V_Id_Bordes_Entidad_A*. Dicho vector es de igual dimensión que el número total de puntos LiDAR existentes en la nube de puntos y en él se marcarán los puntos de borde de la entidad A con el identificador de su polígono de contorno asociado. Para ello, siempre que se esté extrayendo el contorno a partir del cálculo del punto medio de las aristas mixtas, se almacenará el identificador de punto LiDAR del vértice perteneciente al borde de la entidad A de cada una de las aristas y se marcará dentro del vector *V_Id_Bordes_Entidad_A* con el identificador del contorno que se está extrayendo. Este vector será utilizado por algunas aplicaciones como por ejemplo el método de densificación en la detección del terreno.

Dada una entidad A constituida por una serie de objetos cuyos contornos desean ser extraídos, tal y como se observa en el siguiente ejemplo (Figura 107) los identificadores de los puntos LiDAR que pertenecen a aristas mixtas serán almacenados cuando dichos puntos son tomados como origen de extracción en el proceso. Seguidamente, se accede a su registro asociado dentro del vector V_Id_Bordes_*Entidad_A* y se almacena el identificador del contorno que se está extrayendo (en este caso el identificador 673).

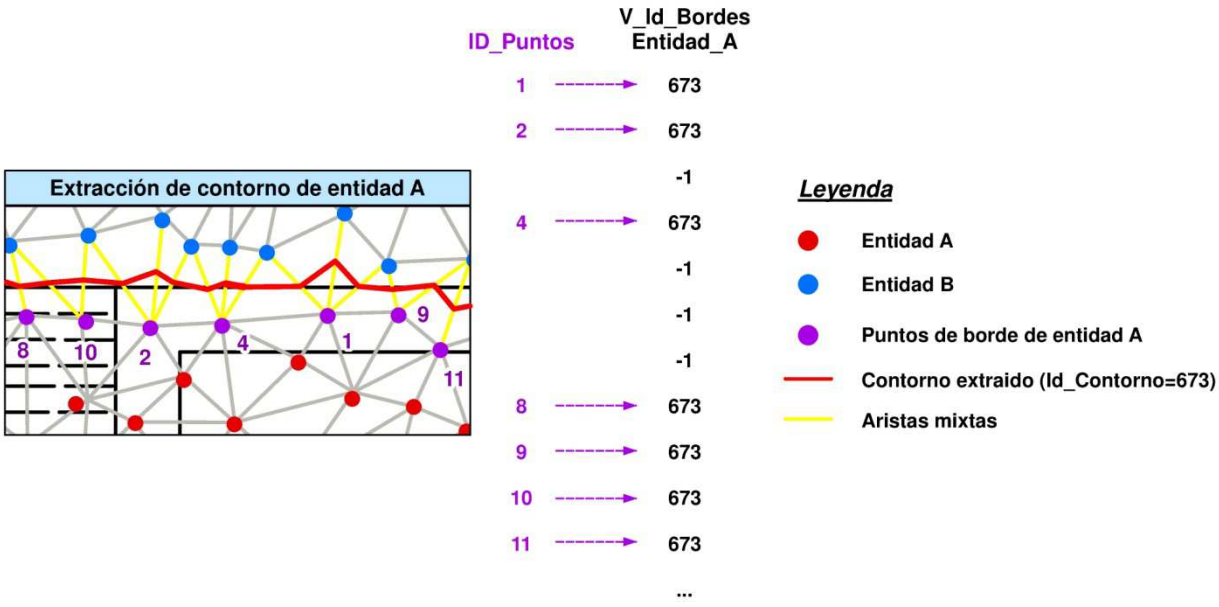


Figura 107. Proceso de definición del vector de bordes *V_Id_Bordes_Entidad_A*

- **Valores de retorno e impresión de archivos de salida**

La función implementada devuelve a la aplicación principal los siguientes datos:

M_Coord_Contor: constituido por la clase *Vector* disponible en C++, se tratará de una matriz de tres columnas, en la primera columna vendrá el identificador de contorno y en las dos restantes las coordenadas de cada vértice del polígono de contorno. Los vértices de los contornos estarán siempre ordenados en el sentido de las agujas del reloj.

V_Id_Bordes_Entidad_A: vector de igual dimensión que el número total de puntos LiDAR existentes en la zona de estudio donde los bordes de la entidad A cuyo contorno se ha extraído estarán marcados con el identificador de contorno asociado. El resto de puntos que no son de borde tendrán asociados el valor "-1".

Además, la función imprimirá el siguiente fichero de salida:

Nombre_Proyecto.Contornos_A.scr: se trata de un script de Autocad donde aparecerán vectorizados los contornos de una determinada entidad A. El contorno de cada uno de los objetos pertenecientes a dicha entidad vendrá dibujado con un color propio. En la Figura 108 se muestran los archivos de salida correspondientes al contorno del terreno para los distintos datos test.

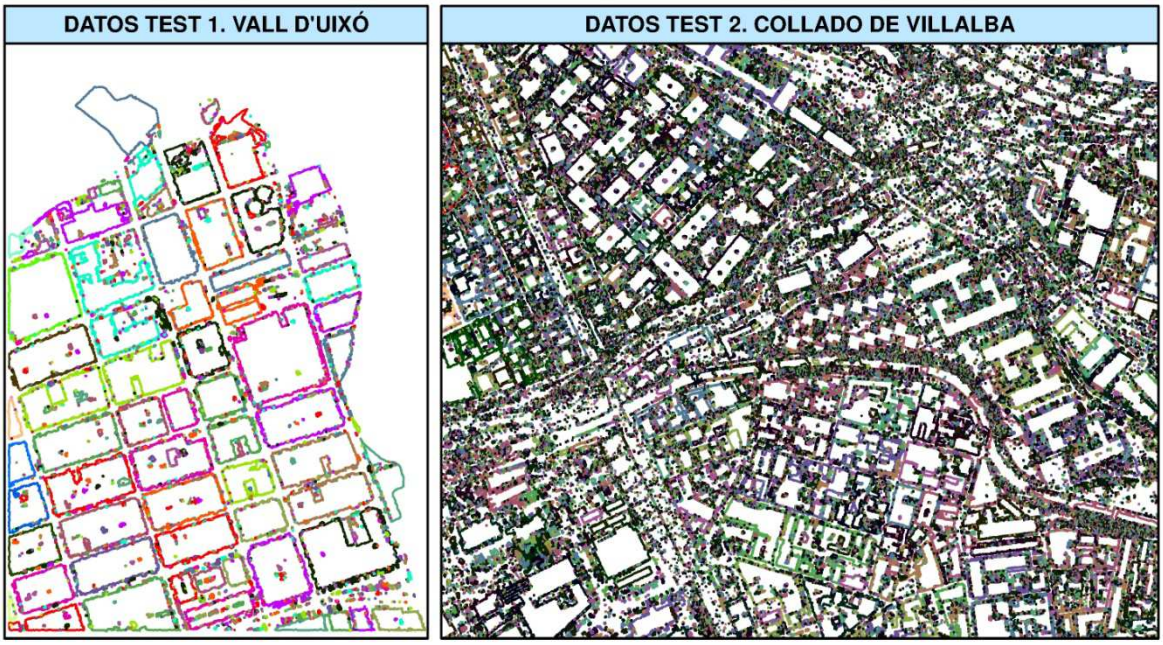


Figura 108. Contornos de la clase terreno para ambos datos test.

Nombre_Proyecto.Bordes_ID_Conotornos_A.scr: se trata de un script de Autocad donde aparecerán los puntos de bordes contenidos en el vector *V_Id_Bordes_Entidad_A* representados cada uno de un color en función de su identificador de contorno asociado. En la Figura 109 se muestran los puntos de borde para los edificios, donde cada borde tiene un color en función de su identificador de contorno asociado.

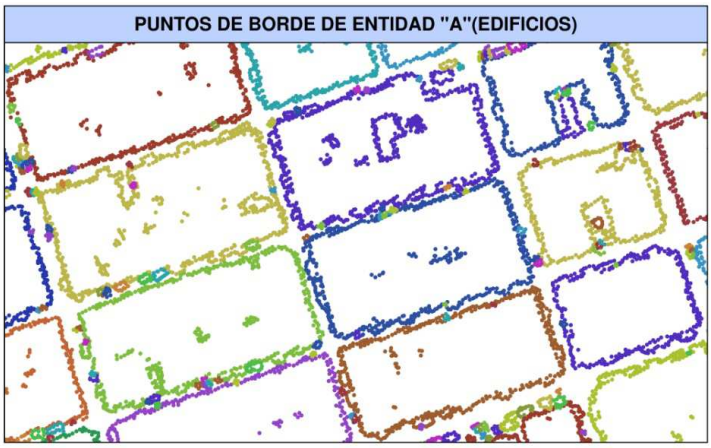


Figura 109. Puntos de borde del vector *V_Id_Bordes_Entidad_A*.

III.10.3. Determinación de la propiedad de inclusión entre una serie de puntos y un conjunto de polígonos

III.10.3.1. Motivación para el desarrollo del algoritmo y relación con los antecedentes

En diferentes fases de los distintos algoritmos implementados es necesario determinar si un punto está dentro de un polígono determinado (detección de polígonos islas, interpolación selectiva en el proceso de densificación, etc.). Para este fin existen distintos algoritmos dentro del campo de la geometría computacional (algoritmos basados en el teorema de la curva de Jordan, algoritmo radial...). De entre todos ellos se opta por un algoritmo basado en el teorema de la curva de Jordan (Márquez et al. s.f), ya que este teorema se cumple para todo tipo de polígonos independientemente de ser convexos o cóncavos. Además, resulta un algoritmo con alta eficiencia computacional y relativamente simple de implementar.

Dado que en todos los casos se trata de establecer la propiedad de inclusión entre una gran cantidad de puntos y polígonos será necesario desarrollar un nuevo algoritmo que mejore el algoritmo básico para adaptarlo a las necesidades de la aplicación.

Como ejemplo, en el proceso de interpolación selectiva se comprueba la propiedad de inclusión para un total de 2.832.195 puntos en 37.958 polígonos (datos test 2. Collado Villalba). Esto conlleva un alto coste computacional que se transmite a un alto tiempo de cálculo que hace inviable la aplicación directa del algoritmo de la curva de Jordan. Es por ello, que se opta por combinar el algoritmo con otros métodos de localización basados en la subdivisión plana del espacio tales como el método de la banda o el método de la cadena (Márquez et al. s.f). Además, antes de realizar intersecciones entre segmentos para comprobar si un punto está dentro de un polígono concreto, el algoritmo realizará un test previo, para determinar si el punto se encuentra dentro del *bounding-box* del polígono, reduciendo así las operaciones de cálculo considerablemente.

III.10.3.2. Descripción del algoritmo programado

Como se ha comentado con anterioridad, el algoritmo desarrollado se basa en el teorema de curva de Jordan. Este teorema se puede enunciar de la siguiente forma:

Si una semirrecta *R* corta a un polígono *Z* un número par de veces entonces el origen de la semirrecta se encuentra fuera del polígono; en cambio, si lo cruza un número impar de veces, el origen está dentro del mismo.

Este teorema permite determinar si un punto se encuentra dentro de un polígono *Z*, para ello basta con tomar como origen de la semirrecta horizontal el propio punto y contabilizar el número de intersecciones que se producen con el polígono en un sentido previamente elegido. Si el número de intersecciones es par el punto estará en todos los casos fuera del polígono; por el contrario, si el número de intersecciones es impar el punto estará dentro.

En la Figura 110 se puede observar el principio del algoritmo. Dado un polígono *Z* y tres puntos (P, Q y R) de los que se quiere determinar su propiedad de inclusión. El primer paso consiste en determinar el sentido donde se contabilizarán las intersecciones. En este caso, se tomarán como intersecciones válidas aquellas que están a la derecha del punto, es decir, las que poseen una coordenada X mayor que la del propio punto.

Para el punto *P*, la semirrecta horizontal *I1* corta al polígono *Z* en dos puntos. Por tanto el número de intersecciones es par y el punto *P* está obligatoriamente fuera del polígono. Por el contrario, la semirrecta *I2* con origen en el punto *Q* corta al polígono *Z* en un único punto y por consiguiente está dentro del polígono al tratarse de un número impar de intersecciones. De igual forma, la semirrecta *I3* cuyo origen está en el punto *R* corta al polígono en tres ocasiones y consecuentemente el punto *R* se encuentra dentro del polígono *Z* al tratarse de un número impar de intersecciones.

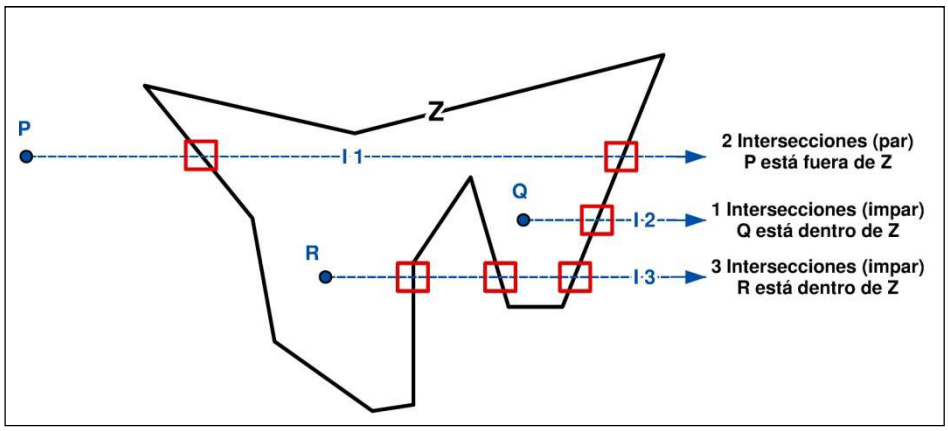


Figura 110. Algoritmo basado en el teorema de la curva de Jordan para determinar la propiedad de inclusión entre punto y polígono

El diagrama de flujo que sigue el algoritmo para determinar la propiedad de inclusión de una serie de puntos contenidos en la matriz *M_Puntos* y un conjunto de polígonos definidos en la *matriz M_Coord_Contor* puede observarse en la siguiente imagen (Figura 111).

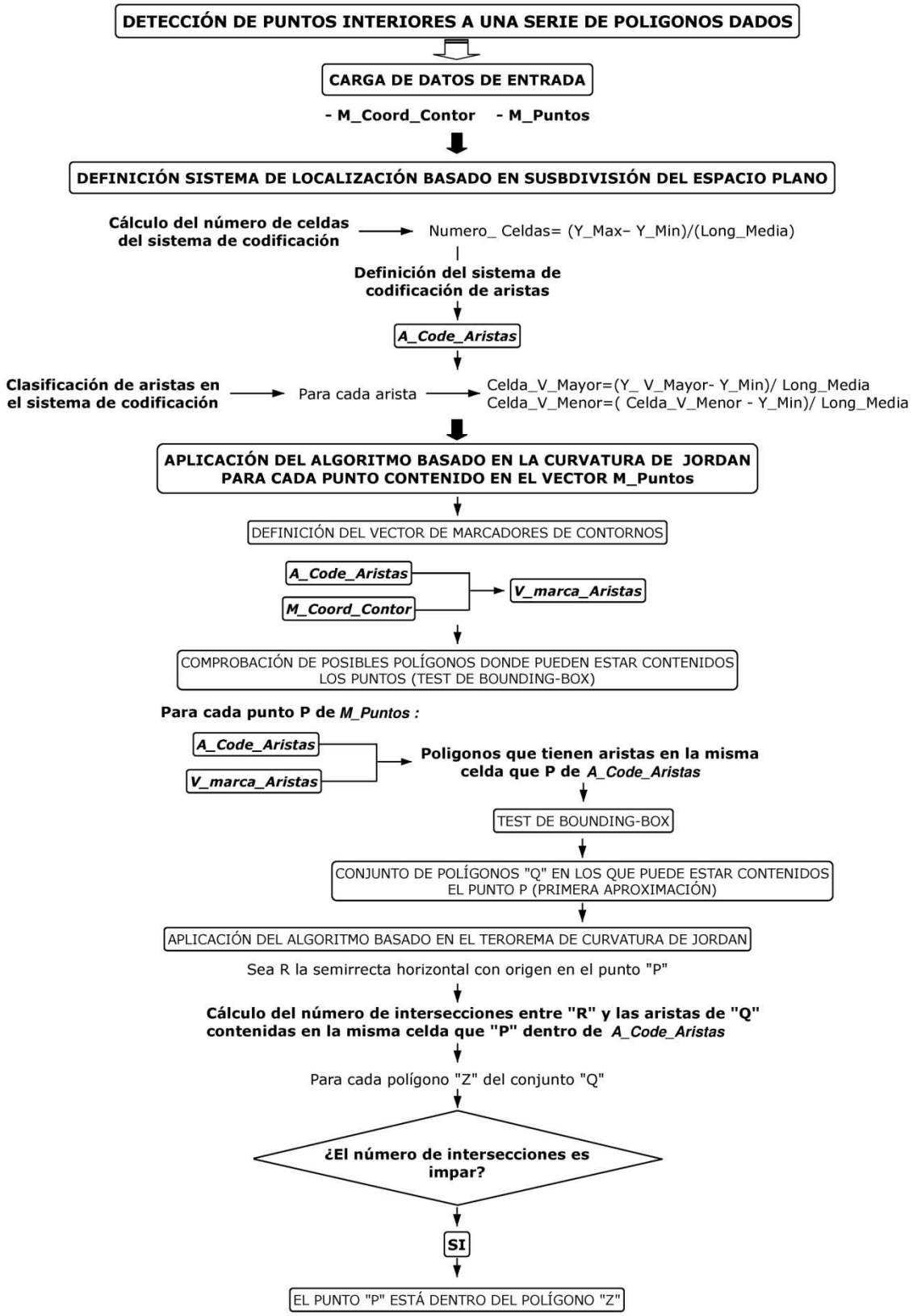


Figura 111. Algoritmo para determinar la propiedad de inclusión de una serie de puntos en un conjunto de polígonos. Diagrama de flujo.

Las distintas fases que sigue el proceso se detallan a continuación:

- **Carga de datos de entrada**

M_Coord_Contor: constituido por la clase *Vector* disponible en C++, se tratará de una matriz de tres columnas, en la primera columna vendrá el identificador de polígono y en las dos restantes las coordenadas de cada vértice del polígono de contorno. Los vértices de los contornos estarán siempre ordenados en el sentido de las agujas del reloj.

M_Puntos: matriz que contiene las coordenadas de los puntos cuya propiedad de inclusión desea analizarse.

- **Definición del sistema de locación basado en la subdivisión plana del espacio**

Dado que las aplicaciones que utilizan el presente algoritmo precisan determinar la propiedad de inclusión para cientos de miles de puntos y polígonos, la aplicación directa del algoritmo basado en la curvatura de Jordan resulta inviable debido al alto tiempo de cómputo. Para solucionar este problema se opta por desarrollar un nuevo sistema que combine el algoritmo mencionado con características propias de otros métodos de localización basados en la subdivisión plana del espacio. En este sentido se opta por utilizar un sistema similar al método de la banda (Márquez et al. s.f) pero que utilice las ventajas que proporciona el sistema de organización del espacio en celdas de altura infinita ya desarrollado (Apartado III.1.2.2).

Teniendo en cuenta que el algoritmo basado en la curva de Jordan utiliza una semirrecta horizontal para posteriormente calcular las intersecciones de dicha semirrecta con las aristas de cada uno de los polígonos existentes, todos los puntos de intersección resultantes tendrán la misma coordenada Y, que coincidirá además con la coordenada Y del punto origen de la semirrecta. El método desarrollado utiliza esta premisa para reducir considerablemente el número de operaciones a la hora de calcular las intersecciones. Para ello, el algoritmo calcula las intersecciones únicamente con aquellas aristas de los polígonos que discurren por la misma coordenada Y que el punto *P* origen de la semirrecta cuya propiedad de inclusión desea analizarse.

Básicamente, el algoritmo divide el espacio de sur a norte en celdas de igual tamaño, de tal forma que los identificadores de aquellas aristas que tienen solape vertical con una celda concreta son almacenados en la misma. Las celdas son cuadradas con una longitud de lado igual a la longitud media de la totalidad de las aristas de los polígonos existentes en la zona de estudio.

De forma general, el proceso que realiza el algoritmo para definir el sistema de codificación puede apreciarse en la Figura 112. Dados cuatro polígonos (*Z1*, *Z2*, *Z3* y *Z4*) cuyas aristas están definidas mediante un identificador numérico único, el primer paso consiste en definir el sistema de codificación (Figura 112a). Este sistema vendrá constituido por celdas de igual tamaño cuyos lados se corresponderán con la longitud media de la totalidad de aristas de los polígonos de la zona de estudio. Además, el origen del sistema estará en la coordenada "Y" mínima de la zona de estudio.

Una vez definido el sistema se proyectarán las aristas de los polígonos sobre las celdas. Gráficamente consiste en proyectar mediante rectas horizontales ambos vértices de cada arista sobre las celdas del sistema de codificación (Figura 112b). Finalmente, se utilizará dicha proyección para completar el sistema de codificación mediante el registro del identificador de cada arista en las celdas donde se produce solape (Figura 112c). Por

ejemplo, la proyección de los vértices *V-1* y *V-2* pertenecientes a la arista *A 97* del polígono *Z1* (Figura 112c) se produce en las celdas 1237 y 1238, por tanto el identificador de la arista se registrará en ambas celdas así como en las celdas intermedias en el caso de que existieran.

Gracias a este sistema el cálculo de intersecciones se reducirá al máximo al aplicar el algoritmo basado en la curva de Jordan en el proceso de determinar la propiedad de inclusión para un determinado punto *P*, ya que únicamente se comprobará sobre qué celda se proyecta horizontalmente el punto *P* y se calcularán las intersecciones con respecto a las aristas registradas en dicha celda dentro del sistema de codificación.

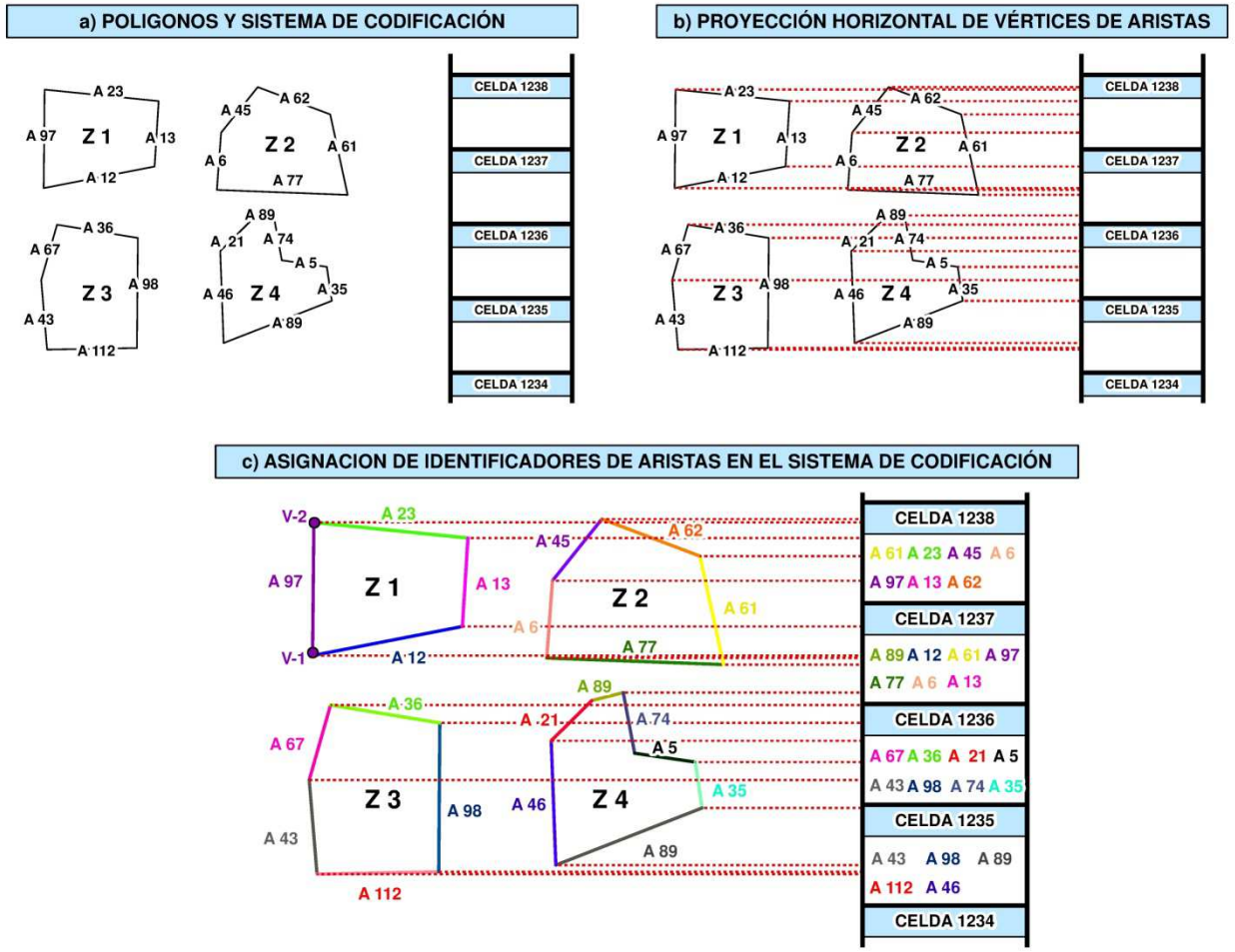


Figura 112. Sistema de localización basado en subdivisión plana del espacio

La aplicación comienza calculando la longitud media de todos los lados de los polígonos existentes en la zona de estudio. Esta longitud media se utilizará para definir el tamaño de los lados de las celdas que servirá para dividir el espacio. La división del espacio se realizará en el sentido del eje Y del sistema de coordenadas, siempre de sur a norte, coincidiendo el origen de la primera celda con la coordenada Y menor de la zona de estudio.

Una vez obtenido el tamaño del lado de las celdas será posible calcular el número de celdas totales que tendrá el sistema de codificación a partir de la siguiente expresión:

$$Numero\_Celdas = (Y\_Max - Y\_Min)/(Long\_Media)$$

Donde:

- Numero_ celdas: será el número de celdas totales que contiene el sistema de codificación.
- Y_Max: coordenada Y máxima de la totalidad de puntos LiDAR de la zona de estudio.
- Y_Min: coordenada Y mínima de la totalidad de puntos LiDAR de la zona de estudio.
- Long_Media: longitud media de las aristas de la totalidad de polígonos existentes en la zona de estudio.

Al disponer del número total de celdas que contendrá el sistema de codificación se procede a definir el mismo mediante un arreglo bidimensional que utiliza un doble puntero. De esta forma cada celda tendrá asociado un vector de longitud variable que contendrá los identificadores de aristas cuya proyección contenga solape con dicha celda. El arreglo se denominará *A_Code_Aristas*.

Tras definir el arreglo se procede a registrar las aristas en los vectores dinámicos asociados a las celdas. Para ello se calcula la celda a la que pertenecerán las proyecciones de los vértices adyacentes asociados a cada arista mediante la siguiente expresión:

$$Celda\_V\_Mayor=(Y\_\ V\_Mayor - Y\_Min)/\ Long\_Media$$
$$Celda\_V\_Menor=(\ Celda\_V\_Menor - Y\_Min)/\ Long\_Media$$

Donde:

- *Celda_V_Mayor*: número de la celda correspondiente al vértice de la arista que tiene la coordenada Y mayor.
- *Celda_V_Menor*: número de la celda correspondiente al vértice de la arista que tiene la coordenada Y menor.
- Y_Min: coordenada Y mínima de la totalidad de puntos LiDAR de la zona de estudio.
- Long_Media: longitud media de las aristas de la totalidad de polígonos existentes en la zona de estudio. Coincidirá con el ancho de celda.

Finalmente se introduce el identificador de cada arista en el vector asociado a las celdas *Celda_V_Menor* y *Celda_V_Mayor* así como en las celdas intermedias comprendidas entre ambas.

- **Utilización del sistema de codificación para el cálculo del número de intersecciones en la aplicación del algoritmo basado en la curva de Jordan**

Llegado a este punto el algoritmo generará un vector de marcadores (*V_marca_Aristas*) de igual dimensión que el número total de aristas existentes en la zona de estudio, donde para cada arista vendrá registrado el identificador del polígono al que pertenece.

El teorema de la curva de Jordan indica que un punto estará dentro de un polígono concreto si la semirrecta que tiene origen en el punto corta las aristas del polígono un número impar de veces. Para aplicar este principio el algoritmo utiliza la codificación de aristas en celdas de igual tamaño (*A_Code_Aristas*). El primer paso consistirá en determinar sobre qué celda del sistema se proyecta el punto cuya propiedad de inclusión desea analizarse. Para ello se utilizará la siguiente expresión:

$$Celda\_P=(Y\_P - Y\_Min)/\ Long\_Media$$

Donde:

- *Celda_P*: número de la celda del sistema de codificación sobre el que se proyectará el punto P cuya propiedad de inclusión se está analizando.
- Y_P: coordenada Y del punto P.
- Y_Min: coordenada Y mínima de la totalidad de puntos LiDAR de la zona de estudio.
- Long_Media: longitud media de las aristas de la totalidad de polígonos existentes en la zona de estudio. Coincidirá con el ancho de celda.

Continuando con el mismo ejemplo (Figura 113), dados dos puntos *P* y *Q* cuya propiedad de inclusión desea analizarse, y sean *R1* y *R2* las semirrectas horizontales con origen en dichos puntos. La primera fase será determinar la celda del sistema de codificación en la que se proyectan ambos puntos. En este caso concreto, tras aplicar la expresión anterior obtenemos la celda *1236* para el punto *Q* y la celda *1235* para el punto *P*. Dentro de dichas celdas estarán las aristas susceptibles de cortar a las semirrectas *R1* y *R2* por poseer la misma coordenada Y que los puntos. No obstante, estarán las aristas de multitud de polígonos, donde la mayoría de ellos no contendrán al punto objeto de análisis. Por este motivo, con el fin de calcular el número mínimo de intersecciones, se realizará un test previo a todos los polígonos que poseen aristas en dichas celdas. Para ello se detectarán los polígonos que contienen dichas aristas a partir del vector *V_marca_Aristas*.

Tal y como muestra la Figura 113, el punto *Q* se proyecta sobre la celda *1236* que contiene aristas pertenecientes al polígono *Z3* y *Z4*. De igual forma, el punto *P* está asociado a la celda *1235* que también contiene aristas pertenecientes a los polígonos *Z3* y *Z4*. Con este método el algoritmo obtendrá una primera aproximación y descartará la mayoría de polígonos de la zona de estudio.

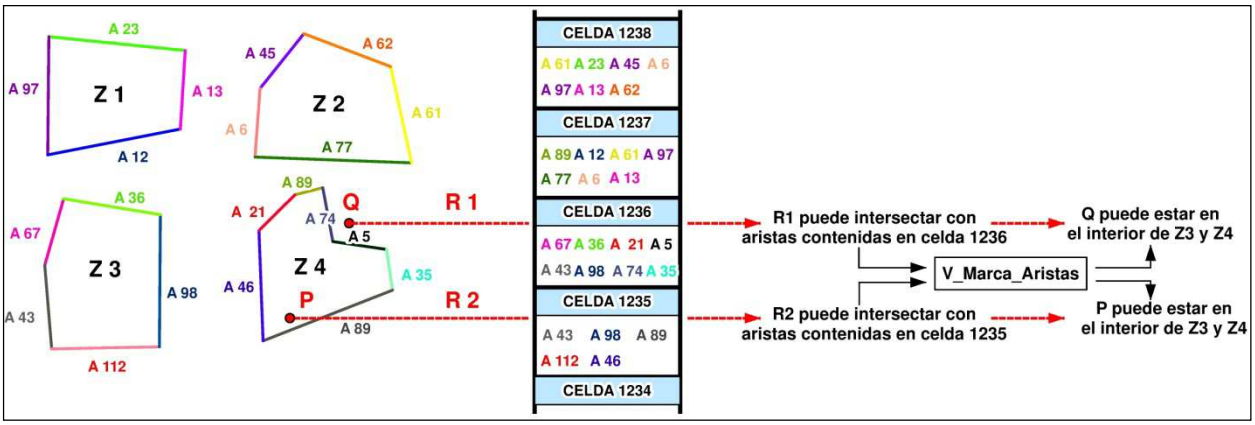


Figura 113. Primera aproximación al polígono en el que puede estar contenido un punto dado

Tras esta primera aproximación y previamente al cálculo de las intersecciones, el algoritmo comprobará si el punto objeto de análisis se encuentra en el interior del *bounding-box* de cada polígono que tiene aristas codificadas en la misma celda que *P*. Dado un polígono *Z*, se denomina *bounding-box* de *Z* al rectángulo de área mínima que lo contiene y cuyos vértices tienen las coordenadas máximas y mínimas del conjunto de vértices de *Z*. Para definir el *bounding-box* de un polígono *Z*, primero se buscarán las coordenadas máximas y mínimas de su conjunto de vértices (Figura 114a) y posteriormente, se definirá un rectángulo cuyos vértices posean dichas coordenadas (Figura 114b).

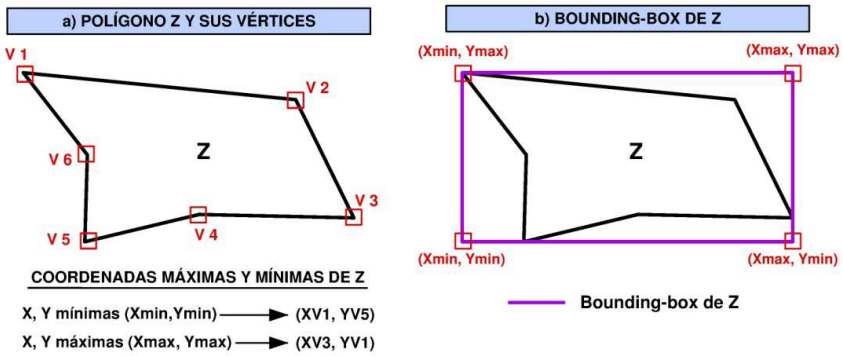


Figura 114. Definición del bounding-box para un polígono Z

Según el ejemplo propuesto, tras descartar los polígonos *Z1* y *Z2* a partir de la codificación en aristas, los puntos *P* y *Q* únicamente pueden estar contenidos en los polígonos *Z3* y *Z4*. Antes de intersecar las semirrectas horizontales con las aristas de dichos polígonos el algoritmo crea el *bounding-box* de los mismos y comprueba si los puntos objeto de análisis están dentro. Para ello únicamente deberá verificarse si las coordenadas de los puntos están comprendidas entre las coordenadas máximas y mínimas del polígono. En este caso, la coordenada X de los puntos *P* y *Q* no cumple esta premisa con respecto al polígono *Z3* (Figura 115a) y por tanto no podrán pertenecer al interior del mismo. Tras aplicar el test del *bounding-box* el polígono *Z3* quedará descartado y los puntos *P* y *Q* únicamente podrán pertenecer al polígono *Z4*.

Finalmente, tras descartar los polígonos no válidos después de la comprobación del *bounding-box,* el algoritmo calcula las intersecciones entre las semirrectas horizontales con origen en cada punto y las aristas contenidas en sus celdas asociadas pertenecientes al polígono *Z4*. Para el punto *Q* cuya celda asociada es la número 1236, el algoritmo calculará las intersecciones entre *R1* y las cuatro aristas contenidas en dicha celda que pertenecen a *Z4* (Figura 115b). Dado que no existe ninguna intersección, el punto *Q* no pertenecerá a *Z4* y finalmente no estará contenido en el interior de ningún polígono. Por el contrario, para el punto *P* cuya celda asociada es la número 1235, se calcularán los posibles puntos de intersección entre *R2* y dos aristas pertenecientes a *Z4* (Figura 115b), obteniendo un punto de intersección válido entre *R1* y la arista *A 89*. Dado que el número de intersecciones es impar, según el teorema de la curva de Jordan puede afirmarse que el punto *Q* pertenecerá al interior del polígono *Z4* (Figura 115c).

Para calcular las intersecciones entre segmentos se aplicará el algoritmo auxiliar de intersecciones (Algoritmo Auxiliar. Apartado III.10.4). El funcionamiento general de este algoritmo para dos segmentos dados consiste en definir la ecuación de la recta que contiene a cada uno de ellos y posteriormente calcular el punto de intersección. Finalmente se toma como punto de intersección válido aquel que satisface ambas ecuaciones y que además posee unas coordenadas dentro del rango que define la acotación de los extremos de los segmentos. Este algoritmo no solamente es válido para la intersección de segmentos, sino también para la intersección entre semirrectas o entre semirrectas y segmentos.

A la hora de calcular las intersecciones de la semirrecta horizontal con las aristas de los distintos polígonos, puede darse el caso de que la semirrecta pase por algún vértice de alguna de las aristas. En este caso el algoritmo propuesto fallaría al no detectar la intersección con la arista. Para solucionar el problema, en estos casos se realiza una rotación infinitesimal de la semirrecta alrededor del punto origen de la mima (*P*) en sentido contrario de las agujas del reloj, consiguiendo evitar así la situación.

Para determinar si el número de intersecciones es par o impar el algoritmo tiene en cuenta la premisa de que cualquier número par es divisible entre dos y consecuentemente el resto siempre es cero. Para ello se utiliza el operador aritmético "%" disponible en C++. Este operador devuelve automáticamente el resto de una división.

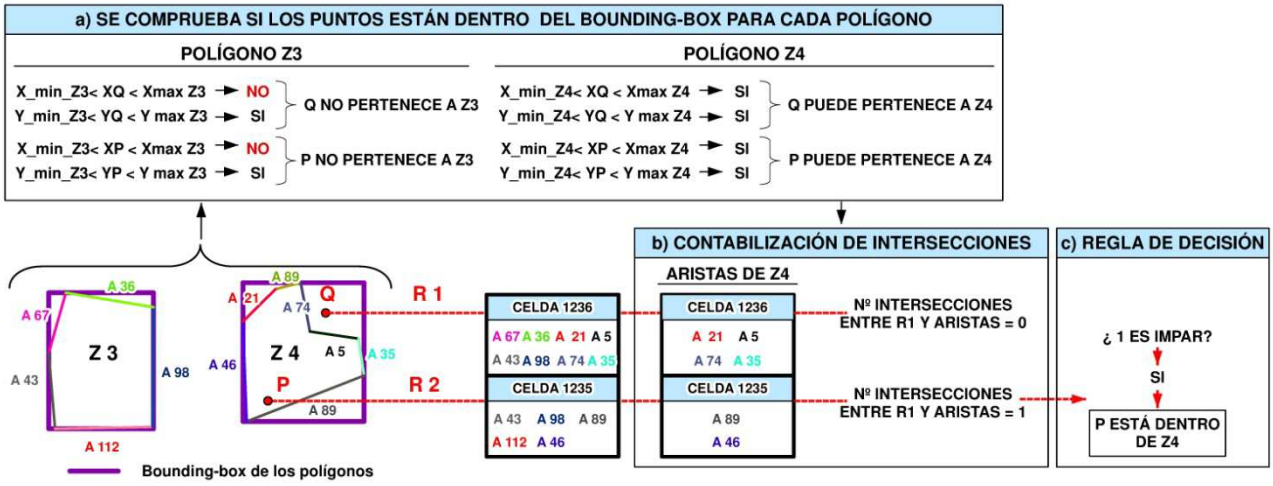


Figura 115. Verificación de puntos en el interior del bounding-box, contabilización de intersecciones y determinación de propiedad de inclusión.

Dado que el cálculo de intersecciones consume un alto coste computacional cuando se realiza para cientos de miles de segmentos, la totalidad del algoritmo ha sido diseñado para reducir al máximo el número de intersecciones a calcular. En el ejemplo propuesto existían cuatro polígonos con un total de veintiuna aristas. La aplicación directa del algoritmo basado en la curva de Jordan precisaría calcular las veintiuna intersecciones tanto para el punto *P* como para el punto *Q*. No obstante, tras realizar la codificación de aristas y el test del *bounding-box* únicamente ha sido preciso calcular dos intersecciones para el punto *P* y cuatro para el punto *Q*. Esta reducción del número de intersecciones a calcular se hace más patente cuando se trabaja con datos LiDAR. Tomando como ejemplo un punto LiDAR (punto con ID=1.533.197) situado en la zona central de los datos test 2 (Collado Villalba). El algoritmo de densificación para detectar el terreno debe de determinar si el punto está dentro de alguno de los 37.958 polígonos que definen el contorno de la clase terreno provisional. A su vez, los 37.958 polígonos están definidos por un total de 1.201.097 aristas. Esto se traduce a que la aplicación directa del algoritmo basado en la curva de Jordan debería calcular más de un millón de intersecciones para determinar si un solo punto está dentro de alguno de los polígonos. No obstante, la codificación en aristas reduce automáticamente el número a tan solo 4.792 aristas cuya intersección con la semirrecta horizontal es posible. Estas 4.792 aristas pertenecen a un total de 342 polígonos. Tras comprobar si el punto se encuentra dentro del *bounding-box* de dichos polígonos 340 polígonos son automáticamente descartados, por lo que el punto únicamente puede pertenecer al interior de 2 polígonos. Dichos polígonos tienen tan solo 9 aristas registradas en la misma celda que el punto cuya propiedad de inclusión se está analizando. Por tanto el número de intersecciones a calcular se reduce de más de un millón doscientos mil a tan solo nueve.

- **Valores de retorno**

A_Puntos_Interiores_Pol: arreglo bidimensional definido mediante un doble puntero. De forma que cada punto LiDAR tendrá un vector asociado con los identificadores de los polígonos en los que está contenido.

III.10.4. Intersección entre los segmentos

III.10.4.1. Motivación para el desarrollo del algoritmo y relación con los antecedentes

En distintas fases del proceso de algunas aplicaciones implementadas es necesario realizar el cálculo de las coordenadas del punto de intersección donde dos segmentos se cruzan. Este problema es de resolución sencilla y es tratado comúnmente en el ámbito de la geometría computacional.

Para determinar si dos segmentos se cruzan resulta de uso común la utilización del producto cruzado de los vectores que definen ambos segmentos. No obstante, en el caso de que las coordenadas del punto de intersección deseen calcularse, la gran mayoría de los algoritmos existentes se basan en determinar las ecuaciones de las rectas que contienen a dichos segmentos para posteriormente generar un sistema de dos ecuaciones con dos incógnitas, siendo precisamente las dos incógnitas las coordenadas del punto intersección a determinar.

III.10.4.2. Descripción del algoritmo programado

Básicamente el algoritmo desarrollado se basa en determinar la ecuación explicita de las rectas que contienen a ambos segmentos para posteriormente calcular el punto de intersección entre ambas rectas. Si dicho punto está acotado con respecto a los extremos de los segmentes, se tratará de un punto de intersección válido.

La función implementada recibe las coordenadas de los vértices extremos de los segmentos y devuelve las coordenadas del punto intersección así como el valor "True" si existe intersección o "False" en caso contrario. Los distintos pasos que realiza el algoritmo se aprecian en el siguiente diagrama de flujo (Figura 116).

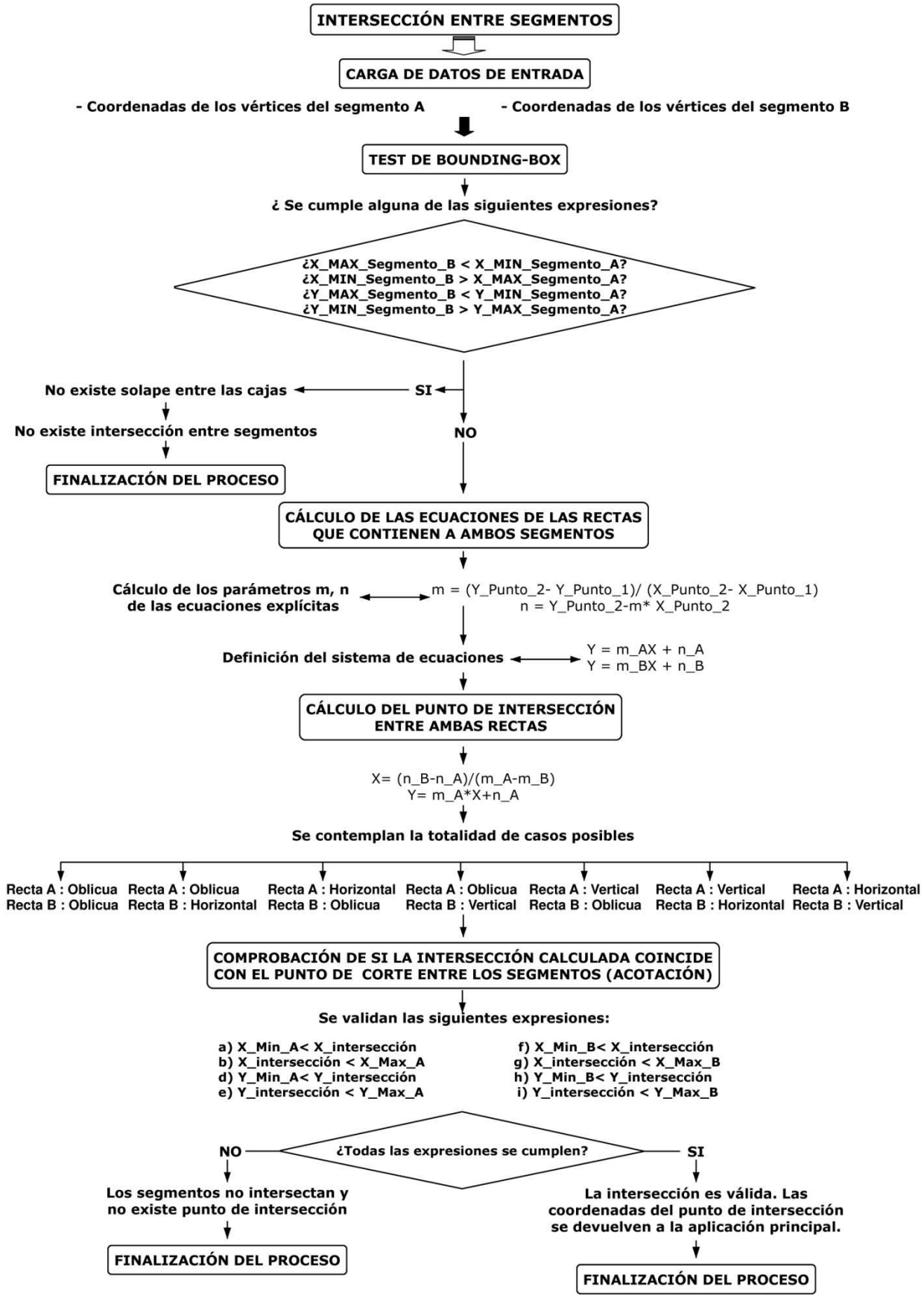


Figura 116. Algoritmo para calcular la intersección entre segmentos. Diagrama de flujo.

De forma detallada, las distintas fases que realiza el algoritmo son las siguientes:

- **Carga de datos de entrada**

Coordenadas de los vértices del primer segmento (segmento A):

XV1_Segmento_A: coordenada X del vértice 1 del segmento A.
XY1_Segmento_A: coordenada Y del vértice 1 del segmento A.
XV2_Segmento_A: coordenada X del vértice 2 del segmento A.
XY2_Segmento_A: coordenada Y del vértice 2 del segmento A.

Coordenadas de los vértices del segundo segmento (segmento B):

XV1_Segmento_B: coordenada X del vértice 1 del segmento B.
XY1_Segmento_B: coordenada Y del vértice 1 del segmento B.
XV2_Segmento_B: coordenada X del vértice 2 del segmento B.
XY2_Segmento_B: coordenada Y del vértice 2 del segmento B.

- **Test del bounding-box**

Previamente al cálculo de las coordenadas del punto de intersección se realizará un test con el fin de descartar aquellos casos donde la intersección de los segmentos no es posible. Este test consiste en verificar que las *boundig-box* de los segmentos tienen alguna superposición.

El bounding-box de los segmentos es la caja que define los vértices extremos de cada uno de ellos. En el caso de que la intersección sea posible siempre existirá solape entre dichas cajas. En la Figura 117 se muestran tres supuestos diferentes. En el primero de ellos (Figura 117a) no hay solape entre las cajas asociadas a ambos segmentos, lo que significa que no existe intersección posible entre ellos. En el segundo caso (Figura 117b) sí existe intersección entre los segmentos y consecuentemente existe solape entre las cajas asociadas a los mismos. El hecho de que no exista solape entre las cajas asegura la nula intersección entre los segmentos. No obstante, el caso contrario no se cumple, ya que la existencia de solape entre las cajas no asegura la existencia de intersección. Este aspecto se aprecia en el último de los casos (Figura 117c), donde sí existe solape entre las cajas pero no intersección entre los segmentos.

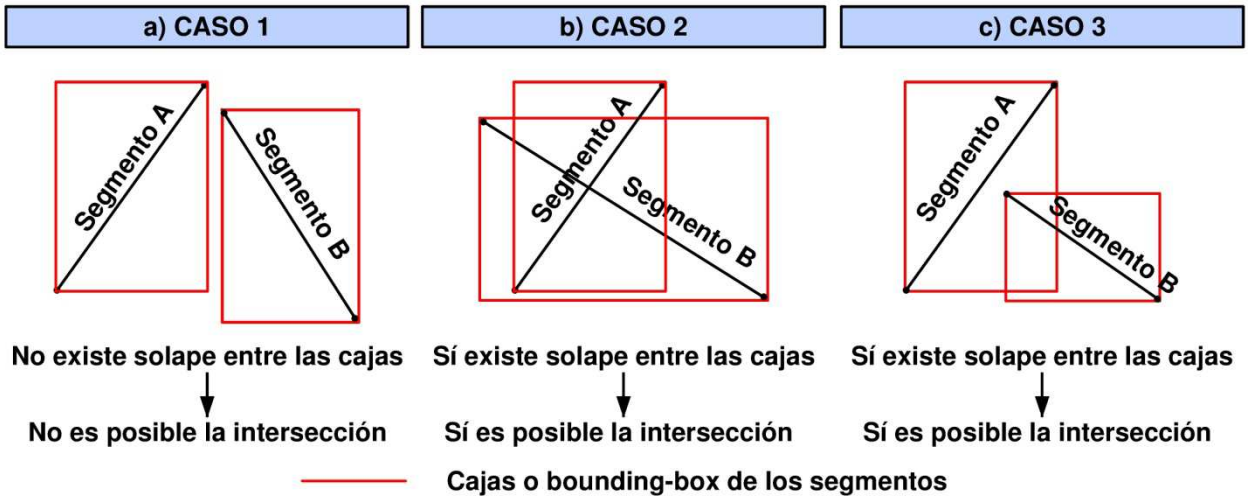


Figura 117. Test del bounding-box en el cálculo de intersecciones entre segmentos.

Como norma general, la mayoría de los casos que se presentan en la práctica se corresponden con el primer y segundo supuesto. En consecuencia este sencillo test permite detectar de forma eficiente aquellos casos donde los segmentos no intersectan (caso 1) y evitar así los consiguientes cálculos asociados a la obtención de las coordenadas del punto de intersección.

El algoritmo comienza realizando el cálculo de las coordenadas máximas y mínimas de los vértices que definen cada uno de los segmentos:

X_MAX_Segmento_A: coordenada X máxima de los vértices del segmento A.
Y_MAX_Segmento_A: coordenada Y máxima de los vértices del segmento A.
X_MIN_Segmento_A: coordenada X mínima de los vértices del segmento A.
Y_MIN_Segmento_A: coordenada Y mínima de los vértices del segmento A.
X_MAX_Segmento_B: coordenada X máxima de los vértices del segmento B.
Y_MAX_Segmento_B: coordenada Y máxima de los vértices del segmento B.
X_MIN_Segmento_B: coordenada X mínima de los vértices del segmento B.
Y_MIN_Segmento_B: coordenada Y mínima de los vértices del segmento B.

Para determinar si existe solape entre las cajas (test del *bounding-box*) correspondiente a ambos segmentos el algoritmo evaluará las siguientes expresiones:

Si (***X_MAX_Segmento_B*** < ***X_MIN_Segmento_A***) -> No hay solape entre cajas.
Si (***X_MIN_Segmento_B*** > ***X_MAX_Segmento_A***) -> No hay solape entre cajas.
Si (***Y_MAX_Segmento_B*** < ***Y_MIN_Segmento_A***) -> No hay solape entre cajas.
Si (***Y_MIN_Segmento_B*** > ***Y_MAX_Segmento_A***) -> No hay solape entre cajas.

Si el algoritmo verifica que se cumple alguna de las expresiones anteriores significará que no existe intersección entre ambos segmentos. En este caso finalizará el proceso y se devolverá a la aplicación principal el valor *False*, indicando así que no existe intersección. En caso contrario, el proceso de cálculo del punto de intersección entre ambos segmentos continúa.

- **Determinación de las ecuaciones de las rectas que contienen ambos segmentos para calcular el punto de intersección entre ambas**

En los casos donde se determine que la intersección es posible tras realizar el test de *bounding-box*, el algoritmo pasará a definir las ecuaciones explicitas de la recta que contiene a cada segmento.

El sistema que se planteará con posterioridad para calcular las coordenadas del punto de intersección puede estar mal condicionado si los coeficientes son muy altos. En estos casos una pequeña variación en los coeficientes puede transmitir grandes errores en los resultados. Dado que al trabajar con datos LiDAR podemos disponer de nubes de puntos en diferentes sistemas de coordenadas y proyecciones, resulta frecuente encontrarnos con magnitudes altas en las coordenadas que se traducen en altos valores de los coeficientes dentro del sistema a plantear. Para evitar sistemas mal condicionados en esta parte del proceso, se realizará una traslación del origen de coordenadas. Concretamente el origen de coordenadas se colocará siempre en las coordenadas X,Y mínimas de entre los cuatro vértices que definen los segmentos.

Dados dos segmentos A y B que estarán contenidos en dos rectas *R_A* y *R_B* (Figura 118). Las ecuaciones explícitas de dichas rectas pueden calcularse a partir de dos puntos de las mismas. Para ello se tomarán los dos vértices de cada segmento. Sea la ecuación explícita de la recta:

$$Y=mX + n$$

Donde:

- *Y*: coordenada Y de cualquier punto de la recta.
- X: coordenada X de cualquier punto de la recta.
- m: pendiente de la recta.
- n: ordenada origen (coordenada Y donde la recta corta al eje de ordenadas).

Podremos obtener el coeficiente (m) y el término independiente (n) a partir de las siguientes expresiones:

m=(Y_Punto_2 - Y_Punto_1)/ (X_Punto_2 - X_Punto_1)
n=Y_Punto_2 - m* X_Punto_2

Donde los puntos (*Punto_1 y Punto_2*) serán los vértices adyacentes de cada segmento. Además de obtener las ecuaciones de las rectas asociadas a segmentos oblicuos el algoritmo también detectará las rectas horizontales (*Y_Punto_1=Y_Punto_2*) y las rectas verticales (*X_Punto_1=X_Punto_2*)

- **Cálculo del punto intersección entre las rectas que contienen a ambos segmentos**

Una vez se han calculado los parámetros de las rectas que contienen a ambos segmentos, las coordenadas del punto de intersección podrán obtenerse resolviendo un sistema de ecuaciones lineales. Dadas dos rectas R_A y R_B que contienen a los segmentos A y B, las coordenadas del punto de intersección se obtendrán mediante la resolución del siguiente sistema de ecuaciones:

Recta R_A → Y=m_AX + n_A
Recta R_B → Y=m_BX + n_B

Donde:

- *Y*: coordenada Y del punto intersección.
- X: coordenada X del punto intersección.
- m_A: pendiente de la recta RA que contiene al segmento A.
- m_B: pendiente de la recta RB que contiene al segmento B.
- n_A: ordenada origen de la recta RA.
- n_B: ordenada origen de la recta RB.

El algoritmo tendrá en cuenta los siguientes casos:

a) Las dos rectas son oblicuas y paralelas. Por tanto no existe intersección: en este caso las dos rectas son oblicuas pero la pendiente "m" es la misma. Por tanto, no existe intersección. El algoritmo finalizará el proceso y devolverá el valor *False* a la aplicación principal.

b) Las dos rectas son horizontales. En este caso no existirá intersección ya que serán paralelas. El algoritmo finalizará el proceso y devolverá el valor False a la aplicación principal.

c) Las dos rectas son verticales. En este caso no existirá intersección ya que serán paralelas. El algoritmo finalizará el proceso y devolverá el valor False a la aplicación principal.

d) Ambas rectas son oblicuas y no paralelas. En este caso se resolverá el sistema de ecuaciones mediante las expresiones:

$$X= (n\_B - n\_A)/(m\_A - m\_B)$$
$$Y= m\_A*X + n\_A$$

e) La recta R_A es oblicua y la recta R_B es horizontal. En este caso las coordenadas del punto intersección entre ambas rectas se obtienen a partir de las expresiones:

$$Y = n\_B = \text{Coordenada Y de cualquier punto de la recta horizontal}$$
$$X=(Y - n\_A)/m\_A$$

f) La recta R_A es horizontal y la recta R_B es oblicua. Las coordenadas del punto de intersección se obtendrán en este caso mediante las expresiones:

$$Y = n\_A = \text{Coordenada Y de cualquier punto de la recta horizontal}$$
$$X = (Y - n\_B)/m\_B$$

g) La recta R_A es oblicua y la recta R_B es vertical. Las coordenadas del punto intersección se obtendrán en este caso mediante las expresiones:

$$X = \text{Coordenada X de cualquier punto de la recta vertical}$$
$$Y=m\_A*X + N\_A$$

h) La recta R_A es vertical y la recta R_B es oblicua. Las coordenadas del punto de intersección se obtendrán en este caso mediante las expresiones:

$$X = \text{Coordenada X de cualquier punto de la recta vertical}$$
$$Y=m\_B*X + N\_B$$

i) La recta R_A es horizontal y la recta R_B es vertical. Las coordenadas del punto de intersección se obtendrán en este caso mediante las expresiones:

$$X = \text{Coordenada X de cualquier punto de la recta vertical (R\_B)}$$
$$Y= \text{Coordenada Y de cualquier punto de la recta horizontal (R\_A)}$$

j) La recta R_A es vertical y la recta R_B es horizontal. Las coordenadas del punto de intersección se obtendrán en este caso mediante las expresiones:

$$X = \text{Coordenada X de cualquier punto de la recta vertical (R\_A)}$$
$$Y = \text{Coordenada Y de cualquier punto de la recta horizontal (R\_B)}$$

- **Acotación del punto de intersección para establecer si se trata de una intersección válida**

En esta parte del proceso se dispondrá de los puntos de intersección entre ambas rectas que contienen a los segmentos. No obstante, el punto de intersección no tiene por qué pertenecer a los segmentos dados. Dados dos pares de segmetos (*A,B* y *C,D*), si consideramos la intersección de las rectas que los contienen (R_*A, R_B* y *R_C, R_D*). Tal y como se aprecia en la Figura 118, puede ocurrir que las rectas se crucen en el mismo punto de intersección que tienen los segmentos (Figura 118a), pero también puede ocurrir que los segmentos no se corten en ningún punto y, sin embargo, las rectas sí (Figura 118b). Por tanto, una vez calculado el punto de intersección deberá determinarse si dicha intersección se corresponde realmente con el punto de corte entre los segmentos.

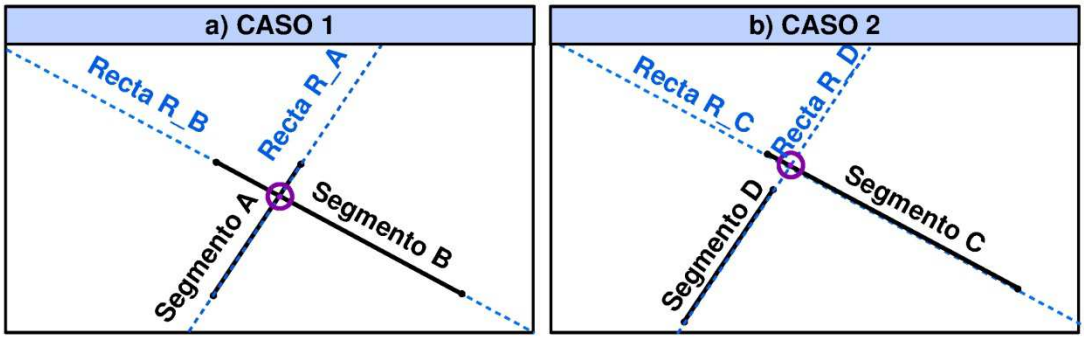


Figura 118. Validación de intersección entre segmentos

Dados dos segmentos A y B, para determinar si ambos se cruzan en algún punto, tras calcular el punto de intersección que determinan las rectas R_A y R_B el algoritmo verificará si se cumplen las siguientes expresiones:

a) X_Min_A< X_intersección
b) X_intersección < X_Max_A
d) Y_Min_A< Y_intersección
e) Y_intersección < Y_Max_A
f) X_Min_B< X_intersección
g) X_intersección < X_Max_B
h) Y_Min_B< Y_intersección
i) Y_intersección < Y_Max_B

Donde:

- X_Min_A , X_Max_A: coordenadas X máximas y mínimas de los vértices del segmento A.
- Y_Min_A , Y_Max_A: coordenadas Y máximas y mínimas de los vértices del segmento A.
- X_Min_B , X_Max_B: coordenadas X máximas y mínimas de los vértices del segmento B.
- Y_Min_B , Y_Max_B: coordenadas Y máximas y mínimas de los vértices del segmento B.
- X_intersección, Y_intersección: coordenadas del punto intersección de las rectas R_A y R_B.

Si las ocho expresiones se cumplen, entonces el punto de intersección se corresponderá con el punto de corte entre ambos segmentos. Sin embargo, si alguna de las expresiones anteriores no se cumple, significará que no existe intersección entre los segmentos dados.

- **Valores de retorno**

El algoritmo devuelve el valor de una variable tipo booleana. En el caso de que exista intersección entre los dos segmentos dados, devolverá el valor *True*; en caso contrario devolverá *False*. Además, si la intersección existe devolverá las coordenadas del punto de intersección.

X_intersección, Y_intersección: Coordenadas del punto de intersección entre los dos segmentos.

III.10.5. Cálculo de la superficie de las regiones obtenidas tras segmentar la escena en superficies continuas

III.10.5.1. Motivación para el desarrollo del algoritmo y relación con los antecedentes

El tamaño de las regiones obtenidas tras segmentar una nube de datos LiDAR, es un parámetro frecuentemente utilizado para diferenciar las distintas entidades que conforman la escena. En este sentido, las regiones de mayor tamaño se corresponden con grandes

superficies continuas y pueden clasificarse como terreno (Lari y Habib 2012; Moussa y El-Sheimy 2012). Además, la utilización de un parámetro umbral en el tamaño de las regiones resulta efectivo para diferenciar los edificios de otros pequeños objetos tales como vegetación o automóviles (Kwak et al. 2012; Vögtle y Steinle 2000; Moussa y El-Sheimy 2012).

En la metodología propuesta en el presente trabajo, la segmentación de los datos LiDAR y el posterior análisis de los distintos atributos asociados a las regiones obtenidas constituye un aspecto fundamental. En concreto, el tamaño de las regiones es utilizado en diferentes aplicaciones tales como la detección y filtrado de errores groseros, clasificación del terreno, detección de edificios y clasificación de vegetación y de pequeños objetos. Consecuentemente, la implementación de un algoritmo para calcular de forma eficiente y precisa el área de las regiones es necesario. A continuación se describe con detalle el algoritmo programado para tal fin.

III.10.5.2. Descripción del algoritmo programado

En geometría computacional, uno de los métodos más empleados para calcular el área de un polígono consiste en triangular dicho polígono y posteriormente realizar el sumatorio de las superficies de los triángulos obtenidos tras la triangulación. En el caso que nos ocupa las regiones obtenidas tras la segmentación cuya área desea obtenerse no vienen dadas en forma de polígonos sino mediante una entidad de puntos, donde los puntos que pertenecen a una misma región poseen el mismo identificador. No obstante, dado que la nube de puntos tiene asociada una triangulación de Delaunay puede aplicarse una metodología análoga donde el área de cada región sea calculada a partir de sus triángulos asociados.

En primer lugar, una clasificación de la totalidad de los triángulos contenidos en la triangulación de Delaunay se lleva a cabo, consiguiendo asociar así cada triángulo a la región que lo contiene. La clasificación se realiza mediante la identificación de la región que contiene a cada uno de los vértices de los triángulos. En este sentido, pueden darse tres casos: que un triángulo tenga todos sus vértices pertenecientes a una misma región; que tenga dos vértices pertenecientes a una misma región y un vértice asociado a una región distinta; o bien que tenga sus tres vértices pertenecientes a distintas regiones.

Básicamente el algoritmo calcula en primer lugar el área de cada uno de los triángulos contenidos en la triangulación de Delaunay y los almacena en un vector (*V_Superficie_Triangulos*). Posteriormente recorre la totalidad de triángulos dentro de la matriz de triángulos (*Tabla_Triangulos*) almacenando los identificadores de cada uno de sus vértices a partir del vector *V_M_Puntos*, obteniendo así tres posibles casos (Figura 119):

a) Los tres vértices pertenecen a una misma región (Figura 119a): en este caso la totalidad del área del triángulo es añadida a la superficie de la región asociada.

b) Dos vértices del triángulo pertenecen a una misma región y el vértice restante a una región diferente (Figura 119b): las dos terceras partes del área del triángulo son añadidas a la región que contiene los dos vértices y la tercera parte restante a la otra región.

c) Los tres vértices del triángulo pertenecen a regiones distintas (Figura 119c): la tercera parte del área del triángulo es añadida a la superficie de las tres regiones asociadas.

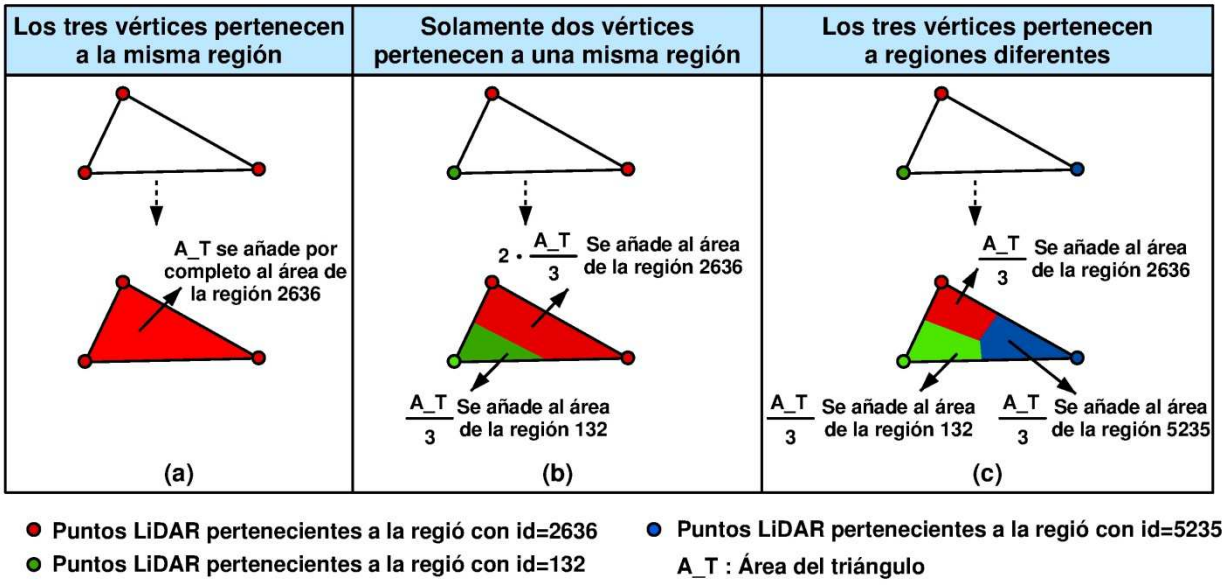


Figura 119. Distintos casos posibles en la asignación de áreas de triángulos para el cálculo de superficie de regiones.

El algoritmo recorre la totalidad de los triángulos asignando el área correspondiente a las regiones que contienen los vértices de cada uno de ellos, esta asignación del área se realiza en un vector de igual dimensión que el número total de regiones obtenidas tras segmentar la escena (*V_Superficie_Regiones*), de forma que el área de cada región es el resultado del sumatorio de las áreas (totales o parciales según el caso) de los distintos triángulos que poseen algún vértice perteneciente a dicha región.

El diagrama de flujo que sigue el algoritmo para calcular el área de las regiones es el siguiente (Figura 129):

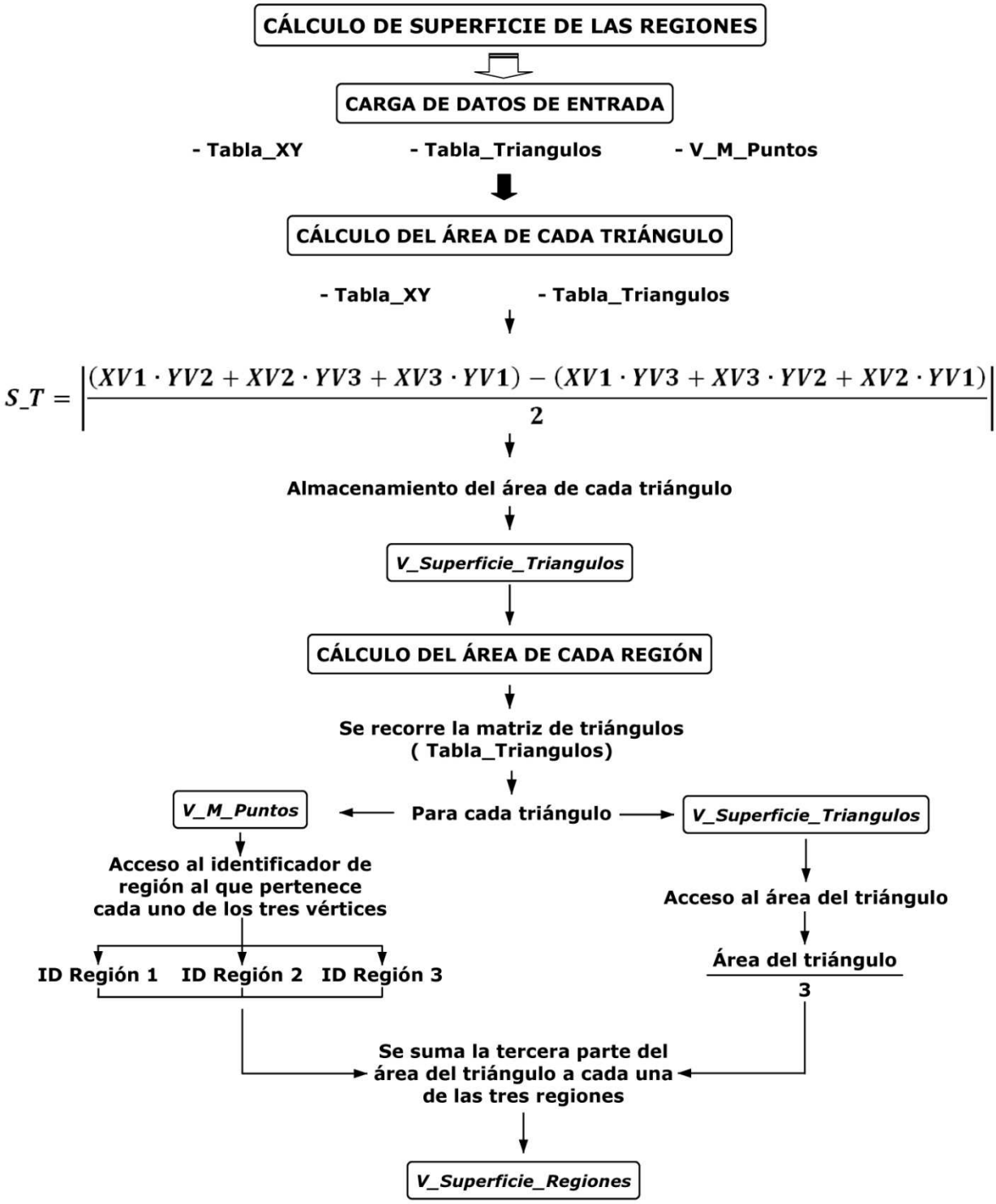


Figura 120. Proceso para el cálculo de áreas de las regiones. Diagrama de flujo.

La descripción detallada de los distintos procesos que realiza el algoritmo es la siguiente:

- **Carga de datos de entrada**

La aplicación implementada es una función que recibe los siguientes parámetros para su funcionamiento:

Tabla_XY: matriz de dos columnas con las coordenadas X,Y de la nube de puntos. Para cada punto, el número de registro se corresponde con el identificador del punto LiDAR.

Tabla_Triangulos: se trata de una matriz de tres columnas donde cada fila se corresponde con un identificador del triángulo de la triangulación. En las diferentes columnas se definen los tres vértices de cada triángulo, que se corresponderán con identificadores de puntos LiDAR de la nube de puntos.

V_M_Puntos: vector de igual dimensión que el número total de datos LiDAR. De forma que para cada punto LiDAR vendrá registrado el identificador de la agrupación o región a la que pertenece.

- **Cálculo de área de cada triángulo**

Una vez clasificados los triángulos en función de la región a la que pertenecen, el área de cada región puede obtenerse a partir del sumatorio del área de sus triángulos asociados. La aplicación calcula en primer lugar el área de cada triángulo y lo almacena en un vector de igual dimensión que el número total de triángulos existentes en la triangulación de Delaunay (*V_Superficie_Triangulos*). El área de cada triángulo se calcula a partir de la siguiente expresión:

$$S\_T = \left| \frac{(XV1 \cdot YV2 + XV2 \cdot YV3 + XV3 \cdot YV1) - (XV1 \cdot YV3 + XV3 \cdot YV2 + XV2 \cdot YV1)}{2} \right|$$

Donde:

- *S_T*: superficie del triángulo
- XV1,YV1: coordenadas X,Y del primer vértice del triángulo.
- XV2,YV2: coordenadas X,Y del segundo vértice del triángulo.
- XV3,YV3: coordenadas X,Y del tercer vértice del triángulo.

Para completar el vector *V_Superficie_Triangulos* el algoritmo recorre la matriz *Tabla_Triangulos* y almacena los identificadores de los tres vértices de cada triángulo. Seguidamente accede a las coordenadas de cada vértice mediante la matriz *Tabla_XY* y aplica la expresión anterior para calcular el área de cada uno de ellos. El área de cada triángulo se almacena en el registro correspondiente del vector *V_Superficie_Triangulos* hasta recorrer por completo la matriz de triángulos, momento en el cual el proceso finaliza.

- **Cálculo del área de cada región**

El área de cada región vendrá determinada por el sumatorio de las superficies (parciales o totales) de sus triángulos asociados. Para una región en concreto, la mayoría de sus triángulos asociados tendrán sus tres vértices que pertenecen a dicha región. Todo ello puede apreciarse en la Figura 121, donde los triángulos cuyos tres vértices pertenecen a una misma región están representados en colores diferentes en función de la región a la que pertenecen.

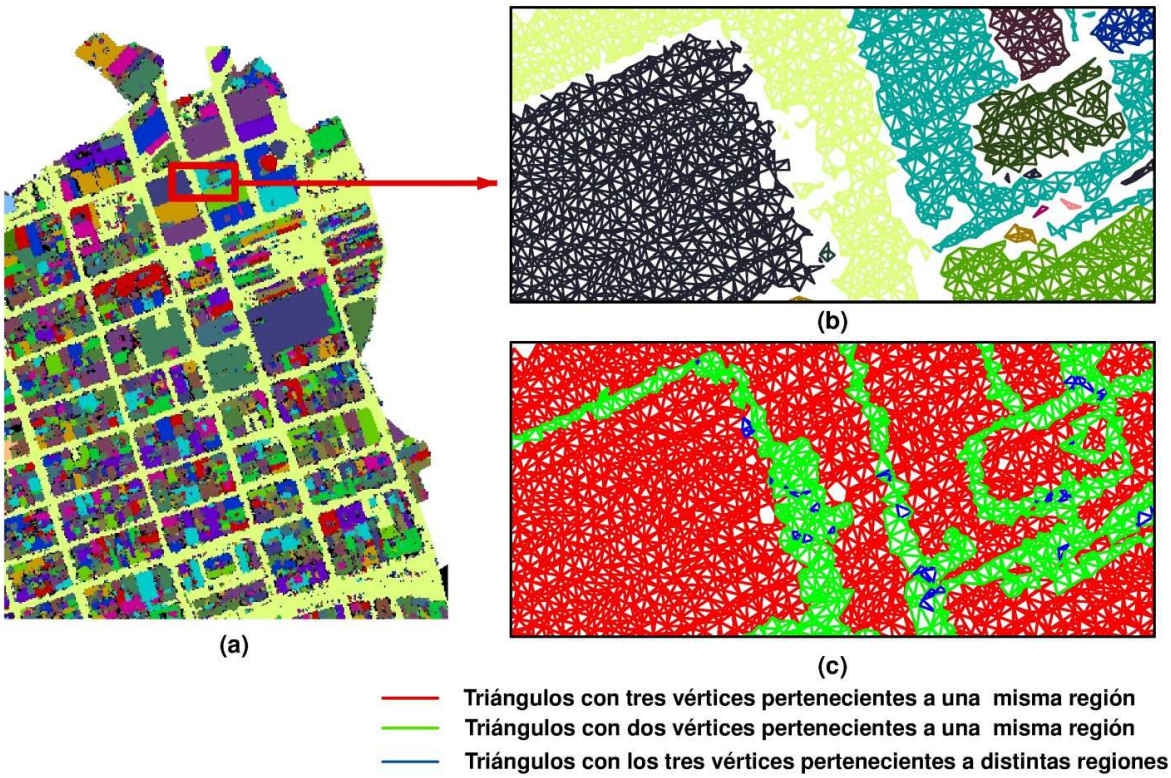


Figura 121. Triángulos clasificados en función de la región a la que pertenecen.

En cualquier caso, no todos los triángulos de la triangulación de Delaunay tienen sus tres vértices pertenecientes a la misma región. Tal y como se aprecia en la imagen de detalle (Figura 121c) existirán triángulos que tienen únicamente dos vértices pertenecientes a una misma región e incluso triángulos cuyos vértices pertenecen a tres regiones distintas. Con el fin de realizar una partición completa del espacio y que no quede superficie sin asignar a ninguna región (ver huecos de la Figura 121b), además de realizar el sumatorio del área de los triángulos cuyos tres vértices pertenecen a una misma región, se añadirá el área proporcional de cada triángulo en los casos restantes. Para ello, el primer paso consiste en definir un vector de igual dimensión que el número total de regiones donde se almacenará el área de cada región (*V_Superficie_Regiones*). Este vector será reiniciado con el valor cero en todos sus registros. Posteriormente bastará con recorrer la matriz de triángulos (Tabla_Triangulos) y acceder al identificador de la región a la que pertenecen los vértices de cada uno de los triángulos mediante el vector *V_M_Puntos*. Para cada triángulo se accederá a su área contenida en el registro correspondiente del vector *V_Superficie_Triangulos* y se dividirá en tres partes, de forma que cada una de las tres partes se sumará al registro correspondiente a la región de cada vértice en el vector *V_Superficie_Regiones*. Con este procedimiento, una vez recorrida la matriz de triángulos el vector de superficies habrá sido completado consiguiendo así una aproximación precisa y rápida del área de cada región.

- **Valor de retorno**

V_Superficie_Regiones: vector de igual dimensión que el número total de regiones obtenidas tras segmentar la escena donde para cada región vendrá registrada la superficie de la misma.

IV. ANÁLISIS DE LOS RESULTADOS Y DISCUSIÓN

Con el fin de analizar de forma precisa los resultados obtenidos tras ejecutar cada uno de los algoritmos desarrollados, se ha generado una clasificación precisa, donde los diferentes puntos LiDAR han sido clasificados en función de la entidad real a la que pertenecen de forma manual. En lo sucesivo nos referiremos a esta clasificación como "clasificación verdadera". La clasificación verdadera se ha realizado para la totalidad de los datos test 1 y 2, de tal forma que no se utilizarán muestras para analizar los resultados, sino la totalidad de puntos LiDAR que conforman la escena.

Para obtener la clasificación verdadera en los datos test 1 (Vall d'Uixó) y datos test 2 (Collado Villalba) se ha seguido un procedimiento similar con diferencias puntuales. Para los datos test 1 (Vall d'Uixó), la zona de estudio fue dividida en bloques cuadrados de 150 m de lado. Posteriormente cada bloque fue visualizado en 3D mediante Autocad y una primera clasificación preliminar se llevó a cabo. Seguidamente, la totalidad de las celdas se visualizaron en 3D junto con las ortofotos correspondientes proyectadas sobre el modelo utilizando el software ArcScene incluido en ArcGis 9.3. Esta segunda fase fue de gran utilidad para corregir algunos errores existentes en la clasificación preliminar. Finalmente, tras realizar la revisión y las correcciones correspondientes, las distintas clasificaciones definitivas correspondientes a cada bloque fueron fusionadas en un único archivo y exportadas a un fichero de texto.

La clasificación verdadera para los datos test 2 (Collado Villalba) se obtuvo de forma análoga, la diferencia radica en que como punto de partida se disponía de una clasificación preliminar realizada con el sofware TerraScan facilitada por la empresa que proporcionó los datos.

Dado que se opta por no utilizar muestras sino la totalidad de puntos LiDAR disponibles, ha sido necesario implementar una aplicación auxiliar para analizar los resultados. Ello es debido a que las aplicaciones disponibles tales como ArcGis o GVSig, no están diseñadas para analizar clasificaciones sobre datos puntuales masivos y por tanto, la generación de matrices de confusión así como la impresión de los errores de omisión y comisión conlleva la ejecución de numerosos análisis asociados a un tiempo de cómputo considerable, hasta el punto, de considerarse inviable.

La aplicación implementada que analiza los resultados para cada una de las clasificaciones llevadas a cabo por los diferentes algoritmos, se basa en generar un identificador único para cada punto LiDAR mediante la concatenación de sus coordenadas, de tal forma que el mismo punto dentro de la clasificación verdadera así como en la clasificación realizada por los algoritmos propuestos tiene el mismo identificador. A la hora de comparar la clasificación verdadera con las clasificaciones obtenidas en los distintos procesos desarrollados basta con indexar ambas clasificaciones a partir del identificador común y comparar los valores obtenidos en cada clasificación.

Para evaluar las clasificaciones obtenidas tras ejecutar cada uno de los algoritmos, se cuantifican los errores de comisión y omisión de cada clase. Además, dado que la mayor precisión se obtiene cuando ambos errores son mínimos, un indicador útil será la media de los errores obtenidos (error de omisión y comisión). Además, también se tendrá en cuenta la exactitud global de cada una de las clasificaciones realizadas.

Las matrices de confusión y los gráficos asociados para la totalidad de los resultados obtenidos tras aplicar los algoritmos con distintos parámetros umbrales aparecen en el Anexo I (Apartado VII). En el presente apartado se detallan los aspectos de mayor relevancia del análisis realizado.

IV. 1. Detección de valores atípicos (errores groseros)

Los algoritmos desarrollados para detectar anomalías en los datos LiDAR se basan en dos características fundamentales de los puntos con altitud atípica. Por un lado el hecho de que estos puntos se presentan como una entidad puntual aislada o como una pequeña agrupación. Por otro, la propia anomalía que posee en cuanto a altitud que lo diferencia de su entorno más próximo.

El diseño del primero de los algoritmos desarrollados (Algoritmo A y B) se fundamenta en el análisis de las dos características anteriormente mencionadas. Para ello, tras segmentar la escena en superficies continuas, se localizan regiones de pequeño tamaño susceptibles de contener anomalías y se analizan las variaciones en desnivel de forma local.

El principal problema reside en ajustar los diferentes parámetros umbrales para identificar dos tipos de errores groseros: aquellos cuya cota difiere en gran medida con respeto al valor esperado y los que tienen una altitud próxima al terreno.

La detección de grandes anomalías en altitud propias de errores en el sistema de medición o a la reflexión del pulso láser sobre aves, precisa de un umbral en desnivel alto (*Umbral_IZ*) y un umbral en porcentaje de desnivel relajado (*Umbral_Porcentajes*), necesario para cuantificar las variaciones en desnivel dentro del vecindario próximo. Por el contrario, para detectar valores atípicos en altitud cuyo valor no difiere en exceso de lo esperado, el umbral en desnivel debe de ser menor y el umbral en porcentaje mayor. Consecuentemente, deben de ajustarse por separado los parámetros para detectar los dos tipos de errores groseros.

Dado que la detección de valores atípicos es el primero de los procesos que debe de llevarse a cabo en el tratamiento de datos LiDAR, tras ejecutar el proceso se obtienen únicamente dos clases: valores atípicos y no atípicos.

Para detectar valores atípicos con una altitud que difiere en gran medida con respecto al valor esperado (algoritmo A) debe de utilizarse un umbral en desnivel alto (*Umbral_IZ*). En este sentido, tras realizar diferentes pruebas experimentales y analizar los resultados de forma cualitativa (análisis visual) se concluye que un desnivel de 10 m es suficiente. En cuanto al tamaño de región máxima para considerar una agrupación como susceptible de contener valores atípicos se decide utilizar un tamaño de 15 m^2 por no ser un parámetro excesivamente restrictivo, pero que permite reducir el tiempo de cómputo notablemente.

Tras probar el algoritmo y analizar los primeros resultados de manera visual, se observa que la precisión obtenida tiene una alta dependencia con respecto al umbral en porcentaje utilizado en desniveles de forma local. Por tanto, para determinar si existe una alta variabilidad en cuanto a la altitud del punto objeto de análisis con respecto a su vecindario se testean distintos valores umbrales que oscilan desde el 70% al 99%.

Los distintos umbrales en porcentaje aplicados para analizar los datos test 1 (Vall d'Uixó) así como los resultados obtenidos se muestran en la Figura 122. Los mayores errores de comisión se obtienen cuando el parámetro en porcentaje (*Umbral_Porcentajes*) es menor (70%). No obstante, este error se reduce al incrementar el umbral hasta que desaparece al utilizar un porcentaje del 85% (A 3). Llegado a este punto, si el umbral en porcentaje sigue aumentando no se introducen errores por comisión pero se incrementa notablemente el error de omisión, alcanzando su máximo cuando el parámetro umbral en porcentaje es del 99% (A 6). El objetivo del análisis es determinar cuál es el parámetro umbral en porcentaje óptimo para detectar este tipo de anomalías, que será aquel que minimiza ambos tipos de errores (errores de comisión y de omisión). En este sentido, un parámetro umbral en porcentaje del 85% (A3) es el que posee una media de errores mínima y el que mejor exactitud global consigue en la clasificación (Figura 122).

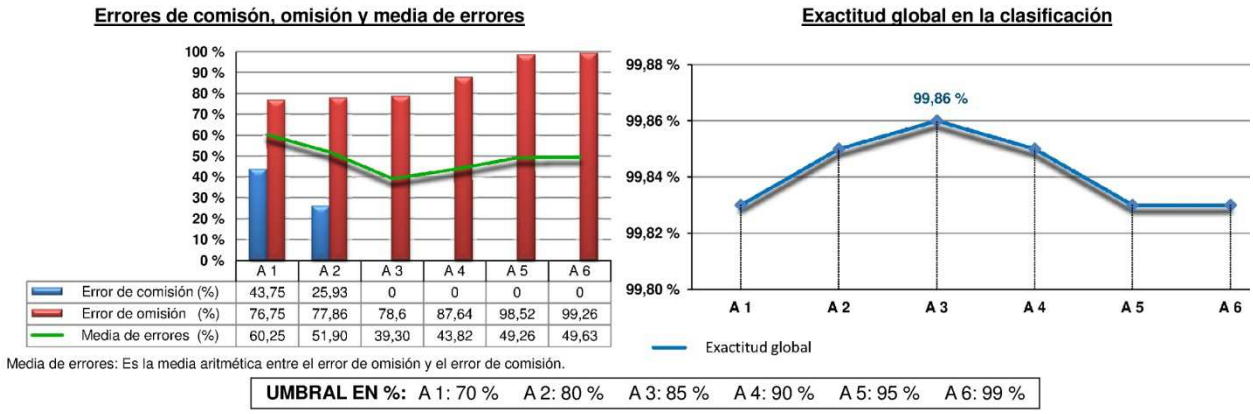


Figura 122. Gráfico de errores y precisión alcanzada en la detección de errores groseros. Algoritmo A. Datos test 1 (Vall d'Uixó).

Tras ejecutar el algoritmo para detectar puntos anómalos con altitud muy diferente a la esperada (algoritmo A) sobre los datos test 2 (Collado Villalba), los resultados son análogos a los obtenidos con los datos test 1 (Vall d'Uixó). Básicamente, se produce un error de comisión considerable cuando el parámetro umbral en porcentaje es menor al 70%, que disminuye al aumentar el parámetro umbral. Por el contrario, a medida que el umbral en porcentaje aumenta también lo hace el error de omisión. Igual que ocurría en los datos test 1, los mejores resultados se obtienen con un parámetro umbral del 85% (A 3), ya que en este caso, la media de los errores de comisión y de omisión es mínima y la exactitud global de la clasificación máxima (Figura 123).

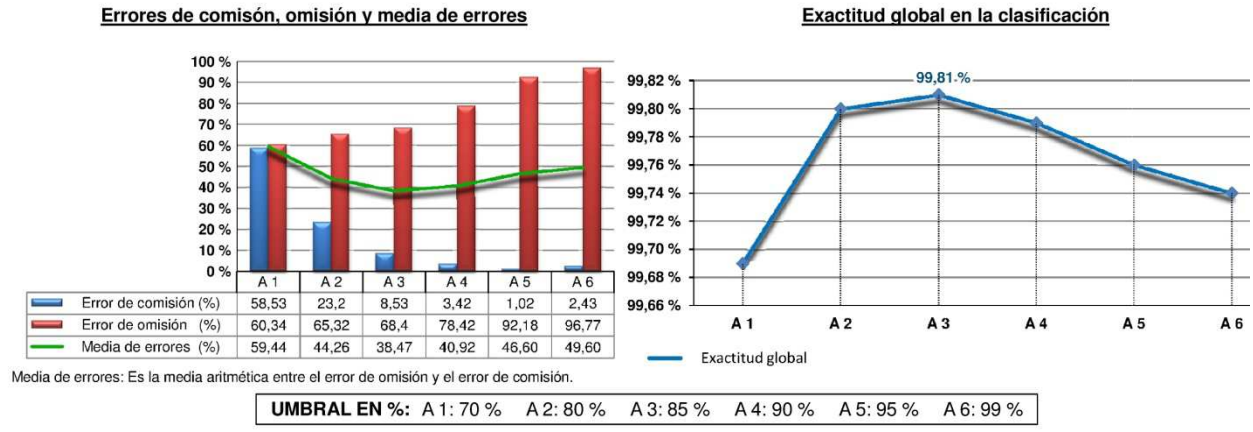


Figura 123. Gráfico de errores y precisión alcanzada en la detección de errores groseros. Algoritmo A. Datos test 2 (Collado Villalba).

La ejecución del *algoritmo A* tiene como finalidad principal detectar anomalías ocasionadas por errores producidos en el sistema de medición láser que se corresponden con puntos de altitud muy distinta al resto de datos que conforman la escena. Los datos test 1 (Vall d'Uixó) no tienen este tipo de errores groseros, por lo que únicamente se detectarán algunos valores atípicos que se corresponden con estructuras artificiales como grúas o líneas eléctricas. En la Figura 124 se muestran las anomalías detectadas tras utilizar tres parámetros umbrales en desnivel (A1, A3 y A6) así como los errores ocasionados. Un parámetro umbral en porcentajes pequeño (A 1) produce errores de comisión que se corresponden con pequeñas regiones asociadas principalmente a puntos de fachada o a árboles. Por el contario, un umbral excesivamente elevado (A 6) elimina por completo los errores de comisión, pero detecta un escaso número de valores atípicos. El parámetro umbral que mejores resultados obtiene es un valor medio (A3), donde no existen errores de comisión, pero sí se detectan algunas regiones que contienen errores groseros asociados a grúas y líneas eléctricas.

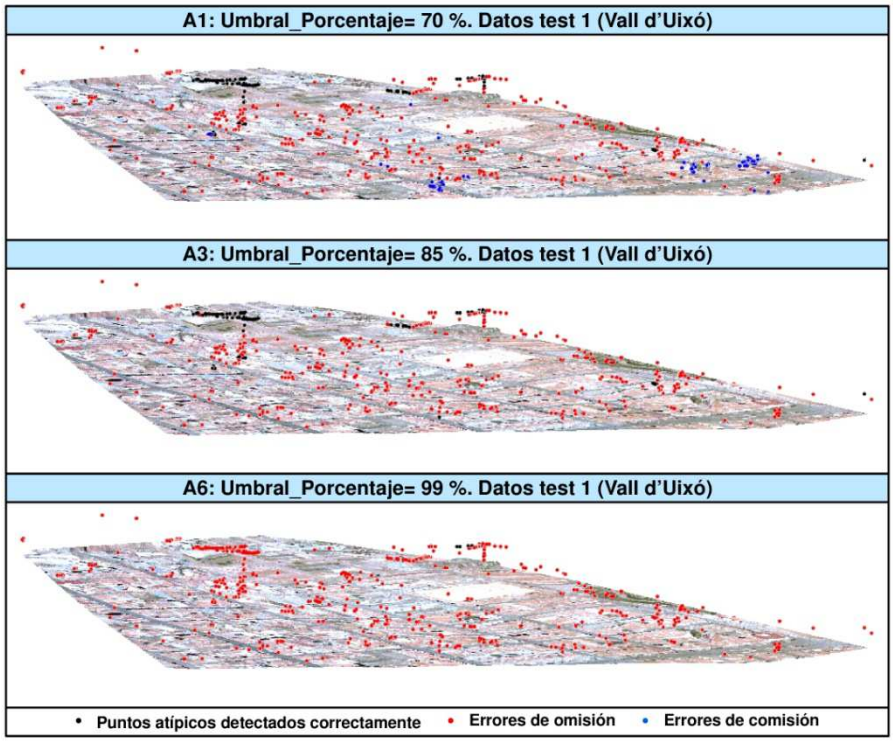


Figura 124. Imagen 3D de precisión alcanzada en la detección de errores groseros. Algoritmo A. Datos test 1 (Vall d'Uixó).

En los datos test 2 (Collado Villalba) si existen numerosos valores atípicos que se corresponden con errores del sistema de medición cuya altitud difiere en gran medida con respecto al valor esperado (Figura 125a). Dado que para estos puntos la altura es muy diferente a la de su vecindario, son detectados sin problemas independientemente del umbral en porcentaje utilizado. Con respecto al resto de valores atípicos asociados a estructuras como grúas o líneas eléctricas, al igual que en los datos test 1, un parámetro umbral relajado (70%) en cuanto a porcentaje ocasiona numerosos errores por exceso (Figura 125c). Por el contrario, un umbral demasiado restrictivo (99%) aumenta considerablemente el error de omisión (Figura 125d). Al igual que sucedía en los datos test 1, el porcentaje que consigue minimizar ambos errores (comisión y omisión) es del 85% (Figura 125a y Figura 125b). En este caso la totalidad de valores atípicos asociados a errores de medición del sistema son detectados, además también se clasifican como errores groseros una parte importante de puntos pertenecientes a grúas y líneas eléctricas.

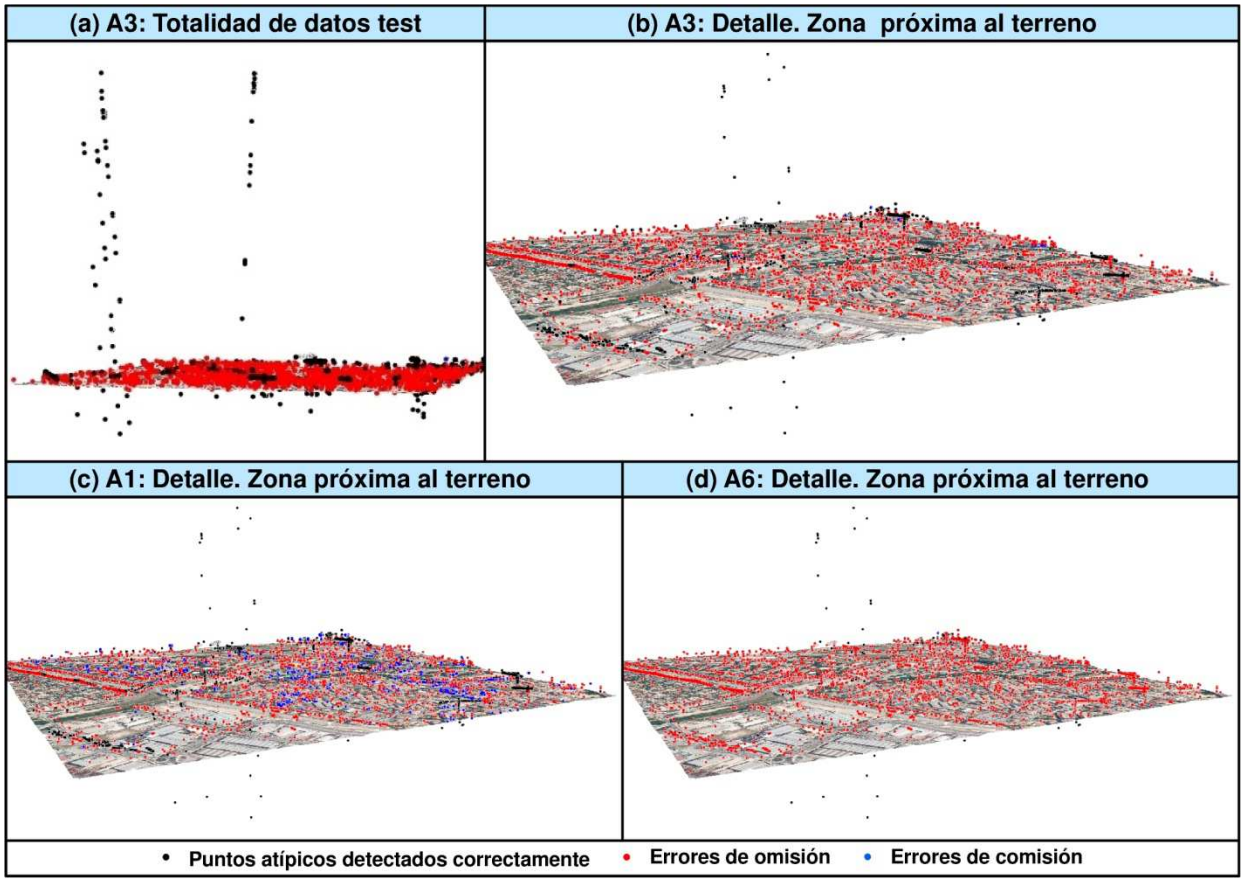


Figura 125. Imagen 3D de precisión alcanzada en la detección de errores groseros. Algoritmo A. Datos test 2 (Collado Villalba).

La utilización de un umbral en desnivel de 10 m asegura la detección de la mayoría de los valores atípicos ocasionados por errores en el sistema de medición, así como los debidos a reflexiones del pulso láser sobre aves. Además, también detecta otras regiones pertenecientes a estructuras artificiales de gran altura. Con su ejecución, se asegura la detección de valores atípicos cuya altitud difiere en gran medida con respecto a la esperada. No obstante, la gran mayoría de errores groseros asociados a estructuras artificiales (grúas, líneas eléctricas, farolas, señales de tráfico...), así como a errores de medición que ocasionan valores atípicos próximos al terreno, no son detectados. Para solucionar este problema, el parámetro umbral en desnivel debe de ser menor (Algoritmo B).

Los parámetros utilizados para detectar errores groseros próximos al terreno (Algoritmo B) son los mismos que los utilizados en la detección de valores atípicos de gran altitud (Algoritmo A), exceptuando el parámetro umbral en desnivel (*Umbral_IZ*) que debe de ser menor para poder detectar valores atípicos próximos al terreno. En este sentido, tras realizar distintas pruebas se opta por reducir el parámetro a 3 m.

De nuevo, el principal problema reside en la dependencia de la precisión de los resultados con respecto al parámetro umbral en porcentajes necesario para analizar la variabilidad en altura del posible valor atípico con respecto a su vecindario más próximo. Consecuentemente, con la finalidad de determinar el parámetro que mejores resultados obtiene en la detección de este tipo de valores atípicos en entornos urbanos, se comprueban distintos parámetros que oscilan entre el 70% y el 99%.

Los diferentes umbrales en porcentaje testeados para los datos test 1 (Vall d'Uixó), así como los resultados obtenidos se muestran en la Figura 126. Los mayores errores de

comisión se producen al utilizar un umbral poco restrictivo (70%). Sin embargo, este error se reduce al aumentar el parámetro umbral hasta desaparecer por completo al utilizar un porcentaje del 95% (B 5). El error por omisión tiene un comportamiento contrario al de comisión, de forma que aumenta al incrementar el umbral en porcentaje. El umbral que minimiza en mayor grado ambos tipos de errores es del 95% (B 5), momento en el que la media de errores se reduce a 9,5% y la exactitud global de la clasificación alcanza el 99,97%.

Figura 126. Gráfico de errores y precisión alcanzada en la detección de errores groseros. Algoritmo B. Datos test 1(Vall d'Uixó).

Los resultados obtenidos para los datos test 2 (Collado Villalba) son similares a los obtenidos en los datos test 1(Figura 127). En este caso, el error por comisión es máximo cuando el umbral en porcentaje es menor del 70% y se reduce a medida que aumenta el parámetro umbral. Por otro lado, el error de omisión sigue una tendencia contraria y aumenta al incrementarse el porcentaje umbral. Al igual que ocurría en los datos test 1, el parámetro umbral en porcentaje que mejores resultados obtiene es del 95% (B 5), donde se produce una media de errores del 15,6% y la exactitud global de la clasificación alcanza el 99,92%.

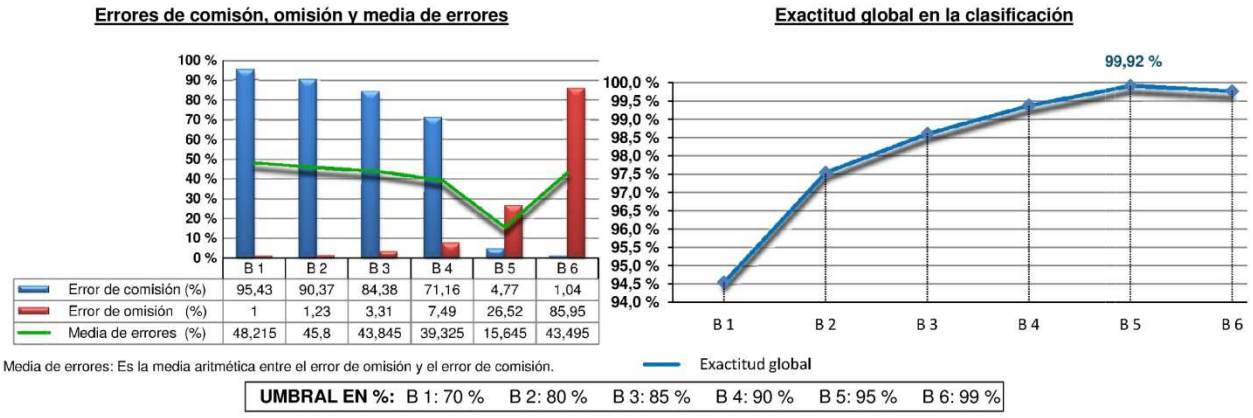


Figura 127. Gráfico de errores y precisión alcanzada en la detección de errores groseros. Algoritmo B. Datos test 2 (Collado Villalba).

El algoritmo B consigue detectar la mayoría de valores atípicos cuya altitud no difiere en exceso de la esperada. En la Figura 128 se muestra un modelo 3D para los datos test 1, donde aparecen las ortofotos proyectadas sobre el terreno, así como el resultado obtenido al utilizar tres parámetros umbrales en porcentaje distintos: 70%, 95% y 99%. La utilización de un parámetro umbral demasiado relajado (B 1) produce un error por comisión

muy alto, ocasionado fundamentalmente por la clasificación errónea de pequeñas regiones asociadas a árboles, puntos de fachada y pequeños planos de techo (Figura 128a). Por el contrario, un umbral en porcentaje muy restrictivo (99%) no consigue detectar prácticamente ningún valor atípico y ocasiona un gran número de errores por omisión (Figura 128c). El mejor resultado se obtiene para un umbral en porcentaje del 95%, donde la mayoría de los valores atípicos son detectados sin producirse errores de comisión (Figura 128b).

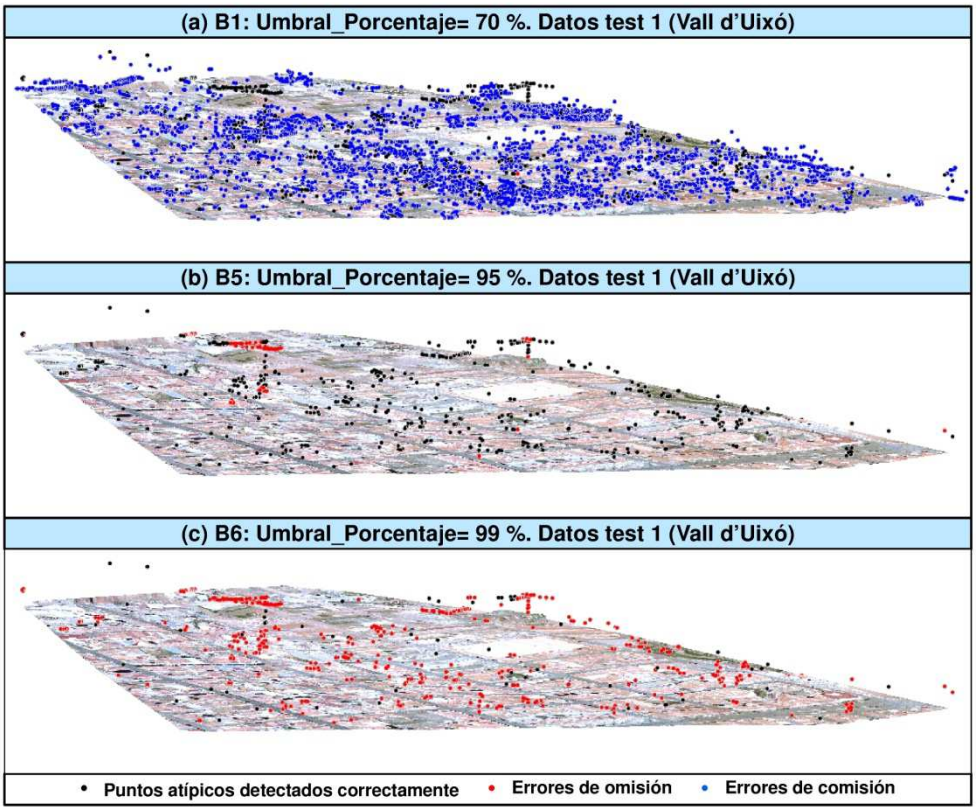


Figura 128. Imagen 3D de precisión alcanzada en la detección de errores groseros. Algoritmo A. Datos test 1 (Vall d'Uixó).

El algoritmo B se comporta para los datos test 2 (Collado Villalba) de igual forma que para los datos test 1. Un parámetro umbral relajado (70%) en cuanto a porcentaje ocasiona numerosos errores de comisión (Figura 129a) asociados a la clasificación errónea de pequeñas regiones pertenecientes a vegetación, planos de techo y puntos de fachada como errores groseros. Por el contrario, un umbral demasiado restrictivo (99%) incrementa el error de omisión (Figura 129c) de manera excesiva. Al igual que sucedía en los datos test 1, el umbral en porcentaje que minimiza ambos errores (comisión y omisión) es del 95%. Con este parámetro se consigue detectar la mayoría de errores groseros próximos al terreno tales como líneas eléctricas, grúas, farolas, señales de tráfico, etc. (Figura 129b).

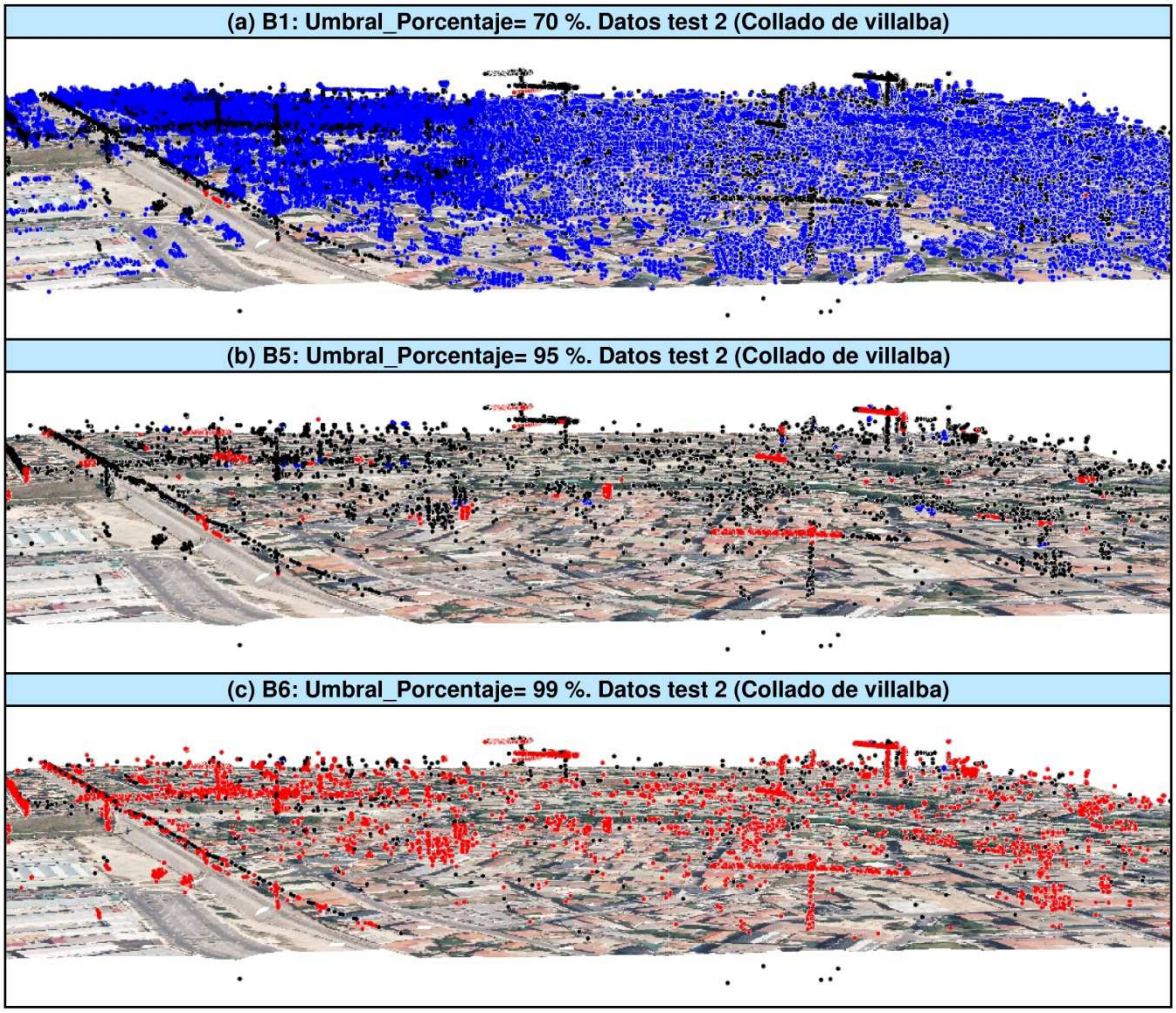


Figura 129. Imagen 3D de precisión alcanzada en la detección de errores groseros. Algoritmo A. Datos test 2 (Collado Villalba).

Hasta ahora mediante la ejecución del algoritmo A y B se han determinado los parámetros umbrales óptimos para identificar dos tipos de errores groseros: aquellos cuya altitud difiere en gran medida del valor esperado y los que tienen una altitud similar al resto de puntos LiDAR. El principal problema es que ambos tipos de valores atípicos no pueden detectarse correctamente a partir de una única ejecución del algoritmo, ya que se precisa de diferentes parámetros umbrales. Para solucionar este aspecto se implementa una nueva aplicación (algoritmo C) que ejecuta el proceso dos veces de forma concatenada.

Los parámetros utilizados en el primer proceso están definidos para detectar puntos de cota anómala muy diferentes al terreno, tales como errores asociados a ruido del sensor o pulsos reflejados sobre aves (algoritmo A). Los parámetros umbrales del segundo proceso se ajustan para detectar puntos de cota anómala próximos al terreno, como pulsos reflejados sobre personas, señales de tráfico, etc. (algoritmo B). Una vez ejecutados ambos procesos se fusionan los resultados obteniendo así la clasificación definitiva de valores atípicos.

Tras analizar los resultados obtenidos en la detección de los dos tipos de errores groseros, se concluye que los parámetros que mejores resultados ofrecen coinciden para ambas zonas test. El conjunto de los datos test utilizados abarca entornos urbanos que incluyen características muy distintas (zonas industriales, zonas residenciales con grandes bloques de edificios, viviendas unifamiliares, zonas arboladas, vegetación densa junto a edificaciones, existencia de puentes, canales, autovías próximas al casco urbano, etc.). Por

tanto, cabe esperar que la precisión de los resultados obtenidos sea muy similar a la obtenida tras ejecutar la aplicación en otros entornos urbanos. Tal y como se ha visto con anterioridad, los mejores resultados para identificar valores atípicos cuya altitud difiere en exceso con respecto al valor esperado se obtiene utilizando un parámetro umbral en desnivel de 10 m y un umbral en porcentaje del 85% (A 3). Por el contrario, para detectar de forma óptima errores groseros próximos al terreno es necesario reducir el umbral en desnivel a 3 m y aumentar el porcentaje umbral al 95% (B 5).

Cabe esperar que los mejores resultados obtenidos tras ejecutar el algoritmo multiproceso (algoritmo C) se obtengan al concatenar el algoritmo A y B con los parámetros anteriormente indicados. No obstante, para conseguir un análisis más preciso se ha ejecutado el algoritmo un total de nueve veces con los siguientes parámetros umbrales (Figura 130):

UMBRAL EN IZ: PROCESO 1: 10 m PROCESO 2: 3 m

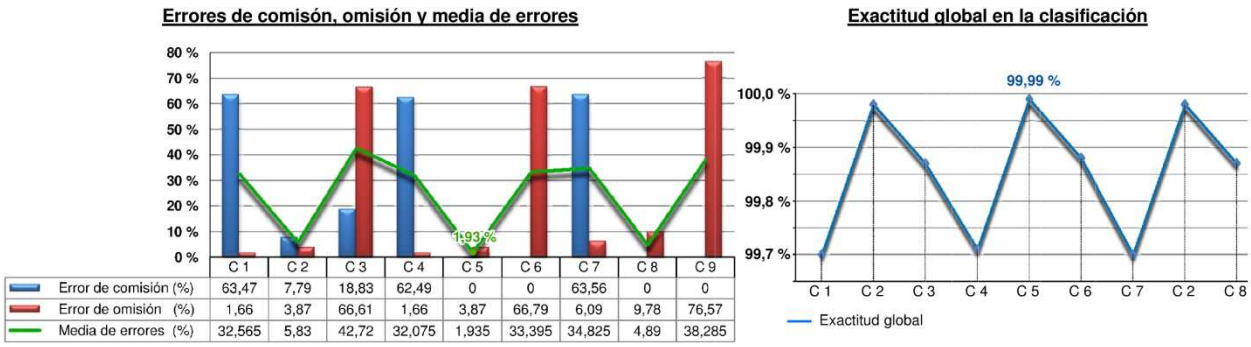
UMBRAL EN %: Se ejecuta el algortimo nueve veces con diferentes parámetros umbrales:

C 1: PROCESO 1: 80 % PROCESO 2: 90 %	**C 2:** PROCESO 1: 80 % PROCESO 2: 95 %	**C 3:** PROCESO 1: 80 % PROCESO 2: 99 %	**C 4:** PROCESO 1: 85 % PROCESO 2: 90 %	**C 5:** PROCESO 1: 85 % PROCESO 2: 95 %
C 6: PROCESO 1: 85 % PROCESO 2: 99 %	**C 7:** PROCESO 1: 90 % PROCESO 2: 90 %	**C 8:** PROCESO 1: 90 % PROCESO 2: 95 %	**C 9:** PROCESO 1: 90 % PROCESO 2: 99 %	

Figura 130. Detección de errores groseros. Parámetros umbrales utilizados. Algoritmo C.

Donde el proceso 1 se corresponde con la detección de valores muy atípicos (algoritmo A) y el proceso 2 con errores groseros próximos al terreno (algoritmo B).

Los resultados obtenidos tras ejecutar el algoritmo C para los datos test 1 (Vall d'Uixó) se muestran en la Figura 131. Tal y como era de esperar, la mayor precisión se alcanza al utilizar los parámetros umbrales que mejores resultados obtienen para identificar los dos tipos de valores atípicos mediante la ejecución de un único proceso (algoritmo A y B); en este caso (C 5), el primer proceso (algoritmo A) detecta valores atípicos con una altitud que difiere en exceso a la esperada utilizando un umbral en desnivel de 10 m y un porcentaje umbral del 85% (A 3). Seguidamente, la ejecución del segundo proceso consigue completar la detección de outliers mediante la identificación de valores atípicos próximos al terreno (algoritmo B); en este caso el umbral en desnivel es de 3 m y el porcentaje umbral del 95% (B 5).



	C 1	C 2	C 3	C 4	C 5	C 6	C 7	C 8	C 9
Error de comisión (%)	63,47	7,79	18,83	62,49	0	0	63,56	0	0
Error de omisión (%)	1,66	3,87	66,61	1,66	3,87	66,79	6,09	9,78	76,57
Media de errores (%)	32,565	5,83	42,72	32,075	1,935	33,395	34,825	4,89	38,285

Figura 131. Gráfico de errores y precisión alcanzada en la detección de errores groseros. Algoritmo C. Datos test 1 (Vall d'Uixó).

La ejecución del algoritmo multiproceso (algoritmo C) sobre los datos test 2 consigue unas precisiones similares a las obtenidas en los datos test 1 (Figura 132). En este caso también

son mayores los logros alcanzados al utilizar los parámetros umbrales que mejor funcionan individualmente (C 5), consiguiendo reducir la media de errores a tan solo un 6,27% y proporcionando una exactitud global de la clasificación del 99,97%.

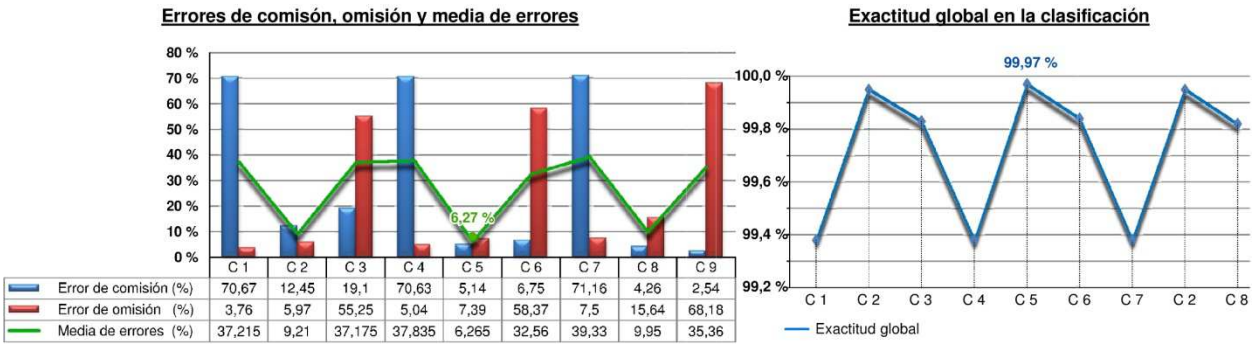


Figura 132. Gráfico de errores y precisión alcanzada en la detección de errores groseros. Algoritmo C. Datos test 2 (Collado Villalba).

Básicamente, el algoritmo C combina dos procesos para detectar los diferentes tipos de valores atípicos minimizando los errores de comisión y omisión. En los datos test 1 (Vall d'Uixó) no existen valores atípicos ocasionados por errores de medición en el sistema, por lo que la ejecución del primer proceso detecta principalmente algunos errores groseros asociados a grúas y líneas eléctricas y produce un alto error de omisión (Figura 133a). La aplicación segundo proceso consigue reducir estos errores notablemente al estar diseñado para detectar puntos con valores atípicos cuya altitud no difiere en exceso del terreno (Figura 133b). Los mejores resultados se consiguen al fusionar los resultados obtenidos tras ejecutar ambos procesos, obteniendo así una clasificación definitiva que minimiza los errores producidos (Figura 133c).

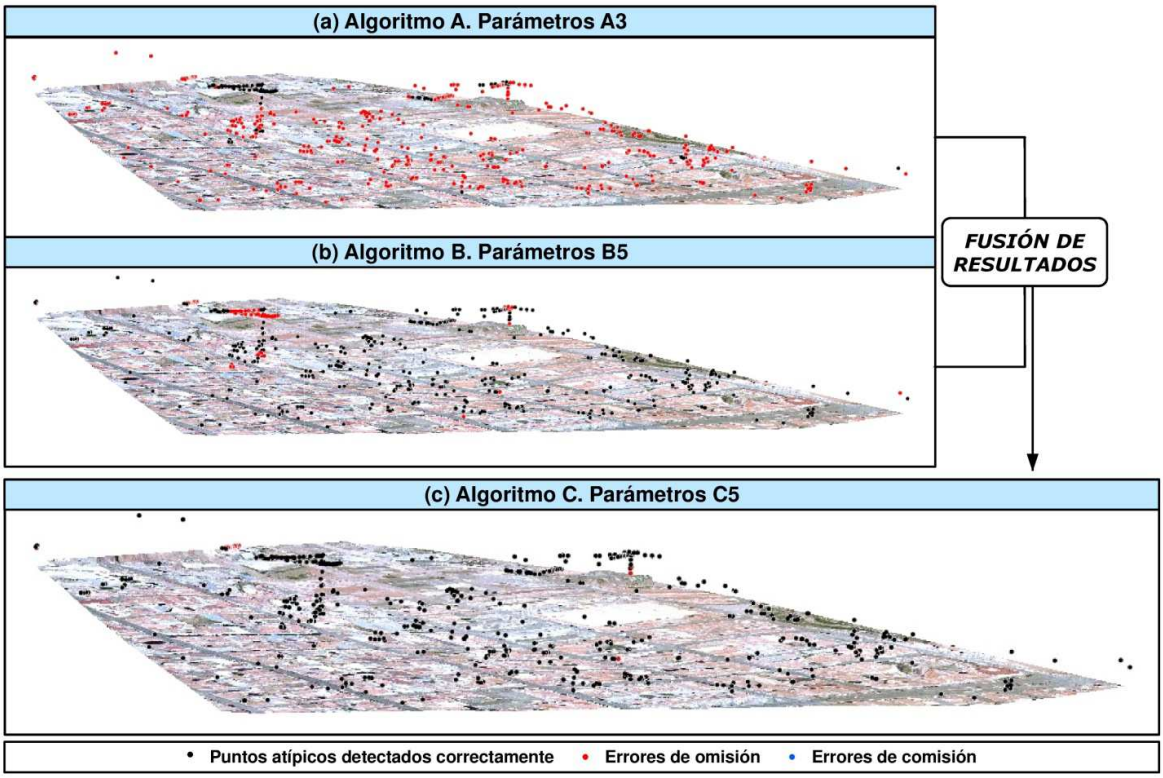


Figura 133. Imagen 3D de precisión alcanzada en la detección de errores groseros. Algoritmo C. Datos test 1 (Vall d'Uixó).

En los datos test 2 (Collado Villalba) existen valores atípicos ocasionados por errores de medición en el sistema que ocasionan errores groseros con una altitud muy diferente a la esperada. Este tipo de anomalías junto con algunos valores atípicos asociados a altas estructuras artificiales son detectados con precisión al ejecutar el primer proceso (Figura 134a). El segundo proceso consigue detectar los valores atípicos próximos al terreno no detectados con anterioridad (Figura 134b), consiguiendo reducir en gran medida los errores por omisión producidos por el primer proceso. Finalmente, la fusión de los resultados obtenidos consigue alcanzar los resultados más precisos (Figura 134c).

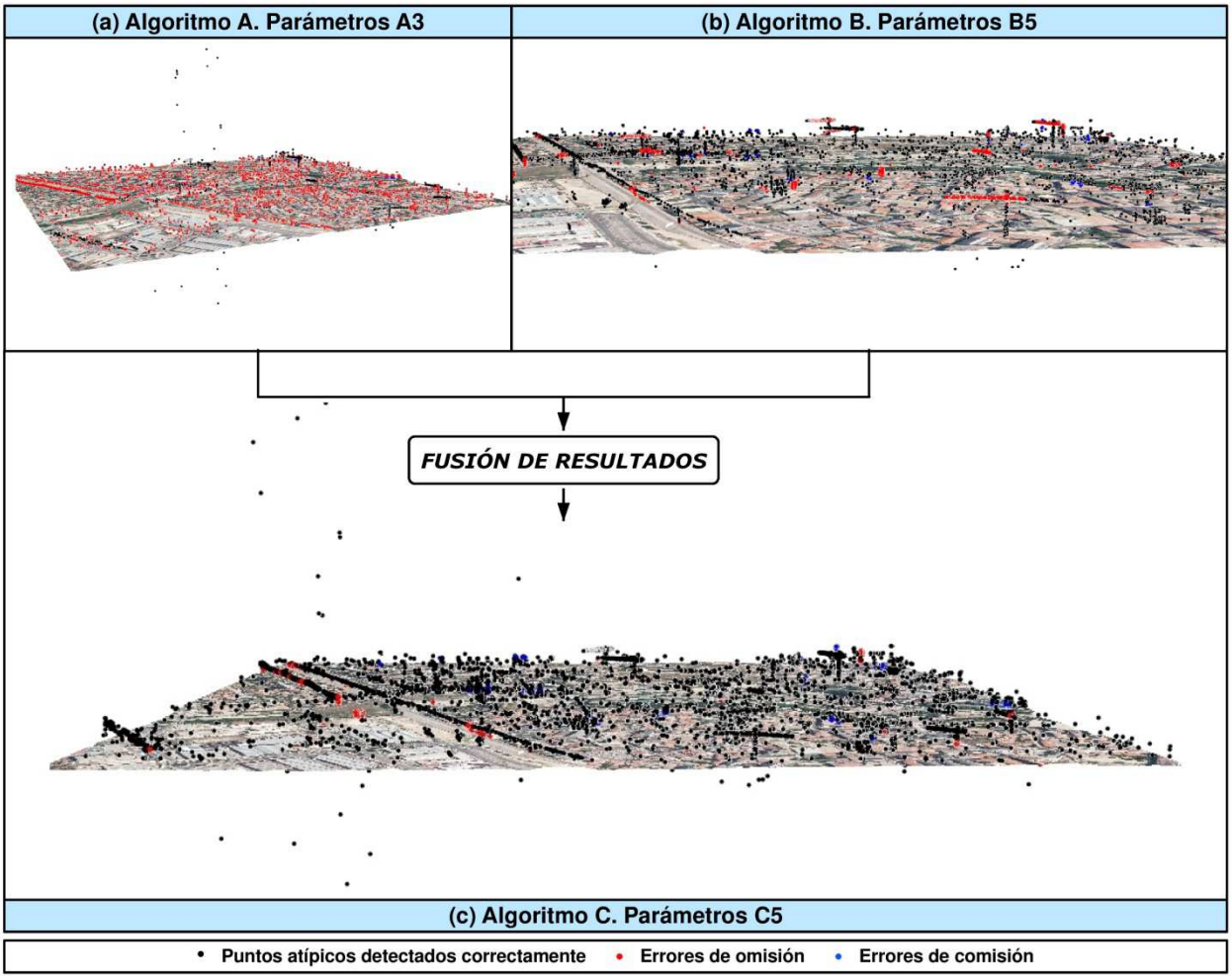


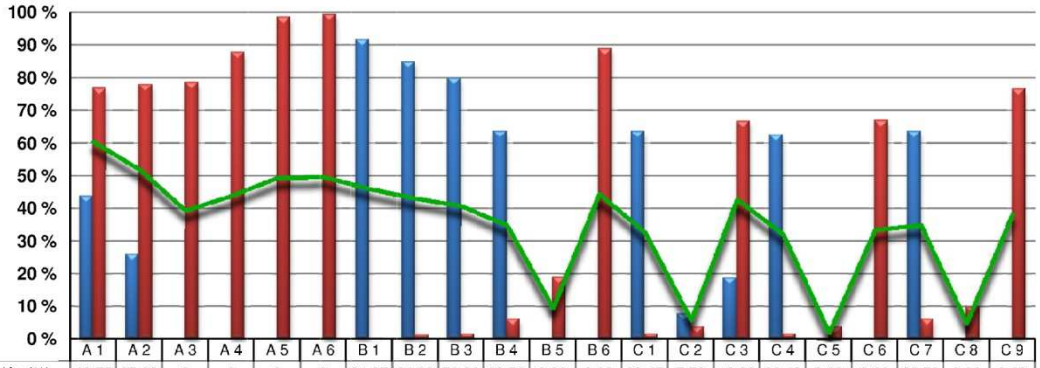
Figura 134. Imagen 3D de precisión alcanzada en la detección de errores groseros. Algoritmo C. Datos test 1 (Vall d'Uixó).

En la Figura 135 se muestran los errores asociados a los tres algoritmos ejecutados (A, B y C) para los datos test 1 y datos test 2. A la vista de los resultados, puede afirmarse que existe una alta correlación entre los errores producidos en ambas zonas test para los distintos parámetros umbrales utilizados. Dado que los datos test pertenecen a distintos municipios con características muy diversas, la alta correlación indica que el algoritmo tiene un comportamiento similar independientemente del entorno urbano objeto de análisis, lo que indica que estamos frente a un algoritmo robusto capaz de ofrecer soluciones precisas con independencia de las características de los datos utilizados.

Con respecto a los precisión de los resultados obtenidos, el primer proceso (Algoritmo A) ocasiona un alto error de omisión independientemente del parámetro umbral en porcentaje utilizado. Ésto es debido a que la aplicación está diseñada para detectar principalmente

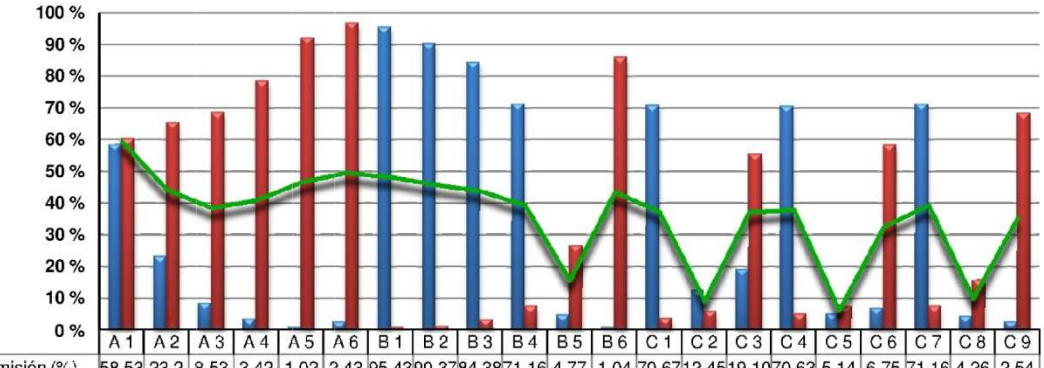
puntos con valores atípicos muy diferentes al valor esperado (errores de medición) y teniendo en cuenta que el número de casos asociados a este tipo de errores groseros es reducido, la mayor parte de los valores atípicos no son identificados. Por el contrario, el segundo proceso ocasiona errores de comisión importantes si los parámetros umbrales no se ajustan adecuadamente. Ello se debe a que este segundo proceso está diseñado para detectar puntos de altitud atípica próximos al terreno, por lo que el uso de un parámetro umbral en porcentajes poco restrictivo ocasiona errores por exceso asociados a pequeñas regiones que pertenecen a vegetación o planos de techo. Los mejores resultados se obtienen al fusionar ambos procesos dentro de un mismo análisis (Algoritmo C), ya que con ello se consiguen detectar tanto los valores atípicos cuya altura difiere en exceso del valor esperado como aquellos que están próximos al terreno. En concreto, los mejores resultados se obtienen al utilizar los parámetros que mayor precisión alcanzan en la ejecución de procesos individuales (A 3 y B 5), consiguiendo minimizar el error de comisión y omisión considerablemente.

DATOS TEST 1 (Vall d'Uixó)



	A 1	A 2	A 3	A 4	A 5	A 6	B 1	B 2	B 3	B 4	B 5	B 6	C 1	C 2	C 3	C 4	C 5	C 6	C 7	C 8	C 9
Error de comisión (%)	43,75	25,93	0	0	0	0	91,67	84,92	79,89	63,56	0,00	0,00	63,47	7,79	18,83	62,49	0,00	0,00	63,56	0,00	0,00
Error de omisión (%)	76,75	77,86	78,6	87,64	98,52	99,26	0,18	1,11	1,48	6,09	19,00	88,75	1,66	3,87	66,61	1,66	3,87	66,79	6,09	9,78	76,57
Media de errores (%)	60,25	51,90	39,30	43,82	49,26	49,63	45,93	43,02	40,69	34,83	9,50	44,38	32,57	5,83	42,72	32,08	1,94	33,40	34,83	4,89	38,29

DATOS TEST 2 (Collado de villalba)



	A 1	A 2	A 3	A 4	A 5	A 6	B 1	B 2	B 3	B 4	B 5	B 6	C 1	C 2	C 3	C 4	C 5	C 6	C 7	C 8	C 9
Error de comisión (%)	58,53	23,2	8,53	3,42	1,02	2,43	95,43	90,37	84,38	71,16	4,77	1,04	70,67	12,45	19,10	70,63	5,14	6,75	71,16	4,26	2,54
Error de omisión (%)	60,34	65,32	68,4	78,42	92,18	96,77	1,00	1,23	3,31	7,49	26,52	85,95	3,76	5,97	55,25	5,04	7,39	58,37	7,50	15,64	68,18
Media de errores (%)	59,44	44,26	38,47	40,92	46,60	49,60	48,22	45,80	43,85	39,33	15,65	43,50	37,22	9,21	37,18	37,84	6,27	32,56	39,33	9,95	35,36

Media de errores: Es la media aritmética entre el error de omisión y el error de comisión.

Figura 135. Gráfico de errores en la detección de errores groseros. Algoritmos A, B y C. Datos test 1 y 2.

La ejecución del algoritmo multiproceso (Algoritmo C) que concatena aquellos procesos individuales que alcanzan mejores resultados para detectar los dos tipos de valores atípicos considerados (A 3 y B 5) es el que mejores resultados obtiene (C 5). En este caso, para los datos test 1, el error por omisión se reduce hasta desaparecer y el de comisión disminuye

hasta el 3,87%. Además, la media de errores se minimiza hasta conseguir un valor del 1,94% y la exactitud global de la clasificación alcanza el 99,99%. Respecto a los datos test 2, el error de comisión se reduce hasta un 5,14% y el de omisión al 7,39%. Además, la media de errores se minimiza hasta conseguir un valor del 6,27% y la exactitud global de la clasificación alcanza el 99,97%.

En cuanto al análisis de los errores producidos, la totalidad de errores de comisión incluidos en los resultados se deben a pequeñas regiones pertenecientes a árboles que han sido clasificadas erróneamente como valores atípicos. Esto es debido a que algunas ramas sobresalen del volumen general del árbol y se extienden sobre el terreno constituyendo una entidad definida por un número reducido de puntos LiDAR. En estos casos, el algoritmo detecta regiones aisladas de pequeño tamaño cuya altitud difiere en gran medida con respecto a su vecindario y por tanto las clasifica erróneamente como error grosero (Figura 136a). De cualquier manera, constituye una fuente de error aislada sin gran relevancia en el conjunto de la clasificación.

Los errores de omisión producidos son mayores a los de comisión. Principalmente, están asociados a líneas eléctricas y a los postes que sustentan el cableado. Ocasionalmente, en los postes y en algunas partes del cableado la densidad de puntos LiDAR aumenta y las pequeñas regiones asociadas contienen una alta cantidad de valores atípicos. En consecuencia, al analizar el vecindario de estos puntos no es posible detectar una alta variabilidad en cuanto a la altitud se refiere y el proceso de identificación de errores groseros falla. Afortunadamente, este tipo de errores no es demasiado frecuente y pueden identificarse fácilmente mediante una revisión visual (Figura 136b).

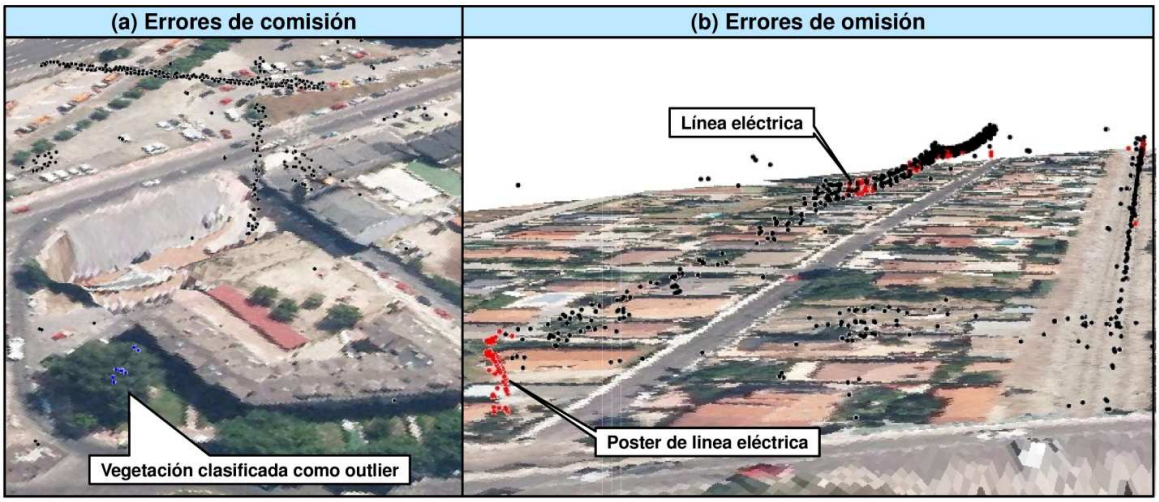


Figura 136. Imagen 3D de errores producidos en la detección de errores groseros. Algoritmo C (C 5).

Existen otras investigaciones donde el enfoque para detectar errores groseros es similar al propuesto en el presente trabajo. Silván-Cárdenas y Wang (2006) tras eliminar los errores groseros cuya altitud difiere en gran medida con respecto al valor esperado mediante un análisis del histograma de frecuencias, analiza el vecindario cilíndrico del resto de los puntos con el fin de identificar entornos donde existe una alta variabilidad en cuanto a altitud. Al igual que en el presente estudio, se utiliza la triangulación de Delaunay para establecer las relaciones de proximidad. No obstante, no se cuantifica la precisión de los resultados obtenidos. Shen et al. (2011) también utiliza un procedimiento similar y tras utilizar un análisis del histograma de frecuencias para detectar errores groseros con valores más atípicos analiza el vecindario de cada punto en cuanto a variaciones locales de altitud. En este caso se emplea una estructura Kd-tree para establecer las relaciones de proximidad

entre puntos y, en vez de utilizar un vecindario cilíndrico como en el presente estudio, se analizan los k vecinos más próximos. En este caso, tampoco son analizadas las precisiones conseguidas.

La comparación de los resultados obtenidos con otros trabajos resulta inviable debido a que en la mayoría de estudios existentes la detección de valores atípicos es un proceso auxiliar y no constituye su objetivo principal. En estos trabajos, los valores atípicos se detectan y filtran de la escena sin analizar los resultados (Akca et al. 2009; Forlani et al. 2006; Chehata et al. 2008; Meng et al. 2009) y en el mejor de los casos, se contabiliza el número de errores groseros detectados o el porcentaje con respecto al total de puntos, pero sin entrar en detalle de las precisiones alcanzadas (Aguilar y Mills 2008; Shen et al. 2011). Además, la utilización de distintos datos test para probar los algoritmos imposibilita la comparación.

Uno de los principales problemas a la hora de detectar valores atípicos radica en la necesidad de detectar errores groseros con características muy distintas y cuya clasificación no es posible mediante la utilización de un único análisis. Para solucionar este problema, el enfoque más generalizado es analizar el histograma de frecuencias para detectar los errores groseros que poseen una altitud que difiere en gran medida a la esperada y posteriormente identificar el resto de valores atípicos mediante un análisis de la variabilidad en altitud del vecindario de cada punto (Meng et al. 2009; Silván-Cárdenas y Wang 2006). El principal problema de este enfoque es que el análisis de la distribución de frecuencias debe de llevarse a cabo por parte de un operador, por lo que no resulta un método completamente automático. En este sentido, uno de los mayores aportes del presente estudio es el diseño de un proceso completamente automático capaz de identificar los distintos tipos de errores groseros sin análisis adicionales. Para ello, se ha desarrollado un algoritmo multiproceso que ajusta automáticamente los parámetros umbrales y concatena dos análisis cuyos resultados son finalmente fusionados, consiguiendo así un procesado de datos completamente automáticos con una alta precisión, que alcanza una exactitud global en la clasificación que oscila entre el 99,97% 99,9% en función de los datos test utilizados.

IV. 2. Detección de terreno

La detección del terreno dentro del conjunto de datos LiDAR es un campo ampliamente estudiado donde multitud de algoritmos han sido desarrollados a lo largo de los últimos años. Entre ellos destacan los filtros morfológicos (Voselman 2000; Sithole 2001; Roggero 2001; Chen et al. 2007; Kim y Shan 2012), los filtros basados en segmentación (Tovari y Pfeifer 2005; Rabbani et al. 2006; Kim et al. 2007; Lari et al. 2011) y procesos de densificación progresiva (Axelsson 2000; Sohn y Dowman 2002; Ekhtari et al. 2008).

El algoritmo desarrollado en el presente estudio tiene en cuenta las conclusiones obtenidas por Sithole y Vosselman (2004). En este estudio se realiza un análisis comparativo de los diferentes filtros existentes y se concluye que aquellos que mejores resultados ofrecen son los que utilizan superficies de referencia aproximadas tales como el método de densificación de TIN. Por lo que se decide desarrollar un algoritmo que utilice el enfoque de densificación progresiva para detectar el terreno. Por otro lado, uno de los problemas más importante ligado a la detección de la clase terreno, consiste en la dificultad de desarrollar algoritmos completamente automáticos que actúen con efectividad independientemente de las características de la escena (Baltsavias 1999). En este sentido, en el presente trabajo se desarrollan dos algoritmos que combinan características asociadas a diferentes enfoques (densificación progresiva, técnicas de segmentación, filtros morfológicos y métodos de interpolación) con la finalidad de conseguir detectar el terreno en entornos urbanos de forma completamente automática y con efectividad, minimizando en la medida de lo posible la dependencia entre la precisión de los resultados y las características de la escena.

Tras ejecutar el primero de los algoritmos (Algoritmo A) y analizar cualitativamente los resultados (análisis visual) se observó una alta dependencia entre los parámetros umbrales utilizados y la precisión de la clasificación obtenida. Por este motivo, se desarrolló un segundo algoritmo (algoritmo B) basado en el primero, que incluye análisis adicionales de las regiones obtenidas tras la segmentación y que permite una mayor flexibilidad de los parámetros umbrales.

El primero de los algoritmos (Algoritmo A) utiliza el enfoque de densificación progresiva combinado con la segmentación de los datos mediante crecimiento de regiones. Básicamente, el algoritmo ejecuta tres procesos o fases de forma concatenada. En el presente apartado se analizarán tanto la precisión de los resultados como su dependencia con respecto a los distintos parámetros umbrales utilizados.

La primera fase consiste en segmentar la escena en superficies continuas, en este caso se utiliza un desnivel umbral de 0,25 m y la relación de proximidad se establece mediante las conexiones que proporciona la triangulación de Delaunay. El parámetro umbral en desnivel ofrece buenos resultados en todos los casos, por lo que no se modificará al realizar las distintas pruebas y el resultado de la segmentación será constante.

La segunda fase tiene como objetivo generar una primera clase de terreno provisional que servirá como datos de entrada en el proceso de densificación. Para ello, se divide la zona de estudio en bloques cuadrados y la región que contiene el punto de menor altitud de cada bloque es clasificada como terreno provisional. Es necesario que el tamaño del bloque sea superior a la superficie del edificio con mayor tamaño de la zona de estudio, ya que de lo contrario, podría seleccionarse como terreno provisional una región asociada a un plano de techo. Consecuentemente, la correcta selección del tamaño del bloque tiene una alta influencia en la precisión de los resultado. En este sentido, la dependencia de la longitud del lado del bloque con respecto a la precisión de los resultados será objeto de estudio en el presente apartado.

Una vez generado el terreno provisional comienza el proceso de densificación progresiva que tiene como finalidad la obtención de la clase terreno definitiva. Esta tercera fase es un proceso iterativo donde se genera una superficie de referencia mediante la interpolación del terreno provisional. Seguidamente, los puntos con una altitud similar a la superficie de referencia son añadidos al terreno provisional y la superficie de referencia se actualiza. El proceso se repite hasta que ningún nuevo punto es clasificado como terreno, momento en que la clase de terreno provisional se convierte en definitiva y finaliza el análisis. En esta fase, se utiliza un parámetro umbral en desnivel para determinar si un punto está próximo o no a la superficie de referencia. La elección de un parámetro umbral en desnivel alto puede ocasionar errores ya que puntos pertenecientes a vegetación, vehículos u otras entidades de altura reducida pueden clasificarse erróneamente como terreno. Por otro lado, la elección de un parámetro en desnivel demasiado restrictivo puede ocasionar errores de omisión en la clasificación. En el presente apartado se exponen los resultados obtenidos tras utilizar distintos valores como parámetro umbral en desnivel y se analizará el grado de dependencia existente entre la precisión de los resultados con respecto al umbral utilizado.

En la Figura 137 se detallan los diferentes parámetros umbrales utilizados sobre los que se realizará el análisis de los resultados.

PARÁMETROS UMBRALES FIJOS PARA ALGORITMO A:

UMBRAL EN IZ SEGMENTACIÓN: 0,25 m **TAMAÑO DE CELDA:** 1 m **EXPONENTE DE PONDERACIÓN EN INTERPOLACIÓN:** 3

PARÁMETROS UMBRALES VARIABLES PARA ALGORITMO A:

	U_IZ	L_Bloque
A 1-1	0,2 m	100 m
A 1-2	0,2 m	50 m
A 1-3	0,2 m	25 m
A 1-4	0,2 m	10 m
A 2-1	0,4 m	100 m
A 2-2	0,4 m	50 m
A 2-3	0,4 m	25 m
A 2-4	0,4 m	10 m

	U_IZ	L_Bloque
A 3-1	0,6 m	100 m
A 3-2	0,6 m	50 m
A 3-3	0,6 m	25 m
A 3-4	0,6 m	10 m
A 4-1	1 m	100 m
A 4-2	1 m	50 m
A 4-3	1 m	25 m
A 4-4	1 m	10 m

	U_IZ	L_Bloque
A 5-1	2 m	100 m
A 5-2	2 m	50 m
A 5-3	2 m	25 m
A 5-4	2 m	10 m

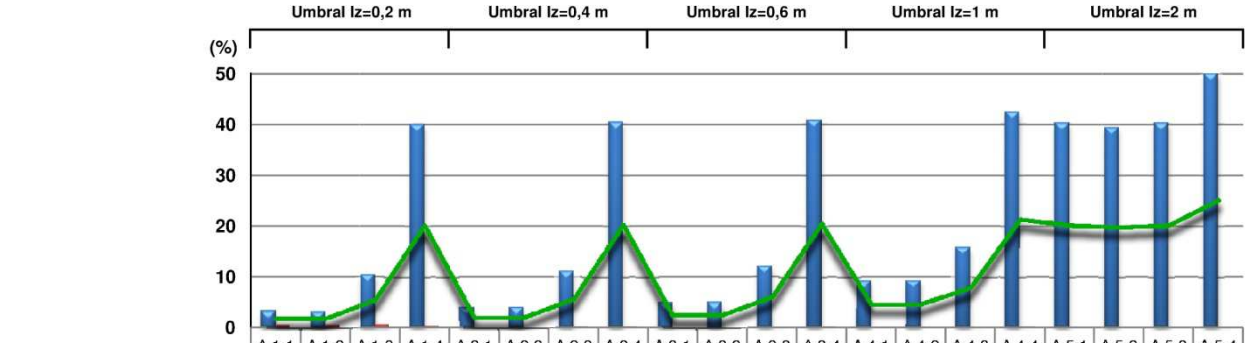
U_IZ: Umbral en desnivel

L_Bloque: Longitud del lado del bloque (m)

Figura 137. Detección del terreno. Parámetros umbrales. Algoritmo A.

En total el algoritmo se ha ejecutado veinte veces con parámetros umbrales distintos. El objetivo del análisis es establecer el grado de dependencia del proceso con respecto al parámetro umbral en desnivel necesario para detectar puntos del terreno próximos a la superficie de referencia. Con tal fin, cinco parámetros en desnivel han sido analizados. Además, para cada desnivel utilizado se ha ejecutado el algoritmo cinco veces modificando el tamaño de los bloques necesarios para generar el terreno provisional inicial. Los resultados obtenidos en cuanto a errores de comisión, omisión y media de errores para los datos test 1 (Vall d'Uixó) se muestran en la Figura 138.

La tendencia indica que la media de errores aumenta al incrementarse el parámetro umbral en desnivel, alcanzando errores máximos al utilizar un valor de 2 m. El error que mayor relevancia presenta es el de comisión, producido al clasificar como puntos de terreno aquellos pulsos láser reflejados sobre vegetación de escasa altura u otros objetos próximos al terreno como automóviles. Consecuentemente, el error de comisión aumenta al incrementarse el umbral en desnivel llegando a un máximo del 50,13% al utilizar un desnivel de 2 m. Con respecto al error por omisión prácticamente es nulo en todo momento. Únicamente se presenta de forma puntual al utilizar el parámetro umbral en desnivel más restrictivo (0,2 m). No obstante, en ningún caso supera el 0,5%.



	A 1-1	A 1-2	A 1-3	A 1-4	A 2-1	A 2-2	A 2-3	A 2-4	A 3-1	A 3-2	A 3-3	A 3-4	A 4-1	A 4-2	A 4-3	A 4-4	A 5-1	A 5-2	A 5-3	A 5-4
Error de comisión (%)	3,26	3,09	10,41	39,97	4,02	3,96	11,15	40,38	4,96	4,96	12,11	40,82	9,30	9,25	15,76	42,50	40,30	39,47	40,28	50,13
Error de omisión (%)	0,50	0,50	0,59	0,35	0,10	0,10	0,21	0,18	0,08	0,08	0,11	0,15	0,07	0,07	0,09	0,14	0,04	0,04	0,06	0,12
Media de errores (%)	1,88	1,80	5,50	20,16	2,06	2,03	5,68	20,28	2,52	2,52	6,11	20,49	4,69	4,66	7,93	21,32	20,17	19,76	20,17	25,13
Tamaños bloques (m)	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10

Figura 138. Gráfico de errores en la detección del terreno. Algoritmo A. Datos test 1.

La dependencia de la precisión alcanzada con respecto al tamaño de los bloques utilizados para generar el terreno provisional inicial también queda patente. En el gráfico puede observarse como el error por comisión aumenta al disminuir la longitud del lado de los bloques, alcanzando su máximo cuando la longitud se reduce a 10 m, con independencia del

umbral en desnivel utilizado. Este incremento del error es debido a que el terreno provisional inicial es generado mediante la selección de las regiones que poseen los puntos de menor altitud de cada bloque. Por tanto, si el tamaño de los bloques es demasiado pequeño, regiones asociadas a planos de techo son clasificadas como terreno incrementando así el error de comisión. Si tomamos como ejemplo un parámetro umbral en desnivel de 0,2 m puede apreciarse como el error de comisión del 3,26% obtenido al utilizar bloques de 100 m de lado, aumenta hasta el 39,97% al reducir los lados a 10 m de longitud.

En la Figura 139 se muestra la exactitud global de la clasificación obtenida para los datos test 1 (Vall d'Uixó). En este caso puede apreciarse con claridad como la exactitud diminuye al incrementar el umbral en desnivel. Además la influencia del tamaño de los bloques también queda reflejada en el gráfico, donde la exactitud global marca una tendencia descendiente a medida que se reduce el tamaño de los bloques.

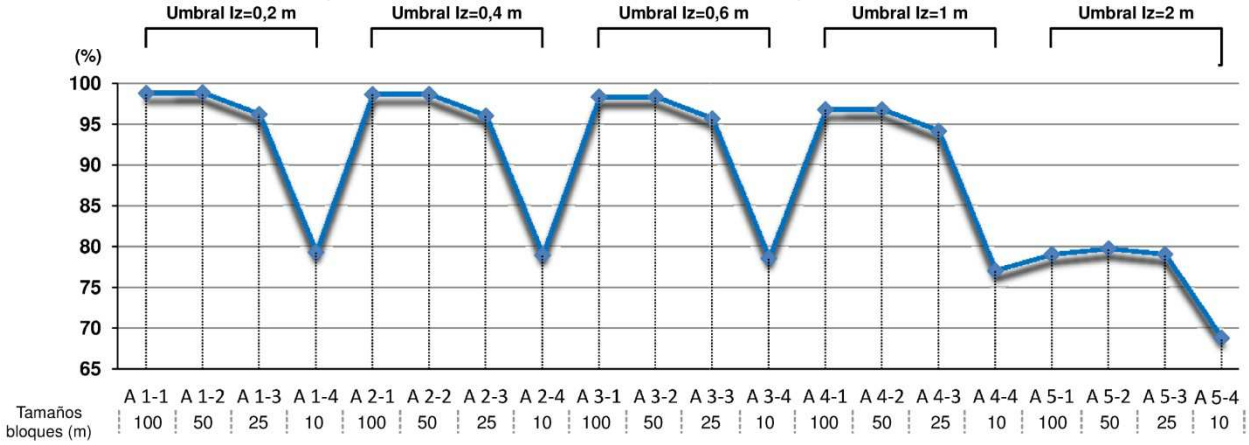


Figura 139. Gráfico de exactitud global en la clasificación. Detección del terreno. Algoritmo A. Datos test 1 (Vall d'Uixó).

La dependencia del algoritmo con respecto al tamaño de los bloques se aprecia con claridad en la Figura 140. En ella se muestra un modelo 3D con los errores obtenidos al utilizar bloques de 10 y 100 m de lado. El parámetro umbral en desnivel utilizado es en ambos casos de 0,2 m. Claramente puede apreciarse como la reducción del tamaño de los bloques ocasiona que multitud de tejados y planos de techo sean clasificados como terreno provisional. Este error se transmite directamente al proceso de densificación y por consiguiente es incluido en el resultados final. Por tanto, la selección del tamaño de los bloques será crucial para obtener resultados precisos, debiendo seleccionar siempre un tamaño mayor a cualquier edificio existente en la escena.

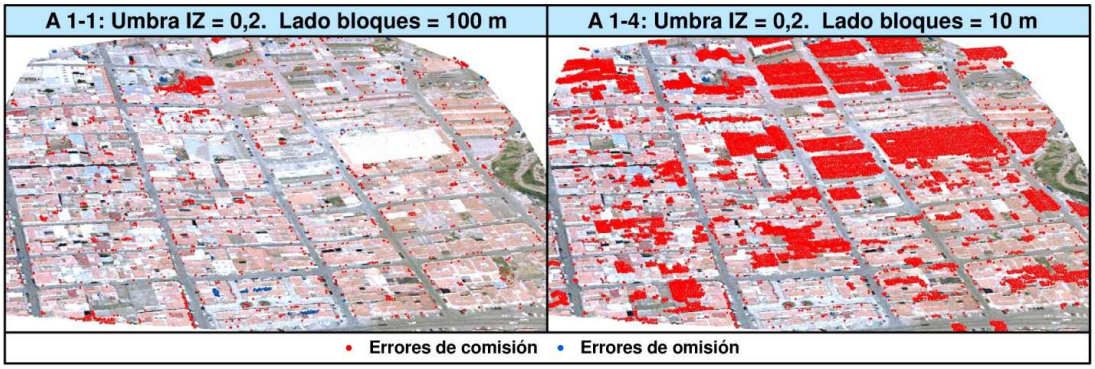
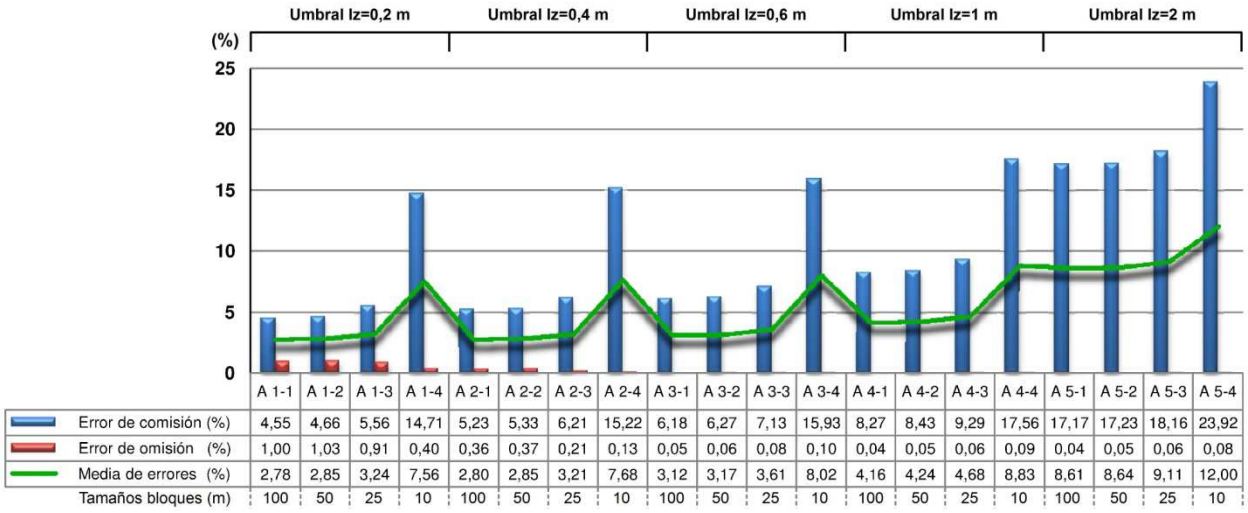


Figura 140. Imagen 3D de precisión alcanzada en la detección del terreno. Algoritmo A. Datos test 1 (Vall d'Uixó).

Las precisiones alcanzadas al ejecutar el algoritmo A en los datos test 2 (Collado Villalba) son similares a las obtenidos sobre los datos test 1(Vall d'Uixó) (Figura 141). Con respecto al error de comisión también se observa una tendencia ascendente al incrementar el umbral en desnivel, llegando a alcanzar un error máximo del 23,92% al utilizar un desnivel umbral de 2 m. Con respecto al error por comisión únicamente se presenta cuando el desnivel umbral toma un pequeño valor (0,2 m y 0,4 m). No obstante, se presenta de forma puntual y no constituye una fuente de error importante, alcanzando un máximo de 1,03% al utilizar un umbral en desnivel de 0,4 m y bloques de 50 m de lado.



	A 1-1	A 1-2	A 1-3	A 1-4	A 2-1	A 2-2	A 2-3	A 2-4	A 3-1	A 3-2	A 3-3	A 3-4	A 4-1	A 4-2	A 4-3	A 4-4	A 5-1	A 5-2	A 5-3	A 5-4
Error de comisión (%)	4,55	4,66	5,56	14,71	5,23	5,33	6,21	15,22	6,18	6,27	7,13	15,93	8,27	8,43	9,29	17,56	17,17	17,23	18,16	23,92
Error de omisión (%)	1,00	1,03	0,91	0,40	0,36	0,37	0,21	0,13	0,05	0,06	0,08	0,10	0,04	0,05	0,06	0,09	0,04	0,05	0,06	0,08
Media de errores (%)	2,78	2,85	3,24	7,56	2,80	2,85	3,21	7,68	3,12	3,17	3,61	8,02	4,16	4,24	4,68	8,83	8,61	8,64	9,11	12,00
Tamaños bloques (m)	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10

Figura 141. Gráfico de errores en la detección del terreno. Algoritmo A. Datos test 2.

La Figura 142 muestra la exactitud global de la clasificación obtenida para los datos test 2. En este caso, también se aprecia con claridad como la exactitud global disminuye al incrementar el parámetro umbral en desnivel. Esta tendencia es consecuencia del incremento del error de comisión producido al aumentar el desnivel umbral. Además, la influencia del tamaño de los bloques también queda reflejada en el gráfico, donde la exactitud global marca una tendencia descendiente a medida que se reduce el tamaño de los bloques, alcanzando su mínimo en todos los casos cuando el lado de los bloques se reduce a 10 m.

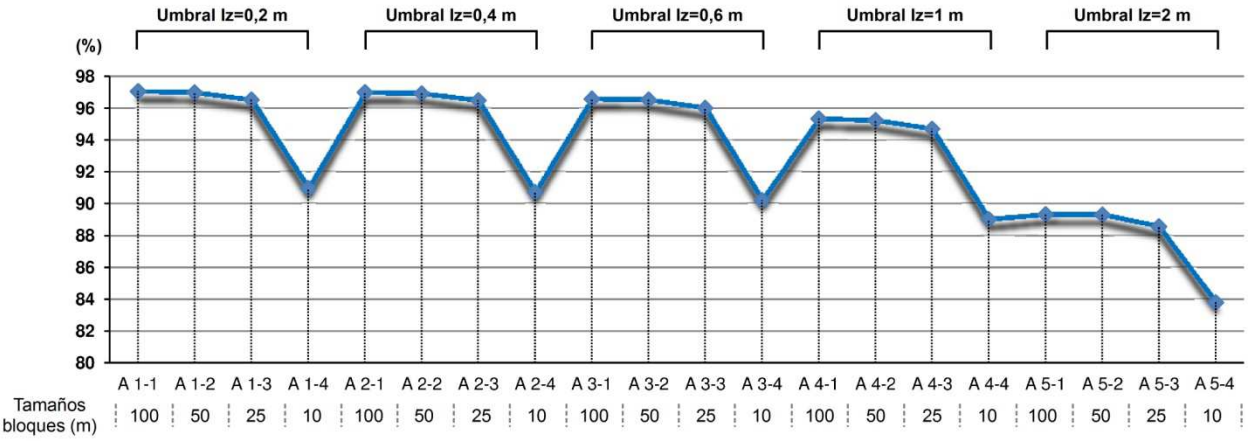


Figura 142. Gráfico de exactitud global en la clasificación. Detección del terreno. Algoritmo A. Datos test 2 (Collado Villalba).

Puede concluirse que el Algoritmo A alcanza resultados muy precisos cuando se utiliza un parámetro umbral en desnivel muy restrictivo (0,2 m o 0,4 m) y el tamaño de los bloques es suficientemente grande para no generar errores de comisión. No obstante, los resultados son altamente dependientes a ambos parámetros umbrales.

El incremento del umbral en desnivel ocasiona un aumento considerable del error de comisión. Esto es debido a que la utilización de un alto desnivel como parámetro umbral implica que puntos que están relativamente alejados de la superficie de referencia sean clasificados como terreno. Consecuentemente, pulsos láser reflejados sobre diferentes entidades (vehículos, vegetación de escasa altura, puntos de fachada, etc.) son incluidas en la clase terreno definitiva.

En la Figura 143 se observan los diferentes errores que ocasiona el algoritmo A, así como el incremento que se produce en los mismos al aumentar el desnivel umbral. En ambos casos, el tamaño de los bloques utilizados para generar el terreno provisional inicial es el mismo. La primera de las imágenes muestra el resultado obtenido al utilizar un parámetro umbral en desnivel de 0,6 m. En ella se pueden observar principalmente tres tipos de errores de comisión: puentes, vegetación y puntos de fachada.

Las calles y carreteras constituyen grandes superficies continuas y al segmentar la escena se asocian a regiones de gran tamaño. Dentro de estas regiones están incluidos los puentes y por tanto, al aplicar el proceso de densificación se clasifican automáticamente como terreno. Esta inclusión se debe a que la superficie de los puentes está conectada con el terreno sin presentar discontinuidades en el inicio y fin de los mismos. Los puentes constituyen una estructura artificial y no deben de considerarse terreno, por tanto se trata de un error de comisión importante. En cualquier caso, en el presente trabajo se desarrolla un algoritmo que consigue detectarlos y filtrarlos de forma precisa.

Además de los puentes, en la primera de las imágenes se han clasificado erróneamente como terreno puntos pertenecientes a vegetación de baja altura. Estos errores se deben a que en algún momento del proceso de densificación, algunos puntos de la cobertura vegetal se han encontrado a una distancia vertical menor a 0,6 metros con respecto a la superficie de referencia y por tanto se han clasificado como terreno. La clasificación errónea de puntos de fachada de edificios se produce por el mismo motivo.

Al aumentar el umbral en desnivel a 2 m puede apreciarse cómo se incrementa el error de comisión asociado tanto a puntos de fachada como a vegetación. Además, se introduce un nuevo error de comisión que se debe a la clasificación de planos de techo de edificios como terreno. Estos casos suelen darse cuando el parámetro umbral en desnivel es demasiado grande, ya que al producirse numerosos errores de comisión la superficie de referencia resulta errónea y se aleja del terreno en exceso, llegando a clasificar tejados de edificios en algunos casos. No obstante, este tipo de errores puede darse también excepcionalmente en zonas asociadas a grandes pendientes a pesar de utilizar un parámetro umbral en desnivel pequeño (Figura 60).

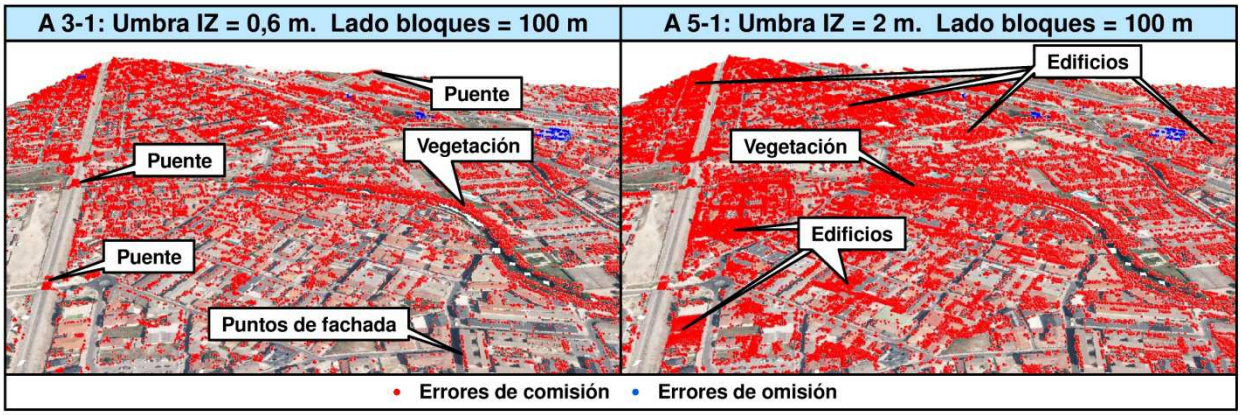


Figura 143. Imagen 3D de errores producidos en la detección del terreno. Algoritmo A. Datos test 2 (Collado Villalba).

Con el fin de reducir en la medida de lo posible los errores que ocasiona el Algoritmo A y además disminuir la dependencia de los parámetros umbrales con respecto a la precisión de los resultados, se diseña un nuevo algoritmo (Algoritmo B) que sigue el mismo enfoque que el anterior pero que incluye dos análisis adicionales.

En estas nuevas fases de procesamiento se analiza el tamaño y la posición relativa de las regiones obtenidas tras la segmentación con el fin de realizar una clasificación previa que evite errores de comisión en el posterior proceso de densificación. Básicamente el análisis analiza dos características de las regiones: tamaño y posición relativa en cuanto a altitud con respecto a las regiones adyacentes.

Las regiones obtenidas tras segmentar la escena se corresponden con superficies continuas delimitadas por cambios bruscos en pendiente. Dado que el terreno en zonas urbanas no presenta grandes discontinuidades, las mayores regiones tendrán una alta probabilidad de pertenecer al terreno. En cuanto a las regiones de menor tamaño, aquellas que pertenezcan al terreno (puntos de tierra bajo cubierta vegetal o patios interiores de edificios) estarán rodeadas de regiones de mayor altitud (árboles, planos de techo...). Por el contrario, las que pertenezcan a vegetación o edificios estarán rodeadas de regiones de igual o menor altitud. El algoritmo B tiene en cuenta estas dos premisas y analiza para cada región tanto su tamaño como la posición relativa con respecto a las regiones adyacentes. Tras realizar el análisis individual sobre cada una de las regiones, se aplican reglas de decisión para descartar las regiones que tienen una menor posibilidad de pertenecer al terreno, forzando así el proceso de densificación y consiguiendo eliminar una gran parte de los errores de comisión.

El Algoritmo B incorpora dos análisis adicionales. El primero se ejecuta previamente a la detección del terreno provisional inicial (más restrictivo) y el segundo dentro del propio proceso de densificación (menos restrictivo). En cualquier caso, la descripción del algoritmo y los análisis adicionales se detallan en el apartado correspondiente (Apartado III. 5).

El Algoritmo B también se ha ejecutado un total de veinte veces con el fin de evaluar tanto la precisión de los resultados como su dependencia con respecto a los parámetros umbrales utilizados. Los parámetros utilizados en las veinte pruebas son los mismos que se han usado en el Algoritmo A (Figura 144).

PARÁMETROS UMBRALES FIJOS PARA ALGORITMO B:

UMBRAL EN IZ SEGMENTACIÓN: 0,25 m **TAMAÑO DE CELDA:** 1 m **EXPONENTE DE PONDERACIÓN EN INTERPOLACIÓN:** 3

PARÁMETROS UMBRALES VARIABLES PARA ALGORITMO B:

	U_IZ	L_Bloque
B 1-1	0,2 m	100 m
B 1-2	0,2 m	50 m
B 1-3	0,2 m	25 m
B 1-4	0,2 m	10 m
B 2-1	0,4 m	100 m
B 2-2	0,4 m	50 m
B 2-3	0,4 m	25 m
B 2-4	0,4 m	10 m

	U_IZ	L_Bloque
B 3-1	0,6 m	100 m
B 3-2	0,6 m	50 m
B 3-3	0,6 m	25 m
B 3-4	0,6 m	10 m
B 4-1	1 m	100 m
B 4-2	1 m	50 m
B 4-3	1 m	25 m
B 4-4	1 m	10 m

	U_IZ	L_Bloque
B 5-1	2 m	100 m
B 5-2	2 m	50 m
B 5-3	2 m	25 m
B 5-4	2 m	10 m

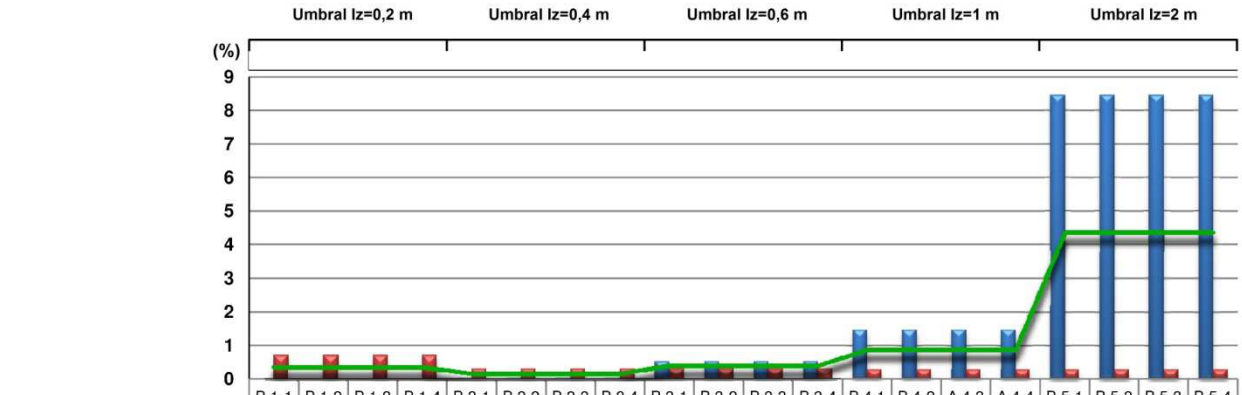
U_IZ: Umbral en desnivel

L_Bloque: Longitud del lado del bloque (m)

Figura 144. Detección del terreno. Parámetros umbrales utilizados. Algoritmo B.

En la Figura 145 se muestra un gráfico con los errores producidos tras ejecutar el Algoritmo B sobre los datos test 1 (Vall d'Uixó). Una de las características más relevantes que puede extraerse del mismo es la invariabilidad de los errores producidos con respecto al tamaño de los bloques. Por ejemplo, para un umbral en desnivel de 0,2 metros el error de comisión es de 0,01% y el de omisión 0,70% tanto al utilizar bloques de 100 m de lado como si se utilizan bloques cuyos lados miden 10 m. Lo mismo ocurre con el resto de parámetros umbrales en desnivel, donde los errores se mantienen constantes al variar el tamaño de los bloques utilizados para generar el terreno provisional inicial.

Referente a la valoración de los resultados con respecto a los diferentes umbrales en desnivel utilizados, los desniveles más restrictivos son los que alcanzan los resultados más precisos. En concreto, el resultado que minimiza en mayor grado los distintos errores se produce al utilizar un desnivel umbral de 0,4 m, donde se obtiene un error de comisión del 0,03%, un error de omisión del 0,30% y una media de errores de tan solo el 0,17%.



	B 1-1	B 1-2	B 1-3	B 1-4	B 2-1	B 2-2	B 2-3	B 2-4	B 3-1	B 3-2	B 3-3	B 3-4	B 4-1	B 4-2	A 4-3	A 4-4	B 5-1	B 5-2	B 5-3	B 5-4
Error de comisión (%)	0,01	0,01	0,01	0,01	0,03	0,03	0,03	0,03	0,51	0,51	0,51	0,51	1,46	1,46	1,46	1,46	8,45	8,45	8,45	8,45
Error de omisión (%)	0,70	0,70	0,70	0,70	0,30	0,30	0,30	0,30	0,30	0,30	0,30	0,30	0,29	0,29	0,29	0,29	0,28	0,28	0,28	0,28
Media de errores (%)	0,36	0,36	0,36	0,36	0,17	0,17	0,17	0,17	0,41	0,41	0,41	0,41	0,88	0,88	0,88	0,88	4,37	4,37	4,37	4,37
Tamaños bloques (m)	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10

Figura 145. Gráfico de errores en la detección del terreno. Algoritmo B. Datos test 1.

Tanto la independencia de los resultados con respecto al tamaño de los bloques como la alta precisión alcanzada al utilizar umbrales en desnivel restrictivos puede apreciarse en el gráfico correspondiente a la exactitud global de la clasificación (Figura 146). Como puede observarse, la exactitud global asociada a cada desnivel umbral permanece invariable al utilizar distintos tamaños de bloque. Por otro lado, la mayor precisión se obtiene al emplear un desnivel umbral de 0,4 m.

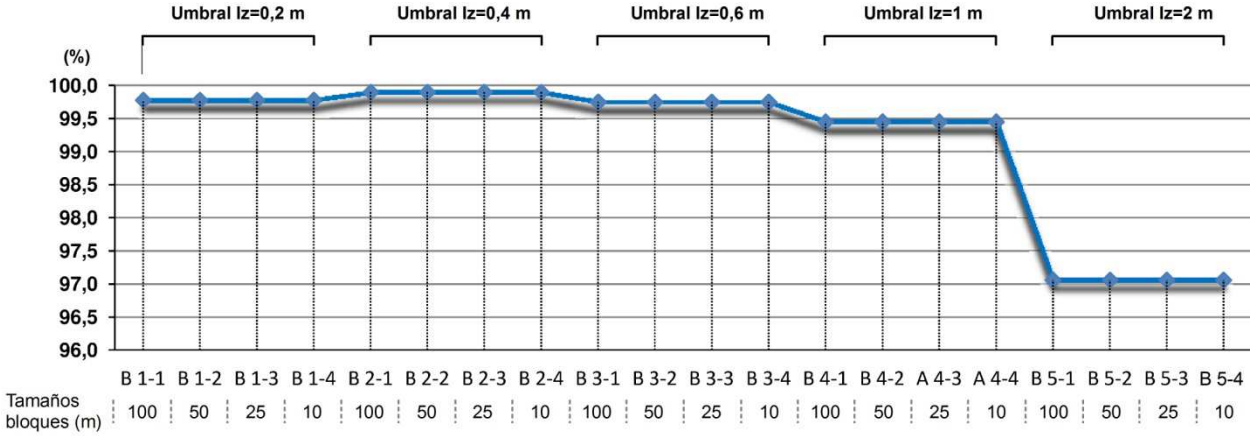
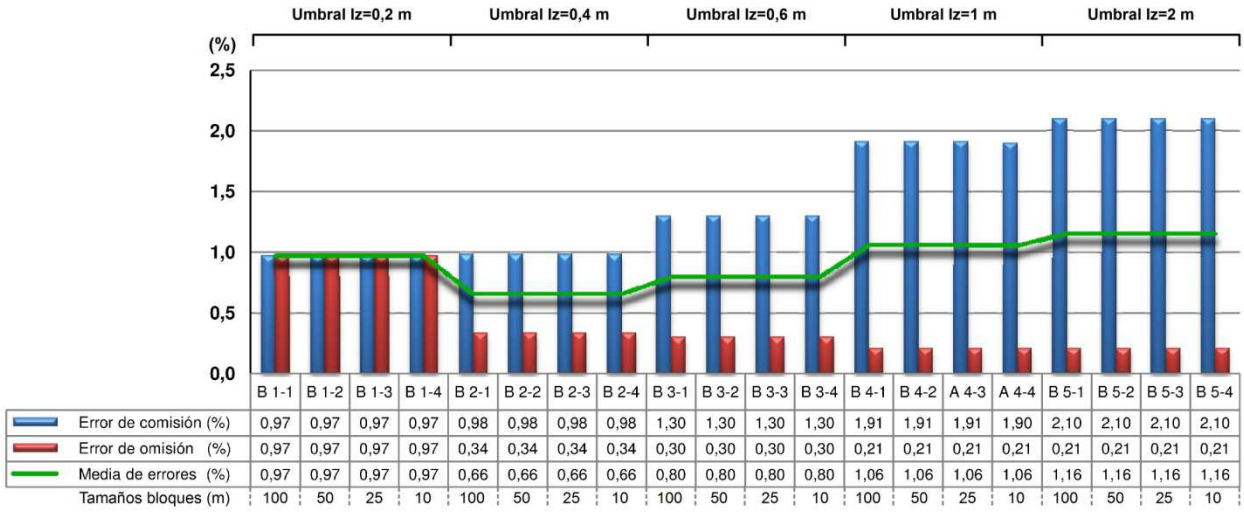


Figura 146. Gráfico de exactitud global en la clasificación. Detección del terreno. Algoritmo B. Datos test 1 (Vall d'Uixó).

Unos resultados similares se han obtenido al ejecutar el Algoritmo B sobre los datos test 2 (Collado Villalba). Con respecto a los errores producidos, éstos se mantienen constantes independientemente del tamaño de los bloques utilizados para generar el terreno provisional (Figura 147), quedando así patente la capacidad del Algoritmo B para reducir la dependencia de los resultados con respecto a los parámetros umbrales. En este caso, el resultado que menos errores genera se obtiene de nuevo al utilizar un parámetro umbral en desnivel de 0,4 m, donde se consigue reducir el error de comisión al 0,98%, el de omisión al 0,34% y la media de errores a tan solo 0,66%.



	B 1-1	B 1-2	B 1-3	B 1-4	B 2-1	B 2-2	B 2-3	B 2-4	B 3-1	B 3-2	B 3-3	B 3-4	B 4-1	B 4-2	A 4-3	A 4-4	B 5-1	B 5-2	B 5-3	B 5-4
Error de comisión (%)	0,97	0,97	0,97	0,97	0,98	0,98	0,98	0,98	1,30	1,30	1,30	1,30	1,91	1,91	1,91	1,90	2,10	2,10	2,10	2,10
Error de omisión (%)	0,97	0,97	0,97	0,97	0,34	0,34	0,34	0,34	0,30	0,30	0,30	0,30	0,21	0,21	0,21	0,21	0,21	0,21	0,21	0,21
Media de errores (%)	0,97	0,97	0,97	0,97	0,66	0,66	0,66	0,66	0,80	0,80	0,80	0,80	1,06	1,06	1,06	1,06	1,16	1,16	1,16	1,16
Tamaños bloques (m)	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10

Figura 147. Gráfico de errores en la detección del terreno. Algoritmo B. Datos test 2.

La exactitud global de la clasificación para los datos test 2, también pone de manifiesto la independencia de los resultados con respecto al tamaño del bloque utilizado a la hora de generar el terreno provisional, de forma que la exactitud global alcanzada permanece constante para cada umbral en desnivel (Figura 148). Al igual que ocurría al ejecutar el algoritmo sobre los datos test 1, la mayor exactitud global se alcanza al utilizar un umbral en desnivel de 0,4 m.

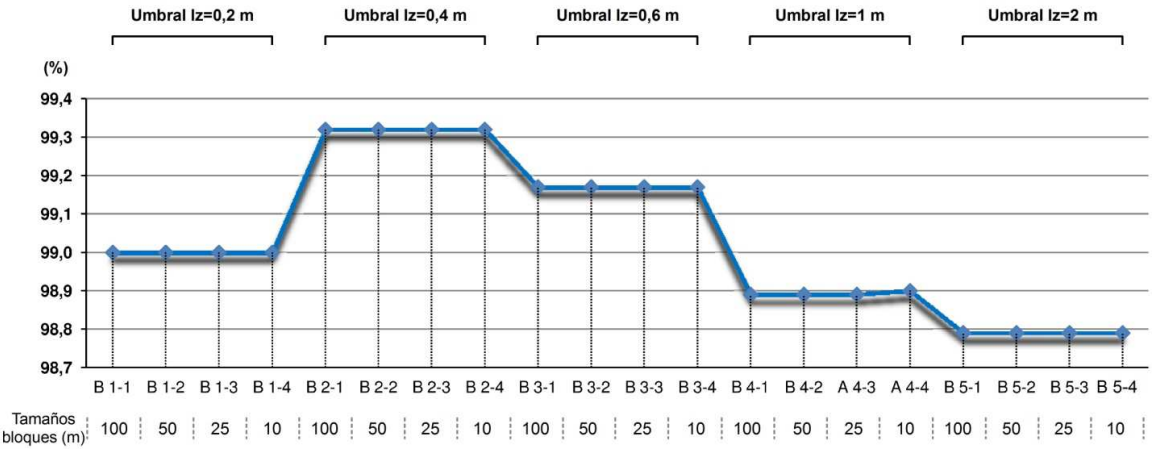
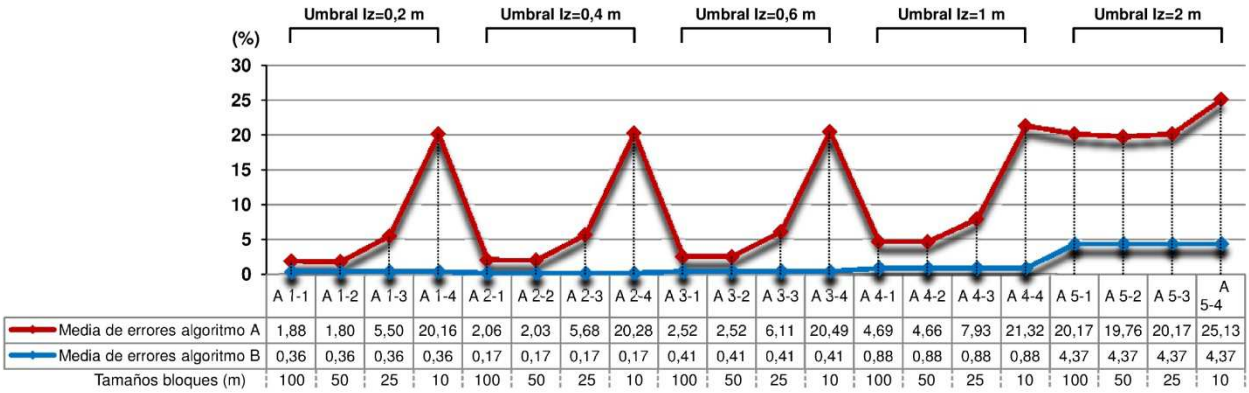


Figura 148. Gráfico de exactitud global en la clasificación. Detección del terreno. Algoritmo B. Datos test 2.

Con el fin de realizar una comparativa precisa de los resultados obtenidos por los algoritmos A y B, se han generado gráficos y tablas que incluyen la media de errores así como la exactitud global obtenida para ambos algoritmos sobre las distintas zonas test. Respecto a los datos test 1 (Vall d'Uixó), el siguiente gráfico (Figura 149) muestra la media de errores obtenida tanto para el Algoritmo A como para el Algoritmo B. En él pueden compararse los resultados obtenidos para los diferentes parámetros umbrales.



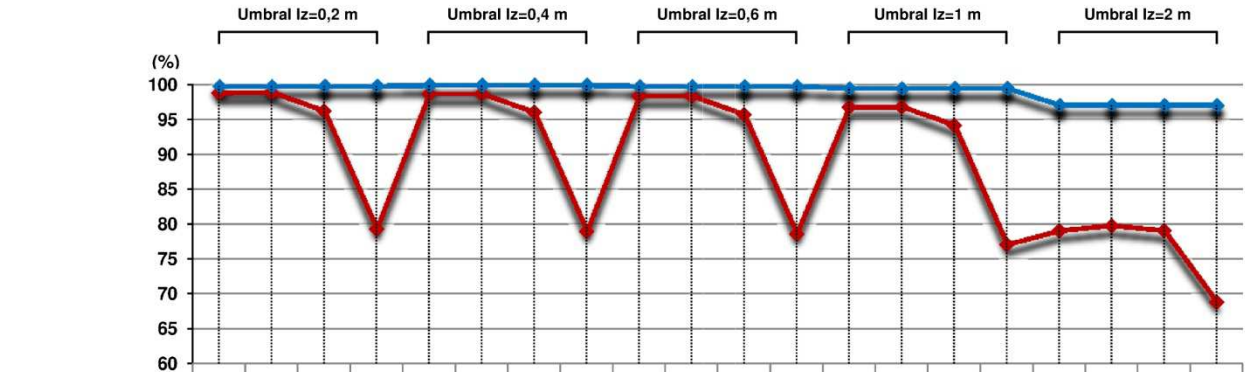
	A 1-1	A 1-2	A 1-3	A 1-4	A 2-1	A 2-2	A 2-3	A 2-4	A 3-1	A 3-2	A 3-3	A 3-4	A 4-1	A 4-2	A 4-3	A 4-4	A 5-1	A 5-2	A 5-3	A 5-4
Media de errores algoritmo A	1,88	1,80	5,50	20,16	2,06	2,03	5,68	20,28	2,52	2,52	6,11	20,49	4,69	4,66	7,93	21,32	20,17	19,76	20,17	25,13
Media de errores algoritmo B	0,36	0,36	0,36	0,36	0,17	0,17	0,17	0,17	0,41	0,41	0,41	0,41	0,88	0,88	0,88	0,88	4,37	4,37	4,37	4,37
Tamaños bloques (m)	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10

Figura 149. Comparación de media de errores entre Algoritmos A y B. Detección del terreno sobre datos test 1.

El primer aspecto relevante es la variabilidad de los resultados al utilizar distintos tamaños de bloques para generar el terreno provisional. En este sentido, mientras que el Algoritmo B obtiene los mismos resultados para cada uno de los parámetros umbrales en desnivel, el Algoritmo A presenta un incremento de la media de errores a medida que el lado de los bloques se reduce. Esta diferencia es debida a la inclusión del primer análisis adicional previamente a la detección del terreno provisional en el Algoritmo B, consiguiendo con ello eliminar por completo la dependencia de los resultados con respecto al tamaño de los bloques. Este aspecto resulta de gran relevancia en cuanto a la automatización del proceso se refiere, ya que para el Algoritmo A es necesario que el usuario realice una revisión visual de la escena con el fin de evitar definir un tamaño de bloque menor a la superficie de los edificios incluidos en los datos. Este proceso de revisión visual no es necesario para el Algoritmo B, ya que el análisis adicional elimina la posibilidad de incluir planos de techos dentro del terreno provisional inicial y, por tanto, la selección del tamaño de los bloques ya no constituye un problema.

Otra información importante que ofrece el gráfico es que el error medio obtenido por el Algoritmo A es siempre mayor que el obtenido por el Algoritmo B. Además, esta diferencia entre errores aumenta a medida que se incrementa el desnivel utilizado como parámetro umbral. Esta tendencia también se aprecia si observamos el gráfico donde se representa la exactitud global de la clasificación obtenida por ambos algoritmos (Figura 150). En este caso, la exactitud global de la clasificación es siempre mayor al utilizar el algoritmo B, acentuándose la diferencia entre la exactitud alcanzada a medida que se incrementa el parámetro umbral en desnivel. Ésto se debe a la inclusión del segundo análisis adicional en el proceso de densificación del Algoritmo B. En el Algoritmo A, cualquier punto que se encuentra a menor distancia de la superficie de referencia que el parámetro umbral en desnivel es incluido en la clase terreno, lo que conlleva que puntos asociados a objetos de baja altura tales como automóviles o vegetación sean clasificados erróneamente como terreno en multitud de ocasiones. Además, a medida que aumenta el parámetro umbral en desnivel un mayor número de puntos asociados a estos errores serán clasificados como terreno, constituyendo una fuente de error de comisión importante. Por el contrario, al ejecutar el Algoritmo B, cuando un punto se encuentra a menor distancia que el parámetro umbral en desnivel no es automáticamente incluido en la clase terreno, sino que debe de superar un análisis adicional en cuanto a tamaño de la región asociada y posición relativa con respecto a las regiones adyacentes. Por consiguiente, el Algoritmo B es más robusto y tiene una menor dependencia al umbral en desnivel elegido.

Los mejores resultados obtenidos con respecto a los Algoritmos A y B para la totalidad de parámetros umbrales analizados se alcanza al ejecutar el Algoritmo B con un parámetro de 0,4 m como desnivel umbral, momento en el que se obtiene un error de comisión del 0,03%, un error de omisión del 0,30% y una media de errores de tan solo el 0,17%. La precisión global de la clasificación alcanzada también es máxima y llega al 99,9%.

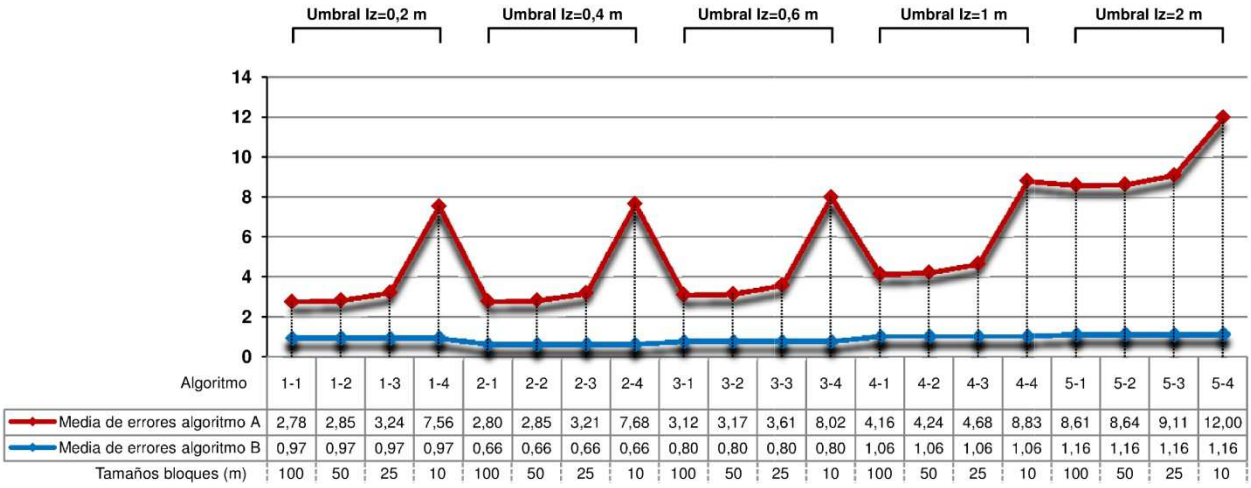


	1-1	1-2	1-3	1-4	2-1	2-2	2-3	2-4	3-1	3-2	3-3	3-4	4-1	4-2	4-3	4-4	5-1	5-2	5-3	5-4
Exactitud global A	98,81	98,86	96,23	79,30	98,67	98,69	96,05	78,96	98,36	98,36	95,70	78,58	96,80	96,82	94,17	77,05	79,05	79,76	79,06	68,80
Exactitud global B	99,78	99,78	99,78	99,78	99,90	99,90	99,90	99,90	99,75	99,75	99,75	99,75	99,45	99,45	99,45	99,45	97,06	97,06	97,06	97,06
Tamaños bloques (m)	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10

Figura 150. Comparación de exactitud global alcanzada por los algoritmos A y B. Detección del terreno sobre datos test 1.

Como se ha comentado anteriormente, las precisiones alcanzados tras ejecutar los algoritmos sobre los datos test 2 (Collado Villalba) son muy similares a los obtenidos en los datos test 1. En este caso también se puede observar la dependencia de la media de errores con respecto al tamaño de los bloques para el Algoritmo A (Figura 151), y como esta dependencia desaparece por completo al utilizar el Algoritmo B. Además, puede apreciarse claramente como los errores producidos por el Algoritmo A son siempre mayores a los ocasionados por el Algoritmo B. Esto se debe a la inclusión de un análisis adicional en el Algoritmo B dentro del propio proceso de densificación que analiza las características de las regiones en cuanto a tamaño y posición relativa. Con ello, se consigue que el Algoritmo B

sea más robusto y se minimiza considerablemente la dependencia de los resultados con respecto al parámetro umbral en desnivel.

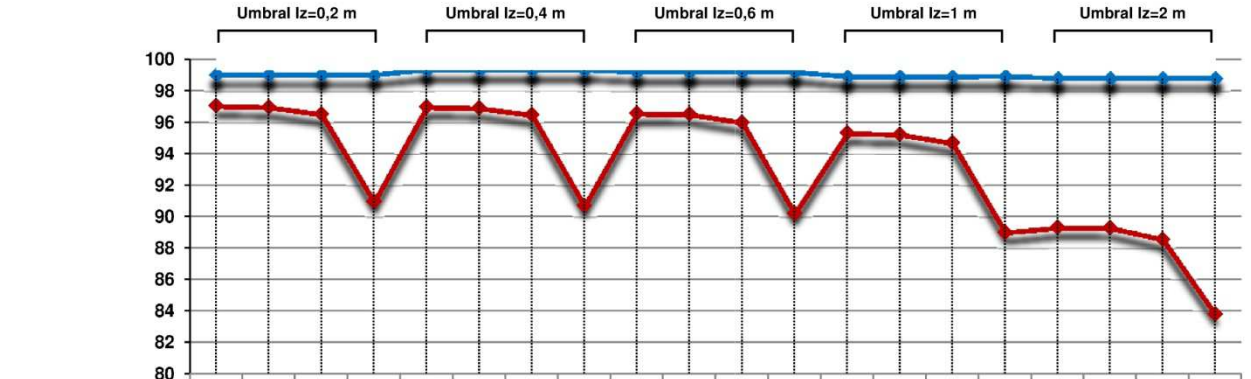


Algoritmo	1-1	1-2	1-3	1-4	2-1	2-2	2-3	2-4	3-1	3-2	3-3	3-4	4-1	4-2	4-3	4-4	5-1	5-2	5-3	5-4
Media de errores algoritmo A	2,78	2,85	3,24	7,56	2,80	2,85	3,21	7,68	3,12	3,17	3,61	8,02	4,16	4,24	4,68	8,83	8,61	8,64	9,11	12,00
Media de errores algoritmo B	0,97	0,97	0,97	0,97	0,66	0,66	0,66	0,66	0,80	0,80	0,80	0,80	1,06	1,06	1,06	1,06	1,16	1,16	1,16	1,16
Tamaños bloques (m)	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10

Figura 151. Comparación de media de errores entre Algoritmos A y B. Detección del terreno sobre datos test 2.

El gráfico comparativo de exactitud global de la clasificación para los datos test 2 (Figura 152) también muestra como la precisión de los resultados obtenidos para el Algoritmo A disminuye a medida que se incrementa el parámetro umbral en desnivel y cómo, sin embargo, ésto no ocurre al ejecutar el Algoritmo B. Además, también se puede observar cómo el Algoritmo B presenta una precisión de los resultados constante con independencia del tamaño de los bloques utilizados para generar el terreno provisional inicial.

Para los datos test 2 (Collado Villalba) al igual que ocurría en los datos test 1, los mejores resultados se alcanza al ejecutar el Algoritmo B con un parámetro de 0,4 m como desnivel umbral, momento en el que se obtiene un error de comisión del 0,98%, un error de omisión del 0,34% y una media de errores de tan solo el 0,66%. La precisión global de la clasificación alcanzada también es máxima y llega al 99,32%.



Algoritmo	1-1	1-2	1-3	1-4	2-1	2-2	2-3	2-4	3-1	3-2	3-3	3-4	4-1	4-2	4-3	4-4	5-1	5-2	5-3	5-4
Exactitud global A	97,06	96,98	96,53	90,96	96,99	96,92	96,49	90,71	96,58	96,53	96,01	90,21	95,34	95,24	94,70	89,01	89,32	89,30	88,56	83,80
Exactitud global B	99,00	99,00	99,00	99,00	99,32	99,32	99,32	99,32	99,17	99,17	99,17	99,17	98,89	98,89	98,89	98,90	98,79	98,79	98,79	98,79
Tamaños bloques (m)	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10

Figura 152. Comparación de exactitud global alcanzada por los Algoritmos A y B. Detección del terreno sobre datos test 2.

La comparativa de las precisiones alcanzadas al ejecutar ambos algoritmos (Algoritmos A y B) con los mismos parámetros umbrales resulta de gran utilidad para determinar el algoritmo que mejores resultados obtiene, así como para analizar otros aspectos como la dependencia a los parámetros umbrales o el grado de automatización.

Referente a la precisión, tras ejecutar ambos algoritmos un total de veinte ocasiones variando los parámetros umbrales tanto para los datos test 1 como para los datos test 2, en la totalidad de casos (40 pruebas) el Algoritmo B consigue mejores resultados. Por consiguiente, puede concluirse sin lugar a dudas que el Algoritmo B es más preciso.

En cuanto al grado de automatismo y a la dependencia con respecto a los parámetros umbrales, dos aspectos han tomado gran relevancia a lo largo del análisis: el tamaño de los bloques necesarios para generar el terreno provisional y el parámetro umbral en desnivel utilizado en el proceso de densificación progresiva.

En el Algoritmo A existe una alta variabilidad de los resultados al utilizar distintos tamaños de bloques para generar el terreno provisional. No obstante, el Algoritmo B mantiene la precisión alcanzada en cada caso con independencia del tamaño de bloque utilizado. Esto se debe a que el algoritmo B incluye un proceso adicional previo a la generación del terreno provisional. En este proceso la aplicación analiza características de las regiones obtenidas tras segmentar la escena tales como el tamaño o la posición relativa entre regiones adyacentes. El análisis consigue una preselección de las agrupaciones que pueden pertenecer al terreno provisional eliminando la mayoría de las regiones que pertenecen a otras entidades como planos de techo o vegetación. Consecuentemente, el tamaño de los bloques deja de tener relevancia.

En efecto, tal y como se muestra para distintos ejemplos en la Figura 153, el Algoritmo B consigue una independencia total entre la precisión de los resultados y el tamaño de los bloques. En la Figura 153a se presenta la clasificación generada tras ejecutar los Algoritmos A y B utilizando los mismos parámetros sobre los datos test 1. En la Figura 153b se observan los resultados obtenidos tras ejecutar de nuevo ambos algoritmos utilizando los mismos parámetros, pero sobre los datos test 2. En ambos casos, se aprecia claramente cómo reducir el tamaño de los bloques o cuadrantes se transmite en un incremento del error de comisión si se ejecuta el Algoritmo A, pero no afecta a los resultados al utilizar el Algoritmo B. Esto constituye un mayor grado de automatización del proceso para el Algoritmo B, ya que no será necesario realizar un análisis visual de la zona de estudio con el fin de seleccionar un tamaño de bloque mayor que los edificios.

El Algoritmo A presenta también una alta dependencia de los resultados con respecto al parámetro umbral en desnivel. Ésto es debido a que el proceso de densificación considera como terreno cualquier punto que esté a menor distancia vertical de la superficie de referencia que el parámetro umbral establecido, por lo que si el valor es demasiado alto puntos pertenecientes a otros objetos son clasificados como terreno. Para solucionar este problema, el Algoritmo B incluye un análisis adicional dentro del propio proceso de densificación, de tal forma que un punto es clasificado como terreno si además de cumplir con el parámetro umbral en desnivel, se verifican una serie de condiciones en el análisis adicional. Con ello se consigue un Algoritmo B más robusto, con resultados más precisos y con una mayor independencia con respecto a los parámetros umbrales utilizados. En este sentido, la Figura 153 muestra cómo el Algoritmo B obtiene mejores resultados en la totalidad de casos propuestos.

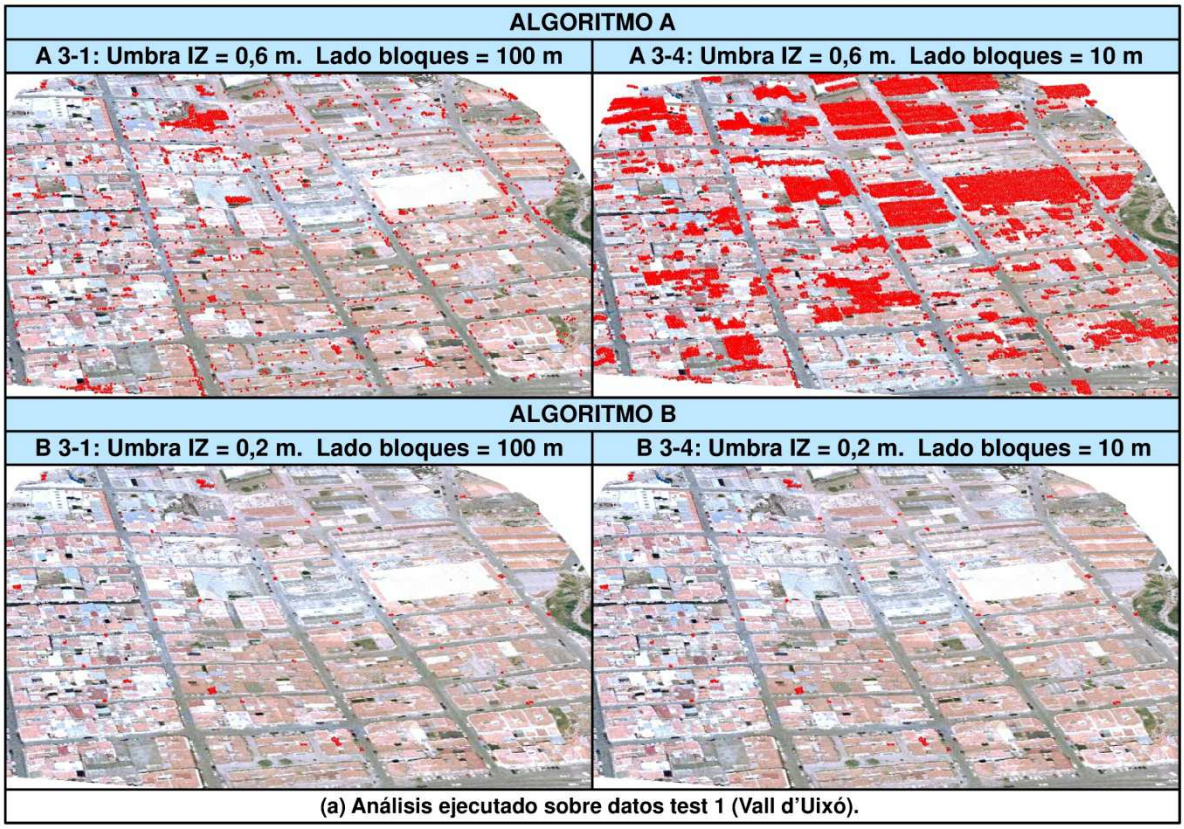


Figura 153. Imagen 3D. Precisión alcanzada por los Algoritmos A y B. Detección del terreno.

Referente a los errores de comisión, el análisis adicional que incorpora el Algoritmo B dentro del propio proceso de densificación permite reducir considerablemente el número de puntos pertenecientes a vegetación y otros objetos que son clasificados erróneamente como terreno. No obstante, la eliminación por completo de esta fuente de error no es posible y algunos puntos pertenecientes a vegetación de baja altura muy próximos al terreno son clasificados como tal. Además, los puentes constituyen un elemento artificial que está conectado al terreno por sus extremos. Consecuentemente, tras segmentar la escena son incluidos en la región de tierra adyacente y clasificados erróneamente como terreno. En los datos test 1 (Vall d'Uixó) no existe ningún puente. No obstante, en los datos test 2 (Collado Villalba) existen un total de seis puentes que constituyen una fuente de error importante. En cualquier caso, en el presente trabajo se ha desarrollado un algoritmo capaz de identificar los puentes y clasificarlos como entidad propia, consiguiendo eliminar esta fuente de error en la clase terreno para los datos test 2, momento en que se consigue reducir la media de errores del 0,66% a tan solo 0,47%.

En la Figura 154 pueden apreciarse las dos fuentes de error descritas para una zona concreta de los datos test 2.

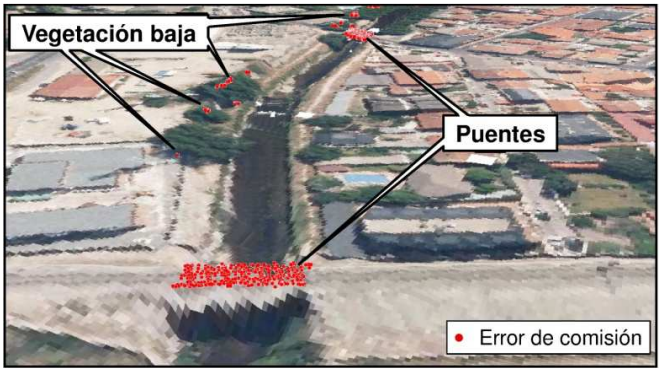


Figura 154. Errores de comisión en la clase terreno.

En cuanto a los errores de omisión, el método de densificación propuesto falla cuando existen desniveles pronunciados del terreno en distancias cortas. Ello se debe a que el enfoque planteado supone que el terreno es continuo y no presenta grandes discontinuidades dentro de entornos urbanos. No obstante, en la zona test 2 (Collado Villalba) existe un solar en construcción donde se han realizado importantes movimientos de tierra con el fin de construir los cimientos de un edificio (Figura 155). En este caso, los desniveles verticales no son clasificados como terreno al ser similares a las fachadas de los edificios. Además, una pequeña terraza de tierra situada a media altura es detectada por el análisis adicional como un plano de techo de edificio y por tanto no se incluye en la clase terreno. Afortunadamente, este tipo de situaciones no se dan con frecuencia en zonas urbanas y por tanto no constituye una fuente de error relevante.

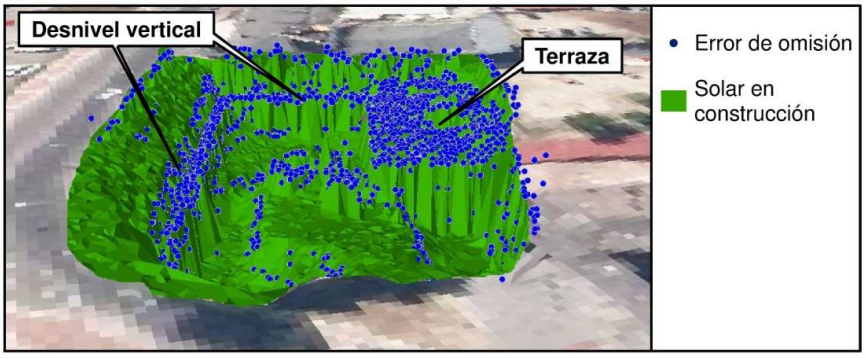


Figura 155. Errores de omisión en la clase terreno

La aplicación desarrollada combina diferentes métodos propuestos por varios autores. El proceso comienza con la segmentación de datos en superficies continuas de igual forma que otros algoritmos utilizados en este campo (Tovari y Pfeifer 2005; Filin y Pfeifer 2006; Lari et al. 2011). Seguidamente, la zona de estudio es dividida en cuadrantes o bloques cuadrados y la región que contiene el punto de menor altitud de cada uno de ellos es seleccionada para generar el terreno provisional, siendo este un procedimiento análogo al utilizado por algunos filtros morfológicos (Hyyppä et al. 2001; Popescu et al. 2002; Wack y Wimmer 2002; Yu et al. 2004 y Clark et al. 2004). Tomando como base el terreno provisional, un método de densificación progresiva se aplica con un enfoque iterativo, donde los puntos más próximos a la superficie de referencia son añadidos a la clase terreno. Este proceso es análogo al utilizado por los filtros de densificación progresiva de TIN (Axelsson 2000; Sohn y Dowman 2002; Ekhtari et al. 2008; Pérez et al. 2012). La principal diferencia radica en que el algoritmo propuesto extiende el terreno bajo los objetos urbanos utilizando la interpolación ponderada inversa a la distancia en lugar de usar una triangulación, siendo esta una característica propia de los filtros basados en interpolación de superficies (Kraus y Pfeifer 1998; Pfeifer et al. 2001; Briese et al. 2002).

Diferentes autores han presentado en los últimos años nuevas metodologías para detectar el terreno en entramados urbanos mediante el uso de tecnología LiDAR. Chehata et al. (2009) procesan la información completa de onda asociada a ecos múltiples para detectar puntos que cumplen con las características del terreno, alcanzando errores del 2,3% en zonas urbanas. Crosilla et al. (2011) utilizaron análisis estadísticos de altos momentos de orden secuencial para detectar puntos de tierra con un error que oscila entre el 0,4 y 4,8%, dependiendo de la zona de aplicación. Pérez et al. (2012) utilizaron un enfoque similar al propuesto en el presente trabajo. Los resultados mostraron mejoras de hasta el 15% con respecto al método clásico de densificación progresiva de TIN. En este sentido, al ejecutar el algoritmo propuesto sobre los datos Test 1 se obtiene una media de errores mínima del 1,88% al emplear el Algoritmo A, que se reduce a tan solo un 0,17% al utilizar el Algoritmo B. Para los datos test 2, el algoritmo A obtiene un error mínimo del 2,78%, valor que se reduce hasta un 0,66% al ejecutar el Algoritmo B.

Tras el análisis de resultados, la muy alta precisión alcanzada por el Algoritmo B queda patente, consiguiendo superar las precisiones obtenidas mediante otras metodologías reflejadas en los diferentes estudios existentes.

IV. 3. Detección de puentes

En el presente estudio, el algoritmo utilizado para detectar el terreno combina un método de densificación progresiva con técnicas de segmentación. Como consecuencia, tras segmentar la escena, dado que no existen discontinuidades entre los extremos de un puente y el terreno adyacente, los puentes son incluidos automáticamente en regiones asociadas al terreno con el error que ello conlleva. Para eliminar este error de comisión en la clase terreno, dos algoritmos para detectar y clasificar puentes como una entidad propia se han desarrollado.

Con el fin de evaluar ambos enfoques, los algoritmos desarrollados utilizados en la detección de puentes tomarán como dato de entrada la clase terreno obtenida tras aplicar el Algoritmo B (detección del terreno) con un parámetro umbral en desnivel de 0,4 m, por ser el que mejores resultados alcanza.

Ambos algoritmos se basan en identificar características geométricas y estructurales de los puentes en el conjunto de datos LiDAR. El primero de ellos (Algoritmo A), detecta los bordes de los puentes y posteriormente analiza las características geométricas de los puntos comprendidos entre ellos con el fin de determinar si pertenecen a un puente.

La identificación de bordes es un proceso relativamente simple, donde el algoritmo realiza una búsqueda de pendientes verticales dentro de la clase terreno. Para ello, el vecindario cilíndrico de cada punto del terreno es analizado y cambios bruscos de altitud son detectados. Tras localizar los bordes, la aplicación analiza distintas características de los puntos del terreno comprendidos entre bordes opuestos, tales como el desnivel con respecto a los puntos de borde próximos, así como la continuidad estructural en la dirección principal del puente.

En la detección de bordes, tres parámetros umbrales deben definirse previamente. Por un lado, el vecindario cilíndrico para analizar la existencia de desniveles verticales no debe de abarcar una superficie excesiva con el fin de evitar falsos positivos, por lo que se opta por un vecindario cilíndrico de 2 m de radio. Con el fin de determinar si el punto está situado en la parte superior de un talud vertical, un umbral en desnivel para identificar pendientes verticales debe de establecerse. En este caso, tras realizar diferentes pruebas y analizar los resultados de forma cualitativa (análisis visual) se observa una alta dependencia de los resultados con respecto al desnivel umbral utilizado, por lo que distintos parámetros umbrales se han probado y la precisión alcanzada para cada uno de ellos se expone en el presente apartado. Por último, un umbral en porcentaje es necesario para asegurar la existencia de un desnivel vertical. Este parámetro no tiene una gran influencia en la precisión alcanzada y tras realizar diferentes pruebas experimentales se opta por un parámetro umbral del 20%.

Respecto al posterior análisis para determinar si un punto situado entre bordes opuestos pertenece a un puente, un nuevo vecindario cilíndrico se utiliza cuyo radio debe de ser mayor a la mitad de la anchura del mayor de los puentes existentes en la zona de estudio. Consecuentemente, se opta por un radio de 18 m al considerar que no existen puentes de más de 36 m de ancho en los datos test. Por otro lado, el umbral en desnivel para determinar si el punto tiene una altitud similar a la de los bordes del puente no debe de ser excesivo y finalmente se utiliza un desnivel de 0,5 m. Referente a la amplitud de los sectores utilizados para analizar si un punto está situado entre bordes opuestos y para analizar la continuidad estructural en la dirección principal del puente, se precisa una pequeña amplitud para obtener una precisión alta, por lo que se opta por sectores de 10^g de amplitud. En cuanto a los semisectores necesarios para establecer si existe continuidad estructural en la dirección principal del puente se utiliza un radio de 6 m y un parámetro umbral en porcentaje alto (95%) para asegurar que no se producen falsos positivos.

Tras realizar numerosas pruebas se aprecia que el parámetro umbral que mayor influencia tiene en los resultados es el desnivel umbral utilizado para identificar pendientes verticales en el proceso de detección de bordes. Por tanto, el algoritmo se ha ejecutado un total de cinco veces con distintos parámetros umbrales. En el presente apartado se analizará la precisión de los resultados obtenidos en cada caso y se identificarán los errores ocasionados. Además, se establecerá el grado de dependencia del algoritmo con respecto a los parámetros umbrales utilizados. Dichos parámetros se muestran en la Figura 156.

DETECCIÓN DE BORDES:

VECINDARIO CILÍNDRICO PARA DETECTAR CAMBIOS BRUSCOS EN DESNIVEL (DETECCIÓN DE BORDES): 2 m

UMBRAL EN PORCENTAJE DE PUNTOS QUE CUMPLEN EL IZ UMBRAL PARA DETECTAR BORDES: 20 %

UMBRAL EN DESNIVEL: Se ejecuta el algortimo cinco veces con diferentes parámetros umbrales:

A 1: 1 m A 2: 2 m A 3: 3 m A 4: 3,5 m A 5: 4 m

DETECCIÓN DE PUENTES A PARTIR DE LOS BORDES DETECTADOS:

RADIO DE SECTORES: 18 m

RADIO DE SEMISECTORES INTERIORES: 6 m

IZ UMBRAL CON RESPECTO A BORDES PARA CONSIDERAR QUE UN PUNTO PERTENECE A UN PUENTE: 0,5 m

UMBRAL EN % DE PUNTOS QUE CUMPLEN UMBRAL IZ EN SEMISECTORES INTERIORES: 95 %

AMPLITUD DE LOS SECTORES DEL CLASIFICADOR ANGULAR MEDIANTE SECTORES: 10 g

Figura 156. Detección de puentes. Parámetros umbrales de algoritmo A.

El algoritmo únicamente se ha ejecutado sobre los datos test 2 (Collado Villalba), ya que los datos test 1 no contienen puente alguno. En la Figura 157 aparecen los errores de comisión, omisión y media de errores para cada uno de los parámetros umbrales en desnivel utilizados. Además, un gráfico muestra la exactitud global de la clasificación obtenida en cada caso. El mayor error de comisión se produce cuando el parámetro umbral en desnivel es menor (1 m) donde alcanza un valor del 93,97%. No obstante, este error disminuye drásticamente al aumentar el desnivel umbral hasta el punto de desaparecer por completo al usar un desnivel superior a 3,5 m. El error de omisión tiene un comportamiento opuesto y aumenta al incrementar el parámetro umbral en desnivel alcanzando un error máximo del 41,89% al utilizar un desnivel de 4 m (A 5).

Los mejores resultados se obtienen al utilizar un umbral en desnivel de 3 m, momento en el que la media de errores consigue un mínimo de un 11,17% y la exactitud global de la clasificación alcanza el 99,49%.

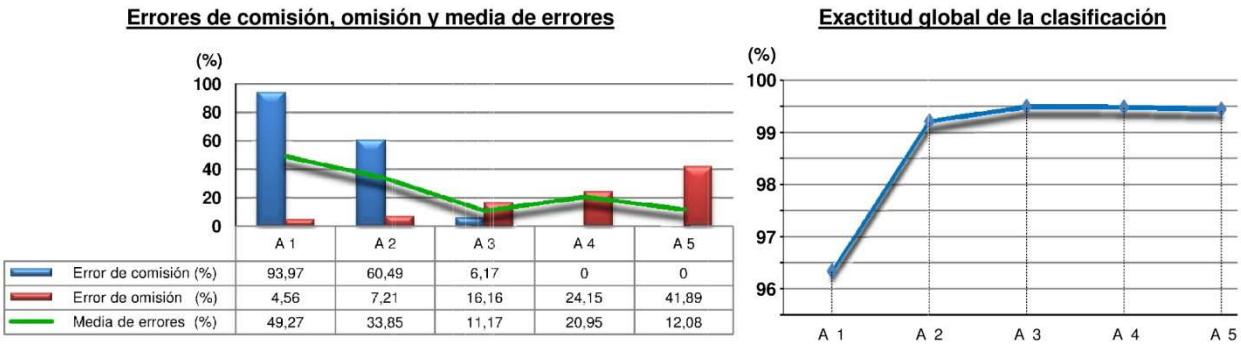


Figura 157. Gráfico de errores y exactitud global de la clasificación. Detección de puentes (Algoritmo A). Datos test 2.

En cuanto a los resultados obtenidos cabe resaltar la influencia del desnivel umbral utilizado en la detección de bordes respecto a los errores ocasionados. En el caso de usar un pequeño desnivel como parámetro umbral, las cabezas de talud de pendientes pronunciadas se detectan por error como bordes de puentes, lo que conlleva que posteriormente puntos situados entre dichos bordes sean erróneamente clasificados como puentes. Un incremento del parámetro umbral en desnivel disminuye esta fuente de error hasta conseguir eliminarla por completo (error de comisión). El error de omisión tiene un comportamiento contrario al de comisión y aumenta al incrementar el umbral en desnivel. Ésto se debe a que la altura del puente con respecto al terreno es mayor en su parte central y disminuye hacia sus extremos llegando a ser nula en el principio e inicio del puente donde existe conexión con el terreno. Por consiguiente, un parámetro umbral en desnivel demasiado alto evita la correcta detección de bordes de puentes próximos al inicio y fin de los mismos, y produce un importante error de omisión en la consiguiente clasificación. Todo ello puede observarse en la Figura 158. En ella se muestran los bordes detectados y los errores en la clasificación obtenidos al utilizar tres parámetros umbrales diferentes (1 m, 3 m y 4 m). En este sentido, puede apreciarse claramente cómo al aumentar el desnivel umbral disminuye el error de comisión hasta desaparecer y cómo el error de omisión tiene una presencia más importante al utilizar los mayores desniveles.

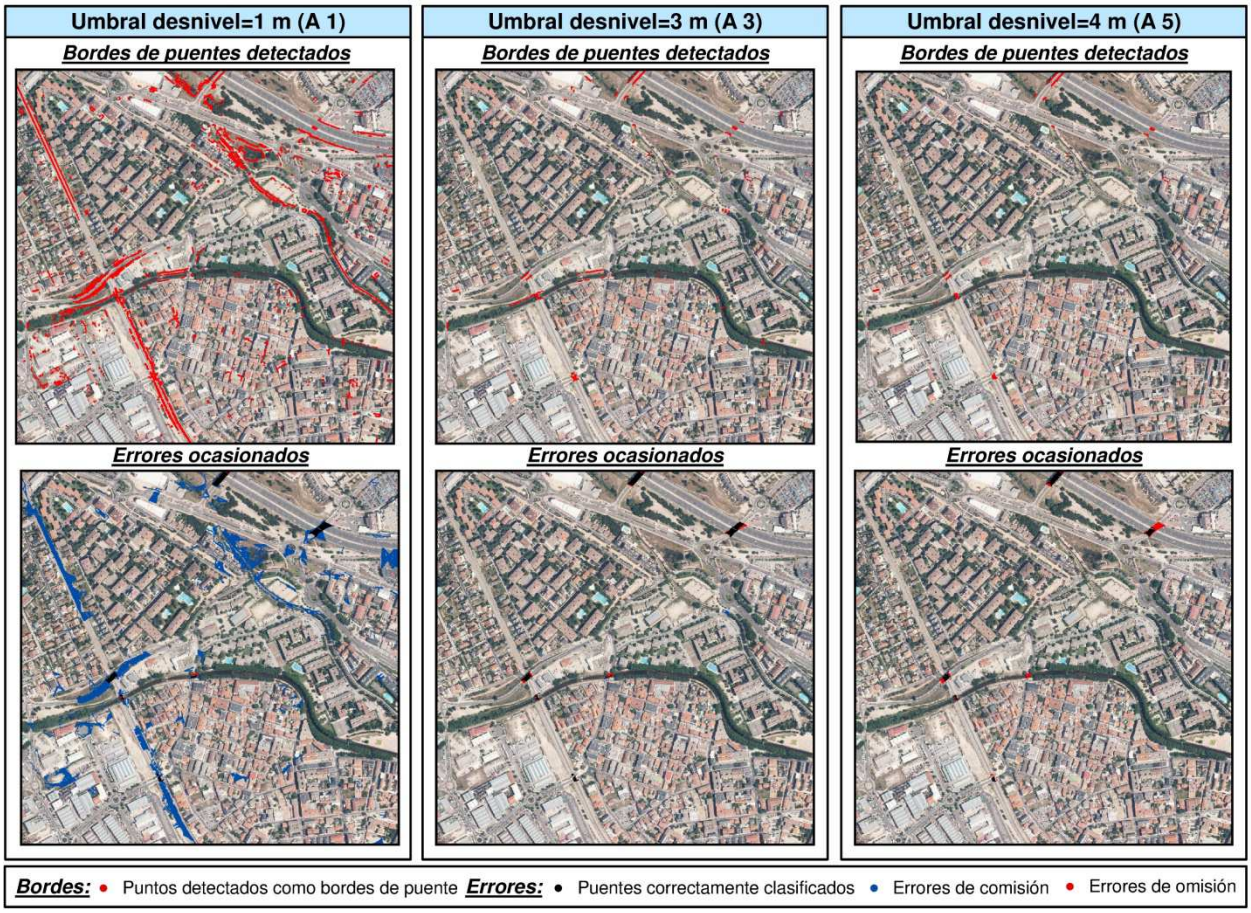


Figura 158. Detección de puentes. Comparación de resultados al usar diferentes umbrales en desnivel y detección de bordes. Algoritmo A. Datos test 2.

Tras analizar en detalle el error de comisión que se produce al utilizar un pequeño desnivel como parámetro umbral, puede afirmarse que en la totalidad de casos el error se deriva de una incorrecta detección previa de los bordes de puentes. En la Figura 159 aparecen dos zonas de detalle pertenecientes a los datos test 2 donde existen errores de este tipo. El parámetro umbral en desnivel utilizado en el ejemplo propuesto es de 1 m (A 1). Para ambas zonas de detalle se muestran los bordes detectados en la parte izquierda y los errores ocasionados en la derecha. En la primera zona de detalle una línea de ferrocarril discurre en altura sobre un pronunciado terraplén; en este caso, la mayor parte del borde del terraplén es detectado como borde de puente (imagen izquierda) lo que ocasiona un importante error de comisión al clasificar con posterioridad los puentes (imagen derecha). En la segunda zona de detalle ocurre una situación similar, en este caso una carretera cruza la línea de ferrocarril mediante un puente. A ambos lados del puente existe un alto terraplén necesario para conseguir la altura apropiada del vial en los extremos de la estructura, por lo que el borde de la cabeza de talud nuevamente es erróneamente detectado como borde de puente (imagen izquierda), lo que conlleva que parte de la carretera sea clasificada como puente (imagen derecha).

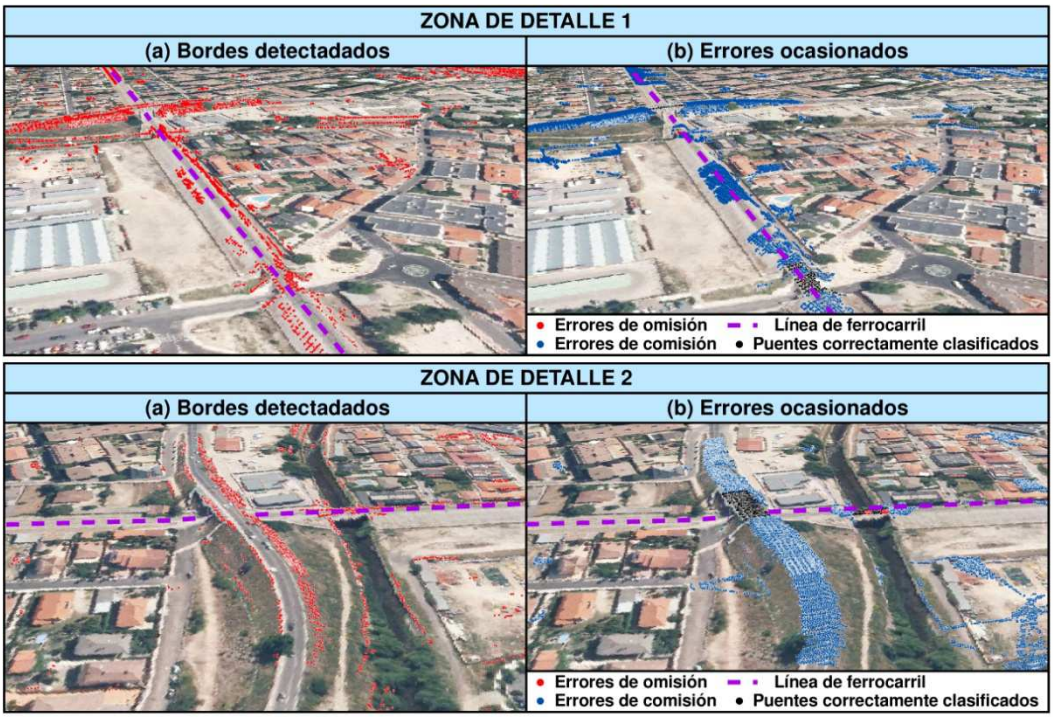


Figura 159. Imagen 3D de bordes detectados y errores en la clasificación de los puentes. Algoritmo A (A 1).

De forma análoga a los errores de comisión, un error de omisión en la detección de bordes ocasiona errores de omisión en la posterior clasificación de puentes. En este sentido, existen dos tipos de errores de omisión a considerar.

El primero de los errores de omisión (error de tipo 1) está asociado a la selección de un umbral en desnivel demasiado alto. Ello es debido a que en el inicio y el fin de los puentes su estructura conecta con el terreno natural y, por tanto, el desnivel entre el puente y el terreno disminuye hasta ser nulo en los extremos. En consecuencia, la utilización de un desnivel alto como parámetro umbral tiene como resultados errores por defecto en el principio y fin de los puentes. Este tipo de errores se muestra en la Figura 160, donde los bordes de los puentes detectados y los errores producidos aparecen en dos imágenes de detalle. En el ejemplo propuesto se ha utilizado un parámetro umbral en desnivel de 4 m (A 5).

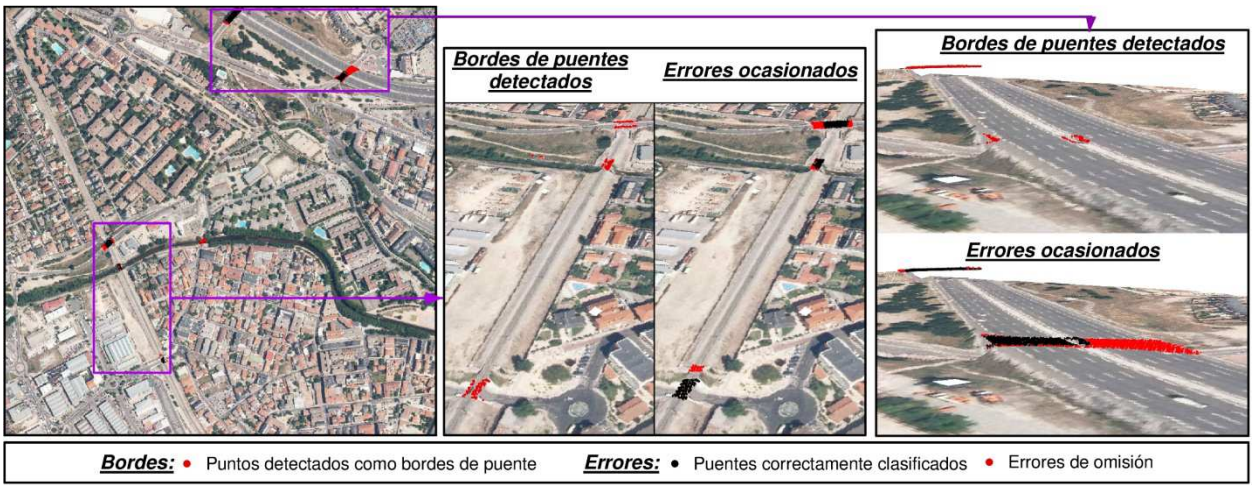


Figura 160. Errores de omisión en la detección de los puentes. Algoritmo A (A 5).

El segundo tipo de error por omisión (error de tipo 2) que nos encontramos está relacionado con la absorción que sufren los pulsos LiDAR por parte del agua. Resulta frecuente que los puentes crucen ríos u otros flujos de agua. En estos casos, el haz láser es absorbido en su mayor parte por la lámina de agua y no se consigue información tridimensional del entorno (zonas de sombra). En los datos test 2 encontramos un puente que cruza un canal de agua y debido a la absorción de los pulsos láser prácticamente no existen puntos LiDAR en el interior del canal (Figura 161a). En el proceso de detección de bordes, el algoritmo detecta desniveles verticales en entornos próximos dentro de la clase terreno. No obstante, debido a la inexistencia de información tridimensional junto a los bordes del puente, prácticamente ningún punto de borde es detectado (Figura 161b), lo que produce un alto error de omisión en la posterior clasificación del puente (Figura 161c).

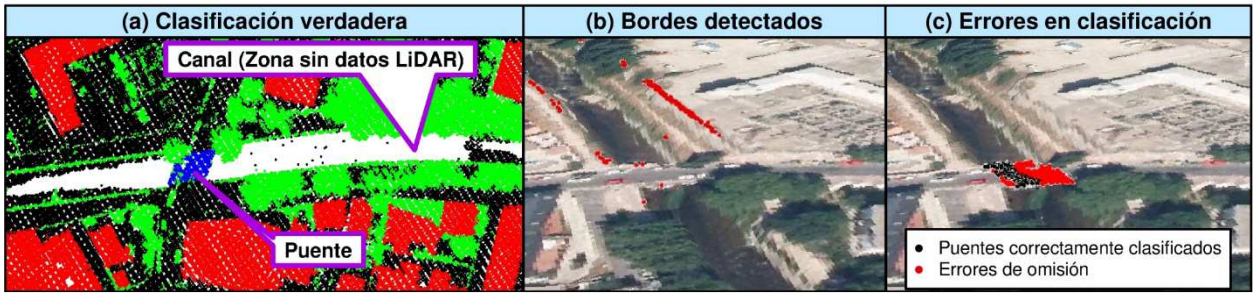


Figura 161. Errores de omisión en la detección de los puentes. Zonas de sombra. Algoritmo A (A 3).

Con el fin de solucionar los dos tipos de error de omisión anteriormente descritos, en el presente estudio se ha desarrollado un segundo algoritmo (Algoritmo B). Para reducir el error de tipo 1, el Algoritmo B detecta en primera instancia los bordes situados en la parte central de los puentes mediante la utilización de un alto desnivel como parámetro umbral. Seguidamente, para detectar los puntos de borde más alejados de la zona central se itera el proceso con un parámetro umbral en desnivel más relajado. No obstante, con el fin de no provocar nuevos errores de comisión, la iteración del proceso únicamente se aplica en las proximidades de los bordes previamente detectados.

Los parámetros umbrales asociados a la detección de bordes de puente para ambos procesos son los mismos que los utilizados por el Algoritmo A: únicamente varía el desnivel umbral utilizado en cada uno de ellos. Dado que el primer proceso detecta los bordes asociados a la parte central del puente, y el segundo los puntos que pertenecen al borde próximo a los extremos, el primer proceso siempre utilizará un parámetro umbral en desnivel más restrictivo. En total se han realizado seis pruebas con los parámetros umbrales que se muestran en la siguiente figura (Figura 162).

B 3-1:	PROCESO 1: IZ UMBRAL= 3 m PROCESO 2: IZ UMBRAL= 1 m	**B 3-2:**	PROCESO 1: IZ UMBRAL= 3 m PROCESO 2: IZ UMBRAL= 2 m	**B 4-1:**	PROCESO 1: IZ UMBRAL= 3,5 m PROCESO 2: IZ UMBRAL= 1 m
B 4-2:	PROCESO 1: IZ UMBRAL= 3,5 m PROCESO 2: IZ UMBRAL= 2 m	**B 5-1:**	PROCESO 1: IZ UMBRAL= 4 m PROCESO 2: IZ UMBRAL= 1 m	**B 5-2:**	PROCESO 1: IZ UMBRAL= 4 m PROCESO 2: IZ UMBRAL= 2 m

Figura 162. Detección de puentes. Parámetros umbrales de algoritmo B.

Referente a la detección de bordes de puentes que son colindantes a las zonas de sombra (error de tipo 2), el Algoritmo B genera el vecindario cilíndrico vertical para cada punto del terreno y lo divide en sectores. Seguidamente, si existe algún sector que no contiene ningún pulso LiDAR significa que existe una zona de sombra colindante. Tras realizar diferentes

pruebas, se concluye que un vecindario cilíndrico de 4 m de radio y sectores iguales de 50^g de amplitud ofrecen buenos resultados.

La Figura 163 muestra el gráficos de errores y de exactitud global de la clasificación obtenida tras ejecutar el Algoritmo B sobre los datos test 2 (Collado Villalba). Los gráficos asociados a la media de errores y a la exactitud global tienen forma de sierra y no muestra una tendencia clara. Ésto se debe a la gran influencia que tiene el parámetro umbral en desnivel sobre los resultados, ya que en cada prueba se han aplicado desniveles umbrales distintos en ambos procesos. Una vez más la dependencia de la precisión de los resultados respecto al umbral en desnivel se refleja en la variabilidad de la precisión alcanzada, donde la media de errores oscila un 18,13%. La mayor precisión se obtiene al ejecutar el primer proceso con un desnivel umbral de 3,5 m y el segundo proceso con 2 m de parámetro umbral en desnivel. En este caso se obtiene un error de comisión mínimo del 5,65% y el error de omisión de tan solo 2,34%, consiguiendo minimizar la media de errores al 3,99%, alcanzando una exactitud global de la clasificación del 99,52%.

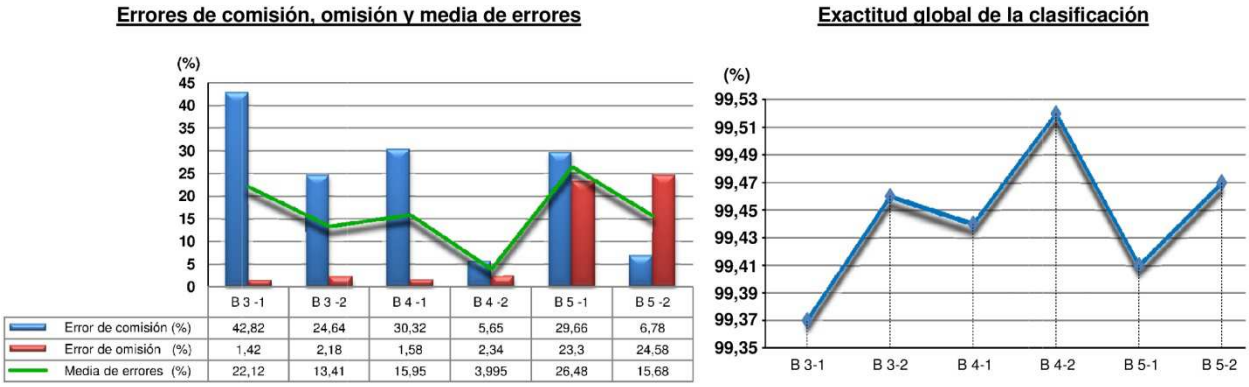
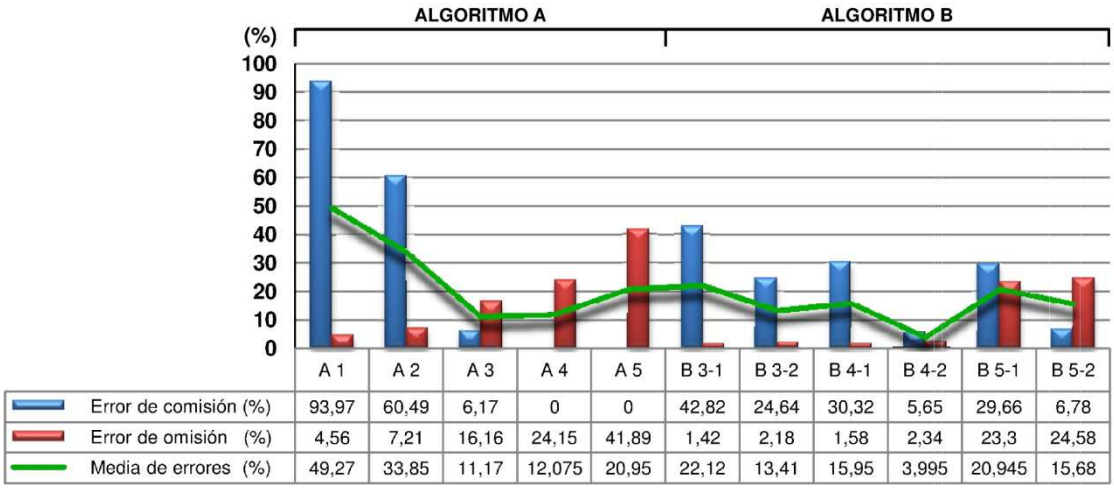


Figura 163. Gráfico de errores y exactitud global de la clasificación. Detección de puentes. Algoritmo B. Datos test 2.

La Figura 164 muestra un gráfico donde aparecen los errores asociados a los Algoritmos A y B. Los mayores errores se producen al aplicar un único proceso para detectar bordes (Algoritmo A). En este caso, un parámetro umbral en desnivel muy relajado ocasiona importantes errores de comisión que pueden llegar hasta el 93,97% (A 1). Por el contrario, a medida que se incrementa el desnivel umbral también aumenta el error de omisión obteniendo un máximo del 41,82% al utilizar el umbral en desnivel más alto (A 4). Este problema es solucionado en gran medida por el Algoritmo B. La ejecución de dos procesos de detección de bordes permite una primera aproximación donde los bordes de puentes son detectados mediante un parámetro umbral en desnivel restrictivo que evita errores de comisión altos. Para reducir los errores de omisión un segundo proceso se ejecuta sobre el vecindario próximo con un parámetro en desnivel más relajado. Además, la inclusión en el segundo proceso de un análisis multidireccional por sectores permite detectar bordes de puentes situados junto a zonas de sombra reduciendo el error por omisión en estos casos. Todo ello queda reflejado en el gráfico comparativo y su tabla adjunta (Figura 164). Mientras que el Algoritmo A produce una media de errores máxima del 49,27% (A 1). El máximo para el Algoritmo B se reduce en más de la mitad, alcanzando una media de errores del 22,12% (B 3-1). Por otro lado, el Algoritmo A alcanza su precisión máxima al utilizar un umbral en desnivel de 3 m con una media de errores del 11,17%. No obstante, el Algoritmo B consigue una precisión mayor y reduce la media de errores a tan solo 3,99% (B 4-2).

En cuanto a la exactitud global de la clasificación, llegado a este punto los datos test estarán organizados en tres clases: terreno, puentes y resto de entidades. Para esta

clasificación el Algoritmo A alcanza una precisión máxima del 99,49% (A 3) que es superada al aplicar el Algoritmo B con un 99,52% (B 4-2).



	A 1	A 2	A 3	A 4	A 5	B 3-1	B 3-2	B 4-1	B 4-2	B 5-1	B 5-2
Error de comisión (%)	93,97	60,49	6,17	0	0	42,82	24,64	30,32	5,65	29,66	6,78
Error de omisión (%)	4,56	7,21	16,16	24,15	41,89	1,42	2,18	1,58	2,34	23,3	24,58
Media de errores (%)	49,27	33,85	11,17	12,075	20,95	22,12	13,41	15,95	3,995	20,945	15,68

Figura 164. Comparación de media de errores entre Algoritmos A y B. Detección del terreno sobre datos test 2.

En la Figura 165 se muestran los mejores resultados obtenidos tras aplicar los Algoritmos A y B. Los resultados de mayor precisión al utilizar el Algoritmo A se consiguen con un parámetro umbral en desnivel de 3 m en la detección de bordes (A 3), mientras que el Algoritmo B alcanza la precisión más alta al utilizar un umbral en desnivel de 3,5 m para el primer proceso y 2 m para el segundo (B 4-2). En la imagen se muestran cinco zonas de detalle con los seis puentes que existen en la zona de estudio. En cada una de ellas se reflejan los errores de comisión y omisión. Para todos los puentes, el Algoritmo B consigue reducir considerablemente el error de omisión e incluso eliminarlo por completo, mientras que el error de comisión prácticamente es inexistente.

Dos nuevos algoritmos para detectar puentes se han presentado en el presente trabajo. El primero de ellos (Algoritmo A) detecta los bordes de los puentes y posteriormente clasifica la totalidad de la entidad mediante el análisis de distintas características asociadas a la propia estructura, tales como la situación de puntos entre bordes paralelos o la continuidad estructural en la dirección principal y perpendicular al puente. Los principales aportes del algoritmo con respecto a los trabajos existentes son dos: por un lado, en ningún caso los datos LiDAR brutos son interpolados a formato imagen; por otro, el análisis de las distintas características de los puentes se realiza en un único proceso, utilizando para ello el nuevo concepto de clasificador direccional por sectores. Los resultados muestran una alta precisión que puede alcanzar una media de errores mínima del 11,17%. No obstante, a partir de las diferentes pruebas realizadas queda patente la alta dependencia de la precisión obtenida con respecto al parámetro umbral en desnivel utilizado en la detección de bordes. El Algoritmo B consigue solucionar en gran medida este problema y la media de errores se reduce hasta obtener tan solo un 3,99%. Para ello, se ejecutan dos procesos con diferentes parámetros umbrales en desnivel. El primero de ellos detecta la parte central de los puentes mediante el uso de un parámetro umbral en desnivel alto, mientras que el segundo clasifica el resto a partir de una reducción del parámetro umbral en desnivel. La principal ventaja reside en que el segundo proceso se ejecuta sobre el entorno próximo a la clasificación obtenida tras ejecutar el primero, con lo que se evita la inclusión de errores de comisión. Además, existe un análisis adicional que permite detectar los bordes de aquellos puentes que son colindantes a las zonas de sombra consiguiendo así reducir los errores de omisión asociados. Este segundo algoritmo (Algoritmo B) aporta dos aspectos nuevos en este campo: por un lado, se plantea un algoritmo adaptativo que utiliza distintos parámetros umbrales para clasificar los puentes sin introducir errores de comisión; por otro, el análisis

adicional permite detectar puentes que discurren sobre el agua donde los pulsos láser son absorbidos y por tanto, no existe información tridimensional en las zonas adyacentes.

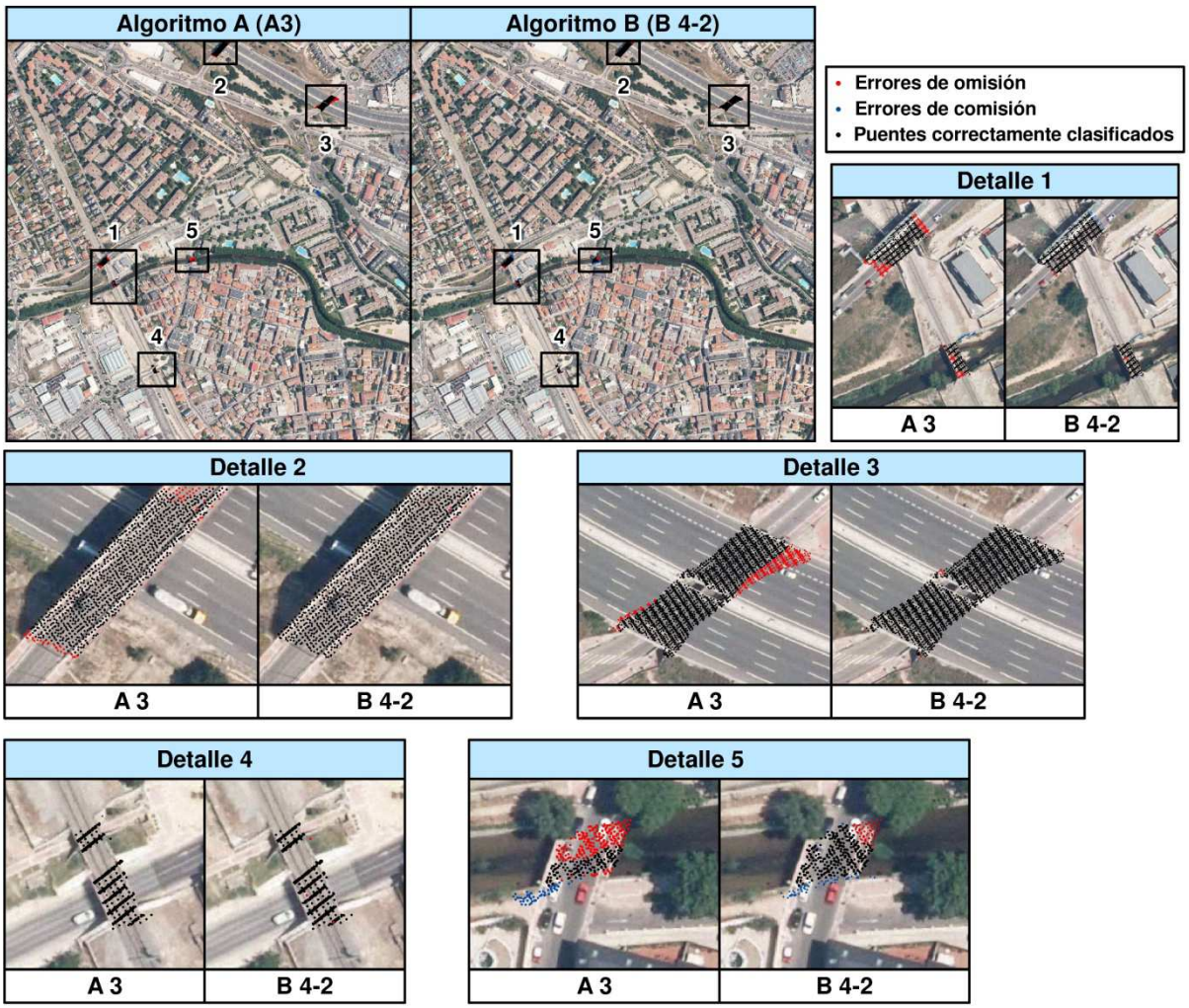


Figura 165. Comparativa de la precisión alcanzada. Algoritmos A y B. Detección de puentes.

En cuanto a la precisión alcanzada en trabajos existentes, al igual que en el presente estudio, Sithole y Vosselman (2006) plantean un algoritmo para detectar puentes que actúa sobre datos LiDAR donde la vegetación, los edificios y otros objetos se filtran con anterioridad. En este caso, el algoritmo redistribuye la información LiDAR en una malla regular de 0,5 m de resolución. Seguidamente, se detectan puntos pertenecientes a puentes mediante el análisis de cuatro perfiles aplicados en distintas direcciones. Para ello, el algoritmo tiene en cuenta que un punto perteneciente a un puente se eleva sobre el terreno en todas direcciones. A continuación, los puntos del puente detectados se utilizan como puntos semillas en una segmentación por proximidad que consigue detectar los bordes de los puentes y extraerlos como polígonos cerrados. Por último, los puntos que están dentro del contorno extraído se clasifican como puentes. En este estudio no se cuantifica la precisión alcanzada. No obstante, los autores indican que el algoritmo puede clasificar rampas como puentes por error. Además, no se da solución al problema de detectar puentes situados junto a zonas de sombra, tal y como consigue el algoritmo propuesto en el presente estudio.

Forsberg (2011) emplea un enfoque basado en el método propuesto por Sithole y Vosselman (2006). En este caso, los datos LiDAR son previamente interpolados a MDS para posteriormente filtrar la vegetación, los edificios y otros objetos. Básicamente, el algoritmo

realiza una búsqueda de desniveles verticales mediante la aplicación de perfiles en distintas direcciones y posteriormente los bordes de los puentes se detectan y extraen en forma poligonal, clasificando como puentes todos los puntos contenidos en los contornos detectados. En este caso los puentes se detectan con una precisión aproximada al 90%. En cualquier caso, debido al enfoque propuesto, el algoritmo no funciona en puentes demasiado anchos ni es capaz de detectar puentes adyacentes a zonas de sombra.

Al igual que en el resto de trabajos existentes, Evans (2008) no trabaja con los datos brutos LiDAR y utiliza un MDT derivado para detectar puentes. En este caso, el algoritmo detecta los bordes de los puentes y posteriormente analiza distintas características geométricas de los mismos (cambio de aspecto de bordes opuestos, variaciones en la anchura) para completar la clasificación. El algoritmo se ejecuta en una zona que contiene cinco puentes y cuatro de ellos se detectaron correctamente. En cualquier caso, no se cuantifica la precisión alcanzada ni los errores cometidos.

Zheng et al. (2013) utilizan un enfoque similar al propuesto donde se lleva a cabo una detección de bordes y un posterior análisis de características asociadas a los puentes, tales como paralelismo entre bordes y análisis de continuidad estructural. En este trabajo los resultados se comparan con una clasificación manual realizada de forma visual por un operador. La totalidad del proceso se realiza sobre los datos LiDAR interpolados a MDS consiguiendo clasificar correctamente el 95% de los píxeles asociados a puentes, pero sin entrar en detalle sobre los errores de comisión producidos.

Al contrario que ocurre en los trabajos existentes, la metodología propuesta en el presente estudio utiliza directamente datos LiDAR brutos dispuestos de forma aleatoria, consiguiendo evitar así la perdida de información ligada al proceso de generalización que se produce al interpolar los datos. Además, el algoritmo B aporta un nuevo enfoque capaz de detectar bordes de puentes que discurren sobre agua y por tanto son adyacentes a zonas de sombra. Cabe resaltar también las ventajas del clasificador direccional por sectores propuesto, que permite en un solo análisis analizar distintas características tales como detección de bordes opuestos y determinación de la continuidad estructural en la dirección principal y transversal del puente. Por último, notar la alta precisión alcanzada ya que el Algoritmo A (A 3) consigue clasificar correctamente el 83,4% de los puntos de puente y el Algoritmo B (B 4-2) aumenta considerablemente la precisión y consigue clasificar un total del 97,66% de los puntos pertenecientes a puentes, consiguiendo superar sensiblemente la precisión alcanzada en otros estudios.

IV. 4. Detección de vegetación y edificios

La clasificación de los edificios y de la vegetación en clases independientes es una tarea de gran relevancia en el procesado de datos LiDAR. Con este fin, se han desarrollado distintos enfoques a lo largo de los últimos años. Una de las líneas de investigación que mayores logros ha alcanzado en este campo consiste en segmentar los datos y posteriormente aplicar análisis de texturas LiDAR (Hug 1997; Brunn y Weidner 1998; Maas 1999; Oude Elberink y Maas 2000). También resulta de gran utilidad la información multi-retorno asociada a los pulsos Lidar para identificar la cobertura vegetal y diferenciarla de los edificios (Pfeifer et al. 2001; Alharthy y Bethel 2002; Tovari y Vögtle 2004; Meng, Wang y Currit 2009). Por último, la combinación de datos LiDAR con otras fuentes de datos es una técnica muy común para este propósito, especialmente cuando se dispone de información multiespectral y es posible identificar la vegetación mediante cálculo del índice de vegetación normalizado (Vögtle y Steinle 2000; Matikainen et al. 2010; Hartfield et al. 2011; Moussa y Sheimy 2012; Wang y Li 2013).

En el presente estudio se han desarrollado dos algoritmos que combinan distintos enfoques como son el análisis de textura, el estudio de múltiples retornos de la señal o la segmentación de la escena y el análisis de las características de las regiones obtenidas. En este apartado, se analizarán los resultados obtenidos por los dos algoritmos propuestos tanto para los datos test 1 (Vall d'Uixó) como para los datos test 2 (Collado Villalba). Además de analizar la precisión de los resultados, se identificarán los errores concretos que ocasiona cada uno de los algoritmos, así como su dependencia con respecto a los parámetros umbrales utilizados.

La detección de edificios y vegetación es el último de los procesos de clasificación y se ejecuta tras disponer del terreno y los puentes previamente clasificados. Con el fin de evaluar los dos algoritmos propuestos, se utilizará como datos de entrada la clase terreno obtenida tras ejecutar el Algoritmo B (detección del terreno) con un parámetro umbral en desnivel de 0,4 m, así como los puentes detectados al ejecutar el Algoritmo B 4-2 (detección de puentes), por ser los resultados que mayor precisión alcanzan.

El primer algoritmo (Algoritmo A) utiliza análisis de texturas con el fin de detectar puntos pertenecientes a vegetación y otros pequeños objetos y separarlos de los edificios. Los trabajos existentes que siguen este enfoque interpolan la nube tridimensional LiDAR a malla regular (GRID) con el fin de facilitar la aplicación de filtros. No obstante, al igual que el resto de algoritmos desarrollados en la presente investigación, la aplicación desarrollada está diseñada para actuar directamente sobre los datos LiDAR brutos dispuestos de forma aleatoria. En este sentido, un sistema basado en la subdivisión del vecindario cilíndrico en sectores y semisectores se ha diseñado con el fin de permitir un análisis en cuanto a la densidad y distribución de las regiones de pequeño tamaño.

Básicamente, el algoritmo segmenta la escena en superficies continuas y posteriormente, teniendo en cuenta que las zonas de la vegetación están formadas por conglomerados de pequeñas regiones, la aplicación recorre los puntos que pertenecen a las regiones de menor tamaño y ejecuta un análisis de textura sobre el vecindario de cada uno de ellos. Este análisis consiste en dividir el vecindario en sectores y semisectores de igual amplitud y posteriormente determinar si existe una alta densidad de regiones de pequeño tamaño dentro de cada sector. En caso afirmativo, se comprueba si la distribución de estas regiones es homogénea para los distintos semisectores en los que se divide el espacio más próximo.

Para llevar a cabo este análisis, diversos parámetros umbrales deben definirse. Tras realizar distintas pruebas y analizar de forma cualitativa los resultados (análisis visual), se determina que un vecindario cilíndrico de 6 m de radio es adecuado. Además, la división del vecindario es idónea cuando se utilizan sectores iguales de 50^g de amplitud. Los semisectores es un sistema para dividir los sectores en dos partes de superficie similar. Por consiguiente, se opta por utilizar un radio de 3,5 m para los semisectores interiores. En cuanto al tamaño máximo de las regiones para ser incluidas en el análisis, siguiendo las mismas pautas utilizadas por Moussa y El-Sheimy (2012), se usa un umbral de 10 m^2. Finalmente, para determinar si existe una distribución homogénea de las regiones de pequeño tamaño en los sectores, deberá de existir un número mínimo de 10 puntos pertenecientes a dichas regiones en cada uno de los semisectores.

Tras realizar distintas pruebas se observa que el parámetro umbral que más influye en la precisión de los resultados es el porcentaje de puntos pertenecientes a las regiones pequeñas utilizadas para determinar si existe una alta densidad de regiones de pequeño tamaño en cada uno de los sectores. Un umbral demasiado alto ocasiona importantes errores de omisión, mientras que un umbral en exceso relajado produce la clasificación de pequeños planos de techo como vegetación. Con el fin de establecer el valor del parámetro que mejores resultados aporta se han realizado cuatro pruebas sobre los datos test 1 y 2. Los parámetros umbrales utilizados en las diferentes pruebas se muestran en la Figura 166.

PARÁMETROS UMBRALES FIJOS PARA ALGORITMO A:

RADIO DE VECINDARIO CILÍNDRICO PARA ANALIZAR TEXTURA: 6 m
RADIO DE LOS SEMISECTORES INTERIORES: 3,5 m
TAMAÑO DE REGIÓN UMBRAL: 10 m^2
AMPLITUD ANGULAR DE LOS SECTORES: 50 g
NÚMERO MÍNIMO DE PUNTOS PERTENECIENTES A REGIONES PEQUEÑAS EN SEMISECTORES: 10

PARÁMETROS UMBRALES VARIABLES PARA ALGORITMO A:

A 1: UMBRAL % = 70 % **A 2:** UMBRAL % = 80 % **A 3:** UMBRAL % = 90 % **A 4:** UMBRAL % = 99 %

Figura 166. Detección de vegetación y edificios. Parámetros umbrales del Algoritmo A.

El proceso se ejecuta sobre los resultados obtenidos tras aplicar los algoritmos de clasificación previos. Llegado a este punto, los datos LiDAR están clasificados en tres clases: terreno, puentes y resto de entidades. En esta última clase (resto de entidades) es donde actúa la aplicación con el fin de separar los edificios de la vegetación y otros pequeños objetos. Básicamente, el proceso consiste en identificar la cobertura vegetal, así como otros pequeños objetos y clasificarlos como "vegetación", de tal forma que al finalizar el proceso, aquellos puntos que no hayan sido etiquetados en clase alguna se incluyen automáticamente en la clase "edificios". Debido al sistema propuesto, los errores de comisión y omisión de las clases edificios y vegetación están relacionados. Un aumento del error de comisión en la clase de vegetación se corresponde con un incremento del error de omisión de la clase edificios y viceversa. En cualquier caso, la variación de errores no se produce con el mismo valor en porcentaje, ya que ello depende del número total de puntos que contiene cada clase.

La Figura 167 muestra las clasificaciones verdaderas para ambas zonas test. Con respecto a los datos test 2 (Collado Villalba) podemos apreciar cómo el volumen de vegetación es similar al de los edificios. En este caso encontramos vegetación que se presenta de forma puntual (árboles o setos aislados) así como densas zonas arboladas. Por el contrario, en los datos test 1 (Vall d'Uixó) hay escasa presencia de vegetación y únicamente encontramos algún árbol aislado. Dado que en este caso la proporción de puntos en edificios es muy superior a los de vegetación, variaciones en los errores producidos al obtener la clase vegetación mediante la utilización de distintos parámetros umbrales, no afectarán de forma relevante a los errores de la clase edificio, que permanecerán prácticamente constantes.

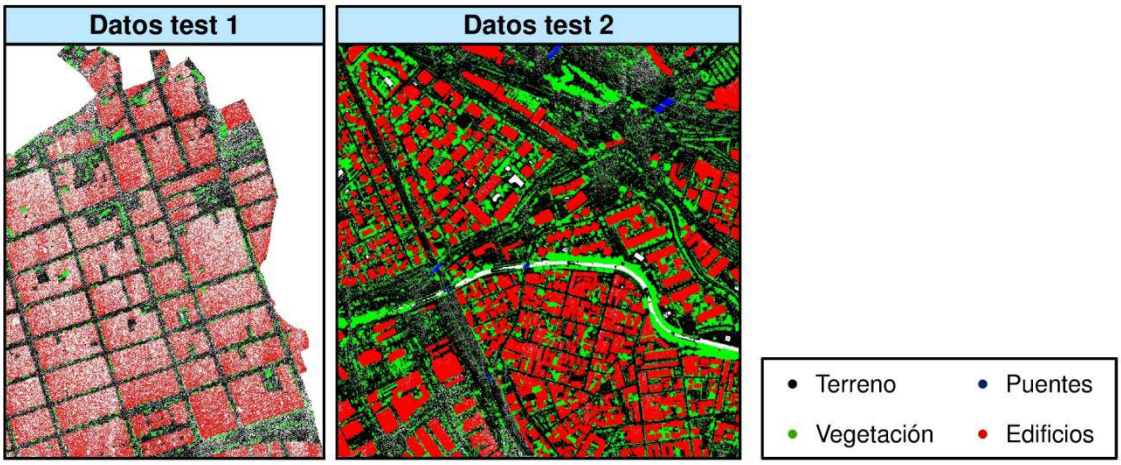
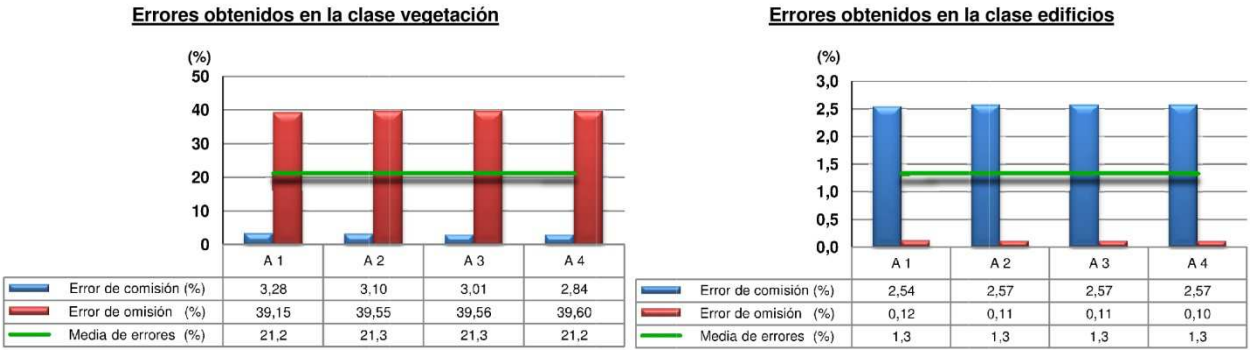


Figura 167. Clasificación verdadera para los datos test 1 y 2.

Los errores obtenidos al ejecutar el análisis de texturas con los diferentes parámetros umbrales sobre los datos test 1 (Vall d'Uixó) se muestran en la Figura 168. En los gráficos se representan los errores de omisión, comisión y media de errores para las clases de

vegetación y edificios. El primer aspecto que llama la atención es la alta diferencia en porcentaje de errores entre ambas clases. Mientras que en los edificios la media de errores es en todos los casos del 1,3%, en la clase vegetación está en torno al 21,3%. Está diferencia radica en la poca presencia de vegetación dentro de los datos test. En estos casos, el proceso de clasificación detecta un número muy reducido de puntos de vegetación comparado con el número total de puntos de construcciones y, por tanto, los errores o aciertos asociados a la vegetación tienen una repercusión mínima en la precisión obtenida para la clase edificio. Otro aspecto a tener en consideración es la poca variabilidad de la precisión de los resultados al utilizar distintos parámetros umbrales. De nuevo es debido a las características de los datos test 1 (Vall d'Uixó). Dado que la escasa vegetación se presenta principalmente como pequeños árboles aislados, en la mayoría de los casos existirán puntos de tierra en todos los sectores y, por tanto, se ejecutará el análisis 1 (Apartado III.8.2). Este análisis no tiene en cuenta los puntos pertenecientes al terreno para el estudio de la densidad de pequeñas regiones en el vecindario. Además, tampoco analiza si existe una distribución homogénea de dichas regiones en el conglomerado, lo que permite obtener unos resultados robustos que presentan una baja dependencia con respecto a los parámetros umbrales.



	A 1	A 2	A 3	A 4
Error de comisión (%)	3,28	3,10	3,01	2,84
Error de omisión (%)	39,15	39,55	39,56	39,60
Media de errores (%)	21,2	21,3	21,3	21,2

	A 1	A 2	A 3	A 4
Error de comisión (%)	2,54	2,57	2,57	2,57
Error de omisión (%)	0,12	0,11	0,11	0,10
Media de errores (%)	1,3	1,3	1,3	1,3

Figura 168. Gráfico de errores en la detección de edificios y de vegetación. Algoritmo A. Datos test 1 (Vall d'Uixó).

En cuanto a la precisión alcanzada, el análisis de texturas obtiene una alta precisión en la detección de edificios. Para esta clase, el error que se presenta en mayor medida independientemente de los parámetros umbrales utilizados es el de comisión. No obstante, los errores producidos se mantienen prácticamente constantes con una media de errores de tan solo el 1,3%. Por el contrario, el error que mayor presencia tiene en la clase vegetación es el de omisión, alcanzando un 39,6% al utilizar un umbral en porcentaje del 99%. No obstante, al igual que ocurre con los edificios, la media de errores se mantiene constante en torno al 21,3%. En cuanto a la exactitud global de la clasificación (Figura 169), la variación de la exactitud al utilizar distintos parámetros umbrales en porcentaje es mínima y los resultados con mayor y menor exactitud global únicamente difieren en dos centésimas.

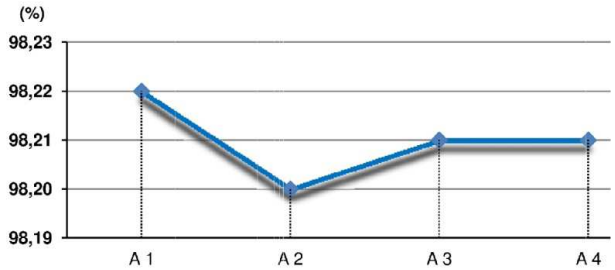


Figura 169. Gráfico de exactitud global en la clasificación. Detección de edificios y vegetación. Algoritmo A. Datos test 1.

La poca variabilidad de los resultados en función del parámetro umbral en porcentaje utilizado se aprecia en la Figura 170. En ella se muestran los resultados obtenidos al clasificar la vegetación utilizando distintos parámetros umbrales (A 1 y A 4) para los datos test 1. En la primera imagen se observa la clase de vegetación correctamente clasificada así como los errores asociadas al utilizar un umbral en porcentaje del 70%. En la segunda, se plasma la misma información pero se corresponde con los resultados obtenidos al usar un umbral en porcentaje del 99%. Tal y como se ha analizado con anterioridad, los resultados presentan diferencias mínimas y alcanzan una precisión similar. Por tanto, puede afirmarse que en caso de procesar datos LiDAR urbanos con escasa presencia de vegetación, la elección del parámetro umbral en cuanto a porcentaje utilizado para analizar la densidad de regiones de pequeño tamaño no conlleva una relevancia importante.

Respecto a los errores de comisión que se producen en la clase vegetación y que coinciden con los errores de omisión de la clase edificio, se corresponden en todos los casos con puntos de fachada aislados y no constituyen una fuente de error importante. En cuanto a los errores de omisión en la clase vegetación (comisión de edificios) se trata principalmente de automóviles aparcados junto a las fachadas de edificios. En estos casos, el análisis ejecutado identifica el automóvil como un plano de techo del edificio colindante, generando el error correspondiente (Figura 170. Detalle).

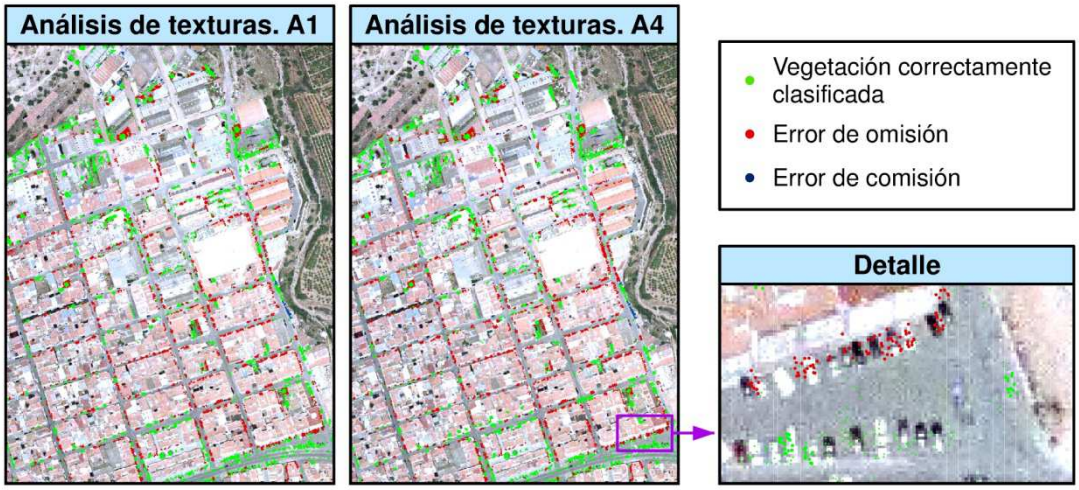


Figura 170. Detección de vegetación mediante análisis de textura (Algoritmo A). Comparación de resultados al usar diferentes umbrales en porcentaje. Datos test 1.

Para los datos test 2 (Collado Villalba) la variación de la precisión alcanzada en función del porcentaje umbral utilizado también es reducida. No obstante, es mayor a la que se presenta en los datos test 1. De nuevo el error que mayor relevancia tiene en la vegetación es el de omisión y en los edificios el de comisión (Figura 171).

Uno de los aspectos a tener en cuenta al analizar los resultados es el incremento del error de omisión producido al aumentar el umbral en porcentaje para la clase vegetación. El algoritmo desarrollado para analizar la textura es capaz de diferenciar entre árboles aislados y zonas de densa vegetación arbolada, con el fin de ejecutar un análisis adaptado a cada caso concreto. El análisis 1 sirve para detectar árboles aislados mientras que el análisis 2 se ejecuta en zonas donde la cobertura vegetal presenta una mayor densidad. En este caso, al contrario que ocurría en los datos test 1, existe un gran volumen de vegetación que se presenta en diversas formas (árboles aislados, pequeños setos, zonas de densa vegetación arbolada). Por consiguiente, los dos tipos de análisis se ejecutan con una alta frecuencia. En este sentido, tal y como se ha apuntado al analizar los datos test 1, el análisis 1 tiene una baja dependencia con respecto al umbral en porcentaje utilizado. No obstante, la precisión alcanzada sufre una mayor dependencia con respecto a los parámetros umbrales en el

análisis 2. Este aspecto puede apreciarse claramente en los errores obtenidos al ejecutar el análisis en los datos test 2, donde el error de omisión puede variar hasta un 12,4% para la clase vegetación y variaciones sobre el 6% se presentan en el error de comisión para los edificios (Figura 171).

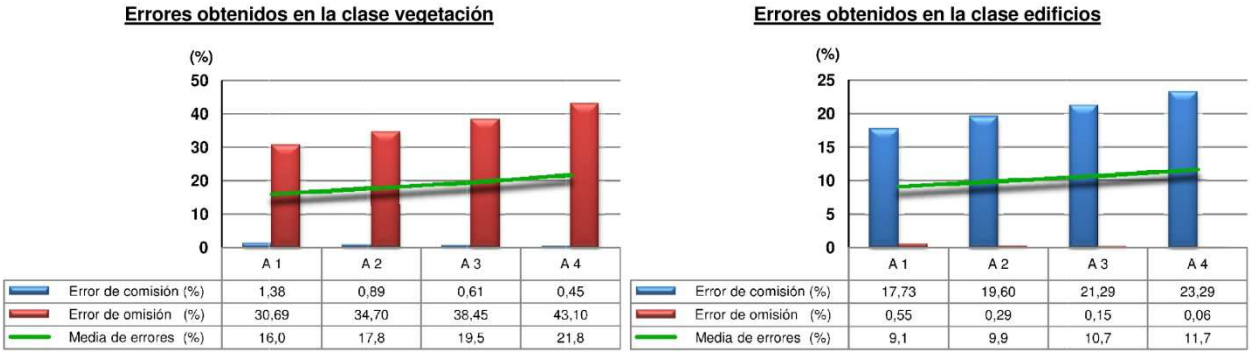


Figura 171. Gráfico de errores en la detección de edificios y vegetación. Algoritmo A. Datos test 2 (Collado Villalba).

Tanto para los edificios como para la vegetación, la media de errores se minimiza al utilizar un umbral en porcentajes del 70% (A 1), donde se alcanza una media de errores del 16% para la vegetación y un 9,1% para los edificios. Además, utilizando este mismo parámetro umbral se logra un valor máximo de 93,46% en la exactitud global de la clasificación (Figura 172).

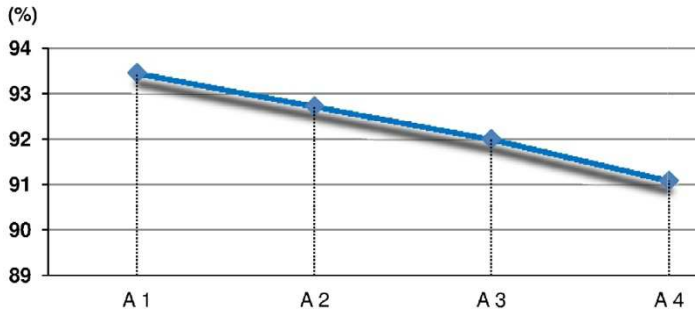


Figura 172. Gráfico de la exactitud global en la clasificación. Detección de edificios y vegetación. Algoritmo A. Datos test 2.

En la Figura 173 se muestran los resultados obtenidos así como los errores asociados al clasificar la vegetación en los datos test 2 utilizando un umbral en porcentaje del 70% (A 1) y del 99% (A 4). Con respecto a los errores de comisión, en ocasiones algunos puntos de fachada próximos a árboles son detectados erróneamente como vegetación. Además, existe una zona perteneciente al terreno que se corresponde con un solar en construcción, donde algunos puntos que no fueron correctamente clasificados como terreno son incluidos en la clase vegetación (imagen de detalle 1). En cualquier caso, el error de comisión en la detección de la vegetación es en general bajo y no presenta demasiada relevancia. Con respecto a los errores de omisión básicamente existen dos tipos. Por un lado, las zonas exteriores en densas áreas de vegetación no son detectadas correctamente mediante el análisis de texturas (imagen de detalle 2). Por otro, aquellos automóviles situados próximos a la fachada de los edificios son incluidos en la clase edificio (imagen de detalle 3).

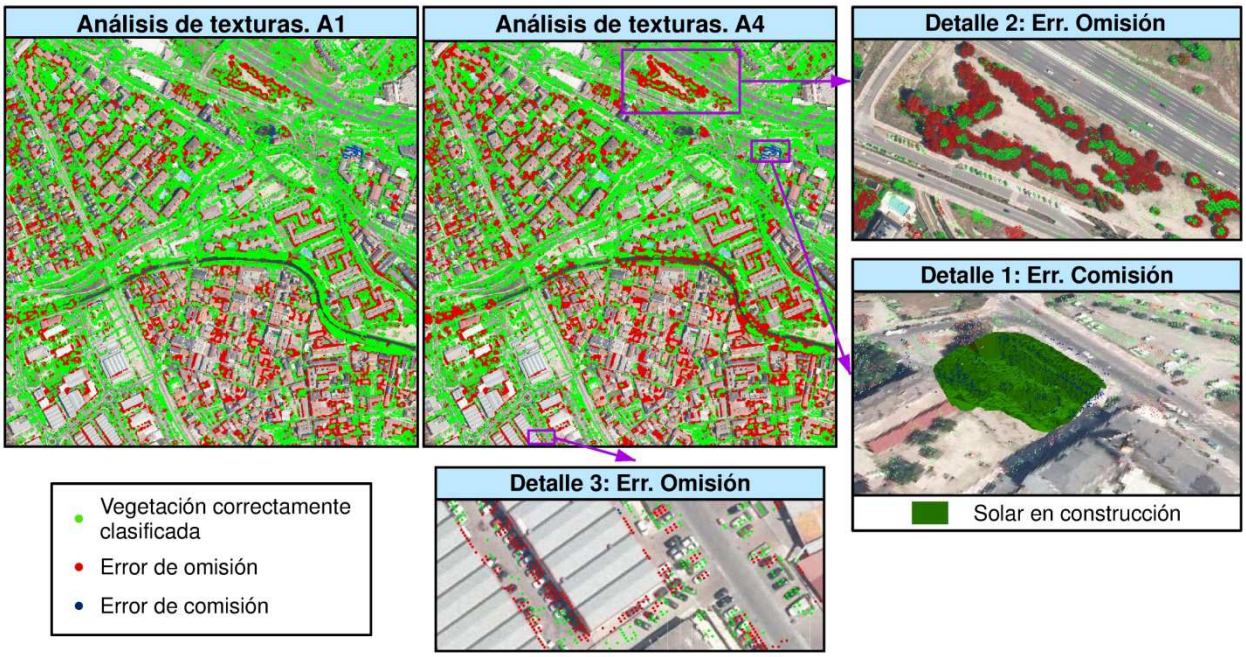


Figura 173. Detección de vegetación mediante análisis de textura (Algoritmo A). Comparación de resultados al usar diferentes umbrales en porcentaje. Datos test 2

En general, el método propuesto tiene una alta capacidad para detectar vegetación y otros pequeños objetos urbanos. Además, es capaz de adaptarse y funcionar correctamente en las proximidades del límite de la zona de estudio o en áreas colindantes a zonas de sombra. En la Figura 174 se muestran los resultados obtenidos tras ejecutar el análisis de texturas (A 1) sobre los datos test 2 (Collado Villalba). En ella aparecen dos zonas de sombra, la primera se corresponde con un canal que discurre por el centro del casco urbano donde el agua absorbe los pulsos LiDAR y, por tanto, no existe ningún tipo de información tridimensional; la segunda está asociada al límite del área de estudio. En ambos casos, la vegetación próxima a los límites de estas dos zonas se detecta correctamente. El buen funcionamiento del algoritmo junto a zonas de sombra se debe a la ejecución del análisis de texturas mediante la división del espacio en sectores, ya que cuando un sector se sitúa sobre una zona de sombra y por tanto no contiene datos, es eliminado automáticamente del análisis.

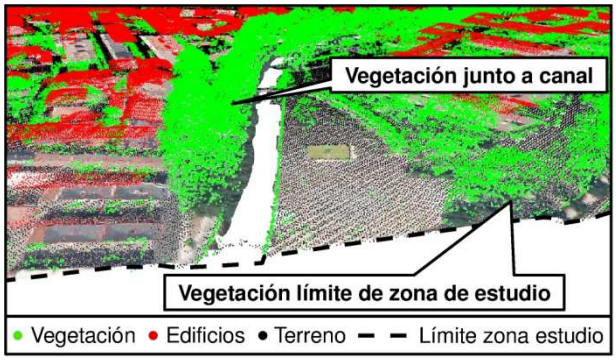


Figura 174. Detección de vegetación junto a zonas de sombra. Análisis de texturas. Datos test 2 (Collado Villalba).

El algoritmo propuesto (Algoritmo A) ocasiona principalmente errores de omisión en la vegetación y consecuentemente errores de comisión en la clase edificios.

Fundamentalmente existen dos problemas que causan estos errores. El primero de ellos es la incapacidad de detectar correctamente la vegetación situada en la parte más exterior de zonas de densa vegetación (Figura 173. Detalle 2). En estos casos, el límite de la vegetación actúa como una barrera que limita el conglomerado de regiones de pequeño tamaño y en consecuencia, para las regiones próximas a dicho límite el análisis en cuanto a densidad y distribución de las regiones de menor tamaño no cumple con los parámetros umbrales establecidos. La segunda fuente de error está asociada a la vegetación o a los automóviles situados junto a los edificios (Figura 173. Detalle 3). En estas circunstancias, al realizar el análisis para los puntos pertenecientes a la vegetación u otros pequeños objetos se incluyen en el vecindario planos de techo de los edificios colindantes. Por consiguiente, el análisis falla al considerar que las regiones objeto de estudio pertenecer a pequeños planos de techo del edificio contiguo.

Además del análisis de texturas (Algoritmo A), en el presente estudio se presenta el nuevo concepto de clasificador angular. Este clasificador se basa en el hecho de que los puntos de vegetación o pequeños objetos están constituidos principalmente por pequeñas regiones que poseen puntos de tierra en su vecindario próximo distribuidos en todas las direcciones. Afortunadamente, el clasificador angular obtiene buenos resultados en los dos tipos de error que ocasiona el análisis e texturas. Básicamente ambos algoritmos se complementan, de tal forma que los errores obtenidos se minimizan si se ejecutan de forma concatenada. Siguiendo este enfoque, se ha desarrollado un algoritmo que ejecuta primero el análisis de texturas para posteriormente ejecutar el clasificador angular sobre los resultados obtenidos (Algoritmo B).

El Algoritmo B precisa la definición de los parámetros umbrales para los dos procesos distintos. En primer lugar deberán definirse los umbrales necesarios para aplicar el análisis de texturas. En este sentido, los mejores resultados obtenidos tanto para los datos test 1 como los datos test 2 se consiguen al utilizar un umbral en porcentaje del 70% (A 1). No obstante, debe de tenerse en cuenta que el clasificador angular consigue reducir el error de omisión asociado al algoritmo A, pero no es capaz de modificar los errores de comisión ocasionados. Por consiguiente, se opta por utilizar un porcentaje umbral del 99% por ser el parámetro más restrictivo y que menores errores de comisión ocasiona (A 4).

En cuanto al clasificador angular, debe de establecerse un parámetro umbral en cuanto al tamaño de las regiones para poder ser clasificadas como vegetación. Este parámetro es común al análisis de textura y como se ha comentado con anterioridad, se considera idóneo un parámetro de 10 m^2 en superficie. Respecto al radio del vecindario cilíndrico, para obtener buenos resultados debe de poseer un diámetro menor a la longitud de las líneas de fachada asociadas a los edificios de menor tamaño. Finalmente se opta por un vecindario cilíndrico de 6 m de radio.

El parámetro umbral que influye más directamente en los resultados es el umbral angular necesario para determinar si los puntos de tierra están dispuestos en todas las direcciones dentro del vecindario próximo del punto objeto de análisis. En consecuencia, se han realizado cinco pruebas sobre las dos zonas test variando el umbral angular. Estas pruebas servirán para determinar los umbrales que mayor precisión alcanzan y poder analizar el grado de dependencia del algoritmo con respecto a los umbrales preestablecidos. La totalidad de parámetros umbrales utilizados se muestran en la Figura 175.

PARÁMETROS UMBRALES ASOCIADOS AL ANÁLISIS DE TEXTURA:

RADIO DE VECINDARIO CILÍNDRICO PARA ANALIZAR TEXTURA: 6 m
RADIO DE LOS SEMISECTORES INTERIORES: 3,5 m
TAMAÑO DE REGIÓN UMBRAL: 10 m^2
AMPLITUD ANGULAR DE LOS SECTORES: 50^g
NÚMERO MÍNIMO DE PUNTOS PERTENECIENTES A REGIONES PEQUEÑAS EN SEMISECTORES: 10
UMBRAL %: 99 %

PARÁMETROS UMBRALES ASOCIADOS AL CLASIFICADO ANGULAR:

PARÁMETROS UMBRALES FIJOS:

RADIO DEL VECINDARIO CILÍNDRICO: 6 m
TAMAÑO DE REGIÓN UMBRAL: 10 m^2

PARÁMETROS UMBRALES FIJOS:

B 1: UMBRAL ANGULAR = 40^g **B 2:** UMBRAL ANGULAR = 50^g **B 3:** UMBRAL ANGULAR = 60^g
B 4: UMBRAL ANGULAR = 70^g **B 5:** UMBRAL ANGULAR = 80^g

Figura 175. Detección de vegetación y edificios. Parámetros umbrales utilizados en el Algoritmo B.

Los errores obtenidos al ejecutar el Algoritmo B con los diferentes parámetros umbrales sobre los datos test 1 (Vall d'Uixó) se muestran en la Figura 176. Tal y como se puede observar en el gráfico, el error de omisión para la clase de vegetación disminuye a medida que el umbral angular se hace más permisivo, consiguiendo un mínimo error de tan solo 0,84% al utilizar un umbral angular de 80^g. Por el contrario, el error de comisión aumenta a medida que crece el umbral angular, alcanzando un mínimo del 2,21% al utilizar un umbral angular de 40^g y un error máximo de 11,27% al utilizar un umbral angular de 80^g.

Tal y como era de esperar, los errores asociados a la clase de edificios tienen un comportamiento inverso al de la clase vegetación. En este caso, el error de omisión aumenta gradualmente a medida que se incrementa el umbral angular, alcanzando un mínimo del 0,12% al utilizar un parámetro de 40^g y un máximo del 0,79% al usar el umbral menos restrictivo (80^g). Por el contrario, el error de comisión obtiene un valor máximo del 0,89% al utilizar un umbral de 40^g y disminuye paulatinamente hasta obtener un mínimo del 0,17%.

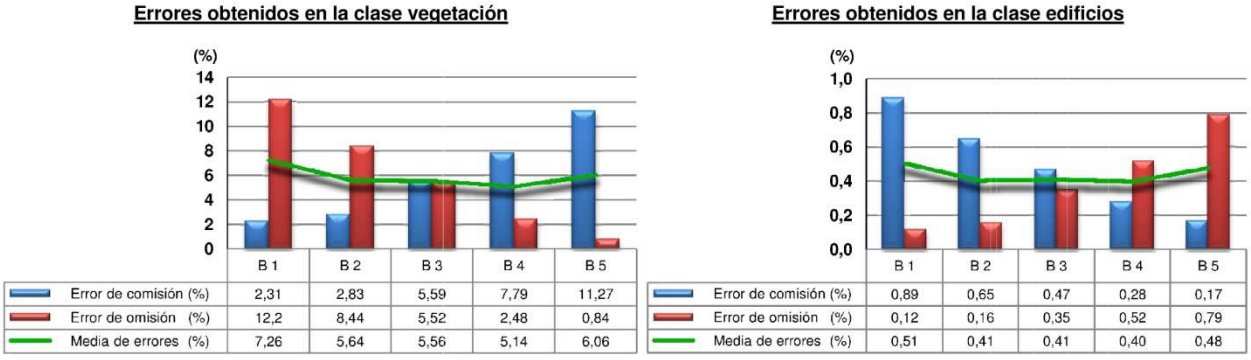


Figura 176. Gráfico de errores en la detección de edificios y vegetación. Algoritmo B. Datos test 1 (Vall d'Uixó).

Al igual que ocurría al ejecutar el análisis de texturas (Algoritmo A), las características de los datos test 1 (Vall d'Uixó) se reflejan en los resultados. En primer lugar, la escasa presencia de vegetación y el hecho de que no existan densas zonas de cobertura vegetal permite obtener resultados con escasa variabilidad en cuanto a los parámetros umbrales utilizados. En este caso, el algoritmo únicamente deberá detectar árboles aislados así como

otros pequeños objetos tales como automóviles, bancos, pulsos reflejados sobre personas, etc. Consecuentemente, variaciones en el parámetro umbral angular no tendrá gran repercusión en la detección de entidades. En este sentido, la media de errores toma valores que oscilan en tan solo un 2% para la clase vegetación y un 0,11% para los edificios. Además, la exactitud global de la clasificación permanece prácticamente constante y únicamente se producen variaciones en torno al 0,15% (Figura 177). En segundo lugar, a pesar de que los errores en las clases edificio y vegetación están relacionados, dado que el número de puntos de vegetación es muy inferior al de edificios, los errores expresados en porcentajes resultan mucho menores en la clase edifico donde se alcanza una media de errores mínima de tan solo 0,40%. En cualquier caso, los mejores resultados obtenidos se consiguen al utilizar un umbral angular que oscila entre 50^g y 70^g.

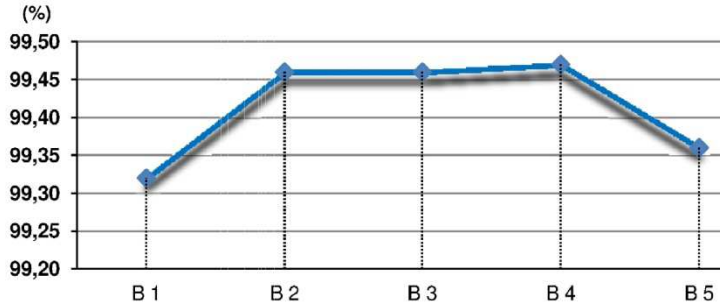


Figura 177. Gráfico de exactitud global en la clasificación. Detección de edificios y vegetación. Algoritmo B. Datos test 1.

La Figura 178 muestra los resultados obtenidos al aplicar tres parámetros umbrales diferentes sobre los datos test 1 (Vall d'Uixó). En esta imagen se muestra la clasificación de la vegetación y pequeños objetos así como los errores asociados. Uno de los problemas que acarrea el análisis de texturas (Algoritmo A) es la inclusión dentro de la clase edificios de automóviles aparcados junto a las construcciones. Este problema se soluciona en gran medida al ejecutar el clasificador angular. No obstante, si el parámetro umbral angular es demasiado restrictivo algunos automóviles siguen produciendo errores de omisión (Figura 178. Detalle 1). A medida que el parámetro umbral angular aumenta disminuyen los errores de omisión dentro de la clase vegetación. Sin embargo, un parámetro umbral angular demasiado alto permite clasificar como vegetación pequeños planos de techo colindantes a las calles del entramado urbano, sobre todo si existen puntos anteriormente clasificados como vegetación en el vecindario próximo (Figura 178. Detalle 2). Este efecto se debe a que el algoritmo clasifica como vegetación aquellas pequeñas regiones que están rodeadas por terreno o vegetación en todas las direcciones. En este sentido, el umbral angular es utilizado para determinar la posición relativa de los puntos de vegetación y tierra existentes en el vecindario de la región objeto de análisis. Consecuentemente, la utilización de un ángulo demasiado grande como parámetro umbral ocasiona que el algoritmo detecte puntos de tierra y vegetación en todas las direcciones, cuando realmente no es así.

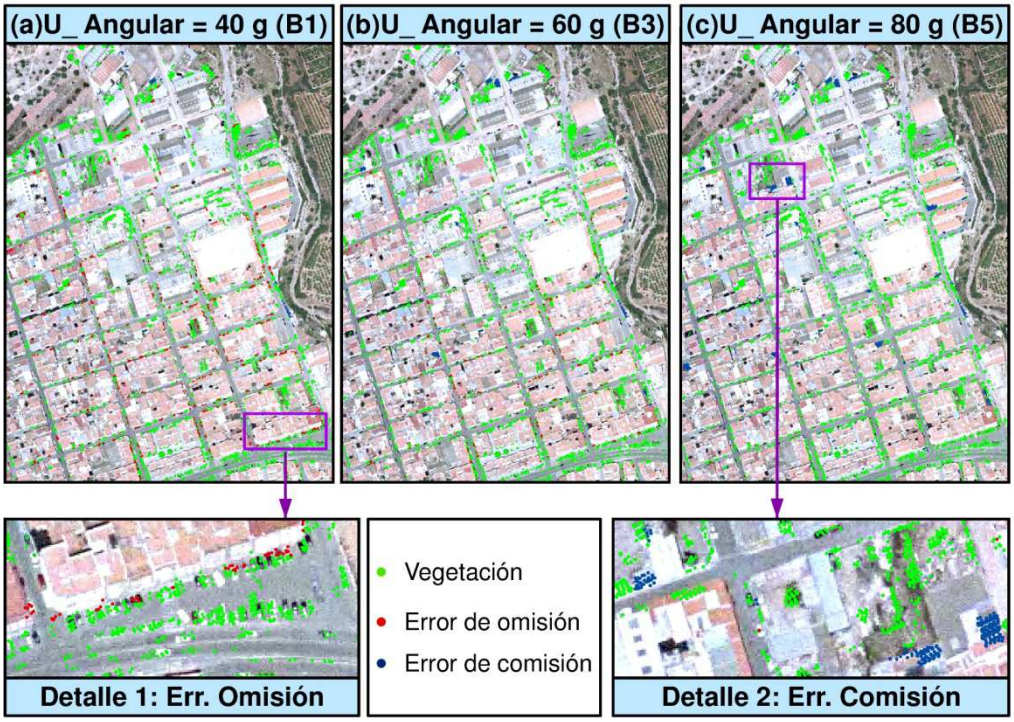


Figura 178. Clasificación de vegetación. Comparación de resultados al usar diferente umbral angular. Algoritmo B. Datos test 1.

El Algoritmo B se comporta de forma similar sobre los datos test 2 (Collado Villalba). En este caso, al igual que ocurría en los datos test 1, el error de omisión asociado a la vegetación disminuye al aumentar el parámetro umbral angular, alcanzando un error máximo del 5% al utilizar 40^g como umbral angular y un mínimo del 1,47% cuando el umbral utilizado es de 80^g (Figura 179). Por el contrario, el error de comisión en la clase vegetación aumenta al incrementar el parámetro umbral angular, consiguiendo alcanzar un error mínimo de tan solo 0,61% al utilizar un umbral angular de 40^g. Como era de esperar, el comportamiento de los errores es inverso en la clase edificios, donde el error de omisión aumenta a medida que se incrementa el umbral angular, obteniendo un error de omisión mínimo de tan solo un 0,18% al utilizar un umbral de 40^g. De forma opuesta, el error de comisión para los edificios disminuye al aumentar el valor del umbral angular, alcanzando un mínimo de 0,4% cuando el umbral es de 80^g.

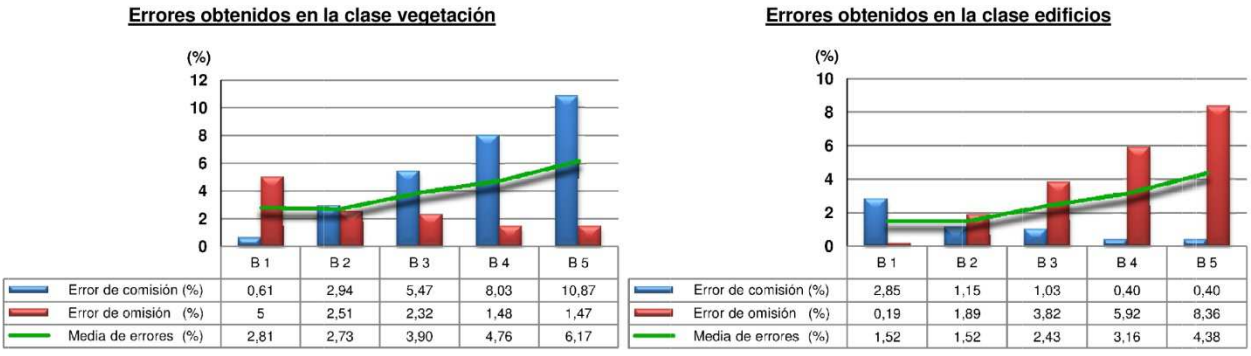


Figura 179. Gráfico de errores en la detección de edificios y vegetación. Algoritmo B. Datos test 2 (Collado Villalba).

En cuanto a la exactitud global de la clasificación (Figura 180), puede apreciarse como disminuye la exactitud a medida que el parámetro umbral angular se relaja, obteniendo la

mínima exactitud al utilizar un umbral angular de 80^g. Principalmente este comportamiento es debido al alto error por comisión que se produce en la clase vegetación al utilizar un umbral angular alto. Cabe destacar que los datos test 2 (Collado Villalba) contienen una alta presencia de cobertura vegetal donde se presentan multitud de árboles próximos a las edificaciones, gran parte de estos árboles se detectan al ejecutar el análisis de texturas (Algoritmo A) y se utilizan por el clasificador angular para completar la clasificación. No obstante, tal y como se ha comentado con anterioridad, la elección de un parámetro umbral angular demasiado alto acarrea errores de comisión en las proximidades. Por este motivo, los resultados obtenidos al ejecutar el Algoritmo B sobre los datos test 2 son menos constantes y se aprecia una mayor dependencia de la precisión de los resultados con respecto al parámetro umbral angular utilizado.

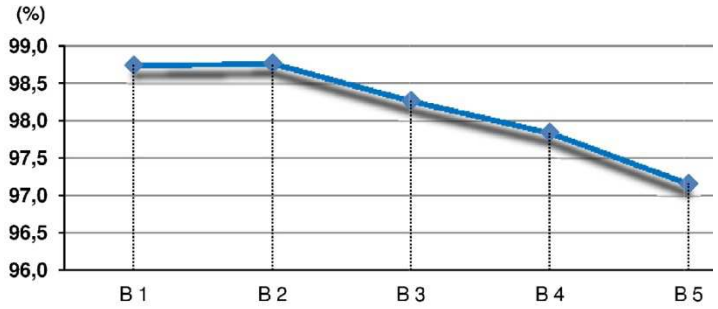


Figura 180. Gráfico de exactitud global en la clasificación. Detección de edificios y vegetación. Algoritmo B. Datos test 2.

La Figura 181 muestra los resultados obtenidos al aplicar tres parámetros umbrales diferentes sobre los datos test 2 (Collado Villalba). En ella aparece la clasificación de vegetación así como los errores de comisión y omisión tras aplicar el análisis de texturas y el clasificador angular de forma concatenada (Algoritmo B). En la primera de las imágenes aparece la clasificación obtenida al utilizar un parámetro umbral angular de 40^g. En este caso, el umbral utilizado es muy restrictivo y prácticamente no se producen errores de comisión. No obstante, existen numerosos errores de omisión asociados a la existencia de pequeñas regiones situadas junto a los edificios. En estos casos, a pesar de que estas regiones pertenecen a vegetación u otros pequeños objetos, dado que son adyacentes a las edificaciones, el algoritmo considera que pertenecen a pequeños planos de techo y consecuentemente son automáticamente incluidas en la clase edificios. Básicamente, pueden diferenciarse dos casos. Por un lado, algunos automóviles u otros objetos como contenedores de camiones aparcados junto a edificios (Figura 181. Detalle 1). Por otro, árboles altos situados junto a edificaciones (Figura 181. Detalle 2).

A medida que el parámetro umbral angular se hace mayor, disminuyen los errores de omisión y aumentan los errores de comisión. En la imagen central se ha utilizado un parámetro umbral angular de 60^g. En este caso todavía existen errores de omisión pero se han reducido considerablemente. No obstante, existe una mayor presencia de errores de comisión. En la imagen de detalle correspondiente (Figura 181. Detalle 3) puede apreciarse cómo todavía se producen algunos errores por defecto asociados a vegetación situada junto a los edificios y como algunos planos de techo colindantes a la calle han sido clasificados como vegetación (error de comisión). La misma zona de detalle se muestra al utilizar un parámetro umbral angular de 80^g (Figura 181. Detalle 4). En este caso el parámetro umbral angular es muy relajado y los errores de omisión han desaparecido por completo. Sin embargo, el error de comisión aumenta considerablemente y una parte importante de planos de techo adyacentes a la calle son clasificados erróneamente como vegetación.

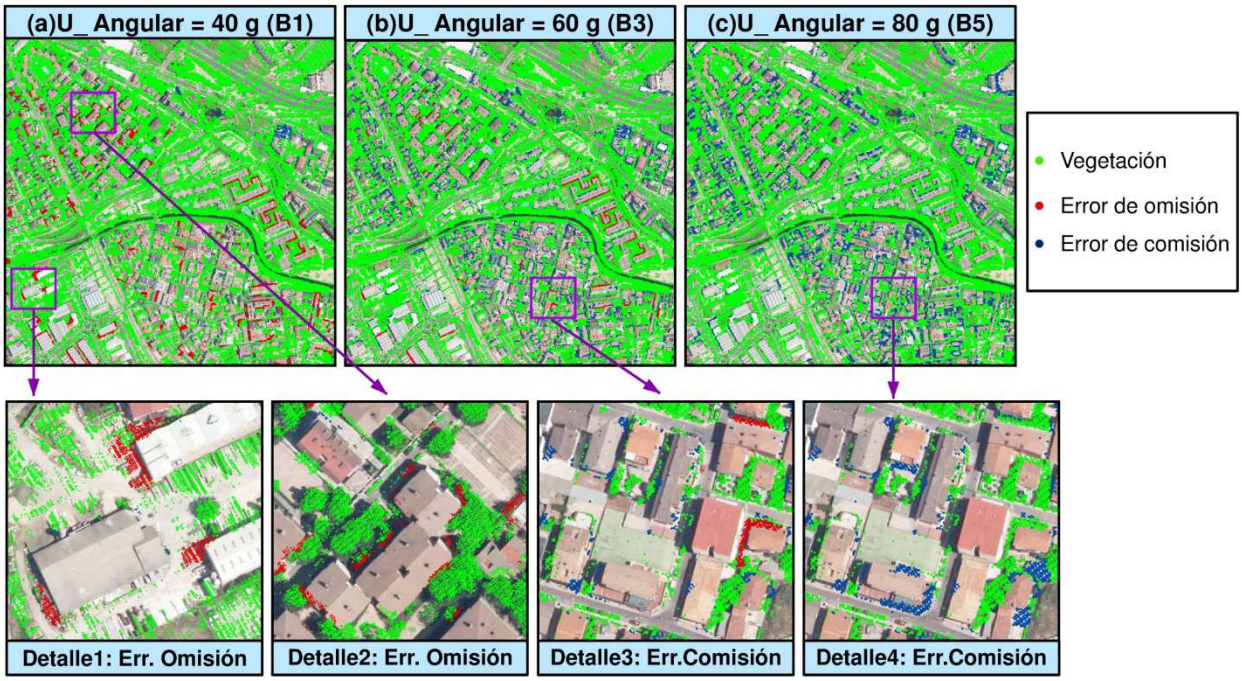


Figura 181. Clasificación de vegetación. Algoritmo B. Comparación de resultados al usar diferente umbral angular. Datos test 2.

Como se ha comentado con anterioridad, los errores existentes en las clases edificio y vegetación están en gran parte relacionados, de forma que los errores de comisión en la clase vegetación se transmiten a errores de omisión en la clase edificio y viceversa. No obstante, existen dos fuentes de error en la vegetación que son independientes con respecto a la clase edificios y que dependen directamente de la clasificación del terreno.

El proceso de densificación progresiva utilizado para detectar el terreno se basa en clasificar como tierra los puntos LiDAR que están próximos a una superficie de referencia auxiliar. Este proceso es iterativo y la superficie de referencia se actualiza cada vez que un nuevo punto de terreno es añadido. Este enfoque ocasiona que puntos de vegetación que están próximos a la superficie de referencia puedan ser clasificados erróneamente como terreno. El algoritmo para detectar el terreno propuesto en el presente trabajo combina técnicas de segmentación con el método clásico de densificación, consiguiendo con ello eliminar considerablemente esta fuente de error. No obstante, la completa eliminación no es posible y algunos puntos asociados a baja vegetación son erróneamente clasificados como terreno (Figura 182b). Además, el algoritmo utilizado para detectar el terreno en entornos urbanos falla en zonas con pendientes verticales ya que considera que las regiones pertenecen a fachadas de edificios (Figura 182a). En cualquier caso, estas dos fuentes de error se presentan de forma puntual y no tienen una gran relevancia en la precisión global de los resultados.

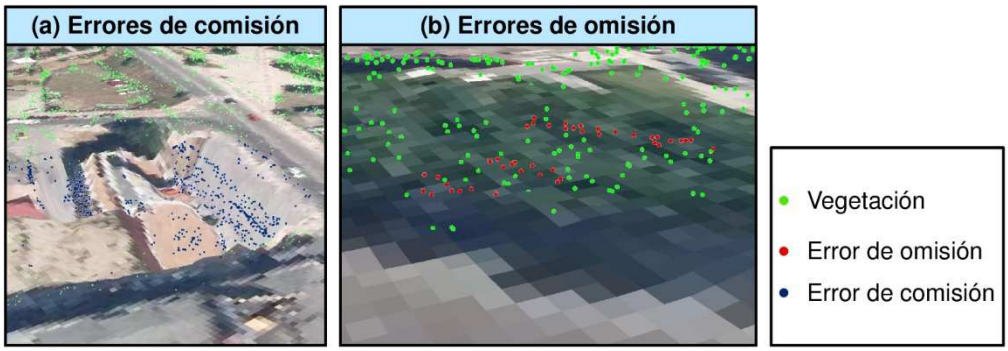


Figura 182. Errores en la detección de vegetación como consecuencia de una incorrecta detección del terreno. Datos test 2.

Tras el análisis de los resultados obtenidos, puede afirmarse que el Algoritmo B sufre una dependencia relativamente alta con respecto al parámetro umbral angular cuando existe una gran presencia de vegetación en el entramado urbano, especialmente, cuando hay densas zonas de vegetación situadas junto a los edificios, tal y como ocurre en los datos test 2. Por el contrario, en zonas urbanas con escasa cobertura vegetal que se presenta fundamentalmente como árboles aislados, el algoritmo es más robusto y la dependencia con respecto al umbral angular disminuye. Este aspecto queda patente al comparar la exactitud global de la clasificación obtenida al ejecutar el algoritmo sobre los datos test 1 y 2 utilizando los mismos parámetros umbrales. En los datos test 1, existe escasa vegetación que se presenta fundamentalmente como árboles aislados y por tanto la exactitud global de la clasificación tan solo oscila en un 0,15% al utilizar diferentes parámetros. Sin embargo, en los datos test 2 hay una alta presencia de vegetación y árboles junto a edificios y consecuentemente las variaciones que experimenta la exactitud global aumentan hasta un 1,57%.

Referente a las ventajas que aporta la ejecución de los dos algoritmos propuestos de forma concatenada (Algoritmo B) frente a utilización únicamente del análisis de texturas (Algoritmo A). Los mejores resultados al utilizar el análisis de texturas en los datos test 1 (Vall d'Uixó) se obtienen al seleccionar un umbral en porcentaje del 70%, momento en que la media de errores se reduce al 21,2% para la vegetación y 1,3% para la clase edificios. Con este parámetro umbral se alcanza una exactitud global de la clasificación del 98,22%. Esta precisión es ampliamente superada al utilizar el Algoritmo B con un parámetro umbral angular de 70^g (B 4), donde se consigue reducir la media de errores para la vegetación al 5,14% y para edificios a tan solo el 0,40%. La exactitud global de la clasificación aumenta hasta alcanzar un 99,47%. Por tanto, la ejecución del Algoritmo B consigue minimizar la media de errores tanto para la vegetación como para los edificios, consiguiendo una reducción de un 16,06% en el caso de la vegetación y una reducción del 0,9% en los edificios. La precisión global aumenta en un 1,25%.

Referente a los datos test 2, los resultados más precisos al ejecutar el Algoritmo A también se obtienen al utilizar un parámetro umbral en porcentaje del 70% (A 1). En este caso se consigue una media de errores para la vegetación del 16% y 9,1% para los edificios. La exactitud global de la clasificación llega hasta un 93,46%. Los resultados mejoran notablemente al ejecutar el Algoritmo B. En este caso, los mejores resultados se obtienen al utilizar un parámetro umbral angular de 50^g, donde la media de errores para la vegetación es del 2,73% y para los edificios del 1,52%. La exactitud global mejora hasta alcanzar el 98,77%. Por tanto, en los datos test 2 también se experimenta una sustancial mejora al ejecutar el Algoritmo B consiguiendo reducir la media de errores un 13,27% para la vegetación y 7,58% para los edificios. La exactitud global aumenta un 5,31% con respecto a la obtenida al ejecutar el Algoritmo A. En la Figura 183 pueden apreciarse los resultados más precisos obtenidos tras ejecutar ambos algoritmos.

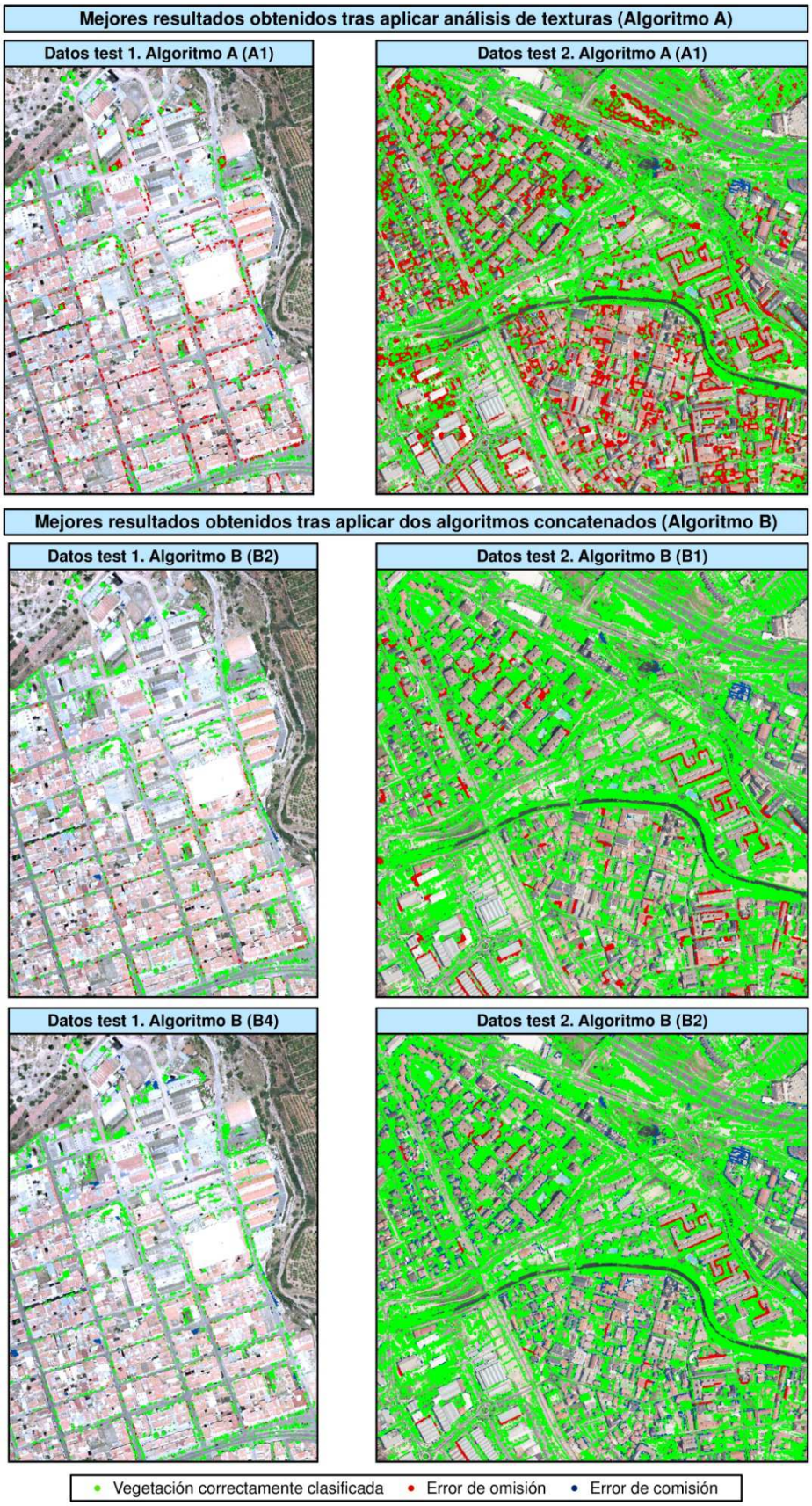


Figura 183. Comparativa de la precisión alcanzada por los Algoritmos A y B. Detección de vegetación y edificios.

La diferenciación entre edificios y vegetación es un aspecto fundamental en el procesamiento de datos LiDAR, por lo que diferentes enfoques han sido propuestos en los últimos años, alcanzando diferentes precisiones.

Chehata et al. (2009) analizaron información completa de onda asociada a ecos múltiples en entornos urbanos y aplicaron técnicas de segmentación sobre datos LiDAR, consiguiendo detectar edificios con un error mínimo del 4,2%. Por otro lado, Samadzadegan y Saeedi (2009) segmentaron datos LiDAR urbanos pertenecientes a zonas residenciales mediante los algoritmos de k-media y PSO produciendo un error de omisión del 5,3% y error comisión de 7,2% en la clase edificio. Otro enfoque muy extendido consiste en combinar datos LiDAR con otras fuentes de datos. En esta línea destaca el trabajo de Moussa y El-Sheimy (2012) que combinaron datos LiDAR e imágenes aéreas para posteriormente aplicar técnicas de segmentación, consiguiendo diferenciar edificios de vegetación en zonas urbanas con una precisión máxima por área del 95,2% en la clase edificio. Finalmente, Ekhtari et al. (2008) combinan procesos de segmentación y densificación progresiva de TIN para extraer edificios en un conjunto de datos LiDAR pertenecientes a entornos urbanos. En este trabajo modelos digitales de superficie se generan a partir del primer y último pulso reflejado y posteriormente los MDS son segmentados en superficies lisas y rugosas. A partir del análisis de las regiones obtenidas tras segmentar la escena consiguieron obtener la clase de edificios con un error de comisión del 1,65% y un error omisión de 10,46%. En cuanto a la vegetación y otros objetos pequeños, esta clase se detectó con un error de comisión del 6,69% y el error omisión de 1,02%. En cualquier caso, los resultados obtenidos al aplicar el Algoritmo B en los datos test 1 y 2 son significativamente mejores.

V. Conclusión y líneas futuras

El objetivo de la presente tesis es el diseño y desarrollo de algoritmos para detectar y clasificar entidades sobre datos LiDAR en zonas urbanas. En este sentido, en los últimos años multitud de enfoques han sido desarrollados a tal fin. No obstante, el desarrollo de algoritmos robustos con un alto grado de automatización resulta una difícil tarea (Sithole y Vosselman 2004; Meng et al. 2010; Chen te al. 2012). Por este motivo, el diseño de la totalidad de algoritmos desarrollados ha perseguido un doble objetivo: reducir la dependencia de los resultados con respecto a las características de la escena y alcanzar un alto grado de automatización.

Las entidades a detectar en los datos son cuatro: terreno, puentes, edificios y vegetación junto con pequeños objetos. La detección de cada una de ellas precisa la ejecución de un proceso propio y diferenciado, donde uno de los enfoques más común consiste en interpolar los datos a malla regular (GRID) para facilitar la clasificación mediante la aplicación de filtros. Este enfoque es utilizado por multitud de autores para detectar el terreno (Roggero 2001; Chen et al. 2007; Kim y Shan 2012), para localizar puentes (Zheng et al. 2013; Evans 2008), así como para diferenciar entre edificios y vegetación (Meng, Wang y Currit 2009; Matikainen et al. 2009; Oude Elberink y Maas 2000). No obstante, interpolar los datos LiDAR brutos a GRID causa una pérdida significativa de información asociada al proceso de generalización que conlleva la propia interpolación (Voselman, 2000). En consecuencia, antes de comenzar con el diseño de algoritmos se optó por utilizar durante todo el proceso datos LiDAR brutos dispuestos de forma aleatoria, siendo ésta una de las premisas considerada en el diseño e implementación de los distintos algoritmos.

Además de las aplicaciones cuyo fin es la detección de entidades, se han desarrollado distintos algoritmos destinados a un pre-procesamiento de los datos. Por un lado, la utilización de datos brutos dispuestos de forma aleatoria precisa del desarrollo de un sistema de codificación y organización que permita establecer relaciones de proximidad y vecindad. En este sentido, un doble sistema de organización se ha desarrollado con el fin de establecer tanto relaciones de vecindad entre puntos pertenecientes al conjunto de datos LiDAR (organización mediante triangulación), así como entre puntos pertenecientes al conjunto de datos y exteriores al mismo (partición del espacio en celdas de altura infinita). Además, la correcta elección de un sistema para determinar de forma eficiente el vecindario próximo es de gran importancia cuando se trabaja con datos tridimensionales masivos. Finalmente se opta por un vecindario cilíndrico de altura infinita, aspecto que deberá ser tenido en consideración en el diseño de los distintos algoritmos de clasificación implementados. Por último, antes de comenzar con la clasificación de entidades es necesario detectar y filtrar errores groseros (valores atípicos contenidos en los datos), a tal efecto, se han desarrollado dos algoritmos.

En cuanto a la automatización del proceso, las metodologías propuestas son completamente automáticas siendo nula la interacción del operador. En la detección de errores groseros la mayoría de trabajos propuestos utilizan un doble proceso (Shen et al. 2011, Lin y Zhang 2014; Meng et al. 2009; Silván-Cárdenas y Wang 2006), donde se lleva a cabo por un

operador un análisis de la distribución de frecuencias y posteriormente un se ejecuta un análisis automático en cuanto a variabilidad de altitud de los puntos del vecindario. De esta forma, el primer análisis consigue detectar errores groseros producidos por errores de medición cuya altura difiere en gran medida con respecto el valor esperado; y el segundo permite filtrar valores atípicos próximos al terreno. El principal problema radica en que el primero de los procesos no es automático y precisa la interacción de un operador. Para solucionar este aspecto, se ha desarrollado un algoritmo que ejecuta un doble análisis del vecindario (Algoritmo C. Detección de outliers). Se trata de un algoritmo que aplica dos análisis concatenados en cuanto a variaciones locales de altitud, ajustando los parámetros umbrales de forma automática, consiguiendo así detectar los distintos tipos de errores groseros. Con ello, es posible prescindir del análisis de la distribución de frecuencias y por tanto no es preciso interacción humana alguna, consiguiendo así la automatización completa del proceso.

Para detectar el terreno, se han propuesto dos algoritmos. El primero de ellos (Algoritmo A) combina características de los distintos enfoques existentes con el fin de obtener un algoritmo robusto que alcance precisos resultados. La aplicación comienza con un proceso de segmentación y posteriormente la zona de estudio se divide en bloques de igual tamaño donde la región que posee el punto de menor altitud de cada bloque es incluida en el terreno provisional inicial. Es en esta fase previa al proceso de densificación cuando es necesaria la interacción del usuario, ya que la selección del tamaño del bloque debe de ser siempre mayor al tamaño de los edificios existentes en la escena, por lo que una revisión por parte de un operador debe de llevarse a cabo. Este aspecto es solventado por el segundo algoritmo propuesto (Algoritmo B) que incorpora un análisis adicional previo a la generación del terreno provisional en cuanto a tamaño y posición relativa de las regiones, que permite eliminar la influencia en la precisión de los resultados del tamaño de los bloques y por tanto, la selección del tamaño de los bloques deja de tener relevancia consiguiendo así un proceso completamente automático. En cuanto a la clasificación del resto de entidades (puentes, edificios y vegetación) la totalidad de algoritmos propuestos son completamente automáticos.

Con respecto a la precisión alcanzada y grado de robustez de los algoritmos propuestos, en el presente estudio se ha realizado una evaluación exhaustiva de la precisión de los resultados obtenidos por cada algoritmo desarrollado. Además, con el fin de evaluar la dependencia de la precisión alcanzada con respecto a los parámetros umbrales utilizados, se han ejecutado distintas pruebas modificando los umbrales y cuantificando la precisión obtenida en cada caso. Por otro lado, con el fin de evaluar la dependencia de la precisión respecto a las características de la escena, la totalidad de algoritmos se han ejecutado sobre dos zonas test con características muy distintas.

Referente a la detección de errores groseros, los resultados más precisos para ambas zonas test se obtienen al ejecutar el Algoritmo B. En ambos casos, la clasificación alcanza una exactitud global del 99,9% y la media de errores se reduce a un 1,94% para los datos test 1 y 6,27% para los datos test 2. Cabe mencionar la alta dependencia del algoritmo con respecto al umbral en porcentaje utilizado para identificar variaciones locales de altura en el vecindario, ya que variaciones en torno al 5% en la magnitud del umbral ocasionan variaciones en la media de errores superiores al 25%. No obstante, el algoritmo se comporta de igual forma sobre ambas zonas test y los mejores resultados se obtienen al utilizar los mismos parámetros umbrales, por lo que teniendo en cuenta la alta complejidad de los datos test utilizados puede considerarse un algoritmo robusto. Con respecto a la detección del terreno, el Algoritmo B no solamente consigue aumentar el grado de automatización con respecto al Algoritmo A, sino que además la influencia de los parámetros umbrales en la precisión de los resultados disminuye considerablemente. Ésto se debe a la inclusión de dos análisis adicionales que permiten un comportamiento más robusto. El primero de los análisis se ejecuta en la detección del terreno provisional inicial y permite eliminar la influencia del tamaño de los bloques; el segundo se localiza en el propio

proceso de densificación y consigue reducir la dependencia del parámetro umbral en desnivel con respecto a la precisión de los resultados. De nuevo los mejores resultados para ambos datos test se obtienen al utilizar los mismos parámetros umbrales, quedando patente la independencia de la precisión alcanzada con respecto a las características de la escena. Cabe resaltar la alta precisión alcanzada por el Algoritmo B en ambas zonas test, consiguiendo una media de errores mínima en la clase terreno de tan solo el 0,17% para los datos test 1 y 0,47% para los datos test 2 (tras filtrar puentes).

En cuanto a la detección de puentes, en el presente estudio se presentan dos nuevos algoritmos. El primero de ellos (Algoritmo A) detecta los bordes de los puentes y posteriormente clasifica la totalidad de la entidad mediante el análisis de distintas características asociadas a la propia estructura, tales como localización de puntos entre bordes paralelos o la continuidad estructural en la dirección principal y transversal al puente. En este caso, a pesar de ofrecer buenos resultados se observa una alta dependencia entre la precisión de los resultados y el parámetro umbral en desnivel necesario para detectar bordes. El Algoritmo B reduce notablemente esta dependencia. Para ello, incorpora dos nuevos aspectos: por un lado, la detección de bordes se realiza mediante un doble proceso donde el parámetro umbral en desnivel se adapta automáticamente para detectar los bordes del puente asociados tanto a la zona central como a los extremos; por otro, un sistema para detectar bordes de puentes colindantes a zonas de sombra se implementa. Como consecuencia, el Algoritmo B resulta mucho más robusto y es capaz de detectar puentes con características muy diversas. Finalmente, se han desarrollado dos algoritmos con el fin de diferenciar los edificios de la vegetación. El Algoritmo A permite clasificar la vegetación y diferenciarla de los edificios mediante técnicas de análisis de texturas. En este caso, el algoritmo es capaz de adaptarse en función del elemento a detectar (arbustos aislados o densas zonas de vegetación arbolada) y, por consiguiente, sufre una escasa variación en la precisión de los resultados al modificar los parámetros umbrales. No obstante, ocasiona importantes errores en la parte exterior de las zonas de vegetación densa. Este tipo de error es corregido automáticamente al concatenar el proceso con el clasificador angular (Algoritmo B) donde se consigue reducir la media de errores al 5,14% para la vegetación y a tan solo 0,40% en los edificios para los datos test 1, así como al 2,73% en la vegetación y 1,52% en los edificios para los datos test 2.

La presente tesis realiza distintos aportes individuales en cada uno de los ámbitos tratados. En cuanto a la detección del terreno, la utilización de la interpolación en vez de redes de triángulos (TIN) para extender el terreno por debajo de los diferentes objetos urbanos es una novedad que permite reducir considerablemente el tiempo de cómputo, además de minimizar la transmisión de errores. Por otro lado, a pesar de que los métodos de densificación han sido combinados anteriormente con técnicas de segmentación (Ekhtari et al. 2008, Pérez et al. 2012, Lin y Zhang 2014), en ningún caso se han analizado las características de las agrupaciones dentro del proceso de densificación. En este sentido, el algoritmo propuesto pone de relevancia las ventajas de incluir análisis adicionales en cuanto al tamaño de las regiones y a la posición relativa entre ellas. Referente a la detección de puentes, la utilización de datos LiDAR brutos sin interpolar la malla regular supone por si misma una novedad. Además, la detección de puentes que discurren sobre el agua y que por tanto son colindantes a zonas de sombra no había sido resuelta hasta ahora. Destaca en este campo el nuevo concepto de clasificador angular por sectores propuesto que permite realizar análisis direccionales (análisis de continuidad estructural y de detección de bordes colindantes a zonas de sombra). Este concepto también ha sido incorporado por el algoritmo de detección de vegetación, consiguiendo así analizar texturas en datos brutos sin interpolar a malla regular (GRID). Además, el nuevo concepto de clasificador angular permite analizar la distribución espacial de los puntos de tierra y detectar la vegetación con excelentes resultados.

Uno de los aspectos más relevantes del presente trabajo es la alta precisión de los resultados obtenida a pesar de trabajar únicamente con datos LiDAR y no ejecutar análisis

que combinan distintas fuentes de datos. En este sentido, cabe destacar que los mejores resultados se obtienen al utilizar algoritmos multiproceso que ajustan automáticamente los parámetros umbrales, o bien al ejecutar algoritmos que combinan procesos con enfoques distintos. En efecto, en la detección de errores groseros, así como en la clasificación de puentes, los resultados son altamente dependientes al parámetro umbral en desnivel utilizado para detectar variaciones locales en altura, aspecto que se soluciona en ambos casos mediante la implementación de un algoritmo multiproceso que itera su ejecución ajustando de forma automática el desnivel umbral. En cuanto a la detección del terreno, el algoritmo ejecuta de forma concatenada distintos procesos propios de diferentes enfoques (segmentación, filtros morfológicos, proceso de densificación y filtros basados en interpolación). Asímismo, los mejores resultados en la detección de la vegetación y edificios se obtienen al combinar un análisis de texturas con el clasificador angular. Consecuentemente, puede concluirse que la combinación de múltiples procesos con parámetros umbrales adaptativos así como la fusión de distintos enfoques proporcionan algoritmos más robustos y precisos.

En cuanto a futuras líneas de investigación, los distintos algoritmos propuestos están diseñados para obtener precisos resultados en zonas urbanas. No obstante, su aplicación en otro tipo de entornos debería ser factible. En cualquier caso, las diferentes aplicaciones deberían probarse y analizar cuantitativamente la precisión de los resultados con el fin de determinar si deben realizarse nuevos ajustes en los parámetros (umbrales).

VI. REFERENCIAS BIBLIOGRÁFICAS

- Aguilar, F. J., & Mills, J., 2008. Accuracy assessment of lidar-derived digital elevation models. *The Photogrammetric Record*, *23*(122), 148-169.

- Akca, D., Gruen, A., Freeman, M., & Sargent, I., 2009. Fast quality control of 3D city models. In *The International LIDAR Mapping Forum, New Orleans, Louisiana, US, January* (pp. 26-28).

- Akel, N. A., Zilberstein, O., & Doytsher, Y., 2003, April. Automatic DTM extraction from dense raw LIDAR data in urban areas. In *Proc. FIG Working Week*.

- Alharthy, A., & Bethel, J., 2002. Heuristic filtering and 3D feature extraction from LIDAR data. *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, 34(3/A), 29-34.

- Alharthy, A., & Bethel, J., 2004. Detailed building reconstruction from airborne laser data using a moving surface method. *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, 35(B3), 213-218.

- Arefi, H., Engels, J., & Hahn, M., 2008. Levels of detail in 3D building reconstruction from lidar data. *ISPRS–International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 39(B3b), 485-490.

- Arefi, H., Engels, J., Hahn, M., & Mayer, H., 2007. Automatic DTM generation from laser-scanning data in residential hilly area. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36, 4.

- Arefi, H., & Hahn, M., 2005. A morphological reconstruction algorithm for separating off-terrain points from terrain points in laser scanning data.*International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(3/W19), 120-125.

- Axelsson, P., 2000. DEM generation from laser scanner data using adaptive TIN models. *International Archives of Photogrammetry and Remote Sensing*, 33(B4/1), 111-118.

- Baltsavias, E. P., 1999. A comparison between photogrammetry and laser scanning. *ISPRS Journal of photogrammetry and Remote Sensing*, 54(2), 83-94.

- Besl, P. J., & Jain, R. C., 1988. Segmentation through variable-order surface fitting. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 10(2), 167-192.

- Biosca, J. M., & Lerma, J. L., 2008. Unsupervised robust planar segmentation of terrestrial laser scanner point clouds based on fuzzy clustering methods. *ISPRS Journal of Photogrammetry and Remote Sensing*, 63(1), 84-98.

- Boykov, Y., Veksler, O., & Zabih, R., 2001. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(11), 1222-1239.

- Briese, C., Pfeifer, N., & Dorninger, P., 2002. Applications of the robust interpolation for DTM determination. *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, 34(3/A), 55-61.

- Brovelli, M. A., Cannata, M., & Longoni, U. M., 2004. LIDAR data filtering and DTM interpolation within GRASS. *Transactions in GIS*, 8(2), 155-174.

- Brunn, A., & Weidner, U., 1998. Hierarchical Bayesian nets for building extraction using dense digital surface models. *ISPRS Journal of Photogrammetry and Remote Sensing*, 53(5), 296-307.

- Burkardt. J. (s.f). Software y documentación adjunta. http://people.sc.fsu.edu/~jburkardt/ (2 de septiembre de 2009).

- Carlberg, M., Gao, P., Chen, G., & Zakhor, A., 2009, November. Classifying urban landscape in aerial LiDAR using 3D shape analysis. In *16th IEEE International Conference on Image Processing (ICIP)* (pp. 1701-1704).

- Chaabouni-Chouayakh, H., Krauss, T., d'Angelo, P., & Reinartz, P., 2010. 3D Change Detection Inside Urban Areas Using Different Digital Surface Models. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci*, 38, 86-91.

- Chauve, A., Mallet, C., Bretar, F., Durrieu, S., Deseilligny, M. P., & Puech, W., 2007, September. Processing full-waveform lidar data: modelling raw signals. In *International archives of photogrammetry, remote sensing and spatial information sciences* (pp. 102-107).

- Chehata, N., David, N., & Bretar, F., 2008, July. LIDAR data classification using hierarchical K-means clustering. In *ISPRS Congress Beijing 2008*, Vol. 37, 325-330.

- Chehata, N., Guo, L., & Mallet, C., 2009. Airborne lidar feature selection for urban classification using random forests. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 39(Part 3/W8), 207-212.

- Chen, H., Cheng, M., Li, J., & Liu, Y., 2012. An iterative terrain recovery approach to automated DTM generation from airborne LiDAR point clouds. *ISPRS–International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 39, 363-368.

- Chen, Q., Gong, P., Baldocchi, D., & Xie, G., 2007. Filtering airborne laser scanning data with morphological methods. *Photogrammetric Engineering & Remote Sensing*, 73(2), 175-185.

- Clark, M. L., Clark, D. B., & Roberts, D. A., 2004. Small-footprint lidar estimation of sub-canopy elevation and tree height in a tropical rain forest landscape. *Remote Sensing of Environment*, 91(1), 68-89.

- Costantino, D., & Angelini, M. G., 2011. Features and Ground Automatic Extraction from Airborne LIDAR Data. *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 38(5/W12), 19-24.

- Crosilla, F., Macorig, D., Sebastianutti, I., & Visintini, D., 2011. Points classification by a sequential higher—order moments statistical analysis of LIDAR data. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 38(5/W12). 1-6.

- Demantké, J., Mallet, C., David, N., & Vallet, B., 2011. Dimensionality Based Scale Selection in 3d LIDAR Point Clouds. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 38, 97-102.

- de Sevilla Riaza, T. F., Molina, A. G., Sampedro, F. J. A., Martínez, R., & Mas, F. M., 2008. Metodología para la caracterización de formaciones de vegetación de ribera y su morfología fluvial asociada utilizando datos LIDAR e imágenes digitales de alta resolución. *Topografía y cartografía: Revista del Ilustre Colegio Oficial de Ingenieros Técnicos en Topografía,* 25(147), 24-29.

- Doneus, M., & Briese, C., 2006. Full-waveform airborne laser scanning as a tool for archaeological reconnaissance. *BAR International Series*, 1568, 99.

- Doneus, M., Briese, C., Fera, M., & Janner, M., 2008. Archaeological prospection of forested areas using full-waveform airborne laser scanning. *Journal of Archaeological Science*, 35(4), 882-893.

- Dorninger, P., & Nothegger, C., 2007. 3D segmentation of unstructured point clouds for building modelling. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 35(3/W49A), 191-196.

- Ekhtari, N., Sahebi, M. R., Zoej, M. V., & Mohammadzadeh, A., 2008. Automatic building detection from LIDAR point cloud data. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 37(B4), 473-477.

- Elberink, S. O., & Maas, H. G., 2000. The use of anisotropic height texture measures for the segmentation of airborne laser scanner data. *International archives of photogrammetry and remote sensing*, 33(B3/2), 678-684.

- Elmqvist, M., 2001, March. Ground Estimation of Lasar Radar Data using Active Shape Models. *Paper presented at the OEEPE workshop on airborne laserscanning and interferometric SAR for detailed digital elevation models 1-3*, paper 5 (8 pages). Royal Institute of Technology Department of Geodesy and Photogrammetry 100 Stockholm, Sweden.

- Elmqvist, M., Jungert, E., Lantz, F., Persson, A., & Soderman, U., 2001. Terrain modelling and analysis using laser scanner data. *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, 34(3/W4), 219-226.

- Estornell, J., Ruiz, L. A., Velázquez-Martí, B., & Fernández-Sarría, A., 2011. Estimation of shrub biomass by airborne LiDAR data in small forest stands. *Forest Ecology and* Management, 262(9), 1697-1703.

- Estornell, J., Ruiz, L. A., Velázquez-Martí, B., & Hermosilla, T., 2012. Estimation of biomass and volume of shrub vegetation using LiDAR and spectral data in a Mediterranean environment. *Biomass and Bioenergy*, 46, 710-721.

- Evans, B., 2008. Automated bridge detection in DEMs via LiDAR data sources for urban flood modelling. In *Proc 11th Int. Conf. on Urban Drainage*.

- Evans, J. S., & Hudak, A. T., 2007. A multiscale curvature algorithm for classifying discrete return lidar in forested environments. *IEEE Transactions on Geoscience and Remote Sensing,* , 45(4), 1029-1038.

- Fagua, J., Campo, A., & Posada, E., 2011. Desarrollo de dos metodologías para la generación de modelos digitales de terreno (MDT) y superficie (MDS) empleando datos LiDAR y programas de licencia. *Análisis Geográfico*, 49, 83-95.

- Filin, S., & Pfeifer, N., 2005. Neighborhood systems for airborne laser data. *Photogrammetric Engineering & Remote Sensing*, 71(6), 743-755.

- Filin, S., & Pfeifer, N., 2006. Segmentation of airborne laser scanning data using a slope adaptive neighborhood. *ISPRS journal of Photogrammetry and Remote Sensing*, 60(2), 71-80.

- Fischler, M. A., & Bolles, R. C., 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6), 381-395.

- Flood, M., 2001. Lidar activities and research priorities in the commercial sector. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 34 (3/W4), 3-8.

- Forlani, G., Nardinocchi, C., Scaioni, M., & Zingaretti, P., 2006. Complete classification of raw LIDAR data and 3D reconstruction of buildings. *Pattern Analysis and Applications*, 8(4), 357-374.

- Forsberg, D, 2011. Automatic bridge detection in airborne laser scanned data. (Master of Science Thesis). Department of Computer Science and Engineering. University of Gothenburg.

- Ghosh, S., & Lohani, B., 2011. Heuristical feature extraction from LiDAR data and their visualization. In *Proceedings of the ISPRS Workshop on Laser Scanning* , September 2011, (Vol. 38). Canada: University of Calgary.

- Haala, N., & Brenner, C.,1999. Extraction of buildings and trees in urban environments. *ISPRS Journal of Photogrammetry and Remote Sensing*, 54(2), 130-137.

- Hartfield, K. A., Landau, K. I., & Van Leeuwen, W. J., 2011. Fusion of high resolution aerial multispectral and LiDAR data: Land cover in the context of urban mosquito habitat. *Remote Sensing*, 3(11), 2364-2383.

- Haugerud, R. A., & Harding, D. J., 2001. Some algorithms for virtual deforestation (VDF) of LIDAR topographic survey data. *International archives of photogrammetry remote sensing and spatial information sciences*, 34(3/W4), 211-218.

- Hermosilla, T., & Ruiz, L. Á., 2009. Detección automática de edificios combinando imágenes de satélite y datos lidar. *Semána Geomática,* 2.

- Hermosilla, T., Ruiz, L. A., Recio, J. A., & Estornell, J., 2011. Evaluation of automatic building detection approaches combining high resolution images and LiDAR data. *Remote Sensing*, 3(6), 1188-1210.

- Hofmann, A. D., 2004. Analysis of TIN-structure parameter spaces in airborne laser scanner data for 3-D building model generation. *International archives of photogrammetry, remote sensing and spatial information sciences*, 35(B3), 302-307.

- Höhle, J., & Höhle, M., 2009. Accuracy assessment of digital elevation models by means of robust statistical methods. *ISPRS Journal of Photogrammetry and Remote Sensing*, 64(4), 398-406.

- Hug, C., 1997. Extracting artificial surface objects from airborne laser scanner data. In *Automatic extraction of man-made objects from aerial and space images (II)* (pp. 203-212). Birkhäuser Basel.

- Hui, L., Di, L., Xianfeng, H., & Deren, L., 2008, July. Laser intensity used in classification of lidar point cloud data. In *IEEE International Geoscience and Remote Sensing Symposium, 2008. IGARSS 2008.* (Vol. 2, pp. II-1140).

- Hutton, C., & Brazier, R., 2012. Quantifying riparian zone structure from airborne LiDAR: Vegetation filtering, anisotropic interpolation, and uncertainty propagation. *Journal of Hydrology*, 442, 36-45.

- Hyyppä, J., Kelle, O., Lehikoinen, M., & Inkinen, M., 2001. A segmentation-based method to retrieve stem volume estimates from 3-D tree height models produced by laser scanners. *Geoscience and Remote Sensing, IEEE Transactions on*, 39(5), 969-975.

- Ju, H., 2012. A New Approach of Digital Bridge Surface Model Generation.*ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 1, 29-34.

- Keinan, E., & Doytsher, Y., 2008, April. Automatic methods toward generating digital true orthophoto by using dense lidar data. In *Proc. ASPRS 2008, Annual Conference, Portland, Oregon* (Vol. 28).

- Kilian, J., Haala, N., & Englich, M., 1996. Capture and evaluation of airborne laser scanner data. *International Archives of Photogrammetry and Remote Sensing*, 31, 383-388.

- Kim, C., Habib, A., & Mrstik, P., 2007. New approach for planar patch segmentation using airborne laser data. *Proceedings of the ASPRS conference*, Tampa, Florida, USA.

- Kim, K., & Shan, J., 2011, July. Adaptive morphological filtering for DEM generation. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS), 2011* (pp. 2539-2542).

- Kobler, A., Pfeifer, N., Ogrinc, P., Todorovski, L., Oštir, K., & Džeroski, S., 2007. Repetitive interpolation: A robust algorithm for DTM generation from Aerial Laser Scanner Data in forested terrain. *Remote Sensing of Environment*, 108(1), 9-23.

- Koch, B., Heyder, U., & Weinacker, H., 2006. Detection of individual tree crowns in airborne lidar data. *Photogrammetric Engineering & Remote Sensing*, 72(4), 357-363.

- Kraus, K., & Pfeifer, N., 1998. Determination of terrain models in wooded areas with airborne laser scanner data. *ISPRS Journal of Photogrammetry and remote Sensing*, 53(4), 193-203.

- Kwak, E., Al-Durgham, M., & Habib, A., 2012, July. Automatic 3D building model generation from lidar and image data using sequential minimum bounding rectangle. In *Proceedings of XXII ISPRS Congress, Melbourne, Australia*.

- Landa, A. F., Rodríguez, F., López, D., Olabarria, J. R. G., Yudego, B. M., Lasala, D., ... & Molina, A. G., 2013. Los sensores aerotransportados LiDAR y multiespectrales en la descripción y cuantificación de los recursos forestales. *Montes: revista de ámbito forestal*, (112), 31-35.

- Lari, Z., Habib, A., & Kwak, E., 2011, May. An adaptive approach for segmentation of 3D laser point cloud. In *ISPRS Workshop Laser Scanning* (pp. 29-31).

- Lari, Z., & Habib, A., 2012. Segmentation-based classification of laser scannin g data. In *ASPRS: Annual Conference, Sacramento, California, USA*.

- LASer (LAS) File Format Exchange Activities. Recuperado el 12 de enero de 2011 En: http://www.asprs.org/Committee-General/LASer-LAS-File-Format-Exchange-Activities.html

- Lee, I., & Schenk, T., 2001. Autonomous extraction of planar surfaces from airborne laser scanning data. In *Proc. ASPRS annual conference, St. Louis, MO, USA*.

- Lesparre, J., & Gorte, B. G. H., 2012, August. Simplified 3D city models from LiDAR. In *XXII ISPRS Congress, Commission II, Melbourne, Australia, 25 August-1 September 2012; IAPRS XXXIX-B2, 2012*. International Society for Photogrammetry and Remote Sensing (ISPRS).

- Li, Y., 2013. Filtering Airborne LiDAR Data by an Improved Morphological Method Based on Multi-Gradient Analysis. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40, 191-194.

- Lin, Y. C., Mills, J., & Smith-Voysey, S., 2008 . Detection of weak and overlapping pulses from waveform airborne laser scanning data. *Hill, R., Suárez, J. & Rosette, J.(Eds.), Proceedings of SilviLaser 2008*.

- Lin, Y. C., & Mills, J., 2009. Integration of full-waveform information into the airborne laser scanning data filtering process. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36, 224-229.

- Lin, X., & Zhang, J., 2014. Segmentation-based filtering of airborne LiDAR point clouds by progressive densification of terrain segments. *Remote Sensing*, 6(2), 1294-1326.

- Lin, X., Zhang, R., & Shen, J., 2012. A template-matching based approach for extraction of roads from very high-resolution remotely sensed imagery. *International Journal of Image and Data Fusion*, 3(2), 149-168.

- Linderberger. J., 1993. Laser-Profilmessungen zur topographischeb Geländeufnahme. *Deutsche Geodätiesche Kommission, Series C, Nº 400, Munich*.

- Linsen, L., & Prautzsch, H., 2001, September. Local versus global triangulations. In *EUROGRAPHICS'01*.

- Liu, J., Shen, J., Zhao, R., & Xu, S., 2013. Extraction of individual tree crowns from airborne LiDAR data in human settlements. *Mathematical and Computer Modelling*, 58(3), 524-535.

- Lohmann, P., 2002. Segmentation and filtering of laser scanner digital surface models. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 34(2), 311-316.

- Lohmann, P., Koch, A., & Schaeffer, M., 2000. Approaches to the filtering of laser scanner data. *International Archives of Photogrammetry and Remote Sensing*, 33(B3/1), 540-547.

- Maas, H. G., 1999, June. The potential of height texture measures for the segmentation of airborne laserscanner data. In *Fourth international airborne remote sensing conference and exhibition/21st Canadian symposium on remote sensing* (pp. 154-161).

- Márquez, A., Alejo, C., Pérez, M. (s.f.). Apuntes: Intersección de objetos. Recuperado el 21 de enero de 2009, del Sitio web del Departamento de Matemática Aplicada I de la Escuela Técnica Superior de Ingeniería Informática de la Universidad de Sevilla: http://personal.us.es/almar/docencia/practicas/localizacion/tema1.html

- Matikainen, L., Hyyppä, J., Ahokas, E., Markelin, L., & Kaartinen, H., 2010. Automatic detection of buildings and changes in buildings for updating of maps. *Remote Sensing*, 2(5), 1217-1248.

- Matikainen, L., Hyyppä, J., & Kaartinen, H., 2004. Automatic detection of changes from laser scanner and aerial image data for updating building maps. *International Archives of Photogrammetry, Remote sensing and Spatial Information Sciences*, 35(B2), 434-439.

- Matikainen, L., Hyyppä, J., & Kaartinen, H., 2009. Comparison between first pulse and last pulse laser scanner data in the automatic detection of buildings. *Photogrammetric Engineering & Remote Sensing*, 75(2), 133-146.

- Meng, X., Currit, N., & Zhao, K., 2010. Ground filtering algorithms for airborne LiDAR data: A review of critical issues. *Remote Sensing*, 2(3), 833-860.

- Meng, X., Wang, L., & Currit, N., 2009. Morphology-based building detection from airborne LIDAR data. *Photogrammetric Engineering & Remote Sensing*, 75(4), 437-442.

- Meng, X., Wang, L., Silván-Cárdenas, J. L., & Currit, N., 2009. A multi-directional ground filtering algorithm for airborne LIDAR. *ISPRS Journal of Photogrammetry and Remote Sensing*, 64(1), 117-124.

- Mongus, D., & Žalik, B., 2012. Parameter-free ground filtering of LiDAR data for automatic DTM generation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 67, 1-12.

- Moussa, A., & El-Sheimy, N., 2012. A new object based method for automated extraction of urban objects from airborne sensors data. *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, 39(B3), 309-314.

- Nakano, K., & Chikatsu, H., 2012. On Object Extraction Using Airborne Laser Scanner Data and Digital Images for 3d Modelling. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 1, 53-58.

- Nardinocchi, C., Forlani, G., & Zingaretti, P., 2003. Classification and filtering of laser data. *International Archives of Photogrammetry and Remote Sensing*, 34(3/W13).

- Naus, T., 2008. Unbiased LIDAR Data Measurements (Draft). *Retrieved from http://www.asprs.org/a/society/committees/lidar/Unbiased_measurement.pdf [116]*

- Niemeyer, J., Rottensteiner, F., & Soergel, U., 2012. Conditional random fields for lidar point cloud classification in complex urban areas. *ISPRS annals of the photogrammetry, remote sensing and spatial information sciences*, 1(3), 263-268.

- Oude Elberink, S. J., & Vosselman, G., 2009. 3D information extraction from laser point clouds covering complex road junctions. *The Photogrammetric Record*, 24(125), 23-36.

- Parian, J.A. and Sargent, I., 2007, Automatic height attribute assignment for building polygons: City modeling with level of detail zero. *Proceedings of the 8th Conference on Optical 3-D Measurement Techniques, Zurich, Switzerland,* (pp 371-378).

- Pauly, M., Keiser, R., & Gross, M., 2003, September. Multi-scale Feature Extraction on Point-Sampled Surfaces. In *Computer graphics forum*, 22(3), 281-289.

- Peng, M. H., & Shih, T. Y., 2006. Error assessment in two lidar-derived TIN datasets. *Photogrammetric Engineering & Remote Sensing*, 72(8), 933-947.

- Pérez-García, J. L., Delgado, J., Cardenal, J., Colomo, C., & Ureña, M. A., 2012. Progressive densification and region growing methods for LIDAR data classification. *International Archives of Photogrammetry and Remote Sensing*, 39(B3), 155-160.

- Pfeifer, N., & Mandlburger, G., 2009. LiDAR data filtering and DTM generation.*Topographic Laser Ranging and Scanning: Principles and Processing. CRC/Taylor & Francis*, 307-333.

- Pfeifer, N., Stadler, P., & Briese, C., 2001, March. Derivation of digital terrain models in the SCOP++ environment. In *Proceedings of OEEPE Workshop on Airborne Laserscanning and Interferometric SAR for Detailed Digital Terrain Models, Stockholm, Sweden* (Vol. 3612).

- Popescu, S. C., Wynne, R. H., & Nelson, R. F., 2002. Estimating plot-level tree heights with lidar: local filtering with a canopy-height based variable window size. *Computers and Electronics in Agriculture*, 37(1), 71-95.

- Pu, S., & Vosselman, G., 2006. Automatic extraction of building features from terrestrial laser scanning. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(5), 25-27.

- Rabbani, T., van den Heuvel, F., & Vosselmann, G., 2006. Segmentation of point clouds using smoothness constraint. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(5), 248-253.

- Rehor, M., Bähr, H. P., Tarsha-Kurdi, F., Landes, T., & Grussenmeyer, P., 2008. Contribution of two plane detection algorithms to recognition of intact and damaged buildings in lidar data. *The Photogrammetric Record*, 23(124), 441-456.

- Riaño, D., Chuvieco, E., Condés, S., González-Matesanz, J., & Ustin, S. L., 2004. Generation of crown bulk density for Pinus sylvestris L. from lidar. *Remote Sensing of Environment*, 92(3), 345-352.

- Roggero, M., 2001. Airborne laser scanning-clustering in raw data.*International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, 34(3/W4), 227-232.

- Rottensteiner, F., 2008, December. Automated updating of building data bases from digital surface models and multi-spectral images: Potential and limitations. In *ISPRS Congress, Beijing, China*, 37, 265-270.

- Samadzadegan, F., & Saeedi, S., 2009. Clustering of lidar data using particle swarm optimization algorithm in urban area. *In: Bretar F, Pierrot-Deseilligny M, Vosselman G (Eds) APRS Laser scanning 2009*, 38, 334-339.

- Sánchez, J., 2006. Tratamiento de datos LIDAR 3D en casco urbano. Segmentación de datos, detección y filtrado de entidades y cartografiado automático. *Proyecto final de carrera. E.T.S.I. Geodésica, Cartográfica y Topográfica*.

- Sánchez, J., & Lerma. J. L., 2012. Actualización de cartografía catastral urbana mediante LiDAR y SIG. *Geofocus,* 12, 53–70.

- Shaker, A., & El-Ashmawy, N. 2012. Land cover information extraction using lidar data. *International Archives of* the *Photogrammetry, Remote Sensing and Spatial Information Sciences*, *39*, 167-172.

- Shen, J., Liu, J., Zhao, R., & Lin, X., 2011, August. A Kd-Tree-Based Outlier Detection Method for Airborne LiDAR Point Clouds. In: *IEEE International Symposium on Image and Data Fusion (ISIDF),* 1-4.

- Shen, J., Liu, J., Lin, X., & Zhao, R.. 2012. Object-based classification of airborne light detection and ranging point clouds in human settlements. *Sensor Letters*, 10(1-2), 221-229.

- Silván-Cárdenas, J. L., & Wang, L., 2006. A multi-resolution approach for filtering LiDAR altimetry data. *ISPRS journal of photogrammetry and remote sensing*, 61(1), 11-22.

- Sithole, G., 2001. Filtering of laser altimetry data using a slope adaptive filter.*International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, 34(3/W4), 203-210.

- Sithole, G., 2005. *Segmentation and classification of airborne laser scanner data*. TU Delft, Delft University of Technology.

- Sithole, G., Vosselman, G., 2003. ISPRS test on extracting DEMs from point clouds: a comparison of existing automatic filters, *Technical Report*, 93 p.

- Sithole, G., & Vosselman, G., 2004. Experimental comparison of filter algorithms for bare-Earth extraction from airborne laser scanning point clouds.*ISPRS journal of photogrammetry and remote sensing*, 59(1), 85-101.

- Sithole, G., & Vosselman, G. 2005. Filtering of airborne laser scanner data based on segmented point clouds. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(3/W19), 66-71.

- Sohn, G., & Dowman, I. J., 2002. Terrain surface reconstruction by the use of tetrahedron model with the MDL criterion. *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, 34(3/A), 336-344.

- Sole, A., Giosa, L., Albano, R., & Cantisani, A., 2012. The Laser Scan Data as a Key Element in the Hydraulic Flood Modelling in Urban Areas. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40, 65-70.

- Song, J. H., Han, S. H., Yu, K. Y., & Kim, Y. I., 2002. Assessing the possibility of land-cover classification using lidar intensity data. *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, 34(3/B), 259-262.

- Sotoodeh, S., 2006. Outlier detection in laser scanner point clouds. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(5), 297-302.

- Sotoodeh, S., 2007. Hierarchical clustered outlier detection in laser scanner point clouds. *Laser07*, *36*, 383. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36*(*3/W52), 383-388.

- Susaki, J., 2012. Adaptive slope filtering of airborne LiDAR data in urban areas for digital terrain model (DTM) generation. *Remote Sensing*, 4(6), 1804-1819.

- Tarsha-Kurdi, F., Landes, T., & Grussenmeyer, P., 2007. Hough-transform and extended ransac algorithms for automatic detection of 3d building roof planes from lidar data. In *ISPRS Workshop on Laser Scanning 2007 and SilviLaser 2007,* 36, 407-412.

- Tóvári, D., & Pfeifer, N., 2005. Segmentation based robust interpolation-a new approach to laser data filtering. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Enschede, Netherlands*, 36(3/W19), 79-84.

- Tóvari, D., & Vögtle, T., 2004. Object classification in laserscanning data. *International Archives of Photogrammetry and Remote Sensing*, 36(8/W2).

- Trinder, J., & Salah, M., 2012. Aerial images and LiDAR data fusion for disaster change detection. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 1, 227-232.

- Ural, S., & Shan, J., 2012. Min-Cut Based Segmentation of Airborne LIDAR Point Clouds. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 39(B3), 167-172.

- Vogtle, T., & Steinle, E., 2000. 3D modelling of buildings using laser scanning and spectral information. *International Archives of Photogrammetry and Remote Sensing*, 33(B3/2), 927-934.

- Vögtle, T., & Steinle, E. 2004. Detection and recognition of changes in building geometry derived from multitemporal laserscanning data. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 35(B2), 428-433.

- Von Hansen, W., & Vögtle, T., 1999. Extraktion der geländeoberfläche aus flugzeuggetragenen laserscanner-aufnahmen. *Photogrammetrie Fernerkundung Geoinformation*, 229-236.

- Vosselman, G., 2000. Slope based filtering of laser altimetry data.*International Archives of Photogrammetry and Remote Sensing*, 33(B3/2), 935-942.

- Vosselman, G., & Dijkman, S., 2001. 3D building model reconstruction from point clouds and ground plans. *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, 34(3/W4), 37-44.

- Vosselman, G., Gorte, B. G. H., & Sithole, G., 2004. Change detection for updating medium scale maps using laser altimetry. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 34, 1-6.

- Verbree, E., & van Oosterom, P. J., 2001. Scanline Forced Delauney-TENs for Surface Representation. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 34(3/W4), 45-52.

- Vu, T. T., Matsuoka, M., & Yamazaki, F., 2004, September. LIDAR-based change detection of buildings in dense urban areas. In: *IEEE International Geoscience and Remote Sensing Symposium (IGARSS'04). Proceedings. 2004,* 5, 3413-3416.

- Wack, R., & Wimmer, A., 2002. Digital terrain models from airborne laserscanner data-a grid based approach. *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, 34(3/B), 293-296.

- Wang, C., Menenti, M., Stoll, M. P., Feola, A., Belluco, E., & Marani, M., 2009. Separation of ground and low vegetation signatures in LiDAR measurements of salt-marsh environments. *IEEE Transactions on Geoscience and Remote Sensing,* 47(7), 2014-2023.

- Wang, X., & Li, P., 2013. Urban Building Collapse Detection Using Very High Resolution Imagery and Airborne Lidar Data. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40, 127-132.

- Wang, J. H., Li, C. R., Tang, L. L., Zhou, M., & Li, J. M., 2012. A Comparison of Two Different Approaches of Point Cloud Classification Based on Full-Waveform Lidar Data. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 39, 179-182.

- Wang, M., & Tseng, Y. H., 2010. Automatic segmentation of LiDAR data into coplanar point clusters using an octree-based split-and-merge algorithm. *Photogrammetric Engineering & Remote Sensing*, 76(4), 407-420.

- Weiler, K.,1977. Hidden surface removal using polygon area sorting. *Computer Graphics*, 11(2), 214-222.

- Yu, X., Hyyppä, J., Kaartinen, H., & Maltamo, M., 2004. Automatic detection of harvested trees and determination of forest growth using airborne laser scanning. *Remote Sensing of Environment*, 90(4), 451-462.

- Yunfei, B., Guoping, L., Chunxiang, C., Xiaowen, L., Hao, Z., Qisheng, H., ... & Chaoyi, C., 2008. Classification of LIDAR point cloud and generation of DTM from LIDAR height and intensity data in forested area. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 37(7), 313-318.

- Zhang, K., & Whitman, D., 2005. Comparison of three algorithms for filtering airborne lidar data. *Photogrammetric Engineering & Remote Sensing*, 71(3), 313-324.

- Zheng, Y., Wang, C. Y., Chen, J. B., & He, D. X., 2013, May. A Method of Bridge Outline Extraction Based on Airborne LiDAR Data. In *Applied Mechanics and Materials*, 303, 1048-1055.

VII. ANEXO I. INFORMES DE PRECISIÓN Y ERRORES

ÍNDICE:

DESCRIPCIÓN DE PARÁMETROS UMBRALES UTILIZADOS

ALGORITMO A: DETECCIÓN DE ERRORES GROSEROS MEDIANTE ANÁLISIS DE VARIACIONES LOCALES DE DESNIVEL. DETECCIÓN DE PUNTOS DE ALTITUD ATÍPICA MUY DIFERENTE AL TERRENO.

DESCRIPCIÓN:

Se ejecuta el algoritmo simple. Los parámetros utilizados están definidos para detectar puntos de cota anómala muy diferente al terreno, tales como errores asociados a ruido del sensor o pulsos reflejados por altas estructuras u objetos (grúas, líneas eléctricas, aves, ...).

PARÁMETROS UMBRALES:

UMBRAL EN IZ: 10 m

UMBRAL EN TAMAÑO DE REGIONES: 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

UMBRAL EN %: Se ejecuta el algoritmo siete veces con diferentes parámetros umbrales:

A 1: 70 % A 2: 80 % A 3: 85 % A 4: 90 % A 5: 95 % A 6: 99 %

ALGORITMO B: DETECCIÓN DE ERRORES GROSEROS MEDIANTE ANÁLISIS DE VARIACIONES LOCALES DE DESNIVEL. DETECCIÓN DE PUNTOS DE ALTITUD ATÍPICA PRÓXIMOS AL TERRENO.

DESCRIPCIÓN:

Se ejecuta el algoritmo simple. Los parámetros utilizados están definidos para detectar puntos de cota anómala próximos al terreno, tales como errores asociados a ruido del sensor o pulsos reflejados en pequeñas estructuras u objetos de baja altura (pulsos reflejados sobre fachadas, personas, ...).

PARÁMETROS UMBRALES:

UMBRAL EN IZ: 3 m

UMBRAL EN TAMAÑO DE REGIONES: 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

UMBRAL EN %: Se ejecuta el algoritmo siete veces con diferentes parámetros umbrales:

B 1: 70 % B 2: 80 % B 3: 85 % B 4: 90 % B 5: 95 % B 6: 99 %

ALGORITMO C: DETECCIÓN DE ERRORES GROSEROS MEDIANTE ANÁLISIS DE VARIACIONES LOCALES DE DESNIVEL. ALGORITMO MULTIPROCESO CON PARÁMETROS ADAPTATIVOS.

DESCRIPCIÓN:

Se ejecuta el algoritmo multiproceso. El algoritmo concatena dos procesos con parámetros umbrales diferentes de forma consecutiva. Los parámetros utilizados por el primer proceso están definidos para detectar puntos de cota anómala muy diferente al terreno, tales como errores asociados a ruido del sensor o pulsos reflejados por altas estructuras u objetos (grúas, líneas eléctricas, aves, ...). Los parámetros umbrales del segundo proceso se ajustan para detectar puntos de cota anómala próximos al terreno, como pulsos reflejados por fachadas de edificos, personas, etc.

PARÁMETROS UMBRALES PROCESO 1:

UMBRAL EN IZ: 10 m

UMBRAL EN TAMAÑO DE REGIONES: 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

PARÁMETROS UMBRALES PROCESO 2:

UMBRAL EN IZ: 3 m

UMBRAL EN TAMAÑO DE REGIONES: 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

UMBRAL EN %: Se ejecuta el algoritmo nueve veces con diferentes parámetros umbrales:

C 1:	PROCESO 1: 80 % PROCESO 2: 90 %	**C 2:**	PROCESO 1: 80 % PROCESO 2: 95 %	**C 3:**	PROCESO 1: 80 % PROCESO 2: 99 %	**C 4:**	PROCESO 1: 85 % PROCESO 2: 90 %	**C 5:**	PROCESO 1: 85 % PROCESO 2: 95 %
C 6:	PROCESO 1: 85 % PROCESO 2: 99 %	**C 7:**	PROCESO 1: 90 % PROCESO 2: 90 %	**C 8:**	PROCESO 1: 90 % PROCESO 2: 95 %	**C 9:**	PROCESO 1: 90 % PROCESO 2: 99 %		

ANÁLISIS DE RESULTADOS (MATRICES DE CONFUSIÓN)

ALGORITMO: DETECCIÓN DE ERRORES GROSEROS MEDIANTE ANÁLISIS DE VARIACIONES LOCALES DE DESNIVEL. UN ÚNICO PROCESO (ALGORITMO A)

ZONA TEST: Vall d'Uixó

DESCRIPCIÓN:

Se ejecuta el algoritmo simple. Los parámetros utilizados están definidos para detectar puntos de cota anómala muy diferente al terreno, tales como errores asociados a ruido del sensor o pulsos reflejados por altas estructuras u objetos (grúas, líneas eléctricas, aves, ...).

ALGORITMO: A 1

PARÁMETROS UMBRALES:

UMBRAL EN IZ: 10 m

UMBRAL EN %: 70 %

UMBRAL EN TAMAÑO DE REGIONES: 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	Categorías deducidas de la clasificación				
	Outliers	No outliers	Total	Ex. Prod.	Error. Om.
Outliers	**126**	416	542	23,25 %	76,75 %
No outliers	98	**307198**	307296	99,97 %	0,03 %
Total	224	307614	**307838**	Exactitud global: 99,83 %	
Ex. Usuario	56,25 %	99,86 %			
Error. Com	43,75 %	0,14 %			

ALGORITMO: A 2

PARÁMETROS UMBRALES:

UMBRAL EN IZ: 10 m

UMBRAL EN %: 80 %

UMBRAL EN TAMAÑO DE REGIONES: 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	Categorías deducidas de la clasificación				
	Outliers	No outliers	Total	Ex. Prod.	Error. Om.
Outliers	**120**	422	542	22,14 %	77,86 %
No outliers	42	**307254**	307296	99,99 %	0,01 %
Total	162	307676	**307838**	Exactitud global: 99,85 %	
Ex. Usuario	74,07 %	99,86 %			
Error. Com	25,93 %	0,14 %			

ALGORITMO: A 3

PARÁMETROS UMBRALES:

UMBRAL EN IZ: 10 m

UMBRAL EN %: 85 %

UMBRAL EN TAMAÑO DE REGIONES: 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	Categorías deducidas de la clasificación				
	Outliers	No outliers	Total	Ex. Prod.	Error. Om.
Outliers	**116**	426	542	21,40 %	78,60 %
No outliers	0	**307296**	307296	100,00 %	0,00 %
Total	116	307722	**307838**	Exactitud global: 99,86 %	
Ex. Usuario	100,00 %	99,86 %			
Error. Com	0,00 %	0,14 %			

ALGORITMO: A 4

PARÁMETROS UMBRALES:

UMBRAL EN IZ: 10 m

UMBRAL EN %: 90 %

UMBRAL EN TAMAÑO DE REGIONES: 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	Categorías deducidas de la clasificación				
	Outliers	No outliers	Total	Ex. Prod.	Error. Om.
Outliers	**67**	475	542	12,36 %	87,64 %
No outliers	0	**307296**	307296	100,00 %	0,00 %
Total	67	307771	**307838**	Exactitud global: 99,85 %	
Ex. Usuario	100,00 %	99,85 %			
Error. Com	0,00 %	0,15 %			

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

ANÁLISIS DE RESULTADOS (MATRICES DE CONFUSIÓN)

ALGORITMO: A 5

PARÁMETROS UMBRALES:

UMBRAL EN IZ: 10 m

UMBRAL EN %: 95 %

UMBRAL EN TAMAÑO DE REGIONES: 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	**Categorías deducidas de la clasificación**				
	Outliers	**No outliers**	**Total**	**Ex. Prod.**	**Error. Om.**
Outliers	**8**	534	542	1,48 %	98,52 %
No outliers	0	**307296**	307296	100,00 %	0,00 %
Total	8	307830	**307838**	Exactitud global: 99,83 %	
Ex. Usuario	100,00 %	99,83 %			
Error. Com	0,00 %	0,17 %			

ALGORITMO: A 6

PARÁMETROS UMBRALES:

UMBRAL EN IZ: 10 m

UMBRAL EN %: 99 %

UMBRAL EN TAMAÑO DE REGIONES: 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	**Categorías deducidas de la clasificación**				
	Outliers	**No outliers**	**Total**	**Ex. Prod.**	**Error. Om.**
Outliers	**4**	538	542	0,74 %	99,26 %
No outliers	0	**307296**	307296	100,00 %	0,00 %
Total	4	307834	**307838**	Exactitud global: 99,83 %	
Ex. Usuario	100,00 %	99,83 %			
Error. Com	0,00 %	0,17 %			

ALGORITMO: DETECCIÓN DE ERRORES GROSEROS MEDIANTE ANÁLISIS DE VARIACIONES LOCALES DE DESNIVEL. UN ÚNICO PROCESO (ALGORITMO B)

ZONA TEST: Vall d'Uixó

DESCRIPCIÓN:

Se ejecuta el algoritmo simple. Los parámetros utilizados están definidos para detectar puntos de cota anómala próximos al terreno, tales como errores asociados a ruido del sensor o pulsos reflejados en pequeñas estructuras u objetos de baja altura (pulsos reflejados sobre fachadas, personas, ...).

ALGORITMO: B 1

PARÁMETROS UMBRALES:

UMBRAL EN IZ: 3 m

UMBRAL EN %: 70 %

UMBRAL EN TAMAÑO DE REGIONES: 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	**Categorías deducidas de la clasificación**				
	Outliers	**No outliers**	**Total**	**Ex. Prod.**	**Error. Om.**
Outliers	**541**	1	542	99,82 %	0,18 %
No outliers	5955	**301341**	307296	98,06 %	1,94 %
Total	6496	301342	**307838**	Exactitud global: 98,07 %	
Ex. Usuario	8,33 %	100,00 %			
Error. Com	91,67 %	0,00 %			

ALGORITMO: B 2

PARÁMETROS UMBRALES:

UMBRAL EN IZ: 3 m

UMBRAL EN %: 80 %

UMBRAL EN TAMAÑO DE REGIONES: 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	**Categorías deducidas de la clasificación**				
	Outliers	**No outliers**	**Total**	**Ex. Prod.**	**Error. Om.**
Outliers	**536**	6	542	98,89 %	1,11 %
No outliers	3019	**304277**	307296	99,02 %	0,98 %
Total	3555	304283	**307838**	Exactitud global: 99,02 %	
Ex. Usuario	15,08 %	100,00 %			
Error. Com	84,92 %	0,00 %			

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

ANÁLISIS DE RESULTADOS (MATRICES DE CONFUSIÓN)

ALGORITMO: B 3

PARÁMETROS UMBRALES:

UMBRAL EN IZ: 3 m

UMBRAL EN %: 85 %

UMBRAL EN TAMAÑO DE REGIONES: 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	**Categorías deducidas de la clasificación**				
	Outliers	**No outliers**	**Total**	**Ex. Prod.**	**Error. Om.**
Outliers	**534**	8	542	98,52 %	1,48 %
No outliers	2121	**305175**	307296	99,31 %	0,69 %
Total	2655	305183	**307838**	Exactitud global: 99,31 %	
Ex. Usuario	20,11 %	100,00 %			
Error. Com	79,89 %	0,00 %			

ALGORITMO: B 4

PARÁMETROS UMBRALES:

UMBRAL EN IZ: 3 m

UMBRAL EN %: 90 %

UMBRAL EN TAMAÑO DE REGIONES: 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	**Categorías deducidas de la clasificación**				
	Outliers	**No outliers**	**Total**	**Ex. Prod.**	**Error. Om.**
Outliers	**509**	33	542	93,91 %	6,09 %
No outliers	888	**306408**	307296	99,71 %	0,29 %
Total	1397	306441	**307838**	Exactitud global: 99,70 %	
Ex. Usuario	36,44 %	99,99 %			
Error. Com	63,56 %	0,01 %			

ALGORITMO: B 5

PARÁMETROS UMBRALES:

UMBRAL EN IZ: 3 m

UMBRAL EN %: 95 %

UMBRAL EN TAMAÑO DE REGIONES: 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	**Categorías deducidas de la clasificación**				
	Outliers	**No outliers**	**Total**	**Ex. Prod.**	**Error. Om.**
Outliers	**439**	103	542	81,00 %	19,00 %
No outliers	0	**307296**	307296	100,00 %	0,00 %
Total	439	307399	**307838**	Exactitud global: 99,97 %	
Ex. Usuario	100,00 %	99,97 %			
Error. Com	0,00 %	0,03 %			

ALGORITMO: B 6

PARÁMETROS UMBRALES:

UMBRAL EN IZ: 3 m

UMBRAL EN %: 99 %

UMBRAL EN TAMAÑO DE REGIONES: 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	**Categorías deducidas de la clasificación**				
	Outliers	**No outliers**	**Total**	**Ex. Prod.**	**Error. Om.**
Outliers	**61**	481	542	11,25 %	88,75 %
No outliers	0	**307296**	307296	100 %	0,00 %
Total	61	307777	**307838**	Exactitud global: 99,84 %	
Ex. Usuario	100,00 %	99,84 %			
Error. Com	0,00 %	0,16 %			

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

ANÁLISIS DE RESULTADOS (MATRICES DE CONFUSIÓN)

ALGORITMO: DETECCIÓN DE ERRORES GROSEROS MEDIANTE ANÁLISIS DE VARIACIONES LOCALES DE DESNIVEL. ALGORITMO MULTIPROCESO CON PARÁMETROS ADAPTATIVOS (ALGORITMO C)

ZONA TEST: Vall d'Uixó

DESCRIPCIÓN:

Se ejecuta el algoritmo multiproceso. El algoritmo concatena dos procesos con parámetros umbrales diferentes de forma consecutiva. Los parámetros utilizados por el primer proceso están definidos para detectar puntos de cota anómala muy diferente al terreno, tales como errores asociados a ruido del sensor o pulsos reflejados por altas estructuras u objetos (grúas, líneas eléctricas, aves, ...). Los parámetros umbrales del segundo proceso se ajustan para detectar puntos de cota anómala próximos al terreno, como pulsos reflejados por fachadas de edificos, personas, etc.

ANÁLISIS DE RESULTADOS:

ALGORITMO: C 1

PARÁMETROS UMBRALES:

PROCESO 1	PROCESO 2
UMBRAL EN IZ: 10 m	**UMBRAL EN IZ:** 3 m
UMBRAL EN %: 80 %	**UMBRAL EN %:** 90 %
UMBRAL EN TAMAÑO DE REGIONES: 15 m^2	**UMBRAL EN TAMAÑO DE REGIONES:** 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	**Categorías deducidas de la clasificación**				
	Outliers	**No outliers**	**Total**	**Ex. Prod.**	**Error. Om.**
Outliers	**533**	9	542	98,34 %	1,66 %
No outliers	926	**306370**	307296	99,70 %	0,30 %
Total	1459	306379	**307838**	Exactitud global: 99,70 %	
Ex. Usuario	36,53 %	100,00 %			
Error. Com	63,47 %	0,00 %			

ALGORITMO: C 2

PARÁMETROS UMBRALES:

PROCESO 1	PROCESO 2
UMBRAL EN IZ: 10 m	**UMBRAL EN IZ:** 3 m
UMBRAL EN %: 80 %	**UMBRAL EN %:** 95 %
UMBRAL EN TAMAÑO DE REGIONES: 15 m^2	**UMBRAL EN TAMAÑO DE REGIONES:** 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	**Categorías deducidas de la clasificación**				
	Outliers	**No outliers**	**Total**	**Ex. Prod.**	**Error. Om.**
Outliers	**521**	21	542	96,13 %	3,87 %
No outliers	44	**307252**	307296	99,99 %	0,01 %
Total	565	307273	**307838**	Exactitud global: 99,98 %	
Ex. Usuario	92,21 %	99,99 %			
Error. Com	7,79 %	0,01 %			

ALGORITMO: C 3

PARÁMETROS UMBRALES:

PROCESO 1	PROCESO 2
UMBRAL EN IZ: 10 m	**UMBRAL EN IZ:** 3 m
UMBRAL EN %: 80 %	**UMBRAL EN %:** 99 %
UMBRAL EN TAMAÑO DE REGIONES: 15 m^2	**UMBRAL EN TAMAÑO DE REGIONES:** 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	**Categorías deducidas de la clasificación**				
	Outliers	**No outliers**	**Total**	**Ex. Prod.**	**Error. Om.**
Outliers	**181**	361	542	33,39 %	66,61 %
No outliers	42	**307254**	307296	99,99 %	0,01 %
Total	223	307615	**307838**	Exactitud global: 99,87 %	
Ex. Usuario	81,17 %	99,88 %			
Error. Com	18,83 %	0,12 %			

ALGORITMO: C 4

PARÁMETROS UMBRALES:

PROCESO 1	PROCESO 2
UMBRAL EN IZ: 10 m	**UMBRAL EN IZ:** 3 m
UMBRAL EN %: 85 %	**UMBRAL EN %:** 90 %
UMBRAL EN TAMAÑO DE REGIONES: 15 m^2	**UMBRAL EN TAMAÑO DE REGIONES:** 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	**Categorías deducidas de la clasificación**				
	Outliers	**No outliers**	**Total**	**Ex. Prod.**	**Error. Om.**
Outliers	**533**	9	542	98,34 %	1,66 %
No outliers	888	**306408**	307296	99,71 %	0,29 %
Total	1421	306417	**307838**	Exactitud global: 99,71 %	
Ex. Usuario	37,51 %	100,00 %			
Error. Com	62,49 %	0,00 %			

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

ALGORITMO: C 5

PARÁMETROS UMBRALES:

PROCESO 1	PROCESO 2
UMBRAL EN IZ: 10 m	**UMBRAL EN IZ:** 3 m
UMBRAL EN %: 85 %	**UMBRAL EN %:** 95 %
UMBRAL EN TAMAÑO DE REGIONES: 15 m^2	**UMBRAL EN TAMAÑO DE REGIONES:** 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	**Categorías deducidas de la clasificación**				
	Outliers	**No outliers**	**Total**	**Ex. Prod.**	**Error. Om.**
Outliers	**521**	21	542	96,13 %	3,87 %
No outliers	0	**307296**	307296	100,00 %	0,00 %
Total	521	307317	**307838**	Exactitud global: 99,99 %	
Ex. Usuario	100,00 %	99,99 %			
Error. Com	0,00 %	0,01 %			

ALGORITMO: C 6

PARÁMETROS UMBRALES:

PROCESO 1	PROCESO 2
UMBRAL EN IZ: 10 m	**UMBRAL EN IZ:** 3 m
UMBRAL EN %: 85 %	**UMBRAL EN %:** 99 %
UMBRAL EN TAMAÑO DE REGIONES: 15 m^2	**UMBRAL EN TAMAÑO DE REGIONES:** 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	**Categorías deducidas de la clasificación**				
	Outliers	**No outliers**	**Total**	**Ex. Prod.**	**Error. Om.**
Outliers	**180**	362	542	33,21 %	66,79 %
No outliers	0	**307296**	307296	100,00 %	0,00 %
Total	180	307658	**307838**	Exactitud global: 99,88 %	
Ex. Usuario	100,00 %	99,88 %			
Error. Com	0,00 %	0,12 %			

ALGORITMO: C 7

PARÁMETROS UMBRALES:

PROCESO 1	PROCESO 2
UMBRAL EN IZ: 10 m	**UMBRAL EN IZ:** 3 m
UMBRAL EN %: 90 %	**UMBRAL EN %:** 90 %
UMBRAL EN TAMAÑO DE REGIONES: 15 m^2	**UMBRAL EN TAMAÑO DE REGIONES:** 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	**Categorías deducidas de la clasificación**				
	Outliers	**No outliers**	**Total**	**Ex. Prod.**	**Error. Om.**
Outliers	**509**	33	542	93,91 %	6,09 %
No outliers	888	**306408**	307296	99,71 %	0,29 %
Total	1397	306441	**307838**	Exactitud global: 99,70 %	
Ex. Usuario	36,44 %	99,99 %			
Error. Com	63,56 %	0,01 %			

ALGORITMO: C 8

PARÁMETROS UMBRALES:

PROCESO 1	PROCESO 2
UMBRAL EN IZ: 10 m	**UMBRAL EN IZ:** 3 m
UMBRAL EN %: 90 %	**UMBRAL EN %:** 95 %
UMBRAL EN TAMAÑO DE REGIONES: 15 m^2	**UMBRAL EN TAMAÑO DE REGIONES:** 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	**Categorías deducidas de la clasificación**				
	Outliers	**No outliers**	**Total**	**Ex. Prod.**	**Error. Om.**
Outliers	**489**	53	542	90,22 %	9,78 %
No outliers	0	**307296**	307296	100,00 %	0,01 %
Total	489	307349	**307838**	Exactitud global: 99,98 %	
Ex. Usuario	100,00 %	99,98 %			
Error. Com	0,00 %	0,02 %			

ALGORITMO: C 9

PARÁMETROS UMBRALES:

PROCESO 1	PROCESO 2
UMBRAL EN IZ: 10 m	**UMBRAL EN IZ:** 3 m
UMBRAL EN %: 90 %	**UMBRAL EN %:** 99 %
UMBRAL EN TAMAÑO DE REGIONES: 15 m^2	**UMBRAL EN TAMAÑO DE REGIONES:** 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	**Categorías deducidas de la clasificación**				
	Outliers	**No outliers**	**Total**	**Ex. Prod.**	**Error. Om.**
Outliers	**127**	415	542	23,43 %	76,57 %
No outliers	0	**307296**	307296	100,00 %	0,00 %
Total	127	307711	**307838**	Exactitud global: 99,87 %	
Ex. Usuario	100,00 %	99,87 %			
Error. Com	0,00 %	0,13 %			

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

ANÁLISIS DE RESULTADOS (ANÁLISIS GRÁFICO)

ALGORITMO: DETECCIÓN DE ERRORES GROSEROS MEDIANTE ANÁLISIS DE VARIACIONES LOCALES DE DESNIVEL. UN ÚNICO PROCESO (ALGORITMO A)

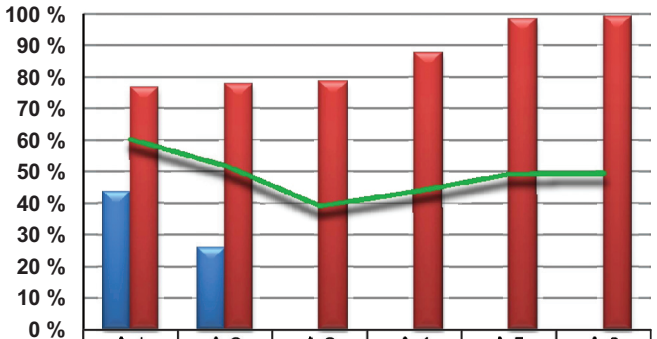
ZONA TEST: Vall d'Uixó

DESCRIPCIÓN:

Se ejecuta el algoritmo simple. Los parámetros utilizados están definidos para detectar puntos de cota anómala muy diferente al terreno, tales como errores asociados a ruido del sensor o pulsos reflejados por altas estructuras u objetos (grúas, líneas eléctricas, aves, ...).

GRAFICO 1:

Muestra los errores de comisión y omisión obtenidos tras ejecutar el algoritmo. El algoritmo se ejectuta 6 veces utilizando distinto parámetro umbral en porcentaje. Además se muestra mediante una línea la media del error de comisión y omisión para cada uno de los resultados obtenidos

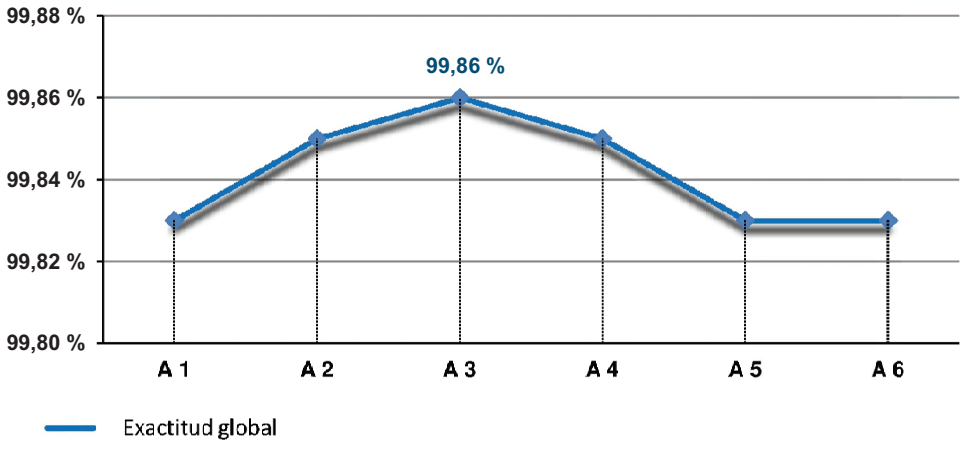


	A 1	A 2	A 3	A 4	A 5	A 6
Error de comisión (%)	43,75	25,93	0	0	0	0
Error de omisión (%)	76,75	77,86	78,6	87,64	98,52	99,26
Media de errores (%)	60,25	51,90	39,30	43,82	49,26	49,63

Media de errores: Es la media aritmética entre el error de omisión y el error de comisión.

GRAFICO 2:

Muestra la exactitud global de la clasificación para cada uno de los seis resultados obtenidos.



ANÁLISIS DE RESULTADOS (ANÁLISIS GRÁFICO)

ALGORITMO: DETECCIÓN DE ERRORES GROSEROS MEDIANTE ANÁLISIS DE VARIACIONES LOCALES DE DESNIVEL. UN ÚNICO PROCESO (ALGORITMO B)

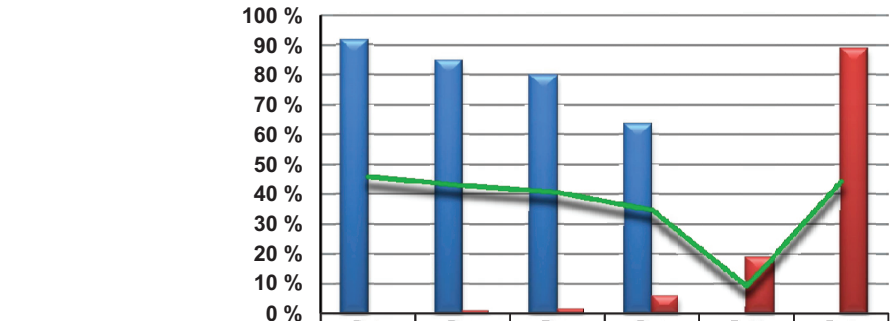
ZONA TEST: Vall d'Uixó

DESCRIPCIÓN:

Se ejecuta el algoritmo simple. Los parámetros utilizados están definidos para detectar puntos de cota anómala próximos al terreno, tales como errores asociados a ruido del sensor o pulsos reflejados en pequeñas estructuras u objetos de baja altura (pulsos reflejados sobre fachadas, personas, ...).

GRAFICO 1:

Muestra los errores de comisión y omisión obtenidos tras ejecutar el algoritmo. El algoritmo se ejectuta 6 veces utilizando distinto parámetro umbral en porcentaje. Además se muestra mediante una línea la media del error de comisión y omisión para cada uno de los resultados obtenidos

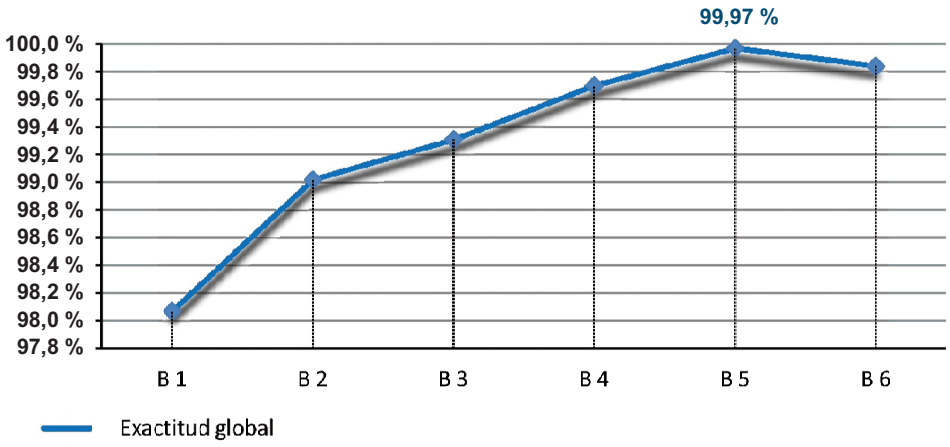


	B 1	B 2	B 3	B 4	B 5	B 6
Error de comisión (%)	91,67	84,92	79,89	63,56	0	0
Error de omisión (%)	0,18	1,11	1,48	6,09	19	88,75
Media de errores (%)	45,925	43,015	40,685	34,825	9,5	44,375

Media de errores: Es la media aritmética entre el error de omisión y el error de comisión.

GRAFICO 2:

Muestra la exactitud global de la clasificación para cada uno de los seis resultados obtenidos.



ANÁLISIS DE RESULTADOS (ANÁLISIS GRÁFICO)

ALGORITMO: DETECCIÓN DE ERRORES GROSEROS MEDIANTE ANÁLISIS DE VARIACIONES LOCALES DE DESNIVEL. ALGORITMO MULTIPROCESO CON PARÁMETROS ADAPTATIVOS (ALGORITMO C)

ZONA TEST: Vall d'Uixó

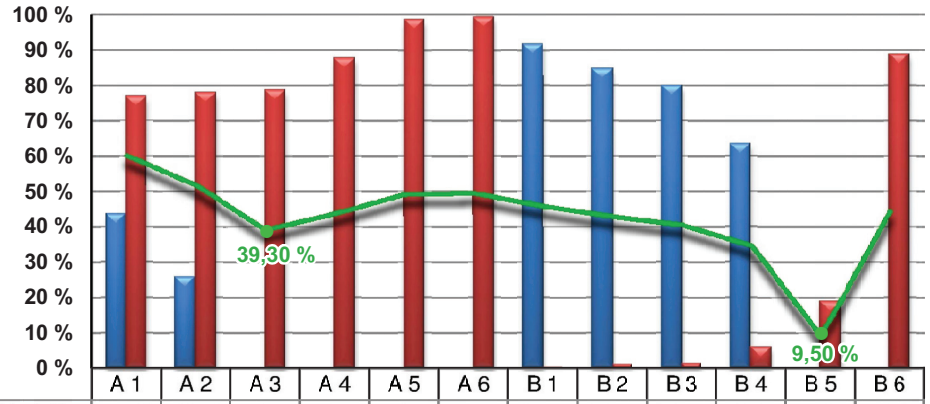
DESCRIPCIÓN:

Se ejecuta el algoritmo multiproceso. El algoritmo concatena dos procesos con parámetros umbrales diferentes de forma consecutiva. Los parámetros utilizados por el primer proceso están definidos para detectar puntos de cota anómala muy diferente al terreno, tales como errores asociados a ruido del sensor o pulsos reflejados por altas estructuras u objetos (grúas, líneas eléctricas, aves, ...). Los parámetros umbrales del segundo proceso se ajustan para detectar puntos de cota anómala próximos al terreno, como pulsos reflejados por fachadas de edificos, personas, etc.

Previamente a la selección de los parámetros umbrales aosciados a ambos procesos se analizan los parámetros umbrales que mejores resultados han obtenido tanto para para detectar puntos de cota anómala muy diferente al terreno (algoritmo A) así como para detectar puntos de cota anómal próximos al terreno (algoritmo B)

GRAFICO 1:

Muestra los errores de comisión y omisión obtenidos tras ejecutar los algoritmos simples (Algoritmo A y algoritmo B), así como la media del error de omisión y comisión. Cada uno de los algoritmos se ha ejecutado seis veces con distintos parámetros umbrales.



	A 1	A 2	A 3	A 4	A 5	A 6	B 1	B 2	B 3	B 4	B 5	B 6
Error de comisión (%)	43,75	25,93	0	0	0	0	91,67	84,92	79,89	63,56	0,00	0,00
Error de cmisión (%)	76,75	77,86	78,6	87,64	98,52	99,26	0,18	1,·1	1,48	6,09	19,00	88,75
Mecia de errores (%)	60,25	51,90	39,30	43,82	49,26	49,63	45,93	43,02	40,69	34,83	9,50	44,38

Media de errores: Es la media aritmética entre el error de omisión y el error de comisión.

GRAFICO 2:

Muestra la exactitud global de la clasificación para cada uno de los seis resultados obtenidos asociados a los algoritmos ejecutados (algoitmo A y algoritmo B).

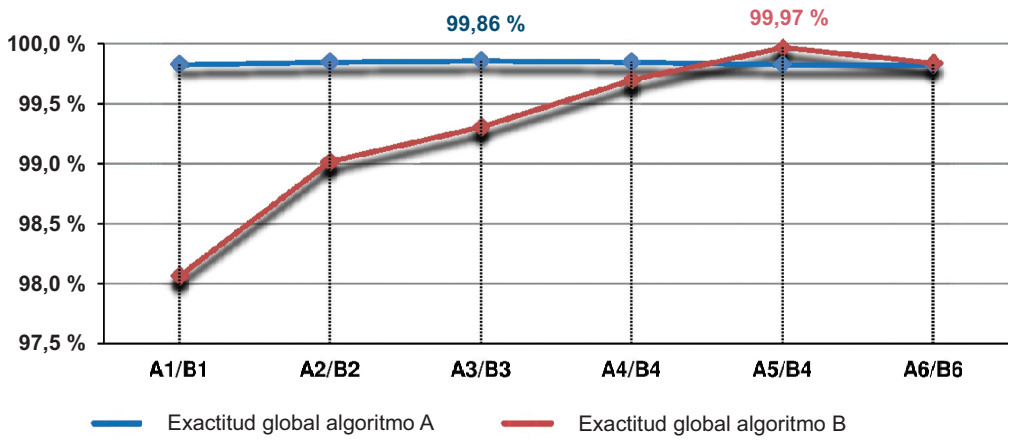
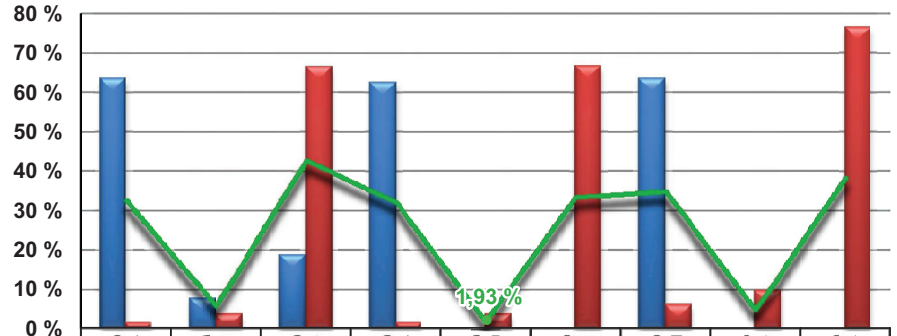


GRAFICO 3:

Muestra los errores de comisión y omisión obtenidos tras ejecutar los algoritmos compuesto (Algoritmo C), así como la media del error de omisión y comisión. El algoritmo se ejectua nueve veces con distintos parámetros umbrales.



	C 1	C 2	C 3	C 4	C 5	C 6	C 7	C 8	C 9
Error de comisión (%)	63,47	7,79	18,83	62,49	0	0	63,56	0	0
Error de omisión (%)	1,66	3,87	66,61	1,66	3,87	66,79	6,09	9,78	76,57
Media de errores (%)	32,565	5.83	42,72	32,075	1,935	33,395	34,825	4,89	38,285

GRAFICO 4:

Muestra la exactitud global de la clasificación para cada uno de los nueve resultados obtenidos asociados tras ejecutar el algoritmo (algoritmo C).

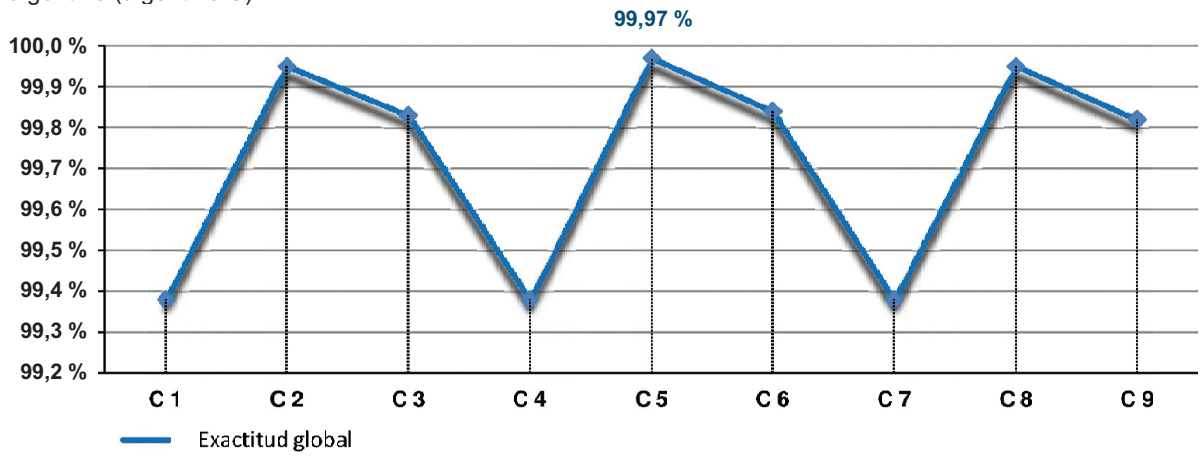
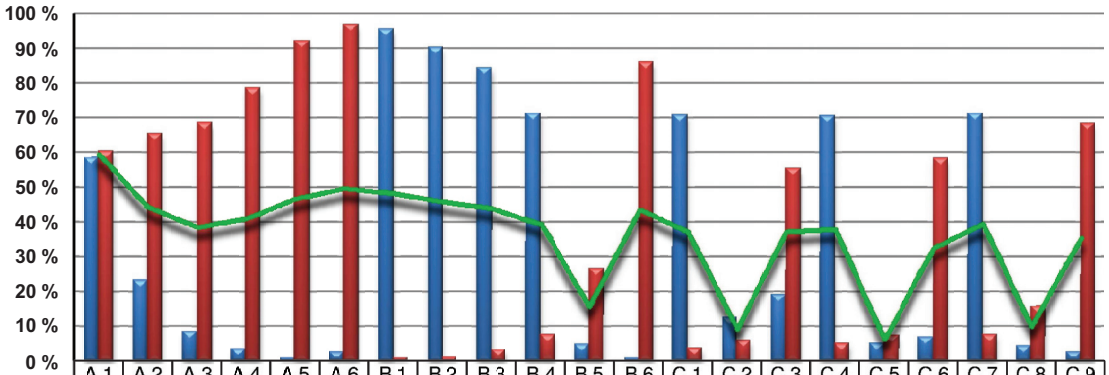


GRAFICO 5:

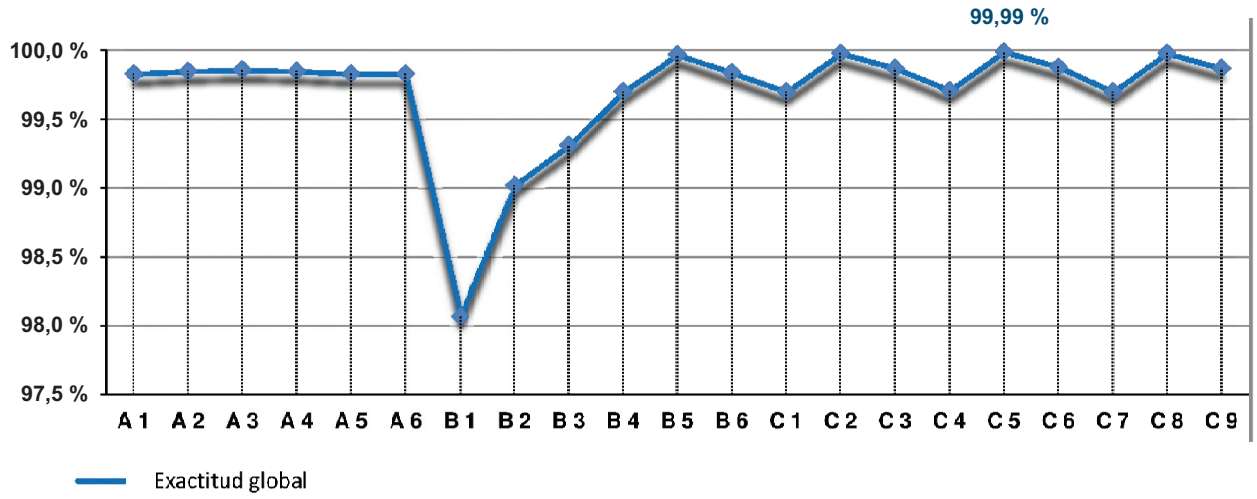
Muestra los errores de comisión y omisión obtenidos tras ejecutar los algoritmos simples (algoritmo A y B) así como el algoritmo compuesto (algoritmo C). Además se muestra la media de los errores.



	A 1	A 2	A 3	A 4	A 5	A 6	B 1	B 2	B 3	B 4	B 5	B 6	C 1	C 2	C 3	C 4	C 5	C 6	C 7	C 8	C 9
Error de comisión (%)	58,53	23,2	8,53	3,42	1,02	2,43	95,43	90,37	84,38	71,16	4,77	1,04	70,67	12,45	19,10	70,63	5,14	6,75	71,16	4,26	2,54
Error de omisión (%)	60,34	65,32	68,4	78,42	92,18	96,77	1,00	1,23	3,31	7,49	26,52	85,95	3,76	5,97	55,25	5,04	7,39	58,37	7,50	15,64	68,18
Media de errores (%)	59,44	44,26	38,47	40,92	46,60	49,60	48,22	45,80	43,85	39,33	15,65	43,50	37,22	9,21	37,18	37,84	6,27	32,56	39,33	9,95	35,36

GRAFICO 6:

Muestra la exactitud global de la clasificación para cada uno de los nueve resultados obtenidos asociados tras ejecutar los algoritmos simples (algoritmo A y B) así como el algoritmo compuesto (algoritmo C).



ALGORITMO: DETECCIÓN DE ERRORES GROSEROS MEDIANTE ANÁLISIS DE VARIACIONES LOCALES DE DESNIVEL. UN ÚNICO PROCESO (ALGORITMO A)

ZONA TEST: Collado de Villalba

DESCRIPCIÓN:

Se ejecuta el algoritmo simple. Los parámetros utilizados están definidos para detectar puntos de cota anómala muy diferente al terreno, tales como errores asociados a ruido del sensor o pulsos reflejados por altas estructuras u objetos (grúas, líneas eléctricas, aves, ...).

ALGORITMO: A 1

PARÁMETROS UMBRALES:

UMBRAL EN IZ: 10 m

UMBRAL EN %: 70 %

UMBRAL EN TAMAÑO DE REGIONES: 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	Categorías deducidas de la clasificación				
	Outliers	No outliers	Total	Ex. Prod.	Error. Om.
Outliers	**2961**	4505	7466	39,66 %	60,34 %
No outliers	4179	**2820550**	2824729	99,85 %	0,15 %
Total	7140	2825055	**2832195**	Exactitud global: 99,69 %	
Ex. Usuario	41,47 %	99,84 %			
Error. Com	58,53 %	0,16 %			

ALGORITMO: A 2

PARÁMETROS UMBRALES:

UMBRAL EN IZ: 10 m

UMBRAL EN %: 80 %

UMBRAL EN TAMAÑO DE REGIONES: 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	Categorías deducidas de la clasificación				
	Outliers	No outliers	Total	Ex. Prod.	Error. Om.
Outliers	**2589**	4877	7466	34,68 %	65,32 %
No outliers	782	**2823947**	2824729	99,97 %	0,03 %
Total	3371	2828824	**2832195**	Exactitud global: 99,80 %	
Ex. Usuario	76,80 %	99,83 %			
Error. Com	23,20 %	0,17 %			

ALGORITMO: A 3

PARÁMETROS UMBRALES:

UMBRAL EN IZ: 10 m

UMBRAL EN %: 85 %

UMBRAL EN TAMAÑO DE REGIONES: 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	Categorías deducidas de la clasificación				
	Outliers	No outliers	Total	Ex. Prod.	Error. Om.
Outliers	**2359**	5107	7466	31,60 %	68,40 %
No outliers	220	**2824509**	2824729	99,99 %	0,01 %
Total	2579	2829616	**2832195**	Exactitud global: 99,81 %	
Ex. Usuario	91,47 %	99,82 %			
Error. Com	8,53 %	0,18 %			

ALGORITMO: A 4

PARÁMETROS UMBRALES:

UMBRAL EN IZ: 10 m

UMBRAL EN %: 90 %

UMBRAL EN TAMAÑO DE REGIONES: 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	Categorías deducidas de la clasificación				
	Outliers	No outliers	Total	Ex. Prod.	Error. Om.
Outliers	**1611**	5855	7466	21.58 %	78,42 %
No outliers	57	**2824672**	2824729	100 %	0,00 %
Total	1668	2830527	**2832195**	Exactitud global: 99,79 %	
Ex. Usuario	96,58 %	99,79 %			
Error. Com	3,42 %	0,21 %			

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

ANÁLISIS DE RESULTADOS (MATRICES DE CONFUSIÓN)

ALGORITMO: A 5

PARÁMETROS UMBRALES:

UMBRAL EN IZ: 10 m

UMBRAL EN %: 95 %

UMBRAL EN TAMAÑO DE REGIONES: 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	**Categorías deducidas de la clasificación**				
	Outliers	**No outliers**	**Total**	**Ex. Prod.**	**Error. Om.**
Outliers	**584**	6882	7466	7,82 %	92,18 %
No outliers	6	**2824723**	2824729	100,00 %	0,00 %
Total	590	2831605	**2832195**	Exactitud global: 99,76 %	
Ex. Usuario	98,98 %	99,76 %			
Error. Com	1,02 %	0,24 %			

ALGORITMO: A 6

PARÁMETROS UMBRALES:

UMBRAL EN IZ: 10 m

UMBRAL EN %: 99 %

UMBRAL EN TAMAÑO DE REGIONES: 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	**Categorías deducidas de la clasificación**				
	Outliers	**No outliers**	**Total**	**Ex. Prod.**	**Error. Om.**
Outliers	**241**	7225	7466	3,23 %	96,77 %
No outliers	6	**2824723**	2824729	100,00 %	0,00 %
Total	247	2831948	**2832195**	Exactitud global: 99,74%	
Ex. Usuario	97,57 %	99,74 %			
Error. Com	2,43 %	0,26 %			

ALGORITMO: DETECCIÓN DE ERRORES GROSEROS MEDIANTE ANÁLISIS DE VARIACIONES LOCALES DE DESNIVEL. UN ÚNICO PROCESO (ALGORITMO B)

ZONA TEST: Collado de Villalba

DESCRIPCIÓN:

Se ejecuta el algoritmo simple. Los parámetros utilizados están definidos para detectar puntos de cota anómala próximos al terreno, tales como errores asociados a ruido del sensor o pulsos reflejados en pequeñas estructuras u objetos de baja altura (pulsos reflejados sobre fachadas, personas, ...).

ALGORITMO: B 1

PARÁMETROS UMBRALES:

UMBRAL EN IZ: 3 m

UMBRAL EN %: 70 %

UMBRAL EN TAMAÑO DE REGIONES: 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	**Categorías deducidas de la clasificación**				
	Outliers	**No outliers**	**Total**	**Ex. Prod.**	**Error. Om.**
Outliers	**7391**	75	7466	99,00 %	1,00 %
No outliers	154373	**2670356**	2824729	94,53 %	5,47 %
Total	161764	2670431	**2832195**	Exactitud global: 94,55 %	
Ex. Usuario	4,57 %	100,00 %			
Error. Com	95,43 %	0,00 %			

ALGORITMO: B 2

PARÁMETROS UMBRALES:

UMBRAL EN IZ: 3 m

UMBRAL EN %: 80 %

UMBRAL EN TAMAÑO DE REGIONES: 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	**Categorías deducidas de la clasificación**				
	Outliers	**No outliers**	**Total**	**Ex. Prod.**	**Error. Om.**
Outliers	**7374**	92	7466	98,77 %	1,23 %
No outliers	69207	**2755522**	2824729	97,55 %	2,45 %
Total	76581	2755614	**2832195**	Exactitud global: 99,97 %	
Ex. Usuario	9,63 %	100,00 %			
Error. Com	90,37 %	0,00 %			

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

ALGORITMO: B 3

PARÁMETROS UMBRALES:

UMBRAL EN IZ: 3 m

UMBRAL EN %: 85 %

UMBRAL EN TAMAÑO DE REGIONES: 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	Categorías deducidas de la clasificación				
	Outliers	No outliers	Total	Ex. Prod.	Error. Om.
Outliers	**7219**	247	7466	96,69 %	3,31 %
No outliers	39003	**2785726**	2824729	98,62 %	1,38 %
Total	46222	2785973	**2832195**	Exactitud global: 98,61 %	
Ex. Usuario	15,62 %	99,99 %			
Error. Com	84,38 %	0,01 %			

ALGORITMO: B 4

PARÁMETROS UMBRALES:

UMBRAL EN IZ: 3 m

UMBRAL EN %: 90 %

UMBRAL EN TAMAÑO DE REGIONES: 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	Categorías deducidas de la clasificación				
	Outliers	No outliers	Total	Ex. Prod.	Error. Om.
Outliers	**6909**	557	7466	92,54 %	7,46 %
No outliers	17046	**2807683**	2824729	99,40 %	0,60 %
Total	23955	2808240	**2832195**	Exactitud global: 99,38 %	
Ex. Usuario	28,84 %	99,98 %			
Error. Com	71,16 %	0,02 %			

ALGORITMO: B 5

PARÁMETROS UMBRALES:

UMBRAL EN IZ: 3 m

UMBRAL EN %: 95 %

UMBRAL EN TAMAÑO DE REGIONES: 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	Categorías deducidas de la clasificación				
	Outliers	No outliers	Total	Ex. Prod.	Error. Om.
Outliers	**5486**	1980	7466	73,48 %	26,52 %
No outliers	275	**2824454**	2824729	99,99 %	0,01 %
Total	5761	2826434	**2832195**	Exactitud global: 99,92 %	
Ex. Usuario	95,23 %	99,93 %			
Error. Com	4,77 %	0,07 %			

ALGORITMO: B 6

PARÁMETROS UMBRALES:

UMBRAL EN IZ: 3 m

UMBRAL EN %: 99 %

UMBRAL EN TAMAÑO DE REGIONES: 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	Categorías deducidas de la clasificación				
	Outliers	No outliers	Total	Ex. Prod.	Error. Om.
Outliers	**1049**	6417	7466	14,05 %	85,95 %
No outliers	11	**2824718**	2824729	100 %	0,00 %
Total	1060	2831135	**2832195**	Exactitud global: 99,77 %	
Ex. Usuario	98,96 %	99,77 %			
Error. Com	1,04 %	0,23 %			

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

ANÁLISIS DE RESULTADOS (MATRICES DE CONFUSIÓN)

ALGORITMO: DETECCIÓN DE ERRORES GROSEROS MEDIANTE ANÁLISIS DE VARIACIONES LOCALES DE DESNIVEL. ALGORITMO MULTIPROCESO CON PARÁMETROS ADAPTATIVOS.(ALGORITMO C)

ZONA TEST: Collado de Villalba

DESCRIPCIÓN:

Se ejecuta el algoritmo multiproceso. El algoritmo concatena dos procesos con parámetros umbrales diferentes de forma consecutiva. Los parámetros utilizados por el primer proceso están definidos para detectar puntos de cota anómala muy diferente al terreno, tales como errores asociados a ruido del sensor o pulsos reflejados por altas estructuras u objetos (grúas, líneas eléctricas, aves, ...). Los parámetros umbrales del segundo proceso se ajustan para detectar puntos de cota anómala próximos al terreno, como pulsos reflejados por fachadas de edificos, personas, etc.

ANÁLISIS DE RESULTADOS:

ALGORITMO: C 1

PARÁMETROS UMBRALES:

PROCESO 1	PROCESO 2
UMBRAL EN IZ: 10 m	**UMBRAL EN IZ:** 3 m
UMBRAL EN %: 80 %	**UMBRAL EN %:** 90 %
UMBRAL EN TAMAÑO DE REGIONES: 15 m^2	**UMBRAL EN TAMAÑO DE REGIONES:** 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	**Categorías deducidas de la clasificación**				
	Outliers	**No outliers**	**Total**	**Ex. Prod.**	**Error. Om.**
Outliers	**7185**	281	7466	96,24 %	3,76 %
No outliers	17314	**2807415**	2824729	99,39 %	0,61 %
Total	24499	2807696	**2832195**	Exactitud global: 99,38 %	
Ex. Usuario	29,33 %	99,99 %			
Error. Com	70,67 %	0,01 %			

ALGORITMO: C 2

PARÁMETROS UMBRALES:

PROCESO 1	PROCESO 2
UMBRAL EN IZ: 10 m	**UMBRAL EN IZ:** 3 m
UMBRAL EN %: 80 %	**UMBRAL EN %:** 95 %
UMBRAL EN TAMAÑO DE REGIONES: 15 m^2	**UMBRAL EN TAMAÑO DE REGIONES:** 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	**Categorías deducidas de la clasificación**				
	Outliers	**No outliers**	**Total**	**Ex. Prod.**	**Error. Om.**
Outliers	**7020**	446	7466	94,03 %	5,97 %
No outliers	998	**2823731**	2824729	99,96 %	0,04 %
Total	8018	2824177	**2832195**	Exactitud global: 99,95 %	
Ex. Usuario	87,55 %	99,98 %			
Error. Com	12,45 %	0,02 %			

ALGORITMO: C 3

PARÁMETROS UMBRALES:

PROCESO 1	PROCESO 2
UMBRAL EN IZ: 10 m	**UMBRAL EN IZ:** 3 m
UMBRAL EN %: 80 %	**UMBRAL EN %:** 99 %
UMBRAL EN TAMAÑO DE REGIONES: 15 m^2	**UMBRAL EN TAMAÑO DE REGIONES:** 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	**Categorías deducidas de la clasificación**				
	Outliers	**No outliers**	**Total**	**Ex. Prod.**	**Error. Om.**
Outliers	**3341**	4125	7466	44,75 %	55,25 %
No outliers	789	**2823940**	2824729	99,97 %	0,03 %
Total	4130	2828065	**2832195**	Exactitud global: 99,83 %	
Ex. Usuario	80,90 %	99,85 %			
Error. Com	19,10 %	0,15 %			

ALGORITMO: C 4

PARÁMETROS UMBRALES:

PROCESO 1	PROCESO 2
UMBRAL EN IZ: 10 m	**UMBRAL EN IZ:** 3 m
UMBRAL EN %: 85 %	**UMBRAL EN %:** 90 %
UMBRAL EN TAMAÑO DE REGIONES: 15 m^2	**UMBRAL EN TAMAÑO DE REGIONES:** 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	**Categorías deducidas de la clasificación**				
	Outliers	**No outliers**	**Total**	**Ex. Prod.**	**Error. Om.**
Outliers	**7090**	376	7466	94,96 %	5,04 %
No outliers	17053	**2807676**	2824729	99,40 %	0,60 %
Total	24143	2808052	**2832195**	Exactitud global: 99,38 %	
Ex. Usuario	29,37 %	99,99 %			
Error. Com	70,63 %	0,01 %			

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

ALGORITMO: C 5

PARÁMETROS UMBRALES:

PROCESO 1	PROCESO 2
UMBRAL EN IZ: 10 m	**UMBRAL EN IZ:** 3 m
UMBRAL EN %: 85 %	**UMBRAL EN %:** 95 %
UMBRAL EN TAMAÑO DE REGIONES: 15 m^2	**UMBRAL EN TAMAÑO DE REGIONES:** 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	**Categorías deducidas de la clasificación**				
	Outliers	**No outliers**	**Total**	**Ex. Prod.**	**Error. Om.**
Outliers	**6914**	552	7466	92,61 %	7,39 %
No outliers	375	**2824354**	2824729	99,99 %	0,01 %
Total	7289	2824906	**2832195**	Exactitud global: 99,97 %	
Ex. Usuario	94,86 %	99,98 %			
Error. Com	5,14 %	0,02 %			

ALGORITMO: C 6

PARÁMETROS UMBRALES:

PROCESO 1	PROCESO 2
UMBRAL EN IZ: 10 m	**UMBRAL EN IZ:** 3 m
UMBRAL EN %: 85 %	**UMBRAL EN %:** 99 %
UMBRAL EN TAMAÑO DE REGIONES: 15 m^2	**UMBRAL EN TAMAÑO DE REGIONES:** 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	**Categorías deducidas de la clasificación**				
	Outliers	**No outliers**	**Total**	**Ex. Prod.**	**Error. Om.**
Outliers	**3108**	4358	7466	41,63 %	58,37 %
No outliers	225	**2824504**	2824729	99,99 %	0,01 %
Total	3333	2828862	**2832195**	Exactitud global: 99,84 %	
Ex. Usuario	93,25 %	99,85 %			
Error. Com	6,75 %	0,15 %			

ALGORITMO: C 7

PARÁMETROS UMBRALES:

PROCESO 1	PROCESO 2
UMBRAL EN IZ: 10 m	**UMBRAL EN IZ:** 3 m
UMBRAL EN %: 90 %	**UMBRAL EN %:** 90 %
UMBRAL EN TAMAÑO DE REGIONES: 15 m^2	**UMBRAL EN TAMAÑO DE REGIONES:** 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	**Categorías deducidas de la clasificación**				
	Outliers	**No outliers**	**Total**	**Ex. Prod.**	**Error. Om.**
Outliers	**6906**	560	7466	92,50 %	7,50 %
No outliers	17040	**2807689**	2824729	99,40 %	0,60 %
Total	23946	2808249	**2832195**	Exactitud global: 99,38 %	
Ex. Usuario	28,84 %	99,98 %			
Error. Com	71,16 %	0,02 %			

ALGORITMO: C 8

PARÁMETROS UMBRALES:

PROCESO 1	PROCESO 2
UMBRAL EN IZ: 10 m	**UMBRAL EN IZ:** 3 m
UMBRAL EN %: 90 %	**UMBRAL EN %:** 95 %
UMBRAL EN TAMAÑO DE REGIONES: 15 m^2	**UMBRAL EN TAMAÑO DE REGIONES:** 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	**Categorías deducidas de la clasificación**				
	Outliers	**No outliers**	**Total**	**Ex. Prod.**	**Error. Om.**
Outliers	**6298**	1168	7466	84,36 %	15,64 %
No outliers	280	**2824449**	2824729	99,99 %	0,01 %
Total	6578	2825617	**2832195**	Exactitud global: 99,95 %	
Ex. Usuario	95,74 %	99,96 %			
Error. Com	4,26 %	0,04 %			

ALGORITMO: C 9

PARÁMETROS UMBRALES:

PROCESO 1	PROCESO 2
UMBRAL EN IZ: 10 m	**UMBRAL EN IZ:** 3 m
UMBRAL EN %: 90 %	**UMBRAL EN %:** 99 %
UMBRAL EN TAMAÑO DE REGIONES: 15 m^2	**UMBRAL EN TAMAÑO DE REGIONES:** 15 m^2

RADIO DE VECINDARIO CILÍNDRICO: 5 m

MATRIZ DE CONFUSIÓN					
Clases de referencia	**Categorías deducidas de la clasificación**				
	Outliers	**No outliers**	**Total**	**Ex. Prod.**	**Error. Om.**
Outliers	**2376**	5090	7466	31,82 %	68,18 %
No outliers	62	**2824667**	2824729	100,00 %	0,00 %
Total	2438	2829757	**2832195**	Exactitud global: 99,82 %	
Ex. Usuario	97,46 %	99,82 %			
Error. Com	2,54 %	0,18%			

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

ALGORITMO: DETECCIÓN DE ERRORES GROSEROS MEDIANTE ANÁLISIS DE VARIACIONES LOCALES DE DESNIVEL. UN ÚNICO PROCESO (ALGORITMO A)

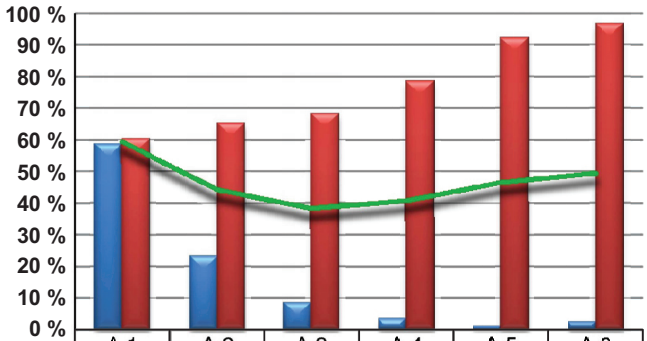
ZONA TEST: Collado de Villalba

DESCRIPCIÓN:

Se ejecuta el algoritmo simple. Los parámetros utilizados están definidos para detectar puntos de cota anómala muy diferente al terreno, tales como errores asociados a ruido del sensor o pulsos reflejados por altas estructuras u objetos (grúas, líneas eléctricas, aves, ...).

GRAFICO 1:

Muestra los errores de comisión y omisión obtenidos tras ejecutar el algoritmo. El algoritmo se ejectuta 6 veces utilizando distinto parámetro umbral en porcentaje. Además se muestra mediante una línea la media del error de comisión y omisión para cada uno de los resultados obtenidos

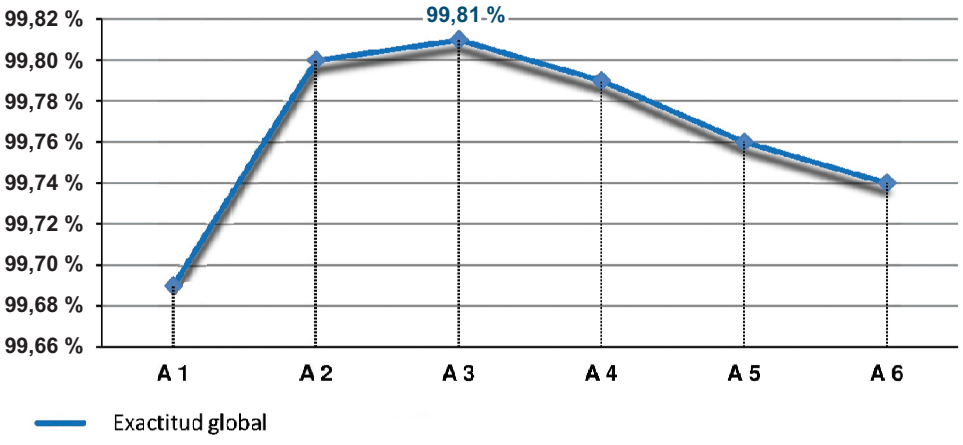


	A 1	A 2	A 3	A 4	A 5	A 6
Error de comisión (%)	58,53	23,2	8,53	3,42	1,02	2,43
Error de omisión (%)	60,34	65,32	68,4	78,42	92,18	96,77
Media de errores (%)	59,44	44,26	38,47	40,92	46,60	49,60

Media de errores: Es la media aritmética entre el error de omisión y el error de comisión.

GRAFICO 2:

Muestra la exactitud global de la clasificación para cada uno de los seis resultados obtenidos.



ANÁLISIS DE RESULTADOS (ANÁLISIS GRÁFICO)

ALGORITMO: DETECCIÓN DE ERRORES GROSEROS MEDIANTE ANÁLISIS DE VARIACIONES LOCALES DE DESNIVEL. UN ÚNICO PROCESO (ALGORITMO B)

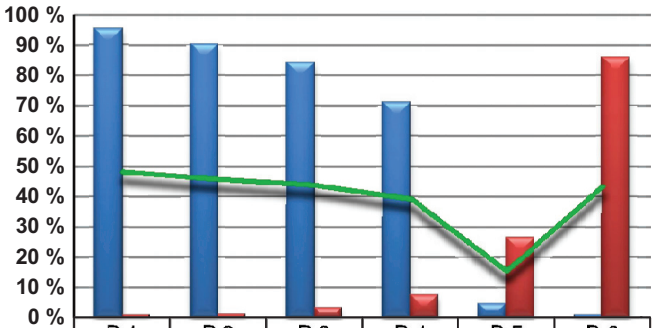
ZONA TEST: Collado de Villalba

DESCRIPCIÓN:

Se ejecuta el algoritmo simple. Los parámetros utilizados están definidos para detectar puntos de cota anómala próximos al terreno, tales como errores asociados a ruido del sensor o pulsos reflejados en pequeñas estructuras u objetos de baja altura (pulsos reflejados sobre fachadas, personas, ...).

GRAFICO 1:

Muestra los errores de comisión y omisión obtenidos tras ejecutar el algoritmo. El algoritmo se ejectuta 6 veces utilizando distinto parámetro umbral en porcentaje. Además se muestra mediante una línea la media del error de comisión y omisión para cada uno de los resultados obtenidos

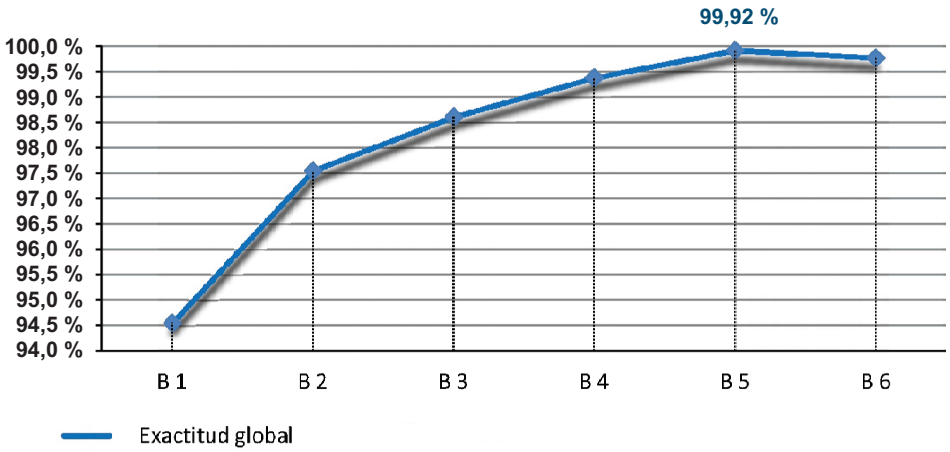


	B 1	B 2	B 3	B 4	B 5	B 6
Error de comisión (%)	95,43	90,37	84,38	71,16	4,77	1,04
Error de omisión (%)	1	1,23	3,31	7,49	26,52	85,95
Media de errores (%)	48,215	45,8	43,845	39,325	15,645	43,495

Media de errores: Es la media aritmética entre el error de omisión y el error de comisión.

GRAFICO 2:

Muestra la exactitud global de la clasificación para cada uno de los seis resultados obtenidos.



ANÁLISIS DE RESULTADOS (ANÁLISIS GRÁFICO)

ALGORITMO: DETECCIÓN DE ERRORES GROSEROS MEDIANTE ANÁLISIS DE VARIACIONES LOCALES DE DESNIVEL. ALGORITMO MULTIPROCESO CON PARÁMETROS ADAPTATIVOS (ALGORITMO C)

ZONA TEST: Collado de Villalba

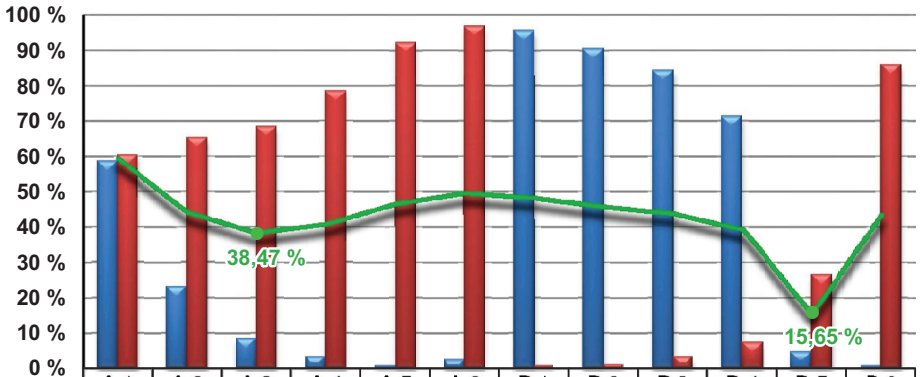
DESCRIPCIÓN:

Se ejecuta el algoritmo multiproceso. El algoritmo concatena dos procesos con parámetros umbrales diferentes de forma consecutiva. Los parámetros utilizados por el primer proceso están definidos para detectar puntos de cota anómala muy diferente al terreno, tales como errores asociados a ruido del sensor o pulsos reflejados por altas estructuras u objetos (grúas, líneas eléctricas, aves, ...). Los parámetros umbrales del segundo proceso se ajustan para detectar puntos de cota anómala próximos al terreno, como pulsos reflejados por fachadas de edificos, personas, etc.

Previamente a la selección de los parámetros umbrales aosciados a ambos procesos se analizan los parámetros umbrales que mejores resultados han obtenido tanto para para detectar puntos de cota anómala muy diferente al terreno (algoritmo A) así como para detectar puntos de cota anómal próximos al terreno (algoritmo B)

GRAFICO 1:

Muestra los errores de comisión y omisión obtenidos tras ejecutar los algoritmos simples (Algoritmo A y algoritmo B), así como la media del error de omisión y comisión. Cada uno de los algoritmos se ha ejecutado seis veces con distintos parámetros umbrales.



	A 1	A 2	A 3	A 4	A 5	A 6	B 1	B 2	B 3	B 4	B 5	B 6
Error de comisión (%)	58,53	23,2	8,53	3,42	1,02	2,43	95,43	90,37	84,38	71,16	4,77	1,04
Error de omisión (%)	60,34	65,32	68,4	78,42	92,18	96,77	1,00	1,23	3,31	7,49	26,52	85,95
Media de errores (%)	59,44	44,26	38,47	40,92	46,60	49,60	48,22	45,80	43,35	39,33	15,65	43,50

GRAFICO 2:

Muestra la exactitud global de la clasificación para cada uno de los seis resultados obtenidos asociados a los algoritmos ejecutados (algoitmo A y algoritmo B).

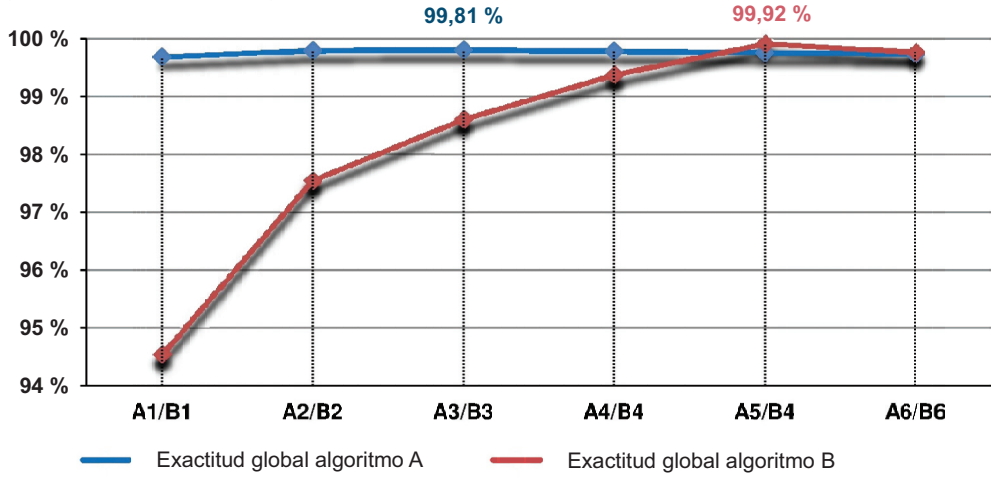
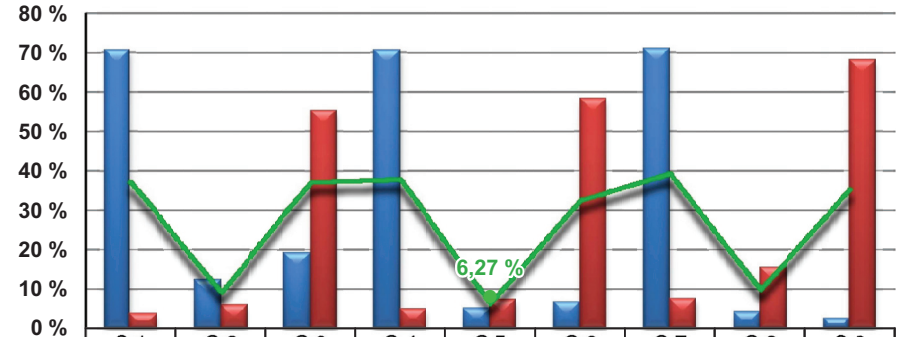


GRAFICO 3:

Muestra los errores de comisión y omisión obtenidos tras ejecutar los algoritmos compuesto (Algoritmo C), así como la media del error de omisión y comisión. El algoritmo se ejectua nueve veces con distintos parámetros umbrales.



	C 1	C 2	C 3	C 4	C 5	C 6	C 7	C 8	C 9
Error de comisión (%)	70,67	12,45	19,1	70,63	5,14	6,75	71,16	4,26	2,54
Error de omisión (%)	3,76	5,97	55,25	5,04	7,39	58,37	7,5	15,64	68,18
Media de errores (%)	37,215	9,21	37,175	37,835	6,265	32,56	39,33	9,95	35,36

GRAFICO 4:

Muestra la exactitud global de la clasificación para cada uno de los nueve resultados obtenidos asociados tras ejecutar el algoritmo (algoritmo C).

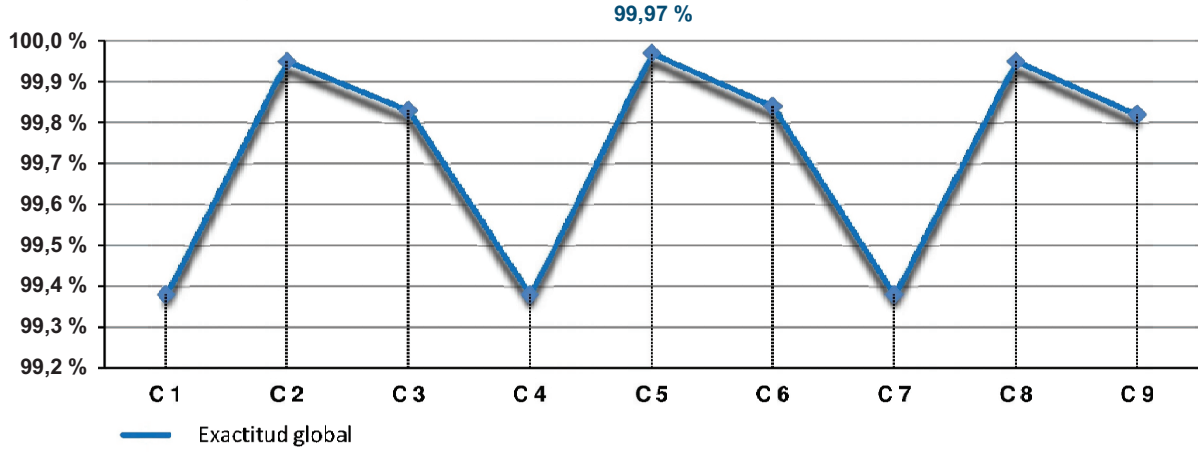
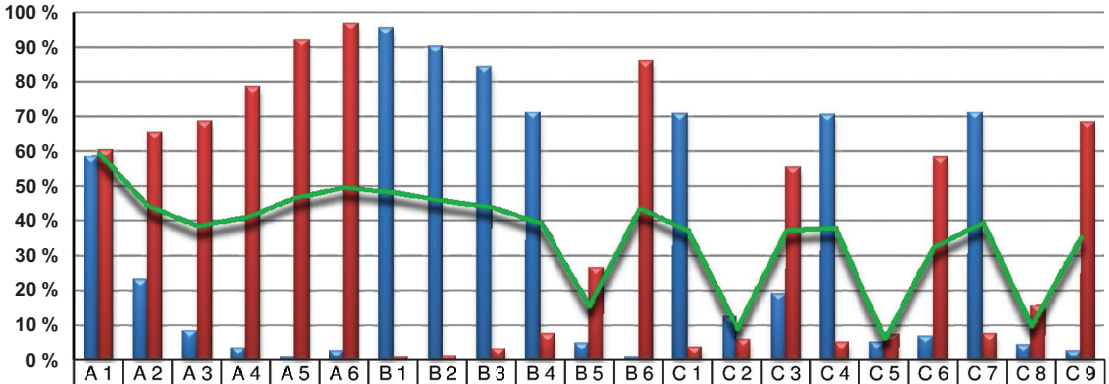


GRAFICO 5:

Muestra los errores de comisión y omisión obtenidos tras ejecutar los algoritmos simples (algoritmo A y B) así como el algoritmo compuesto (algoritmo C). Además se muestra la media de los errores.

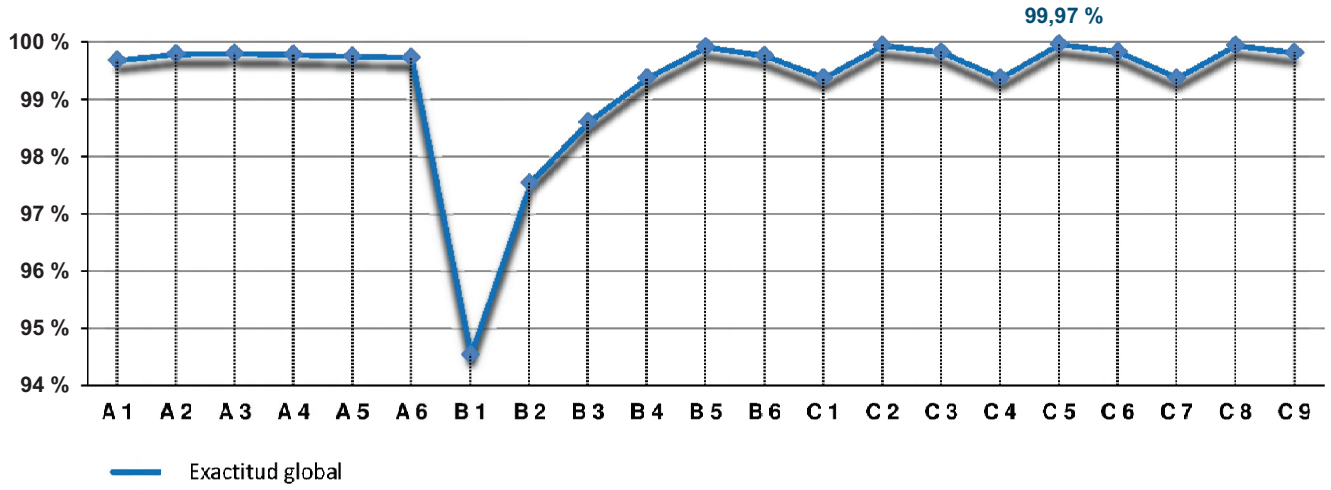


	A 1	A 2	A 3	A 4	A 5	A 6	B 1	B 2	B 3	B 4	B 5	B 6	C 1	C 2	C 3	C 4	C 5	C 6	C 7	C 8	C 9
Error de comisión (%)	58,53	23,2	8,53	3,42	1,02	2,43	95,43	90,37	84,38	71,16	4,77	1,04	70,67	12,45	19,10	70,63	5,14	6,75	71,16	4,26	2,54
Error de omisión (%)	60,34	65,32	68,4	78,42	92,18	96,77	1,00	1,23	3,31	7,49	26,52	85,95	3,76	5,97	55,25	5,04	7,39	58,37	7,50	15,64	68,18
Media de errores (%)	59,44	44,26	38,47	40,92	46,60	49,60	48,22	45,80	43,85	39,33	15,65	43,50	37,22	9,21	37,18	37,84	6,27	32,56	39,33	9,95	35,36

Media de errores: Es la media aritmética entre el error de omisión y el error de comisión.

GRAFICO 6:

Muestra la exactitud global de la clasificación para cada uno de los nueve resultados obtenidos asociados tras ejecutar los algoritmos simples (algoritmo A y B) así como el algoritmo compuesto (algoritmo C).



DESCRIPCIÓN DE PARÁMETROS UMBRALES UTILIZADOS

ALGORITMO: DETECCIÓN DEL TERRENO MEDIANTE METODO DE DENSIFICACIÓN QUE COMBINA INTERPOLACIÓN Y SEGMENTACIÓN SIN ANALIZAR CARACTERÍSTICAS DE LAS AGRUPACIONES (ALGORITMO A)

DESCRIPCIÓN:

Algoritmo de densificación progresiva para detectar la clase terreno. La superficie de referencia se obtiene mediante interpolación de distantancia inversa ponderada. El algoritmo de densificación se combina con un proceso de segmentación. Las agrupaciones obtenidas tras la segmentación son clasificadas como terreno en un proceso iterativo en función de su distancia a la superficie de referencia. No se realiza análisis alguno de las características de agrupaciones colindantes para determinar la clase terreno.

PARÁMETROS UMBRALES FIJOS PARA ALGORITMO A:

UMBRAL EN IZ SEGMENTACIÓN: 0,25 m **TAMAÑO DE CELDA:** 1 m **EXPONENTE DE PONDERACIÓN EN INTERPOLACIÓN:** 3

PARÁMETROS UMBRALES VARIABLES PARA ALGORITMO A:

	U_IZ	L_Bloque
A 1-1	0,2 m	100 m
A 1-2	0,2 m	50 m
A 1-3	0,2 m	25 m
A 1-4	0,2 m	10 m
A 2-1	0,4 m	100 m
A 2-2	0,4 m	50 m
A 2-3	0,4 m	25 m
A 2-4	0,4 m	10 m

	U_IZ	L_Bloque
A 3-1	0,6 m	100 m
A 3-2	0,6 m	50 m
A 3-3	0,6 m	25 m
A 3-4	0,6 m	10 m
A 4-1	1 m	100 m
A 4-2	1 m	50 m
A 4-3	1 m	25 m
A 4-4	1 m	10 m

	U_IZ	L_Bloque
A 5-1	2 m	100 m
A 5-2	2 m	50 m
A 5-3	2 m	25 m
A 5-4	2 m	10 m

U_IZ: Umbral en desnivel

L_Bloque: Longitud del lado del bloque (m)

ALGORITMO: DETECCIÓN DEL TERRENO MEDIANTE METODO DE DENSIFICACIÓN QUE COMBINA INTERPOLACIÓN Y SEGMENTACIÓN Y ANALIZA CARACTERÍSTICAS DE LAS AGRUPACIONES (ALGORITMO B)

ZONA TEST: Collado de Villalba

DESCRIPCIÓN:

Algoritmo de densificación progresiva para detectar la clase terreno. La superficie de referencia se obtiene mediante interpolación de distantancia inversa ponderada. El algoritmo de densificación se combina con un proceso de segmentación. Las agrupaciones obtenidas tras la segmentación son clasificadas como terreno en un proceso iterativo en función de su distancia a la superficie de referencia. El proceso lleva asociado un análisis de las características de agrupaciones colindantes con el fin de reducir al máximo errores en la clasificación y conseguir además una mayor automatización del proceso.

PARÁMETROS UMBRALES FIJOS PARA ALGORITMO B:

UMBRAL EN IZ SEGMENTACIÓN: 0,25 m **TAMAÑO DE CELDA:** 1 m **EXPONENTE DE PONDERACIÓN EN INTERPOLACIÓN:** 3

PARÁMETROS UMBRALES VARIABLES PARA ALGORITMO B:

	U_IZ	L_Bloque
B 1-1	0,2 m	100 m
B 1-2	0,2 m	50 m
B 1-3	0,2 m	25 m
B 1-4	0,2 m	10 m
B 2-1	0,4 m	100 m
B 2-2	0,4 m	50 m
B 2-3	0,4 m	25 m
B 2-4	0,4 m	10 m

	U_IZ	L_Bloque
B 3-1	0,6 m	100 m
B 3-2	0,6 m	50 m
B 3-3	0,6 m	25 m
B 3-4	0,6 m	10 m
B 4-1	1 m	100 m
B 4-2	1 m	50 m
B 4-3	1 m	25 m
B 4-4	1 m	10 m

	U_IZ	L_Bloque
B 5-1	2 m	100 m
B 5-2	2 m	50 m
B 5-3	2 m	25 m
B 5-4	2 m	10 m

U_IZ: Umbral en desnivel

L_Bloque: Longitud del lado del bloque (m)

ALGORITMO: DETECCIÓN DEL TERRENO MEDIANTE METODO DE DENSIFICACIÓN QUE COMBINA INTERPOLACIÓN Y SEGMENTACIÓN SIN ANALIZAR CARACTERÍSTICAS DE LAS AGRUPACIONES (ALGORITMO A)

ZONA TEST: Vall d'Uixó

DESCRIPCIÓN:

Algoritmo de densificación progresiva para detectar la clase terreno. La superficie de referencia se obtiene mediante interpolación de distantancia inversa ponderada. El algoritmo de densificación se combina con un proceso de segmentación. Las agrupaciones obtenidas tras la segmentación son clasificadas como terreno en un proceso iterativo en función de su distancia a la superficie de referencia. No se realiza análisis alguno de las características de agrupaciones colindantes para determinar la clase terreno.

PARÁMETROS UMBRALES FIJOS PARA ALGORITMO A:

UMBRAL EN IZ SEGMENTACIÓN: 0,25 m **TAMAÑO DE CELDA:** 1 m **EXPONENTE DE PONDERACIÓN EN INTERPOLACIÓN:** 3

ALGORITMO: A 1 - 1

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 0,2 m **TAMAÑO DE CUADRANTES:** 100 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**94895**	474	95369	99,50 %	0,50 %
No terreno	3194	**208733**	211927	98,49 %	1,51 %
Total	98089	209207	**307296**	Exactitud global: 98,81 %	
Ex. Usuario	96,74 %	99,77 %			
Error. Com	3,26 %	0,23 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Total	
Terreno	2243	951	3194	N puntos
	70,23%	29,77%	100 %	%

ALGORITMO: A 1 - 2

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 0,2 m **TAMAÑO DE CUADRANTES:** 50 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**94891**	478	95369	99,50 %	0,50 %
No terreno	3030	**208897**	211927	98,57 %	1,43 %
Total	97921	209375	**307296**	Exactitud global: 98,86 %	
Ex. Usuario	96,91 %	99,77 %			
Error. Com	3,09 %	0,23 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Total	
Terreno	2143	887	3030	N puntos
	70,73 %	29,27 %	100 %	%

ALGORITMO: A 1 - 3

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 0,2 m **TAMAÑO DE CUADRANTES:** 25 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**94806**	563	95369	99,41 %	0,59 %
No terreno	11021	**200906**	211927	94,80 %	5,20 %
Total	105827	201469	**307296**	Exactitud global: 96,23 %	
Ex. Usuario	89,59 %	99,72 %			
Error. Com	10,41 %	0,28 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Total	
Terreno	10165	856	11021	N puntos
	92,23 %	7,77 %	100 %	%

Categorías d.c : Categorías deducidas de la clasificación

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

ALGORITMO: A 1 - 4

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 0,2 m **TAMAÑO DE CUADRANTES:** 10 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**95039**	330	95369	99,65 %	0,35 %
No terreno	63282	**148645**	211927	70,14 %	29,86 %
Total	158321	148975	**307296**	Exactitud global: 79,30 %	
Ex. Usuario	60,03 %	99,78 %			
Error. Com	39,97 %	0,22 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Total	
Terreno	61598	1684	63282	N puntos
	97,34 %	2,66 %	100 %	%

ALGORITMO: A 2 - 1

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 0,4 m **TAMAÑO DE CUADRANTES:** 100 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**95275**	94	95369	99,90 %	0,10 %
No terreno	3989	**207938**	211927	98,12 %	1,88 %
Total	99264	208032	**307296**	Exactitud global: 98,67 %	
Ex. Usuario	95,98 %	99,95 %			
Error. Com	4,02 %	0,05 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Total	
Terreno	2696	1293	3989	N puntos
	67,59 %	32,41 %	100 %	%

ALGORITMO: A 2 - 2

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 0,4 m **TAMAÑO DE CUADRANTES:** 50 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**95273**	96	95369	99,90 %	0,10 %
No terreno	3932	**207995**	211927	98,14 %	1,86 %
Total	99205	208091	**307296**	Exactitud global: 98,69 %	
Ex. Usuario	96,04 %	99,95 %			
Error. Com	3,96 %	0,05 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Total	
Terreno	2641	1291	3932	N puntos
	67,17 %	32,83 %	100 %	%

ALGORITMO: A 2 - 3

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 0,4 m **TAMAÑO DE CUADRANTES:** 25 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**95169**	200	95369	99,79 %	0,21 %
No terreno	11949	**199978**	211927	94,36 %	5,64 %
Total	107118	200178	**307296**	Exactitud global: 96,05 %	
Ex. Usuario	88,85 %	99,90 %			
Error. Com	11,15 %	0,10 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Total	
Terreno	10749	1200	11949	N puntos
	89,96 %	10,04 %	100 %	%

Categorías d.c : Categorías deducidas de la clasificación

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

ALGORITMO: A 2 - 4

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 0,4 m **TAMAÑO DE CUADRANTES:** 10 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**95200**	169	95369	99,82 %	0,18 %
No terreno	64476	**147451**	211927	69,58 %	30,42 %
Total	159676	147620	**307296**	Exactitud global: 78,96 %	
Ex. Usuario	59,62 %	99,89 %			
Error. Com	40,38 %	0,11 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Total	
Terreno	62350	2126	64476	N puntos
	96,70 %	3,30 %	100 %	%

ALGORITMO: A 3 - 1

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 0,6 m **TAMAÑO DE CUADRANTES:** 100 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**95291**	78	95369	99,92 %	0,08 %
No terreno	4977	**206950**	211927	97,65 %	2,35 %
Total	100268	207028	**307296**	Exactitud global: 98,36 %	
Ex. Usuario	95,04 %	99,96 %			
Error. Com	4,96 %	0,04 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Total	
Terreno	3217	1760	4977	N puntos
	64,64 %	35,36 %	100 %	%

ALGORITMO: A 3 - 2

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 0,6 m **TAMAÑO DE CUADRANTES:** 50 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**95291**	78	95369	99,92 %	0,08 %
No terreno	4977	**206950**	211927	97,65 %	2,35 %
Total	100268	207028	**307296**	Exactitud global: 98,36 %	
Ex. Usuario	95,04 %	99,96 %			
Error. Com	4,96 %	0,04 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Total	
Terreno	3217	1760	4977	N puntos
	64,64 %	35,36 %	100 %	%

ALGORITMO: A 3 - 3

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 0,4 m **TAMAÑO DE CUADRANTES:** 25 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**95264**	105	95369	99,89 %	0,11 %
No terreno	13122	**198805**	211927	93,81 %	6,19 %
Total	108386	198910	**307296**	Exactitud global: 95,70 %	
Ex. Usuario	87,89 %	99,95 %			
Error. Com	12,11 %	0,05 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Total	
Terreno	11413	1709	13122	N puntos
	86,98 %	13,02 %	100 %	%

Categorías d.c : Categorías deducidas de la clasificación

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

ALGORITMO: A 3 - 4

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 0,6 m **TAMAÑO DE CUADRANTES:** 10 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**95223**	146	95369	99,85 %	0,15 %
No terreno	65681	**146246**	211927	69,01 %	30,99 %
Total	160904	146392	**307296**	Exactitud global: 78,58 %	
Ex. Usuario	59,18 %	99,90 %			
Error. Com	40,82 %	0,10 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Total	
Terreno	63051	2630	65681	N puntos
	96,00 %	4,00 %	100 %	%

ALGORITMO: A 4 - 1

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 1 m **TAMAÑO DE CUADRANTES:** 100 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**95302**	67	95369	99,93 %	0,07 %
No terreno	9771	**202156**	211927	95,39 %	4,61 %
Total	105073	202223	**307296**	Exactitud global: 96,80 %	
Ex. Usuario	90,70 %	99,97 %			
Error. Com	9,30 %	0,03 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Total	
Terreno	6469	3302	9771	N puntos
	66,21 %	33,79 %	100 %	%

ALGORITMO: A 4 - 2

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 1 m **TAMAÑO DE CUADRANTES:** 50 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**95302**	67	95369	99,93 %	0,07 %
No terreno	9715	**202212**	211927	95,42 %	4,58 %
Total	105017	202279	**307296**	Exactitud global: 96,82 %	
Ex. Usuario	90,75 %	99,97 %			
Error. Com	9,25 %	0,03 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Total	
Terreno	6414	3301	9715	N puntos
	66,02 %	33,98 %	100 %	%

ALGORITMO: A 4 - 3

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 1 m **TAMAÑO DE CUADRANTES:** 25 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**95285**	84	95369	99,91 %	0,09 %
No terreno	17831	**194096**	211927	91,59 %	8,41 %
Total	113116	194180	**307296**	Exactitud global: 94,17 %	
Ex. Usuario	84,24 %	99,96 %			
Error. Com	15,76 %	0,04 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Total	
Terreno	14693	3138	17831	N puntos
	82,40 %	17,60 %	100 %	%

Categorías d.c : Categorías deducidas de la clasificación

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

ALGORITMO: A 4 - 4

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 1 m **TAMAÑO DE CUADRANTES:** 10 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**95237**	132	95369	99,86 %	0,14 %
No terreno	70379	**141548**	211927	66,79 %	33,21 %
Total	165616	141680	**307296**	Exactitud global: 77,05 %	
Ex. Usuario	57,50 %	99,91 %			
Error. Com	42,50 %	0,09 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Total	
Terreno	66531	3848	70379	N puntos
	94,53 %	5,47 %	100 %	%

ALGORITMO: A 5 - 1

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 2 m **TAMAÑO DE CUADRANTES:** 100 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**95330**	39	95369	99,96 %	0,04 %
No terreno	64345	**147582**	211927	69,64 %	30,36 %
Total	159675	147621	**307296**	Exactitud global: 79,05 %	
Ex. Usuario	59,70 %	99,97 %			
Error. Com	40,30 %	0,03 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Total	
Terreno	58193	6152	64345	N puntos
	90,44 %	9,56 %	100 %	%

ALGORITMO: A 5 - 2

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 2 m **TAMAÑO DE CUADRANTES:** 50 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**95330**	39	95369	99,96 %	0,04 %
No terreno	62151	**149776**	211927	70,67 %	29,33 %
Total	157481	149815	**307296**	Exactitud global: 79,76 %	
Ex. Usuario	60,53 %	99,97 %			
Error. Com	39,47 %	0,03 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Categorías deducidas de la clasificación			
	Edificio	Vegetación	Total	
Terreno	56005	6146	62151	N puntos
	90,11 %	9,89 %	100 %	%

ALGORITMO: A 5 - 3

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 2 m **TAMAÑO DE CUADRANTES:** 25 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**95312**	57	95369	99,94 %	0,06 %
No terreno	64293	**147634**	211927	69,66 %	30,34 %
Total	159605	147691	**307296**	Exactitud global: 79,06 %	
Ex. Usuario	59,72 %	99,96 %			
Error. Com	40,28 %	0,04 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Total	
Terreno	58296	5997	64293	N puntos
	90,67 %	9,33 %	100 %	%

Categorías d.c : Categorías deducidas de la clasificación

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

ALGORITMO: A 5 - 4

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 1 m **TAMAÑO DE CUADRANTES:** 10 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	**No terreno**	**Total**	**Ex. Prod.**	**Error. Om.**
Terreno	**95252**	117	95369	99,88 %	0,12 %
No terreno	95745	**116182**	211927	54,82 %	45,18 %
Total	190997	116299	**307296**	Exactitud global: 68,80 %	
Ex. Usuario	49,87 %	99,90 %			
Error. Com	50,13 %	0,10 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	**Vegetación**	**Total**	
Terreno	89847	5898	95745	**N puntos**
	93,84 %	6,16 %	100 %	**%**

Categorías d.c : Categorías deducidas de la clasificación

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

ALGORITMO: DETECCIÓN DEL TERRENO MEDIANTE METODO DE DENSIFICACIÓN QUE COMBINA INTERPOLACIÓN Y SEGMENTACIÓN Y ANALIZA CARACTERÍSTICAS DE LAS AGRUPACIONES (ALGORITMO B)

ZONA TEST: Vall d'Uixó

DESCRIPCIÓN:

Algoritmo de densificación progresiva para detectar la clase terreno. La superficie de referencia se obtiene mediante interpolación de distantancia inversa ponderada. El algoritmo de densificación se combina con un proceso de segmentación. Las agrupaciones obtenidas tras la segmentación son clasificadas como terreno en un proceso iterativo en función de su distancia a la superficie de referencia. El proceso lleva asociado un análisis de las características de agrupaciones colindantes con el fin de reducir al máximo errores en la clasificación y conseguir además una mayor automatización del proceso.

PARÁMETROS UMBRALES FIJOS PARA ALGORITMO B:

UMBRAL EN IZ SEGMENTACIÓN: 0,25 m **TAMAÑO DE CELDA:** 1 m **EXPONENTE DE PONDERACIÓN EN INTERPOLACIÓN:** 3

ALGORITMO: B 1 - 1, B 1 - 2, B 1 - 3, B 1 - 4

PARÁMETROS UMBRALES VARIABLES (B 1-1): **UMBRAL EN IZ:** 0,2 m **TAMAÑO DE CUADRANTES:** 100 m

PARÁMETROS UMBRALES VARIABLES (B 1-2): **UMBRAL EN IZ:** 0,2 m **TAMAÑO DE CUADRANTES:** 50 m

PARÁMETROS UMBRALES VARIABLES (B 1-3): **UMBRAL EN IZ:** 0,2 m **TAMAÑO DE CUADRANTES:** 25 m

PARÁMETROS UMBRALES VARIABLES (B 1-4): **UMBRAL EN IZ:** 0,2 m **TAMAÑO DE CUADRANTES:** 10 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**94698**	671	95369	99,30 %	0,70 %
No terreno	9	**211918**	211927	100 %	0,00 %
Total	94707	212589	**307296**	Exactitud global: 99,78 %	
Ex. Usuario	99,99 %	99,68 %			
Error. Com	0,01 %	0,32 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Total	
Terreno	5	4	9	N puntos
	55,56 %	44,44 %	100 %	%

ALGORITMO: B 2 - 1, B 2 - 2, B 2 - 3, B 2 - 4

PARÁMETROS UMBRALES VARIABLES (B 2-1): **UMBRAL EN IZ:** 0,4 m **TAMAÑO DE CUADRANTES:** 100 m

PARÁMETROS UMBRALES VARIABLES (B 2-2): **UMBRAL EN IZ:** 0,4 m **TAMAÑO DE CUADRANTES:** 50 m

PARÁMETROS UMBRALES VARIABLES (B 2-3): **UMBRAL EN IZ:** 0,4 m **TAMAÑO DE CUADRANTES:** 25 m

PARÁMETROS UMBRALES VARIABLES (B 2-4): **UMBRAL EN IZ:** 0,4 m **TAMAÑO DE CUADRANTES:** 10 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**95085**	284	95369	99,70 %	0,30 %
No terreno	32	**211895**	211927	99,98 %	0,02 %
Total	95117	212179	**307296**	Exactitud global: 99,90 %	
Ex. Usuario	99,97 %	99,87 %			
Error. Com	0,03 %	0,13 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Total	
Terreno	11	21	32	N puntos
	34,38 %	65,62 %	100 %	%

Categorías d.c : Categorías deducidas de la clasificación

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

ALGORITMO: B 3 - 1, B 3 - 2, B 3 - 3, B 3 - 4

PARÁMETROS UMBRALES VARIABLES (B 3-1): **UMBRAL EN IZ:** 0,6 m **TAMAÑO DE CUADRANTES:** 100 m

PARÁMETROS UMBRALES VARIABLES (B 3-2): **UMBRAL EN IZ:** 0,6 m **TAMAÑO DE CUADRANTES:** 50 m

PARÁMETROS UMBRALES VARIABLES (B 3-3): **UMBRAL EN IZ:** 0,6 m **TAMAÑO DE CUADRANTES:** 25 m

PARÁMETROS UMBRALES VARIABLES (B 3-4): **UMBRAL EN IZ:** 0,6 m **TAMAÑO DE CUADRANTES:** 10 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**95085**	284	95369	99,70 %	0,30 %
No terreno	486	**211441**	211927	99,77 %	0,23 %
Total	95571	211725	**307296**	Exactitud global: 99,75 %	
Ex. Usuario	99,49 %	99,87 %			
Error. Com	0,51 %	0,13 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Total	
Terreno	248	238	486	N puntos
	51,03 %	48,97 %	100 %	%

ALGORITMO: B 4 - 1, B 4 - 2, B 4 - 3, B 4 - 4

PARÁMETROS UMBRALES VARIABLES (B 4-1): **UMBRAL EN IZ:** 1 m **TAMAÑO DE CUADRANTES:** 100 m

PARÁMETROS UMBRALES VARIABLES (B 4-2): **UMBRAL EN IZ:** 1 m **TAMAÑO DE CUADRANTES:** 50 m

PARÁMETROS UMBRALES VARIABLES (B 4-3): **UMBRAL EN IZ:** 1 m **TAMAÑO DE CUADRANTES:** 25 m

PARÁMETROS UMBRALES VARIABLES (B 4-4): **UMBRAL EN IZ:** 1 m **TAMAÑO DE CUADRANTES:** 10 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**95089**	280	95369	99,71 %	0,29 %
No terreno	1405	**210522**	211927	99,34 %	0,66 %
Total	96494	210802	**307296**	Exactitud global: 99,45 %	
Ex. Usuario	98,54 %	99,87 %			
Error. Com	1,46 %	0,13 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Total	
Terreno	390	1015	1405	N puntos
	27,76 %	72,24 %	100 %	%

ALGORITMO: B 5 - 1, B 5 - 2, B 5 - 3, B 5 - 4

PARÁMETROS UMBRALES VARIABLES (B 5-1): **UMBRAL EN IZ:** 2 m **TAMAÑO DE CUADRANTES:** 100 m

PARÁMETROS UMBRALES VARIABLES (B 5-2): **UMBRAL EN IZ:** 2 m **TAMAÑO DE CUADRANTES:** 50 m

PARÁMETROS UMBRALES VARIABLES (B 5-3): **UMBRAL EN IZ:** 2 m **TAMAÑO DE CUADRANTES:** 25 m

PARÁMETROS UMBRALES VARIABLES (B 5-4): **UMBRAL EN IZ:** 2 m **TAMAÑO DE CUADRANTES:** 10 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**95100**	269	95369	99,72 %	0,28 %
No terreno	8772	**203155**	211927	95,86 %	4,14 %
Total	103872	203424	**307296**	Exactitud global: 97,06 %	
Ex. Usuario	91,55 %	99,87 %			
Error. Com	8,45 %	0,13 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Total	
Terreno	7643	1129	8772	N puntos
	87,13 %	12,87 %	100 %	%

Categorías d.c : Categorías deducidas de la clasificación

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

ALGORITMO: DETECCIÓN DEL TERRENO MEDIANTE METODO DE DENSIFICACIÓN QUE COMBINA INTERPOLACIÓN Y SEGMENTACIÓN SIN ANALIZAR CARACTERÍSTICAS DE LAS AGRUPACIONES (ALGORITMO A)

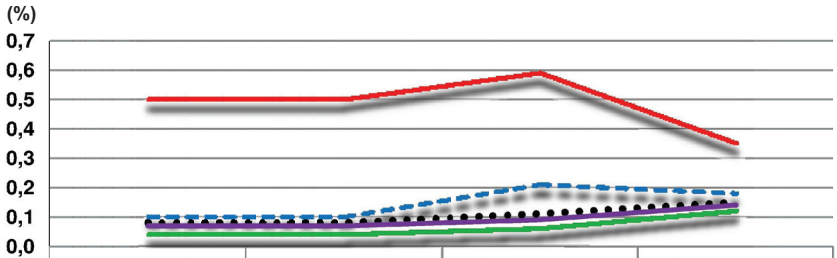
ZONA TEST: Vall d'Uixó

DESCRIPCIÓN:

Algoritmo de densificación progresiva para detectar la clase terreno. La superficie de referencia se obtiene mediante interpolación de distantancia inversa ponderada. El algoritmo de densificación se combina con un proceso de segmentación. Las agrupaciones obtenidas tras la segmentación son clasificadas como terreno en un proceso iterativo en función de su distancia a la superficie de referencia. No se realiza análisis alguno de las características de agrupaciones colindantes para determinar la clase terreno.

GRAFICO 1:

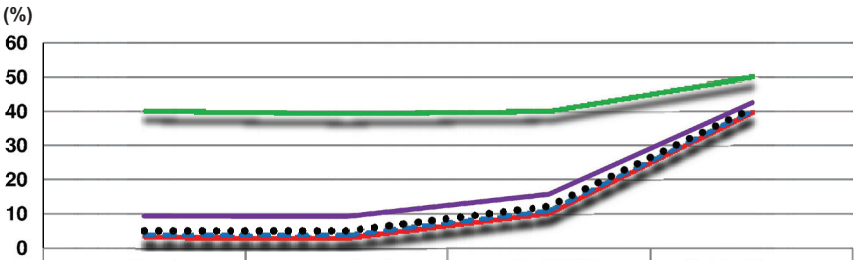
Muestra los errores de omisión obtenidos tras ejecutar el algoritmo A para los cinco umbrales en desnivel utilizados. Además el gráfico muestra la variabilidad de los resultados en función del tamaño de bloque o celda utilizado para generar el terreno provisional inicial.



	Celda: 100x100 m	Celda: 50x50 m	Celda: 25x25 m	Celda: 10x10 m
Error de omisión (%) (Umbral Iz=0,2 m)	0,50	0,50	0,59	0,35
Error de omisión (%) (Umbral Iz=0,4 m)	0,10	0,10	0,21	0,18
Error de omisión (%) (Umbral Iz=0,6 m)	0,08	0,08	0,11	0,15
Error de omisión (%) (Umbral Iz=1 m)	0,07	0,07	0,09	0,14
Error de omisión (%) (Umbral Iz=2 m)	0,04	0,04	0,06	0,12

GRAFICO 2:

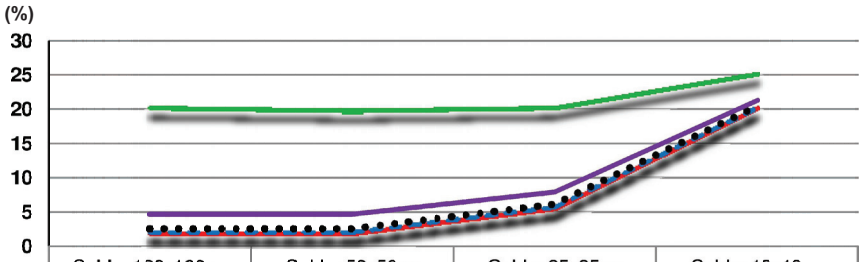
Muestra los errores de comisión obtenidos tras ejecutar el algoritmo A para los cinco umbrales en desnivel utilizados. Además el gráfico muestra la variabilidad de los resultados en función del tamaño de bloque o celda utilizado para generar el terreno provisional inicial.



	Celda: 100x100 m	Celda: 50x50 m	Celda: 25x25 m	Celda: 10x10 m
Error de comisión (%) (Umbral Iz=0,2 m)	3,26	3,09	10,41	39,97
Error de comisión (%) (Umbral Iz=0,4 m)	4,02	3,96	11,15	40,38
Error de comisión (%) (Umbral Iz=0,6 m)	4,96	4,96	12,11	40,82
Error de comisión (%) (Umbral Iz=1 m)	9,30	9,25	15,76	42,50
Error de comisión (%) (Umbral Iz=2 m)	40,30	39,47	40,28	50,13

GRAFICO 3:

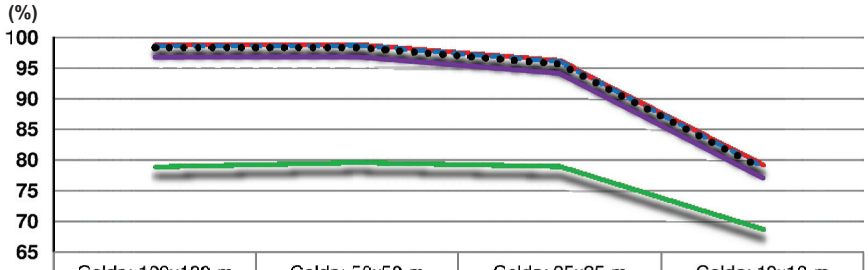
Muestra la media entre los errores de omisión y comisión obtenidos tras ejecutar el algoritmo A para los cinco umbrales en desnivel utilizados. Además el gráfico muestra la variabilidad de los resultados en función del tamaño de bloque o celda utilizado para generar el terreno provisional inicial.



	Celda: 100x100 m	Celda: 50x50 m	Celda: 25x25 m	Celda: 10x10 m
Media de errores (%) (Umbral Iz=0,2 m)	1,88	1,80	5,50	20,16
Media de errores (%) (Umbral Iz=0,4 m)	2,06	2,03	5,68	20,28
Media de errores (%) (Umbral Iz=0,6 m)	2,52	2,52	6,11	20,49
Media de errores (%) (Umbral Iz=1 m)	4,69	4,66	7,93	21,32
Media de errores (%) (Umbral Iz=2 m)	20,17	19,76	20,17	25,13

GRAFICO 4:

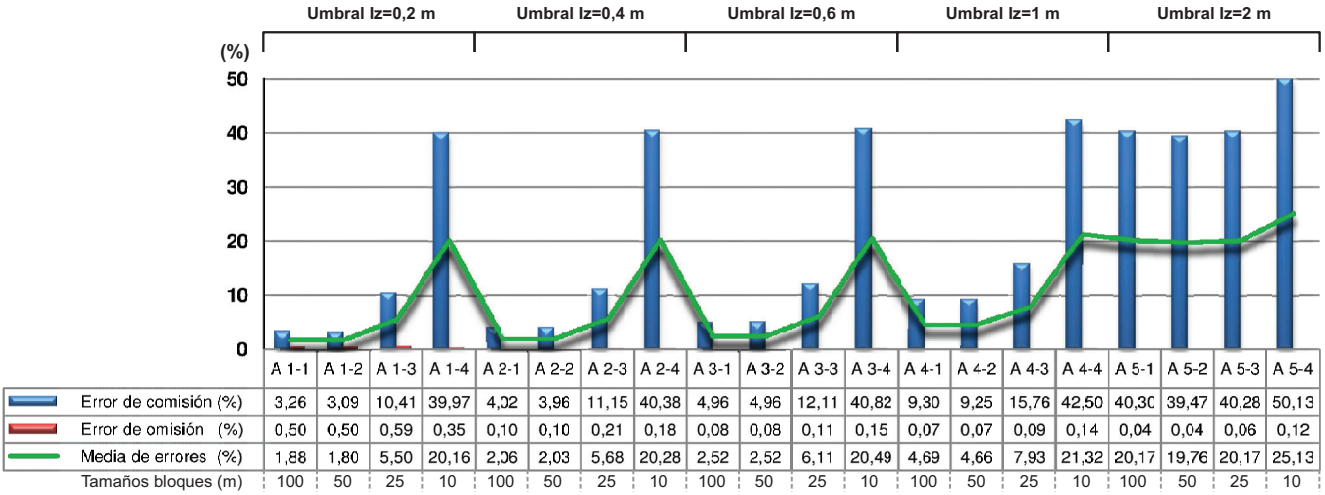
Muestra la exactitud global de la clase terreno obtenidos tras ejecutar el algoritmo A para los cinco umbrales en desnivel utilizados. Además el gráfico muestra la variabilidad de los resultados en función del tamaño de bloque o celda utilizado para generar el terreno provisional inicial.



	Celda: 100x100 m	Celda: 50x50 m	Celda: 25x25 m	Celda: 10x10 m
Exactitud global (%) (Umbral Iz=0,2 m)	98,81	98,86	96,23	79,30
Exactitud global (%) (Umbral Iz=0,4 m)	98,67	98,69	96,05	78,96
Exactitud global (%) (Umbral Iz=0,6 m)	98,36	98,36	95,70	78,58
Exactituc global (%) (Umbral Iz=1 m)	96,80	96,82	94,17	77,05
Exactituc global (%) (Umbral Iz=2 m)	79,05	79,76	79,06	68,80

GRAFICO 5:

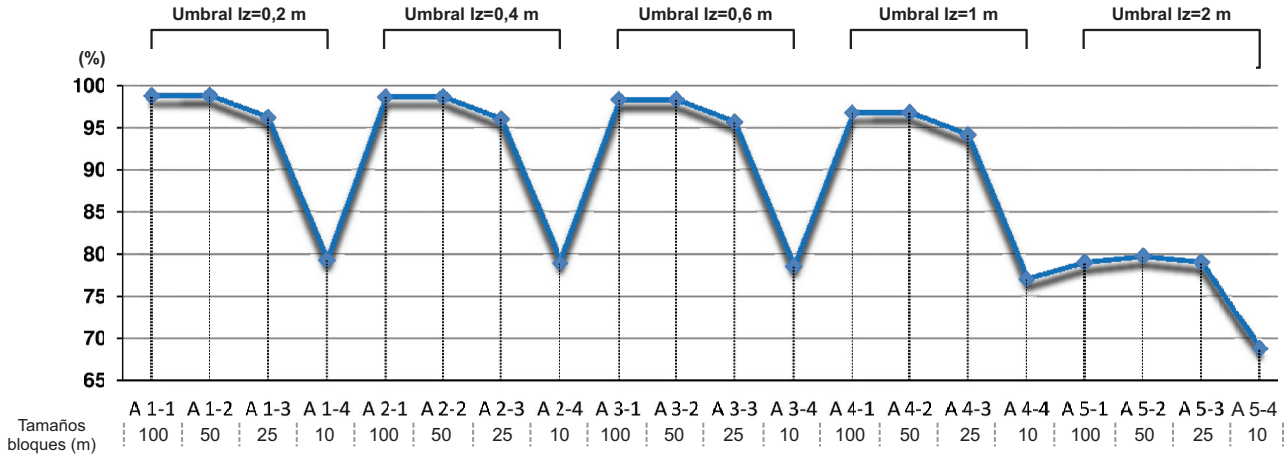
Muestra la media entre los errores de omisión y comisión obtenidos tras ejecutar el algoritmo A para los cinco umbrales en desnivel utilizados. Además el gráfico muestra la variabilidad de los resultados en función del tamaño de bloque o celda utilizado para generar el terreno provisional inicial.



	A 1-1	A 1-2	A 1-3	A 1-4	A 2-1	A 2-2	A 2-3	A 2-4	A 3-1	A 3-2	A 3-3	A 3-4	A 4-1	A 4-2	A 4-3	A 4-4	A 5-1	A 5-2	A 5-3	A 5-4
Error de comisión (%)	3,26	3,09	10,41	39,97	4,02	3,96	11,15	40,38	4,96	4,96	12,11	40,82	9,30	9,25	15,76	42,50	40,30	39,47	40,28	50,13
Error de omisión (%)	0,50	0,50	0,59	0,35	0,10	0,10	0,21	0,18	0,08	0,08	0,11	0,15	0,07	0,07	0,09	0,14	0,04	0,04	0,06	0,12
Media de errores (%)	1,88	1,80	5,50	20,16	2,06	2,03	5,68	20,28	2,52	2,52	6,11	20,49	4,69	4,66	7,93	21,32	20,17	19,76	20,17	25,13
Tamaños bloques (m)	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10

GRAFICO 6:

Muestra la exactitud global para la totalidad de resultados obtenidos tras ejecutar el algoritmo A de clasificación del terreno. En el gráfico puede apreciarse la variabilidad de la precisión de los resultados en función del umbral en desnivel y su dependencia al tamaño de bloque utilizado para definir el terreno provisional inicial.



ALGORITMO: DETECCIÓN DEL TERRENO MEDIANTE METODO DE DENSIFICACIÓN QUE COMBINA INTERPOLACIÓN Y SEGMENTACIÓN Y ANALIZA CARACTERÍSTICAS DE LAS AGRUPACIONES (ALGORITMO B)

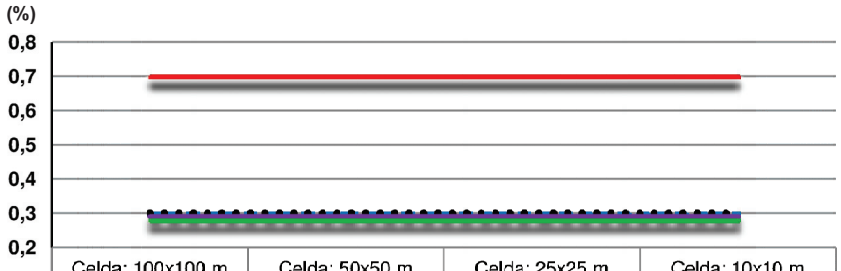
ZONA TEST: Vall d'Uixó

DESCRIPCIÓN:

Algoritmo de densificación progresiva para detectar la clase terreno. La superficie de referencia se obtiene mediante interpolación de distantancia inversa ponderada. El algoritmo de densificación se combina con un proceso de segmentación. Las agrupaciones obtenidas tras la segmentación son clasificadas como terreno en un proceso iterativo en función de su distancia a la superficie de referencia. El proceso lleva asociado un análisis de las características de agrupaciones colindantes con el fin de reducir al máximo errores en la clasificación y conseguir además una mayor automatización del proceso.

GRAFICO 1:

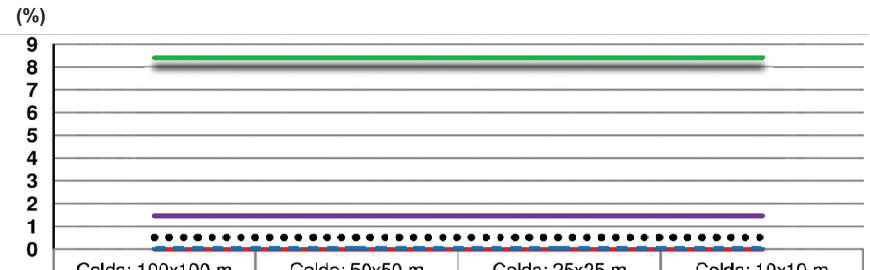
Muestra los errores de omisión obtenidos tras ejecutar el algoritmo B para los cinco umbrales en desnivel utilizados. Además el gráfico muestra la variabilidad de los resultados en función del tamaño de bloque o celda utilizado para generar el terreno provisional inicial.



	Celda: 100x100 m	Celda: 50x50 m	Celda: 25x25 m	Celda: 10x10 m
Error de omisión (%) (Umbral Iz=0,2 m)	0,70	0,70	0,70	0,70
Error de omisión (%) (Umbral Iz=0,4 m)	0,30	0,30	0,30	0,30
Error de omisión (%) (Umbral Iz=0,6 m)	0,30	0,30	0,30	0,30
Error de omisión (%) (Umbral Iz=1 m)	0,29	0,29	0,29	0,29
Error de omisión (%) (Umbral Iz=2 m)	0,28	0,28	0,28	0,28

GRAFICO 2:

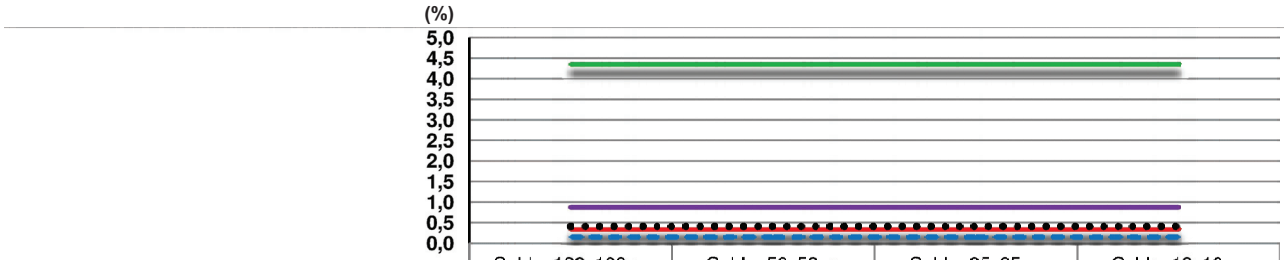
Muestra los errores de comisión obtenidos tras ejecutar el algoritmo B para los cinco umbrales en desnivel utilizados. Además el gráfico muestra la variabilidad de los resultados en función del tamaño de bloque o celda utilizado para generar el terreno provisional inicial.



	Celda: 100x100 m	Celda: 50x50 m	Celda: 25x25 m	Celda: 10x10 m
Error de comisión (%) (Umbral Iz=0,2 m)	0,01	0,01	0,01	0,01
Error de comisión (%) (Umbral Iz=0,4 m)	0,03	0,03	0,03	0,03
Error de comisión (%) (Umbral Iz=0,6 m)	0,51	0,51	0,51	0,51
Error de comisión (%) (Umbral Iz=1 m)	1,46	1,46	1,46	1,46
Error de comisión (%) (Umbral Iz=2 m)	8,45	8,45	8,45	8,45

GRAFICO 3:

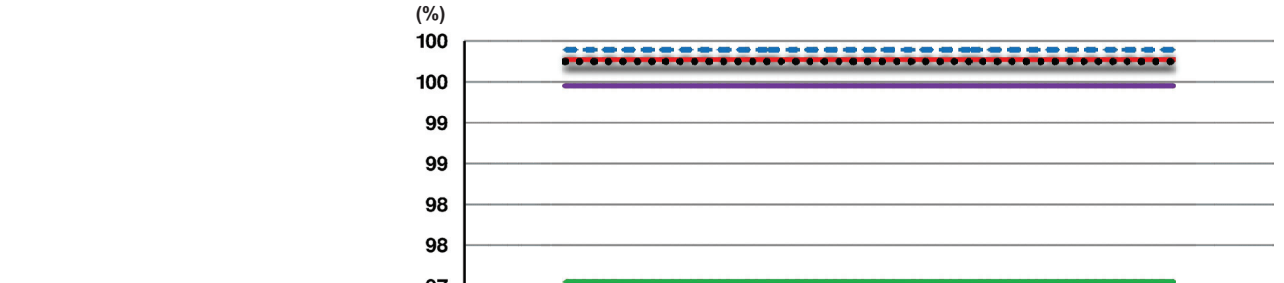
Muestra la media entre los errores de omisión y comisión obtenidos tras ejecutar el algoritmo B para los cinco umbrales en desnivel utilizados. Además el gráfico muestra la variabilidad de los resultados en función del tamaño de bloque o celda utilizado para generar el terreno provisional inicial.



	Celda: 100x100 m	Celda: 50x50 m	Celda: 25x25 m	Celda: 10x10 m
Media de errores (%) (Umbral Iz=0,2 m)	0,36	0,36	0,36	0,36
Media de errores (%) (Umbral Iz=0,4 m)	0,17	0,17	0,17	0,17
Media de errores (%) (Umbral Iz=0,6 m)	0,41	0,41	0,41	0,41
Media de errores (%) (Umbral Iz=1 m)	0,88	0,88	0,88	0,88
Media de errores (%) (Umbral Iz=2 m)	4,37	4,37	4,37	4,37

GRAFICO 4:

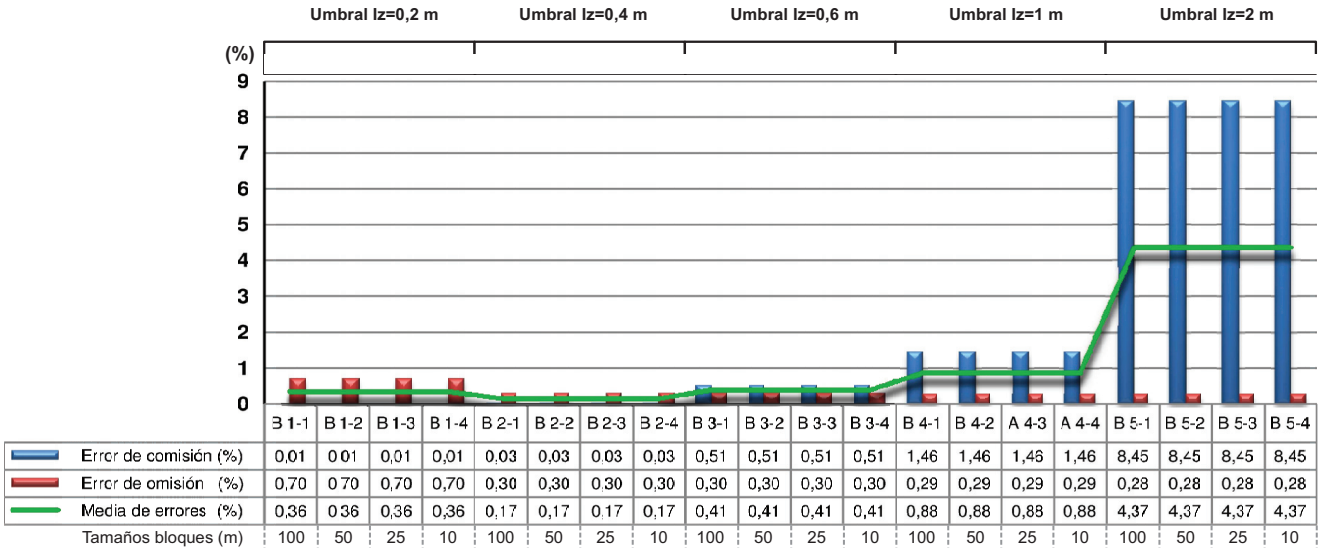
Muestra la exactitud global de la clase terreno obtenidos tras ejecutar el algoritmo B para los cinco umbrales en desnivel utilizados. Además el gráfico muestra la variabilidad de los resultados en función del tamaño de bloque o celda utilizado para generar el terreno provisional inicial.



	Celda: 100x100 m	Celda: 50x50 m	Celda: 25x25 m	Celda: 10x10 m
Exactitud global (%) (Umbral Iz=0,2 m)	99,78	99,78	99,78	99,78
Exactitud global (%) (Umbral Iz=0,4 m)	99,90	99,90	99,90	99,90
Exactitud global (%) (Umbral Iz=0,6 m)	99,75	99,75	99,75	99,75
Exactitud global (%) (Umbral Iz=1 m)	99,45	99,45	99,45	99,45
Exactitud global (%) (Umbral Iz=2 m)	97,06	97,06	97,06	97,06

GRAFICO 5:

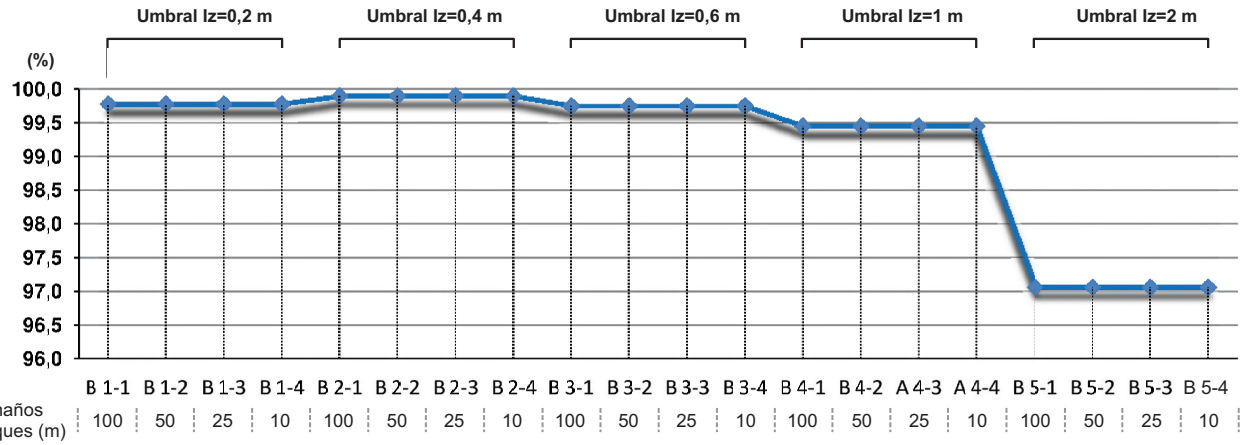
Muestra la media entre los errores de omisión y comisión obtenidos tras ejecutar el algoritmo B para los cinco umbrales en desnivel utilizados. Además el gráfico muestra la consistencia del algoritmo B, ya que la precisión de los resultados se mantiene constante al variar el tamaño del bloque o celda utilizado para generar el terreno provisional inicial.



	B 1-1	B 1-2	B 1-3	B 1-4	B 2-1	B 2-2	B 2-3	B 2-4	B 3-1	B 3-2	B 3-3	B 3-4	B 4-1	B 4-2	A 4-3	A 4-4	B 5-1	B 5-2	B 5-3	B 5-4
Error de comisión (%)	0,01	0 01	0,01	0,01	0,03	0,03	0 03	0,03	0,51	0,51	0,51	0,51	1,46	1,46	1,46	1,46	8,45	8,45	8,45	8,45
Error de omisión (%)	0,70	0 70	0,70	0,70	0,30	0,30	0 30	0,30	0,30	0,30	0,30	0,30	0,29	0,29	0,29	0,29	0,28	0,28	0,28	0,28
Media de errores (%)	0,36	0 36	0,36	0,36	0,17	0,17	0 17	0,17	0,41	0,41	0,41	0,41	0,88	0,88	0,88	0,88	4,37	4,37	4,37	4,37
Tamaños bloques (m)	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10

GRAFICO 6:

Muestra la exactitud global para la totalidad de resultados obtenidos tras ejecutar el algoritmo B de clasificación del terreno. Además el gráfico muestra la consistencia del algoritmo B, ya que la exactitud global se mantiene constante al variar el tamaño del bloque o celda utilizado para generar el terreno provisional inicial.



	B 1-1	B 1-2	B 1-3	B 1-4	B 2-1	B 2-2	B 2-3	B 2-4	B 3-1	B 3-2	B 3-3	B 3-4	B 4-1	B 4-2	A 4-3	A 4-4	B 5-1	B 5-2	B 5-3	B 5-4
Tamaños bloques (m)	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10

ALGORITMO: DETECCIÓN DEL TERRENO MEDIANTE METODO DE DENSIFICACIÓN QUE COMBINA INTERPOLACIÓN Y SEGMENTACIÓN. COMPARACIÓN ENTRE LOS RESULTADOS DEL ALGORITMO A Y B.

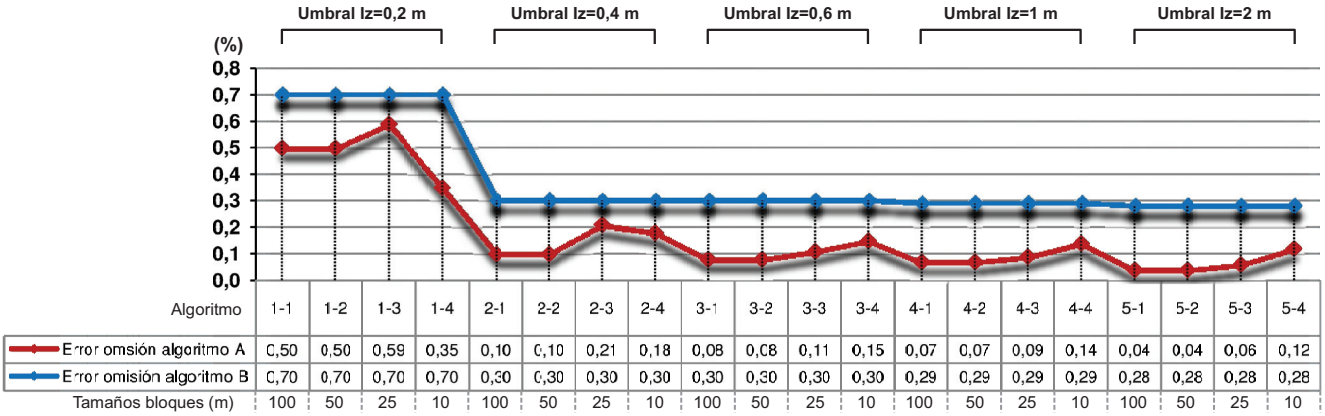
ZONA TEST: Vall d'Uixó

DESCRIPCIÓN:

Comparación de los resultados obtenidos tras ejecutar los dos algoritmos para detectar el terreno (algoritmo A y B). Ambos algoritmos utilizan un proceso de densificación progresiva combinado con un proceso de segmentación. El algoritmo A no realiza análisis alguno de las características de agrupaciones colindantes para determinar la clase terreno, mientras que el algoritmo B analiza las características de agrupaciones colindantes con el fin de reducir al máximo errores en la clasificación y conseguir además una mayor automatización del proceso.

GRAFICO 1:

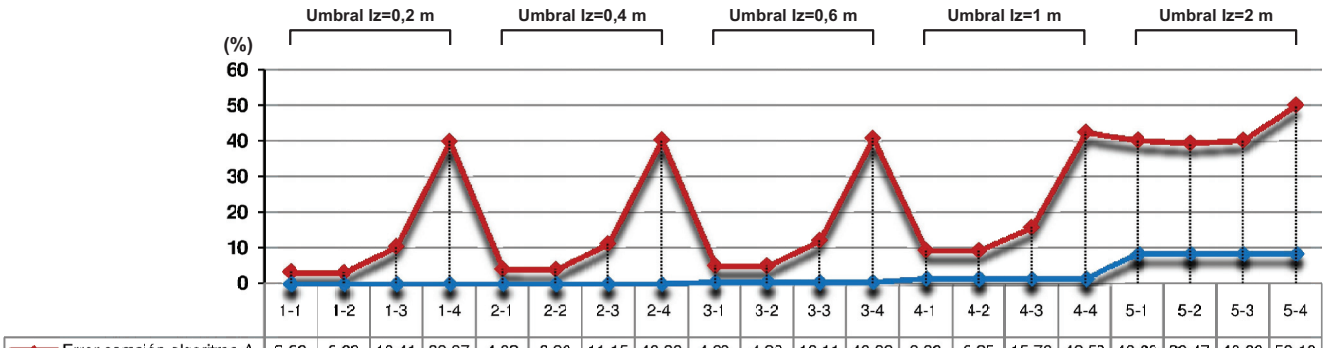
Muestra los errores de omisión obtenidos tras ejecutar el algoritmo A y el algoritmo B.



Algoritmo	1-1	1-2	1-3	1-4	2-1	2-2	2-3	2-4	3-1	3-2	3-3	3-4	4-1	4-2	4-3	4-4	5-1	5-2	5-3	5-4
Error omsión algoritmo A	0,50	0,50	0,59	0,35	0,10	0,10	0,21	0,18	0,08	0,08	0,11	0,15	0,07	0,07	0,09	0,14	0,04	0,04	0,06	0,12
Error omisión algoritmo B	0,70	0,70	0,70	0,70	0,30	0,30	0,30	0,30	0,30	0,30	0,30	0,30	0,29	0,29	0,29	0,29	0,28	0,28	0,28	0,28
Tamaños bloques (m)	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10

GRAFICO 2:

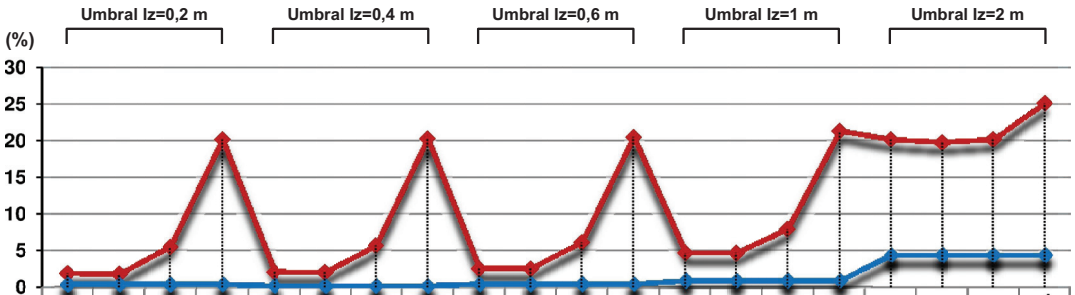
Muestra los errores de comisión obtenidos tras ejecutar el algoritmo A y el algoritmo B.



	1-1	1-2	1-3	1-4	2-1	2-2	2-3	2-4	3-1	3-2	3-3	3-4	4-1	4-2	4-3	4-4	5-1	5-2	5-3	5-4
Error comsión algoritmo A	3,26	3,09	10,41	39,97	4,02	3,96	11,15	40,38	4,96	4,95	12,11	40,82	9,30	9,25	15,76	42,50	40,30	39,47	40,28	50,13
Error comisión algoritmo B	0,01	0,01	0,01	0,01	0,03	0,03	0,03	0,03	0,51	0,51	0,51	0,51	1,46	1,46	1,46	1,46	8,45	8,45	8,45	8,45
Tamaños bloques (m)	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10

GRAFICO 3:

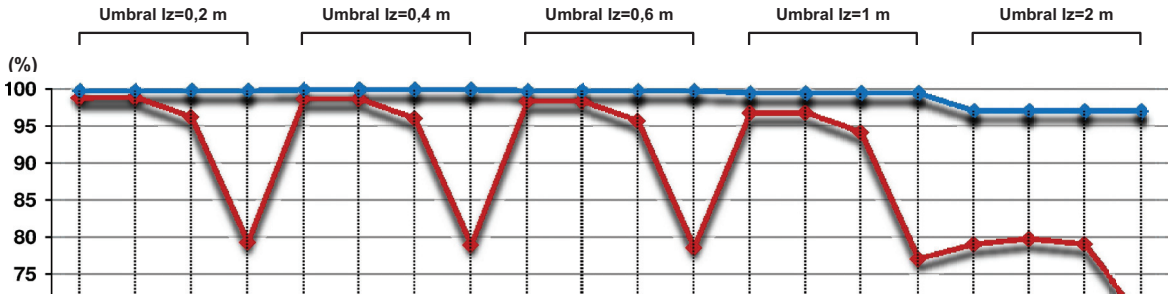
Muestra la media entre los errores de comisión y omisiónn obtenidos tras ejecutar el algoritmo A y el algoritmo B.



	A 1-1	A 1-2	A 1-3	A 1-4	A 2-1	A 2-2	A 2-3	A 2-4	A 3-1	A 3-2	A 3-3	A 3-4	A 4-1	A 4-2	A 4-3	A 4-4	A 5-1	A 5-2	A 5-3	A 5-4
Media de errores algoritmo A	1,88	1,80	5,50	20,16	2,06	2,03	5,68	20,28	2,52	2,52	6,11	20,49	4,69	4,66	7,93	21,32	20,17	19,76	20,17	25,13
Media de errores algoritmo B	0,36	0,36	0,36	0,36	0,17	0,17	0,17	0,17	0,41	0,41	0,41	0,41	0,88	0,88	0,88	0,88	4,37	4,37	4,37	4,37
Tamaños bloques (m)	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10

GRAFICO 4:

Muestra la exactitud global obtenida tras ejecutar el algoritmo A y el algoritmo B.



	1-1	1-2	1-3	1-4	2-1	2-2	2-3	2-4	3-1	3-2	3-3	3-4	4-1	4-2	4-3	4-4	5-1	5-2	5-3	5-4
Exactitud global A	98,81	98,86	96,23	79,30	98,67	98,69	96,05	78,96	98,36	98,36	95,70	78,58	96,80	96,82	94,17	77,05	79,05	79,76	79,06	68,80
Exactitud global B	99,78	99,78	99,78	99,78	99,90	99,90	99,90	99,90	99,75	99,75	99,75	99,75	99,45	99,45	99,45	99,45	97,06	97,06	97,06	97,06
Tamaños bloques (m)	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10

ALGORITMO: DETECCIÓN DEL TERRENO MEDIANTE METODO DE DENSIFICACIÓN QUE COMBINA INTERPOLACIÓN Y SEGMENTACIÓN SIN ANALIZAR CARACTERÍSTICAS DE LAS AGRUPACIONES (ALGORITMO A)

ZONA TEST: Collado de Villalba

DESCRIPCIÓN:

Algoritmo de densificación progresiva para detectar la clase terreno. La superficie de referencia se obtiene mediante interpolación de distantancia inversa ponderada. El algoritmo de densificación se combina con un proceso de segmentación. Las agrupaciones obtenidas tras la segmentación son clasificadas como terreno en un proceso iterativo en función de su distancia a la superficie de referencia. No se realiza análisis alguno de las características de agrupaciones colindantes para determinar la clase terreno.

PARÁMETROS UMBRALES FIJOS PARA ALGORITMO A:

UMBRAL EN IZ SEGMENTACIÓN: 0,25 m **TAMAÑO DE CELDA:** 1 m **EXPONENTE DE PONDERACIÓN EN INTERPOLACIÓN:** 3

ALGORITMO: A 1 - 1

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 0,2 m **TAMAÑO DE CUADRANTES:** 100 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**1438698**	14507	1453205	99,00 %	1,00 %
No terreno	68570	**1302947**	1371517	95,00 %	5,00 %
Total	1507268	1317454	**2824722**	Exactitud global: 97,06 %	
Ex. Usuario	95,45 %	98,90 %			
Error. Com	4,55 %	1,10 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Puentes	
Terreno	11663	50862	6045	Nº pts
	17,01 %	74,18 %	8,82 %	%

ALGORITMO: A 1 - 2

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 0,2 m **TAMAÑO DE CUADRANTES:** 50 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**1438295**	14910	1453205	98,97 %	1,03 %
No terreno	70320	**1301197**	1371517	94,87 %	5,13 %
Total	1508615	1316107	**2824722**	Exactitud global: 96,98 %	
Ex. Usuario	95,34 %	98,87 %			
Error. Com	4,66 %	1,13 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Puentes	
Terreno	14220	50055	6045	Nº pts
	20,22 %	71,18 %	8,60 %	%

ALGORITMO: A 1 - 3

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 0,2 m **TAMAÑO DE CUADRANTES:** 25 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**1439952**	13253	1453205	99,09 %	0,91 %
No terreno	84762	**1286755**	1371517	93,82 %	6,18 %
Total	1524714	1300008	**2824722**	Exactitud global: 96,53 %	
Ex. Usuario	94,44 %	98,98 %			
Error. Com	5,56 %	1,02 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Categorías deducidas de la clasificación			
	Edificio	Vegetación	Puentes	
Terreno	29546	49171	6045	Nº pts
	34,86 %	58,01 %	7,13 %	%

Categorías d.c : Categorías deducidas de la clasificación

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

ALGORITMO: A 1 - 4

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 0,2 m **TAMAÑO DE CUADRANTES:** 10 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**1447350**	5855	1453205	99,60 %	0,40 %
No terreno	248857	**1121962**	1371517	81,80 %	18,20 %
Total	1696905	1127817	**2824722**	Exactitud global: 90,96 %	
Ex. Usuario	85,29 %	99,48 %			
Error. Com	14,71 %	0,52 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Puentes	
Terreno	188955	54555	6045	**Nº pts**
	75,72 %	21,86 %	2,42 %	**%**

ALGORITMO: A 2 - 1

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 0,4 m **TAMAÑO DE CUADRANTES:** 100 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**1448001**	5204	1453205	99,64 %	0,36 %
No terreno	79860	**1291657**	1371517	94,18 %	5,82 %
Total	1527861	1296861	**2824722**	Exactitud global: 96,99 %	
Ex. Usuario	94,77 %	99,60 %			
Error. Com	5,23 %	0,40 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Puentes	
Terreno	14976	58839	5347	**Nº pts**
	18,92 %	74,33 %	6,75 %	**%**

ALGORITMO: A 2 - 2

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 0,4 m **TAMAÑO DE CUADRANTES:** 50 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**1447841**	5364	1453205	99,63 %	0,37 %
No terreno	81575	**1289942**	1371517	94,05 %	5,95 %
Total	1529416	1295306	**2824722**	Exactitud global: 96,92 %	
Ex. Usuario	94,67 %	99,59 %			
Error. Com	5,33 %	0,41 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Puentes	
Terreno	17545	57985	6045	**Nº pts**
	21,51 %	71,08 %	7,41 %	**%**

ALGORITMO: A 2 - 3

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 0,4 m **TAMAÑO DE CUADRANTES:** 25 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**1450115**	3090	1453205	99,79 %	0,21 %
No terreno	96045	**1275472**	1371517	93,00 %	7,00 %
Total	1546160	1278562	**2824722**	Exactitud global: 96,49 %	
Ex. Usuario	93,79 %	99,76 %			
Error. Com	6,21 %	0,24 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Puentes	
Terreno	32829	5717	6045	**Nº pts**
	34,18 %	59,53 %	6,29 %	**%**

Categorías d.c : Categorías deducidas de la clasificación

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

ALGORITMO: A 2 - 4

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 0,4 m **TAMAÑO DE CUADRANTES:** 10 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**1451338**	1867	1453205	99,87 %	0,13 %
No terreno	260629	**1110888**	1371517	81,00 %	19,00 %
Total	1711967	1112755	**2824722**	Exactitud global: 90,71 %	
Ex. Usuario	84,78 %	99,83 %			
Error. Com	15,22 %	0,17 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Puentes	
Terreno	192518	62066	6045	Nº pts
	73,87 %	23,81 %	2,32 %	%

ALGORITMO: A 3- 1

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 0,6 m **TAMAÑO DE CUADRANTES:** 100 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**1452436**	769	1453205	99,95 %	0,05 %
No terreno	95701	**1275816**	1371517	93,02 %	6,98 %
Total	1548137	1276585	**2824722**	Exactitud global: 96,58 %	
Ex. Usuario	93,82 %	99,94 %			
Error. Com	6,18 %	0,06 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Puentes	
Terreno	20075	69581	6045	Nº pts
	20,98 %	72,71 %	6,32 %	%

ALGORITMO: A 3 - 2

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 0,6 m **TAMAÑO DE CUADRANTES:** 50 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**1452270**	935	1453205	99,94 %	0,06 %
No terreno	97104	**1274413**	1371517	92,92 %	7,08 %
Total	1549374	1275348	**2824722**	Exactitud global: 96,53 %	
Ex. Usuario	93,73 %	99,93 %			
Error. Com	6,27 %	0,07 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Puentes	
Terreno	22402	68657	6045	Nº pts
	23,07 %	70,70 %	6,23 %	%

ALGORITMO: A 3 - 3

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 0,6 m **TAMAÑO DE CUADRANTES:** 25 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**1452025**	1180	1453205	99,92 %	0,08 %
No terreno	111433	**1260084**	1371517	91,88 %	8,12 %
Total	1563458	1261264	**2824722**	Exactitud global: 96,01 %	
Ex. Usuario	92,87 %	99,91 %			
Error. Com	7,13 %	0,09 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Puentes	
Terreno	37944	67444	6045	Nº pts
	34,05 %	60,52 %	5,42 %	%

Categorías d.c : Categorías deducidas de la clasificación

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

ALGORITMO: A 3 - 4

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 0,6 m **TAMAÑO DE CUADRANTES:** 10 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**1451704**	1501	1453205	99,90 %	0,10 %
No terreno	275167	**1096350**	1371517	79,94 %	20,06 %
Total	1726871	1097851	**2824722**	Exactitud global: 90,21 %	
Ex. Usuario	84,07 %	99,86 %			
Error. Com	15,93 %	0,14 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Puentes	
Terreno	197017	72105	6045	Nº pts
	71,60 %	26,27 %	2,20 %	%

ALGORITMO: A 4- 1

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 1 m **TAMAÑO DE CUADRANTES:** 100 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**1452591**	614	1453205	99,96 %	0,04 %
No terreno	130943	**1240574**	1371517	90,45 %	9,55 %
Total	1583534	1241188	**2824722**	Exactitud global: 95,34 %	
Ex. Usuario	91,73 %	99,95 %			
Error. Com	8,27 %	0,05 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Puentes	
Terreno	29645	95253	6045	Nº pts
	22,64 %	72,74 %	4,62 %	%

ALGORITMO: A 4 - 2

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 1 m **TAMAÑO DE CUADRANTES:** 50 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**1452455**	750	1453205	99,95 %	0,05 %
No terreno	133799	**1237718**	1371517	90,24 %	9,76 %
Total	1586254	1238468	**2824722**	Exactitud global: 95,24 %	
Ex. Usuario	91,57 %	99,94 %			
Error. Com	8,43 %	0,06 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Puentes	
Terreno	33267	94487	6045	Nº pts
	24,86 %	70,62 %	4,52 %	%

ALGORITMO: A 4 - 3

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 1 m **TAMAÑO DE CUADRANTES:** 25 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**1452274**	931	1453205	99,94 %	0,06 %
No terreno	148764	**1222753**	1371517	89,15 %	10,85 %
Total	1601038	1223684	**2824722**	Exactitud global: 94,70 %	
Ex. Usuario	90,71 %	99,92 %			
Error. Com	9,29 %	0,08 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Puentes	
Terreno	49930	92789	6045	Nº pts
	33,56 %	62,37 %	4,06 %	%

Categorías d.c : Categorías deducidas de la clasificación

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

ALGORITMO: A 4 - 4

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 1 m **TAMAÑO DE CUADRANTES:** 10 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**1451965**	1240	1453205	99,91 %	0,09 %
No terreno	309271	**1062246**	1371517	77,45 %	22,55 %
Total	1761236	1063486	**2824722**	Exactitud global: 89,01 %	
Ex. Usuario	82,44 %	99,88 %			
Error. Com	17,56 %	0,12 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Puentes	
Terreno	207105	96121	6045	**Nº pts**
	66,97 %	31,08 %	1,95 %	**%**

ALGORITMO: A 5- 1

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 2 m **TAMAÑO DE CUADRANTES:** 100 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**1452696**	509	1453205	99,96 %	0,04 %
No terreno	301120	**1070397**	1371517	78,04 %	21,96 %
Total	1753816	1070906	**2824722**	Exactitud global: 89,32 %	
Ex. Usuario	82,83 %	99,95 %			
Error. Com	17,17 %	0,05 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Puentes	
Terreno	129947	165128	6045	**Nº pts**
	43,15 %	54,84 %	2,01 %	**%**

ALGORITMO: A 5 - 2

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 2 m **TAMAÑO DE CUADRANTES:** 50 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**1452543**	662	1453205	99,95 %	0,05 %
No terreno	302419	**1069098**	1371517	77,95 %	22,05 %
Total	1754962	1069760	**2824722**	Exactitud global: 89,30 %	
Ex. Usuario	82,77 %	99,94 %			
Error. Com	17,23 %	0,06 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Puentes	
Terreno	132981	163393	6045	**Nº pts**
	43,97 %	54,03 %	2,00 %	**%**

ALGORITMO: A 5 - 3

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 2 m **TAMAÑO DE CUADRANTES:** 25 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**1452403**	802	1453205	99,94 %	0,06 %
No terreno	322384	**1049133**	1371517	76,49 %	23,51 %
Total	1774787	1049935	**2824722**	Exactitud global: 88,56 %	
Ex. Usuario	81,84 %	99,92 %			
Error. Com	18,16 %	0,08 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Puentes	
Terreno	155301	161038	6045	**Nº pts**
	48,17 %	49,95 %	1,88 %	**%**

Categorías d.c : Categorías deducidas de la clasificación

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

ALGORITMO: A 5 - 4

PARÁMETROS UMBRALES VARIABLES: **UMBRAL EN IZ:** 2 m **TAMAÑO DE CUADRANTES:** 10 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	**No terreno**	**Total**	**Ex. Prod.**	**Error. Om.**
Terreno	**1452076**	1129	1453205	99,92 %	0,08 %
No terreno	456427	**915090**	1371517	66,72 %	33,28 %
Total	1908503	916219	**2824722**	Exactitud global: 83,80 %	
Ex. Usuario	76,08 %	99,88 %			
Error. Com	23,92 %	0,12 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	**Vegetación**	**Puentes**	
Terreno	290518	159864	6045	**Nº pts**
	63,65 %	35,03 %	1,32 %	**%**

Categorías d.c : Categorías deducidas de la clasificación

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

ALGORITMO: DETECCIÓN DEL TERRENO MEDIANTE METODO DE DENSIFICACIÓN QUE COMBINA INTERPOLACIÓN Y SEGMENTACIÓN Y ANALIZA CARACTERÍSTICAS DE LAS AGRUPACIONES (ALGORITMO B)

ZONA TEST: Collado de Villalba

DESCRIPCIÓN:

Algoritmo de densificación progresiva para detectar la clase terreno. La superficie de referencia se obtiene mediante interpolación de distantancia inversa ponderada. El algoritmo de densificación se combina con un proceso de segmentación. Las agrupaciones obtenidas tras la segmentación son clasificadas como terreno en un proceso iterativo en función de su distancia a la superficie de referencia. El proceso lleva asociado un análisis de las características de agrupaciones colindantes con el fin de reducir al máximo errores en la clasificación y conseguir además una mayor automatización del proceso.

PARÁMETROS UMBRALES FIJOS PARA ALGORITMO B:

UMBRAL EN IZ SEGMENTACIÓN: 0,25 m **TAMAÑO DE CELDA:** 1 m **EXPONENTE DE PONDERACIÓN EN INTERPOLACIÓN:** 3

ALGORITMO: B 1 - 1, B 1 - 2

PARÁMETROS UMBRALES VARIABLES (B 1-1): **UMBRAL EN IZ:** 0,2 m **TAMAÑO DE CUADRANTES:** 100 m

PARÁMETROS UMBRALES VARIABLES (B 1-2): **UMBRAL EN IZ:** 0,2 m **TAMAÑO DE CUADRANTES:** 50 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**1439060**	14145	1453205	99,03 %	0,97 %
No terreno	14040	**1357477**	1371517	98,98 %	1,02 %
Total	1453100	1371622	**2824722**	Exactitud global: 99,00 %	
Ex. Usuario	99,03 %	98,97 %			
Error. Com	0,97 %	1,03 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Puentes	
Terreno	36	7959	6045	Nº pts
	0,26 %	56,69 %	43,06 %	%

ALGORITMO: B 1 - 3, B 1 - 4

PARÁMETROS UMBRALES VARIABLES (B 1-3): **UMBRAL EN IZ:** 0,2 m **TAMAÑO DE CUADRANTES:** 25 m

PARÁMETROS UMBRALES VARIABLES (B 1-4): **UMBRAL EN IZ:** 0,2 m **TAMAÑO DE CUADRANTES:** 10 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**1439046**	14159	1453205	99,03 %	0,97 %
No terreno	14040	**1357477**	1371517	98,98 %	1,02 %
Total	1453086	1371636	**2824722**	Exactitud global: 99,00 %	
Ex. Usuario	99,03 %	98,97 %			
Error. Com	0,97 %	1,03 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Puentes	
Terreno	36	7959	6045	Nº pts
	0,26 %	56,69 %	43,06 %	%

Categorías d.c : Categorías deducidas de la clasificación

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

ALGORITMO: B 2 - 1, B 2 - 2

PARÁMETROS UMBRALES VARIABLES (B 2-1): **UMBRAL EN IZ:** 0,4 m **TAMAÑO DE CUADRANTES:** 100 m

PARÁMETROS UMBRALES VARIABLES (B 2-2): **UMBRAL EN IZ:** 0,4 m **TAMAÑO DE CUADRANTES:** 50 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**1448331**	4874	1453205	99,66 %	0,34 %
No terreno	14368	**1357149**	1371517	98,95 %	1,05 %
Total	1462699	1362023	**2824722**	Exactitud global: 99,32 %	
Ex. Usuario	99,02 %	99,64 %			
Error. Com	0,98 %	0,36 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Puentes	
Terreno	96	8227	6045	Nº pts
	0,67 %	57,26 %	42,07 %	%

ALGORITMO: B 2 - 3, B 2 - 4

PARÁMETROS UMBRALES VARIABLES (B 2-3): **UMBRAL EN IZ:** 0,4 m **TAMAÑO DE CUADRANTES:** 25 m

PARÁMETROS UMBRALES VARIABLES (B 2-4): **UMBRAL EN IZ:** 0,4 m **TAMAÑO DE CUADRANTES:** 10 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**1448283**	4922	1453205	99,66 %	0,34 %
No terreno	14305	**1357212**	1371517	98,96 %	1,04 %
Total	1462588	1362134	**2824722**	Exactitud global: 99,32 %	
Ex. Usuario	99,02 %	99,64 %			
Error. Com	0,98 %	0,36 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Puentes	
Terreno	36	8224	6045	Nº pts
	0,25 %	57,49 %	42,26 %	%

ALGORITMO: B 3 - 1, B 3 - 2

PARÁMETROS UMBRALES VARIABLES (B 3-1): **UMBRAL EN IZ:** 0,6 m **TAMAÑO DE CUADRANTES:** 100 m

PARÁMETROS UMBRALES VARIABLES (B 3-2): **UMBRAL EN IZ:** 0,6 m **TAMAÑO DE CUADRANTES:** 50 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**1448806**	4399	1453903	99,70 %	0,30 %
No terreno	19029	**1352488**	1370819	98,61 %	1,39 %
Total	1467835	1356887	**2824722**	Exactitud global: 99,17 %	
Ex. Usuario	98,70 %	99,68 %			
Error. Com	1,30 %	0,32 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Puentes	
Terreno	1200	11784	6045	Nº pts
	6,31 %	61,93 %	31,77 %	%

ALGORITMO: B 3 - 3, B 3 - 4

PARÁMETROS UMBRALES VARIABLES (B 3-3): **UMBRAL EN IZ:** 0,6 m **TAMAÑO DE CUADRANTES:** 25 m

PARÁMETROS UMBRALES VARIABLES (B 3-4): **UMBRAL EN IZ:** 0,6 m **TAMAÑO DE CUADRANTES:** 10 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**1448799**	4406	1453205	99,70 %	0,30 %
No terreno	19029	**1352488**	1371517	98,61 %	1,39 %
Total	1467828	1356894	**2824722**	Exactitud global: 99,17 %	
Ex. Usuario	98,70 %	99,68 %			
Error. Com	1,30 %	0,32 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Puentes	
Terreno	1200	11784	6045	Nº pts
	6,31 %	61,93 %	31,77 %	%

Categorías d.c : Categorías deducidas de la clasificación

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

ANÁLISIS DE RESULTADOS (MATRICES DE CONFUSIÓN)

ALGORITMO: B 4 - 1, B 4 - 2, B 4 - 3

PARÁMETROS UMBRALES VARIABLES (B 4-1): **UMBRAL EN IZ:** 1 m **TAMAÑO DE CUADRANTES:** 100 m

PARÁMETROS UMBRALES VARIABLES (B 4-2): **UMBRAL EN IZ:** 1 m **TAMAÑO DE CUADRANTES:** 50 m

PARÁMETROS UMBRALES VARIABLES (B 4-3): **UMBRAL EN IZ:** 1 m **TAMAÑO DE CUADRANTES:** 25 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**1450146**	3059	1453205	99,79 %	0,21 %
No terreno	28191	**1343326**	1371517	97,94 %	2,06 %
Total	1478337	1346385	**2824722**	Exactitud global: 98,89 %	
Ex. Usuario	98,09 %	99,77 %			
Error. Com	1,91 %	0,23 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Puentes	
Terreno	3243	18903	6045	Nº pts
	11,50 %	67,05 %	21,44 %	%

ALGORITMO: B 4 - 4

PARÁMETROS UMBRALES VARIABLES (B 4-4): **UMBRAL EN IZ:** 1 m **TAMAÑO DE CUADRANTES:** 10 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**1450146**	3059	1453205	99,79 %	0,21 %
No terreno	28106	**1343411**	1371517	97,95 %	2,05 %
Total	1478252	1346470	**2824722**	Exactitud global: 98,90 %	
Ex. Usuario	98,10 %	99,77 %			
Error. Com	1,90 %	0,23 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Puentes	
Terreno	3227	18834	6045	Nº pts
	11,48 %	67,01 %	21,51 %	%

ALGORITMO: B 5 - 1, B 5 - 2, B 5 - 3, B 5 - 4

PARÁMETROS UMBRALES VARIABLES (B 5-1): **UMBRAL EN IZ:** 2 m **TAMAÑO DE CUADRANTES:** 100 m

PARÁMETROS UMBRALES VARIABLES (B 5-2): **UMBRAL EN IZ:** 2 m **TAMAÑO DE CUADRANTES:** 50 m

PARÁMETROS UMBRALES VARIABLES (B 5-3): **UMBRAL EN IZ:** 2 m **TAMAÑO DE CUADRANTES:** 25 m

PARÁMETROS UMBRALES VARIABLES (B 5-4): **UMBRAL EN IZ:** 2 m **TAMAÑO DE CUADRANTES:** 10 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación				
	Terreno	No terreno	Total	Ex. Prod.	Error. Om.
Terreno	**1450146**	3059	1453205	99,79 %	0,21 %
No terreno	31141	**1340376**	1371517	97,73 %	2,27 %
Total	1481287	1343435	**2824722**	Exactitud global: 98,79 %	
Ex. Usuario	97,90 %	99,77 %			
Error. Com	2,10 %	0,32 %			

INFORMACIÓN DE ERRORES POR EXCESO

Categorías d.c	Clases de referencia			
	Edificio	Vegetación	Puentes	
Terreno	5248	19848	6045	Nº pts
	16,85 %	63,74 %	19,41 %	%

Categorías d.c : Categorías deducidas de la clasificación

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

ALGORITMO: DETECCIÓN DEL TERRENO MEDIANTE METODO DE DENSIFICACIÓN QUE COMBINA INTERPOLACIÓN Y SEGMENTACIÓN SIN ANALIZAR CARACTERÍSTICAS DE LAS AGRUPACIONES (ALGORITMO A)

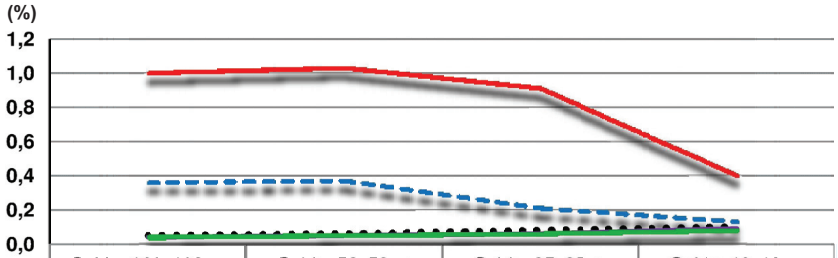
ZONA TEST: Collado de Villalba

DESCRIPCIÓN:

Algoritmo de densificación progresiva para detectar la clase terreno. La superficie de referencia se obtiene mediante interpolación de distantancia inversa ponderada. El algoritmo de densificación se combina con un proceso de segmentación. Las agrupaciones obtenidas tras la segmentación son clasificadas como terreno en un proceso iterativo en función de su distancia a la superficie de referencia. No se realiza análisis alguno de las características de agrupaciones colindantes para determinar la clase terreno.

GRAFICO 1:

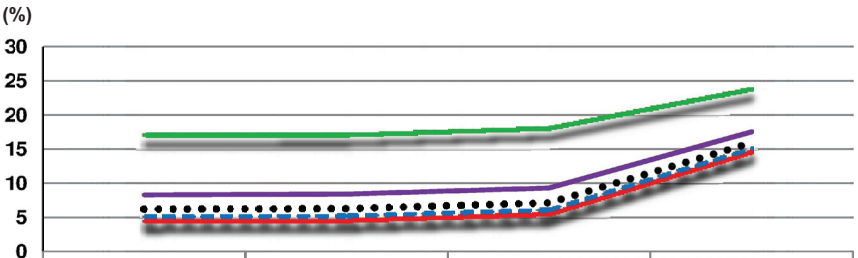
Muestra los errores de omisión obtenidos tras ejecutar el algoritmo A para los cinco umbrales en desnivel utilizados. Además el gráfico muestra la variabilidad de los resultados en función del tamaño de bloque o celda utilizado para generar el terreno provisional inicial.



	Celda: 100x100 m	Celda: 50x50 m	Celda: 25x25 m	Celda: 10x10 m
Error de omisión (%) (Umbral Iz=0,2 m)	1,00	1,03	0,91	0,40
Error de omisión (%) (Umbral Iz=0,4 m)	0,36	0,37	0,21	0,13
Error de omisión (%) (Umbral Iz=0,6 m)	0,05	0,06	0,08	0,10
Error de omisión (%) (Umbral Iz=1 m)	0,04	0,05	0,06	0,09
Error de omisión (%) (Umbral Iz=2 m)	0,04	0,05	0,06	0,08

GRAFICO 2:

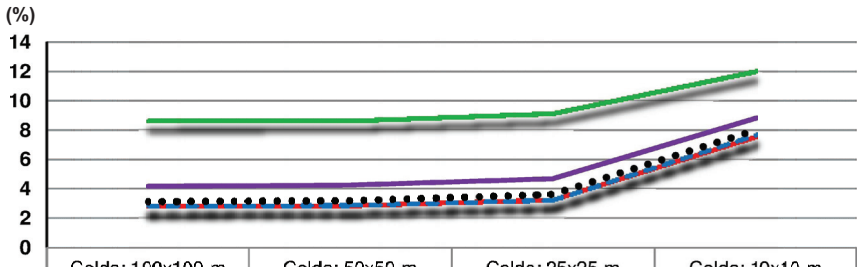
Muestra los errores de comisión obtenidos tras ejecutar el algoritmo A para los cinco umbrales en desnivel utilizados. Además el gráfico muestra la variabilidad de los resultados en función del tamaño de bloque o celda utilizado para generar el terreno provisional inicial.



	Celda: 100x100 m	Celda: 50x50 m	Celda: 25x25 m	Celda: 10x10 m
Error de comisión (%) (Umbral Iz=0,2 m)	4,55	4,66	5,56	14,71
Error de comisión (%) (Umbral Iz=0,4 m)	5,23	5,33	6,21	15,22
Error de comisión (%) (Umbral Iz=0,6 m)	6,18	6,27	7,13	15,93
Error de comisión (%) (Umbral Iz=1 m)	8,27	8,43	9,29	17,56
Error de comisión (%) (Umbral Iz=2 m)	17,17	17,23	18,16	23,92

GRAFICO 3:

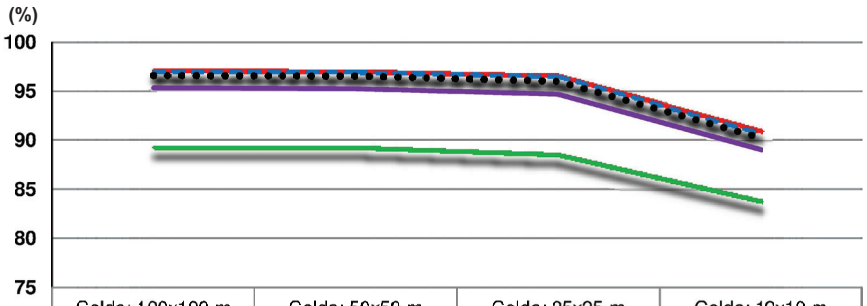
Muestra la media entre los errores de omisión y comisión obtenidos tras ejecutar el algoritmo A para los cinco umbrales en desnivel utilizados. Además el gráfico muestra la variabilidad de los resultados en función del tamaño de bloque o celda utilizado para generar el terreno provisional inicial.



	Celda: 100x100 m	Celda: 50x50 m	Celda: 25x25 m	Celda: 10x10 m
Media de errores (%) (Umbral Iz=0,2 m)	2,78	2,85	3,24	7,56
Media de errores (%) (Umbral Iz=0,4 m)	2,80	2,85	3,21	7,68
Media de errores (%) (Umbral Iz=0,6 m)	3,12	3,17	3,61	8,02
Media de errores (%) (Umbral Iz=1 m)	4,16	4,24	4,68	8,83
Media de errores (%) (Umbral Iz=2 m)	8,61	8,64	9,11	12,00

GRAFICO 4:

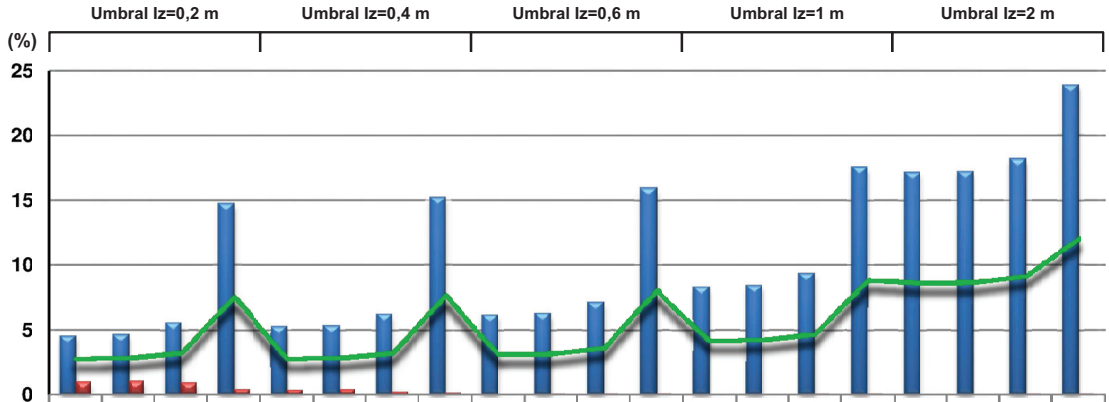
Muestra la exactitud global de la clase terreno obtenidos tras ejecutar el algoritmo A para los cinco umbrales en desnivel utilizados. Además el gráfico muestra la variabilidad de los resultados en función del tamaño de bloque o celda utilizado para generar el terreno provisional inicial.



	Celda: 100x100 m	Celda: 50x50 m	Celda: 25x25 m	Celda: 10x10 m
Exactitud global (%) (Umbral Iz=0,2 m)	97,06	96,98	96,53	90,96
Exactitud global (%) (Umbral Iz=0,4 m)	96,99	96,92	96,49	90,71
Exactitud global (%) (Umbral Iz=0,6 m)	96,58	96,53	96,01	90,21
Exactitud global (%) (Umbral Iz=1 m)	95,34	95,24	94,70	89,01
Exactitud global (%) (Umbral Iz=2 m)	89,32	89,30	88,56	83,80

GRAFICO 5:

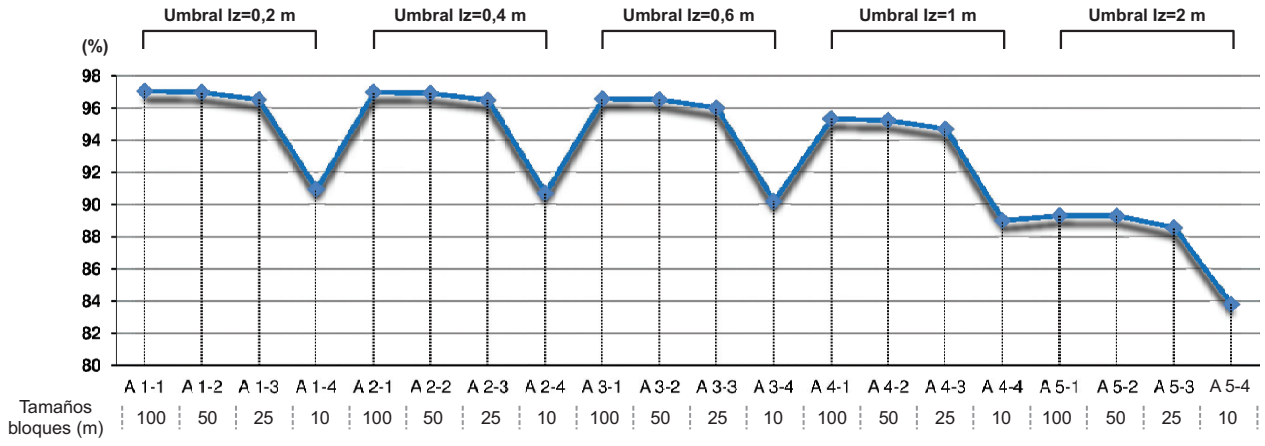
Muestra la media entre los errores de omisión y comisión obtenidos tras ejecutar el algoritmo A para los cinco umbrales en desnivel utilizados. Además el gráfico muestra la variabilidad de los resultados en función del tamaño de bloque o celda utilizado para generar el terreno provisional inicial.



	A 1-1	A 1-2	A 1-3	A 1-4	A 2-1	A 2-2	A 2-3	A 2-4	A 3-1	A 3-2	A 3-3	A 3-4	A 4-1	A 4-2	A 4-3	A 4-4	A 5-1	A 5-2	A 5-3	A 5-4
Error de comisión (%)	4,55	4,66	5,56	14,71	5,23	5,33	6,21	15,22	6,18	6,27	7,13	15,93	8,27	8,43	9,29	17,56	17,17	17,23	18,16	23,92
Error de omisión (%)	1,00	1,03	0,91	0,40	0,36	0,37	0,21	0,13	0,05	0,06	0,08	0,10	0,04	0,05	0,06	0,09	0,04	0,05	0,06	0,08
Media de errores (%)	2,78	2,85	3,24	7,56	2,80	2,85	3,21	7,68	3,12	3,17	3,61	8,02	4,16	4,24	4,68	8,83	8,31	8,64	9,11	12,00
Tamaños bloques (m)	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10

GRAFICO 6:

Muestra la exactitud global para la totalidad de resultados obtenidos tras ejecutar el algoritmo A de clasificación del terreno. En el gráfico puede apreciarse la variabilidad de la precisión de los resultados en función del umbral en desnivel y su dependencia al tamaño de bloque utilizado para definir el terreno provisional inicial.



ALGORITMO: DETECCIÓN DEL TERRENO MEDIANTE METODO DE DENSIFICACIÓN QUE COMBINA INTERPOLACIÓN Y SEGMENTACIÓN Y ANALIZA CARACTERÍSTICAS DE LAS AGRUPACIONES (ALGORITMO B)

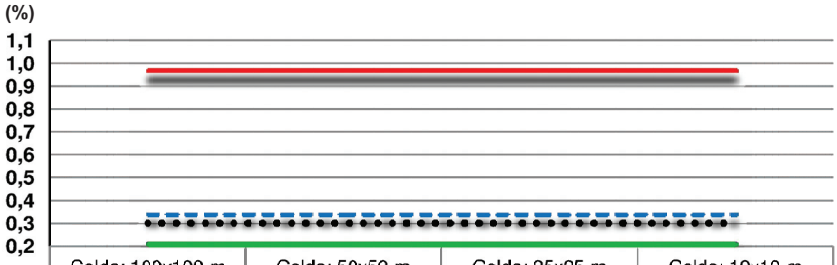
ZONA TEST: Collado de Villalba

DESCRIPCIÓN:

Algoritmo de densificación progresiva para detectar la clase terreno. La superficie de referencia se obtiene mediante interpolación de distantancia inversa ponderada. El algoritmo de densificación se combina con un proceso de segmentación. Las agrupaciones obtenidas tras la segmentación son clasificadas como terreno en un proceso iterativo en función de su distancia a la superficie de referencia. El proceso lleva asociado un análisis de las características de agrupaciones colindantes con el fin de reducir al máximo errores en la clasificación y conseguir además una mayor automatización del proceso.

GRAFICO 1:

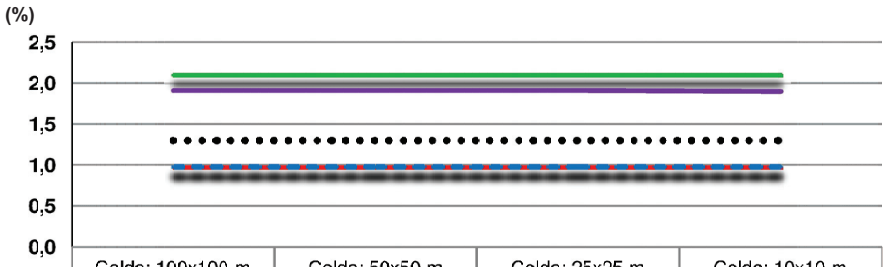
Muestra los errores de omisión obtenidos tras ejecutar el algoritmo B para los cinco umbrales en desnivel utilizados. Además el gráfico muestra la variabilidad de los resultados en función del tamaño de bloque o celda utilizado para generar el terreno provisional inicial.



	Celda: 100x100 m	Celda: 50x50 m	Celda: 25x25 m	Celda: 10x10 m
Error de omisión (%) (Umbral Iz=0,2 m)	0,97	0,97	0,97	0,97
Error de omisión (%) (Umbral Iz=0,4 m)	0,34	0,34	0,34	0,34
Error de omisión (%) (Umbral Iz=0,6 m)	0,30	0,30	0,30	0,30
Error de omisión (%) (Umbral Iz=1 m)	0,21	0,21	0,21	0,21
Error de omisión (%) (Umbral Iz=2 m)	0,21	0,21	0,21	0,21

GRAFICO 2:

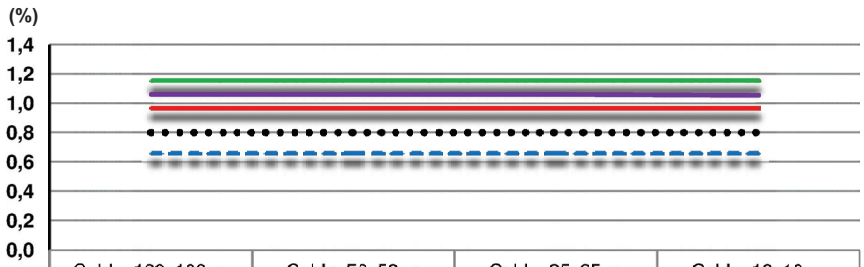
Muestra los errores de comisión obtenidos tras ejecutar el algoritmo B para los cinco umbrales en desnivel utilizados. Además el gráfico muestra la variabilidad de los resultados en función del tamaño de bloque o celda utilizado para generar el terreno provisional inicial.



	Celda: 100x100 m	Celda: 50x50 m	Celda: 25x25 m	Celda: 10x10 m
Error de comisión (%) (Umbral Iz=0,2 m)	0,97	0,97	0,97	0,97
Error de comisión (%) (Umbral Iz=0,4 m)	0,98	0,98	0,98	0,98
Error de comisión (%) (Umbral Iz=0,6 m)	1,30	1,30	1,30	1,30
Error de comisión (%) (Umbral Iz=1 m)	1,91	1,91	1,91	1,90
Error de comisión (%) (Umbral Iz=2 m)	2,10	2,10	2,10	2,10

GRAFICO 3:

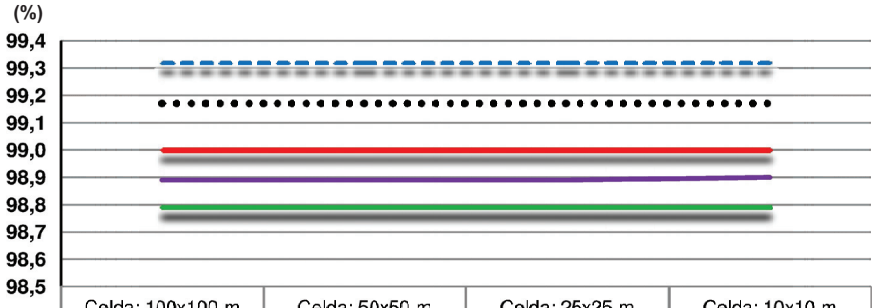
Muestra la media entre los errores de omisión y comisión obtenidos tras ejecutar el algoritmo B para los cinco umbrales en desnivel utilizados. Además el gráfico muestra la variabilidad de los resultados en función del tamaño de bloque o celda utilizado para generar el terreno provisional inicial.



	Celda: 100x100 m	Celda: 50x50 m	Celda: 25x25 m	Celda: 10x10 m
Media de errores (%) (Umbral Iz=0,2 m)	0,97	0,97	0,97	0,97
Media de errores (%) (Umbral Iz=0,4 m)	0,66	0,66	0,66	0,66
Media de errores (%) (Umbral Iz=0,6 m)	0,80	0,80	0,80	0,80
Media de errores (%) (Umbral Iz=1 m)	1,06	1,06	1,06	1,06
Media de errores (%) (Umbral Iz=2 m)	1,16	1,16	1,16	1,16

GRAFICO 4:

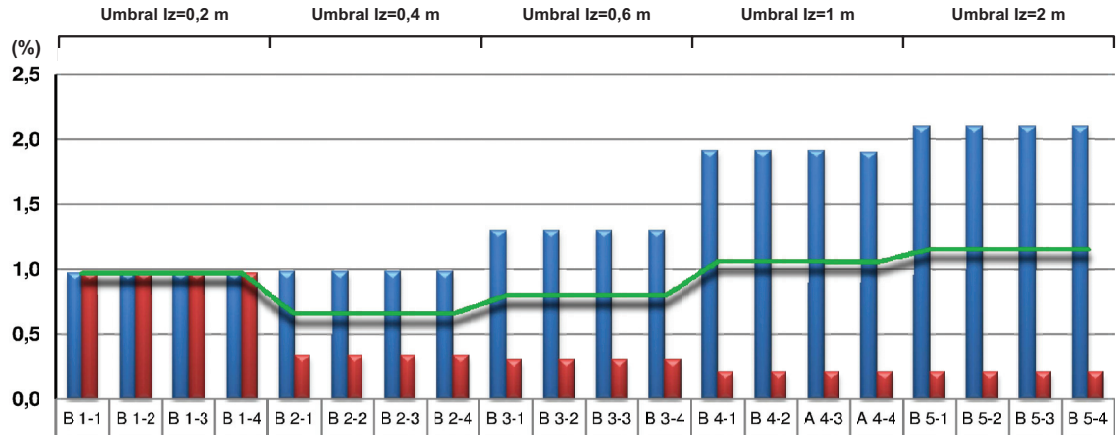
Muestra la exactitud global de la clase terreno obtenidos tras ejecutar el algoritmo B para los cinco umbrales en desnivel utilizados. Además el gráfico muestra la variabilidad de los resultados en función del tamaño de bloque o celda utilizado para generar el terreno provisional inicial.



	Celda: 100x100 m	Celda: 50x50 m	Celda: 25x25 m	Celda: 10x10 m
Exactitud global (%) (Umbral Iz=0,2 m)	99,00	99,00	99,00	99,00
Exactitud global (%) (Umbral Iz=0,4 m)	99,32	99,32	99,32	99,32
Exactitud global (%) (Umbral Iz=0,6 m)	99,17	99,17	99,17	99,17
Exactitud global (%) (Umbral Iz=1 m)	98,89	98,89	98,89	98,90
Exactitud global (%) (Umbral Iz=2 m)	98,79	98,79	98,79	98,79

GRAFICO 5:

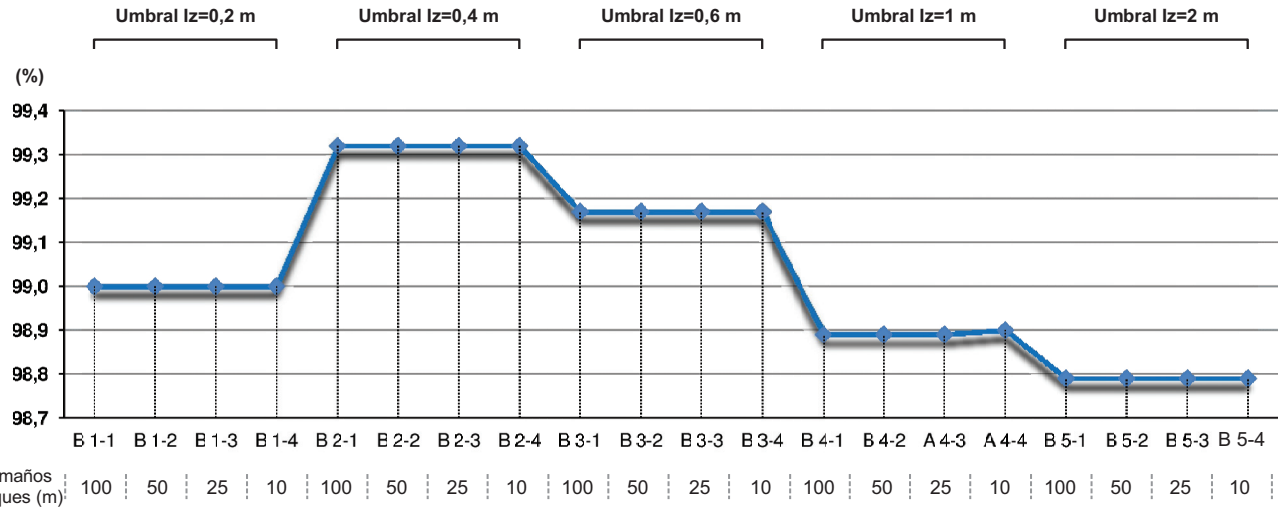
Muestra la media entre los errores de omisión y comisión obtenidos tras ejecutar el algoritmo B para los cinco umbrales en desnivel utilizados. Además el gráfico muestra la consistencia del algoritmo B, ya que la precisión de los resultados se mantiene constante al variar el tamaño del bloque o celda utilizado para generar el terreno provisional inicial.



	B 1-1	B 1-2	B 1-3	B 1-4	B 2-1	B 2-2	B 2-3	B 2-4	B 3-1	B 3-2	B 3-3	B 3-4	B 4-1	B 4-2	A 4-3	A 4-4	B 5-1	B 5-2	B 5-3	B 5-4
Error de comisión (%)	0,97	0,97	0,97	0,97	0,98	0,98	0,98	0,98	1,30	1,30	1,30	1,30	1,91	1,91	1,91	1,90	2,10	2,10	2,10	2,10
Error de omisión (%)	0,97	0,97	0,97	0,97	0,34	0,34	0,34	0,34	0,30	0,30	0,30	0,30	0,21	0,21	0,21	0,21	0,21	0,21	0,21	0,21
Media de errores (%)	0,97	0,97	0,97	0,97	0,66	0,66	0,66	0,66	0,80	0,80	0,80	0,80	1,06	1,06	1,06	1,06	1,16	1,16	1,16	1,16
Tamaños bloques (m)	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10

GRAFICO 6:

Muestra la exactitud global para la totalidad de resultados obtenidos tras ejecutar el algoritmo B de clasificación del terreno. Además el gráfico muestra la consistencia del algoritmo B, ya que la exactitud global se mantiene constante al variar el tamaño del bloque o celda utilizado para generar el terreno provisional inicial.



ALGORITMO: DETECCIÓN DEL TERRENO MEDIANTE METODO DE DENSIFICACIÓN QUE COMBINA INTERPOLACIÓN Y SEGMENTACIÓN. COMPARACIÓN ENTRE LOS RESULTADOS DEL ALGORITMO A Y B.

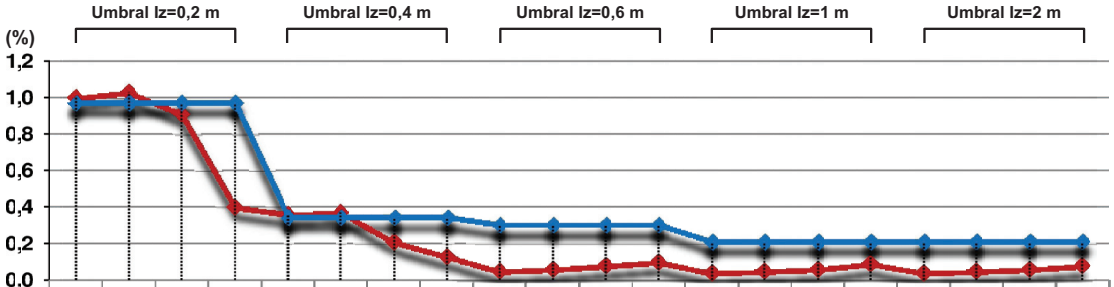
ZONA TEST: Collado de Villalba

DESCRIPCIÓN:

Comparación de los resultados obtenidos tras ejecutar los dos algoritmos para detectar el terreno (algoritmo A y B). Ambos algoritmos utilizan un proceso de densificación progresiva combinado con un proceso de segmentación. El algoritmo A no realiza análisis alguno de las características de agrupaciones colindantes para determinar la clase terreno, mientras que el algoritmo B analiza las características de agrupaciones colindantes con el fin de reducir al máximo errores en la clasificación y conseguir además una mayor automatización del proceso.

GRAFICO 1:

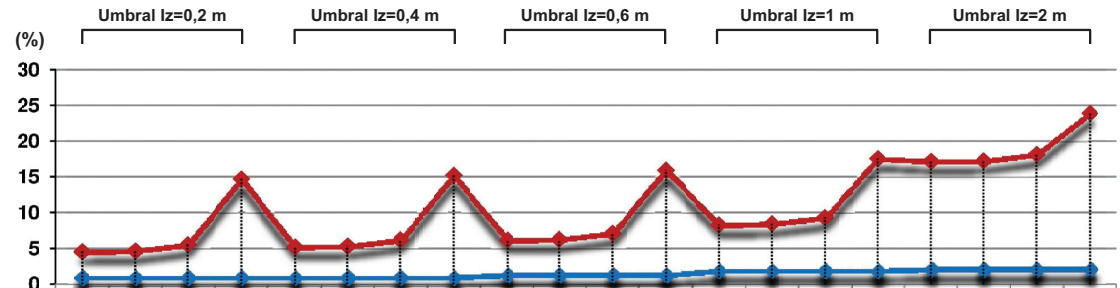
Muestra los errores de omisión obtenidos tras ejecutar el algoritmo A y el algoritmo B.



	1-1	1-2	1-3	1-4	2-1	2-2	2-3	2-4	3-1	3-2	3-3	3-4	4-1	4-2	4-3	4-4	5-1	5-2	5-3	5-4
Error omsión algoritmo A	1,00	1,03	0,91	0,40	0,36	0,37	0,21	0,13	0,05	0,06	0,08	0,10	0,04	0,05	0,06	0,09	0,04	0,05	0,06	0,08
Error omisión algoritmo B	0,97	0,97	0,97	0,97	0,34	0,34	0,34	0,34	0,30	0,30	0,30	0,30	0,21	0,21	0,21	0,21	0,21	0,21	0,21	0,21
Tamaños bloques (m)	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10

GRAFICO 2:

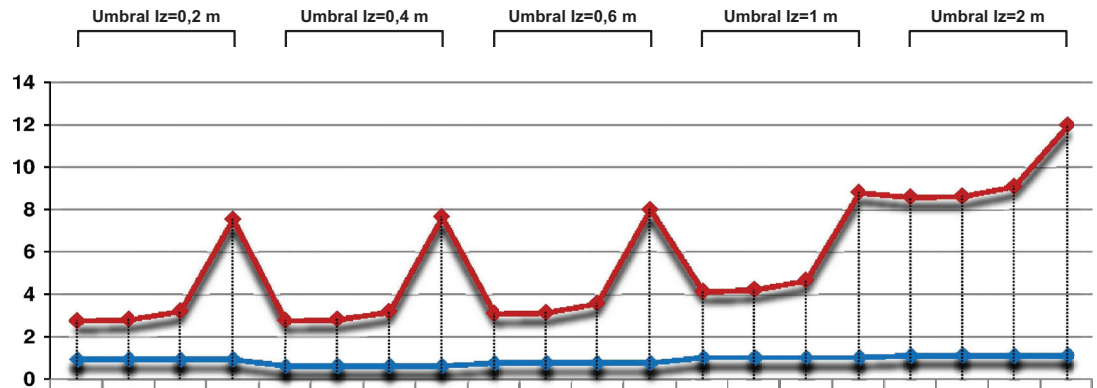
Muestra los errores de comisión obtenidos tras ejecutar el algoritmo A y el algoritmo B.



Algoritmo	1-1	1-2	1-3	1-4	2-1	2-2	2-3	2-4	3-1	3-2	3-3	3-4	4-1	4-2	4-3	4-4	5-1	5-2	5-3	5-4
Error comsión algoritmo A	4,55	4,66	5,56	14,71	5,23	5,33	6,21	15,22	6,18	6,27	7,13	15,93	8,27	8,43	9,29	17,56	17,17	17,23	18,16	23,92
Error comisión algoritmo B	0,97	0,97	0,97	0,97	0,98	0,98	0,98	0,98	1,30	1,30	1,30	1,30	1,91	1,91	1,91	1,90	2,10	2,10	2,10	2,10
Tamaños bloques (m)	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10

GRAFICO 3:

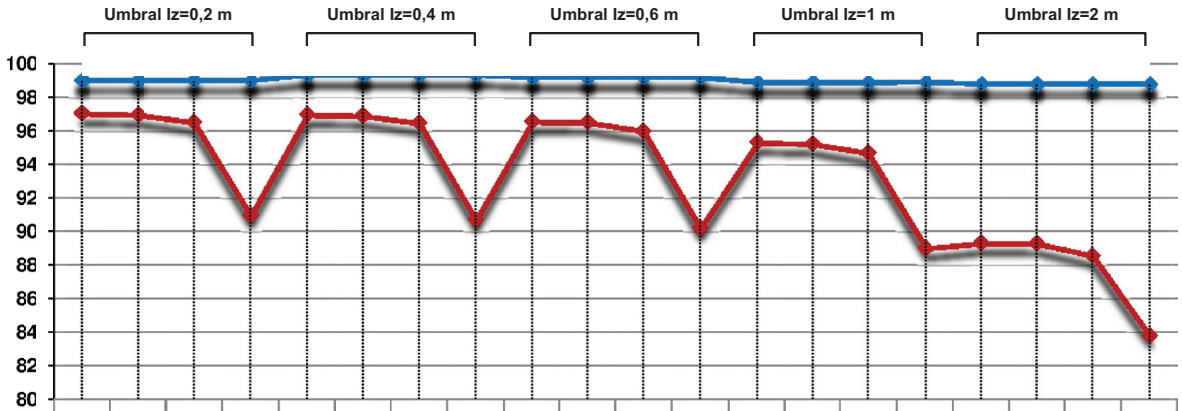
Muestra la media entre los errores de comisión y omisiónn obtenidos tras ejecutar el algoritmo A y el algoritmo B.



Algoritmo	1-1	1-2	1-3	1-4	2-1	2-2	2-3	2-4	3-1	3-2	3-3	3-4	4-1	4-2	4-3	4-4	5-1	5-2	5-3	5-4
Media de errores algoritmo A	2,78	2,85	3,24	7,56	2,80	2,85	3,21	7,68	3,12	3,17	3,61	8,02	4,16	4,24	4,68	8,83	8,61	8,64	9,11	12,00
Media de errores algoritmo B	0,97	0,97	0,97	0,97	0,66	0,66	0,66	0,66	0,80	0,80	0,80	0,80	1,06	1,06	1,06	1,06	1,16	1,16	1,16	1,16
Tamaños bloques (m)	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10

GRAFICO 4:

Muestra la exactitud global obtenida tras ejecutar el algoritmo A y el algoritmo B.



Algoritmo	1-1	1-2	1-3	1-4	2-1	2-2	2-3	2-4	3-1	3-2	3-3	3-4	4-1	4-2	4-3	4-4	5-1	5-2	5-3	5-4
Exactitud global A	97,06	96,98	96,53	90,96	96,99	96,92	96,49	90,71	96,58	96,53	96,01	90,21	95,34	95,24	94,70	89,01	89,32	89,30	88,56	83,80
Exactitud global B	99,00	99,00	99,00	99,00	99,32	99,32	99,32	99,32	99,17	99,17	99,17	99,17	98,89	98,89	98,89	98,90	98,79	98,79	98,79	98,79
Tamaños bloques (m)	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10	100	50	25	10

DESCRIPCIÓN DE PARÁMETROS UMBRALES UTILIZADOS

ALGORITMO A: CLASIFICACIÓN DE PUENTES MEDIANTE DETECCIÓN DE BORDES Y CLASIFICADOR ANGULAR POR SECTORES. PROCESO SIMPLE SIN ITERACIONES (ALGORITMO A)

DESCRIPCIÓN:

Tras detectar los bordes de los puentes mediante la localización de cambios bruscos en altitud dentro de la clase terreno (pendientes verticales), los puntos de puentes son clasificados mediante un algoritmo de clasificación angular por sectores. Se trata de un proceso sin iteraciones (Algoritmo A)

PARÁMETROS UMBRALES FIJOS PARA ALGORITMO A:

DETECCIÓN DE BORDES:

VECINDARIO CILÍNDRICO PARA DETECTAR CAMBIOS BRUSCOS EN DESNIVEL (DETECCIÓN DE BORDES): 2 m

UMBRAL EN PORCENTAJE DE PUNTOS QUE CUMPLEN EL IZ UMBRAL PARA DETECTAR BORDES: 20 %

UMBRAL EN DESNIVEL: Se ejecuta el algoritmo cinco veces con diferentes parámetros umbrales:

A 1: 1 m A 2: 2 m A 3: 3 m A 4: 3,5 m A 5: 4 m

DETECCIÓN DE PUENTES A PARTIR DE LOS BORDES DETECTADOS:

RADIO DE SECTORES: 18 m

RADIO DE SEMISECTORES INTERIORES: 6 m

IZ UMBRAL CON RESPECTO A BORDES PARA CONSIDERAR QUE UN PUNTO PERTENECE A UN PUENTE: 0,5 m

UMBRAL EN % DE PUNTOS QUE CUMPLEN UMBRAL IZ EN SEMISECTORES INTERIORES: 95 %

AMPLITUD DE LOS SECTORES DEL CLASIFICADOR DIRECCIONAL : 10 g

ALGORITMO B: CLASIFICACIÓN DE PUENTES MEDIANTE DETECCIÓN DE BORDES Y CLASIFICADOR ANGULAR POR SECTORES. ALGORITMO MULTIPROCESO CON PARÁMETROS ADAPTATIVOS (ALGORITMO B)

DESCRIPCIÓN:

Tras detectar los bordes de los puentes mediante la localización de cambios bruscos en la altitud dentro de la clase terreno (pendientes verticales), los puntos de puentes son clasificados mediante un algoritmo de clasificación angular por sectores. Se trata de un doble proceso donde tras detectar el grueso de los puentes mediante parámetros umbrales restrictivos (algoritmo A), se aplica un proceso iterativo con un parámetro umbral en IZ más permisivo en las proximidades de los puentes previamente detectados, consiguiendo así reducir notablemente errores de omisión en la clasificación.

PARÁMETROS UMBRALES FIJOS PARA ALGORITMO B:

El algoritmo B consiste en ejecutar dos veces concatenadas el algoritmo A adaptando el parámetro umbral en desnivel, por lo que el resto de parámetros no presentan variaciones. Los parámetros nuevos parámetros utilizados son los siguientes:

DETECCIÓN DE ÁREA DE INFLUENCIA PARA EJECUTAR EL SEGUNDO PROCESO:

VECINDARIO CILÍNDRICO PARA BUSCAR NUEVOS PUNTOS DE PUENTES Y FUSIONARLOS AL RESULTADO OBTENIDO TRAS EJECUTAR EL PRIMER PROCESO: 8 m

PARÁMETROS PARA DETECTAR BORDES COLINDANTES A ZONAS DE SOMBRA:

VECINDARIO CILÍNDRICO PARA DETECTAR BORDES COLINDANTES A ZONAS DE SOMBRA: 4 m

AMPLITUD DE LOS SECTORES DEL CLASIFICADOR ANGULAR PARA DETECTAR BORDES COLINDANTES A ZONAS DE SOMBRA: 50 g

PARÁMETROS UMBRALES VARIABLES:

Se ejecuta el algoritmo B un total de 6 veces modificando el parámetro umbral en desnivel para los dos procesos de detección de bordes utilizados

B 3-1:	PROCESO 1: IZ UMBRAL= 3 m PROCESO 2: IZ UMBRAL= 1 m	**B 3-2:**	PROCESO 1: IZ UMBRAL= 3 m PROCESO 2: IZ UMBRAL= 2 m	**B 4-1:**	PROCESO 1: IZ UMBRAL= 3,5 m PROCESO 2: IZ UMBRAL= 1 m
B 4-2:	PROCESO 1: IZ UMBRAL= 3,5 m PROCESO 2: IZ UMBRAL= 2 m	**B 5-1:**	PROCESO 1: IZ UMBRAL= 4 m PROCESO 2: IZ UMBRAL= 1 m	**B 5-2:**	PROCESO 1: IZ UMBRAL= 4 m PROCESO 2: IZ UMBRAL= 2 m

ALGORITMO: CLASIFICACIÓN DE PUENTES MEDIANTE DETECCIÓN DE BORDES Y CLASIFICADOR ANGULAR POR SECTORES. PROCESO SIMPLE SIN ITERACIONES (ALGORITMO A)

ZONA TEST: Collado de Villalba

DESCRIPCIÓN:

Tras detectar los bordes de los puentes mediante la localización de cambios bruscos en altitud dentro de la clase terreno (pendientes verticales), los puntos de puentes son clasificados mediante un algoritmo de clasificación angular por sectores. Se trata de un proceso sin iteraciones con parámetros umbrales fijos (Algoritmo A)

PARÁMETROS UMBRALES FIJOS PARA ALGORITMO A:

DETECCIÓN DE BORDES:

VECINDARIO CILÍNDRICO PARA DETECTAR CAMBIOS BRUSCOS EN DESNIVEL (DETECCIÓN DE BORDES): 2 m

UMBRAL EN PORCENTAJE DE PUNTOS QUE CUMPLEN EL IZ UMBRAL PARA DETECTAR BORDES: 20 %

DETECCIÓN DE PUENTES A PARTIR DE LOS BORDES DETECTADOS:

RADIO DE SECTORES: 18 m

RADIO DE SEMISECTORES INTERIORES: 6 m

IZ UMBRAL CON RESPECTO A BORDES PARA CONSIDERAR QUE UN PUNTO PERTENECE A UN PUENTE: 0,5 m

UMBRAL EN % DE PUNTOS QUE CUMPLEN UMBRAL IZ EN SEMISECTORES INTERIORES: 95 %

AMPLITUD DE LOS SECTORES DEL CLASIFICADOR ANGULAR MEDIANTE SECTORES: 10 g

ALGORITMO: A-1

UMBRAL EN IZ PARA DETECTAR BORDES: 1 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación					
	Terreno	**Puentes**	**Otras entidades**	**Total**	**Ex. Prod.**	**Error. Om.**
Terreno	**1358330**	90001	4874	1453205	93,47 %	6,53 %
Puentes	264	**5781**	12	6057	95,44 %	4,56 %
Otras entidades	8196	127	**1357137**	1365460	99,39 %	0,61 %
Total	1366790	95909	1362023	**2824722**	Exactitud global: 96,34 %	
Ex. Usuario	99,38 %	6,03 %	99,64 %			
Error. Com	0,62 %	93,97 %	0,36 %			

ALGORITMO: A-2

UMBRAL EN IZ PARA DETECTAR BORDES: 2 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación					
	Terreno	**Puentes**	**Otras entidades**	**Total**	**Ex. Prod.**	**Error. Om.**
Terreno	**1439736**	8595	4874	1453205	99,07 %	0,93 %
Puentes	425	**5620**	12	6057	92,79 %	7,21 %
Otras entidades	8313	10	**1357137**	1365460	99,39 %	0,61 %
Total	1448474	14225	1362023	**2824722**	Exactitud global: 99,21 %	
Ex. Usuario	99,40 %	39,51 %	99,64 %			
Error. Com	0,60 %	60,49 %	0,36 %			

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

ALGORITMO: A-3

UMBRAL EN IZ PARA DETECTAR BORDES: 3 m

MATRIZ DE CONFUSIÓN						
Clases de referencia	**Categorías deducidas de la clasificación**					
	Terreno	**Puentes**	**Otras entidades**	**Total**	**Ex. Prod.**	**Error. Om.**
Terreno	**1447997**	334	4874	1453205	99,64 %	0,36 %
Puentes	967	**5078**	12	6057	83,84 %	16,16 %
Otras entidades	8323	0	**1357137**	1365460	99,39 %	0,61 %
Total	1457287	5412	1362023	**2824722**	Exactitud global: 99,49 %	
Ex. Usuario	99,36 %	93,83 %	99,64 %			
Error. Com	0,64 %	6,17 %	0,36 %			

ALGORITMO: A-4

UMBRAL EN IZ PARA DETECTAR BORDES: 3,5 m

MATRIZ DE CONFUSIÓN						
Clases de referencia	**Categorías deducidas de la clasificación**					
	Terreno	**Puentes**	**Otras entidades**	**Total**	**Ex. Prod.**	**Error. Om.**
Terreno	**1448331**	0	4874	1453205	99,66 %	0,34 %
Puentes	1451	**4594**	12	6057	75,85 %	24,15 %
Otras entidades	8323	0	**1357137**	1365460	99,39 %	0,61 %
Total	1458105	4594	1362023	**2824722**	Exactitud global: 99,48 %	
Ex. Usuario	99,33 %	100 %	99,64 %			
Error. Com	0,67 %	0 %	0,36 %			

ALGORITMO: A-5

UMBRAL EN IZ PARA DETECTAR BORDES: 4 m

MATRIZ DE CONFUSIÓN						
Clases de referencia	**Categorías deducidas de la clasificación**					
	Terreno	**Puentes**	**Otras entidades**	**Total**	**Ex. Prod.**	**Error. Om.**
Terreno	**1448331**	0	4874	1453205	99,66 %	0,34 %
Puentes	2525	**3520**	12	6057	58,11 %	41,89 %
Otras entidades	8323	0	**1357137**	1365460	99,39 %	0,61 %
Total	1459179	3520	1362023	**2824722**	Exactitud global: 99,44 %	
Ex. Usuario	99,26 %	100 %	99,64 %			
Error. Com	0,74 %	0 %	0,36 %			

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

ALGORITMO: CLASIFICACIÓN DE PUENTES MEDIANTE DETECCIÓN DE BORDES Y CLASIFICADOR ANGULAR POR SECTORES. MULTIPROCESO CON PARÁMETROS UMBRALES ADAPTATIVOS (ALGORITMO B)

ZONA TEST: Collado de Villalba

DESCRIPCIÓN:

Tras detectar los bordes de los puentes mediante la localización de cambios bruscos en la altitud dentro de la clase terreno (pendientes verticales), los puntos de puentes son clasificados mediante un algoritmo de clasificación angular por sectores. Se trata de un doble proceso donde tras detectar el grueso de los puentes mediante parámetros umbrales restrictivos (algoritmo A), se aplica un proceso iterativo con un parámetro umbral en IZ más permisivo en las proximidades de los puentes previamente detectados, consiguiendo así reducir notablemente errores de omisión en la clasificación.

PARÁMETROS UMBRALES FIJOS PARA ALGORITMO B:

DETECCIÓN DE BORDES:

VECINDARIO CILÍNDRICO PARA DETECTAR BORDES COLINDANTES A ZONAS DE SOMBRA: 4 m

VECINDARIO CILÍNDRICO PARA BUSCAR NUEVOS PUNTOS DE PUENTES Y FUSIONARLOS AL RESULTADO OBTENIDO TRAS EJECUTAR ALGORITMO A: 8 m

AMPLITUD DE LOS SECTORES DEL CLASIFICADOR ANGULAR PARA DETECTAR BORDES COLINDANTES A ZONAS DE SOMBRA: 20 g

UMBRAL EN PORCENTAJE DE PUNTOS QUE CUMPLEN EL IZ UMBRAL PARA DETECTAR BORDES: 20 %

DETECCIÓN DE PUENTES DEFINITIVO:

RADIO DE SECTORES: 18 m

RADIO DE SEMISECTORES INTERIORES: 6 m

IZ UMBRAL CON RESPECTO A BORDES PARA CONSIDERAR QUE UN PUNTO PERTENECE A UN PUENTE: 0,5 m

UMBRAL EN % DE PUNTOS QUE CUMPLEN UMBRAL IZ EN SEMISECTORES INTERIORES: 95 %

AMPLITUD DE LOS SECTORES DEL CLASIFICADOR ANGULAR MEDIANTE SECTORES: 10 g

ALGORITMO: B 3-1

PARÁMETROS UMBRALES VARIABLES PARA ALGORITMO B:

PROCESO 1. DETECCIÓN DEL GRUESO DE PUENTES: UMBRALES ASOCIADOS AL ALGORITMO A3 (IZ UMBRAL= 3 m)

PROCESO 2. DETECCIÓN DEPUENTES DEFINITIVA: IZ UMBRAL EN LA DETECCIÓN DE BORDES: 1 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación					
	Terreno	**Puentes**	**Otras entidades**	**Total**	**Ex. Prod.**	**Error. Om.**
Terreno	**1443867**	4464	4874	1453205	99,36 %	0,64 %
Puentes	74	**5971**	12	6057	98,58 %	1,42 %
Otras entidades	8316	7	**1357137**	1365460	99,39 %	0,61 %
Total	1452257	10442	1362023	**2824722**	Exactitud global: 99,37 %	
Ex. Usuario	99,42 %	57,18 %	99,64 %			
Error. Com	0,58 %	42,82 %	0,36 %			

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

ALGORITMO: B 3-2

PARÁMETROS UMBRALES VARIABLES PARA ALGORITMO B:

PROCESO 1. DETECCIÓN DEL GRUESO DE PUENTES: UMBRALES ASOCIADOS AL ALGORITMO A3 (IZ UMBRAL= 3 m)

PROCESO 2. DETECCIÓN DEPUENTES DEFINITIVA: IZ UMBRAL EN LA DETECCIÓN DE BORDES: 2 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación					
	Terreno	Puentes	Otras entidades	Total	Ex. Prod.	Error. Om.
Terreno	**1446401**	1930	4874	1453205	99,53 %	0,47 %
Puentes	120	**5925**	12	6057	97,82 %	2,18 %
Otras entidades	8316	7	**1357137**	1365460	99,39 %	0,61 %
Total	1454837	7862	1362023	**2824722**	Exactitud global: 99,46 %	
Ex. Usuario	99,42 %	75,36 %	99,64 %			
Error. Com	0,58 %	24,64 %	0,36 %			

ALGORITMO: B 4-1

PARÁMETROS UMBRALES VARIABLES PARA ALGORITMO B:

PROCESO 1. DETECCIÓN DEL GRUESO DE PUENTES: UMBRALES ASOCIADOS AL ALGORITMO A3 (IZ UMBRAL= 3,5 m)

PROCESO 2. DETECCIÓN DEPUENTES DEFINITIVA: IZ UMBRAL EN LA DETECCIÓN DE BORDES: 1 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación					
	Terreno	Puentes	Otras entidades	Total	Ex. Prod.	Error. Om.
Terreno	**1445737**	2594	4874	1453205	99,49 %	0,51 %
Puentes	84	**5961**	12	6057	98,42 %	1,58 %
Otras entidades	8323	0	**1357137**	1365460	99,39 %	0,61 %
Total	1454144	8555	1362023	**2824722**	Exactitud global: 99,44 %	
Ex. Usuario	99,42 %	69,68 %	99,64 %			
Error. Com	0,58 %	30,32 %	0,36 %			

ALGORITMO: B 4-2

PARÁMETROS UMBRALES VARIABLES PARA ALGORITMO B:

PROCESO 1. DETECCIÓN DEL GRUESO DE PUENTES: UMBRALES ASOCIADOS AL ALGORITMO A3 (IZ UMBRAL= 3,5 m)

PROCESO 2. DETECCIÓN DEPUENTES DEFINITIVA: IZ UMBRAL EN LA DETECCIÓN DE BORDES: 2 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación					
	Terreno	Puentes	Otras entidades	Total	Ex. Prod.	Error. Om.
Terreno	**1447977**	354	4874	1453205	99,64 %	0,36 %
Puentes	130	**5915**	12	6057	97,66 %	2,34 %
Otras entidades	8323	0	**1357137**	1365460	99,39 %	0,61 %
Total	1456430	6269	1362023	**2824722**	Exactitud global: 99,52 %	
Ex. Usuario	99,42 %	94,35 %	99,64 %			
Error. Com	0,58 %	5,65 %	0,36 %			

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

ALGORITMO: B 5-1

PARÁMETROS UMBRALES VARIABLES PARA ALGORITMO B:

PROCESO 1. DETECCIÓN DEL GRUESO DE PUENTES: UMBRALES ASOCIADOS AL ALGORITMO A3 (IZ UMBRAL= 4 m)

PROCESO 2. DETECCIÓN DEPUENTES DEFINITIVA: IZ UMBRAL EN LA DETECCIÓN DE BORDES: 1 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación					
	Terreno	Puentes	Otras entidades	Total	Ex. Prod.	Error. Om.
Terreno	**1446372**	1959	4874	1453205	99,53 %	0,47 %
Puentes	1399	**4646**	12	6057	76,70 %	23,30 %
Otras entidades	8323	0	**1357137**	1365460	99,39 %	0,61 %
Total	1456094	6605	1362023	**2824722**	Exactitud global: 99,41 %	
Ex. Usuario	99,33 %	70,34 %	99,64 %			
Error. Com	0,67 %	29,66 %	0,36 %			

ALGORITMO: B 5-2

PARÁMETROS UMBRALES VARIABLES PARA ALGORITMO B:

PROCESO 1. DETECCIÓN DEL GRUESO DE PUENTES: UMBRALES ASOCIADOS AL ALGORITMO A3 (IZ UMBRAL= 4 m)

PROCESO 2. DETECCIÓN DEPUENTES DEFINITIVA: IZ UMBRAL EN LA DETECCIÓN DE BORDES: 2 m

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación					
	Terreno	Puentes	Otras entidades	Total	Ex. Prod.	Error. Om.
Terreno	**1447999**	332	4874	1453205	99,64 %	0,36 %
Puentes	1477	**4568**	12	6057	75,42 %	24,58 %
Otras entidades	8323	0	**1357137**	1365460	99,39 %	0,61 %
Total	1457799	4900	1362023	**2824722**	Exactitud global: 99,47 %	
Ex. Usuario	99,33 %	93,22 %	99,64 %			
Error. Com	0,67 %	6,78 %	0,36 %			

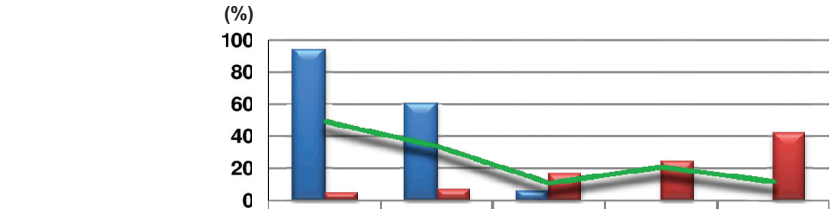
Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

ANÁLISIS DE RESULTADOS (ANÁLISIS GRÁFICO)

ALGORITMO: CLASIFICACIÓN DE PUENTES MEDIANTE DETECCIÓN DE BORDES Y CLASIFICADOR ANGULAR POR SECTORES. PROCESO SIMPLE SIN ITERACIONES (ALGORITMO A)

GRAFICO 1:

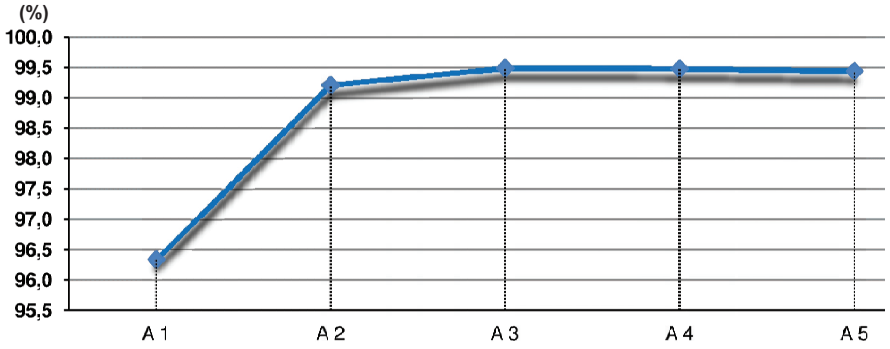
Muestra los errores de omisión, comisión y su media, obtenidos tras ejecutar el algoritmo A para los cinco parámetros umbrales utilizados.



	A 1	A 2	A 3	A 4	A 5
Error de comisión (%)	93,97	60,49	6,17	0	0
Error de omisión (%)	4,56	7,21	16,16	24,15	41,89
Media de errores (%)	49,27	33,85	11,17	20,95	12,08

GRAFICO 2:

Muestra la exactitud global de la clasificación para los resultados obtenidos tras ejecutar el algoritmo A (Proceso simple).



PARAMETROS UMBRALES EN DETECCIÓN DE BORDES:

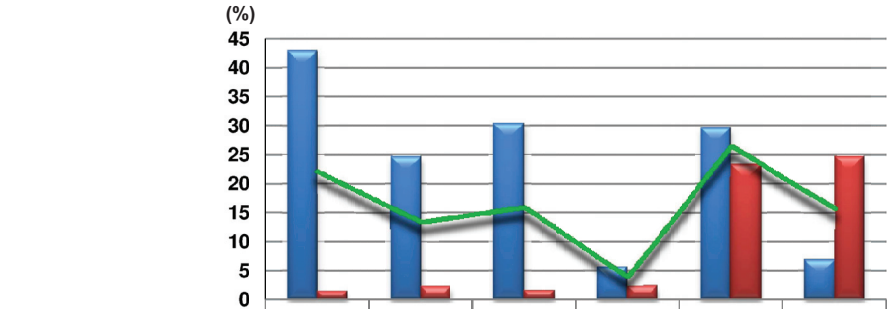
ALGORITMO	*IZ UMBRAL*
A1	1 m
A2	2 m
A3	3 m
A4	3,5 m
A5	4 m

ANÁLISIS DE RESULTADOS (ANÁLISIS GRÁFICO)

ALGORITMO: CLASIFICACIÓN DE PUENTES MEDIANTE DETECCIÓN DE BORDES Y CLASIFICADOR ANGULAR POR SECTORES. PROCESO CON ITERACIONES (ALGORITMO B)

GRAFICO 1:

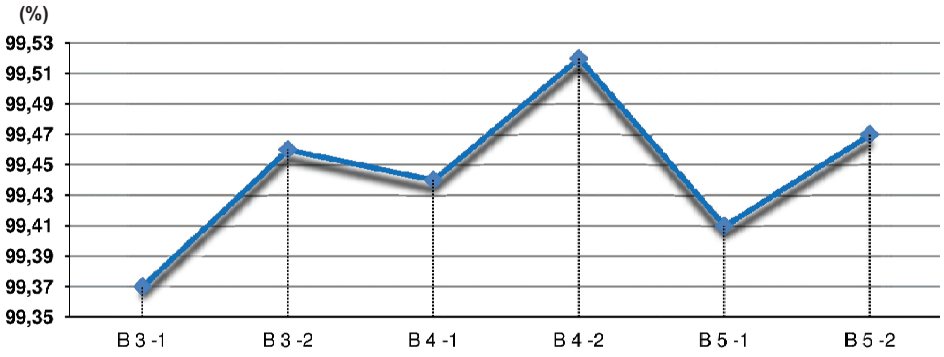
Muestra los errores de omisión, comisión y su media, obtenidos tras ejecutar el algoritmo B para los seis parámetros umbrales utilizados.



	B 3 -1	B 3 -2	B 4 -1	B 4 -2	B 5 -1	B 5 -2
Error de comisión (%)	42,62	24,64	30,32	5,65	29,66	6,78
Error de omisión (%)	1,42	2,18	1,58	2,34	23,3	24,58
Media de errores (%)	22,12	13,41	15,95	3,995	26,48	15,68

GRAFICO 2:

Muestra la exactitud global de la clasificación para los resultados obtenidos tras ejecutar el algoritmo B.



PARAMETROS UMBRALES EN IZ PARA DETECCIÓN DE BORDES:

ALGORITMO	*PROCESO 1*	*PROCESO 2*
B 3-1	3 m	1 m
B 3-2	3 m	2 m
B 4-1	3,5 m	1 m
B 4-2	3,5 m	2 m
B 5-1	4 m	1 m
B 5-2	4 m	2 m

ANÁLISIS DE RESULTADOS (ANÁLISIS GRÁFICO)

ALGORITMO: DETECCIÓN DE PUENTES MEDIANTE CLASIFICADOR ANGULAR POR SECTORES. COMPARACIÓN ENTRE LOS RESULTADOS OBTENIDOS POR EL ALGORITMO A Y B.

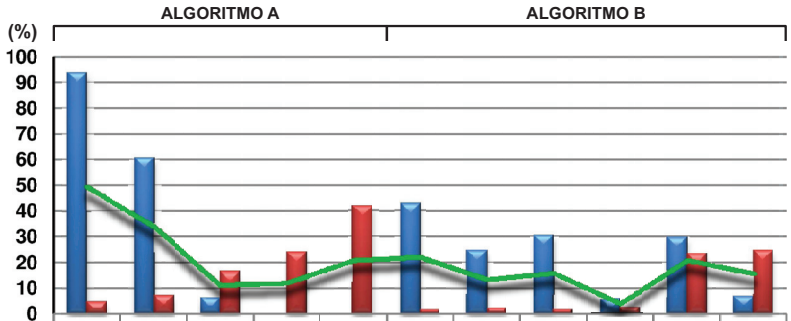
ZONA TEST: Collado de Villalba

DESCRIPCIÓN:

Comparación de los resultados obtenidos tras ejecutar los dos algoritmos para detectar puentes (algoritmo A y B). Ambos algoritmos se basan en la detección de bordes de puentes y su posterior clasificación mediante un clasificador angular por sectores. El algoritmo A concatena la detección de bordes de puentes con la posterior aplicación del clasificador angular, mientras que el algoritmo B utiliza como punto de partida el resultado obtenido tras ejecutar el Algoritmo A y completa la clasificación de puentes relajando los parámetros en el entorno de los puentes previamente detectados.

GRAFICO 1:

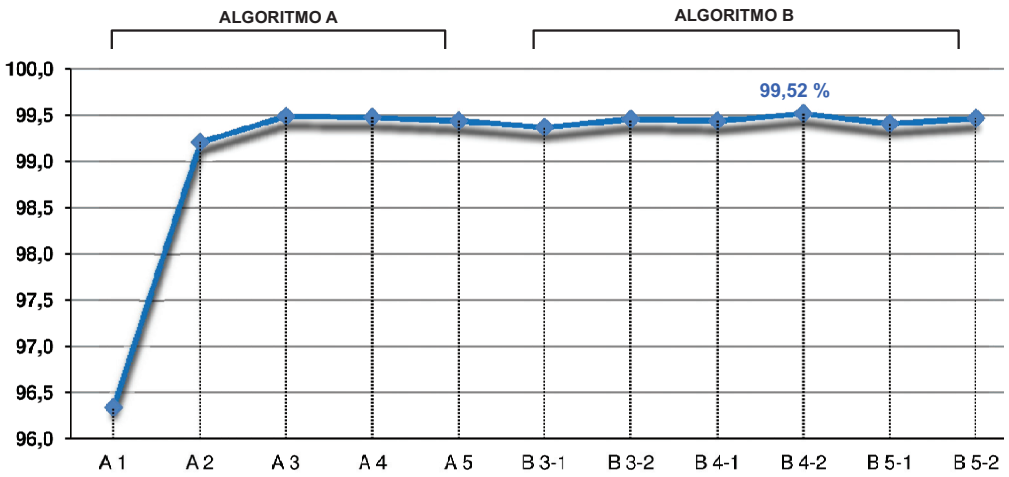
Muestra los errores de omisión, comisión y su media, obtenidos tras ejecutar el algoritmo A y el algoritmo B.



	A 1	A 2	A 3	A 4	A 5	B 3-1	B 3-2	B 4-1	B 4-2	B 5-1	B 5-2
Error de comisión (%)	93,97	60,49	6,17	0	0	42,82	24,64	30,32	5,65	29,66	6,78
Error de omisión (%)	4,56	7,21	16,16	24,15	41,89	1,42	2,18	1,58	2,34	23,3	24,58
Media de errores (%)	49,27	33,85	11,17	12,075	20,95	22,12	13,41	15,95	3,995	20,945	15,68

GRAFICO 2:

Muestra la exactitud global de la clasificación obtenida tras ejecutar el algoritmo A y el algoritmo B.



DESCRIPCIÓN DE PARÁMETROS UMBRALES UTILIZADOS

ALGORITMO: DIFERENCIACIÓN DE EDIFICIOS Y VEGETACIÓN.ANÁLISIS DE TEXTURAS (ALGORITMO A)

DESCRIPCIÓN:

Diferencia los edificios de la vegetación y otros pequeños objetos mediante técnicas de análisis de texturas. La aplicación segmenta la escena y analiza la distribución de las regiones que tienen pequeño tamaño. Las zonas donde se presenten aglomeraciones de regiones de pequeño tamaño se corresponderán con zonas de vegetación. El algoritmo determina el vecindario cilíndrico para cada punto LiDAR y lo subdivide en sectores y semisectores de igual amplitud, posteriormente un análisis con respecto a la densidad de regiones pequeñas y su distribución se realiza para cada sector.

PARÁMETROS UMBRALES FIJOS PARA ALGORITMO A:

RADIO DE VECINDARIO CILÍNDRICO PARA ANALIZAR TEXTURA: 6 m
RADIO DE LOS SEMISECTORES INTERIORES: 3,5 m
TAMAÑO DE REGIÓN UMBRAL: 10 m^2
AMPLITUD ANGULAR DE LOS SECTORES: 50 g
NÚMERO MÍNIMO DE PUNTOS PERTENECIENTES A REGIONES PEQUEÑAS EN SEMISECTORES: 10

PARÁMETROS UMBRALES VARIABLES PARA ALGORITMO A:

A 1: UMBRAL % = 70 % **A 2:** UMBRAL % = 80 % **A 3:** UMBRAL % = 90 % **A 4:** UMBRAL % = 99 %

ALGORITMO: DIFERENCIACIÓN DE EDIFICIOS Y VEGETACIÓN. ANÁLISIS DE TEXTURAS Y CLASIFICADOR ANGULAR CONCATENADOS (ALGORITMO B)

DESCRIPCIÓN:

Diferencia los edificios de la vegetación y otros pequeños objetos mediante mediante la contatenación de dos procesos. El primer proceso es un análisis de texturas (Algoritmo A) con el fin de clasificar las zonas interiores de zonas de densa vegetación. El segundo proceso se corresponde con un clasificador angular cuyo enfoque se basa en que las regiones pertenecientes a vegetación tienen puntos de tierra próximos distribuidos en todas las direcciones.

PARÁMETROS UMBRALES ASOCIADOS AL ANÁLISIS DE TEXTURA:

RADIO DE VECINDARIO CILÍNDRICO PARA ANALIZAR TEXTURA: 6 m
RADIO DE LOS SEMISECTORES INTERIORES: 3,5 m
TAMAÑO DE REGIÓN UMBRAL: 10 m^2
AMPLITUD ANGULAR DE LOS SECTORES: 50^g
NÚMERO MÍNIMO DE PUNTOS PERTENECIENTES A REGIONES PEQUEÑAS EN SEMISECTORES: 10
UMBRAL %: 99 %

PARÁMETROS UMBRALES ASOCIADOS AL CLASIFICADO ANGULAR:

PARÁMETROS UMBRALES FIJOS:

RADIO DEL VECINDARIO CILÍNDRICO: 6 m
TAMAÑO DE REGIÓN UMBRAL: 10 m^2

PARÁMETROS UMBRALES FIJOS:

B 1: UMBRAL ANGULAR = 40^g **B 2:** UMBRAL ANGULAR = 50^g **B 3:** UMBRAL ANGULAR = 60^g
B 4: UMBRAL ANGULAR = 70^g **B 5:** UMBRAL ANGULAR = 80^g

ALGORITMO: DIFERENCIACIÓN DE EDIFICIOS Y VEGETACIÓN. ANÁLISIS DE TEXTURAS (ALGORITMO A)

ZONA TEST: Vall d'Uixó

DESCRIPCIÓN:

Diferencia los edificios de la vegetación y otros pequeños objetos mediante técnicas de análisis de texturas. La aplicación segmenta la escena y analiza la distribución de las regiones que tienen pequeño tamaño. Las zonas donde se presenten aglomeraciones de regiones de pequeño tamaño se corresponderán con zonas de vegetación. El algoritmo determina el vecindario cilíndrico para cada punto LiDAR y lo subdivide en sectores y semisectores de igual amplitud, posteriormente un análisis con respecto a la densidad de regiones pequeñas y su distribución se realiza para cada sector.

PARÁMETROS UMBRALES FIJOS PARA ALGORITMO A:

RADIO DE VECINDARIO CILÍNDRICO PARA ANALIZAR TEXTURA: 6 m
RADIO DE LOS SEMISECTORES INTERIORES: 3,5 m
TAMAÑO DE REGIÓN UMBRAL: 10 m^2
AMPLITUD ANGULAR DE LOS SECTORES: 50 g

ALGORITMO: A 1

UMBRAL EN PORCENTAJE DE REGIONES PEQUEÑAS EN SECTORES: 70 %

MATRIZ DE CONFUSIÓN						
Clases de referencia	**Categorías deducidas de la clasificación**					
	Terreno	**Edificios**	**Vegetación**	**Total**	**Ex. Prod.**	**Error. Om.**
Terreno	**95085**	257	27	95369	99,70 %	0,30 %
Edificios	11	**199029**	234	199274	99,88 %	0,12 %
Vegetación	21	4933	**7699**	12653	60,85 %	39,15 %
Total	95117	204219	7960	**307296**	Exactitud global: 98,22 %	
Ex. Usuario	99,97 %	97,46 %	96,72 %			
Error. Com	0,03 %	2,54 %	3,28 %			

ALGORITMO: A 2

UMBRAL EN PORCENTAJE DE REGIONES PEQUEÑAS EN SECTORES: 80 %

MATRIZ DE CONFUSIÓN						
Clases de referencia	**Categorías deducidas de la clasificación**					
	Terreno	**Edificios**	**Vegetación**	**Total**	**Ex. Prod.**	**Error. Om.**
Terreno	**95085**	257	27	95369	99,70 %	0,30 %
Edificios	11	**199045**	218	199274	99,89 %	0,11 %
Vegetación	21	4983	**7649**	12653	60,45 %	39,55 %
Total	95117	204285	7894	**307296**	Exactitud global: 98,20 %	
Ex. Usuario	99,97 %	97,43 %	96,90 %			
Error. Com	0,03 %	2,57 %	3,10 %			

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

Nota: La clase " Vegetación" también incluye pequeños obtjetos próximos al terreno tales como automóviles, bancos, juegos infantiles en parques, ...

ALGORITMO: A 3

UMBRAL EN PORCENTAJE DE REGIONES PEQUEÑAS EN SECTORES: 90 %

MATRIZ DE CONFUSIÓN						
Clases de referencia	**Categorías deducidas de la clasificación**					
	Terreno	**Edificios**	**Vegetación**	**Total**	**Ex. Prod.**	**Error. Om.**
Terreno	**95085**	257	27	95369	99,70 %	0,30 %
Edificios	11	**199053**	210	199274	99,89 %	0,11 %
Vegetación	21	4985	**7647**	12653	60,44 %	39,56 %
Total	95117	204295	7884	**307296**	Exactitud global: 98,21 %	
Ex. Usuario	99,97 %	97,43 %	96,99 %			
Error. Com	0,03 %	2,57 %	3,01 %			

ALGORITMO: A 4

UMBRAL EN PORCENTAJE DE REGIONES PEQUEÑAS EN SECTORES: 99 %

MATRIZ DE CONFUSIÓN						
Clases de referencia	**Categorías deducidas de la clasificación**					
	Terreno	**Edificios**	**Vegetación**	**Total**	**Ex. Prod.**	**Error. Om.**
Terreno	**95085**	257	27	95369	99,70 %	0,30 %
Edificios	11	**199067**	196	199274	99,90 %	0,10 %
Vegetación	21	4990	**7642**	12653	60,40 %	39,60 %
Total	95117	204314	7865	**307296**	Exactitud global: 98,21 %	
Ex. Usuario	99,97 %	97,43 %	97,16 %			
Error. Com	0,03 %	2,57 %	2,84 %			

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

Nota: La clase " Vegetación" también incluye pequeños obtjetos próximos al terreno tales como automóviles, bancos, juegos infantiles en parques, ...

ALGORITMO: DIFERENCIACIÓN DE EDIFICIOS Y VEGETACIÓN. ANÁLISIS DE TEXTURAS Y CLASIFICADOR ANGULAR CONCATENADOS (ALGORITMO B)

ZONA TEST: Vall d'Uixó

DESCRIPCIÓN:

Diferencia los edificios de la vegetación y otros pequeños objetos mediante mediante la contatenación de dos procesos. El primer proceso es un análisis de texturas (Algoritmo A) con el fin de clasificar las zonas interiores de zonas de densa vegetación. El segundo proceso se corresponde con un clasificador angular cuyo enfoque se basa en que las regiones pertenecientes a vegetación tienen puntos de tierra próximos distribuidos en todas las direcciones.

PARÁMETROS UMBRALES FIJOS:

ANÁLISIS DE TEXTURAS:

RADIO DE VECINDARIO CILÍNDRICO PARA ANALIZAR TEXTURA: 6 m
RADIO DE LOS SEMISECTORES INTERIORES: 3,5 m
TAMAÑO DE REGIÓN UMBRAL: 10 m^2
AMPLITUD ANGULAR DE LOS SECTORES: 50 ^{g}
UMBRAL EN PORCENTAJE DE REGIONES PEQUEÑAS EN SECTORES: 99 %

CLASIFICADOR ANGULAR:

RADIO DE VECINDARIO CILÍNDRICO PARA EL CLASIFICADOR ANGULAR: 6 m
TAMAÑO DE REGIÓN UMBRAL: 10 m^2

ALGORITMO: B 1

UMBRAL ANGULAR: 40 ^{g}

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación					
	Terreno	**Edificios**	**Vegetación**	**Total**	**Ex. Prod.**	**Error. Om.**
Terreno	**95085**	257	27	95369	99,70 %	0,30 %
Edificios	11	**199027**	236	199274	99,89 %	0,12 %
Vegetación	21	1523	**11109**	12653	87,80 %	12,20 %
Total	95117	200807	11372	**307296**	Exactitud global: 99,32 %	
Ex. Usuario	99,97 %	99,11 %	97,69 %			
Error. Com	0,03 %	0,89 %	2,31 %			

ALGORITMO:B 2

UMBRAL ANGULAR: 50 ^{g}

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación					
	Terreno	**Edificios**	**Vegetación**	**Total**	**Ex. Prod.**	**Error. Om.**
Terreno	**95085**	257	27	95369	99,70 %	0,30 %
Edificios	11	**198953**	310	199274	99,84 %	0,16 %
Vegetación	21	1047	**11585**	12653	91,56 %	8,44 %
Total	95117	200257	11922	**307296**	Exactitud global: 99,46 %	
Ex. Usuario	99,97 %	99,35 %	97,17 %			
Error. Com	0,03 %	0,65 %	2,83 %			

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

Nota: La clase " Vegetación" también incluye pequeños obtjetos próximos al terreno tales como automóviles, bancos, juegos infantiles en parques, ...

ALGORITMO:B 3

UMBRAL ANGULAR: 60^{g}

MATRIZ DE CONFUSIÓN						
Clases de referencia	**Categorías deducidas de la clasificación**					
	Terreno	**Edificios**	**Vegetación**	**Total**	**Ex. Prod.**	**Error. Om.**
Terreno	**95085**	257	27	95369	99,70 %	0,30 %
Edificios	11	**198582**	681	199274	99,65 %	0,35 %
Vegetación	21	677	**11955**	12653	94,48 %	5,52 %
Total	95117	199516	12663	**307296**	Exactitud global: 99,46 %	
Ex. Usuario	99,97 %	99,35 %	94,41 %			
Error. Com	0,03 %	0,47 %	5,59 %			

ALGORITMO:B 4

UMBRAL ANGULAR: 70^{g}

MATRIZ DE CONFUSIÓN						
Clases de referencia	**Categorías deducidas de la clasificación**					
	Terreno	**Edificios**	**Vegetación**	**Total**	**Ex. Prod.**	**Error. Om.**
Terreno	**95085**	257	27	95369	99,70 %	0,30 %
Edificios	11	**198247**	1016	199274	99,48 %	0,52 %
Vegetación	21	293	**12339**	12653	97,52 %	2,48 %
Total	95117	198797	13382	**307296**	Exactitud global: 99,47 %	
Ex. Usuario	99,97 %	99,72 %	92,21 %			
Error. Com	0,03 %	0,28 %	7,79 %			

ALGORITMO:B 5

UMBRAL ANGULAR: 80^{g}

MATRIZ DE CONFUSIÓN						
Clases de referencia	**Categorías deducidas de la clasificación**					
	Terreno	**Edificios**	**Vegetación**	**Total**	**Ex. Prod.**	**Error. Om.**
Terreno	**95085**	257	27	95369	99,70 %	0,30 %
Edificios	11	**197699**	1564	199274	99,21 %	0,79 %
Vegetación	21	85	**12547**	12653	99,16 %	0,84 %
Total	95117	198039	14140	**307296**	Exactitud global: 99,36 %	
Ex. Usuario	99,97 %	99,83 %	83,73 %			
Error. Com	0,03 %	0,17 %	11,27 %			

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

Nota: La clase " Vegetación" también incluye pequeños obtjetos próximos al terreno tales como automóviles, bancos, juegos infantiles en parques, ...

ANÁLISIS DE RESULTADOS (ANÁLISIS GRÁFICO)

ALGORITMO: DIFERENCIACIÓN DE EDIFICIOS Y VEGETACIÓN. ANÁLISIS DE TEXTURAS (ALGORITMO A)

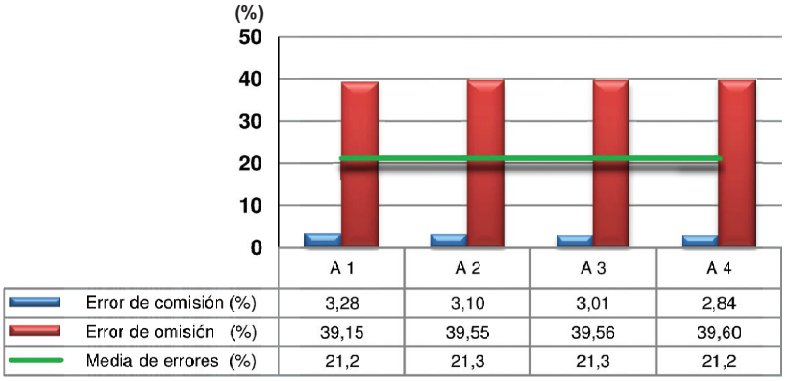
ZONA TEST: Vall d'Uixó

DESCRIPCIÓN:

Diferencia los edificios de la vegetación y otros pequeños objetos mediante técnicas de análisis de texturas. La aplicación segmenta la escena y analiza la distribución de las regiones que tienen pequeño tamaño. Las zonas donde se presenten aglomeraciones de regiones de pequeño tamaño se corresponderán con vegetación. El algoritmo determina el vecindario cilíndrico para cada punto LiDAR y lo subdivide en sectores y semisectores de igual amplitud, posteriormente un análisis con respecto a la densidad de regiones pequeñas y su distribución se realiza para cada sector.

GRAFICO 1:

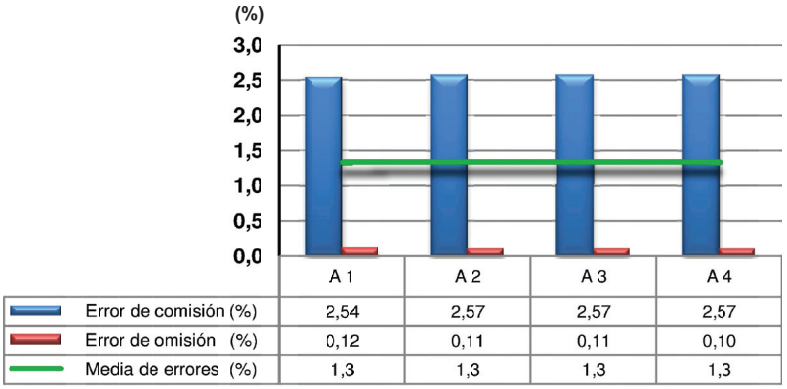
Muestra los errores de omisión, comisión y su media, obtenidos tras ejecutar el algoritmo A para los cuatro parámetros umbrales utilizados. Los errorres se corresponden con la clasificación de vegetación y pequeños objetos.



	A 1	A 2	A 3	A 4
Error de comisión (%)	3,28	3,10	3,01	2,84
Error de omisión (%)	39,15	39,55	39,56	39,60
Media de errores (%)	21,2	21,3	21,3	21,2

GRAFICO 2:

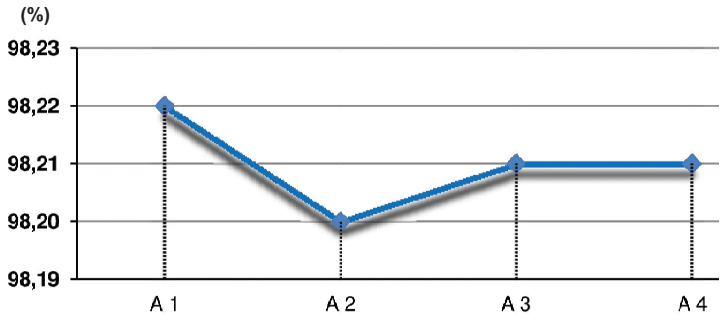
Muestra los errores de omisión, comisión y su media, obtenidos tras ejecutar el algoritmo A para los cuatro parámetros umbrales utilizados. Los errorres se corresponden con la clasificación de los edificios.



	A 1	A 2	A 3	A 4
Error de comisión (%)	2,54	2,57	2,57	2,57
Error de omisión (%)	0,12	0,11	0,11	0,10
Media de errores (%)	1,3	1,3	1,3	1,3

GRAFICO 3:

Muestra la exactitud global de la clasificación para los resultados obtenidos tras ejecutar el algoritmo A.



ANÁLISIS DE RESULTADOS (ANÁLISIS GRÁFICO)

ALGORITMO: DIFERENCIACIÓN DE EDIFICIOS Y VEGETACIÓN. ANÁLISIS DE TEXTURAS Y CLASIFICADOR ANGULAR CONCATENADOS (ALGORITMO B)

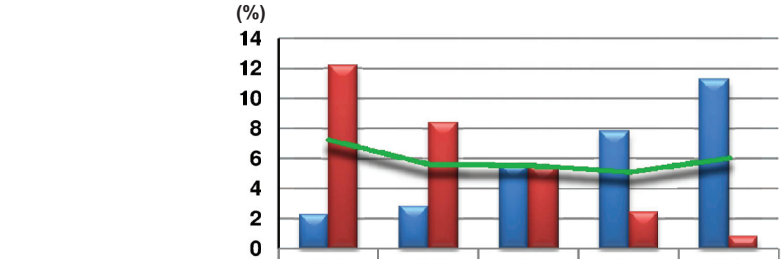
ZONA TEST: Vall d'Uixó

DESCRIPCIÓN:

Diferencia los edificios de la vegetación y otros pequeños objetos mediante mediante la contatenación de dos procesos. El primer proceso es un análisis de texturas (Algoritmo A) con el fin de clasificar las zonas interiores de zonas de densa vegetación. El segundo proceso se corresponde con un clasificador angular cuyo enfoque se basa en que las regiones pertenecientes a vegetación tienen puntos de tierra próximos distribuidos en todas las direcciones.

GRAFICO 1:

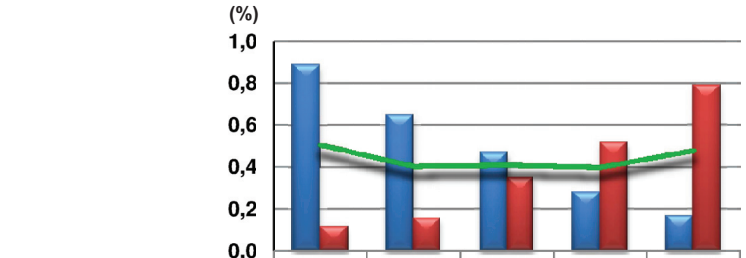
Muestra los errores de omisión, comisión y su media, obtenidos tras ejecutar el algoritmo B para los cinco parámetros umbrales utilizados. Los errorres se corresponden con la clasificación de vegetación y pequeños objetos.



	B 1	B 2	B 3	B 4	B 5
Error de comisión (%)	2,31	2,83	5,59	7,79	11,27
Error de omisión (%)	12,2	8,44	5,52	2,48	0,84
Media de errores (%)	7,26	5,64	5,53	5,14	6,06

GRAFICO 2:

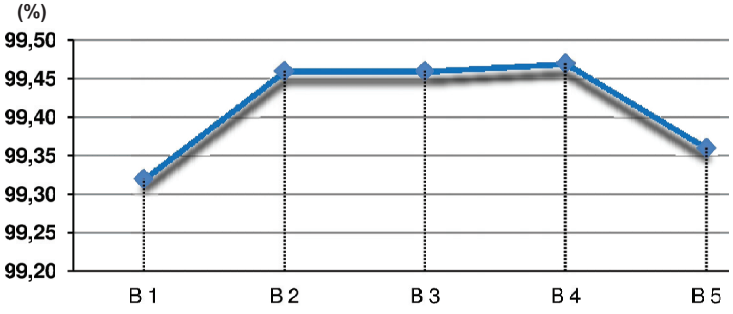
Muestra los errores de omisión, comisión y su media, obtenidos tras ejecutar el algoritmo B para los cinco parámetros umbrales utilizados. Los errorres se corresponden con la clasificación de los edificios.



	B 1	B 2	B 3	B 4	B 5
Error de comisión (%)	0,89	0,65	0,47	0,28	0,17
Error de omisión (%)	0,12	0,16	0,35	0,52	0,79
Media de errores (%)	0,51	0,41	0,41	0,40	0,48

GRAFICO 3:

Muestra la exactitud global de la clasificación para los resultados obtenidos tras ejecutar el algoritmo B.



ALGORITMO: DIFERENCIACIÓN DE EDIFICIOS Y VEGETACIÓN. ANÁLISIS DE TEXTURAS (ALGORITMO A)

ZONA TEST: Collado de Villalba

DESCRIPCIÓN:

Diferencia los edificios de la vegetación y otros pequeños objetos mediante técnicas de análisis de texturas. La aplicación segmenta la escena y analiza la distribución de las regiones que tienen pequeño tamaño. Las zonas donde se presenten aglomeraciones de regiones de pequeño tamaño se corresponderán con zonas de vegetación. El algoritmo determina el vecindario cilíndrico para cada punto LiDAR y lo subdivide en sectores y semisectores de igual amplitud, posteriormente un análisis con respecto a la densidad de regiones pequeñas y su distribución se realiza para cada sector.

PARÁMETROS UMBRALES FIJOS PARA ALGORITMO A:

RADIO DE VECINDARIO CILÍNDRICO PARA ANALIZAR TEXTURA: 6 m
RADIO DE LOS SEMISECTORES INTERIORES: 3,5 m
TAMAÑO DE REGIÓN UMBRAL: 10 m^2
AMPLITUD ANGULAR DE LOS SECTORES: 50 g
NÚMERO MÍNIMO DE PUNTOS PERTENECIENTES A REGIONES PEQUEÑAS EN SEMISECTORES: 10

ALGORITMO: A 1

UMBRAL EN PORCENTAJE DE REGIONES PEQUEÑAS EN SECTORES: 70 %

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación						
	Terreno	**Puentes**	**Edificios**	**Vegetación**	**Total**	**Ex. Prod.**	**Error. Om.**
Terreno	**1447977**	354	3666	1208	1453205	99,64 %	0,36 %
Puentes	130	**5951**	0	12	6057	97,66 %	2,34 %
Edificios	96	0	**790916**	4299	795311	99,45 %	0,55 %
Vegetación	8227	0	166745	**395177**	570149	99,39 %	30,69 %
Total	1456430	6269	961327	400696	**2824722**	Exactitud global: 93,46 %	
Ex. Usuario	99,42 %	94,35 %	82,27 %	98,62 %			
Error. Com	0,58 %	5,65 %	17,73 %	1,38 %			

ALGORITMO: A 2

UMBRAL EN PORCENTAJE DE REGIONES PEQUEÑAS EN SECTORES: 80 %

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación						
	Terreno	**Puentes**	**Edificios**	**Vegetación**	**Total**	**Ex. Prod.**	**Error. Om.**
Terreno	**1447977**	354	3725	1149	1453205	99,64 %	0,36 %
Puentes	130	**5951**	0	12	6057	97,66 %	0,27 %
Edificios	96	0	**793028**	2187	795311	99,71 %	0,29 %
Vegetación	8227	0	189591	**372331**	570149	65,30 %	34,70 %
Total	1456430	6269	986344	375679	**2824722**	Exactitud global: 92,73 %	
Ex. Usuario	99,42 %	94,35 %	80,40 %	99,11 %			
Error. Com	0,58 %	5,65 %	19,60 %	0,89 %			

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

Nota: La clase " Vegetación" también incluye pequeños obtjetos próximos al terreno tales como automóviles, bancos, juegos infantiles en parques, ...

ALGORITMO: A 3

UMBRAL EN PORCENTAJE DE REGIONES PEQUEÑAS EN SECTORES: 90 %

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación						
	Terreno	**Puentes**	**Edificios**	**Vegetación**	**Total**	**Ex. Prod.**	**Error. Om.**
Terreno	**1447977**	354	3816	1058	1453205	99,64 %	0,36 %
Puentes	130	**5951**	0	12	6057	97,66 %	2,34 %
Edificios	96	0	**794130**	1085	795311	99,85 %	0,15 %
Vegetación	8227	0	211009	**350913**	570149	61,55 %	38,45 %
Total	1456430	6269	1008955	353068	**2824722**	Exactitud global: 92,01 %	
Ex. Usuario	99,42 %	94,35 %	78,71 %	99,39 %			
Error. Com	0,58 %	5,65 %	21,29 %	0,61 %			

ALGORITMO: A 4

UMBRAL EN PORCENTAJE DE REGIONES PEQUEÑAS EN SECTORES: 99 %

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación						
	Terreno	**Puentes**	**Edificios**	**Vegetación**	**Total**	**Ex. Prod.**	**Error. Om.**
Terreno	**1447977**	354	3835	1039	1453205	99,64 %	0,36 %
Puentes	130	**5951**	0	12	6057	97,66 %	2,34 %
Edificios	96	0	**794802**	413	795311	99,94 %	0,06 %
Vegetación	8227	0	237520	**324402**	570149	56,90 %	43,10 %
Total	1456430	6269	1036157	329866	**2824722**	Exactitud global: 91,09 %	
Ex. Usuario	99,42 %	94,35 %	76,71 %	99,55 %			
Error. Com	0,58 %	5,65 %	23,29 %	0,45 %			

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

Nota: La clase " Vegetación" también incluye pequeños obtjetos próximos al terreno tales como automóviles, bancos, juegos infantiles en parques, ...

ANÁLISIS DE RESULTADOS (MATRICES DE CONFUSIÓN)

ALGORITMO: DIFERENCIACIÓN DE EDIFICIOS Y VEGETACIÓN. ANÁLISIS DE TEXTURAS Y CLASIFICADOR ANGULAR CONCATENADOS (ALGORITMO B)

ZONA TEST: Collado de Villalba

DESCRIPCIÓN:

Diferencia los edificios de la vegetación y otros pequeños objetos mediante mediante la contatenación de dos procesos. El primer proceso es un análisis de texturas (Algoritmo A) con el fin de clasificar las zonas interiores de zonas de densa vegetación. El segundo proceso se corresponde con un clasificador angular cuyo enfoque se basa en que las regiones pertenecientes a vegetación tienen puntos de tierra próximos distribuidos en todas las direcciones.

PARÁMETROS UMBRALES FIJOS:

ANÁLISIS DE TEXTURAS:

RADIO DE VECINDARIO CILÍNDRICO PARA ANALIZAR TEXTURA: 6 m
RADIO DE LOS SEMISECTORES INTERIORES: 3,5 m
TAMAÑO DE REGIÓN UMBRAL: 10 m^2
AMPLITUD ANGULAR DE LOS SECTORES: 50 ^{g}
UMBRAL EN PORCENTAJE DE REGIONES PEQUEÑAS EN SECTORES: 99 %

CLASIFICADOR ANGULAR:

RADIO DE VECINDARIO CILÍNDRICO PARA EL CLASIFICADOR ANGULAR: 6 m
TAMAÑO DE REGIÓN UMBRAL: 10 m^2

UMBRAL ANGULAR: 40 ^{g} **ALGORITMO: B 1**

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación						
	Terreno	**Puentes**	**Edificios**	**Vegetación**	**Total**	**Ex. Prod.**	**Error. Om.**
Terreno	**1447977**	354	3000	1874	1453205	99,64 %	0,36 %
Puentes	130	**5951**	0	12	6057	97,66 %	2,34 %
Edificios	96	0	**793765**	1450	795311	99,81 %	0,19 %
Vegetación	8227	0	20304	**541618**	570149	95,00 %	5,00 %
Total	1456430	6269	817069	544954	**2824722**	Exactitud global: 98,75 %	
Ex. Usuario	99,42 %	94,35 %	97,15 %	99,39 %			
Error. Com	0,58 %	5,65 %	2,85 %	0,61 %			

UMBRAL ANGULAR: 50 ^{g} **ALGORITMO: B 2**

MATRIZ DE CONFUSIÓN

Clases de referencia	Categorías deducidas de la clasificación						
	Terreno	**Puentes**	**Edificios**	**Vegetación**	**Total**	**Ex. Prod.**	**Error. Om.**
Terreno	**1447977**	354	2984	1890	1453205	99,64 %	0,36 %
Puentes	130	**5951**	0	12	6057	97,66 %	2,34 %
Edificios	96	0	**780270**	14945	795311	98,11 %	1,89 %
Vegetación	8227	0	6072	**555850**	570149	97,49 %	2,51 %
Total	1456430	6269	789326	572697	**2824722**	Exactitud global: 98,77 %	
Ex. Usuario	99,42 %	94,35 %	98,85 %	97,06 %			
Error. Com	0,58 %	5,65 %	1,15 %	2,94 %			

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

Nota: La clase " Vegetación" también incluye pequeños obtjetos próximos al terreno tales como automóviles, bancos, juegos infantiles en parques, ...

ALGORITMO: B 3

UMBRAL EN PORCENTAJE DE REGIONES PEQUEÑAS EN SECTORES: 60 %

MATRIZ DE CONFUSIÓN							
Clases de referencia	**Categorías deducidas de la clasificación**						
	Terreno	**Puentes**	**Edificios**	**Vegetación**	**Total**	**Ex. Prod.**	**Error. Om.**
Terreno	**1447977**	354	2924	1950	1453205	99,64 %	0,36 %
Puentes	130	**5951**	0	12	6057	97,66 %	2,34 %
Edificios	96	0	**794130**	30285	795311	96,18 %	2,82 %
Vegetación	8227	0	5023	**556899**	570149	97,68 %	2,32 %
Total	1456430	6269	772877	589146	**2824722**	Exactitud global: 98,27 %	
Ex. Usuario	99,42 %	94,35 %	98,97 %	94,53 %			
Error. Com	0,58 %	5,65 %	1,03 %	5,47 %			

ALGORITMO: B 4

UMBRAL EN PORCENTAJE DE REGIONES PEQUEÑAS EN SECTORES: 70 %

MATRIZ DE CONFUSIÓN							
Clases de referencia	**Categorías deducidas de la clasificación**						
	Terreno	**Puentes**	**Edificios**	**Vegetación**	**Total**	**Ex. Prod.**	**Error. Om.**
Terreno	**1447977**	354	2822	2052	1453205	99,64 %	0,36 %
Puentes	130	**5951**	0	12	6057	97,66 %	2,34 %
Edificios	96	0	**748265**	46950	795311	94,08 %	5,92 %
Vegetación	8227	0	220	**561702**	570149	98,52 %	1,48 %
Total	1456430	6269	751307	610716	**2824722**	Exactitud global: 97,85 %	
Ex. Usuario	99,42 %	94,35 %	99,60 %	91,97 %			
Error. Com	0,58 %	5,65 %	0,40 %	8,03 %			

ALGORITMO: B 5

UMBRAL EN PORCENTAJE DE REGIONES PEQUEÑAS EN SECTORES: 80 %

MATRIZ DE CONFUSIÓN							
Clases de referencia	**Categorías deducidas de la clasificación**						
	Terreno	**Puentes**	**Edificios**	**Vegetación**	**Total**	**Ex. Prod.**	**Error. Om.**
Terreno	**1447977**	354	2804	2070	1453205	99,64 %	0,36 %
Puentes	130	**5951**	0	12	6057	97,66 %	2,34 %
Edificios	96	0	**728794**	66421	795311	91,64 %	8,36 %
Vegetación	8227	0	148	**561774**	570149	98,53 %	1,47 %
Total	1456430	6269	731746	630277	**2824722**	Exactitud global: 97,2 %	
Ex. Usuario	99,42 %	94,35 %	99,60 %	89,13 %			
Error. Com	0,58 %	5,65 %	0,40 %	10,87 %			

Error. Com: Error de comisión Error. Om: Error de omisión Ex. Usuario: Exactitud de usuario Ex. Prod: Exactitud del productor

Nota: La clase " Vegetación" también incluye pequeños obtjetos próximos al terreno tales como automóviles, bancos, juegos infantiles en parques, ...

ALGORITMO: DIFERENCIACIÓN DE EDIFICIOS Y VEGETACIÓN. ANÁLISIS DE TEXTURAS (ALGORITMO A)

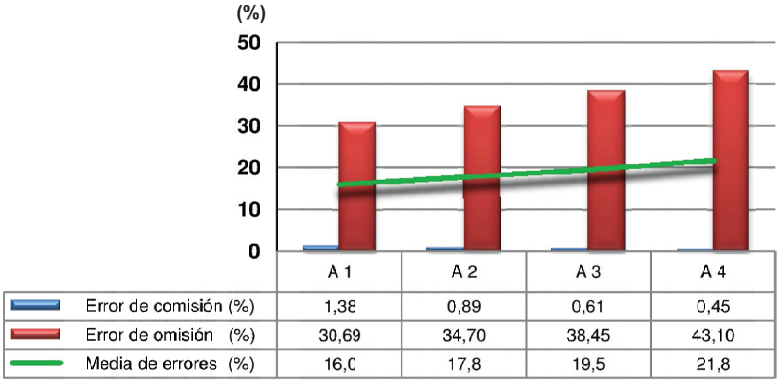
ZONA TEST: Collado de Villalba

DESCRIPCIÓN:

Diferencia los edificios de la vegetación y otros pequeños objetos mediante técnicas de análisis de texturas. La aplicación segmenta la escena y analiza la distribución de las regiones que tienen pequeño tamaño. Las zonas donde se presenten aglomeraciones de regiones de pequeño tamaño se corresponderán con vegetación. El algoritmo determina el vecindario cilíndrico para cada punto LiDAR y lo subdivide en sectores y semisectores de igual amplitud, posteriormente un análisis con respecto a la densidad de regiones pequeñas y su distribución se realiza para cada sector.

GRAFICO 1:

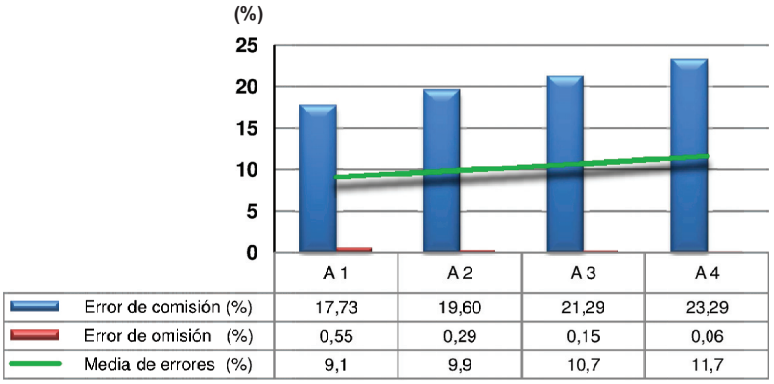
Muestra los errores de omisión, comisión y su media, obtenidos tras ejecutar el algoritmo A para los cuatro parámetros umbrales utilizados. Los errorres se corresponden con la clasificación de vegetación y pequeños objetos.



	A 1	A 2	A 3	A 4
Error de comisión (%)	1,38	0,89	0,61	0,45
Error de omisión (%)	30,69	34,70	38,45	43,10
Media de errores (%)	16,0	17,8	19,5	21,8

GRAFICO 2:

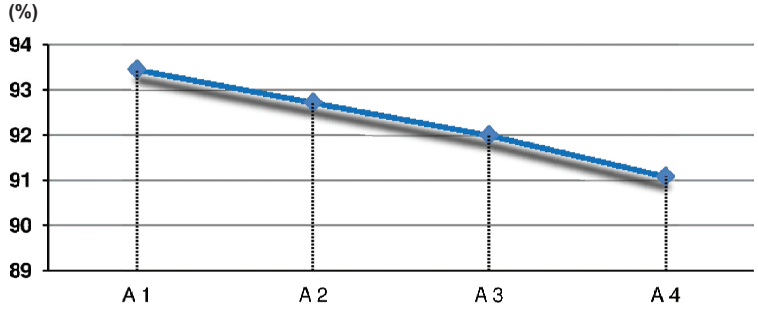
Muestra los errores de omisión, comisión y su media, obtenidos tras ejecutar el algoritmo A para los cuatro parámetros umbrales utilizados. Los errorres se corresponden con la clasificación de los edificios.



	A 1	A 2	A 3	A 4
Error de comisión (%)	17,73	19,60	21,29	23,29
Error de omisión (%)	0,55	0,29	0,15	0,06
Media de errores (%)	9,1	9,9	10,7	11,7

GRAFICO 3:

Muestra la exactitud global de la clasificación para los resultados obtenidos tras ejecutar el algoritmo A.



ANÁLISIS DE RESULTADOS (ANÁLISIS GRÁFICO)

ALGORITMO: DIFERENCIACIÓN DE EDIFICIOS Y VEGETACIÓN. ANÁLISIS DE TEXTURAS Y CLASIFICADOR ANGULAR CONCATENADOS (ALGORITMO B)

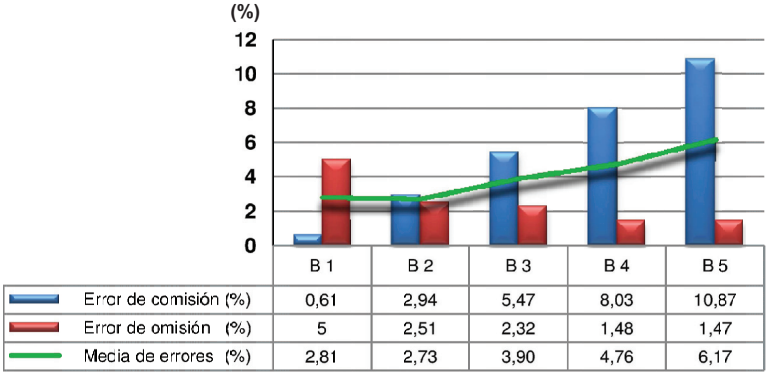
ZONA TEST: Collado de Villalba

DESCRIPCIÓN:

Diferencia los edificios de la vegetación y otros pequeños objetos mediante mediante la contatenación de dos procesos. El primer proceso es un análisis de texturas (Algoritmo A) con el fin de clasificar las zonas interiores de zonas de densa vegetación. El segundo proceso se corresponde con un clasificador angular cuyo enfoque se basa en que las regiones pertenecientes a vegetación tienen puntos de tierra próximos distribuidos en todas las direcciones.

GRAFICO 1:

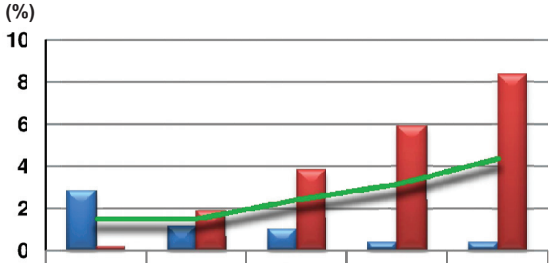
Muestra los errores de omisión, comisión y su media, obtenidos tras ejecutar el algoritmo B para los cinco parámetros umbrales utilizados. Los errorres se corresponden con la clasificación de vegetación y pequeños objetos.



	B 1	B 2	B 3	B 4	B 5
Error de comisión (%)	0,61	2,94	5,47	8,03	10,87
Error de omisión (%)	5	2,51	2,32	1,48	1,47
Media de errores (%)	2,81	2,73	3,90	4,76	6,17

GRAFICO 2:

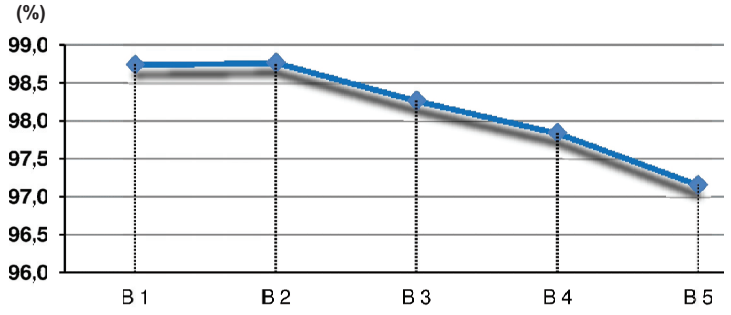
Muestra los errores de omisión, comisión y su media, obtenidos tras ejecutar el algoritmo B para los cinco parámetros umbrales utilizados. Los errorres se corresponden con la clasificación de los edificios.



	B 1	B 2	B 3	B 4	B 5
Error de comisión (%)	2,85	1,15	1,03	0,40	0,40
Error de omisión (%)	0,19	1,89	3,82	5,92	8,36
Media de errores (%)	1,52	1,52	2,43	3,16	4,38

GRAFICO 3:

Muestra la exactitud global de la clasificación para los resultados obtenidos tras ejecutar el algoritmo B.



VIII. ANEXO II. MAPAS

DESCRIPCIÓN:

En el presente apartado se recogen los mapas de errores para cada uno de los algoritmos implementados. Los resultados se corresponden con los ensayos que mayor precisión han alcanzado.

LISTADO DE MAPAS:

Vall Duixó:

1.1 Mapa de situación
1.2 Detección de errores groseros
1.3 Detección del terreno
1.4 Detección de edificios
1.5 Detección de vegetación y otros pequeños objetos
1.6 Clasificación automática y clasificación verdadera

Collado Villalba:

2.1 Mapa de situación
2.2 Detección de errores groseros
2.3 Detección del terreno
2.4 Detección de puentes
2.5 Detección de edificios
2.6 Detección de vegetación y otros pequeños objetos
2.7 Clasificación automática y clasificación verdadera

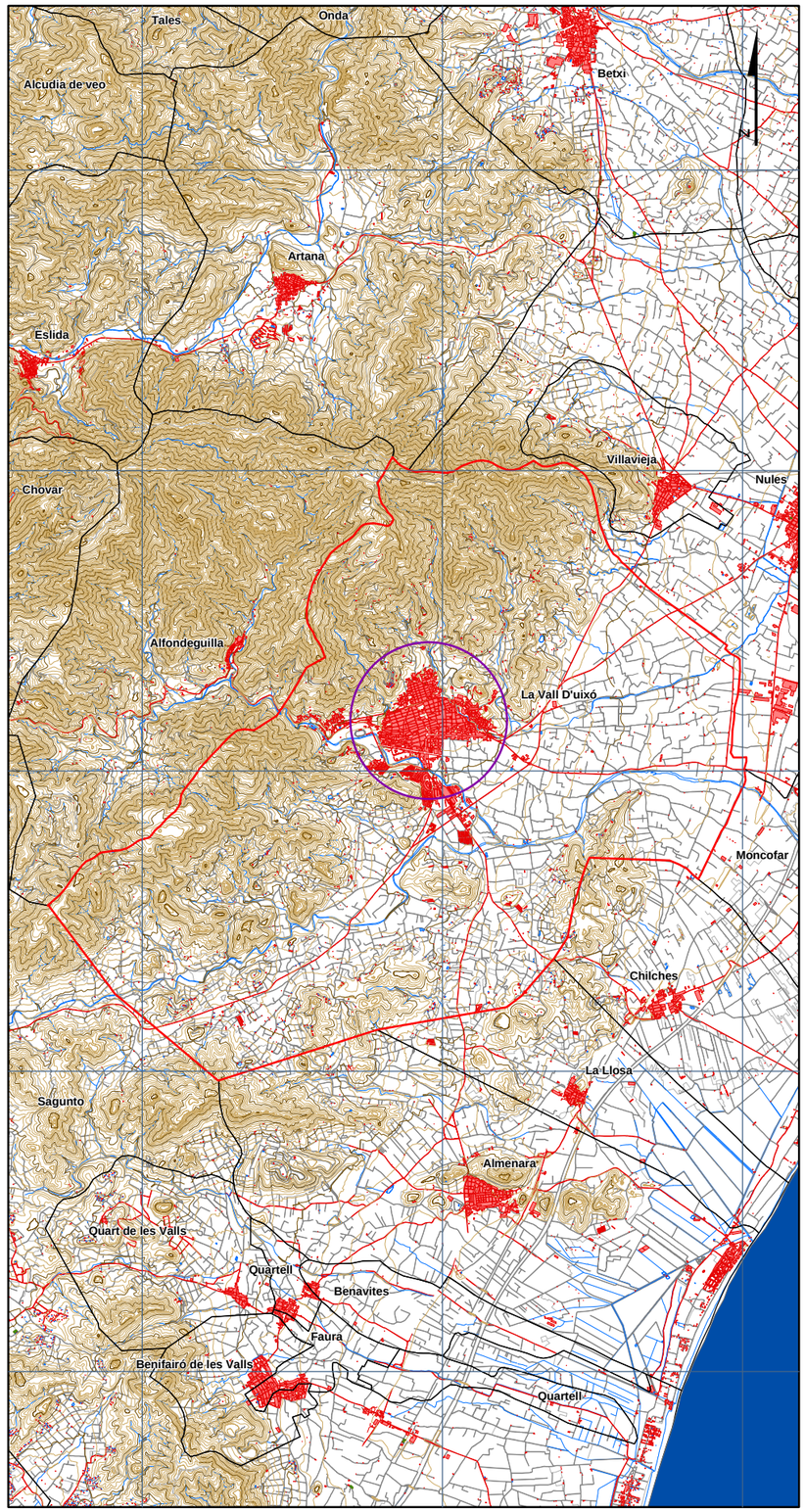
FUENTES CARTOGRÁFICAS UTILIZADAS:

Vall Duixó:

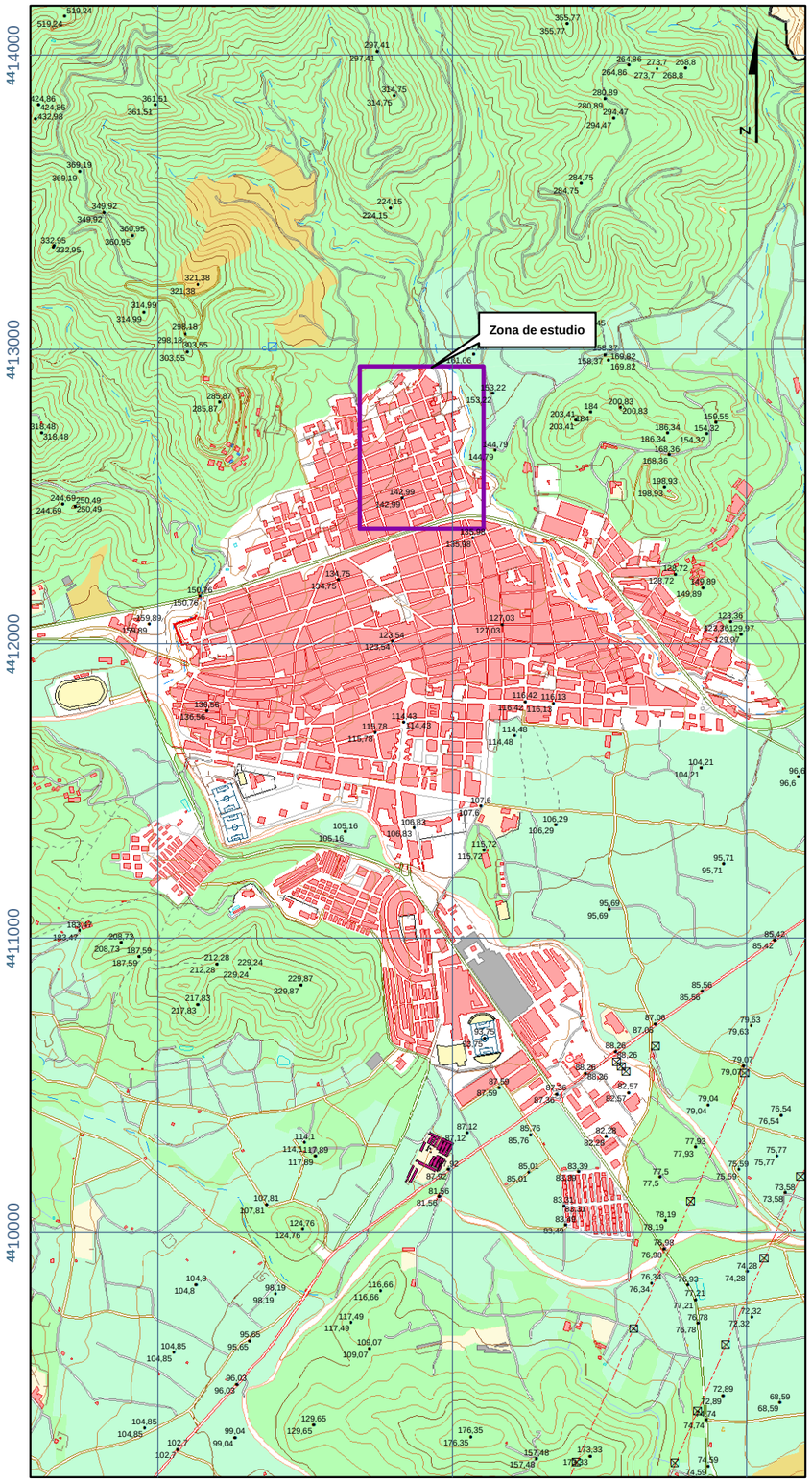
Cartografía 1:10.000 del Instituto Cartográfico Valenciano.
Cartografía 1:25.000 del Instituto Geográfico Nacional.
Ortofototos de 2005 del Instituto Cartográfico Valenciano.

Collado Villalba:

Cartografía 1:25.000 del Instituto Geográfico Nacional.
Ortofototos de 2006 del Instituto Geográfico Nacional.



Datum ETRS89. Proyección UTM. Huso 30

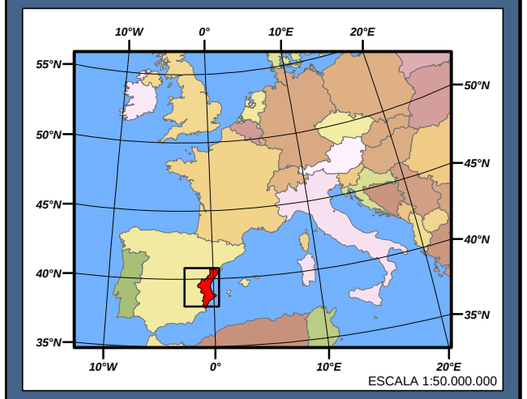


Datum ETRS89. Proyección UTM. Huso 30

DESARROLLO E IMPLEMENTACIÓN DE ALGORITMOS PARA CLASIFICAR DATOS LIDAR EN ÁREAS URBANAS

MAPAS DE SITUACIÓN

ÁMBITO EUROPEO



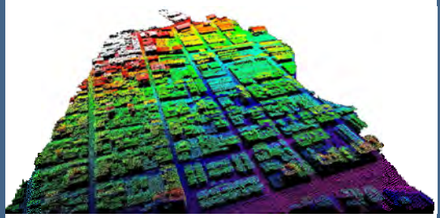
ESCALA 1:50.000.000

ÁMBITO COMUNITARIO

PROYECCIÓN: UTM ESCALA 1:4.000.0000

AUTOR	**José Sánchez Lopera**
DIRECTOR DE TESIS	**José Luis Lerma García**
ZONA TEST	**Vall d'Uixó**
NÚMERO DE PLANO	**1.1**

MAPA DE SITUACIÓN

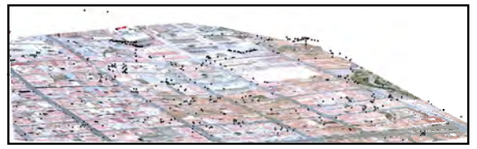


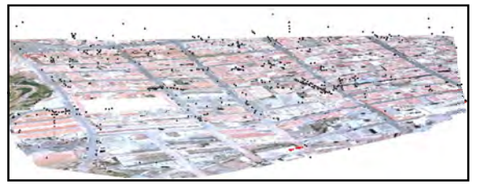
AGOSTO 2015

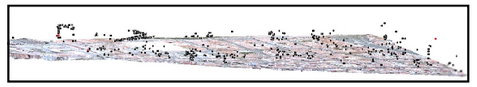
Universitat Politècnica de València

Departamento de Ingeniería Cartográfica, Geodesia y Fotogrametría

RESULTADOS OBTENIDOS. IMÁGENES 3D







Leyenda

- Errores groserors correctamente clasificados
- Error de omisión
- Error de comisión

0 25 50 100 150 m

Datum ETRS89. Proyección UTM. Huso 30

DESARROLLO E IMPLEMENTACIÓN DE ALGORITMOS PARA CLASIFICAR DATOS LIDAR EN ÁREAS URBANAS

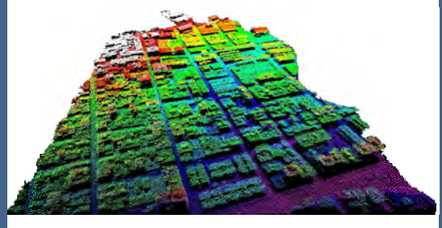
AUTOR	José Sánchez Lopera
DIRECTOR DE TESIS	José Luis Lerma García
ZONA TEST	Vall d'Uixó
NÚMERO DE PLANO	1.2

Detección de errores groseros. Algoritmo C: Análisis de variaciones de desnivel en vecindario próximo. Multiproceso con parámetros adaptativos.	
Ensayo	C 5
PROCESO 1: PARÁMETROS UMBRALES	
Desnivel (IZ)	10 m
Tamaño regiones	15 m^2
Radio vecindario	5 m
Porcentaje (%)	85 %
PROCESO 2: PARÁMETROS UMBRALES	
Desnivel (IZ)	10 m
Tamaño regiones	15 m^2
Radio vecindario	5 m
Porcentaje (%)	95 %

ERRORES COMETIDOS	
Error de comisión	0,00 %
Error de omisión	3,87 %
Media de errores	1,935 %

DETECCIÓN DE ERRORES GROSEROS

1:2.000

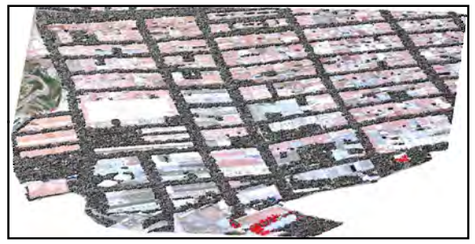


AGOSTO 2015

Universitat Politècnica de València

Departamento de Ingeniería Cartográfica, Geodesia y Fotogrametría

RESULTADOS OBTENIDOS. IMÁGENES 3D



Leyenda

- Terreno correctamente clasificado
- Error de omisión
- Error de comisión

0 25 50 100 150 m

Datum ETRS89. Proyección UTM. Huso 30

DESARROLLO E IMPLEMENTACIÓN DE ALGORITMOS PARA CLASIFICAR DATOS LIDAR EN ÁREAS URBANAS

AUTOR	José Sánchez Lopera
DIRECTOR DE TESIS	José Luis Lerma García
ZONA TEST	Vall d'Uixó
NÚMERO DE PLANO	1.3

Detección automática de puntos pertenecientes al terreno. Algoritmo B: densificación progresiva mediante interpolación selectiva combinada con segmentación y aplicación de reglas de decisión.	
Ensayo	B 2-1, B 2-2, B 2-3 y B 2-4
PARÁMETROS UMBRALES	
IZ en segmentación	0,25 m^2
Tamaño celda	1 m
B	3
L_Bloque	10 m, 25 m, 50 m y 100 m
U_IZ	0,4 m

B: Exponente de ponderación en la interpolación.

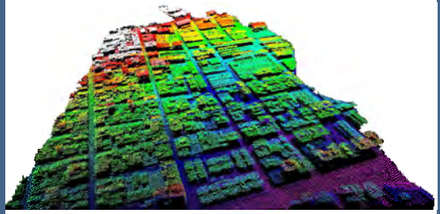
L_Bloque: Longitud del lado de los bloques necesarios para generar el terreno provisional inicial.

U_IZ: Umbral en desnivel con respecto a la superficie de referencia interpolada.

ERRORES COMETIDOS	
Error de comisión	0,03 %
Error de omisión	0,30 %
Media de errores	0,17 %

DETECCIÓN DEL TERRENO

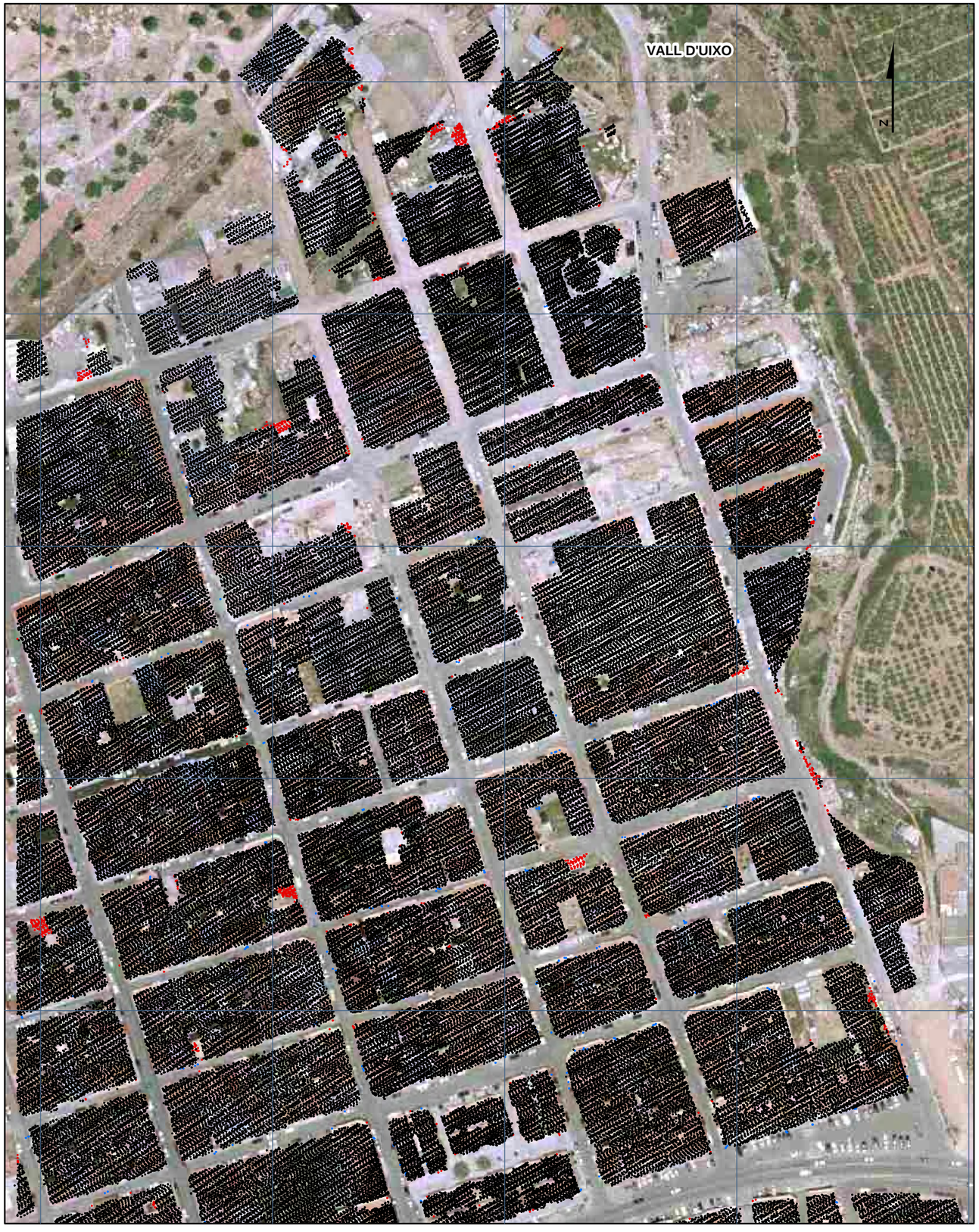
1:2.000



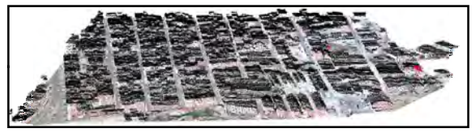
AGOSTO 2015

Universitat Politècnica de València

Departamento de Ingeniería Cartográfica, Geodesia y Fotogrametría



RESULTADOS OBTENIDOS. IMÁGENES 3D



Leyenda

- Edificios correctamente clasificados
- Error de omisión
- Error de comisión

0 25 50 100 150 m

Datum ETRS89. Proyección UTM. Huso 30

DESARROLLO E IMPLEMENTACIÓN DE ALGORITMOS PARA CLASIFICAR DATOS LIDAR EN ÁREAS URBANAS

AUTOR	José Sánchez Lopera
DIRECTOR DE TESIS	José Luis Lerma García
ZONA TEST	Vall d'Uixó
NÚMERO DE PLANO	1.4

Detección de edificios así como vegetació y otros pequeños objetos. Algoritmo B: análisis de textura y clasificador angular.

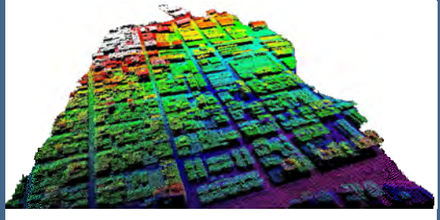
Ensayo	B 4
PARÁMETROS UMBRALES. PROCESO 1: ANÁLISIS DE TEXTURAS	
Radio_V	6 m
Radio_S	3,5 m
Tamaño regiones	10 m^2
Amplitud angular	50 ^{g}
Nº Ptos_S	10
U_%	99 %
PARÁMETROS UMBRALES. PROCESO 2: CLASIFICADOR ANGULAR	
Radio_V	6 m
Tamaño regiones	10 m^2
Amplitud angular	70 ^{g}

Radio_V: Radio del vecindario cilíndrico.
Radio_S: Radio de los semisectores interiores.
Nº Ptos_S: Número mínimo de puntos pertenecientes a regiones pequeñas en semisectores.
U_%: Porcentaje de puntos mínimo pertenecientes a las regiones pequeñas en cada sector.

ERRORES COMETIDOS	
Error de comisión	0,28 %
Error de omisión	0,52 %
Media de errores	0,40 %

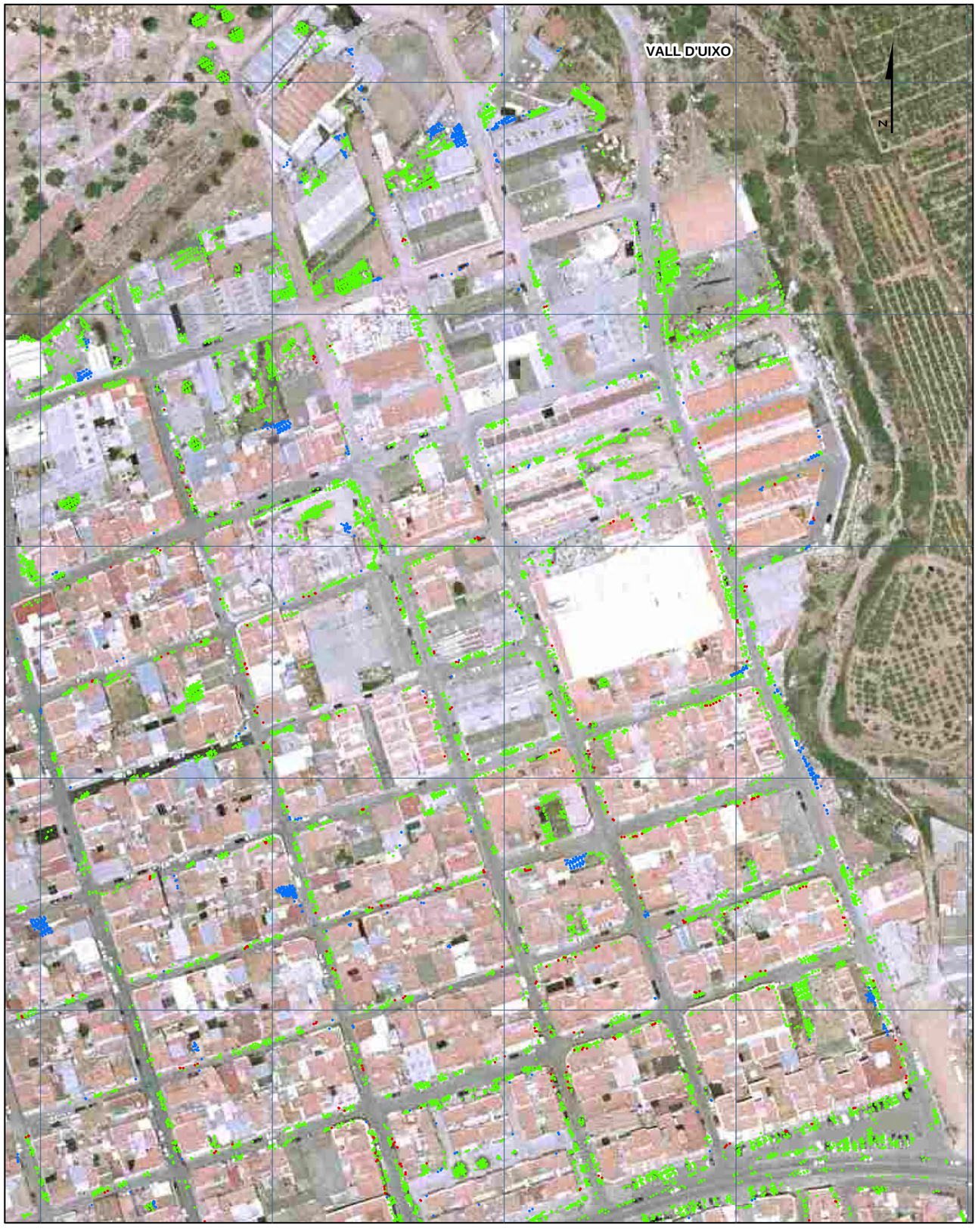
DETECCIÓN DE EDIFICIOS

1:2.000

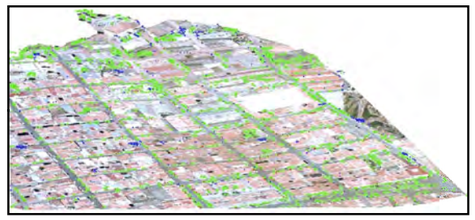


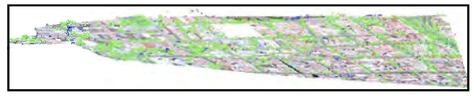
AGOSTO 2015

Universitat Politècnica de València
Departamento de Ingeniería Cartográfica, Geodesia y Fotogrametría



RESULTADOS OBTENIDOS. IMÁGENES 3D





Leyenda

- Vegetación y otros pequeños objetos correctamente clasificados
- Error de omisión
- Error de comisión

0 25 50 100 150 m

Datum ETRS89. Proyección UTM. Huso 30

DESARROLLO E IMPLEMENTACIÓN DE ALGORITMOS PARA CLASIFICAR DATOS LIDAR EN ÁREAS URBANAS

AUTOR	José Sánchez Lopera
DIRECTOR DE TESIS	José Luis Lerma García
ZONA TEST	Vall d'Uixó
NÚMERO DE PLANO	1.5

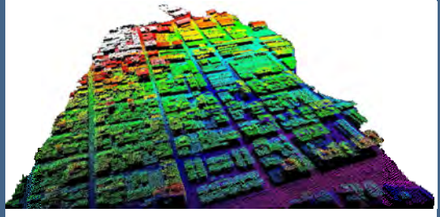
Detección de edificios así como vegetació y otros pequeños objetos. Algoritmo B: análisis de textura y clasificador angular.	
Ensayo	B 4
PARÁMETROS UMBRALES. PROCESO 1: ANÁLISIS DE TEXTURAS	
Radio_V	6 m
Radio_S	3,5 m
Tamaño regiones	10 m^2
Amplitud angular	50 ^{g}
Nº Ptos_S	10
U_%	99 %
PARÁMETROS UMBRALES. PROCESO 2: CLASIFICADOR ANGULAR	
Radio_V	6 m
Tamaño regiones	10 m^2
Amplitud angular	70 ^{g}

Radio_V: Radio del vecindario cilíndrico.
Radio_S: Radio de los semisectores interiores.
Nº Ptos_S: Número mínimo de puntos pertenecientes a regiones pequeñas en semisectores.
U_%: Porcentaje de puntos mínimo pertenecientes a las regiones pequeñas en cada sector.

ERRORES COMETIDOS	
Error de comisión	7,79 %
Error de omisión	2,48 %
Media de errores	5,14 %

DETECCIÓN DE VEGETACIÓN Y OTROS PEQUEÑOS OBJETOS

1:2.000

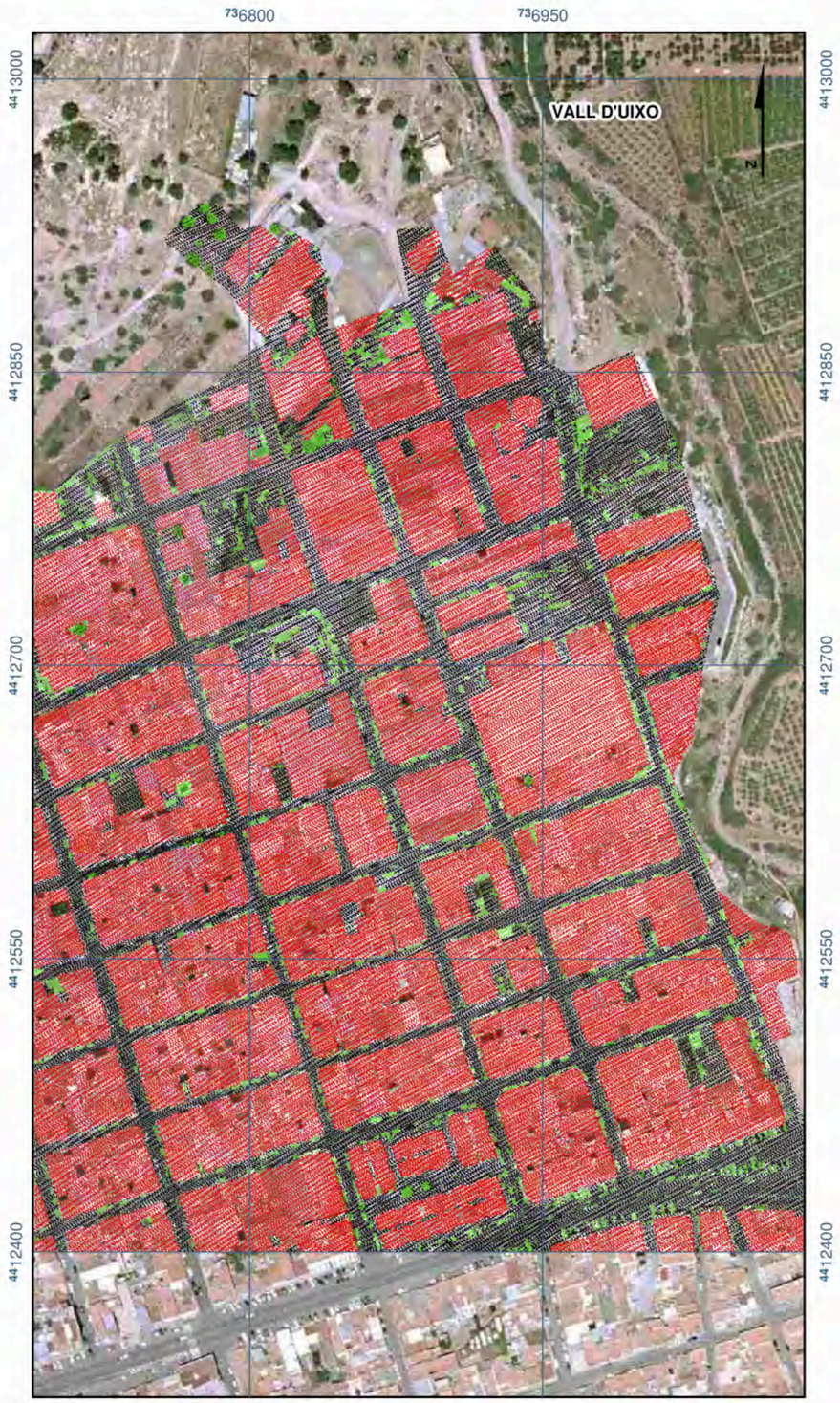


AGOSTO 2015

Universitat Politècnica de València

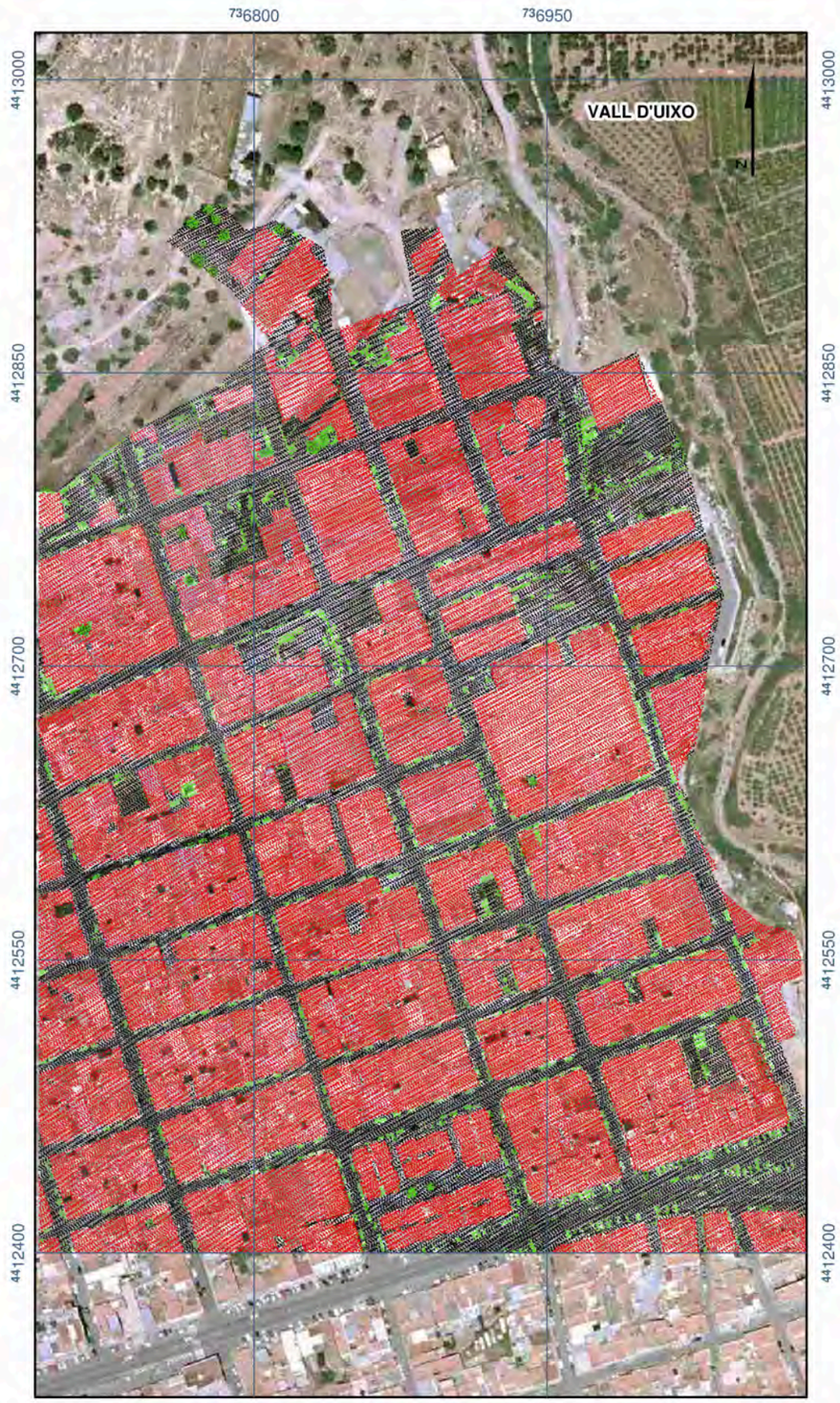
Departamento de Ingeniería Cartográfica, Geodesia y Fotogrametría

Clasificación automática obtenida tras ejecutar los algoritmos



Datum ETRS89. Proyección UTM. Huso 30

Clasificación verdadera obtenida manualmente por un operador



Datum ETRS89. Proyección UTM. Huso 30

DESARROLLO E IMPLEMENTACIÓN DE ALGORITMOS PARA CLASIFICAR DATOS LIDAR EN ÁREAS URBANAS

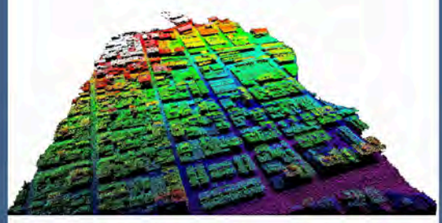
Leyenda

- Terreno
- Edificios
- Vegetación y pequeños objetos

AUTOR	José Sánchez Lopera
DIRECTOR DE TESIS	José Luis Lerma García
ZONA TEST	Vall d'Uixó
NÚMERO DE PLANO	1.6

CLASIFICACIÓN AUTOMÁTICA Y CLASIFICACIÓN VERDADERA

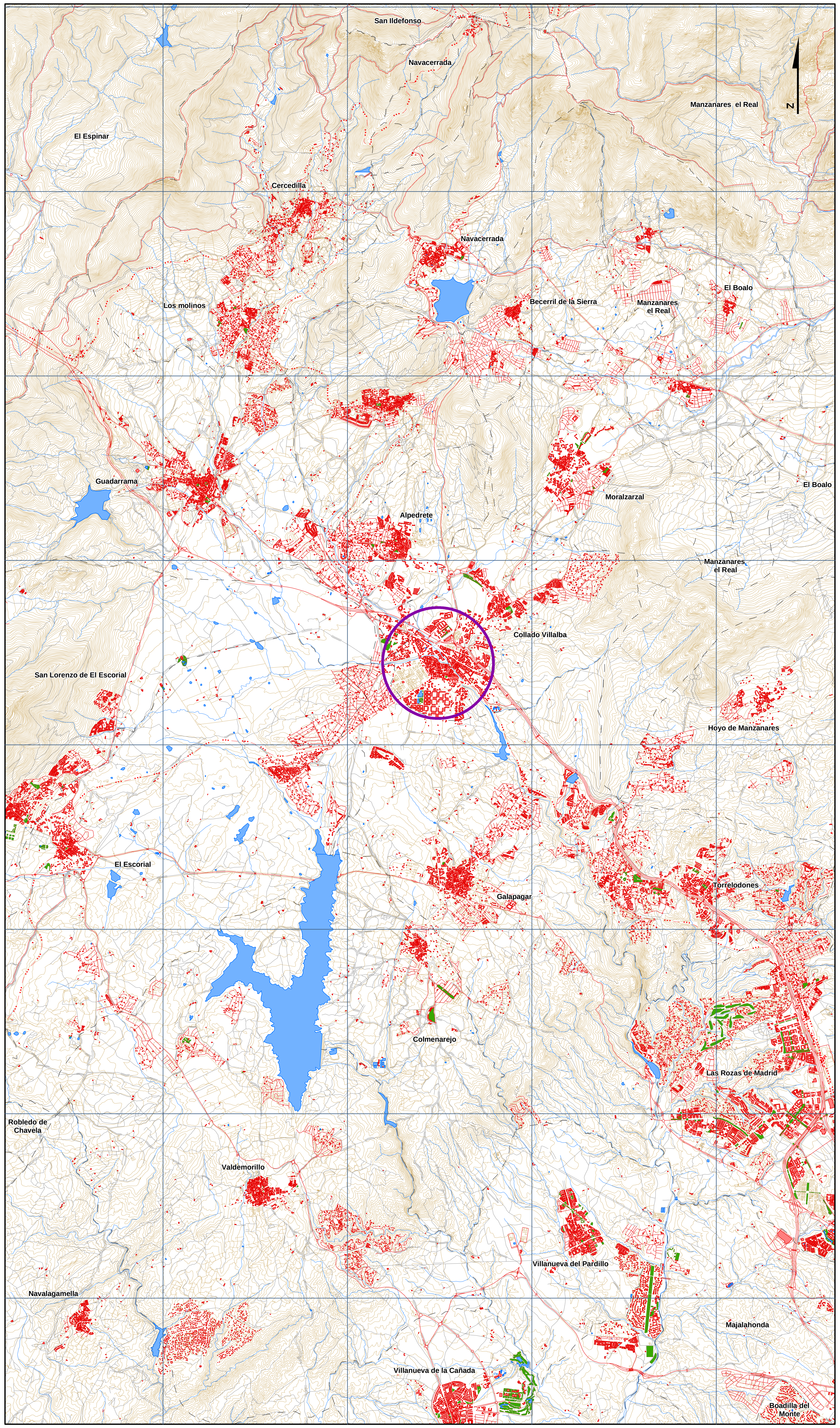
1:3.000



AGOSTO 2015

Universitat Politècnica de València

Departamento de Ingeniería Cartográfica, Geodesia y Fotogrametría



1:100.000

0 2.500 5.000 10.000 15.000 m

Datum ETRS89. Proyección UTM. Huso 30

1:25.000

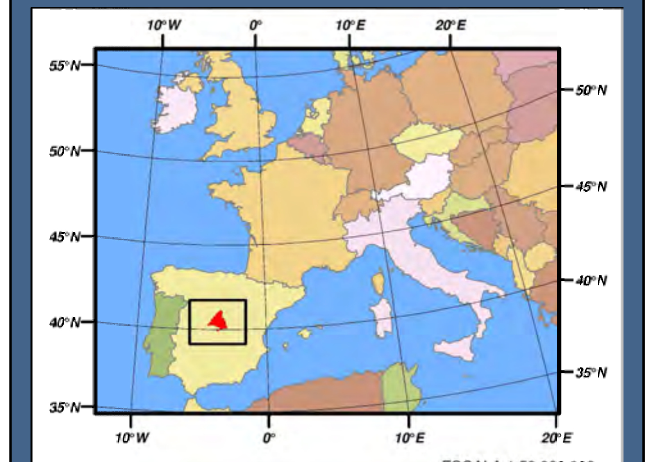
1.000 500 0 1.000 2.000 3.000 m

Datum ETRS89. Proyección UTM. Huso 30

DESARROLLO E IMPLEMENTACIÓN DE ALGORITMOS PARA CLASIFICAR DATOS LIDAR EN ÁREAS URBANAS

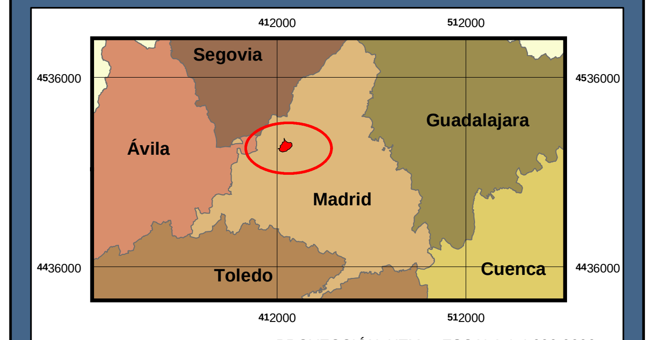
MAPAS DE SITUACIÓN

ÁMBITO EUROPEO



ESCALA 1:50.000.000

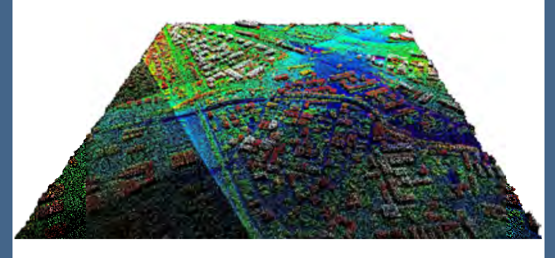
ÁMBITO COMUNITARIO



PROYECCIÓN: UTM ESCALA 1:4.000.0000

AUTOR	José Sánchez Lopera
DIRECTOR DE TESIS	José Luis Lerma García
ZONA TEST	Collado Villalba
NÚMERO DE PLANO	2.1

MAPA DE SITUACIÓN

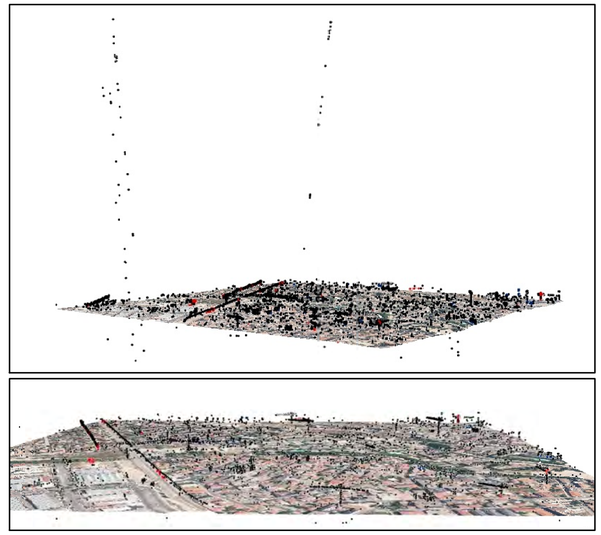


AGOSTO 2015

Universitat Politècnica de València

Departamento de Ingeniería Cartográfica, Geodesia y Fotogrametría

RESULTADOS OBTENIDOS. IMÁGENES 3D



DESARROLLO E IMPLEMENTACIÓN DE ALGORITMOS PARA CLASIFICAR DATOS LIDAR EN ÁREAS URBANAS

AUTOR	José Sánchez Lopera
DIRECTOR DE TESIS	José Luis Lerma García
ZONA TEST	Collado Villalba
NÚMERO DE PLANO	2.2

Detección de errores groseros. Algoritmo C: Análisis de variaciones de desnivel en vecindario próximo. Multiproceso con parámetros adaptativos.	
Ensayo	C 5
PROCESO 1: PARÁMETROS UMBRALES	
Desnivel (IZ)	10 m
Tamaño regiones	15 m^2
Radio vecindario	5 m
Porcentaje (%)	85 %
PROCESO 2: PARÁMETROS UMBRALES	
Desnivel (IZ)	10 m
Tamaño regiones	15 m^2
Radio vecindario	5 m
Porcentaje (%)	95 %

ERRORES COMETIDOS	
Error de comisión	5,140 %
Error de omisión	7,390 %
Media de errores	6,265 %

Leyenda

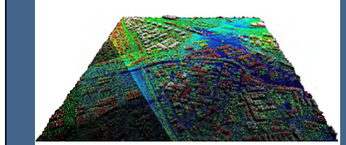
- Errores groserors correctamente clasificados
- Error de omisión
- Error de comisión

0 25 50 100 150 m

Datum ETRS89. Proyección UTM. Huso 30

DETECCIÓN DE ERRORES GROSEROS

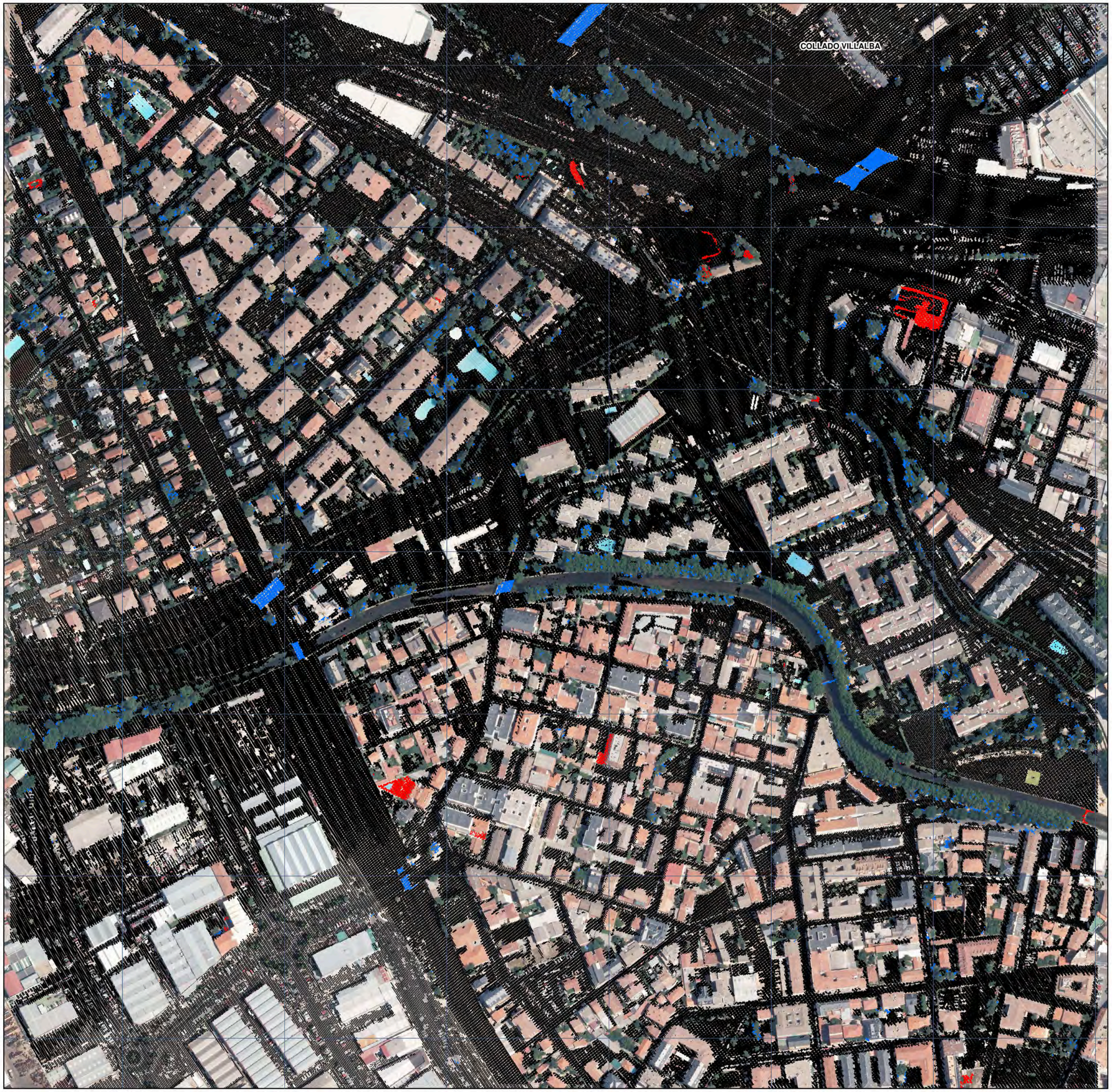
1:2.500



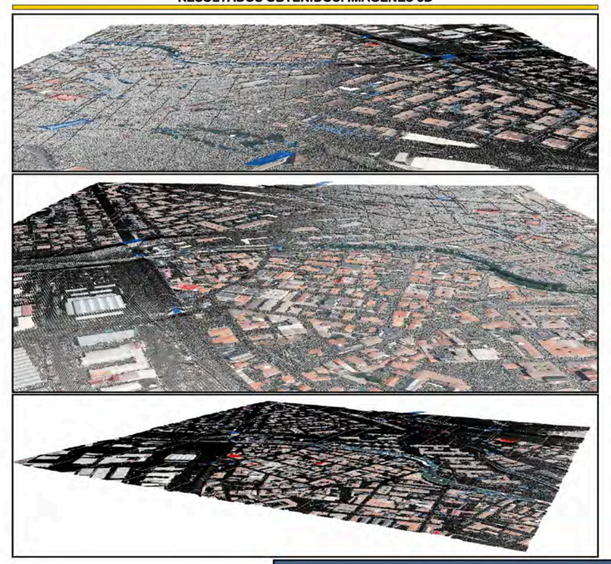
AGOSTO 2015

Universitat Politècnica de València

Departamento de Ingeniería Cartográfica, Geodesia y Fotogrametría



RESULTADOS OBTENIDOS. IMÁGENES 3D



DESARROLLO E IMPLEMENTACIÓN DE ALGORITMOS PARA CLASIFICAR DATOS LIDAR EN ÁREAS URBANAS

AUTOR	José Sánchez Lopera
DIRECTOR DE TESIS	José Luis Lerma García
ZONA TEST	Collado Villalba
NÚMERO DE PLANO	2.3

Detección automática de puntos pertenecientes al terreno. Algoritmo B: densificación progresiva mediante interpolación selectiva combinada con segmentación y aplicación de reglas de decisión.

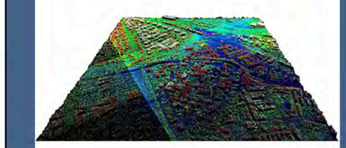
Ensayo	B 2-1, B 2-2, B 2-3 y B 2-4
PARÁMETROS UMBRALES	
IZ en segmentación	10 m
Tamaño regiones	0,25 m^2
Tamaño celda	1 m
B	3
L_Bloque	10 m, 25 m, 50 m y 100 m
U_IZ	0,4 m

B: Exponente de ponderación en la interpolación.
L_Bloque: Longitud del lado de los bloques necesarios para generar el terreno provisional inicial.
U_IZ: Umbral en desnivel con respecto a la superficie de referencia interpolada.

ERRORES COMETIDOS	
Error de comisión	0,98 %
Error de omisión	0,34 %
Media de errores	0,66 %

DETECCIÓN DEL TERRENO

1:2.500



Leyenda

- Terreno correctamente clasificado
- Error de omisión
- Error de comisión

0 25 50 100 150 m

Datum ETRS89. Proyección UTM. Huso 30

AGOSTO 2015

Universitat Politècnica de València
Departamento de Ingeniería Cartográfica, Geodesia y Fotogrametría

IMÁGENES DE DETALLE. IMÁGENES 3D

IMAGEN DE DETALLE 1

IMAGEN DE DETALLE 2

IMAGEN DE DETALLE 3

IMAGEN DE DETALLE 4

IMAGEN DE DETALLE 5

Leyenda

- Puente correctamente clasificado
- Error de omisión
- Error de comisión

0 25 50 100 150 m

Datum ETRS89. Proyección UTM. Huso 30

DESARROLLO E IMPLEMENTACIÓN DE ALGORITMOS PARA CLASIFICAR DATOS LIDAR EN ÁREAS URBANAS

AUTOR	José Sánchez Lopera
DIRECTOR DE TESIS	José Luis Lerma García
ZONA TEST	Collado Villalba
NÚMERO DE PLANO	2.4

Detección de puentes. Algoritmo B: localización de bordes y análisis direccional de la continuidad estructural. Multiproceso con parámetros umbrales adaptativos

Ensayo	B 4-2
PROCESO 1: PARÁMETROS UMBRALES	
Radio vecindario	2 m
Desnivel	3,5 m
U_%	20%
Porcentaje (%)	85%
Radio sectores	18 m
Radio_S	6 m
U_S_%	95%
Amplitud sectores	10 g
PROCESO 2: PARÁMETROS UMBRALES	
R_V_1	8 m
R_V_2	4 m
Amplitud_Sectores_2	50 g
Desnivel	2 m

R_V_1: Radio del vecindario cilíndrico sobre el que se ejecuta el proceso 2.

R_V_2: Radio del vecindario cilíndrio necesario para detectar puntos de borde colindantes a zonas de sombra.

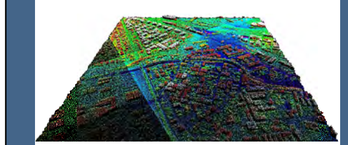
U_%: Umbral en porcentaje de desniveles dentro del vecindario que incumplen el desnivel umbral

U_S_%: Umbral en porcentaje de desniveles dentro de semisectores que incumplen el desnivel umbral

ERRORES COMETIDOS	
Error de comisión	5,65 %
Error de omisión	2,34 %
Media de errores	3,99 %

DETECCIÓN DE PUENTES

1:2.500

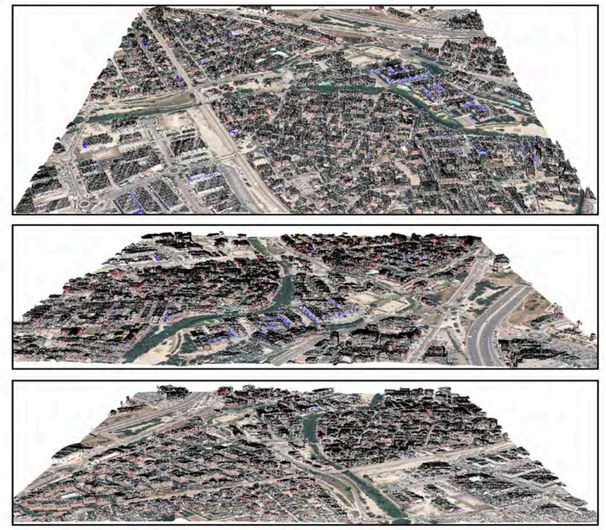


AGOSTO 2015

Universitat Politècnica de València

Departamento de Ingeniería Cartográfica, Geodesia y Fotogrametría

RESULTADOS OBTENIDOS. IMÁGENES 3D



DESARROLLO E IMPLEMENTACIÓN DE ALGORITMOS PARA CLASIFICAR DATOS LIDAR EN ÁREAS URBANAS

AUTOR	José Sánchez Lopera
DIRECTOR DE TESIS	José Luis Lerma García
ZONA TEST	Collado Villalba
NÚMERO DE PLANO	2.5

Detección de edificios así como vegetació y otros pequeños objetos. Algoritmo B: análisis de textura y clasificador angular.

Ensayo	B 2
PARÁMETROS UMBRALES. PROCESO 1: ANÁLISIS DE TEXTURAS	
Radio_V	6 m
Radio_S	3,5 m
Tamaño regiones	10 m^2
Amplitud angular	50^9
Nº Ptos_S	10
U_%	99 %
PARÁMETROS UMBRALES. PROCESO 2: CLASIFICADOR ANGULAR	
Radio_V	6 m
Tamaño regiones	10 m^2
Amplitud angular	50^9

Radio_V: Radio del vecindario cilíndrico.
Radio_S: Radio de los semisectores interiores.
Nº Ptos_S: Número mínimo de puntos pertenecientes a regiones pequeñas en semisectores.
U_%: Porcentaje de puntos mínimo pertenecientes a las regiones pequeñas en cada sector.

ERRORES COMETIDOS	
Error de comisión	1,15 %
Error de omisión	1,89 %
Media de errores	1,52 %

Leyenda

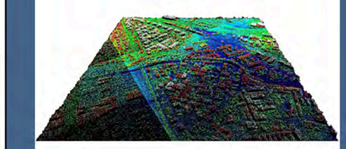
- Edificios correctamente clasificados
- Error de omisión
- Error de comisión

0 25 50 100 150 m

Datum ETRS89. Proyección UTM. Huso 30

DETECCIÓN DE EDIFICIOS

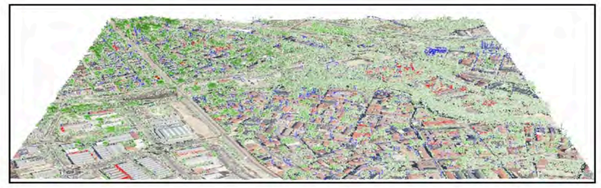
1:2.500



AGOSTO 2015

Universitat Politècnica de València
Departamento de Ingeniería Cartográfica, Geodesia y Fotogrametría

RESULTADOS OBTENIDOS. IMÁGENES 3D



DESARROLLO E IMPLEMENTACIÓN DE ALGORITMOS PARA CLASIFICAR DATOS LIDAR EN ÁREAS URBANAS

AUTOR	José Sánchez Lopera
DIRECTOR DE TESIS	José Luis Lerma García
ZONA TEST	Collado Villalba
NÚMERO DE PLANO	2.6

Detección de edificios así como vegetació y otros pequeños objetos. Algoritmo B: análisis de textura y clasificador angular.

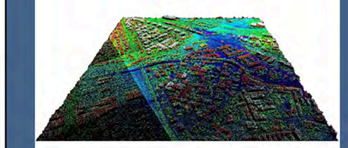
Ensayo	B 2
PARÁMETROS UMBRALES. PROCESO 1: ANÁLISIS DE TEXTURAS	
Radio_V	6 m
Radio_S	3,5 m
Tamaño regiones	10 m^2
Amplitud angular	50^0
Nº Ptos_S	10
U_%	99 %
PARÁMETROS UMBRALES. PROCESO 2: CLASIFICADOR ANGULAR	
Radio_V	6 m
Tamaño regiones	10 m^2
Amplitud angular	50^0

Radio_V: Radio del vecindario cilíndrico.
Radio_S: Radio de los semisectores interiores.
Nº Ptos_S: Número mínimo de puntos pertenecientes a regiones pequeñas en semisectores.
U_%: Porcentaje de puntos mínimo pertenecientes a las regiones pequeñas en cada sector.

ERRORES COMETIDOS	
Error de comisión	2,94 %
Error de omisión	2,51 %
Media de errores	2,73 %

DETECCIÓN DE VEGETACIÓN Y OTROS PEQUEÑOS OBJETOS

1:2.500



AGOSTO 2015

Universitat Politècnica de València
Departamento de Ingeniería Cartográfica, Geodesia y Fotogrametría

Leyenda

- Vegetación y otros pequeños objetos correctamente clasificados
- Error de omisión
- Error de comisión

0 25 50 100 150 m

Datum ETRS89. Proyección UTM. Huso 30

Clasificación automática obtenida tras ejecutar los algoritmos

1:5.000

0 150 300 600 900 m

Datum ETRS89. Proyección UTM. Huso 30

Clasificación verdadera obtenida manualmente por un operador

1:5.000

300 150 0 300 600 m

Datum ETRS89. Proyección UTM. Huso 30

Leyenda

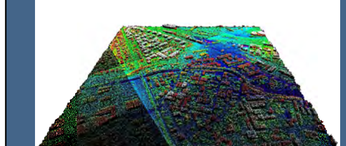
- Terreno
- Puentes
- Edificios
- Vegetación

DESARROLLO E IMPLEMENTACIÓN DE ALGORITMOS PARA CLASIFICAR DATOS LIDAR EN ÁREAS URBANAS

AUTOR	José Sánchez Lopera
DIRECTOR DE TESIS	José Luis Lerma García
ZONA TEST	Collado Villalba
NÚMERO DE PLANO	2.7

CLASIFICACIÓN AUTOMÁTICA Y CLASIFICACIÓN VERDADERA

1:5.000



AGOSTO 2015

Universitat Politècnica de València
Departamento de Ingeniería Cartográfica, Geodesia y Fotogrametría