Segmentación y clasificación de mallas 3D

Master Universitario en Automática e Informática Industrial

Universidad Politécnica de Valencia

ALUMNO: Borja Javier Herráez Concejo

DIRECTOR: Eduardo Vendrell Vidal

Índice

1.- INTRODUCCIÓN

El presente trabajo final de master pretende abordar el problema de la segmentación de mallas poligonales enfocada a modelos que representen edificios reales, es decir, ser capaces de reconocer características sobre modelos geométricos entendiendo por características esas porciones del modelo que difieren de la estructura principal, tales como salientes, depresiones, agujeros, etc. En el caso que nos ocupa, enfocado concretamente a edificios, el objetivo es diferenciar y reconocer los elementos que no son pared o fachada como puertas, ventanas, etc.

Según el diccionario de la lengua española la "segmentación" se define como el acto de dividir o formar partes o segmentos, siendo un segmento una porción o parte cortada o separada de una cosa, de un elemento geométrico o de un "todo". Este método se aplica en muchos ámbitos de la vida diaria, por ejemplo en economía se utiliza el término "segmentación de mercados" donde se segmenta una masa grande de consumidores en un grupo reducido que son los clientes potenciales [1]. En biología se utiliza el término para denotar la división de algunos animales y plantas en una serie de segmentos repetitivos [2]. En el ámbito de la informática, que es el que nos ocupa y más concretamente en la detección de elementos y características de un edificio, que es el objetivo planteado en esta tesina, se ha aplicado la segmentación a partir de diferentes supuestos según el formato del modelo disponible: imágenes, nubes de puntos y modelos tridimensionales poligonales (mallas 3D).

En el caso de disponer de imágenes, la segmentación se refiere a la partición de una imagen en un conjunto de regiones que la cubren. El objetivo en muchas tareas es que las regiones representen áreas significativas de la imagen, como por ejemplo, los cultivos, zonas urbanas o bosques en una imagen de un satélite. Cuando las regiones de interés no cubren toda la imagen, se puede seguir hablando de segmentación, pero en este caso hablamos de segmentación de las regiones de interés y regiones del fondo de la imagen a ignorar. La segmentación de imágenes tiene dos objetivos, el primero descomponer la imagen en partes para su posterior análisis y el segundo hacer un cambio de representación. Los pixeles de la imagen deben estar organizados en unidades de más alto nivel, es decir que estas agrupaciones tengan mayor significado o sean más eficientes para su análisis [3].

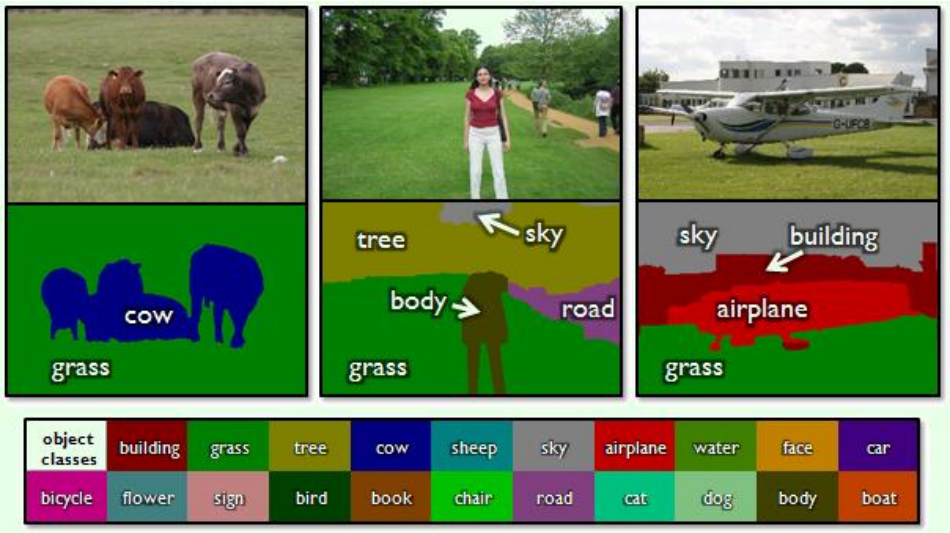


Ilustración 1: Ejemplo de segmentación de imágenes (Fuente: Jamie Shotton et.al. 2006).

Cuando se realiza un proceso de escaneado 3D de un objeto real, el resultado más habitual es el de una nube de puntos de la superficie del objeto escaneado. Se trata de un método comúnmente utilizado en arquitectura a la hora de realizar levantamientos de edificios existentes para su estudio [4]. La segmentación de una nube de puntos consiste en organizar, parametrizar y post-procesar los puntos 3D que contiene para agruparlos en elementos identificables. Esta organización depende de las ciertas características homogéneas que presentan los puntos de la nube, un criterio de homogeneidad seria la curvatura o las caras planas definidas por un conjunto de puntos [5].

Usualmente las nubes de puntos reciben un tratamiento posterior de reconstrucción de cara a obtener un modelo geométrico poligonal o malla 3D triangular del objeto escaneado. Se trata del método más usual a la hora de representar un objeto en un computador.

Ilustración 2: Segmentación de nubes de puntos (Fuente: Hui Lin, Jizhou Gao et al. 2013).

Segmentar una malla 3D consiste en la separación de la misma en partes diferentes, aunque no significa necesariamente separar la malla en sí, si no que también se utilizan los procedimientos necesarios para llegar a esa clasificación de partes que es lo que indicara a qué grupos (elementos o características) pertenece cada triangulo. Posteriormente se asignará a

cada grupo una etiqueta, dependiendo de características geométricas. Es habitual identificar los triángulos pertenecientes a una característica o elemento determinado mediante un color, que acabará identificando la misma parte semántica del objeto.

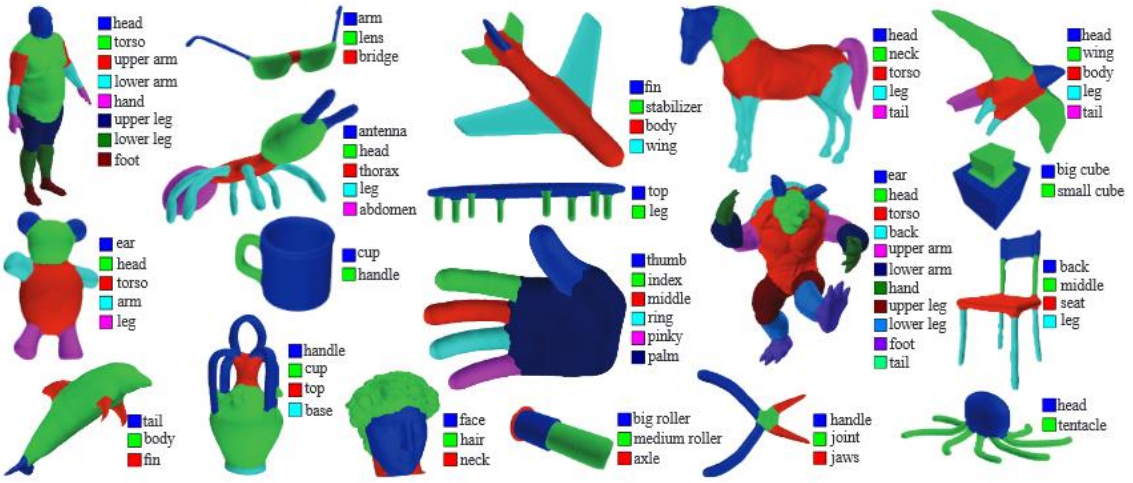


Ilustración 3: Segmentación de mallas 3D (Fuente: E. Kalogerakis et al. 2010).

Este trabajo fin de master surgió con la idea de continuar y ser una extensión de un proyecto colaborativo entre el Grupo de Robótica del Instituto Universitario de Automática e Informática Industrial (AI2) de la Universitat Politècnica de València y el Dipartamento di Architettura de la Università degli Studi di Firenze [6]. En concreto, esta colaboración se plasma en el apoyo tecnológico por parte del AI2 a los procesos de levantamiento que llevan a cabo en Dipartamento di Architettura de la Università degli Studi di Firenze.

Un levantamiento arquitectónico consiste en el conocimiento completo de lo relativo a la estructura de un edificio por lo que en este sentido los sistemas informáticos pueden ayudar en el almacenamiento y gestión de la información como pueden ser datos, dibujos, planos, etc. El trabajo de colaboración se ha plasmado sobre el estudio realizado en el pueblo de Pietrabuona en la región de la Toscana, Italia.

Para este propósito, se realizó una aplicación que relaciona un sistema de información con los modelos 3D de esta ciudad. A partir de los modelos de los edificios obtenidos con escáneres laser, la nube de puntos resultantes se trataron con un software de poligonización que los transformó los puntos en una malla triangular. Por otro lado el sistema de información utilizado en primera instancia es un GIS (Geographic Information System) diseñado para capturar, almacenar, analizar, manejar y presentar todo tipo de datos geográficos [7]. A partir de este esquema se diseñó una aplicación que permitía por un lado introducir información a los modelos (“Back-end”, mostrado en la ilustración 4) seleccionar las partes de la malla 3D y la posterior introducción de información relativa a éstas, lo que conforma un proceso de etiquetado semántico manual, es decir, asignar un conjunto de información a un grupo de triángulos que forman una parte de la estructura del edificio. El problema radica en que este proceso es costoso temporalmente hablando para el usuario y requiere de mucha repetición.

Por otra parte existe un “Front-end”, en el que se permite la visualización de la información introducida en el sistema. Para la clarificación del sistema total se muestra la siguiente imagen.

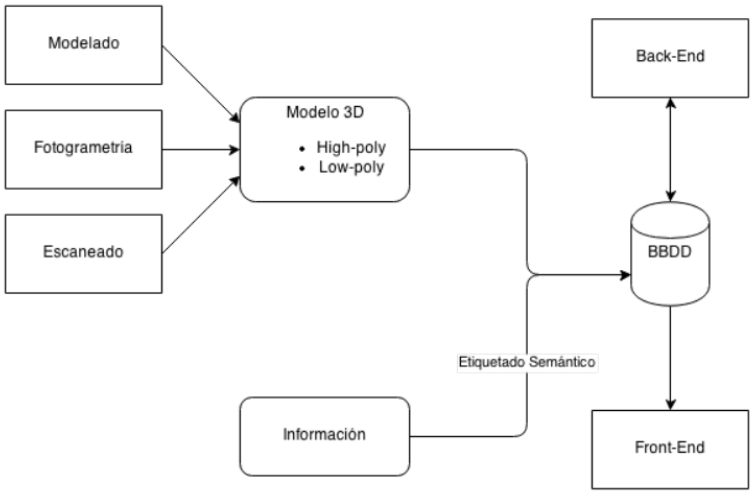


Ilustración 4: Esquema general funcionamiento (Fuente:”Aproximación a la gestión de modelos 3D para el levantamiento arquitectónico”.Andrés García. 2014).

La actual tesina de master pretende ser una aproximación a un etiquetado semántico automático, que sea capaz de reconocer las partes del objeto tridimensional y así se les pueda asignar información según su tipo, permitiendo que en un futuro pueda añadirse al desarrollo anterior y reduzca los tiempos que requiere el usuario para realizar este proceso de manera manual.

Para ello se ha realizado un algoritmo que es capaz de identificar y etiquetar a un conjunto de triángulos los cuales forman parte de un elemento de la fachada del edificio u objeto 3D. En este caso se entiende por “etiqueta” al nombre que describe ese elemento, por ejemplo “puerta”, “ventana” o “pared” y que está representado por un color concreto.

2.- ESTADO DEL ARTE

El principal problema de este trabajo fin de master es el de abordar el problema de reconocimiento de características determinadas en una malla 3D. La palabra "característica" (*feature* en inglés) tiene diferentes significados según el contexto dependiendo del dominio específico. Por ejemplo, en diseño puede referirse a una *muesca*, mientras que en manufacturación se refiere a *huecos* o *agujeros*, por otro lado en inspección la palabra "característica" se usa como *dato* o *referencia* de una parte. La clasificación de características es totalmente dependiente de la aplicación. Es muy difícil hacer una clasificación de características independiente de la aplicación [8]. Algunas de las definiciones que se pueden encontrar en la bibliografía son las siguientes:

- "Una característica es una entidad usada en el razonamiento del diseño, ingeniería o manufacturación de un producto" [9].
- "Una forma geométrica o entidad cuya presencia o dimensiones son requeridas para realizar al menos una función CIM y cuya disponibilidad como primitiva permite que ocurra el proceso de diseño" [10].
- Una característica también puede ser definida generalmente como un conjunto de entidades geométricas que tienen patrones topológicos y geométricos únicos a esa característica [11].

Emad S. Abouel Nasr y Ali K. Kamrani [12] estipulan que la base de estas definiciones es que las características representan el significado de ingeniería de la geometría de la parte, ensamblado u otra actividad de manufactura. Con ello, aportan una definición propia en la que una característica es una constituyente física de una parte diseñada mapeable a una forma genérica y que tiene relevancia en la ingeniería.

El problema que se plantea en este trabajo fin de master consiste entonces en detectar estas características sobre las mallas tridimensionales, utilizándose para ello un método de segmentación que se explicará en el capítulo 3 de este documento.

2.1 Segmentación

Como ya se ha dicho anteriormente la segmentación se define como el acto de dividir o formar partes o segmentos. En el caso que nos ocupa el objetivo es separar la malla 3D en partes significativas para entender y reconocer formas o características de la misma.

En [13] se indica que la segmentación de mallas, y de forma más general la segmentación de formas, se puede interpretar de manera puramente geométrica o de manera más orientada a la semántica. La primera separa la malla en un número de conjuntos que son uniformes respecto a una propiedad, en la segunda el objetivo es identificar partes que corresponden con características relevantes de la forma o volumen del objeto.

Las mallas 3D y la segmentación de éstas se describe de manera formal en [14] de la siguiente manera:

- Una malla tridimensional M está definida por una tupla {V, E, F}, donde los vértices son $V = \{p_i \mid p_i \in \mathbb{R}^3, 1 \leq i \leq m\}$, las aristas son $E = \{e_{ij} = (p_i, p_j) \mid p_i, p_j \in V, i \neq j\}$ y F son las caras, usualmente triángulos $F = \{f_{ijk} = (p_i, p_j, p_k) \mid p_i, p_j, p_k \in V, i \neq j, i \neq k, j \neq k\}$.

También define la segmentación de una malla como:

- Sea M una malla 3D y S el conjunto de elementos de la malla que pueden ser V, E o F. Una segmentación Σ de M es el conjunto de sub-mallas $\Sigma = \{M_0, \ldots, M_{k-1}\}$ inducidas por una partición de S en k sub-conjuntos disjuntos.

Así pues, la segmentación de una malla depende del criterio que se siga para determinar que elementos de ésta (el conjunto "S" de vértices o aristas, etc.) forman parte del mismo conjunto. En la literatura existen diferentes tipos de algoritmos para conseguir este propósito y este método se aplica a otros ámbitos relacionados con el que nos ocupa como la segmentación de imágenes o la segmentación de nubes de puntos.

2.1.1 Segmentación de imágenes

Es un mecanismo usado para dividir la imagen en múltiples segmentos. El proceso también ayuda a encontrar regiones de interés en una imagen específica. El objetivo primario es el de hacer la imagen más simple y con más significado. Cada segmento simbolizara un tipo de información, ya sea color, intensidad o textura. Generalmente el proceso de segmentación asignara un valor a cada pixel de la imagen para que sea fácil diferenciar cada región encontrada.

La segmentación de imágenes sigue siendo hoy en día un problema desafiante y uno de los pasos clave para el entendimiento de las imágenes. Una gran variedad de aplicaciones como reconocimiento de objetos, codificación o anotación de imágenes usan el algún momento un algoritmo de segmentación, y campos como el médico o militar lo utilizan diariamente. Existen varias técnicas para segmentar imágenes, algunas de las cuales se explicaran a continuación.

2.1.1.1 Segmentación de Imágenes por umbralización

Estas técnicas se basan en la umbralización de histogramas para segmentar las imágenes. Umbralizar significa establecer límites/umbrales según los cuales se realiza una opción u otra. Los histogramas se definen como una gráfica que indica el número de ocurrencias de unos valores determinados. En el caso de imágenes, se refiere usualmente al número de apariciones de los valores de color que pueden tener los píxeles.

Se hace un estudio de estas técnicas en [15]. Por un lado, se tiene el *método de la media* donde se utiliza el valor de la media de los pixeles como umbral. Esta técnica funciona bien en los casos donde la mitad de los pixeles de la imagen pertenecen al objeto y la otra mitad al fondo, aunque se trata de una técnica poco utilizada.

Por otro lado, la técnica "*p-tile*" es un método basado en el histograma de valores de grises, que asume que los objetos de la imagen son más brillantes que los objetos del fondo y que ocupan un porcentaje "P" determinado del área de la imagen. En este caso, el "threshold" o umbral se define como el nivel de gris que se corresponde con un área dentro del objeto lo más cercana posible a "P".

El siguiente método que se explica es el HDT (Histogram Dependent Technique), cuyo éxito depende de la estimación del umbral que separa la imagen y que este sea capaz de dividirla en dos segmentos homogéneos.

Por último, otro procedimiento de umbralización que se revisa es la técnica EMT (*Edge Maximization Technique*), utilizada cuando hay más de una región homogénea en la imagen o hay un cambio de iluminación entre el objeto y el fondo. Este procedimiento se basa en la detección de bordes con el objetivo de separar las regiones de la imagen.

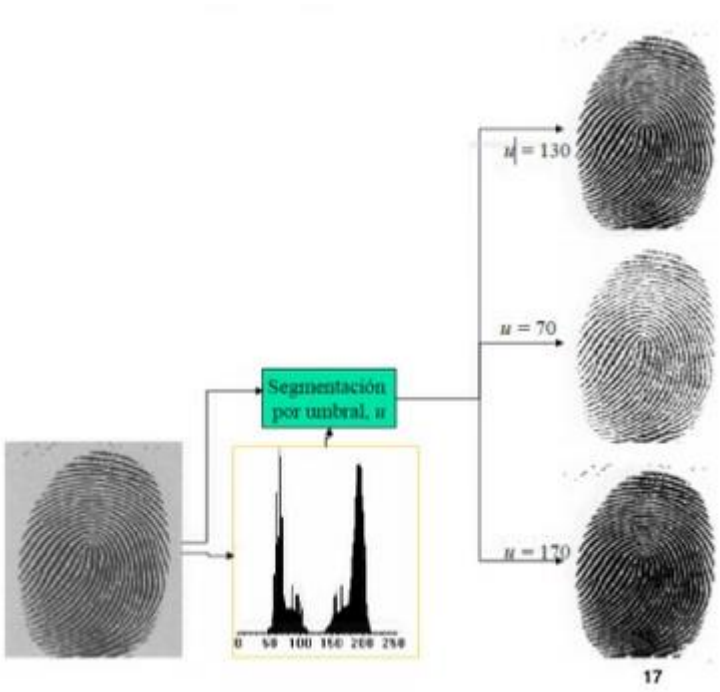


Ilustración 5: Ejemplo umbralización por histograma. (Imagen cortesía de José C. Benítez P. Universidad Tecnológica de Perú).

2.1.1.2 Segmentación de imágenes basadas en regiones

La segmentación de imágenes basada en regiones es un procedimiento sencillo y robusto contra el ruido. El "ruido" es definido como una variación aleatoria de brillo y color en las imágenes, debido a los dispositivos de entrada como por ejemplo una cámara.

Los procedimientos aquí explicados dividen la imagen en regiones basándose en criterios predefinidos. Se pueden dividir en 3 categorías: crecimiento de regiones, división de regiones y fusión de regiones [16].

Existe un método no supervisado cuyo fin es segmentar texturas utilizando métodos de nivel establecido y estadísticas de texturas. La implementación incluye un paso de selección de características, con el fin de ajustar los pesos de cada una para la segmentación usando la medida de divergencia de Kullback-Leibler [17]. En la fase experimental utiliza un histograma de filtro de respuesta para calcular el número de distribuciones [18].

Otro algoritmo existente es el "mean-shift clustering". Primero se extrae el color, textura y posición en la característica de cada pixel de la imagen. Después se crean los grupos (clusters) según esas características, utilizando el algoritmo descrito anteriormente. A continuación se etiqueta cada región y por último se crean los segmentos de la imagen según estas etiquetas. [19][20].

Otro procedimiento utiliza un grafo no dirigido con pesos para representar la imagen, donde cada nodo representa una región y no un pixel, lo que reduce la complejidad del algoritmo. Los pesos entre los nodos representan el grado de parecido entre regiones vecinas, que son calculados en función de los parecidos de intensidad entre estas. Por último, la segmentación se consigue haciendo una bipartición del grafo de forma recursiva según la minimización del método de corte normalizado, que consigue superar el problema de la sobre-segmentación [21].

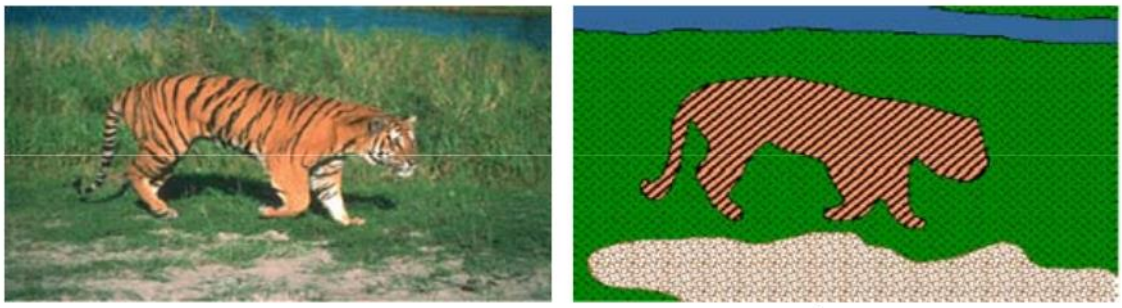


Ilustración 6: a) Imagen original, b) Imagen Segmentada por regiones. (Imagen H. G. Kaganami and Z. Beij).

2.1.1.3 Segmentación de imágenes basadas en bordes

Esta clase de métodos dividen la imagen en objeto y fondo. La división de la imagen se obtiene observando los cambios de intensidad en los pixeles de la imagen. Dos de los métodos principales son los histogramas de escala de grises y la computación del gradiente [22].

Un procedimiento posible consiste en la mezcla de crecimiento de regiones y detección de bordes. Este método híbrido consigue solventar los errores de cada técnica por separado. Se utiliza el crecimiento de regiones para detectar los pixeles de las aristas de la imagen, por otro lado utiliza derivadas de segundo orden para la detección de aristas [23].

De manera similar, se utilizan los modelos de Markov para implementar un sistema hibrido de detección de bordes y regiones para imágenes en color [24].

Asimismo se pueden combinar operadores morfológicos con técnicas de crecimiento de regiones. Primero se utilizan operadores de cierre, y después se detectan los bordes usando un detector de dilatación residual, obteniendo la imagen de bordes. Tras eso, se aplican las semillas y se inicia el proceso de expansión de regiones en la imagen de bordes. Por último, se compara el resultado de esta expansión de regiones con la detección de bordes para obtener las líneas de la imagen, que definen los segmentos en los que se divide [25].

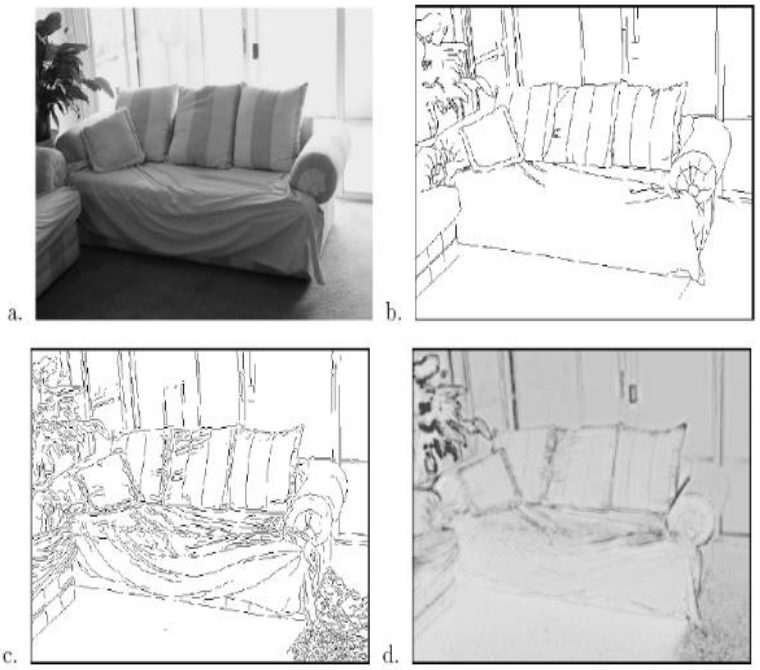


Ilustración 7: a) Imagen Original, b) c) d) Aplicación de varias técnicas de detección de bordes [23].

2.1.1.4 Segmentación de imágenes con lógica borrosa

Existe un algoritmo que integra las características de imagen fuzzy en matemática morfológica para después aplicar esta matemática en la detección de bordes [26]. La lógica borrosa también se aplica para segmentar imágenes e incorpora las relaciones espaciales entre los píxeles [27].

Por otro lado, una nueva técnica basada en la conectividad fuzzy usa pesos dinámicos, e introduce el algoritmo DyW (del inglés *"fuzzy connectedness using Dynamic Weights"*) para la segmentación de imágenes médicas [28].

El algoritmo que se utiliza en [29] está basado en la diferencia de características y la disimilitud borrosa. Primero se transforma la imagen a escala de grises, después se crea el histograma, a continuación se crean los clusters, para después aplicar la técnica de Fuzzy C-means [30] a

cada cluster. Por último, se aplican operadores morfológicos como dilatación o erosión y expansión de regiones a la imagen resultante y así obtener el resultado final.

2.1.1.5 Segmentación de imágenes por redes neuronales

Las redes neuronales artificiales representan un tipo de computación que imita la manera en la que el cerebro realiza computaciones. Son adecuadas para funciones no lineales y reconocimiento de patrones. Se usan en campos como la ingeniería aeroespacial, defensa, en el campo financiero, manufactura, robótica, telecomunicaciones, etc. [31] [32].

La red neuronal "Fast Learning Artificial Neural Network (FLANN)" [33], es utilizada para segmentar imágenes. Esta red neuronal adapta rápidamente los datos de entrada que se le pasan y genera representaciones de estos datos. El "FLANN" se utiliza en [34] para segmentar imágenes en el espacio RGBSV, que es una combinación del RGB y el HSV, el método descrito no pretende segmentar de forma completamente correcta, sino que se busca encontrar los puntos de interés de una escena de forma similar a como lo haría el ser humano. Así pues con el algoritmo de segmentación, se generan puntos de interés en la escena que sean posibles posiciones de objetos. En este caso, un "punto de interés" se define como los centros de los segmentos que se producen en el algoritmo de segmentación.

Las redes neuronales "fuzzy Hopfield" [35] también son utilizadas para segmentar imágenes. Una técnica existente procede de la siguiente manera: primero se hacen los clusters de la información que se le pasa, es decir, se calculan los niveles de gris, se calculan los centroides, se computan las distancias, se encuentran los nuevos centroides, luego se encuentra el nuevo valor de calidad de miembro del grupo, utilizando el algoritmo fuzzy C-medias, y por último se calcula la función objetivo [36].

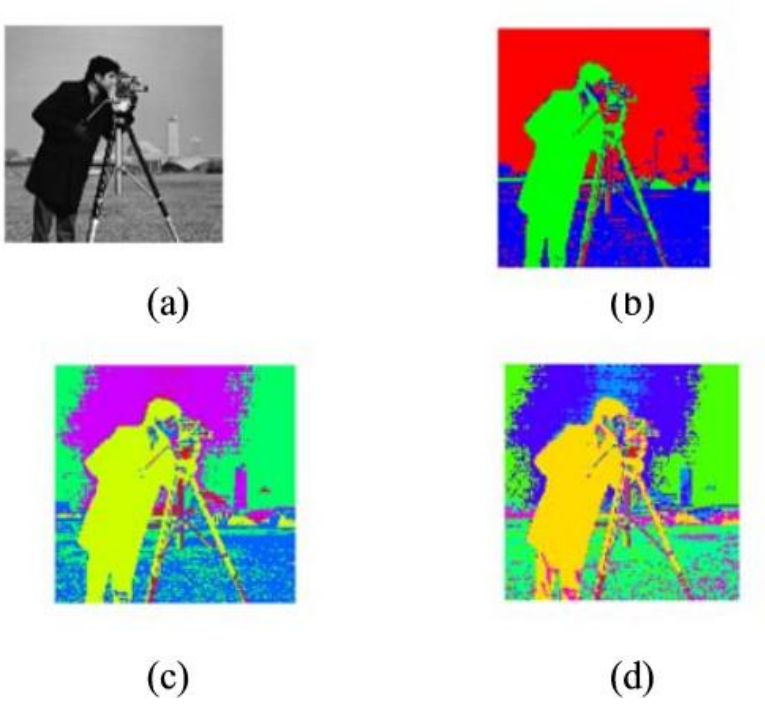


Ilustración 8: Imagen original y resultados de Farhad Mohaman Kazemi [36].

2.1.1.6 Segmentación de videos

La segmentación de videos se considera como una variación de la segmentación de imágenes, dado que un video no es más que una secuencia rápida de imágenes. Con los datos de un video, se puede comprobar la consistencia temporal entre "frames" consecutivos y usarse para suavizar errores de etiquetado. Este método es incorporado en [37], se utiliza un video como dato de entrada. En los videos, cuando la cámara o los objetos se mueven, destapan porciones de la escena que antes no se veían, y por el contrario tapan otras porciones que eran visibles anteriormente. Esto provee de información espacial a las clases y también de información de profundidad, que relaciona los objetos más al frente o más hacia el fondo de la imagen. Se explotan estas características para conseguir segmentar las imágenes del video.

Otra técnica se basa en construir un grafo 3D del video y utilizar el algoritmo descrito en [38], el cual segmenta imágenes con un método basado en grafos, con el objetivo de obtener una sobre-segmentación del volumen de video. Tras esto, se construye un grafo de regiones sobre la segmentación anterior e iterativamente se repite el proceso sobre múltiples niveles para crear un árbol jerárquico de segmentaciones. Con este árbol se puede seleccionar la granuralidad y post-segmentación, de manera contraria a volver a lanzar el algoritmo con diferentes parámetros [39].

También se han utilizado modelos de Markov para segmentar videos [40], se combina información de características basadas en las imágenes, audio y movimiento para segmentar el video. Los modelos de Markov (hidden markov model en adelante HMM) proporcionan un marco unificador para estas características. Los modelos HMM contienen arcos entre los estados que permiten progresión entre ellos. Los parámetros del HMM se aprenden usando

datos de entrenamiento en forma de frames de otros videos con los valores de transición, audio, etc. Una vez entrenado el HMM, se puede usar para segmentar el video.

2.1.2 Nubes de puntos

Como ya se ha explicado anteriormente, las nubes de puntos son el resultado devuelto hoy en día por los escáneres laser. La introducción de estos escáneres ha facilitado la adquisición de datos 3D y ha aumentado su precisión, gracias a estos datos, surgieron una gran cantidad de aplicaciones como “Google Earth”, “Geoportail”, “iTowns” y “Elyx-3D”. Muchas de ellas utilizan modelos 3D realistas de las ciudades, estos modelos son útiles para muchas aplicaciones: planificación urbanística, simulación de respuestas de emergencia, documentación de herencia histórica de edificios, turismo virtual, planificación de itinerarios, etc.

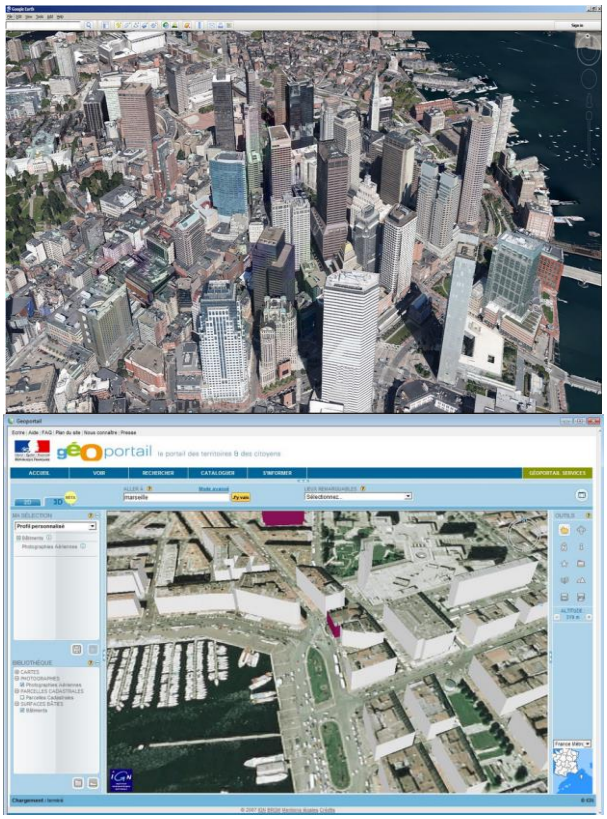


Ilustración 9: Modelos de Google Earth (Arriba), Modelos de GeoPortail (Abajo).

El principal objetivo, una vez obtenida la nube de puntos a partir del escaneado de un edificio, será la reconstrucción del modelo 3D del mismo, en este sentido dos algoritmos destacan en la bibliografía, la transformada de Hough y el algoritmo de RANSAC [41].

La transformada de Hough surgió para detectar líneas y curvas en imágenes [42], pero se extendió a tres dimensiones para detectar planos. Este algoritmo consiste en definir un plano usando 3 parámetros {θ, φ, ρ}, donde “ρ” es la distancia del plano al sistema de coordenadas origen y cos θ, cos φ son las direcciones del plano, . En este proceso se discretizan los valores de θ, φ entre 0 y 90 grados y 0 y 360 grados respectivamente y después se evalúan los valores

de “ρ” para los puntos de la nube. Después el triplete {θ, φ, ρ}, cada uno representa un único plano que se analiza para un recuento recurrente, los planos que llegan a cierto conteo se tienen en consideración como posibles planos. No obstante este proceso resulta en numerosos planos falsos, por lo que muchas implementaciones utilizan restricciones adicionales para aumentar su eficiencia, por ejemplo un método consiste en combinar la transformada de Hough con planos de suelo para eliminar falsos positivos en el algoritmo, después se refina todavía más aplicando la transformada de Hough en 2D para eliminar planos adicionales [43]. El estudio de las áreas que generan las proyecciones de clusters de puntos se añade como una restricción posterior a esta técnica en [44].

Ilustración 10: Reconstrucción conseguida por [43] utilizando la transformada de Hough.

Por otra parte, el algoritmo de RANSAC (RANdom SAmple Consensus) [45], muestrea de forma aleatoria e iterativa el menor número de puntos necesarios para determinar los parámetros del modelo. Entonces, estos parámetros se testean con el resto de puntos. El proceso se detiene cuando un número determinado de puntos del resto de datos encajan con los parámetros hallados. En este punto, los parámetros se reestiman con el nuevo conjunto de puntos. El método de RANSAC se aplica al conjunto entero de puntos y genera en su terminación un solo conjunto de parámetros. Este algoritmo también ha sido utilizado con el objetivo de estimar planos sobre nubes de puntos de edificios para su posterior reconstrucción tridimensional, pero de la misma manera que la transformada de Hough, puede generar falsos positivos, sobre todo si no se aplican restricciones adicionales o el conjunto de datos contiene ruido. Un ejemplo de uso básico del algoritmo de RANSAC se tiene en [46].

El problema de las nubes de puntos es que generalmente sufren desperfectos debido a imperfecciones en el escaneado como ruido, agujeros, etc. Este problema es solventado calculando propiedades locales para cada punto, como el valor de muestreo o la desviación estándar. Después el principio de RANSAC es utilizado para detectar primitivas en la nube de puntos, acto seguido se extraen puntos de los bordes de cada primitiva y se optimizan estos bordes. Finalmente se crea una malla con cada borde detectado y se ensamblan para formar la malla final [47].

En el segundo artículo, se hace un análisis de los valores propios de cada punto de la nube para determinar su planaridad, los objetos no planos son descartados del proceso para garantizar más robustez en el método. En el siguiente paso se utiliza el algoritmo fuzzy c-medias sobre el resto de puntos para agruparlos en segmentos de tejados basándose en la normal de su superficie.

Las nubes de puntos extraídas con escáneres LIDAR (Light Detection And Ranging) son utilizadas para reconstruir tejados de edificios a base de planos, después se extienden estos planos hasta el suelo con el objetivo de conseguir modelos simples de edificios. Una aplicación de esto es una mejora de la transformada de Hough utilizada para detectar múltiples planos al mismo tiempo, después se utilizan grafos de adyacencia y se consigue establecer relaciones entre estos planos para obtener resultados regulares [48]. Otra metodología consiste en el análisis de los valores propios de cada punto de la nube para determinar su planaridad, los objetos no planos se descartan del proceso para garantizar más robustez en el método. En el siguiente paso el algoritmo "fuzzy" c-medias es utilizado sobre el resto de puntos para agruparlos en segmentos de tejados tomando como criterio la normal de su superficie [49].

Un método semiautomático de reconstrucción de fachadas que junta la información de los escáneres laser y las imágenes de los edificios es definido en [50]. Primero se reconoce la estructura general de la fachada con los datos del escáner, luego se extraen los bordes de las imágenes utilizando el método "Canny" de detección de bordes [51] y la transformada de Hough, a continuación se comparan los resultados con el modelo obtenido del escáner para mejorar los bordes extraídos. Finalmente se utilizan las texturas con mejor visibilidad del conjunto de imágenes y se aplican sobre el modelo 3D.

Los modelos digitales de elevación son utilizados para segmentar y clasificar objetos de nubes de puntos sobre un entorno urbano. Primero se segmenta el suelo y se detectan los objetos como discontinuidades en el suelo. Entonces se utiliza el algoritmo *watershed* para separar los objetos. Finalmente se clasifican los objetos usando una máquina de vectores de soporte, "support vector marchine(SVM)" con características geométricas y contextuales [52].

Un trabajo sobre nubes de puntos enfocado a la manipulación robótica se presenta en [53]. Un sistema recibe como entrada una nube de puntos, los objetos son segmentados por vistas parciales y se reconstruyen encajando primitivas como planos, esferas o cilindros. Se calculan coeficientes geométricos que ayudan a reconstruir las partes perdidas del paso anterior. Por lo que al final se tiene un sistema hibrido, que combina los modelos geométricos con las mallas para crear una superficie suave.

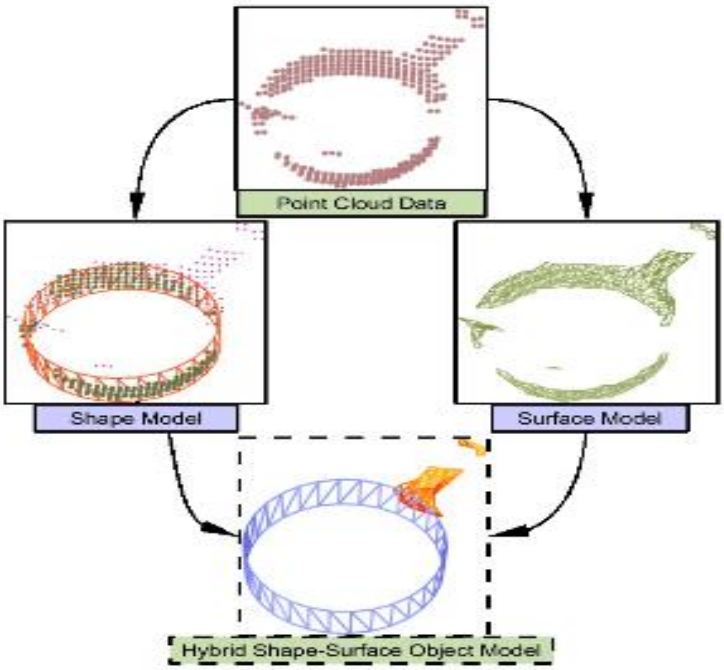


Ilustración 11: Sistema híbrido forma-superficie [53].

Otra técnica existente descompone la nube de puntos en primitivas como esferas o cilindros ayudándose del algoritmo de RANSAC, seguidamente un grafo llamado "grafo topológico" se construye representando las relaciones existentes entre las diferentes formas. Cada característica de la malla es representada como un grafo llamado "grafo restringido". Así se buscan coincidencias entre estos grafos restringidos y los grafos topológicos, y aplicando otras restricciones dentro del algoritmo de comparación de grafos, se consigue detectar las características [54]. El método que presenta este trabajo final de master está basado en parte en esta metodología.

2.1.3 Mallas 3D

2.1.3.1 ¿Qué es una malla 3D?

Las mallas 3D o mallas poligonales son representaciones tridimensionales de una superficie, que se basan en una colección de vértices unidos por aristas. Un vértice se define como un punto en el espacio tridimensional y que puede almacenar cierta información, una arista une 2 vértices y es un punto de unión entre 2 triángulos, 3 vértices diferentes unidos por aristas forman una faceta/triángulos, que es la unidad básica de representación de los polígonos tridimensionales. Algunos sistemas utilizan polígonos creados por 4 vértices, no obstante internamente se representen como 2 triángulos. Un conjunto de caras o polígonos unidos entre sí, forman una superficie.

Aunque se puede construir una malla de manera manual especificando vértices y caras, es mucho más eficiente utilizar el conjunto de herramientas software para gráficos 3d que existen actualmente. Estos programas hacen uso de la subdivisión y la extrusión. La subdivisión como su propio nombre indica, fracciona los triángulos en otros más pequeños añadiendo vértices y aristas. Por otro lado, la extrusión se aplica a un triángulo o a un conjunto de ellos. Siguiendo el contorno de estas facetas se crea una nueva con la misma forma, que conecta cada uno de sus vértices con una arista creando un volumen. Así pues la extrusión de un cuadrado sería un prisma.

Las mallas poligonales se pueden clasificar según el método que se utilice para almacenar su información.

En las tablas vértice-vértice, se indica para cada vértice su coordenada en el espacio y la lista de otros vértices con los que conecta. Es la representación más sencilla de implementar, pero se tiene que inferir la información de las aristas y las caras, ya que solo se tiene la información de los vértices, lo que dificulta los cálculos y operaciones [55].

Por otro lado están las tablas cara-vértice, que consiste en dos tablas, una tabla de caras con los vértices que componen cada una, y una tabla de vértices que indica para cada vértice su coordenada espacial y el conjunto de caras que lo incluye. Es la representación más usada. Tiene la ventaja de que cambios en la geometría del objeto, representan actualizaciones en la tabla de vértices, pero no en la tabla de caras, ya que la conectividad en las caras con los vértices existentes sigue siendo la misma. Las desventajas que presenta este método son por una parte, que la información de las aristas sigue siendo implícita y por otro lado, realizar las operaciones “división” o “unión” de mallas es más complejo con esta representación.

Otra representación muy utilizada es la representación “arista-alada” [56]. Se basa en proporcionar para cada arista los 2 vértices que la definen, las caras a las que pertenece y las 4 aristas más cercanas con las que conecta, 2 en el sentido horario y 2 en el sentido antihorario. Con este método se representan de manera explícita los vértices, aristas y caras del modelo, lo que ofrece una gran flexibilidad a la hora de cambiar la geometría de la malla, además las operaciones de unión y división se hacen de manera rápida.

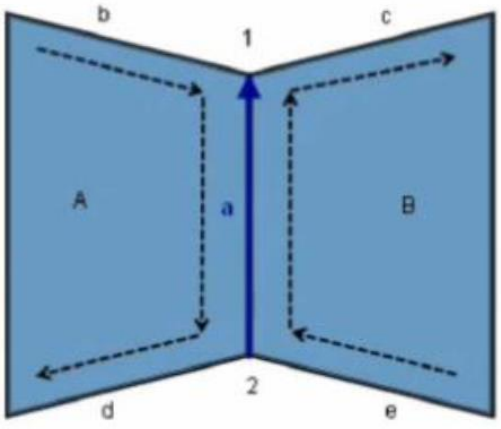


Ilustración 12: Representación arista alada (winged edge).

2.1.3.2 Region Growing

Primeramente se tienen los algoritmos de crecimiento/expansión de regiones o "región growing", que al igual que con imágenes, consisten en que los conjuntos de elementos solución se forman expandiendo diferentes elementos semilla ("seed" en inglés), que son vértices, triángulos o regiones. Luego un conjunto de reglas se aplica para saber si a partir de esta semilla se expande más el conjunto o se detiene la expansión. Este algoritmo es aplicado sobre imágenes en [57] y se extendió para poder aplicarlo sobre mallas 3D [58]. Otra solución consiste en que los puntos se etiqueten según su curvatura Gaussiana. Etiquetando los vértices con una alta curvatura negativa como "puntos de frontera", y seleccionando puntos que no son de frontera como semillas para comenzar el crecimiento de las regiones en [59].

2.1.3.3 Watersheds

Otro método para segmentar mallas consiste en utilizar el algoritmo "watersheds" (en español "línea de división de aguas") usado para imágenes 2D, es una técnica de segmentación que permite extraer las fronteras de las regiones de la imagen. Combina la detección de contornos y crecimiento de regiones al mismo tiempo. Para aclarar la explicación, se puede considerar una imagen con escala de grises como la imagen topográfica de un relieve terrestre, a cada pixel se le asociaría como "altura" el correspondiente nivel de gris, así los valores de grises más altos serian "montañas", mientras que los valores menores serian "valles". Tras esto se aplica un proceso de "inundación" desde los niveles más altos a los más bajos, la inundación continúa hasta que "valles" contiguos se unen, formando las líneas de unión que representan las fronteras de la imagen. Al utilizar este algoritmo para objetos 3D se aplica una función de "watersehed" $f:R^3 \rightarrow R$ que guía las cuencas para identificar las crestas de la malla. Hay una correspondencia 1 a 1 entre el mínimo de la función "f" y las cuencas [60]. Las aristas del objeto son utilizadas para llegar al mínimo a través de la "inundación" usando el ángulo diedro entre las aristas [61].

La técnica "Fast Marching Watersheds" segmenta la malla poligonal en partes visuales. Lo que busca el algoritmo es que se cumpla la teoría llamada "minima rule" que describe como la percepción humana descompone los objetos en sus diferentes partes. Esta regla define los límites de las partes sobre líneas de curvatura negativa mínima, y por otro lado, se busca mantener la robustez del algoritmo "watershed". Para ello se aplica un algoritmo basado en la estructura de datos "heap", para adaptar el "watershed" a mallas tridimensionales. También se define un mapa direccional de altura para la aplicación de la "minima rule" usando valores de curvatura local [62].

2.1.3.4 Métodos basados en grafos

Los procedimientos descritos a continuación utilizan grafos para adquirir información topológica de la malla tridimensional, y en función de estos infieren propiedades. Un grafo

especial llamado “Generalized Edge-Face Graph (GEFG)” es creado en [63] para identificar y clasificar las características del objeto, como son salientes, depresiones o agujeros. Este grafo contiene la información topológica de los contornos de la figura según sus caras y en base a él se hace un análisis de la conectividad de aristas y caras, esto junto con otras consideraciones geométricas permite diferenciar estas características.

En otro método, salientes Y depresiones del objeto se representan como “grafos de cavidad”, que reflejan la intersección o cavidad existente entre dos caras. Los nodos del grafo representan la orientación relativa de las caras. En siguientes pasos se realiza un análisis de estos grafos para la clasificación de características [64].

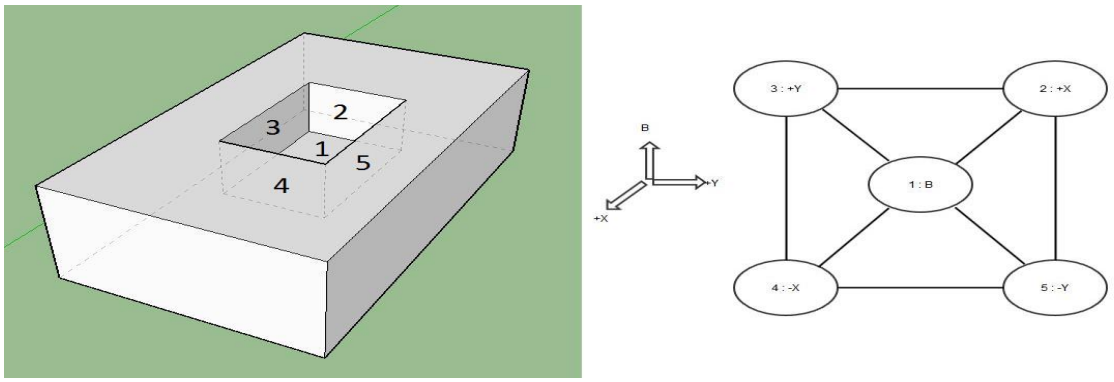


Ilustración 13: Grafo de características de M.Marefat y R.L.Kashyap [64].

2.1.3.5 Métodos basados en modelos

Se forman modelos a través de la superficie de la malla para conseguir segmentarla. Distribuciones de carga eléctrica sobre la superficie son utilizadas en [65], la carga es muy baja en áreas de mucha concavidad de la superficie, y muy alta en áreas de convexidad. Las áreas de segmentación son entonces las áreas con carga mínima. Los contornos son trazados donde los mínimos de cargas locales existen, estos contornos son los límites que definen la segmentación.

2.1.3.6 Métodos basados en esqueletos

El esqueleto de los objetos es obtenido para guiar la segmentación. La superficie de los objetos es simplificada mediante el método de contracción de aristas, más tarde un plano barre las aristas del esqueleto con el fin de hallar las áreas de segmentación del objeto. La intersección del plano y la malla resulta en uno o más contornos de polígonos. La forma en que estos contornos varían es examinada según cambia el plano de barrido. Se definen funciones paramétricas en estos contornos, los contornos que contienen puntos críticos en estas funciones, formarán parte de los contornos límite de la segmentación [66].

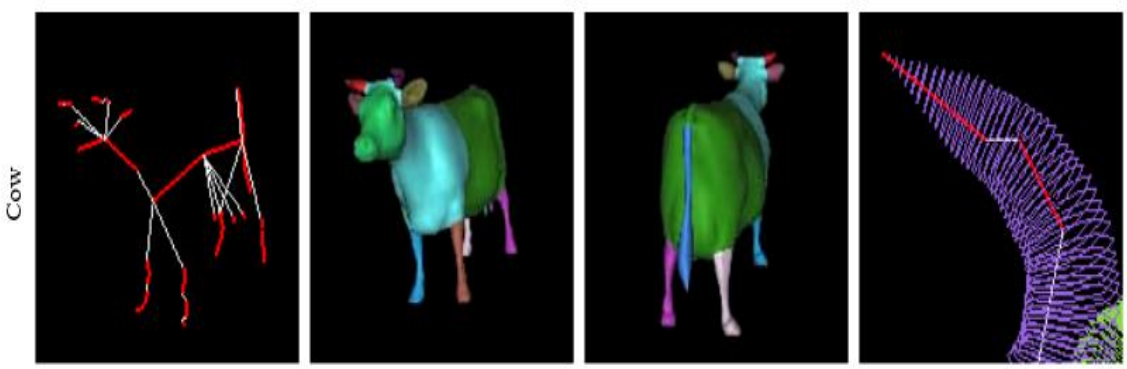


Ilustración 14: Ejemplo segmentación basada en esqueletos [66].

2.1.3.7 Métodos basados en clusters

Los métodos de *clusters* (en español racimo o grupo) y de crecimiento de regiones son similares. En los clusters los representantes van cambiando en base a una función de minimización, mientras que en los métodos de crecimiento de regiones, los elementos semilla se eligen al principio del algoritmo y no cambian.

Un algoritmo existente descompone la malla en parches. Se toma la decisión de si dos caras determinadas deben pertenecer al mismo parche. El criterio utilizado es la "proximidad" de las caras, en este caso la proximidad física, que se calcula como la suma de las distancias entre los centros de masa de las caras, y a ello se le suma la distancia angular, calculada como el ángulo diedro [67].

El algoritmo de "K-means" es utilizado para descomponer mallas, su implementación es sencilla y no muy robusta pero de bajo coste temporal [68]. El susodicho algoritmo se basa en agrupar una serie de muestras en "K" conjuntos diferentes, cada muestra es asociada al grupo cuya media de valores se aproxime más a ella misma. La "muestra" utilizada es una medida de la distancia entre caras adyacentes del modelo. Esta muestra se calcula como la suma de la distancia geodésica (distancia a lo largo de la superficie), más el ángulo dihedro (ángulo entre las normales de las caras).

De manera similar se aplica agrupamiento espectral ("spectral clustering") para segmentar modelos 3D, se construye una matriz de afinidad que contiene la probabilidad de cada pareja de caras de pertenecer a la misma partición de la malla, y se calcula un número "K" apropiado de vectores propios. Más tarde estos vectores son utilizados para obtener una inclusión de las caras de la malla en una esfera unitaria K-dimensional. A continuación se aplica el algoritmo "K-means" para la clasificación sobre los puntos obtenidos en la esfera. Este método obtiene buenos resultados sobre mallas sencillas, pero para mallas muy complicadas no produce resultados naturales con límites suavizados en las partes segmentadas [69].

En [70], se propone un método de segmentación y reconocimiento de mallas que incorpora conocimiento de mallas ya etiquetadas y segmentadas y mallas no segmentadas. El método propone un modelo CRF ("Conditional Random Field" en inglés o Campo Aleatorio Condicional)

que es un modelo estocástico utilizado para etiquetar secuencias de datos. Estudios revelan que un buen CRF para segmentación de mallas debe tener 2 partes, una que mida la relación entre la etiqueta y la malla, y otra parte que mida la consistencia entre la etiqueta y su vecindad. Además de lo anterior, se añaden al modelo términos correspondientes a las mallas no etiquetadas. Este modelo se utiliza para crear un conjunto de entrenamiento de un clasificador, dado que entrenar conjuntos grandes de datos con modelos CRF es una tarea costosa temporal y espacialmente, se adoptó el método "Virtual Evidence Boosting [71]. El método obtiene resultados satisfactorios, pero dependiendo de las mallas de prueba utilizadas y problemas específicos como el de etiquetar mallas tridimensionales de seres humanos presentan fallos.

2. 1.3.8 Métodos de análisis espectral

Estas técnicas proponen utilizar los valores propios de matrices definidas según la conectividad del grafo dual de la malla para segmentarla. Por ejemplo en [69] una matriz de afinidad "W" es definida, cuyos elementos son valores entre 0 y 1, que muestran la probabilidad de la faceta "i" y "j" de pertenecer al mismo conjunto. Con el fin de definir estos elementos emplean la función de distancia definida por [72] y un kernel exponencial. Después se construye una matriz "V" cuyas columnas son los "k" vectores propios más grandes de la matriz de afinidad normalizada "W_n", a continuación se aplica el algoritmo "k-means" explicado anteriormente sobre las filas de "V", usando como medida la distancia Euclidea estándar. Cada uno de estos vectores corresponderá a triángulos de la malla correspondientes al mismo conjunto.

2.1.3.9 Métodos basados en puntos críticos

Explotan los "puntos críticos", que son puntos de la malla donde hay características de tipo "saliente" y son utilizados para guiar la segmentación y distinguir los salientes entre ellos.

Los puntos de máximos locales según la función del saliente definido en el grafo dual de la malla son detectados en [73]. Estos son aprovechados para identificar los bordes de los salientes para después aplicar el algoritmo de corte mínimo [74] para separar el saliente del resto del objeto, la principal ventaja de este método es que identifica correctamente los salientes aún en presencia de mucho ruido, y su principal desventaja es que no funciona bien en mallas que no tengan salientes.

2. 1.3.10 Métodos para edificios

Aparte de las metodologías explicadas anteriormente, la bibliografía se centra en la reconstrucción e identificación de objetos sobre nubes de puntos. Las nubes de puntos son los datos básicos que extraen los escáneres 3D y sobre los que se han hecho multitud de métodos.

Se ha desarrollado un sistema que toma como entrada una nube de puntos representando una ciudad y un conjunto de objetos de entrenamiento, y como salida da la segmentación y etiquetado, es decir, cada punto de la nube es asociado con un segmento y cada segmento tiene una etiqueta semántica. El sistema procede en cuatro pasos, primero se genera una lista de localizaciones con objetos de potencial interés, después se predice para cada una de estas localizaciones cuales de sus puntos pertenecen al objeto y cuales al fondo de la escena, luego una serie de características describiendo la forma y contexto del objeto son extraídas para finalmente compararlos y clasificarlos según los ejemplos etiquetados de las muestras de entrenamiento [75].

Igualmente existen métodos que utilizan las nubes de puntos del escáner LIDAR (Light Detection And Ranging) para reconstruir tejados de edificios. Primero se hace un análisis de los valores propios para cada punto del tejado dentro de su vecindad de Voronoi, con esto se consigue separar los puntos planares de los no planares lo que aumenta la robustez del algoritmo. El segundo paso consiste en usar el algoritmo borroso de "K-medias" para agrupar los puntos en segmentos del tejado según las normales de la superficie. En el paso final se usa un algoritmo basado en múltiples parejas de líneas paralelas y perpendiculares para la reconstrucción, y obtener así modelos topológica y geométricamente correctos [41].

De la misma forma hay técnicas que además de utilizar nubes de puntos usan otro tipo de datos, un método semiautomático de reconstrucción de fachadas combina información de nubes de puntos obtenidas por láser y fotografías de corta distancia. Por un lado la extracción de líneas de las imágenes es muy precisa mientras que los puntos obtenidos por láser son más adecuados para detección de características planas. Al principio se usan estas características para generar un poliedro que será el modelo inicial de la fachada, posteriormente las líneas detectadas en las imágenes son empleadas y se comparan con las aristas del modelo para refinar los polígonos. Por último se seleccionan automáticamente texturas de diferentes imágenes para aplicarlas a la fachada y aumentar su visibilidad. [50][76].

Por otro lado la segmentación de modelos 3d de fachadas a través de escaneados laser ha ido creciendo en interés para la comunidad científica estos últimos años, como en la mayoría de casos los elementos de las fachadas son planos, se hace prioritario la detección y segmentación de estos elementos, para este propósito de reconstrucción, el algoritmo más utilizado es el algoritmo de RANSAC mencionado anteriormente, que estima de manera robusta los planos incluidos por conjuntos de puntos 3D como los existentes en nubes de puntos, en [46] se utiliza este algoritmo y se compara con planos extraídos de forma manual.

Un sistema de detección de ventanas sobre imágenes de fachadas utilizando datos de escáneres 3D es empleado en [77], donde se obtienen coordenadas esféricas de un entorno urbano, medidas de distancia para cada punto son utilizadas y así se calcula un umbral adaptativo que puede usarse para binarizar la imagen, después se aplican operaciones morfológicas de cierre (dilatación más erosión) para acabar con la detección de contornos de cada componente disjunta.

En cuanto a identificación de ventanas en [78] se propone otro método para identificarlas sobre nubes de puntos obtenidas con el LIDAR. El sistema separa primeramente los puntos 3D del entorno de los puntos de la fachada y entonces, proyecta estos puntos en un plano 2D

paralelo a la fachada del edificio. A continuación se aplica un algoritmo de límites y de inversión de puntos y entonces se segmentan las ventanas teniendo en cuenta también restricciones geométricas. En este punto se explota la simetría buscando correspondencias entre ventanas y la fachada del edificio, esto consigue reducir el tamaño de las muestras y facilitar la segmentación. Una vez encontradas las ventanas estas se usan para refinar la nube de puntos de la fachada.

También se han implementado métodos de segmentación de fachadas basados en gramáticas. Es el caso de [79] donde se manipulan los datos utilizados por los escáneres LIDAR, aplicando previamente un filtro para reducir el ruido, después se hace una descomposición de profundidad en capas, y así se convierten los datos 3D en datos 2.5D con lo que se reduce la complejidad. Para cada capa se aplica el algoritmo de segmentación relacionando cada capa con una gramática particular óptima. Por último se usan las formas segmentadas para extrusionarlas a lo largo de los mapas de profundidad y así se consigue la reconstrucción poligonal del edificio.

2.2. Resumen

Los métodos anteriormente explicados junto con otros se detallan en la siguiente tabla a modo de resumen de todo lo explicado anteriormente.

METODO	AUTOR	CARACTERISTICAS	CRITERIO
Region Growing	Besl and Jain	Polinomios de aproximación de orden variable, media y curvatura gaussiana.	Distancia de los puntos desde la superficie polinómica, estimaciones derivadas de puntos cercanos a la superficie polinómica derivada.
	Sapidis and Besl	Aproximando la superficie polinómica, Normales de la nube de puntos	Distancia de los puntos desde la superficie polinómica, Orientación de la normal de los puntos comparada con la dirección del eje Z
	Lavoue et. al.	Curvaturas principales	Un triángulo es añadido a la region basada en los “clusters” de curvaturas principales
	Zhang et. al.	Curvatura Gaussiana	Valor de la curvatura gaussiana según un umbral definido por el usuario
	Zuckerberger et.al.	Ángulo diedro de triángulos adyacentes.	Validación de la convexidad basándose en ángulos diedros
	Leonardis et. al.	Supercuádricas	Promedio de error de

			ajuste entre las superficies de puntos y supercuádricas	
Basados en "Watershed"	Mangan and Whitaker	Desviación de planaridad	Watershed Function f	Puntos que pertenecen a las cuencas de captación que la función f crea.
	Pulla et. al.	Curvatura gaussiana, Curvatura media, Raíz cuadrada de la media de la curvatura, Curvatura Absoluta		Puntos que pertenecen a las cuencas de captación que la función f crea.
	Page et. al.	Curvatura mínima, curvatura de las normales.		Puntos pertenecen a una Cuenca si s u curvatura mínima está por encima de un umbra, adicionalmente los puntos son añadidos a los segmentos en base a la curvatura de sus normales.
	Zuckerberger et.al.	Ángulos diedros		Bordes sobre la cuenca de captación que crea la función f
Reeb Graphs	Antini et. al.	Media de curvatura, Función del Saliente	Conectividad del "Reeb ", Curvatura Media.	
Basados en Modelos	Wu and Levine	Distribución de densidad de la carga eléctrica.	Los puntos frontera se eligen según la mínima distribución de densidad de la carga eléctrica.	
Basados en Esqueletos	Li et. al.	Funciones paramétricas y geométricas.	Los puntos críticos de las funciones paramétricas y geométricas definen los bordes de la segmentación.	
Clusters	Garland et. al.	Normales de la superficie y medidas de la irregularidad.	Planaridad, orientación de las formas.	

	Inoue et. al.	Área y perímetro de las regiones, normales de la superficie	Tamaño del área, suavidad de los bordes, planaridad de las regiones.
	Attene et. al.	Primitivas como planos, esferas y cilindros	Los puntos se agrupan según su mayor coincidencia en forma de plano, esfera o cilindro.
	Gelfand and Guibas	Movimientos de deslizamiento	El número y compatibilidad de los movimientos de deslizamiento define la segmentación de las áreas.
	Shlafman et. al.	Distancia geodésica y ángulos diedros.	Los triángulos se agrupan según una función de distancia definida por una suma ponderada de La distancia geodésica y los ángulos diedros.
	Katz and Tal	Distancia geodésica y ángulos diedros.	Los triángulos se agrupan según una función de distancia definida por una suma ponderada de La distancia geodésica y los ángulos diedros.
Análisis Espectral	Liu and Zhang	Valores propios de la matriz de afinidad.	El agrupamiento según los valores propios de la matriz de afinidad define el etiquetado de los triángulos en áreas de segmentación.
	Zhang and Liu	Valores propios de la matriz de afinidad.	Los dos vectores propios líderes de la matriz de afinidad definen una bipartición del conjunto de triángulos
Extracción Explicita de Contornos	Lee et. al.	Curvatura mínima	Centralidad del contorno, prueba de prominencia
Puntos Críticos	Katz et. al.	Puntos críticos y ángulos diedros	Centralidad de los puntos de la malla.
	Lin et. al.	Puntos críticos y ángulos diedros, función de salientes	La distancia geodésica y angular de los puntos críticos de la malla definen zonas donde pertenecen posibles límites de la segmentación.

Descriptores de Forma	Mortara et. al.	Curvatura Gaussiana Integral, contornos generados por la intersección de la esfera con la malla.	Numero de contornos generados por la intersección de la esfera y la superficie.
Modelos de Markov	Pichler et al.	Indice de forma, Curvatura	Convexidad.
Segmentación Directa	Benko and Varady	Vectores normales, varios tipos de filtros que distinguen entre los tipos de superficie	Los puntos de la malla se prueban en una secuencia de las posibles superficies solución con el fin de determinar a qué tipo de superficie pertenecen.

Tabla 1: Resumen características/criterio métodos segmentación mallas 3D [80]

2.3 Justificación del método elegido

Se ha implementado un método que primeramente, identifica las caras planas del objeto tridimensional. Para lograr este objetivo se ha optado por un algoritmo de crecimiento de regiones. Estos algoritmos se basan en la proximidad y similitud de los puntos. En el caso de una malla tridimensional los puntos son los triángulos que conforman el mallado de los objetos, los cuales cumplen la condición de proximidad entre sus vecinos al estar unidos directamente mediante aristas. Uno de los principales problemas de estos métodos es la elección de las "semillas" iniciales para expandir las regiones, pero este problema se simplifica en el método desarrollado, ya que cualquier triángulo puede actuar como semilla inicial. Además, debido a que solo se dispone de información geométrica de los triángulos, resulta sencillo establecer el criterio de expansión de las "semillas", el cual se explicará más detalladamente en siguientes secciones del documento. Por este motivo, los algoritmos de crecimiento de regiones proporcionan simplicidad de diseño para el problema de identificar las caras planas y rapidez a la hora de procesar todos los triángulos del modelo 3D.

Por otro lado se ha optado por un algoritmo basado en grafos para el reconocimiento de las características. Los grafos son una estructura ampliamente utilizada en todo tipo de ámbitos de ciencia y tecnología, y permiten modelizar y resolver una amplia variedad de problemas. En el caso que nos ocupa, modelaremos el problema de reconocimiento de características como un problema de comparación entre grafos. Asimismo, los grafos contienen una gran variedad de métodos disponibles y resulta sencilla su implementación. Por otra parte, permiten reducir drásticamente la información utilizada para el problema que se plantea, ya que se reduce gran parte de la información de la geometría (vértices, aristas y caras) en un solo nodo del grafo, por lo que resultan una opción apropiada para el tratamiento de mallas 3D .

3.- SOLUCIÓN PROPUESTA

El objetivo a alcanzar es realizar un procedimiento que permita clasificar las características de una malla tridimensional y más concretamente en modelos que representan edificios (o al menos parte de ellos, donde la fachada es el elemento principal). Para validar el método algorítmico propuesto, se ha elaborado una aplicación que sirve de banco de pruebas y permite visualizar los resultados obtenidos a lo largo de la elaboración de este trabajo final de master con sus diferentes etapas.

El método desarrollado utiliza como entrada de datos un modelo poligonal simplificado, que representa una fachada de un edificio como resultado del escaneado del mismo. Esta malla tridimensional no dispone de textura, por lo que de la única información de la que se dispone es información geométrica, esto es vértices, aristas, triángulos y las normales de cada triángulo. Como ya se ha comentado anteriormente en el contexto de la colaboración con el *Dipartamento di Architettura de la Università degli Studi di Firenze* y el Grupo de Robótica del Instituto Universitario de Automática e Informática Industrial (AI2) de la Universitat Politècnica de València, la institución italiana ha proporcionado los modelos tridimensionales de fachadas adquiridos con escáneres laser en la ciudad de Pietrabuona. Además se ha decidido utilizar el mismo tipo de formato de archivo, de cara a evitar posibles problemas futuros de compatibilidad o de conversión de ficheros. El formato en cuestión es el OBJ, utilizado para representar geometría tridimensional como vértices, normales, caras, etc. Este formato se estructura en texto plano a partir de una sucesión de líneas en las que primero aparece una clave y después una sucesión de valores. La clave indica el tipo de información que reflejan los valores sucesivos. Debido a ello este formato no requiere de una cabecera al principio del fichero [81]. A continuación se muestra una tabla con algunos de los tipos de claves y su significado:

Clave	Descripción
#	Comentario
v	Vértice
l	Línea
f	Cara
vt	Coordenada de textura
vn	Normal
g	Grupo

Tabla 2: Claves del formato OBJ.

En este trabajo se ha abordado en primera instancia, la detección de edificios con características ortogonales, es decir, que las caras que las componen forman ángulos rectos entre sí. Se trata, por otra parte, del caso más habitual. No obstante, el método empleado también es capaz de detectar características no ortogonales, como ventanas circulares por ejemplo.

Ilustración 15: Fachada con características ortogonales (izquierda), fachada con características no ortogonales (derecha).

El método propuesto se divide en 2 partes principales:

- Un algoritmo de crecimiento de regiones que distingue las caras planas.
- Un algoritmo basado en grafos que discierne las características de la malla.

Ambos algoritmos se aplican de manera secuencial. En primer lugar se distinguen las caras planas, con el fin de crear un grafo que será utilizado como entrada para el algoritmo que, en una segunda instancia, distingue las características.

También hay una etapa de pre-procesado al inicio de la carga de cada modelo tridimensional. La etapa consiste en almacenar los triángulos en una estructura de datos y para cada uno de ellos, referencias a todos sus vecinos. De esta manera, se aumentan ligeramente los tiempos de carga de cada modelo, pero se reducen los costes temporales de los algoritmos implementados.

La aplicación desarrollada para visualizar los resultados de ambos algoritmos, permite hacer pruebas con una serie de mallas de ejemplo clasificadas en “Formas regulares”, “Formas irregulares”, “Edificios Modelados” y “Edificios escaneados”.

3.1 Distinción de caras planas

Para distinguir las caras planas se utiliza un algoritmo de crecimiento de regiones [58]. Como se ha explicado ya, este tipo de algoritmos empiezan buscando un conjunto de elementos que se denominan “semilla” (*seed* en inglés), una vez encontrados los elementos semilla, se expanden hasta que se alcanza una condición de parada.

Dos caras vecinas formarán parte del mismo plano si sus normales son paralelas entre ellas y apuntan a la misma dirección, considerando una cierta tolerancia.

Esta condición es la que se utiliza para desarrollar el algoritmo de crecimiento de regiones. El algoritmo sigue los pasos explicados en los siguientes apartados.

3.1.1 Identificación de los triángulos/facetas semilla

En este caso, se considera que cualquier triángulo que todavía no tenga asignada una región es una "cara semilla". Se identifican las regiones por colores, de esta manera triángulos con el mismo color pertenecerán a la misma región. Al inicio del algoritmo, ninguna cara tiene asignada una región, por lo que todas ellas son "caras semilla". El procedimiento que se sigue, es recorrer la lista de triángulos de la malla en busca de elementos sin colores asignados, que serán los elementos semilla, en cuanto se encuentra uno, se empieza a expandir esa semilla.

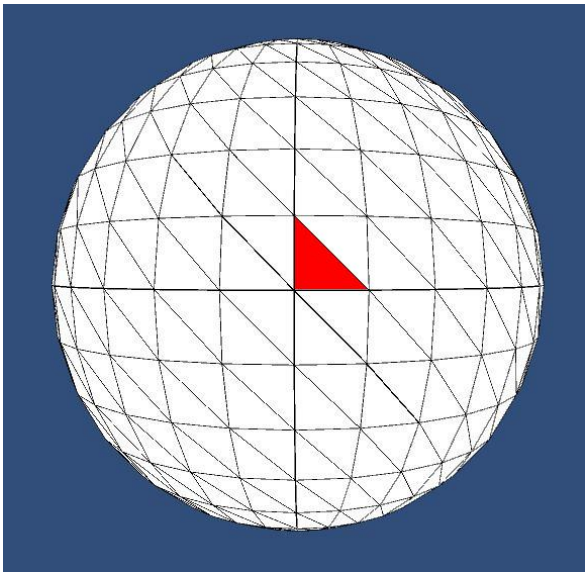


Ilustración 16: Detección de "cara semilla".

3.1.2 Expansión de la región.

Una semilla puede ser considerada como una región con un solo elemento. A partir de ella se comprueban sus triángulos vecinos. Un triángulo es vecino de otro, si comparten 1 o 2 vértices. En el caso de compartir 2 vértices significa por tanto que comparten una arista.

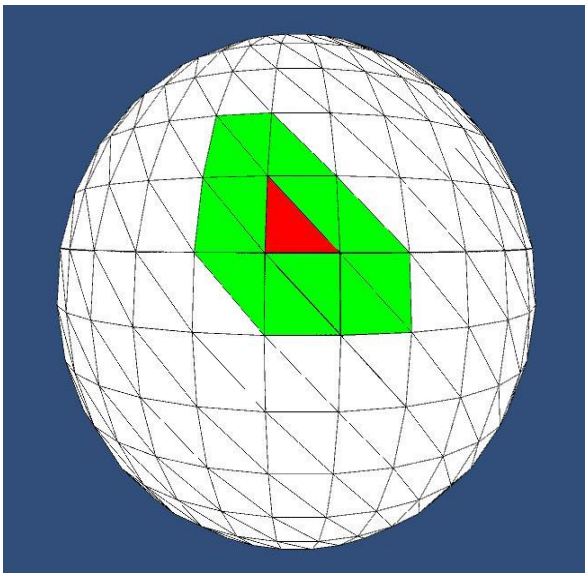


Ilustración 17: Caras vecinas (verde) a la cara semilla detectada (rojo).

Si un triángulo es vecino de otro, se calcula el producto escalar de sus normales. El resultado de este producto indica el ángulo entre ellas. Las 2 normales definen 2 planos en un espacio cuyo ángulo entre ellos se denomina ángulo diedro.

Un ángulo pequeño o nulo, indica que los triángulos forman parte de la misma región plana, mientras que un ángulo grande indica que forman parte de regiones diferentes. Se establece un umbral o factor de coplanaridad para este ángulo, de manera que, si el ángulo calculado para las normales está por debajo de este umbral, se asigna la misma región a ambos triángulos. En caso contrario, se asignan regiones diferentes. En las siguientes imágenes se pueden ver diferentes resultados para el mismo triángulo, según el valor umbral que se seleccione.

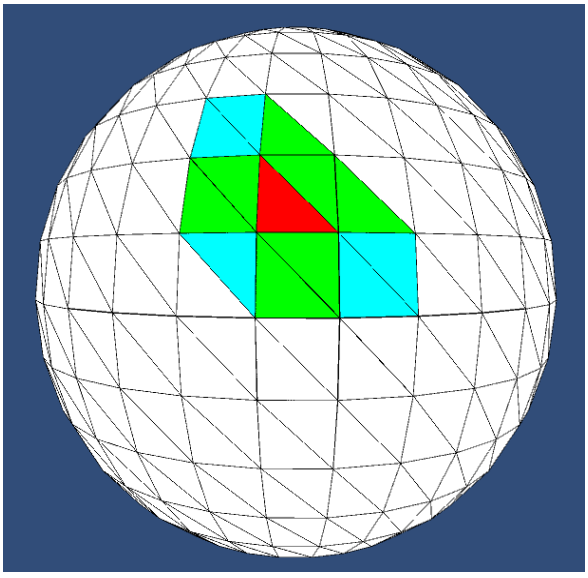


Ilustración 18: Cara semilla (rojo), Cara vecina para expandir la región (verde) y, caras vecinas descartadas para la expansión de la región (azul).

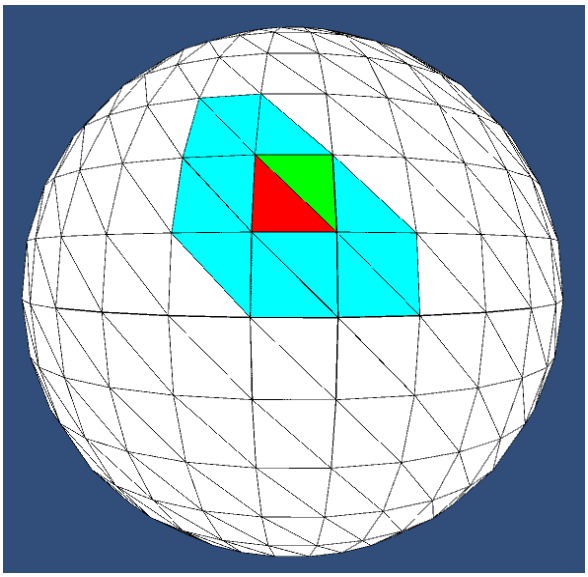


Ilustración 19: Mismo caso que el anterior pero con un valor umbral más pequeño.

En la siguiente ilustración se tiene un ejemplo de lo explicado anteriormente, en este caso, el ángulo formado por las normales representadas en negro entre la cara semilla y uno de sus vecinos es cercano a cero, ya que las normales son paralelas, por lo que se considera que estos dos triángulos forman parte de la misma cara plana.

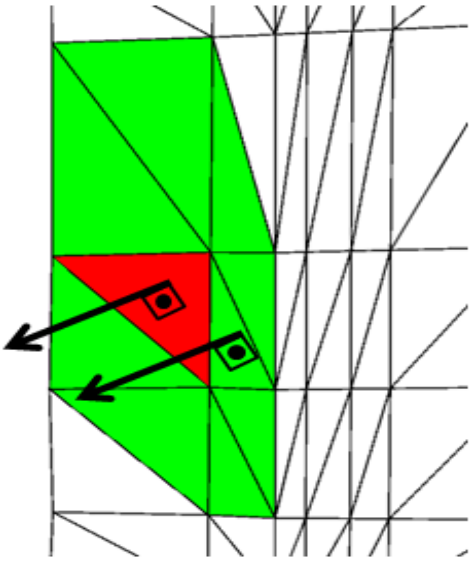


Ilustración 20: Triángulos de la misma cara plana.

En cambio la siguiente ilustración, el ángulo entre las normales es mayor, aproximadamente de noventa grados, por lo que los triángulos se corresponderían con caras planas diferentes.

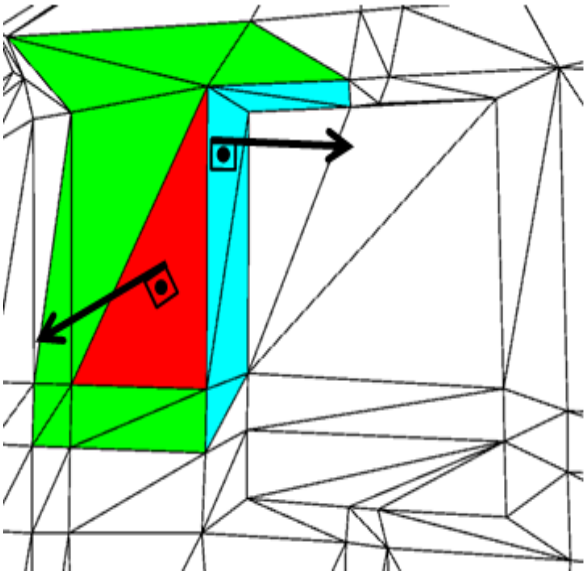


Ilustración 21: Triángulos de caras planas diferentes.

Una vez se han procesado todos los triángulos vecinos a la “cara semilla”, se detiene el proceso de expansión de la región y se pasa a buscar otra “cara semilla” para repetir el procedimiento. El algoritmo continua hasta que no quedan más “caras semilla” y se tiene como resultado un conjunto de pequeñas regiones formadas por triángulos y sus vecinos correspondientes a la misma cara plana. Para gestionar estas regiones se ha utilizado un “MFSet”.

Un MFSet (del inglés *Merge-Find Set*) es una estructura de datos donde se organizan una serie de conjuntos disjuntos con un número de elementos fijos, cada subconjunto se identifica por un miembro que es el “representante” del grupo. Esta estructura contiene dos operaciones que la caracterizan:

- Combinación (*Merge*): Se unen dos conjuntos.
- Encontrar (*Find*): Determina el conjunto al que pertenece el elemento pasado como entrada.

Se representa la estructura de datos como un vector “V”, donde “$V[i]$” indica el padre del elemento i-ésimo.

Cada subconjunto de la estructura es un árbol, un árbol es un tipo de estructura de grafos que se explicará más adelante. Los nodos del árbol los elementos del conjunto. El nodo raíz del subconjunto es el representante del mismo. En el método propuesto se considera un nodo de este árbol como un triángulo y cada árbol representa una cara plana. Al hacer una operación *“Find”* se encuentra el representante de ese elemento, que contendrá un identificador de color. De esta forma se asignan a todos los elemento del grupo, que son triángulos, el color indicado por su representante, con lo que se tendrán todos los triángulos de la misma cara plana con el mismo color.

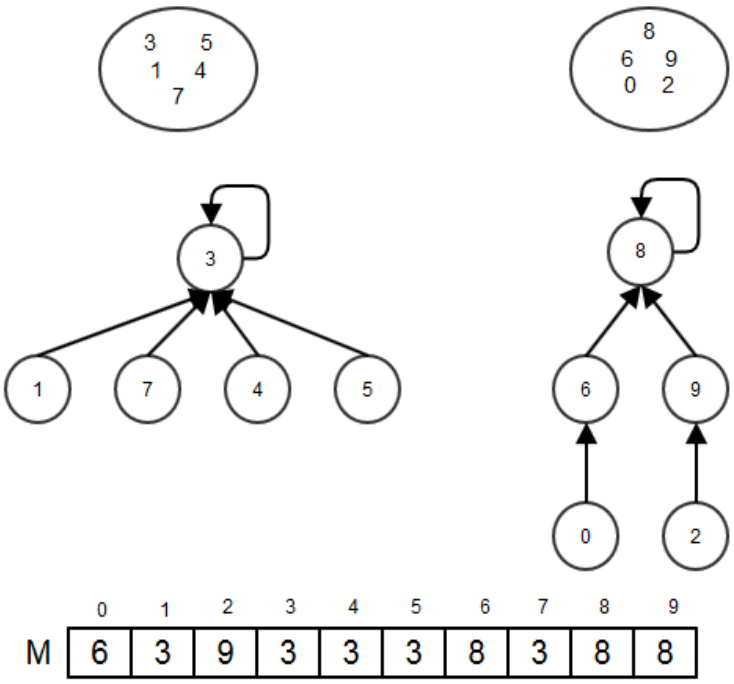


Ilustración 22: Ejemplo representación de la estructura de datos utilizada.

En el método desarrollado lo que ha permitido la estructura de datos es agrupar en un mismo conjunto estas pequeñas regiones aunque tengan color diferente. Si dos triángulos forman parte de la misma cara plana, se hace una operación *"merge"* con los índices de los triángulos, de esta manera pasarán a formar parte del mismo conjunto en la estructura. Por otro lado si se comprueba un triángulo vecino que ya tiene asignado un color, se hace la comprobación con sus normales y si es positiva se buscan sus representantes y se hace una operación *"merge"* de ambos con lo que se consigue que 2 conjuntos de triángulos que tenían asignados diferentes colores, pero pertenecían a la misma cara plana, ahora tengan un único representante. Una vez hecho esto se puede pasar al siguiente punto.

3.1.3 Coloreado de las regiones.

Gracias a la estructura de datos explicada anteriormente, se puede consultar para cada triángulo el representante del conjunto al que pertenece, para ello se crea un "diccionario". Esta estructura de datos contiene pares "clave-valor", cuya característica es que un valor es accesible a través de su clave. Así, se crea una estructura cuyas claves son números enteros que referencian colores, con lo que podemos asignar a un número de representante un color determinado. Para conseguir colorear todos los triángulos se itera por su lista y se comprueba su representante y con este número se puede comprobar su color. Si la estructura no contiene el número se crea un nuevo color y se añade a la misma. De esta forma se consigue colorear todos los triángulos que pertenecen a una misma cara plana.

El coloreado permite distinguir las regiones en la visualización del resultado del algoritmo.

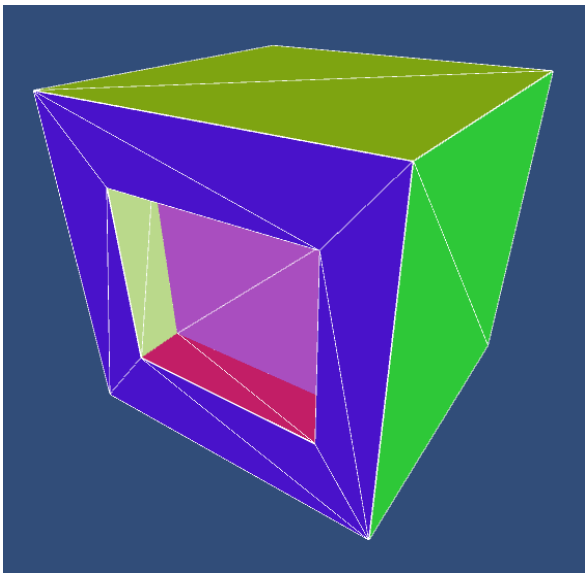


Ilustración 23: Resultado detección de caras planas.

A continuación se muestra el algoritmo que realiza el método explicado anteriormente.

***Algoritmo* 1.** *Detección de las caras planas.*

```
1:  for i ∈ [0 … nº triángulos] do
2:    for j ∈ [0 … número_vecinos_triángulo_i] do
3:      If Vecino no tiene color asignado and
            AnguloNormales(triángulo_i, vecino_j) < límite then
4:        Asignar a vecino_j el mismo color que triángulo_i;
5:        mfs.Merge(triángulo _i.Indice, vecinos_l.Indice);
6:      else
7:        if Vecino ya tiene color asignado and
            AnguloNormales(triángulo_i, vecinos_j) < límite then
8:        mfs.Merge(mfs.Find(triángulo_i.Indice),mfs.Find(vecinos_j.Indice));
9:      end if
10:   end if
11:  end for
12: end for
13: for i ∈ [0 … nº triángulos] do
14:  m ← MFSet.Find(i);
15:  if Diccionario contiene clave "m" then
16:    triángulo _i.Color ← Color devuelto por el diccionario con clave "m";
17:  else
18:    Agregar al diccionario clave "m" con un color aleatorio;
19:    triángulo _i.Color ← Color devuelto por el diccionario con clave "m";
20:  end if
21: end for
```

3.2 Distinción de características

Una vez se tienen las caras planas del objeto diferenciadas, el siguiente paso es construir un "grafo topológico", en este caso un grafo no dirigido, se definen estos conceptos a continuación:

- Un grafo se define como un par $G = (V, E)$, donde "V" es un conjunto de vértices o nodos, y "E" es un conjunto de aristas o arcos que relacionan estos nodos.
- En un grafo no dirigido las aristas "E" son un conjunto de pares no ordenados, es decir, $(v_i, v_j) \equiv (v_j, v_i), v_i \; y \; v_j \; \epsilon \; V, (v_i, v_j) \; \epsilon \; E \; y \; (v_j, v_i) \; \epsilon \; E.$

En este caso el grafo topológico describe las relaciones de vecindad entre las caras planas detectadas en el paso anterior. De esta manera en el conjunto de caras planas "P", cada cara "p_i" contiene su correspondiente nodo "v_i" del conjunto "V" de nodos del grafo. Dos vértices "v_i" y "v_j" están unidos por una arista $e = (v_i , v_j) \;\; si \; \exists r \epsilon P_i, s \epsilon P_j : P_i , P_j$ son caras planas unidas del objeto tridimensional.

Se diferencian dos tipos de características, por un lado las características cóncavas tales como huecos, hendiduras o ventanas y por otro lado las características convexas como salientes, canalones, escudos ornamentales en las fachadas, etc.

El grafo topológico presenta la siguiente propiedad: los vértices que forman las características del modelo se unen al resto del grafo por un vértice de corte, también llamado punto de articulación.

Un vértice de corte es definido como aquel vértice que al quitarlo, aumenta el número de componentes conexas del grafo.

Una componente conexa de un grafo es todo aquel subgrafo inducido por los vértices de una clase de equivalencia.

Las clases de equivalencia son subconjuntos en los que se divide el conjunto original, con las propiedades siguientes:

- Los subconjuntos al unirlos dan el conjunto total.
- Los subconjuntos tienen que ser disjuntos.
- La intersección entre ellos debe dar como resultado el conjunto vacío, es decir, los subconjuntos no comparten elementos.

Si solo existe una componente conexa, o lo que es lo mismo, si todos los vértices se alcanzan mutuamente, se dice que el grafo es conexo. En el caso de la malla 3D que representa una fachada, el "grafo topológico" siempre va a ser un grafo conexo, debido a que la malla de la fachada no va a presentar discontinuidades, es decir, no van a existir fachadas con dos partes no unidas, las fachadas de los edificios se establecen en una sola superficie que se identifica con un único edificio. En la siguiente imagen se muestra un ejemplo de grafo no dirigido con 3 componentes conexas y sus correspondientes clases de equivalencia.

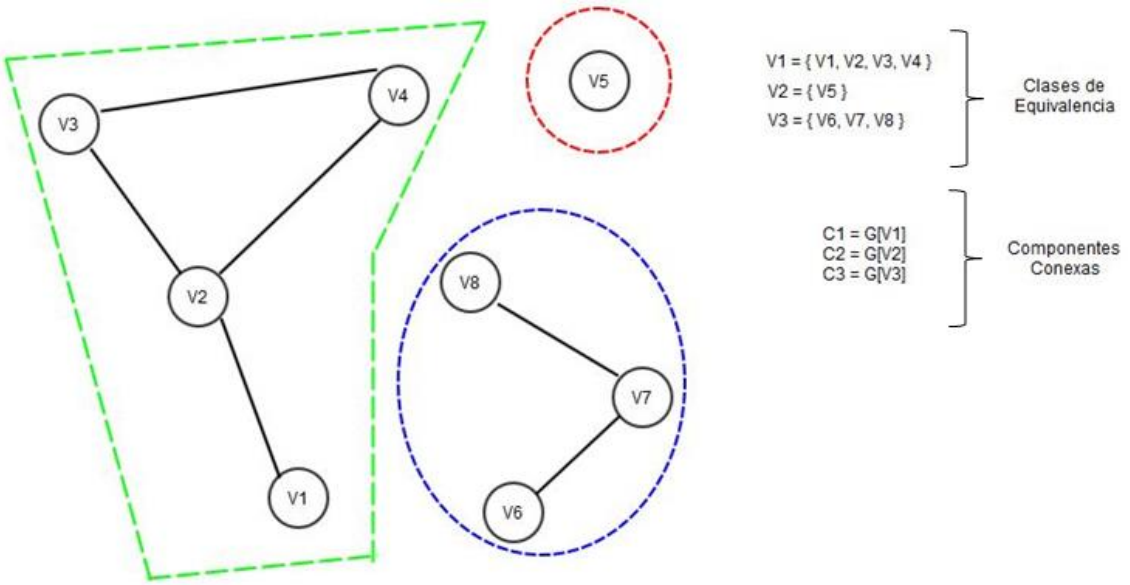


Ilustración 24: Ejemplo de grafo con 3 componentes conexas.

Los vértices de corte se corresponden con caras planas del modelo 3D que contienen las características que se buscan. Por ejemplo, en el caso de fachadas de un edificio, un vértice de corte seria la pared. Al eliminar este nodo del grafo, queda un nuevo grafo de varias componentes conexas, siendo cada una de ellas una característica en concreto. Por ello al detectar los vértices de corte de un grafo topológico, se pueden aislar las características en forma de grafos (componentes conexas).

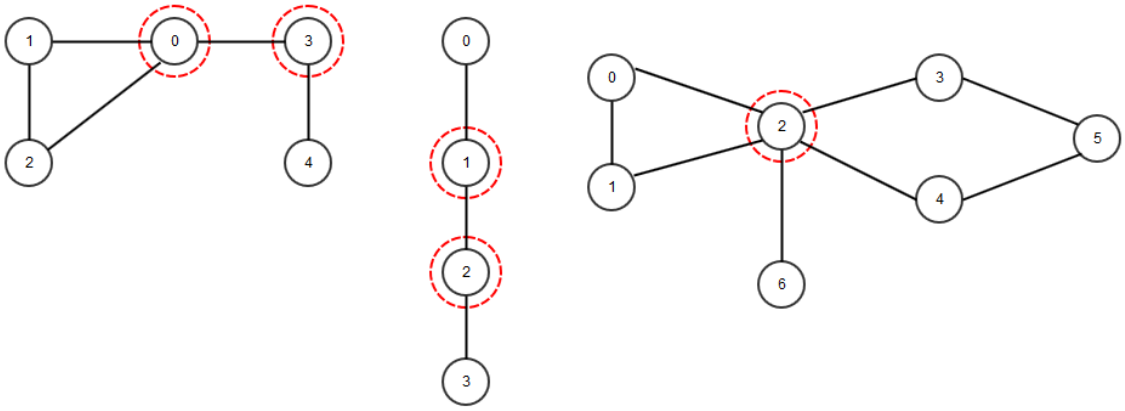


Ilustración 25: Ejemplo de vértices de corte.

Para detectar los vértices de corte de un grafo se ha utilizado el algoritmo de Tarjan [82], que se basa en el algoritmo DFS (*"Depth-First Search"*, búsqueda primero en profundidad) de recorrido de grafos. La estrategia explora sistemáticamente las aristas del grafo, de forma que se visitan primero los vértices vecinos a los visitados más recientemente. De esta forma se va profundizando en el grafo, o lo que es lo mismo, se aleja cada vez más del vértice inicial. Se continúa con el proceso, hasta que todos los vértices alcanzables desde el vértice de la llamada original han sido descubiertos, es decir, visitados por primera vez.

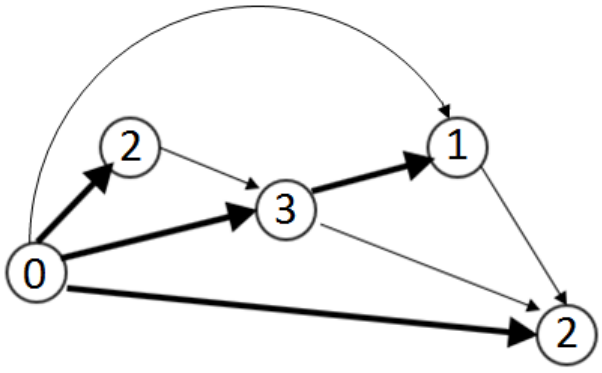


Ilustración 26: Ejemplo recorrido grafo utilizando DFS, Orden: 0, 4, 3, 1, 2.

El procedimiento a seguir para lograr este propósito es la recursividad. Los algoritmos recursivos se llaman a sí mismos para encontrar la solución al problema, hasta que llegan a un caso base, momento en el cual se retornan las llamadas y se calcula el resultado final.

El recorrido de un árbol topológico se realiza en forma de árbol, denominado “árbol DFS”. A continuación se incluyen una serie de definiciones relacionadas con el concepto de estructura arbórea en grafos:

- Un grafo no dirigido, es un “árbol” si es conexo y acíclico.
- Un grafo G se dice conexo si, para cualquier par de vértices “u” y “v” en G, existe al menos una sucesión de vértices adyacentes que, sin repetir vértices, vaya de “u” a “v”.
- Un grafo G se dice que es acíclico si no contiene ciclos, es decir, si no existen caminos que empiecen y acaben en el mismo vértice.
- En el árbol DFS un vértice “u” es padre de otro vértice “v”, si “u” descubre a “v” (y por tanto son nodos adyacentes en el grafo).

En un grafo cualquiera un vértice “u” es vértice de corte si se cumplen cualquiera de las siguientes condiciones:

- “u” es raíz del árbol DFS y tiene al menos dos hijos.
- “u” no es raíz del árbol DFS y tiene un hijo “v” tal que ninguno de sus descendientes está conectado con alguno de los ancestros en el árbol DFS de “u”.

Así, la búsqueda de los vértices de corte, se realiza recorriendo el grafo con el algoritmo DFS comprobando estas restricciones y almacenando los vértices solución.

Una vez se detectan los vértices de corte, el siguiente paso consiste en borrarlos del grafo actual, en consecuencia se consigue un grafo en el que cada una de sus componentes conexas representa una característica de la fachada. Para extraer estas componentes conexas, se usa de nuevo el algoritmo DFS, devolviendo de forma recursiva los nodos que pertenecen a la misma componente conexa.

Para continuar se muestra el algoritmo utilizado que detecta los vértices de corte.

Algoritmo 2. *Detección de vértices de corte.*

```
1:  Vertices_Articulacion_Aux(){
2:    Inicializar cuenta de los hijos del árbol DFS
3:    Se marca el nodo actual como visitado
4:    Inicializar tiempo de descubrimiento y valor 'low'
5:    for cada nodo vecino del actual do
6:      if( v no se ha visitado todavía) then
7:        Se hace hijo de "u" en el árbol DFS y lo llamamos recursivamente.
8:        Vertices_Articulacion_Aux(v);
9:        Se comprueba si el subárbol con raíz v tiene conexión con algún ancestro de u
10:       if( u es raíz del árbol DFS y tiene 2 o más hijos) then
11:         Se añade "u" a la lista solución.
12:       end if
13:       if( Si u no es raíz y el valor low de uno de sus hijos es mayor que el valor de
      descubrimiento de u) then
14:         Se añade "u" a la lista solución.
15:     else
16:         Actualizar el valor de u para las llamadas de los padres
17:     end if
18:   end for
19: }
```

A continuación se definen "grafos de consulta". Cada uno de ellos recoge la configuración de la forma de un objeto. Básicamente se trata de "grafos topológicos" de una característica en concreto, por lo que una característica queda tipificada como un grafo de consulta. Estos grafos se introducen manualmente por el usuario en la aplicación. Lo que se pretende es aprovechar el conocimiento que posee el usuario sobre la topología de las características, representando ésta como un grafo donde cada nodo es una de las caras que representan las características a encontrar y las relaciones entre los nodos definen la forma de éstas. En este punto, el problema que queda es el de comparar los grafos de las componentes conexas con los grafos de consulta, de esta forma se encuentran características con la misma topología que los grafos de consulta que se han definido. No obstante esto no suele ser suficiente para distinguir entre todas las características posibles, y será necesario por tanto añadir posteriormente más restricciones.

El problema de comparar dos grafos se conoce formalmente como "isomorfismo de subgrafos", en el que se dan como entrada dos grafos "A" y "B" y se debe determinar si "G" contiene un subgrafo que sea isomorfo a "B". Se demuestra que el problema de isomorfismo de subgrafos es NP-completo en [83][84].

Un "isomorfismo de grafos" es una biyección entre los vértices de los grafos que mantiene sus relaciones de adyacencia. A continuación se define el concepto de función biyectiva:

- Una función es biyectiva, si al mismo tiempo es inyectiva y sobreyectiva, lo cual significa que todos los elementos del conjunto de salida tienen una imagen distinta en el conjunto de llegada, y a cada elemento del conjunto de llegada le corresponde un elemento del conjunto de salida. Explicado de otra manera, una biyección "f" de un conjunto "X" en un conjunto "Y", determina una correspondencia 1-a-1 entre los dos conjuntos cuando para cada $x\epsilon\mathbb{R}$, hay un único $y\epsilon\mathbb{R}$ con $f(x) = y$, y de la misma manera para cada $y\epsilon\mathbb{R}$ hay un único $x\epsilon\mathbb{R}$ con $y = f(x)$.

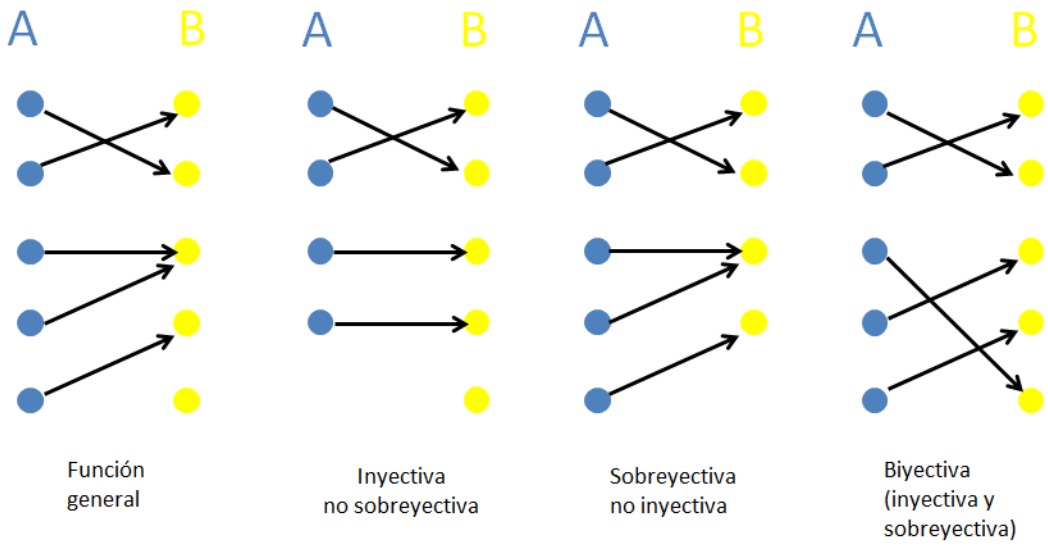


Ilustración 27: Tipos de funciones.

Dado que en el método desarrollado, los grafos (tanto topológicos como de consulta) van a tener un número muy reducido de nodos, se ha optado por utilizar el algoritmo recursivo propuesto por Ullman [85], que a pesar de no tener un coste temporal óptimo, ofrece buenos resultados y es el más citado en la bibliografía, ya que fue el primer enfoque y es en el que se basan muchos de los algoritmos de isomorfismo de grafos actuales [86]. El algoritmo de Ullman emplea la técnica de "backtracking" (o vuelta atrás) sobre estructuras arbóreas.

El objetivo de la técnica de "backtracking" es hallar una o todas las soluciones que pueda tener un problema, utilizando para ello un recorrido en profundidad del árbol (método DFS explicado anteriormente). Soluciones parciales son creadas a medida que se profundiza en el árbol. Se tiene éxito si se alcanza una solución completa. En este momento se devuelve la solución encontrada y se detiene o se sigue buscando más soluciones. Si no se encuentra una solución al problema, el algoritmo vuelve atrás de la misma manera que el algoritmo DFS, eliminando los nodos utilizados de la solución errónea o imposible. Cuando se vuelve a un nodo que tiene uno o más vecinos sin explorar, se prosigue el recorrido de una solución y así sucesivamente hasta devolver la primera solución encontrada o todas.

Con el mencionado algoritmo se encuentran todos los isomorfismos entre sub-grafos de $G_a = (V_a, E_a)$ y un grafo $G_b = (V_b, E_b)$. Las matrices de adyacencia de G_a y G_b son A y B respectivamente. Las matrices de adyacencia son un tipo de representación matricial de grafos.

Se llama matriz de adyacencia del grafo $G = (V, E)$ a la matriz nxn $D = (d_{ij})$ donde:

$$d_{ij} = \begin{cases} 1 \; si \; (v_i, v_j) \in E \\ 0 \; si \; (v_i, v_j) \notin E \end{cases}$$

En el algoritmo de Ullman se define una matriz M' de A_{FILAS} X $B_{columnas}$, cuyos elementos son ceros o unos, de forma que cada fila debe contener solo un "1" y no hay ninguna columna que tenga más que un "1". Esta matriz M' , se puede usar para permutar las filas y las columnas de la matriz B para producir una matriz "C" tal que $C = M' * (M' * B)^T$, indicando la "T" transposición matricial.

Si se cumple que para la todo elemento cuyo valor sea 1 en M' hay un 1 en la misma posición de "C" , entonces M' denota un isomorfismo entre los grafos G_a y G_b, se llamará a esta condición la "condición de isomorfismo".

M'_{ij} indica que el elemento i-ésimo de la matriz "A" corresponde con el elemento j-ésimo de la matriz "B" en el isomorfismo. Al inicio del algoritmo, se construye una matriz M^0 de A_{FILAS} X Bcolumnas donde:

- $M_{ij}^0 = 1$ si el grado del elemento j-ésimo de "B" es mayor o igual al grado del elemento i-ésimo de "A".
- $M_{ij}^0 = 0$ en cualquier otro caso.

El algoritmo procede generando todas las posibles matrices M' , de forma que se cumpla la "condición de isomorfismo". Las matrices M' se construyen cambiando a 0 todos menos uno de los 1's en cada una de las filas de M^0, y se tiene que seguir cumpliendo que ninguna de las columnas de M' contenga más de un 1.

Un vector binario es utilizado con la función para saber que columnas han sido utilizadas en los estados intermedios de la computación. El algoritmo emplea un segundo vector binario para saber que columna ha sido seleccionada y a que profundidad del árbol de recorrido.

Ullman también propone un refinamiento para convertir 1's de las matrices M' a 0's y de así reducir el coste temporal, ya que de esta manera se "poda" el árbol de búsqueda, eliminando soluciones incorrectas. Esta poda se basa en la observación siguiente: si se hace corresponder un nodo N_a en G_a con un nodo N_b de G_b, entonces también se deberían de corresponder los vecinos de N_a con otros nodos vecinos del de N_b. Por tanto, si se encuentra algún vecino de N_a que no corresponde con ningún vecino de N_b se puede poner a 0 tanto su valor como el de N_a en la matriz M'.

En la implementación realizada en este trabajo final de master, la función de Ullman devuelve "verdadero" si se ha encontrado un isomorfismo, lo cual indica que el patrón que se ha

definido (grafo de consulta) existe en esa componente conexa. La función devuelve falso en caso contrario.

Volviendo al algoritmo principal, lo que resta es comparar cada componente conexa con los grafos de consulta que se tienen y aplicar las restricciones adicionales para cada patrón.

Se ofrece a continuación un ejemplo para el caso detección de características ortogonales como es el caso de las ventanas que se muestran en el siguiente modelo:

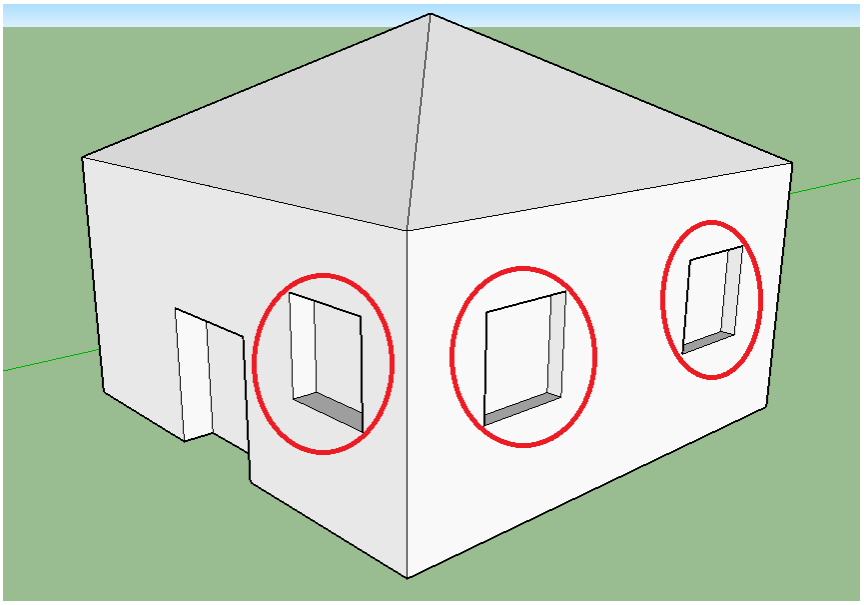


Ilustración 28: Ejemplo ventanas como características ortogonales.

Si se despliegan las caras planas de cada una de estas ventanas se conseguiría una forma como la mostrada en la ilustración 29, que estaría representada por el grafo de la misma imagen.

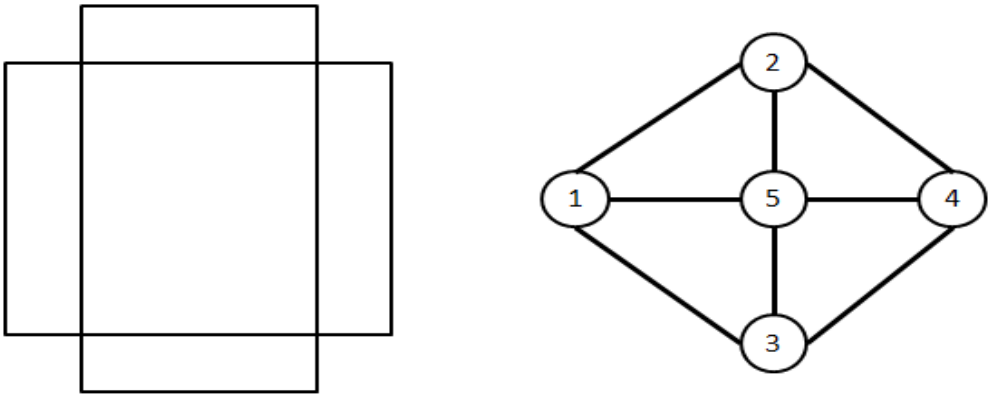


Ilustración 29: Desplegado ventana cuadrada y grafo resultante.

La comparación de este grafo con las componentes conexas del grafo topológico, nos indica donde están este tipo de características. No obstante existe el problema de que este grafo represente tanto características cóncavas como convexas (ver ilustración 30).

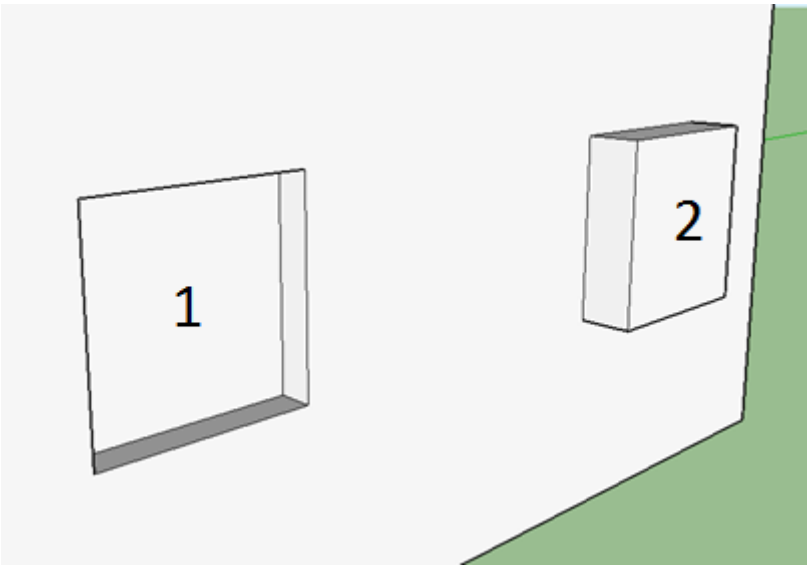


Ilustración 30: Característica cóncava y convexa.

Se requiere la capacidad de distinguir entre estos dos casos. Para ello, primero se busca el nodo del grafo con mayor grado, entendiendo como *grado de un nodo* el número de aristas que inciden en él.

En el caso presentado, estos nodos corresponderán a la cara más interna si es una característica cóncava, y a la cara más externa si es una característica convexa(caras 1 y 2 en la ilustración anterior). A continuación se utiliza la normal de esta cara y por otro lado el punto central de un triángulo cualquiera de las otras caras. Con estos dos elementos (vector normal y un punto), se puede crear un plano paralelo a las caras 1 o 2.

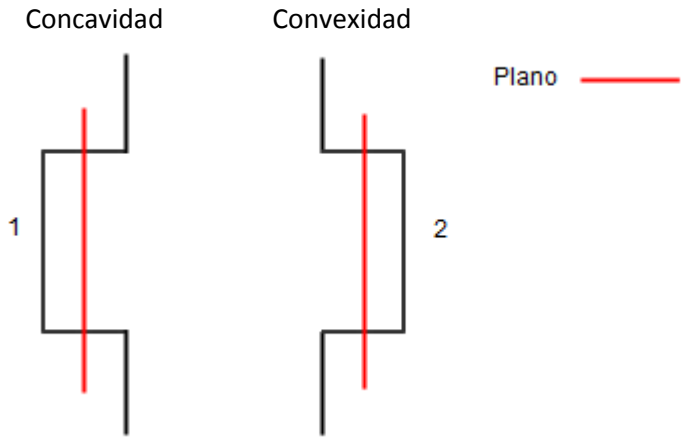


Ilustración 31: Creación de plano.

En el siguiente paso, se obtiene un punto de la cara más externa (cara 1 o 2 en la imagen anterior), y se comprueba en qué parte del plano se sitúa. Como se puede ver en la ilustración 31 si el punto se sitúa a la izquierda del plano entonces esta componente conexa representa una característica cóncava (en este caso una ventana), si en cambio el punto se encuentra a la derecha del plano se tendrá una característica convexa.

Para finalizar el método, solo faltaría colorear del mismo color todos los triángulos de la componente conexa en la que se ha detectado una característica. Esto se consigue gracias a la relación existente siguiente "Componente conexa → Nodos → Caras → Triángulos". Se muestra seguidamente el algoritmo utilizado en esta metodología.

Algoritmo 3. *Extracción de características.*

```
1:  for i ∈ [0 … nº triangulos] do
2:    for l ∈ [0 … n_vecinos_triangulo_i] do
3:      if((Triangulos[i].Color != vecinos[l].Color)) then
4:        if(!grafo_caras.Contains(Triangulos[i].Color)) then
5:          grafo_caras.AddNode(Triangulos[i].Color);
6:        end if
7:        if(!grafo_caras.Contains(vecinos[l].Color)) then
8:          grafo_caras.AddNode(vecinos[l].Color);
9:        end if
10:       if(!NoExisteArco) then
11:         Añadir arco.
12:       end if
13:     end if
14:   end for
15: end for
16: vertices_corte ← grafo_caras.Vertices_Articulación();
17: for nodo ∈ vertices_corte do
18:   grafo_caras.Remove (n.Value);
19: end for
20: componentes_conexas ← grafo_caras.ComponentesConexas;
21: for grafo_a ∈ componentes_conexas do
22:   for grafo_b ∈ grafos consulta do
23:     Comparar(grafo_a, grafo_b);
24:     if grafo_a y grafo_b iguales then
25:       Colorear sus triángulos
26:     end if
27:   end for
28: end for
```

4.- EJEMPLO DE APLICACIÓN

El método desarrollado en este trabajo final de master utiliza modelos 3D sobre los que aplica el algoritmo descrito en la sección anterior para detectar características. Para visualizar los resultados obtenidos se ha desarrollado una aplicación de la cual se muestra un ejemplo de utilización en esta sección con una serie de imágenes que ilustran la explicación de los diferentes estados.

En primer lugar, se carga una malla triangular con el modelo del edificio sobre el que se va a aplicar el algoritmo. Este primer paso conlleva un pre-procesado de la malla que consiste en referenciar para cada triángulo sus vecinos, de cara a poder aumentar la eficiencia en etapas posteriores. La siguiente ilustración muestra un ejemplo de malla sobre la que se aplicará la técnica descrita.

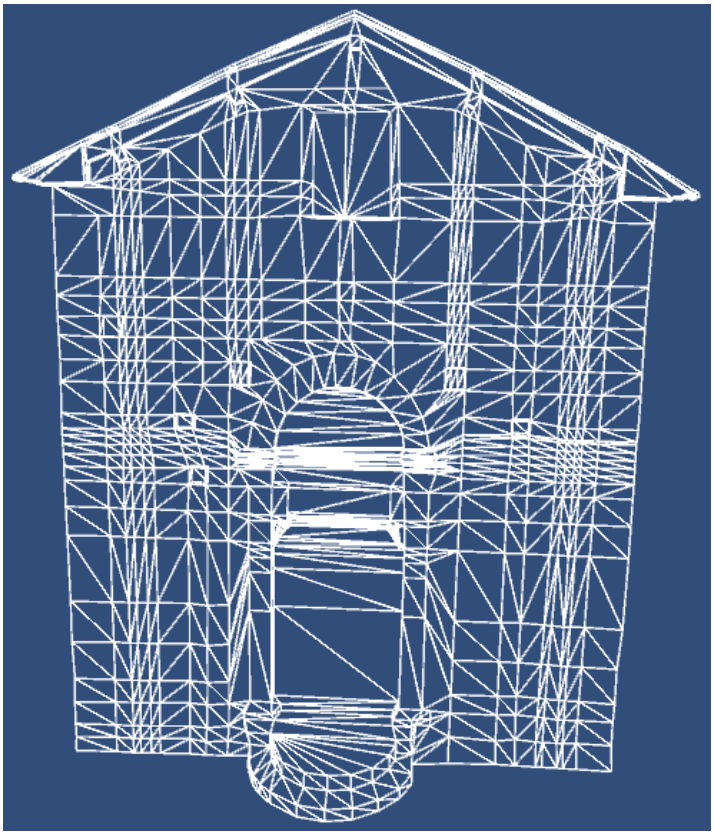


Ilustración 32: Estado inicial del visualizador.

A continuación se muestra el resultado obtenido para el caso base en el que a cada triángulo se le asigna un color aleatorio:

Ilustración 33: Reconocimiento de triángulos/facetas.

Seguidamente, en la ilustración 34, se muestra el resultado de la detección de las caras planas. Se puede apreciar que la pared de la fachada forma una misma componente, mientras que el resto de características forman parte de componentes diferentes.

Existe la posibilidad de cambiar el umbral que se establece para saber si 2 triángulos pertenecen a la misma cara plana. El cambio determinado por este límite se puede valorar en la ilustración 35, donde se aprecia que una característica con curvatura se detecta de forma correcta o incorrecta en función de este valor umbral. Se considera que es correcta cuando todas las caras que conforman la curvatura del objeto forman parte de la misma cara, es decir, tienen asignado el mismo color. En cambio, se considera incorrecta en caso contrario, tal y como se aprecia en la ilustración.

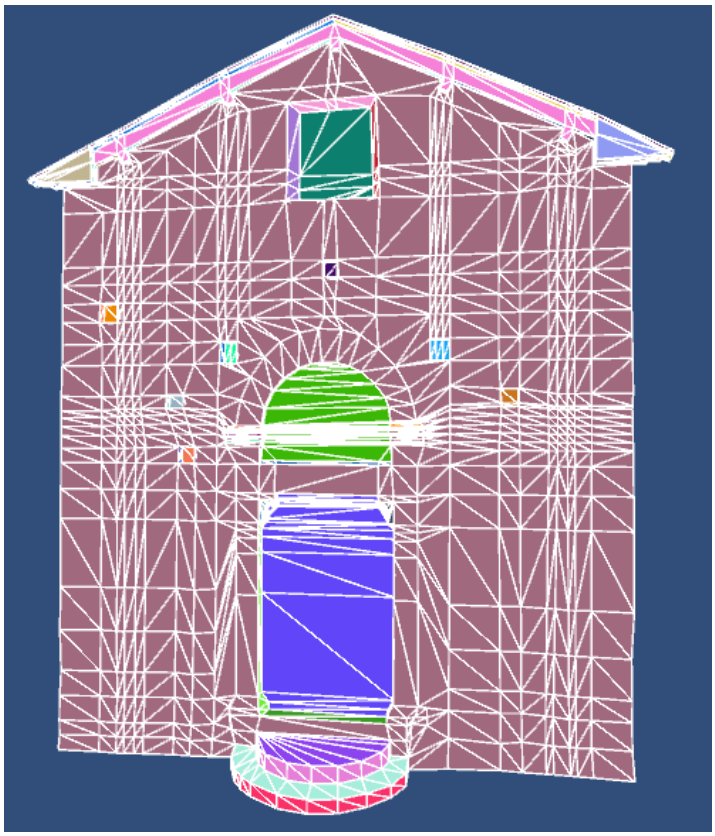


Ilustración 34: Reconocimiento de caras planas.

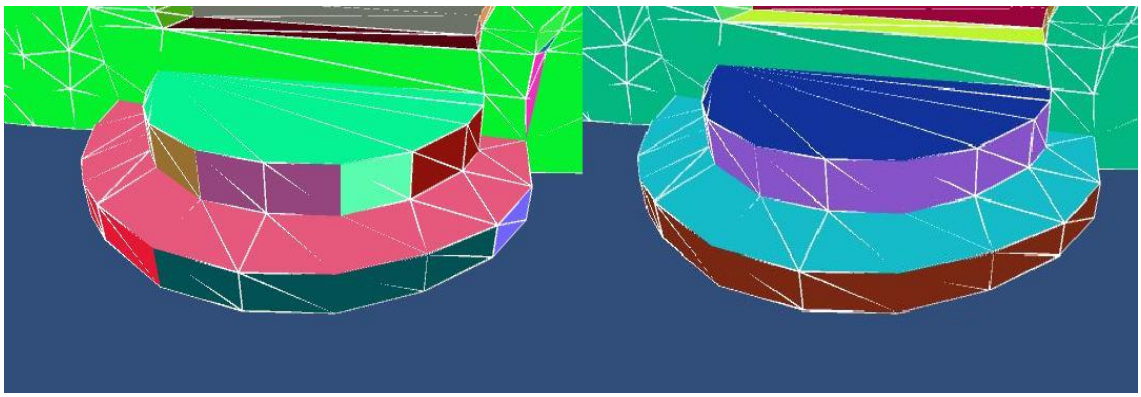


Ilustración 35: Detalle de caras planas en característica curva.

Por último se muestra el reconocimiento de características donde se puede ver que todas las características ortogonales se visualizan del mismo color, identificando las posiciones de las ventanas y características cóncavas en la fachada.

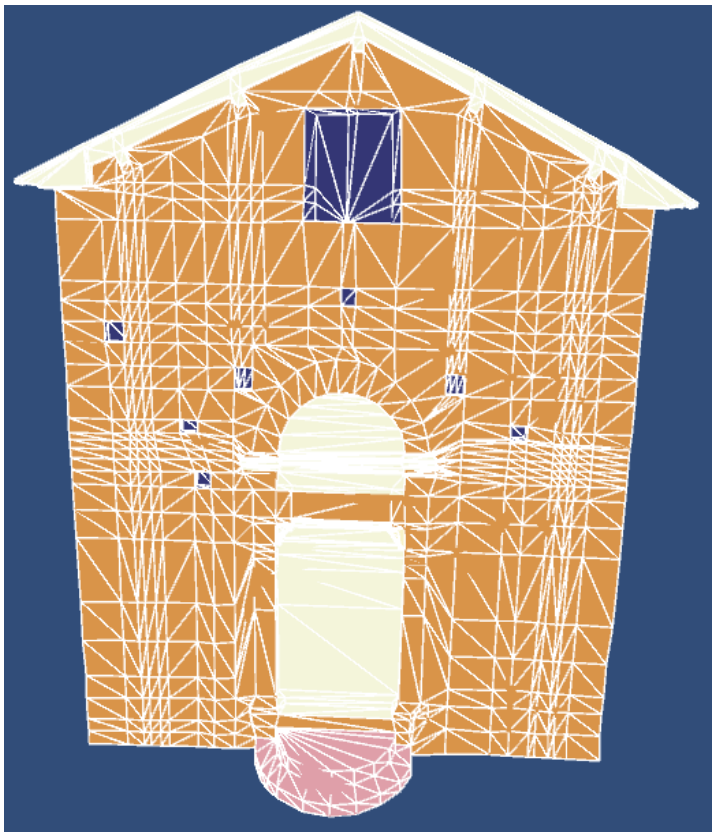


Ilustración 36: Reconocimiento de características.

En este proceso se han utilizado los grafos de consulta que definen los patrones topológicos a buscar dentro de los grafos que especifican cada característica. Así, la ilustración siguiente muestra el ejemplo de detectar los escalones en el modelo anterior a partir del grafo patrón que los define. El resultado se muestra identificando la característica del mismo color.

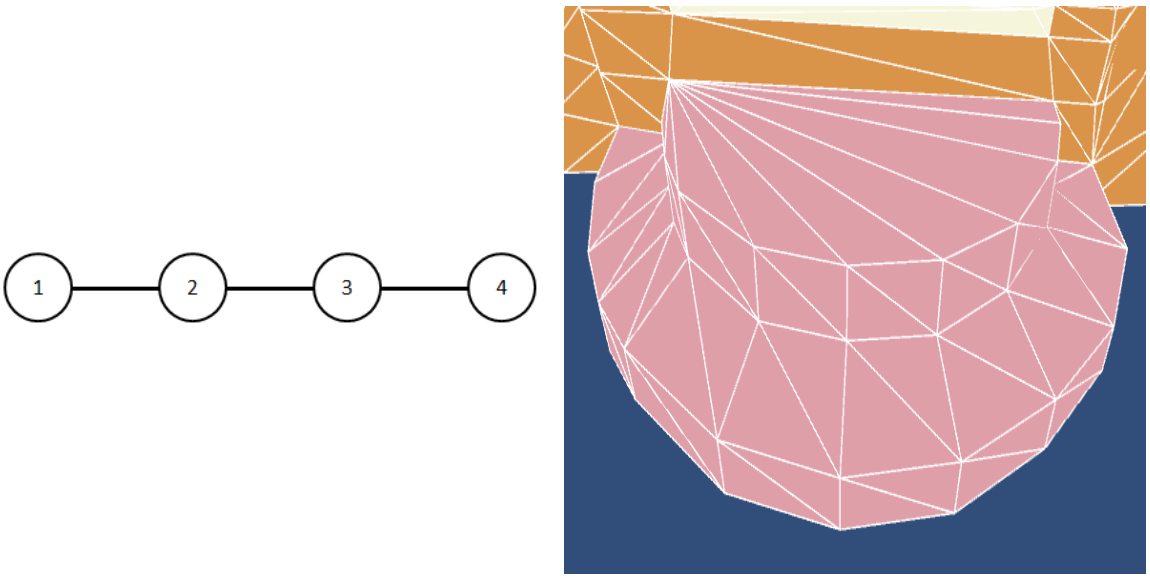


Ilustración 37: Patrón y característica encontrada en el modelo.

Si comparamos este grafo con la imagen derecha de la ilustración 35, se puede apreciar como las caras planas de los escalones formarían un grafo como el mostrado en la figura anterior. Al comparar estos grafos y encontrarlos iguales se colorea toda la característica del mismo color.

5.-RESULTADOS

5.1 Planteamiento

En esta sección se realiza el análisis de los datos obtenidos al ejecutar pruebas sobre diferentes tipos de modelos 3D. Se estudian principalmente los costes temporales de los algoritmos y el número de caras planas o características detectadas.

Todos los algoritmos han sido programados con el lenguaje "C#" y testeados sobre un procesador Intel Core i7 a 2.30GHz y 4Gbytes de memoria RAM, ordenador con velocidad de procesamiento común hoy en día. A la vista de los resultados, no son necesarios equipos más potentes para obtener resultados aceptables en cuanto a coste temporal se refiere.

Las pruebas realizadas se han aplicado sobre 4 tipos de modelos diferentes. En primer lugar se tiene un grupo de "Formas regulares" que contienen figuras geométricas usuales, como una pirámide, un cilindro o un dodecaedro. El siguiente grupo es el de "Formas Irregulares" que como su nombre indica contiene modelos irregulares y que en la mayoría de casos presentan características curvas. Algunos ejemplos utilizados como formas irregulares son el modelo de una piedra y el modelo de una mesa. A continuación, el grupo de "Edificios modelados", que representan mallas 3D de edificios o fachadas realizadas con programas de modelado. Por último se tiene el grupo de "Edificios escaneados", que son las mallas 3D que se corresponden con los edificios escaneados de la ciudad de "Pietrabuona", proporcionadas por el *Dipartamento di Architettura de la Università degli Studi di Firenze.*

Para cada uno de los modelos a analizar se han establecido diferentes rangos de valores angulares para el factor de coplanaridad. Estos rangos van de 0 a 30, de 30 a 60, de 60 a 90 y de 90 a 120 grados. Para cada uno de estos rangos se han estudiado los siguientes factores: en primer lugar el coste temporal en segundos del algoritmo que reconoce los triángulos, después el coste de detección de las caras planas y a continuación el coste de detección de las características. Por último, se incluye el número de caras planas y el número de características detectadas.

Seguidamente se muestran las tablas utilizadas para la toma de datos. Se ha realizado una batería de pruebas más amplia con una cantidad mayor de modelos, pero para este capítulo solo se muestra un ejemplo de modelo de cada uno de los grupos descritos anteriormente.

Forma Regular:	Dodecahedron				
Triángulos	36				
Rango Ángulo coplanaridad	Tiempo de reconocimiento de triángulos (seg.)	Tiempo de reconocimiento de caras planas (seg.)	Tiempo de reconocimiento de características (seg.)	Caras Planas reconocidas	Características reconocidas
0-30	0,000175476	0,00050354	0,003036499	12	1
30-60	0,000137329	0,000640869	0,002349854	12	1
60-90	0,000091552	0,000549316	0,000976563	1	1
90-120	0,000135899	0,000459671	0,001365662	1	1

Tabla 3: Datos de prueba para una malla 3D que representa un dodecaedro (forma regular).

Formas Irregular:	Rock				
Triángulos	500				
Rango Ángulo coplanaridad	Tiempo de reconocimiento de triángulos (seg.)	Tiempo de reconocimiento de caras planas (seg.)	Tiempo de reconocimiento de características (seg.)	Caras Planas reconocidas	Características reconocidas
0-30	0,000457764	0,008422852	0,05093384	46	1
30-60	0,000549316	0,0100708	0,04699707	2	1
60-90	0,000358582	0,01010132	0,01071167	1	1
90-120	0,000396729	0,008483887	0,01119995	1	1

Tabla 4: Datos de prueba para una malla 3D que representa una roca (forma irregular).

Edificio Modelado:	Facade 1				
Triángulos	1331				
Rango Ángulo coplanaridad	Tiempo de reconocimiento de triángulos (seg.)	Tiempo de reconocimiento de caras planas (seg.)	Tiempo de reconocimiento de características (seg.)	Caras Planas reconocidas	Características reconocidas
0-30	0,001739502	0,01715088	1,511597	338	28
30-60	0,000671387	0,01660156	1,415344	278	29
60-90	0,000549316	0,01721191	1,12854	227	29
90-120	0,000488281	0,02248001	0,4725342	5	1

Tabla 5: Datos de prueba para una malla 3D que representa la fachada modelada de un edificio (edificio modelado).

Edificio Escaneado:	Pietrabuona_2				
Triángulos	1487				
Rango Ángulo coplanaridad	Tiempo de reconocimiento de triángulos (seg.)	Tiempo de reconocimiento de caras planas (seg.)	Tiempo de reconocimiento de características (seg.)	Caras Planas reconocidas	Características reconocidas
0-30	0,001953125	0,02050781	0,3417969	102	5
30-60	0,001464844	0,02099609	0,1982422	41	7
60-90	0,001953125	0,02246094	0,1572266	11	3
90-120	0,000976563	0,0234375	0,1494141	2	1

Tabla 6: Datos de prueba para una malla 3D que representa la fachada escaneada de un edificio (edificio escaneado).

5.2 Análisis y consideraciones

Teniendo en cuenta el conjunto de datos total y las pruebas realizadas, a continuación se exponen las conclusiones observadas.

Para analizar la bondad del algoritmo de detección de caras planas, se requiere contrastar los resultados obtenidos contra modelos conocidos, de los que se sabe previamente el resultado esperado, es decir, el número de caras planas que contiene el modelo. Para ello se han utilizado para el análisis poliedros regulares, los cuales se conoce de antemano el número de caras planas que contienen. Por ejemplo, se sabe que un hexaedro tiene 6 caras planas o que un octaedro está compuesto por 8 caras planas.

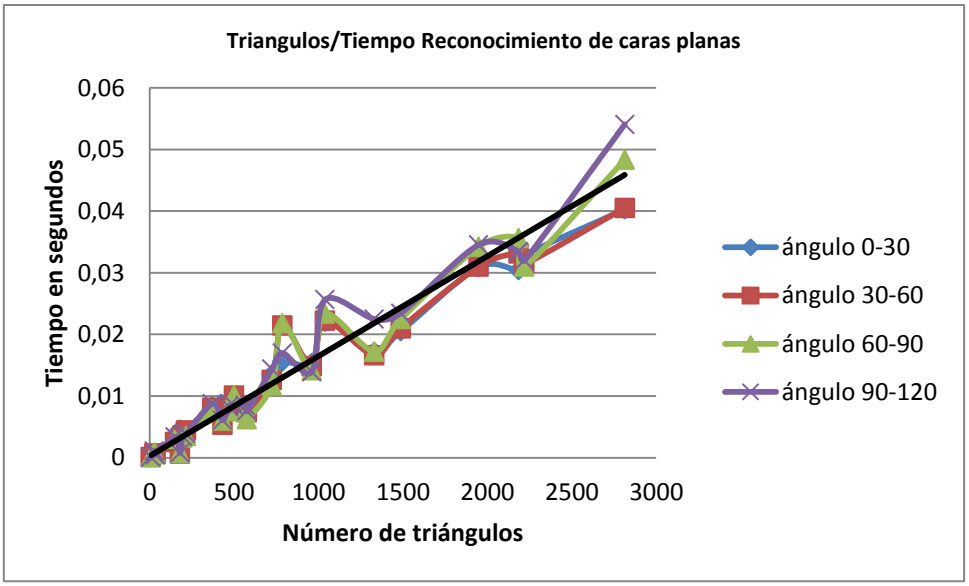
Seguidamente, se muestra la tabla con los resultados obtenidos. Para cada poliedro se muestra: el número de triángulos de su malla 3D, el número de caras planas conocido a priori del poliedro, el número de caras planas detectadas por el algoritmo y el valor mínimo del ángulo de coplanaridad que consigue que el algoritmo identifique correctamente las caras planas.

Poliedro	NºTriángulos	Caras Planas del poliedro	Caras Planas Detectadas	Mínimo Ángulo de Coplanaridad para detección correcta
Pirámide	8	5	5	1
Prisma Triangular	8	5	5	1
Hexaedro	12	6	6	1
Octaedro	8	8	8	1
Dodecaedro	36	12	12	1
Icosaedro	20	20	20	1

Tabla 7: Datos de prueba para la detección de caras planas sobre poliedros regulares.

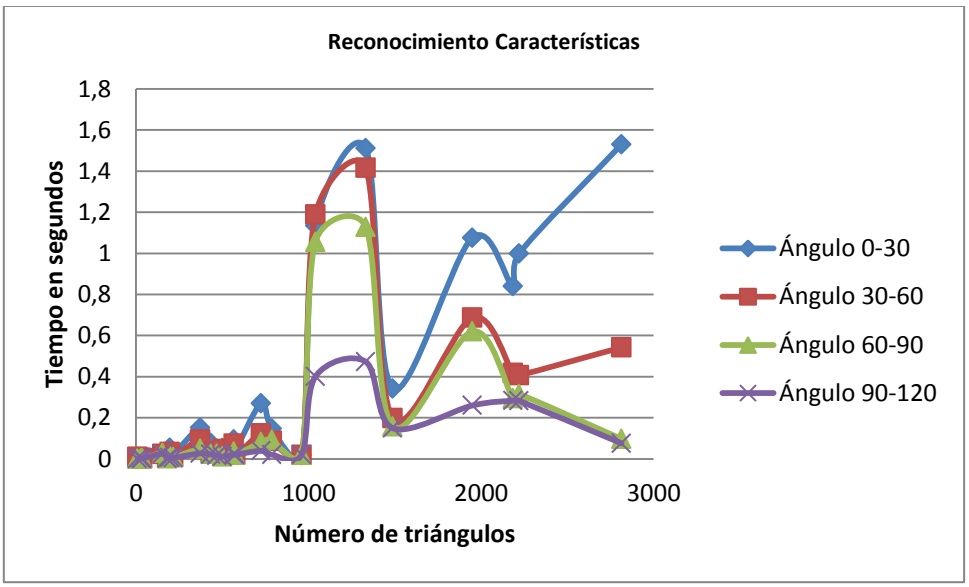
A la vista de estos resultados se concluye que el método, no solo reconoce las caras planas de forma correcta, sino que además lo hace con un ángulo de coplanaridad muy bajo, lo cual evidencia la bondad del método propuesto.

La siguiente gráfica muestra el coste temporal del algoritmo de reconocimiento de caras planas, calculado para todos los modelos 3D disponibles cuyo número de triángulos se encuentre dentro del rango 0 a 3000. Se ve reflejada una tendencia lineal entre el número de triángulos y el coste temporal, de manera general a mayor número de triángulos, se tendrán modelos con mayor número de detalles y por tanto mayor número de caras planas.



Gráfica 1: Coste de reconocimiento de caras planas.

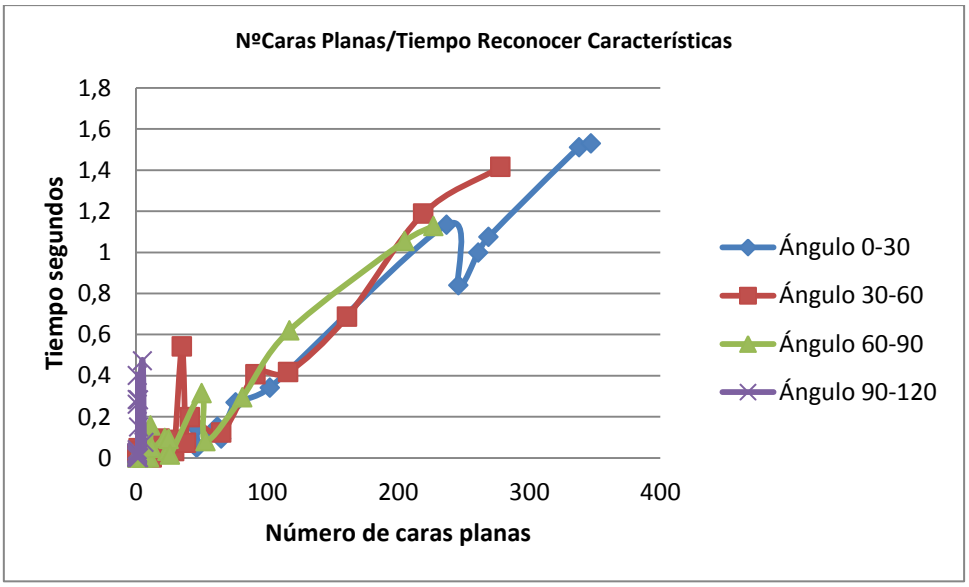
El tiempo que se tarda en reconocer las características es observado en la gráfica 2. Se pueden tener en cuenta dos factores a la vista de los resultados. En primer lugar, no se aprecia una relación directa entre el número de triángulos de la malla 3D y el tiempo que se tarda en reconocer las características de la misma. El coste temporal asociado a la detección de características en una malla 3D viene determinado por el número de caras planas existentes en el modelo tridimensional, que se analiza más adelante en la gráfica 3. Este factor se evidencia en la gráfica 2, donde en los rangos 1000 a 1500 y 1500 a 2200, se generan una serie de picos que ocurren debido a que cuando un modelo contiene un alto número de características planas, el coste temporal aumenta causando los picos de la gráfica. El segundo factor que se aprecia es que conforme aumenta el ángulo de coplanaridad que se establece, disminuye el coste temporal.



Gráfica 2: Coste de reconocimiento de características en el modelo.

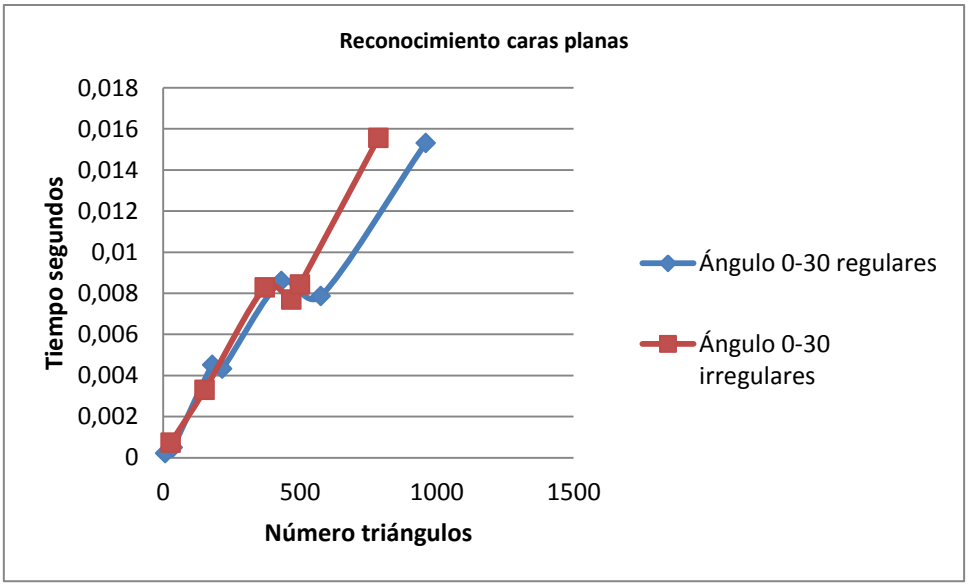
A mayor ángulo considerando para la coplanaridad de los triángulos, se detectan un menor número de caras planas, lo que se traduce en un menor número de nodos a utilizar por el algoritmo de detección de características, disminuyendo por tanto el coste temporal.

Este hecho queda demostrado en la siguiente gráfica, en la que se aprecia que, de manera general, conforme aumenta el número de caras planas, aumenta también el coste temporal. El caso especial se da cuando el ángulo de coplanaridad es de noventa grados o mayor, ya que en esta situación el modelo tiene un número muy reducido de caras planas (1 o 2 en los modelos testeados), con lo que el coste temporal apenas varía entre los diferentes modelos en comparación con el resto de rangos angulares.



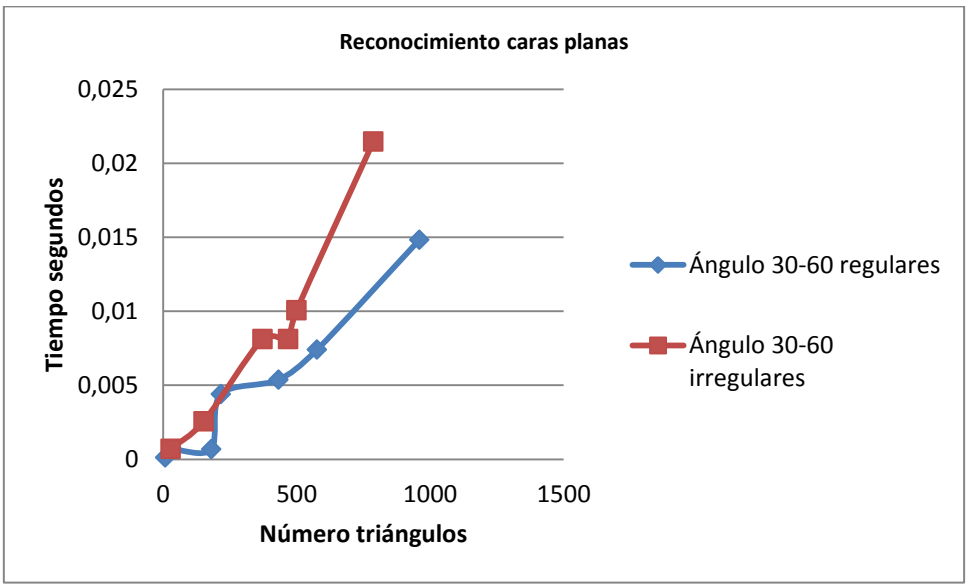
Gráfica 3: Relación entre el número de caras planas detectadas y el coste de reconocimiento de las características del modelo.

Como ya se ha dicho anteriormente dos de los conjuntos de modelos utilizados durante las pruebas son “Formas Regulares” y “Formas Irregulares”, estos modelos se han comparado entre sí a la hora de detectar caras planas, ya que los modelos regulares carecen de características y están compuestos en su mayoría por caras planas.



Gráfica 4: Reconocimiento de caras planas en Modelos Regulares vs. Irregulares.

En primera instancia se observa que, con un rango establecido para el factor de coplanaridad entre 0 y 30 grados, ambas líneas siguen una tendencia casi idéntica que tiende a aumentar el coste temporal conforme aumenta el número de triángulos. En cambio, al aumentar el ángulo, podemos ver como el coste temporal de los modelos irregulares siempre es superior (ver gráfica 5). Esto se debe a que los modelos irregulares suelen tener características curvas en contraposición a los modelos regulares, y esto hace que aumente su número de caras planas (las características curvas se aproximan en caras planas). Esta tendencia se repite conforme se aumenta en ángulo de coplanaridad.

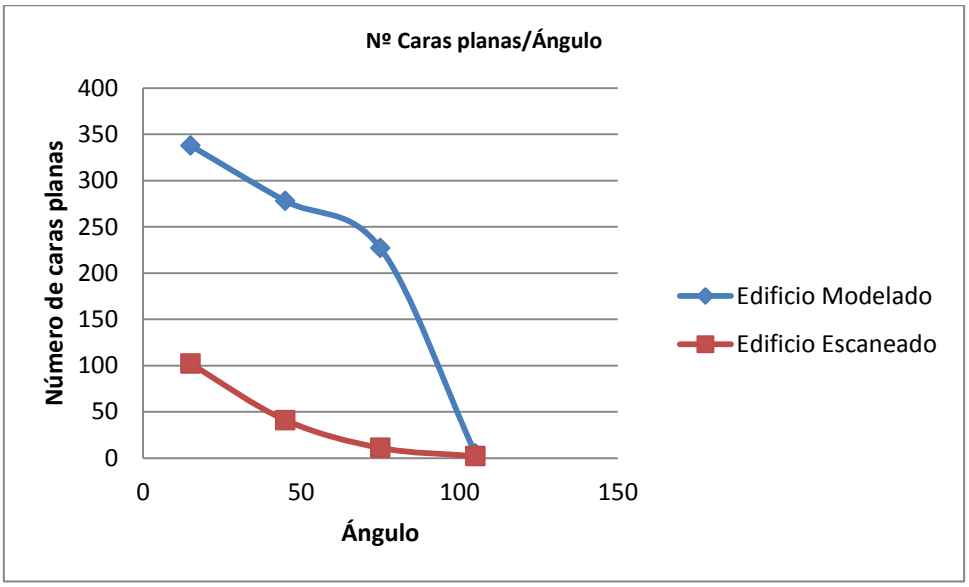


Gráfica 5: Reconocimiento de caras planas Modelos Regulares e Irregulares para un ángulo de 30 a 60.

5.3 Comparación edificio modelado vs. edificio escaneado

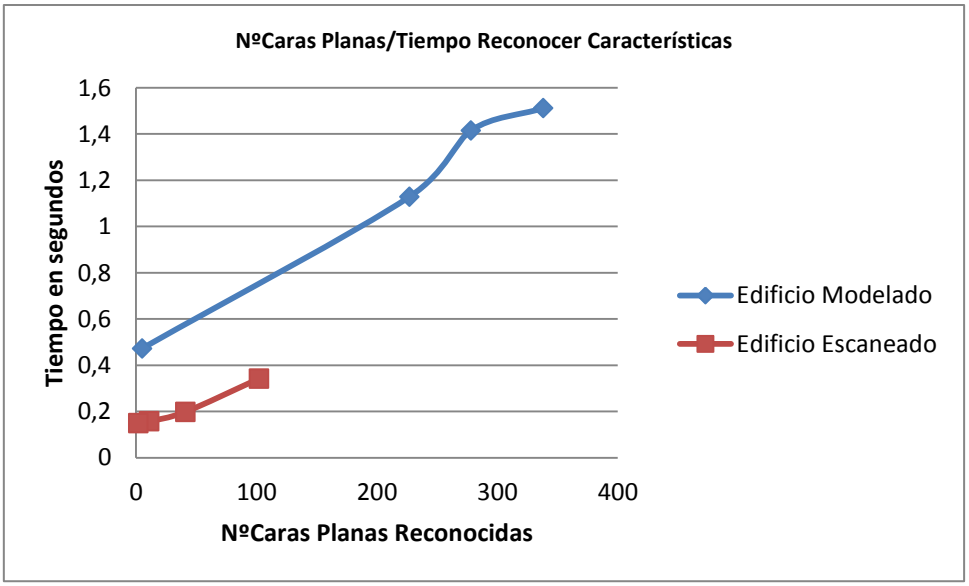
Se ha comparado la ejecución del algoritmo de detección de características sobre un modelo del grupo "Edificios modelados" con otro de "Edificios escaneados" ambos con un similar número de triángulos, (1331 triángulos para la malla que representa el edificio modelado y 1487 para la malla que representa al edificio escaneado).

Lo primero que se observa en la siguiente gráfica es que el edificio modelado dispone de un mayor número de caras planas, en comparación con el edificio escaneado y en ambos casos ocurre que al aumentar el ángulo, disminuye el número de caras planas reconocido.



Gráfica 6: Número de caras planas detectadas según el ángulo de coplanaridad.

En relación con la anterior apreciación, se muestra la siguiente gráfica, donde se refleja una disminución de los costes temporales al reconocer características conforme disminuye el número de caras planas. Este hecho ya se explicó anteriormente, ya que al haber menos caras planas el algoritmo de reconocimiento de características trabaja con grafos más pequeños y por tanto disminuye su coste temporal.



Gráfica 7: Comparación número de caras planas reconocidas y tiempo en reconocer características.

Por lo visto en las gráficas anteriores, tanto en la comparativa del coste temporal del reconocimiento de caras y características de manera global, como en la comparativa entre el edificio modelado y el edificio escaneado, se puede concluir, que el método de reconocimiento de características es directamente dependiente del número caras planas y del factor de coplanaridad establecido, e independiente del tipo de malla, tanto si es modelada como si es escaneada.

5.4 Comparación de tiempos de búsqueda de patrones

Por último, se ha hecho un estudio de los costes temporales a la hora de comparar patrones para encontrar características. En este caso se hace un estudio de 2 patrones sobre grafos con diferente número de nodos. Un patrón de 4 nodos, con el que se busca encontrar características ortogonales (ventanas) sobre las fachadas, y un patrón de 10 nodos, con el que se buscarán vigas. Ambas características se muestran en la siguiente imagen.

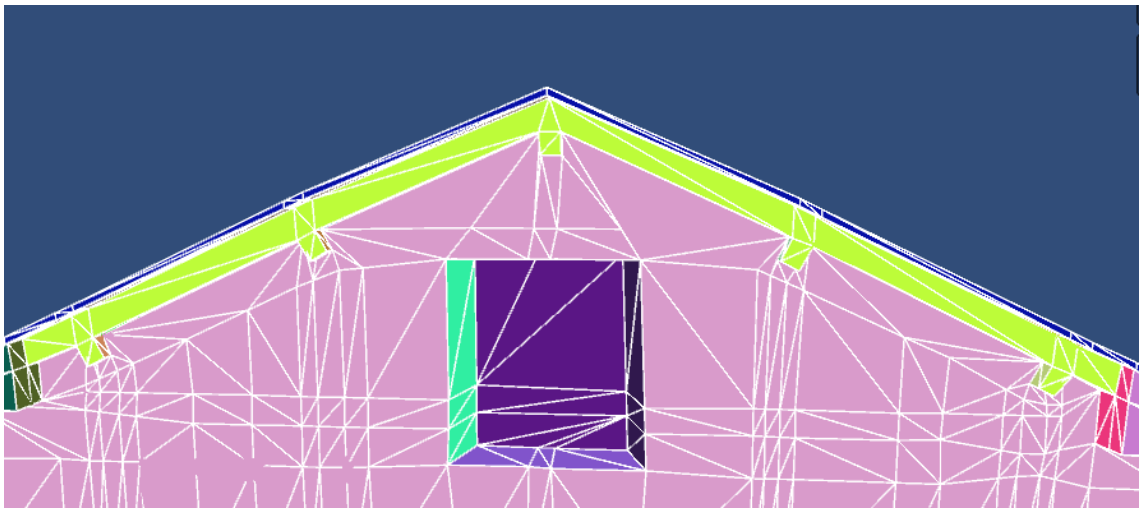
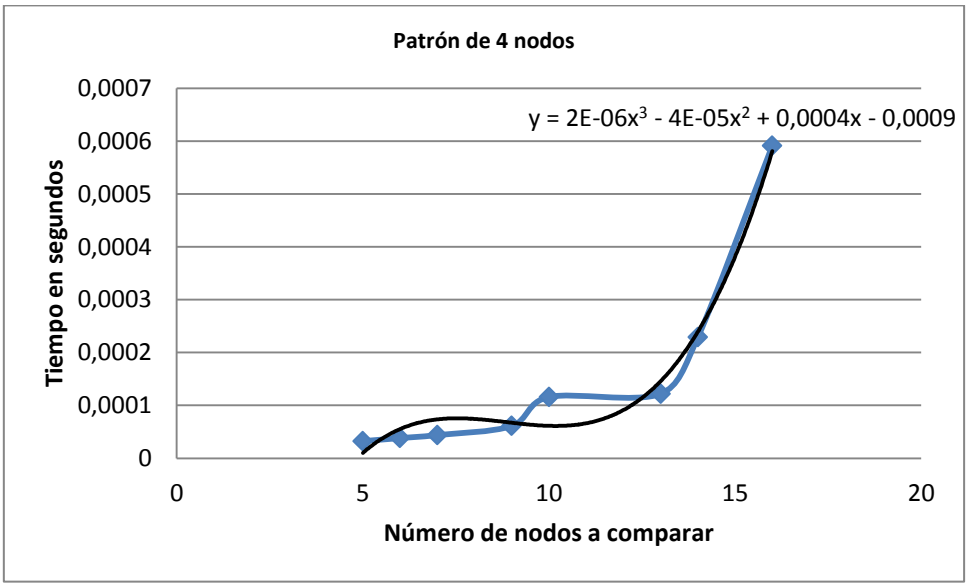


Ilustración 38: Características a buscar con los patrones de 4 nodos (ventana) y 10 nodos (viga).

En la gráfica 8 se muestra el tiempo de búsqueda de un patrón de 4 nodos. Como ya se ha explicado anteriormente, se buscan los vértices de corte del "grafo topológico" que describe el modelo, con ello conseguimos aislar las características de la fachada del edificio, quedando entonces una serie de subgrafos que son en última instancia con los que comparamos los patrones. El número de nodos que contienen estos subgrafos conforman el eje horizontal de la gráfica 8, y en el eje vertical se visualiza el coste temporal en segundos.

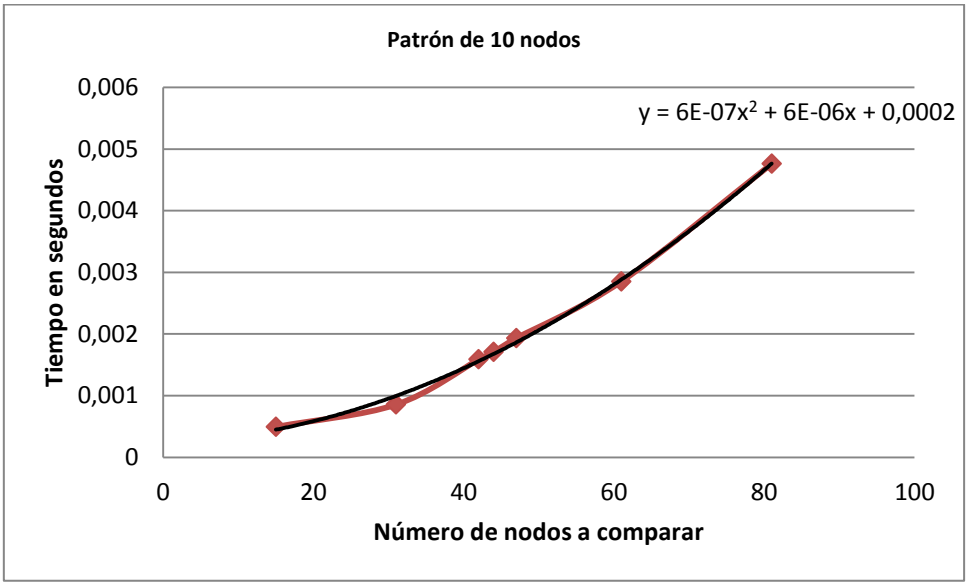
Lo que se observa es que cuando el número de nodos a comparar es bajo (de 0 a 10), el coste temporal crece muy lentamente. En cambio, con un número de nodos mayor que 10, el tiempo para buscar el patrón aumenta muy rápidamente, esto se refleja en la línea de tendencia marcada en negro que como se puede ver en la ecuación, tiene un crecimiento cúbico.



Gráfica 8: Coste temporal para la búsqueda de un patrón de 4 nodos.

En contraste cuando se busca un patrón de 10 nodos (gráfica 9), lo primero que destaca es el aumento del coste temporal, que pasa de un orden de decimas de milisegundos a milisegundos.

En este caso se observa que la línea de tendencia es cuadrática respecto al número de nodos a comparar, por lo que el coste temporal aumentará rápidamente conforme aumente este factor.



Gráfica 9: Coste temporal para la búsqueda de un patrón de 10 nodos.

Como conclusión a este ejercicio de comparación de búsqueda de características, en un modelo 3D, se puede establecer que el coste temporal depende directamente del número de nodos que conforma el patrón que define la característica. En cualquier caso, dado que la búsqueda se hace sobre una malla que ya está segmentada y que los patrones que definen características típicas en un edificio (ventanas, puertas, vigas, etc.) no incluyen un elevado número de nodos, el orden del coste temporal se sitúa en torno a los milisegundos.

6.- CONCLUSIONES Y POSIBLES AMPLIACIONES

En este capítulo se presentan las conclusiones del desarrollo de este trabajo final de master, incluyendo una descripción de las posibles líneas de trabajo futuro.

6.1 Conclusiones

La principal conclusión que se extrae de este trabajo, es que se ha formalizado un procedimiento que ha permitido la detección de características sobre modelos 3D que representan fachadas de edificios.

La metodología desarrollada incluye en primera instancia, un algoritmo de detección de caras planas de los modelos 3D. Mediante este algoritmo se analiza para cada triángulo, el grado de coplanaridad con sus vecinos, analizando el ángulo diedro que forman las normales de los triángulos. Este factor umbral es modificable, y determina el ángulo máximo que puede haber entre dos triángulos vecinos para que sean considerados de la misma cara plana. De esta manera se ha establecido grupos de triángulos con un factor de coplanaridad similar. A continuación con la ayuda de una estructura de datos denominada "MFSet", se han determinado los conjuntos de triángulos que pertenecen a la misma cara plana.

Con el algoritmo anterior, se ha creado un grafo cuyos nodos son las caras planas del modelo. Las aristas del grafo definen las relaciones de conectividad existentes entre sus nodos y por tanto la relación entre las distintas caras planas del modelo. Se llama a este grafo, "grafo topológico" de la malla 3D.

En segunda instancia se ha creado un algoritmo que utiliza el grafo anterior como entrada y en base a él detecta las características del edificio modelado. Primero se hace una detección de los vértices de corte del grafo utilizando el algoritmo DFS de recorrido de grafos. Al quitar estos nodos, se consigue tener de forma aislada pequeños subgrafos, que formaran parte cada uno de una característica. A continuación se comparan estos grafos con otros "grafos de consulta" definidos por el usuario, estos son grafos que representan la topología de las características. Los patrones a buscar quedan entonces tipificados como "grafos de consulta", estos grafos se definen mediante código de manera manual. Cuando hay una coincidencia, se asigna a los triángulos que pertenecen a las caras representadas por los nodos de los "grafos topológicos" el mismo color.

Para comprobar el correcto funcionamiento de la metodología propuesta, se ha realizado la implementación de una pequeña aplicación que permite: cargar modelos, visualizarlos y ejecutar los algoritmos descritos sobre diferentes mallas 3D.

Asimismo se han realizado una serie de pruebas sobre diferentes tipos de modelos (formas regulares, formas irregulares, edificios modelados y edificios escaneados), donde se ha comprobado la robustez de los algoritmos diseñados, y se han analizado sus costes temporales estudiando los factores que más influían sobre éstos.

6.2 Ampliaciones futuras

Una de las posibles líneas de trabajo futuro sería aumentar la robustez de los patrones. Como ya se ha visto anteriormente, existen patrones que pueden representar dos objetos diferentes por lo que se requiere de comprobaciones adicionales para poder distinguir los objetos entre sí. Ya que los "grafos de consulta" son los patrones que utilizamos, se podría dotar de información adicional a los nodos de este grafo para así hacer cada patrón más único, por lo que sería más fácil distinguir las diferentes características existentes.

Por otro lado, actualmente los patrones se definen de manera totalmente manual, a través del código de programación utilizado, no obstante esta tarea puede ser un tanto tediosa e incluso peligrosa para un usuario sin nociones de programación, ya que podría resultar fatal un cambio no debido sobre el código del programa. Para solucionar esto se propone mejorar la definición de los patrones, haciendo más accesible la creación de estos "grafos de consulta" para usuarios inexpertos en la programación.

Una manera de automatizar el proceso de creación de los "grafos de consulta" sería utilizar el algoritmo descrito en esta memoria para reconocer caras planas. En lugar de definir grafos de manera manual, se podría utilizar un modelo 3D que representara una característica que se quisiera buscar sobre otra malla, es decir, un patrón. Al disponer ya de un algoritmo de reconocimiento de caras planas que nos define un "grafo topológico" de una malla, se puede explotar para automatizar el proceso de creación de los "grafos de consulta", pues solo se tendría que usar la malla 3D que se quisiera utilizar como patrón a buscar y ejecutar el algoritmo. La desventaja radicaría entonces en la capacidad que tenga el usuario para modelar en 3D las características que se deseen encontrar.

Al automatizar estos procesos, el número de patrones crecería rápidamente, por lo que dejaría de ser eficiente tener estos recursos en la memoria del ordenador. Sería más adecuado introducir los patrones sobre una base de datos externa introduciendo computación en la "nube", que permitiría centralizar el acceso a estos recursos para multitud de usuarios.

7.- REFERENCIAS

[1]https://segmentaciondemercado.files.wordpress.com/2010/11/definicion-de-investigacion-de-mercados-segmentaciondemercado-wordpress-com.pdf

[2] Diethard Tautz, “Segmentation”, Developmental Cell, Vol. 7, 301–312, September, 2004, Copyright 2004 by Cell Press.

[3] Linda G. Shapiro and George C. Stockman (2001): “Computer Vision”, pp 279-325, New Jersey, Prentice-Hall.

[4] Xuehan Xiong, Antonio Adan, Burcu Akinci, Daniel Huber, “Automatic Creation of Semantically Rich 3D Building Models from Laser Scanner Data”, 2013.

[5] Juan Manuel Corso Sarmiento, “PROCESOS DE SEGMENTACIÓN DE NUBES DE PUNTOS Segmentación y clasificación de la tecnología de Láser Escáner Terrestre TLS”, 2012.

[6] Andrés García Morro, “Aproximación a la gestión de modelos 3D para el levantamiento arquitectónico” 2014.

[7] "Geographic Information Systems as an Integrating Technology: Context, Concepts, and Definitions". ESRI. http://www.colorado.edu/geography/gcraft/notes/intro/intro.html. 2011.

[8] Devireddy, C. R., & Ghosh, K. (1999). Feature-based modeling and neural networks-based CAPP for integrated manufacturing. International Journal of computer Integrated Manufacturing, 12(1), 61–74.

[9] Sreevalsan, P. C., & Shah, J. J. (1992). Unification of form feature definition methods. Proceedings of the IFIP WG 5.2 WorkingConference on Intelligent Computer Aided Design, 83–106.

[10] Luby, S. C., Dixon, J. R., & Simmons, M. K. (1986). Creating and using a feature database. Computers in Mechanical Engineering, 5(3), 25–33.

[11] Hiroshi Sakurai “Volume decomposition and feature recognition: Part 1 – Polyhedral Objects” 1995.

[12] Emad S. Abouel Nasr, Ali K. Kamrani “A new methodology for extracting manufacturing features from CAD system” 2006.

[13] M. Attene et al. “Mesh segmentation – A comparative study” 2006 http://cgm.technion.ac.il/Computer-Graphics-Multimedia/Publications/Papers/2006/206_Mesh_segmentation_A_comparative_study.pdf

[14] Ariel Shamir “A Survey on Mesh Segmentation Techniques” 2007.

[15] Salem Saleh Al-amri, N.V.Kalyankar and Khamitkar S.D, “Image Segmentation by Using Threshold Techniques”, Journal Of Computing, Volume 2, Issue 5, May 2010.

[16] H. G. Kaganami and Z. Beij, “Region based detection versus edge detection,” IEEE Transactions on Intelligent Information Hiding and Multimedia Signal Processing, pp. 1217-1221, 2009.

[17] S. Kullback, Information theory and statistics, Wiley, New York, 1959.

[18] I. Karoui, R. Fablet, J. Boucher, and J. Augustin, “Unsupervised region-based image segmentation using texture statistics and level-set methods,” in Proc. WISP IEEE International Symposium on Intelligent Signal Processing, 2007, pp. 1-5,2007.

[19] Y. M. Zhou, S. Y. Jiang, and M. L. Yin, “A region-based image segmentation method with mean-shift clustering algorithm,” in Proc. Fifth International Conference on Fuzzy Systems and Knowledge Discovery, 2008, pp. 366-370

[20] Konstantinos G. Derpanis “Mean Shift Clustering”, August 15, 2005

[21] C. Cigla and A. A. Alatan, “Region-based image segmentation via graph cuts,” in Proc. 15th IEEE International Conference on Image Processing, 2008, pp. 2272-2275.

[22] S. Lakshmi and D. V. Sankaranarayanan, “A study of edge detection techniques for segmentation computing approaches,” IJCA Special Issue on “Computer Aided Soft Computing Techniques for Imaging and Biomedical Applications” CASCT, 2010.

[23] X. Yu and J. Yla-Jaaski, “A new algorithm for image segmentation based on region growing and edge detection,” in Proc. IEEE International Sympoisum on Circuits and Systems, pp. 516-519, 1991.

[24] Slavo Wesolkowski and Paul Fieguth, “A markov random fields model for hybrid edge-an region-based color image segmentation”, 2002

[25] Ying-Tung Hsiao, Cheng-Long Chuang, Joe-Air Jiang, and Cheng-Chih Chien, “A Contour based Image Segmentation Algorithm using Morphological Edge Detection”, 2005 IEEE International Conference on Systems, Man and Cybernetics Waikoloa, Hawaii October 10-12, 2005.

[26] Dong Hu Xianzhong Tian, “A Multi-directions Algorithm for Edge Detection Based on Fuzzy Mathematical Morphology”, 2006.

[27] Karmakar, G. C. and Dooley, L. S. (2001). “A generic fuzzy rule based technique for image segmentation.” In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP ’01), 7-11 May 2001, Salt Lake City, Utah.

[28] A. S. Pednekar and I. A. Kakadiaris, “Image segmentation based on fuzzy connectedness using dynamic weights,” IEEE Transactions on Image Processing, vol. 15, pp. 1555-1562, 2006.

[29] L. Yaju, Z. Baoliang, Z. Li, L. Dongming, C. Zhenjiang, and L. Lihua, “Research on image segmentation based on fuzzy theory,” in Proc. WRI World Congress on Computer Science and Information Engineering, 2009, pp. 790-794.

[30] James C.Bezdek, Robert Ehrlich, William Full , “FCM: THE FUZZY c-MEANS CLUSTERING ALGORITHM” 1984

[31] Martin Hagan, “Neural Network Design”, Course Announcement, 2007.

[32] Simon Haykin, “Neural Networks, A comprehensive foundation”, Prentice Hall.

[33] D. J. Evans and L. P. Tay, "Fast Learning Artificial Neural Networks for continuous input applications," Kybernetes, vol. 24, pp. 11-23, 1995.

[34] X. Zhang and A. L. P. Tay, “Fast learning artificial neural network (FLANN) based color image segmentation in RGBSV cluster space,” in Proc. International Joint Conference on Neural Networks, 2007, pp. 563-568.

[35] C.L. Chang and Y.T. Ching, “ Fuzzy Hopfield neural network with fixed weight for medical image segmentation ” Optical Engineering, vol. 41, pp. 351-358, 2002.

[36] F. M. Kazemi, M. R. Akbarzadeh, T. S. Rahati, and H. Rajabi, “Fast image segmentation using C-means based Fuzzy Hopfield neural network,” in Proc. Canadian Conference on Electrical and Computer Engineering, 2008, pp. 001855-001860.

[37] Brian Taylor, Alper Ayvaci, Avinash Ravichandran, and Stefano Soatto “Semantic Video Segmentation From Occlusion Relations Within a Convex Optimization Framework” 2013.

[38] P. Felzenszwalb and D. Huttenlocher. “Efficient graph-based image segmentation”. IJCV, 59(2), 2004.

[39] Matthias Grundmann, Vivek Kwatra, Mei Han, Irfan Essa, “Efficient Hierarchical Graph-Based Video Segmentation”, 2010.

[40] John S. Boreczky and Lynn D. Wilcox, “Hidden Markov Model Framework for Video Segmentation Using Audio and Image Features”, 1998.

[41] Aparajithan Smpath and Jie Shan “Segmentation and Reconstruction of Polyhedral Building Roofs From Aerial Lidar Point Clouds” 2010.

[42] R.O. Duda and P.E.Hart, “Use of the Hough transformation to detect lines and curves in pictures”, Commun. ACM, vol. 15, no. 1, pp. 11-15, Jan. 1972.

[43] G. Vosselman and S. Dijkman, “3D building model reconstruction from point clouds and ground plans,” Int. Arch. Photogramm. Remote Sens.,vol. 34, no. 3/W4, pp. 37–43, 2001

[44] J. Overby, L. Bodum, E. Kjems, and P. M. Iisoe, “Automatic 3D building reconstruction from airborne laser scanning and cadastral data using Hough transform,” Int. Arch. Photogramm. Remote Sens., vol. 35, pt. B3, pp. 296–301, 2004

[45] M. Fischler and R. Bolles, “Random sample cosenus: A paradigm for model fitting with applications to image analysis and automated cartography”, Commun. ACM, vol. 24, no. 6, pp. 381-395, 1981.

[46] Hakim Boulaassal, Tania Landes, Pierre Grussenmeyer, Fayez Tarsha-Kurdi."Automatic segmentation of building facades using Terrestrial Laser Data." ISPRS Workshop on Laser Scanning 2007 and SilviLaser 2007, Sep 2007, Espoo, Finland. pp.65-70.

[47] Philipp Jenke, Bastian Kr¨uckeberg, Wolfgang Straßer, "Surface Reconstruction from Fitted Shape Primitives", 2008.

[48] Hai Huang and Claus Brenner, "Rule-based Roof Plane Detection and Segmentation from Laser Point Clouds", 2011.

[49] Aparajithan Sampath and Jie Shan, "Segmentation and Reconstruction of Polyhedral Building Roofs From Aerial Lidar Point Clouds", 2010.

[50] Shi Pu and George Vosselman, "Building Facade Reconstruction by Fusing Terresrial Laser Points and Images", 2009.

[51] Bruno Kewitz Demarchi, Felipe Pilon, "Canny Edge Detector".

[52] William S Noble , "What is a support vector machine?", NATURE BIOTECHNOLOGY VOLUME 24 NUMBER 12 DECEMBER 2006

[53] Rusu, R.B. Blodow, N. ; Marton, Z.C. ; Beetz, M., "Close-range scene segmentation and reconstruction of 3D point cloud maps for mobile manipulation in domestic environments", 2009.

[54] Ruwen Schnabel, Raoul Wessel, Roland Wahl, Reinhard Klein"Shape Recognition in 3D Point-Clouds", 2008.

[55] Colin Smith, On Vertex-Vertex Meshes and Their Use in Geometric and Biological Modeling, http://algorithmicbotany.org/papers/smithco.dis2006.pdf

[56]Bruce G. Baumgart, Winged Edge Polyhedron representation. http://www.dtic.mil/dtic/tr/fulltext/u2/755141.pdf

[57] Besl, P. J.; Jain, R.: "Segmentation through Variable-Order Surface Fitting" , IEEE PAMI, 10(2), 1988, 167-192.

[58] Vieira, M.; Shimada, K.: "Surface mesh segmentation and smooth surface extraction through region growing", Computer-Aided Geometric Design, 22(8), 2005, 771-792.

[59] Zhang, Y.; Paik, J.; Koschan, A.; Abidi, M. A.: "A simple and efficient algorithm for part decomposition of 3D triangulated models based on curvature analysis", Proc. Intl. Conf. on Image Processing, 2002, 273-276.

[60] Mangan, A. P.; Whitaker, R. T.: "Partitioning 3D surface meshes using watershed segmentation", IEEE Transactions on Visualization and Computer Graphics, 5(4), 1999, 308-321.

[61] Zuckerberger, E.; Tal, A., Shlafman, S.: "Polyhedral surface decomposition with applications", Computers Graphics, 26(5), 2002, 733-743.

[62] Page, D. L.; Koschan, A.; Abidi, M.: “Perception-based 3D triangle mesh segmentation using fast marching watersheds”, In Proc. of Computer Vision and Pattern Recognition, 2003, 27-32.

[63] L. De Floriani, “A graph based approach to object feature recognition”, 1987.

[64] Marefat, M., Kashyap, R., “Geometric reasoning for recognition of three-dimensional object features”, 1990.

[65] Wu, K.; Levine, M. D.: 3D Part Segmentation Using Simulated Electrical Charge Distributions, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1997, 1223-1235.

[66] Li, X.; Toon, T. W.; Huang, Z.: Decomposing polygon meshes for interactive applications, SI3D, 2001, 35-42.

[67] Shlafman, S.; Tal, A.; Katz, S.: Metamorphosis of polyhedral surfaces using decomposition, Eurographics,2002, 219-228.

[68] Chris Tralie, http://www.ctralie.com/Teaching/MeshSeg/

[69] Liu, R.; Zhang, H.: Segmentation of 3D meshes through spectral clustering, Pacific Conference on Computer Graphics and Applications, 2004, 298–305. REPETIDO

[70] Jiajun Lv, Xinlei Chen, Jin Huang, Hujun Bao, “Semi-supervised Mesh Segmentation and Labeling”, 2012.

[71] Lin Liao, Tanzeem Choudhury, Dieter Fox, Henry Kautz, “Training Conditional Random Fields using Virtual Evidence Boosting”, Proc. of the International Joint Conference on Artificial Intelligence (IJCAI), 2007

[72] Katz, S.; Tal, A.: Hierarchical Mesh Decomposition Using Fuzzy Clustering and Cuts, ACM Trans. on Graphics, 22(3), 2003, 954-961.

[73] Lin, H. S.: Liao, H. M.: Lin, J.: Visual Salience-Guided Mesh Decomposition, IEEE Int. Workshop on Multimedia Signal Processing, Siena (ITALY), 2004, 331-334.

[74] Katz, S.; Leifman, G.; Tal, A.: Mesh Segmentation using Feature Point and Core Extraction, The Visual Computer, 21(8-10), 2005, 649-658.

[75] Aleksey Golovinskiy, Vladimir G. Kim, Thomas Funkhouser, “Shape-based Recognition of 3D Point Clouds in Urban Environments”.

[76] Yangyan Li, Qian Zheng, Andrei Sharf, Daniel Cohen-Or, Baoquan Chen, Niloy J. Mitra. “2D-3D Fusion for Layer Decomposition of Urban Facades”, 2011.

[77] Ali, Haider ; Ahmed, Basheer ; Paar, Gerhard, “Robust Window Detection from 3D Laser Scanner Data”, 2008.

[78] A. K. Aijazi, P. Checchin and L. Trassoudaine, “Automatic Dectection and Feature Estimation of Window For Refining Building Facades in 3D Urban Point Clouds”, 2014.

[79] Guowei Wana, Andrei Sharf, “Grammar-based 3D facade segmentation and reconstruction”, 2012.

[80] Alexander Agathos, Ioannis Pratikakis, Stavros Perantonis, Nikolaos Sapidis and Philip Azariadis “3D Mesh Segmentation Methodologies for CAD applications”

[81] Kenton McHenry and Peter Bajcsy, “An Overview of 3D Data Content, File Formats and Viewers”, 2008

[82] Hopcroft, J.; Tarjan, R. (1973). "Algorithm 447: efficient algorithms for graph manipulation". Communications of the ACM 16 (6): 372–378.

[83] Wegener, Ingo (2005), Complexity Theory: Exploring the Limits of Efficient Algorithms, Springer, p. 81.

[84] de la Higuera, Colin; Janodet, Jean-Christophe; Samuel, Émilie; Damiand, Guillaume; Solnon, Christine (2013), "Polynomial algorithms for open plane graph and subgraph isomorphisms", Theoretical Computer Science 498: 76–99.

[85] J. R. ULLMAN, “An algorithm for Subgraph Isomorphism”.1976

[86] Jinsoo Lee, Wook-Shin Han, Romans Kasperovics, Jeong-Hoon Lee, “An In-depth Comparison of Subgraph Isomorphism Algorithms in Graph Databases”, 2012.