Revista Iberoamericana de Automática e Informática Industrial

Revista Iberoamericana de Automática e Informática Industrial 17 (2020) 22-33

www.revista-riai.org

Reconstrucción virtual tridimensional de entornos urbanos complejos

A. García-Moreno[a,b,*], J. González-Barbosa[a]

[a] *Centro de Investigación en Ciencia Aplicada y Tecnología Avanzada (CICATA), Instituto Politécnico Nacional. Cerro Blanco 141 Col. Colinas del Cimatario, Querétaro. México.*

[b] *Centro de Ingeniería y Desarrollo Industrial (CIDESI), Av. Playa Pie de la Cuesta No. 702, Desarrollo San Pablo, 76125, Santiago de Querétaro, Qro.*

Resumen

Este trabajo presenta una metodología para la generación de modelos tridimensionales de entornos urbanos. Se utiliza una plataforma terrestre multi-sensores compuesta por un LIDAR, una cámara esférica, GPS y otros sistemas inerciales. Los datos de los sensores están sincronizados con el sistema de navegación y georrefenciados. La metodología de digitalización se centra en 3 procesos principales. (1) La reconstrucción tridimensional, en el cual se elimina el ruido en los datos 3D y se disminuye la distorsión en las imágenes. Posteriormente se construye una imagen panorámica. (2) La texturización, se describe a detalle el algoritmo para asegurar la menor incertidumbre en el proceso de extracción de color. (3) La generación de mallas, se describe el proceso de mallado basado en octree's, desde la generación de la semilla, el teselado, así como la eliminación de huecos en las mallas. Por último, se realiza una evaluación cuantitativa de la propuesta y se compara con otros enfoques existentes en el estado del arte. Se discuten a detalle los resultados obtenidos.

Palabras Clave:
Reconstrucción 3D, Texturizado, Mallado, LIDAR

Virtual 3D reconstruction of complex urban environments

Abstract

This paper presents a methodology for the generation of three-dimensional models of urban environments. A multi-sensor terrestrial platform composed of a LIDAR, a spherical camera, GPS and IMU systems is used. The data of the sensors are synchronized with the navigation system and georeferenced. The digitalization methodology is focused on 3 main processes. (1) The three-dimensional reconstruction, in which the noise in the 3D data is eliminated and the distortion in the images is reduced. Later, a panoramic image is built. (2) Texturing, the algorithm is described in detail to ensure the least uncertainty in this color extraction process. (3) Mesh generation, the meshing process based on octree's is described, from the generation of the seed, the tessellation, as well as the elimination of gaps in the meshes. Finally, a quantitative evaluation of our proposal is made and compared with other existing approaches in the state-of-the-art. The results obtained are discussed in detail.

Keywords:
3D reconstruction, texturing, Meshing, LIDAR

1. Introducción

Recientemente muchos sectores, tanto industriales como científicos, han visto en la reconstrucción 3D, un área de oportunidad con grandes beneficios. El modelado y la reconstrucción de ciudades urbanas de gran escala tienen una variedad de aplicaciones con gran valor añadido. Las limitaciones de hardware, el procesamiento y el almacenamiento hoy en día ya no son un impedimento para procesar grandes cantidades de datos. La reconstrucción de entornos urbanos tiene aplicaciones prometedoras en campos como la arquitectura, la ingeniería civil, la gestión de ciudades, los videojuegos, la navegación y la seguridad, por nombrar algunos. Hay comercialmente muchos tipos de sensores que utilizan la tecnología LIDAR (Light Detection and Ranging). Estos sensores permiten captar la información tridimensional de los entornos urbanos. Esta información nos permite representar todas las geometrías de la escena urbana y los objetos que la componen.

*Autor para correspondencia: angelivan.garciam@gmail.com
To cite this article: A. García-Moreno, J. González-Barbosa, 2020. Virtual 3D reconstruction of complex urban environments. Revista Iberoamericana de Automática e Informática Industrial 17, 22-33. https://doi.org/10.4995/riai.2019.11203
Attribution-NonCommercial-NoDerivatives 4,0 International (CC BY-NC-ND 4,0)

El proceso de reconstrucción es una tarea que requiere mucho tiempo y sólidas capacidades computacionales para procesar la enorme cantidad de datos adquiridos. La idea principal es generar un mapa que represente con exactitud todo el entorno urbano. La diversidad de formas en una escena urbana requiere un post-procesamiento manual con la menor intervención hu-mana. Por no mencionar la baja calidad de los datos obtenidos que en la mayoría de los casos son causados por una mala cali-bración o el efecto multi-trayecto. Otro problema son las oclu-siones que en el proceso de mallado generan discontinuidades en la construcción de la superficie. En consecuencia, es impera-tivo desarrollar metodologías para optimizar y automatizar to-das las tareas involucradas en la reconstrucción urbana.

En este artículo se propone una metodología completa para generar modelos tridimensionales (texturizados volumétricos) utilizando la información procedente de una plataforma móvil terrestre. Integrada por: un sensor LIDAR, una cámara esférica, un GPS diferencial y varios sistemas inerciales. El artículo se enfoca sólo en los procesos de generación de mapas globales tridimensionales, su posterior texturizado y finalmente la generación de las superficies. Dejando de lado cuestiones como la calibración de los sensores y la calibración de la propia plataforma de adquisición. Además de los subprocesos como correlación entre los diferentes sensores y el análisis de la propagación del error, la incertidumbre y la sensibilidad son temas que ya han sido abordados en trabajos previos, García-Moreno et al. (2016).

Después de realizadas las tareas de calibración y puesta en marcha de la plataforma de adquisición presentada por García-Moreno et al. (2013), los datos se capturan dinámicamente. Es decir, la plataforma se traslada dentro de un entorno urbano. Donde muchos objetos también están en movimiento, coches y gente principalmente. Los datos de la cámara se procesan después para poder generar una imagen panorámica con la menor distorsión radial y tangencial. Los datos provenientes del LIDAR se muestran, se suavizan y se disminuye el ruido. La textura se obtiene y posteriormente se correlaciona con los datos tridimensionales. En un siguiente paso, las diferentes nubes de puntos se fusionan para generar un mapa de escala global. Con la nube de puntos del mapa global, se generan las superficies. Este proceso esta basado en *octrees*. La malla creada también se post-procesa reduciendo el número de huecos creados debido a las oclusiones.

El artículo se organiza de la siguiente manera: en esta primera sección, se abordan los trabajos relacionados con los procesos de reconstrucción, texturizado y mallado. Se continua con la descripción de la plataforma de adquisición, y posteriormente de la metodología propuesta. Para continuar con los resultados y su discusión. Y finalmente, se presentan las conclusiones. Las principales aportaciones de este trabajo son: (1) se diseñó una metodología funcional para la reconstrucción automática de ambientes urbanos utilizando una plataforma terrestre, (2) se propone un método robusto para la extracción de texturas que minimiza la distorsión radial y tangencial y (3) se propone un algoritmo de mallado capaz de reducir significativamente las oclusiones y la corrupción del ruido.

1.1. Reconstrucción tridimensional.

Las capacidades tecnológicas actuales permiten que los escáneres láser terrestres digitalicen grandes áreas. Estos sistemas permiten la generación directa de nubes de puntos tridimensionales. El aspecto visual de las escenas es muy complejo de interpretar debido a la interacción entre la geometría de los objetos, las oclusiones existentes en el campo de visión de los sensores y las condiciones de iluminación. Esta complejidad es lo que hace que el problema sea sumamente interesante desde la perspectiva de la investigación. También significa que el problema no tiene una solución trivial.

Este problema a sido abordado por diversos investigadores. Por ejemplo Wu et al. (2017) presenta un enfoque baso en grafos para detectar la estructura topológica de los edificios, luego separan los edificios en diferentes partes mediante el análisis de sus relaciones topológicas y finalmente reconstruye el modelo de construcción mediante la integración de todos los modelos individuales establecidos a través de un proceso de emparejamiento de grafos bipartitos. Es un método robusto a la perspectiva del LIDAR y proporciona una descripción topológica y geométrica muy completa de los modelos 3D.

El entorno tridimensional es una forma interactiva, intuitiva y fácil para ver y usar información basada en la ubicación. En Abayowa et al. (2015) presenta el desarrollo de un proceso semiautomático para generar modelos 3D texturizados utilizando técnicas de fotográmetría. Primero se definen las mejores prácticas para la adquisición de datos (imágenes), posteriormente se desarrolla un algoritmo para la superposición óptima de las imágenes consecutivas. Lo que permitirá generar suficiente información para texturizar cada dato tridimensional. Los autores desarrollan un software para realizar el proceso de recostrucción urbana utilizando datos texturizados. Chen et al. (2018) presenta un modelo para la reconstrucción automática de modelos digitales de edificios (MDE) en áreas urbanas de alta densidad. Al recuperar los datos geográficos de la base, la altura y la huella del edificio de mapas topográficos, se construyen los MDE usando los datos de un LIDAR para detectar las primitivas geométricas de los edificios. Posteriormente se rectificaron utilizando el conocimiento arquitectónico sobre las geometrías . Por ejemplo, la azotea normalmente estaría en paralelo con el borde exterior del techo. Los autores muestra el resultado de reconstruir 1361 edificios en el área de Hong Kong.

Yi et al. (2017) presenta un método para generar estructuras volumétricas de edificios urbanos directamente de los datos sin procesar del LiDAR. Los autores primero segmentan las nubes de puntos en subconjutos, cada uno de ellos contiene una estructura urbana (edificios, casas, etc). Recursivamente van dividiendo los datos 3D en clústers mas pequeños para facilitar la reconstrucción. Un enfoque similar es presentado por Kim and Hilton (2015) pero utilizando unicamente imágenes.

Otra aplicación de la reconstrucción tridimensional es presentada en Hatger and Brenner (2003). Los autores utilizan los datos de un LIDAR junto con la información existente en bases de datos libres para estimar la geometría de las carreteras y algunos elementos urbanos estáticos. Estas bases de datos contienen información como las coordenadas geográficas, el ancho de las carreteras, el número de carriles, el tipo de vehículos que pueden circular sobre él, su gradiente, señales de tráfico, etc. La mezcla de los datos recopilados con los datos almacenados en los repositorio requiere un proceso de correlación muy preciso. Que a menudo se acompaña con sistemas inerciales.

Hamzah et al. (2018); Yang et al. (2016); Zolanvari et al. (2018) se describen varias metodologías que fusionan y correlacionan la información proveniente de diferentes sensores. Se describen enfoques basado en estereoscopía óptica y enfoques estáticos, donde es más simple generar una relación entre las imágenes, los datos 3D y la posición georreferenciada.

1.2. Texturización

Cuando se construyen modelos 3D de escenas urbanas reales, la textura juega un papel importante. Esta textura definirá objetos y dará mayor percepción visual a la profundidad

y perspectiva. El proceso de extracción de color de las imágenes y su posterior fusión con los datos tridimensionales puede ser dividido en 3 enfoques:

1.2.1. Enfoques estáticos

Si la plataforma de reconstrucción no se mueve y tampoco lo hacen los objetos que conforman la escena urbana, se esta hablando de un enfoque estático. Una metodología robusta y eficiente es presentada por Liu and Stamos (2012). La orientación de la cámara es recuperada haciendo un emparejamiento con los puntos de fuga (*vanishing points*) encontrados en las imágenes con respecto a las direcciones 3D calculadas para el modelo tridimensional. Después se calcula la posición de la cámara con respecto al láser haciendo coincidir características lineales encontradas en los datos de ambos sensores. Las posiciones se optimizan minimizando la distancia línea-a-línea entre estas características. El método es bastante eficiente en comparación con otros trabajos como los presentados por Iwaszczuk and Stilla (2017); Chen et al. (2018), ya que no necesitan características complejas en la información de ambos sensores (por ejemplo planos) para calcular la relación de los datos.

Otro enfoque se presenta en Yang et al. (2011). En este trabajo los autores usan la información capturada con el láser para mejorar la fusión de las imágenes y después utilizan características geométricas para mejorar la unión de nubes de puntos capturadas en distintas zonas. Su investigación esta enfocada en dos pasos. Primero fusionan las imágenes con los datos 3D del láser y se obtienen las correspondencias entre los sensores para después texturizar utilizando el algoritmo *SIFT* (Scale-invariant feature transform). En una segunda etapa se desarrolla un método para generar características invariantes a la perspectiva y con ellas poder fusionar nubes de puntos adquiridas en diferente tiempo. Primero se extraen los planos en la nube de puntos y se usan para describir la distribución espacial de la escena. En la imagen las características extraídas son normalizadas con respecto a la distribución 3D para lograr la invarianza. Después son rectificadas para formar la parte frontal de las fachadas de los edificios en la escena. Al contar con la correspondencia píxel a píxel de las características en varias adquisiciones, se hace la fusión de diferentes nubes de puntos a diferentes perspectivas.

1.2.2. Enfoques dinámicos

Cuando la plataforma de reconstrucción es móvil está sometida a tareas que alteran la posición de los sensores en la plataforma, por ejemplo vibraciones. En este escenario un enfoque de calibración que no este basado en un patrón es adecuado ya que la tarea de calibración se vuelve más sencilla. En Pandey et al. (2014) se presenta un algoritmo para la estimación automática de la transformación rígida LIDAR-cámara sin el uso de un patrón de calibración. La correlación de los datos se basa en el supuesto de que existen características en común entre la reflectividad de los datos 3D y la intensidad de los píxeles en las imágenes. Los autores presentan una función de costo que maximiza la relación estadística de los datos, donde la varianza de los datos alcanza la cota de *Cramér-rao*. Además su metodología puede ser utilizada para cualquier tipo de sensor láser que devuelva información de reflectividad y cualquier cámara esférica.

Wang et al. (2012) plantea una metodología para la texturización de entornos urbanos mediante la conversión de las coordenadas del láser a un sistema euclidiano llamado ECEF (*Earth-centered, Earth-fixed*). Despúes son transformadas al sistema de coordenadas LTP (*Local tangent plane*) para tener los datos de ambos sensores en las mismas unidades. Posteriormente se fusionan varios mapas 3D para generar un mapa de mayor resolución y calcular, según la perspectiva elegida, que datos del LIDAR pueden ser proyectados sobre la imagen esférica y poder interpolar el color que se le asignará al punto 3D.

Por otro lado, Zeng and Zhong (2014) describe una plataforma similar a la utilizada en este trabajo. Los autores toman ventaja del principio de colinealidad al utilizar sensores con un campo de visión de 360°. En donde el centro de la cámara omnidireccional, un punto en la imagen sobre la esfera y el punto del objeto están alineados. Después se describe un algoritmo que transforma las coordenadas geodésicas de las nubes de puntos y las proyecta sobre la sección del plano que le corresponde, después se obtiene la textura y se genera el mapa texturizado. Un enfoque similiar se presenta en Shi et al. (2015).

1.3. Mallado

El proceso de generar superficies a partir de datos tridimensionales de una zona urbana se ha vuelto un problema de interés entre la comunidad científica. La rapidez de la triangulación, la precisión y la exactitud del mallado, así como la robustez del procedimiento de reconstrucción son factores importantes que se evalúan en la generación de las superficies.

Una malla triangular M consiste en un componente geométrico y otro topológico, donde la superficie esta representada por un conjunto de vértices $V = \{v_1, v_2, \ldots, v_K\}$, y un conjunto de caras triangulares $T = \{t_1, t_2, \ldots, t_m\}, \forall\ t_i \in \mathbb{R}^3$ que los conectan. La conectividad de la malla también viene definida en términos de las aristas $A = \{a_1, a_2, \ldots, a_z\}, \forall\ a_i \in \mathbb{R}^2$. Por lo tanto, la superficie esta formada por una red de triángulos y cuya conectividad esta defina por la terna $\{V, A, T\}$.

Actualmente existen diferentes dispositivos 3D capaces de entregar datos con una resolución micrométrica. Por un lado hace posible reconstruir objetos pequeños pero también generan una densidad enorme de datos. Según Di Angelo et al. (2011), los algoritmos de reconstrucción de superficies se dividen en dos categorías: métodos implícitos y métodos explícitos.

Los métodos implícitos reconstruyen una función $f(p) = 0$, donde p es una nube de puntos. El mallado se obtiene al extraer la iso-superficie para $f(p) = 0$. Estos métodos llevan a cabo una reconstrucción cerrada, no importando si se tienen pocos puntos. Esto implica que se requieran muchos cálculos, además se requiere la normal de los puntos y la normal principal de la superficie. Una desventaja de estos métodos es que la triangulación no es perfecta, ya que existen puntos que no son tomados en cuenta, lo que genera pérdida de detalles. Por ejemplo el método Función de base Radial (RBF por sus siglas en inglés) presentado por Carr et al. (2001).

Los métodos explícitos por su parte, generan la triangulación directamente con los puntos, sin la necesidad de las normales u otros cálculos. Esto hace que sean menos precisos, aunque si son más rápidos. Estos métodos se dividen en dos categorías: los enfoques basados en Voronoi/Delaunay Dey and Goswami (2004); Lhuillier (2018); dou y los basados en crecimiento de malla Digne (2014); Guo et al. (2016); Morel et al. (2018).

El trabajo realizado en esta investigación se clasifica como: a) una plataforma terrestre de reconstrucción, b) un enfoque dinámico de texturización y c) un método de crecimiento de malla en la generación de superficies.

2. Descripción del sistema

Nuestra plataforma multi-sensores (ver Figura 1) esta compuesta por un escáner láser *Velodyne HDL-64E*, una cámara panorámica *Point Grey Ladybug2*, un GPS *ProMark3* con tecnología RTK (Real Time Kinematic) y un giroscopio *CHR UM6*.

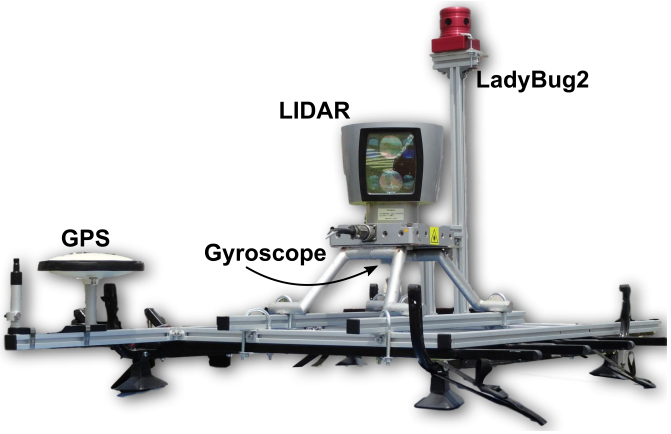

Figura 1: Plataforma multi-sensores compuesta por un LIDAR Velodyne HDL-64E (*L*), una cámara *Ladybug2* (*LB*), un GPS direfencial de alta presición y un giroscopio *CHR UM6*.

El LIDAR mide el tiempo de cuelo de un haz láser desde que es emitido hasta que regresa al ser reflejado por la superficie de un objeto. Esta compuesto por un arreglo de 64 láseres que giran sobre su eje vertical ofreciendo un campo de visión de 360°. Verticalmente el LIDAR cuenta con un ángulo de visión de 26,8°. Cada láser es pulsado cada 4 nanosegundos. La cámara *Ladybug2* es un sensor omnidireccional de alta resolución compuesto por seis cámaras de 0,8 − *Megapixeles*. La forma de anillo en la que están dispuestas cinco de las seis cámaras permite tener imágenes con una visión de más del 75 % de la esfera total. El GPS es un dispositivo de frecuencia simple y que utiliza una doble constelación (GPS+SBAS) el cual nos permite levantamientos GNSS con una precisión centimétrica. El giroscopio nos provee de medidas de orientación en ángulos de Euler y cuaternios, además tiene la capacidad de interactuar con el GPS y proveer la información de posición, velocidad, rumbo y velocidad.

La Figura 2 muestra todas las relaciones de la plataforma de reconstrucciñon. Un punto reconstruido por el LIDAR (*ds*, θ, φ) se transforma a $[X^L, Y^L, Z^L]$. Donde *ds*, es la distancia a la que se encuentra el objeto en el que se refleja el láser, θ es el ángulo cenital, y φ el ángulo azimutal al que esta orientado el láser. Este punto se proyecta a la imagen utilizando los parámetros intrínsecos de la cámara A^{Ci} (matriz de parámetros intrínsecos de la cámara C_i) y la transformación rígida entre la cámara y LIDAR $[R_L{}^{Ci}, T_L{}^{Ci}]$. La transformación LIDAR-cámara se calcula como se define en ecuación 1, este modelo se describe en trabajos previosGarcáa-Moreno et al. (2013).

$$[R,T]_L^{Ci} = [R,T]_W^{Ci} * ([R,T]_W^L)^{-1} \quad (1)$$

donde $[R,T]_L^{Ci}$ indica la rotación y la transformación de la cámara C_i con respecto al LIDAR. *W* indica la referencia del mundo.

3. Metodología

Como se mencionó anteriormente, la metodología presentada se estructura en 3 procesos principales. Primeramente es necesario ajustar las configuraciones de todos los sensores y postprocesar los datos. Esto con el fin de reducir el ruido y el error. La distorsión radial y tangencial se eliminan de las imágenes. El ruido se reduce en los datos LIDAR, para posteriormente submuestrear los datos. El paso siguiente es realizar la extracción de la textura, evaluando las zonas óptimas de las imágenes para reducir el error de proyección. Con las nubes de puntos texturizadas, se genera el mapa global digital de la escena urbana. Por último, las superficies se generan mediante un enfoque basado en *octrees*.

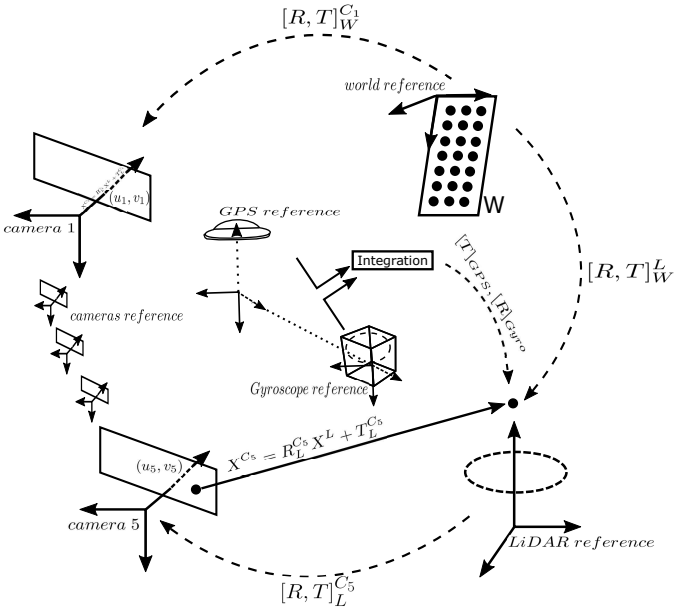

Figura 2: Marco de referencia del LIDAR *L* y el sistema multi-cámaras $C_1, \cdots, C_6$. $[R,\mathbf{T}]_W^{C_i}$ representa la transformación mundo-LIDAR y $[R,\mathbf{T}]_L^{C_i}$ la transformación LIDAR-cámara

3.1. Procesamiento cámara

De manera individual se extraen las imágenes de las 5 cámaras que componen la *Ladybug2*. Estos datos están en formato *raw* para facilitar el proceso de ajuste de distorsión esférica. En este proceso se utiliza una matriz de mapeo proporcionada por el fabricante del sensor[1]. Después, la imagen es rectificada, ya que el proceso anterior genera áreas *vacías* en la imagen (3b). Posteriormente se genera una sola imagen panorámica. Para ello, se extraen puntos de interés usando el algoritmo multi-escala de Harris Harris (1993). Para cada imagen $I(x,y)$ se genera una Piramide Gaussiana de Imágenes $I_P(x,y)$ con submuestreo a razón de $s = 2$ y un suavizado de $\sigma_P = 1{,}0$. Para cada nivel de la pirámide se extrajeron los puntos de interés. La matriz de Harris *P* en el nivel *L* y la posición (x,y) es el producto tensorial de los gradientes ∇:

$$H_L(x,y) = \nabla_{\sigma_d} P_L(x,y) \otimes \nabla_{\sigma d} P_L(x,y)^T * g_{\sigma_i}(x,y) \quad (2)$$

donde la escala de integración es $\sigma_i = 1{,}5$, la escala de derivación es $\sigma_d = 1{,}0$. Para encontrar los puntos de interés, primero se calculo una función que permita definir la "resistencia"de los puntos de interés, la cual esta dada por:

$$f_{hm}(x,y) = \frac{|H_L(x,y)|}{\mathbf{tr}H_L(x,y)} = \frac{\lambda_1\lambda_2}{\lambda_1+\lambda_2} \quad (3)$$

para los valores propios (λ_1, λ_2). Los puntos de interés están ubicados donde $f_{hm}(x,y)$ sea un máximo local en un vecindario

[1] https://www.ptgrey.com/tan/10621

3×3 y este por debajo del umbral $\mathbf{tr} = 10$. Después se repite esta evaluación pero ahora a nivel subpixélico para refinar la ubicación del punto de interés.

Para cada uno de estos puntos se calcula su orientación $\theta_{x,y}$, en donde el vector de orientación $[\cos\theta_{x,y}, \sin\theta_{x,y}] = \frac{u}{|u|}$ para $u_L(x,y) = \nabla_{\sigma_o} P_L(x,y)$. La escala de orientación $\sigma_o = 4{,}5$. Ya con los puntos de interés se procede a fusionar las 5 imágenes.

Este proceso tiene un costo computacional alto, es por ello que solo los puntos más representativos y mejor distribuidos espacialmente son utilizados. Para este trabajo, se implemento el algoritmo de Supresión no-máxima adaptativa descrito por Brown et al. (2005). El cual se basa en la ecuación 3, la orientación de los puntos y un vecindario de pixeles definido, donde solo los puntos de interés con un máximo en ese vecindario son conservados. La siguiente etapa es la fusión de las cinco imágenes, donde el objetivo es encontrar características geométricas a relacionar entre todas las imágenes. Para esto, primero se define un conjunto de puntos de interés usando el algoritmo de *aproximación de vecinos cercanos* propuesto por Kushilevitz et al. (2000).

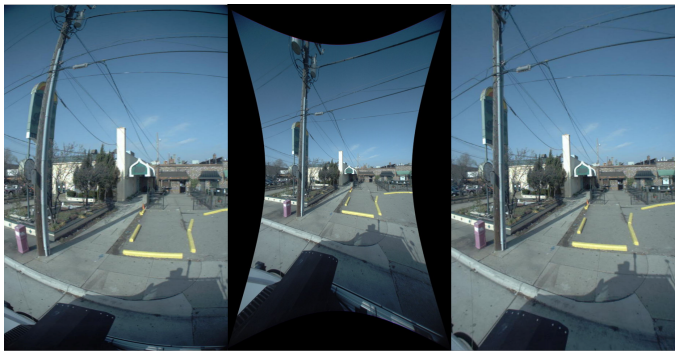

Figura 3: Transformación de las imágenes capturadas por la *Ladybug*. Estas imágenes se adquieren por separado para cada cámara del sensor, de esta manera el error debido a la distorsión es menor. La figura (a) muestra la imagen original, después utilizando una matriz de mapeo se elimina la distorsión en la imagen y la se ajusta (b-c).

Sea un punto $q \in \mathbb{R}$, con un error relativo ϵ y p' como el vecino más cercano de q. Dado un $\epsilon > 0$, el punto $p \in S$ es $(1+\epsilon)$-vecino más cercano de q si:

$$dist(p,q) \leq (1+\epsilon) dist(p',q) \tag{4}$$

entonces p está dentro del error relativo ϵ del vecino más cercano, en otros palabras, para $1 \leq k \leq n$, el enésimo $(1+\epsilon)$-vecino más cercano de q es un punto cuyo error relativo del verdadero enésimo vecino más cercano de q es ϵ.

Después se refina la correspondencia de puntos usando un procedimiento de rechazo basándonos en los datos estadísticos de cuantos falsos positivos existen en la concordancia de los puntos de interés entre las imágenes. Un segundo proceso de refinamiento se implementa, el algoritmo de *RANSAC* ajusta mas la correspondencia de los puntos basándose en ciertas restricciones geométricas y finalmente se realiza la fusión de las 5 imágenes para generar una imagen panorámica.

3.2. *Post-procesado datos del LIDAR*

Los datos del sensor láser son demasiados, cada nube tridimensional esta compuesta alrededor de $700k - 1000k$ de puntos, muchos de ellos pertenecientes al suelo. Por otro lado, las nubes de puntos tienen error de medición ya sea por factores internos o externos. Se hacen dos ajustes a los datos del LIDAR. Primero se reduce la cantidad de puntos haciendo un sub-muestreo. Después se suavizan ajustando su posición con respecto a la superficie a la que pertenecen.

3.2.1. *Sub-muestreo*

La idea central es buscar una ecualización en las direcciones de los puntos 3D con una resolución angular de referencia $\Delta\vartheta_s$. Por definición técnica se sabe que el campo de visión de un sensor láser esta definido por dos ángulos, Γ para la apertura de barrido y Ξ para la rotación alrededor de su propio eje. La resolución angular esta dada por ϕ y θ para las rotaciones alrededor de los ejes Z y X respectivamente. Para los datos de entrada al método se considera:

$$\Delta\vartheta_s = \frac{\text{máx}(\phi,\theta)}{p} \tag{5}$$

donde $0 < p \leqslant 1$ es un factor de ecualización. Mientras más cercano a 0 sea el valor de p la ecualización de la densidad de las direcciones del láser será más fina. Como los datos recogidos por el láser están distribuidos de forma uniforme, la misma cantidad de datos (c_n) pueden ser almacenados en cada registro dentro de una matriz Np, $c_n = round(1 + \frac{\Gamma}{\Delta\vartheta_s})$. Después un conjunto de columnas (J) son seleccionadas uniformemente para todos los datos según:

$$J = \left\{1 + round((\tfrac{N_p - 1}{s-1})(k-1))\right\}_{k=1,\ldots,c_n} \tag{6}$$

La densidad de dirección de las columnas en Np varia de acuerdo a la longitud del arco de barrido, que tiene su máximo en la columnas centrales. Una longitud normalizada del arco $0 \leqslant a_j \leqslant 1$ puede ser definida para la *enésima* columna j como:

$$a_j = \sin\left(\tfrac{\pi - \Gamma}{2} + (j-1)\Delta\gamma\right) \tag{7}$$

donde $\Gamma \leqslant \pi$. Por lo tanto, el numero $r_n(j)$ de puntos seleccionados para la *enésima* columna j es una función de a_j:

$$r_n(j) = \begin{cases} round(1 + (\frac{\Xi}{\Delta\vartheta_s} - 1)a_j) & \text{si } j \in J \\ 0 & \text{si } \textit{cualquier otro} \end{cases} \tag{8}$$

Entonces, el conjunto de puntos P_j seleccionados para la *enésima* columna j como:

$$P_j = \left\{1 + round((\tfrac{N_p - 1}{r_n(j)})(k-1))\right\}_{k=1,\ldots,r_n(j)} \tag{9}$$

Por último, se crea una máscara binaria M del tamaño de Np, en la cual se define a 1 los índices de los elementos obtenidos en 6 y 9. Esta matriz máscara, que representa los puntos seleccionados de N_p debe ser calculada solo una vez para cada tipo de sensor a determinadas revoluciones. La multiplicación *elemento-por-elemento* de la máscara M por la matriz Np producirá una matriz sub-muestreada Np_s donde los elementos vacíos no se tomaran en cuenta.

Por otro lado, cabe señalar que para fines prácticos de esta investigación no es relevante el impacto que los objetos dinámicos tienen en el desempeño de la metodología. Ya que la segmentación y extracción de objetos urbanos, como personas y vehículos en movimiento, requiere por si mismo todo un trabajo de investigación.

3.2.2. *Suavizado y eliminación de error*

Un modelo ruidoso esta dado por $y = \hat{y} + \varepsilon$, ε representa ruido Gaussiano con media igual a cero y varianza desconocida tal como lo define Garcia (2010), $\hat{y}$ necesita ser suavizada, por ejemplo con continuas derivadas en todo su dominio. El valor

de y esta en función de encontrar el mejor valor para $\hat{y}$. El algoritmo inicializa $\hat{y} = 0$ y calcula el $DCT(\hat{y})$ (Transformada de coseno discreta) de forma iterativa hasta que $\hat{y}$ converge, esto de acuerdo a:

$$\hat{y}_{k+1} = IDCTN(\Gamma^N \circ DCTN(W \circ (y - \hat{y}_k) + \hat{y}_k)) \quad (10)$$

donde $IDCTN$ es la transformada inversa discreta del coseno *n-dimensional*, $\circ$ expresa el producto de *Schur* y Γ es un tensor de rango N definido por $\Gamma^N = 1^N \oslash (1^N + s\Lambda^N \circ \Lambda^N)$, para 1^N como un tensor de rango N de unos y Λ^N definido por:

$$\Lambda^N_{i_1,\cdots,i_N} = \sum_{j=1}^{N} \left(-2 + 2\cos\tfrac{\pi(i_j-1)}{n_j}\right) \quad (11)$$

donde n_j expresa el tamaño de Λ^N a lo largo de la *j-dimensión*. La determinación automática de s requiere calcular el valor de la validación cruzada generalizada (GCV) de la siguiente forma:

$$GCV(s) = \frac{\|\hat{y} - y\|_F^2 / n}{(1 - \left\|\Gamma^N\right\|_1 / n)^2} \quad (12)$$

$\| \|_F$ expresa la norma de Frobenius, $\| \|_1$ es la primera norma y $n = \prod_{j=1}^{N}$ es el número de elementos en y.

Debido a que el proceso minimiza iterativamente los puntos 3D para encontrar un factor que los suavice por medio de la DCT, los datos tienden a distorsionarse en las orillas de la nube de puntos, especialmente cuando la cantidad de datos es pequeña.

Algoritmo 1 Extracción de textura.Cada una de las nubes de puntos son filtradas para obtener la textura, si no se encuentran en el rango de $5 - 25m$ no se toman en cuenta, ya que como se mencionó, distancias fuera de este rango contienen mayor error.

```
Entrada: nubes de puntos 3D, imágenes panorámicas
Salida: nube de puntos con textura
for all nubePuntos do
    load(imagenPanorámica)
    for i = 0 : size(nubePuntos) do
        [x, y] = proyeccion_3D_a_2D(nubePuntos[i])
        if y > 5  &&  y < 25 then
            textura = [textura;  imagenPanoramica(x, y)]
        else
            textura_no_valida
        end if
    end for
    nubeTexturizada = [nubePuntos textura]
end for
```

3.3. Texturizado

Cuando se construyen modelos 3D de escenas urbanas reales, la textura juega un papel importante. Esta textura definirá objetos y dará mayor percepción visual a la profundidad y perspectiva. A continuación se muestra el proceso de extracción de color de las imágenes y su posterior correlación con los datos del LIDAR y los demás sensores de la plataforma.

3.3.1. Texturizado nubes de puntos 3D

Como lo demuestra García-Moreno et al. (2016), la distancia de los objetos con respecto a la plataforma de digitalización es importante. Del mismo modo, la extracción de la textura está en relación de la distancia. Mientras que un objeto visto por la cámara está en dos dimensiones, los datos del LIDAR se encuentra en 3. Cuando se proyectan las puntos tridimensionales sobre el plano imagen se tiene un error. La precisión en la extracción de textura está en función de la distancia (profundidad), es decir, el correcto valor de textura de cierto punto 3D es más preciso mientras se encuentra dentro de un rango de distancia óptimo. Por lo que solo los puntos 3D del LIDAR que se encuentran en ese intervalo son tomados en cuenta para extraer su textura (ver Figura 4). Por lo tanto, cada punto 3D a representar es proyectado sobre la imagen panorámica y clasificado de acuerdo a la distancia con respecto a la cámara. De esta manera es posible evaluar los puntos que se encuentran en el rango óptimo(ver algoritmo 1).

La fortaleza de este algoritmo es que el tiempo de cómputo es lineal con el número de puntos 3D y píxeles, por lo que para sistemas en tiempo real es la solución más acertada. Otro punto a favor es la posibilidad de aplicar técnicas estadísticas para calcular los valores de los pixeles. Las cuales consisten en asignarle a cada punto 3D un vector donde se guardan los distintos valores de color que le asigna cada vista de la cámara, siempre y cuando la cámara vea al mismo objeto en diferentes adquisiciones. Una vez guardados todos los valores se les aplican la media o se interpola el valor para escoger el color apropiado.

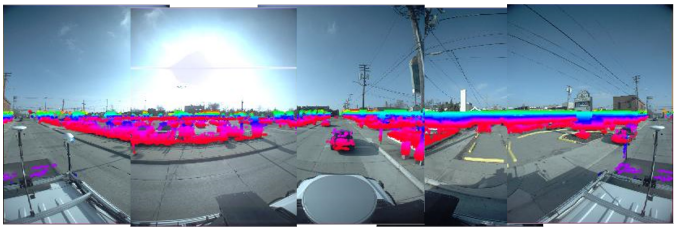

Figura 4: Solo los puntos tridimensionales que se encuentran en el rango de $5 - 25m$ son utilizados para extraer la textura. La figura también muestra los datos clasificados en diferentes colores según su altura. Se observa la importancia que tiene el hacer un *stitching* preciso para extraer la textura.

3.4. Generación de mallas

En este artículo se utiliza un enfoque para mallar una nube de puntos de forma rápida y eficiente. En un primer paso las nubes de puntos se segmentan en un árbol octal, más formalmente conocido como *Octree*. De modo que a cada clúster segmentado se le aplica el algoritmo de *Ball-Pivoting* para generar la superficie. La metodología implementada se denomina *Algoritmo Octree-Pivoting de fusión*.

Un elemento *octree* subdivide recursivamente un espacio 3D en ocho subspacios, llamados nodos. El algoritmo funciona basado en una estructura *octree*, que sirve como un índice espacial de los conjuntos de puntos subdivididos. La conexión de los conjuntos se representa por la relación de nodos del *octree*.

Una nube de puntos es segmentada en clusters de puntos coplanares. En el proceso de división, las nubes de puntos de entrada se dividen continuamente en ocho subespacios hasta que cada subespacio contiene sólo puntos coplanares. Utilizando el *octree* construido en el proceso de división todos los puntos coplanares son unidos recursivamente. Implementando el método propuesto por Wang and Tseng (2010) se define un nodo raíz que represente un espacio cúbico cuya dimensión es la extensión máxima de los puntos. A continuación, el nodo raíz se ramificará a ocho nodos hoja, representando los ocho subespacios divididos, todos de igual tamaño. Si el conjunto de datos no pasa la prueba coplanar (se evalúan las desviaciones de puntos con respecto a un plano teórico), el conjunto de datos se subdivide en ocho subespacios nuevamente. Cada nodo hoja desarrolla de manera recursiva ramas hasta que el conjunto de puntos en cada subespacio contiene sólo puntos coplanares. Recursivamente se definen dos nodos vecinos en el *octree* y se fusiona sus subespacios correspondientes para generar un plano combinado. Esto se hace en dos pasos: primero se define una tabla de índices espaciales que registra todos los nodos vecinos para cada nodo inicial, después se realiza la prueba coplanar y se fusiona nodos vecinos. La tabla de índices espaciales se puede establecer explorando las relaciones inherentes de cada nodo en un *octree*.

Cuando dos nodos vecinos contienen suficientes puntos para formar un plano tendrán prioridad para fusionarse. Una evaluación en tres etapas se realiza para encontrar la posibilidad de unir dos nodos vecinos que contengan puntos coplanares. Se comprueba el ángulo entre los vectores normales de cada plano. Si la diferencia entre ambos ángulos es amplia, los dos nodos vecinos no son apropiados para fusionarse. Se realiza una segunda evaluación para verificar si los dos planos pertenecen al mismo objeto. Pero a diferencia de Wang and Tseng (2010), que solo evalúa y define un umbral mínimo de distancia entre dos planos para poder fusionarse, en esta parte del algoritmo también se calcula y evalúa la curvatura principal de la superficie. Si esta curvatura tiene poca variación, entonces el proceso de fusión puede realizarse entre los dos planos. Esto evita que se fusionen planos similares que pertenezcan a dos objetos diferentes. Esta parte del proceso es importante, ya que a medida que más nodos se fusionan, la superficie aproximada tiene más precisión (ver algoritmo 2).

Después del proceso de división y fusión, se tienen clusters de objetos indexados (fachadas, coches, postes de luz, etc). Cada uno de ellos es mallado usando el algoritmo de *Ball-Pivoting*.

3.4.1. Método de formación de mallas

Sea un triangulo XYZ, con $x(resp.y, z)$ como el tamaño del lado opuesto a $X(resp.Y, Z)$. El circuncentro CC del triángulo XYZ tiene una coordenada baricéntrica de $(x^2(y^2 + z^2 - x^2),\ y^2(x^2 + z^2 - y^2),\ z^2(x^2 + y^2 - z^2))$, con circunradio:

$$r_c = \sqrt{\frac{x^2 \cdot y^2 \cdot z^2}{(x + y + z) \cdot (y + z - x) \cdot (x + y - z)}} \qquad (13)$$

Una esfera existe solo sí $r^2 - r_c^2 \geq 0$. Se define n como la normal al plano del triángulo, orientado de forma que tiene un producto escalar no negativo con cada una de las normales de los vértices $(v_1 - v_0) \otimes (v_2 - v_0)$. La idea principal del algoritmo es encontrar dos vértices que intersecten la esfera de radio r con centro en p.

La implementación del algoritmo de *Ball-Pivoting* esta diseñada en tres etapas, selección del triángulo semilla, teselado y eliminación de huecos.

Selección del triángulo semilla

Para cada *octree* definido, se elige un triángulo semilla usando un simple criterio. Se considera un triángulo, si la esfera más pequeña circunscrita en el está vacía y se encuentra sobre la superficie externa del objeto (*octree*). Por lo tanto, cuanto más plana la superficie sea localmente, el criterio usando tiende a ser más un mallado en 2D y se garantiza una mejor reconstrucción. Para hacer esto se selecciona un punto dentro del *octree* de forma aleatoria. Al buscar un punto vecino circundante, un borde e_i se define con esos dos puntos. A continuación, se define una esfera con centro en el punto medio del borde e_i. La esfera tiene radio de k veces la longitud de e_i. El valor de k se define de acuerdo a la distribución de los puntos dentro del *octree*. Una búsqueda de rango se realiza dentro de la esfera para buscar un triángulo con una esfera circunscrita vacía y asegurándose que el triángulo se encuentra en sobre la superficie. El valor de k es incrementado o decrementado si no se encuentran puntos dentro del área de búsqueda. Este proceso es repetido hasta que se encuentra un triángulo que satisface las dos condiciones (ver algoritmo 2).

Algoritmo 2 Algoritmo de mallado. La nube de puntos se segmenta en clusters utilizando un enfoque basado en *octree*. Un nodo raíz se define y después se ramifica a ocho nodos hoja. El algoritmo encuentra dos nodos vecinos en el *octree*, y fusiona sus subespacios generando un plano siempre y cuando contengan suficientes puntos. Se elige el triángulo semilla si la esfera más pequeña circunscrita en el está vacía y se encuentra sobre la superficie externa del objeto (*octree*). El radio de la esfera es incrementado o decrementado si no se encuentran puntos dentro del área de búsqueda. Este proceso es repetido hasta que se encuentra un triángulo que satisface las dos condiciones anteriores.

```
Entrada: nube de puntos (PC) mapa global
Salida: nube de puntos mallada

Lista-nodos = []
for all puntos(pc_i) ∈ PC(nodo raiz) do
    while existan nodos sin procesar do
        if esCoplanar(pc_i) then
            calcular parámtros del plano
            Lista-nodos.append(pc_i) definir nodos procesados e indexados
        else
            dividir nodo en 8 subnodos
        end if
    end while
end for
Lista-octree = []
Lista-node(LN) = sort(Lista-nodos)
while existan nodos sin procesar LN_i ∈ LN do
    if LN_i tiene puntos aislados then
        descartar LN_i definir nodo procesado LN_i
    end if
    while LN_i tiene nodos vecinos do
                                ▷ obtener el nodo vecino más cercano para LN_i (cbn)
        if evaluación 3-estados = TRUE then
            LN_i = fusionar nodos(LN_i, cbn)
        else
            definir nodo sin procesar LN_i
        end if
    end while
    Lista-octree.append(LN_i) definir nodo procesado e indexado LN_i
end while
r→ radio
N_r(p) → vecindario con radio r con centro en p
for all LN_i ∈ Lista-octree do
    for puntos(p) ∈ LN_i do
                    ▷ Buscar todos los puntos dentro de la esfera del vecindario N_2r(p)
        for (q, s) ∈ N_2r(p) do
            if p, q, s esfera vacía & (q − p) ⊗ (s − p) > 0 then
                semillaT = (p, q, s) definir triángulo semilla
            end if
        end for
    end for
end for
```

Teselado

Para cada punto dentro de la esfera se define un posible borde p_e para generar un triángulo. Para tener en cuenta los puntos a ser mallados, se usa solamente el borde p_e que se encuentra en el plano del triángulo de enfrente, en la dirección del crecimiento de la malla y con un radio k. Dado que la región de búsqueda se encuentra en el mismo plano que el triángulo frontal, el método de inspección contribuye a definir los puntos candidatos en la región donde se espera que la superficie aumente. El proceso excluye cualquier punto que se encuentra fuera de la región de búsqueda (*outliers*) o que podrían generar triángulos muy delgados. Para seleccionar un punto de referencia en el área de inspección cuando se tiene más de una opción, se implementó una variación del algoritmo propuesto por Di Angelo et al. (2011). Primero se calcula el centro de masas de todos los puntos dentro de la región de búsqueda y la esfera que los circunscribe. Entonces, si la esfera no está vacía, el procedimiento se repite con nuevos puntos dentro de una esfera nueva. El proceso termina cuando la esfera pasa por el punto de referencia elegido y los extremos de p_e están vacíos. Para verificar y acelerar el proceso de teselado del triángulo definido como el borde frontal, se realiza una validación adicional. Si solo se encuentra un punto dentro de la región de búsqueda, se asume que ese punto identifica a un triángulo candidato con p_e sin verificar si existe o no una esfera mínima circunscrita vacía. Por último, si no se encuentra ningún punto dentro de la región de búsqueda, el borde p_e se elimina de la cola frontal. Ahora, con todos los triángulos posibles, se realiza una prueba de pertinencia. Esta prueba se lleva a cabo necesariamente para obtener una superficie teselada lo más aproximada a la realidad. Dado que cada posible triángulo está formado por un borde p_e y dos bordes adicionales f_{e1}, f_{e2}), el algoritmo comprueba si

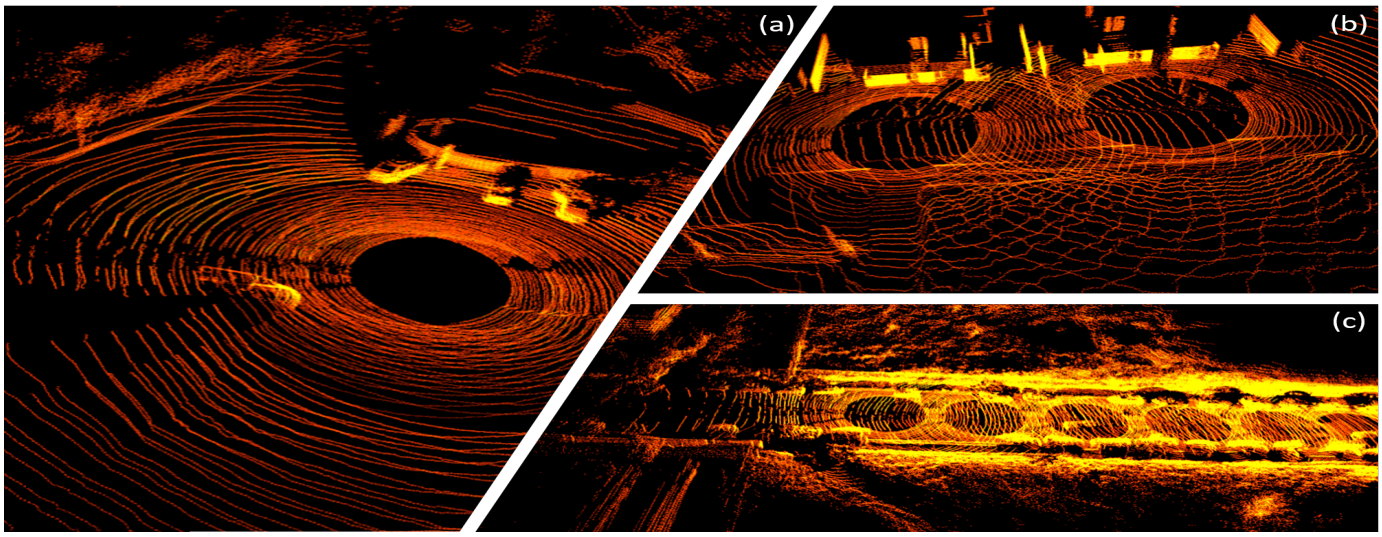

Figura 5: Se muestra la fusión de varias nubes puntos de un recorrido. Se observa el punto ciego que tiene el sensor LIDAR, un diámetro de 2 metros. Existe demasiada información que pertenece al suelo, derivado de ello, es necesario muestrear los datos y disminuir el error en los datos.

estos dos bordes son realmente nuevos o ya pertenecen a otros triángulos. Si pertenecen a un triángulo que ya existe, se debe comprobar la consistencia de la orientación del nuevo triángulo con el que comparte algún borde. De lo contrario, el nuevo triángulo y el borde p_e se eliminan de la cola delantera.

Eliminación de huecos

Un hueco aparece generalmente cuando la nube de puntos, en nuestro caso la estructura *octree*, no tiene suficientes puntos para generar una malla continua. Para eliminar esta discontinuidad, se rellena *cosiendo* los dos lados de la abertura con nuevos triángulos. En este proceso, no se consideran todos los bordes, solo aquellos cercanos entre sí, esto con la idea de no alterar la forma de la superficie. El proceso comienza definiendo una esfera cuyo centro se encuentra en un punto que pertenezca a un borde en el contorno del hueco. Un radio k se define con la longitud de extremo a extremo del hueco. Si la esfera contiene un punto que dentro del otro lado del hueco, se realiza el teselado para todos esos puntos. El proceso se repite moviendo la esfera en dirección del mallado buscando nuevos puntos. Observe que si un hueco es grande, el método puede fallar ya que la esfera puede contener puntos de diferentes huecos, es decir, que se trataría de eliminar diferentes huecos que pertenezcan a diferentes superficies.

Uno de los mayores problemas observados en la reconstrucción urbana tridimensional es la eliminación de huecos con pequeñas superficies dentro de ellas (islas). Una isla está limitada por los bordes exteriores de una frontera. Después de que una isla es detectada, se identifica la brecha que la contiene de la siguiente manera:

$$d_{I,H_j} = \frac{\sum_{i=1}^{n} d(\hat{v}_i, v'_i)}{n} \tag{14}$$

donde v'_i es límite más cercano desde el hueco H_j hasta el punto $\hat{v}_i$, y $d(\hat{v}_i, v'_i)$ es la distancia entre los puntos $\hat{v}_i$ y v'_i. Un hueco H con distancia $min(d_{I,H_j})$ contiene una isla I.

Para abordar este problema (el mallado de huecos), nuevos puntos son considerados antes de añadir nuevos elementos al hueco. Si una esfera cuyo centro se encuentra sobre un posible punto y no contiene ningún vértice del objeto, el punto se añade al conjunto de los nuevos datos. Después de crear un nuevo punto en el hueco, las mallas se construyen a partir de los datos agregados, es decir, a partir de los vértices en los límites del hueco y la isla. Si las distancias euclidianas entre los nuevos puntos agregados y los vértices de I son mayores a la longitud promedio del hueco, no será posible eliminar ese hueco. Este método es robusto para eliminar huecos, incluso huecos con islas.

4. Resultados

Un proceso importante en la generación de mapas globales es la fusión de los datos tridimensionales. Cientos de nubes de puntos son llevadas a un mismo marco de referencia para obtener un solo mapa. La Figura 5 muestra la fusión de varias nubes puntos de un recorrido. Se observa el punto ciego que tiene el sensor LIDAR, un diámetro de 2 metros alrededor de el. Existe demasiada información que pertenece al suelo, lo que genera nubes de puntos con demasiados datos. Derivado de ello, es necesario muestrear los datos y disminuir el error en los datos. La correlación de los datos tridimensionales es eficiente. Los tonos amarillos indican mayor concentración de puntos por área. Un tema por considerar es que debido a la configuración del sensor LIDAR, no siempre es posible escanear por completo los edificios. En trabajos futuros se diseñará una estructura que permita invertir el sensor para permitir un alcance de altura en su reconstrucción.

Ahora bien, con el fin de evaluar, validar y darle trazabilidad a nuestra metodología, se utilizó una base de datos libre, distribuida por Pandey et al. (2011). Esta base de datos fue adquirida con una plataforma terrestre de adquisición similar a la nuestra. La plataforma está compuesta por un Applanix POS LV profesional y una unidad de medición inercial Xsens MTI-G para el consumidor, un escáner Velodyne 3D-lidar, dos Lars Riegl orientados hacia adelante y un sistema de cámara omnidireccional Point Grey Ladybug3. Los datos se recopilaron en el campus de Ford Research y en el centro de Dearborn, Michigan. La trayectoria de la trayectoria del vehículo en estos conjuntos de datos contiene varios cierres de bucles grandes y de pequeña escala, que son útiles para probar nuestro propio algoritmo. El tamaño del conjunto de datos es 100 GB.

Los experimentos se realizaron en diferentes entornos urbanos que contienen diferentes geometrías, tales como postes de luz, coches, fachadas de edificios y la maleza. Estos objetos

Figura 6: Diferentes vistas dentro de un mapa global texturizado y mallado. La discontinuidad del mallado se debe a la velocidad de adquisición de la plataforma.

se encuentran en diferentes capas en la dirección de profundidad de la imagen, lo que convierte al problema de extracción de textura en algo complejo, ver Figura 7.

En la primer secuencia de imágenes de la Figura 7(a) se puede observar una camioneta tipo *Van* en movimiento que pasa junto a la plataforma de reconstrucción. Es importante hacer notar como la proyección de los puntos 3D sobre la imagen en la sección que corresponde a la camioneta está desfasada, esto se debe propiamente a la dinámica de la escena. Cuando la plataforma está en movimiento y además los objetos de las escenas también lo están, existe un pequeño error en la correlación de los datos. La segunda parte 7(b), muestra una escena con edificios y calles. En primer lugar, se puede observar cómo los edificios que están en el primer plano de la imagen no son escaneados en su totalidad, esto debido a la configuración mecánica del LIDAR, pero conforme la plataforma se aleja es posible adquirir los datos faltantes (esto por la perspectiva del láser). La coloración de los puntos proyectados en las imágenes corresponde a la distancia de los puntos. También se observa que los puntos que corresponde al suelo no son usados, permitiéndonos trabajar con menos datos y dedicar mayor poder de cómputo a las estructuras principales en la escena.

En relación al proceso de generación de superficies, se puede observar en la Figura 6 diferentes vistas de un mapa global texturizado y mallado. La discontinuidad del mallado se debe a la velocidad de adquisición de la plataforma y a que el sistema es dinámico, es decir, la plataforma está en movimiento. Se aprecia el detalle de los objetos que conforman las escenas. El algoritmo es completamente funcional en entornos estáticos.

El requerimiento de tener imágenes de alta calidad, así como nubes de puntos robustas nos permite tener una reconstrucción y texturización con alta precisión. Por otro lado, después de filtrar y post-procesar todos los datos se generan nubes de puntos livianas. En la Figura 8 se puede observar una reconstrucción ya texturizada de un recorrido alrededor de 3 calles. Nuestras pruebas se realizaron en el área urbana cerca del centro de investigación CICATA donde se desarrolló el trabajo, en Querétaro, México. Hemos elegido este lugar porque contiene una variedad de escenas urbanas adecuadas, zonas habitacionales y edificios, estacionamientos, centros comerciales, áreas sin edificios, avenidas y calles con poca afluencia vehicular. Esto nos permitió generar una base de datos adecuada con una variedad de escenas variada. Nos desplazamos sobre una trayectoria que cerraba ciclos que posteriormente nos ayudaría para su análisis. Las escenas digitalizadas contienen infraestructura urbana, objetos dinámicos como personas y vehículos. Una desventaja de este conjunto de datos es que, en la ciudad, los calles son muy estrechas y dada la configuración de la plataforma de adquisición se tienen varias oclusiones y no se completa la reconstrucción de edificios.

El archivo final que almacena toda la información no supera los 1,5 Mb de tamaño y está formado por casi dos millones de puntos. Este archivo aun cuenta con los puntos pertenecientes al suelo. Cuando se hace el filtrado de los puntos del suelo el archivo final está formado por aproximadamente 1 millón de puntos y un tamaño de 0,85Mb. Se pierden algunos detalles cercanos o pertenecientes al suelo, pero se conservan los objetos principales en cada escena.

Es importante tener en cuenta que la plataforma de reconstrucción generalmente trabaja entre 20 y 30 minutos consecutivamente. Durante este tiempo, los sensores son susceptibles a las vibraciones que provienen de diferentes fuentes. Por ejemplo, la falta de estabilidad de la estructura en la que se montan los sensores y las vibraciones por irregularidades en el suelo pueden inducir grandes errores en los ángulos medidos. Debido a ello, la plataforma debe ser calibrada antes de usarse. Por otro lado, las vibraciones que se producen durante el proceso de adquisición y que son percibidas por el giroscopio, serán compensadas al momento de generar un mapa global. En García-Moreno et al. (2014) el trabajo previo se abordo abordado el análisis de la incertidumbre combinada y propagación del error debido a factores externos del sistema, por ejemplo las vibraciones.

4.1. Evaluación cuantitativa

Nuestros resultados han demostrado visualmente la efectividad de nuestra metodología para procesar nubes de puntos a gran escala. Además, se proporcionan algunas métricas cuantitativas para establecer su robustez a diferentes niveles de ruido. Estas métricas son el factor de cobertura y la raíz del error cuadrático medio (RSME por sus siglas en inglés). Según Yi et al. (2017), la métrica de cobertura mide la integridad de los resultados del modelado contra el modelo teórico, este valor representa el porcentaje de los puntos de *inliners* asociados con la reconstrucción de las estructuras. Dada la nube de puntos de entrada ρ y su modelo reconstruido M, la métrica de cobertura se define de la siguiente manera:

$$coverage = \frac{|\{p|\forall p \,\epsilon\, \rho, s.t. \|p - M\| < \epsilon\}|}{|\rho|} \tag{15}$$

donde $\|p - M\|$ es la distancia euclidiana entre p y M; $|\cdot|$ es la cardinalidad del conjunto; ϵ es un umbral pre-establecido que se fija comúnmente en 0,012.

La RSME mide la exactitud geométrica de los resultados reconstruidos con respecto a las nubes de puntos de entrada. Supongamos que el modelo reconstruido M comprende n planos primitivos, que se generan extruyendo segmentos lineales dentro de todos los bloques, $M = \varepsilon_{i}{}_{i=1}^{n}$. Para cada sección ε_i, el subconjunto de puntos asociado es denotado por ρ_i, en el que cada punto está más cerca de ε_i que cualquier otro plano. La métrica RSME se define de la siguiente manera:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} \sum_{\forall p \epsilon \rho_i} \|p - \varepsilon_i\|^2}{\sum_{i=1}^{n} |\rho_i|}} \tag{16}$$

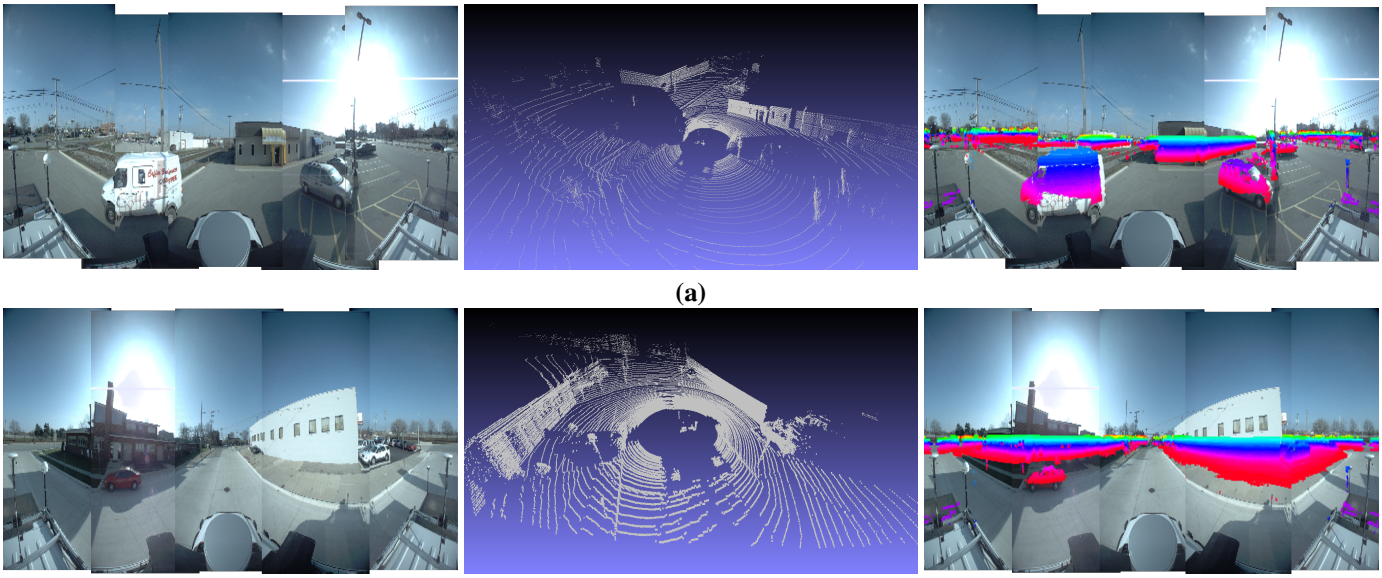

Figura 7: Diferentes entornos urbanos son escaneados. Edificios, personas, fachadas de casas, automóviles estacionados y en movimiento, objetos urbanos como postes, señales de tránsito, etc. La propuesta de esta investigación es obtener un algoritmo que pueda fusionar la información de todos los sensores y además generar mapas tridimensionales texturizados en ambientes no estructurados. La columna de la izquierda muestra la imagen panorámica, la columna central la misma escena adquirida por el LIDAR y la columna de la derecha muestra los puntos 3D proyectados en la imagen panorámica clasificados por color de acuerdo a su distancia.

donde $\|p - \varepsilon_i\|$ es la distancia euclidiana entre p y ε_i; $|\rho_i|$ es el número de puntos dentro de ρ_i.

Para fines de evaluación, nuestra metodología se compara con los trabajos propuestos por Yi et al. (2017); Pandey et al. (2014). Pandey et al. (2014) presenta un conjunto de algoritmos para la correlación de datos obtenidos por sensores láser, cámaras y sistemas inerciales. Los autores maximizan el uso de los datos de todos los sensores. La meta principal alcanzada es la texturización de las nubes 3D. Yi et al. (2017) presentan una metodología que permite aproximar las nubes de puntos a volúmenes estructurales predefinidos. Los cubos y esferoides se utilizan para reconstruir los ambientes urbanos. Posteriormente, estas primitivas geométricas son post-procesadas para ajustar la calidad visual de la reconstrucción. El objetivo principal alcanzado es una aproximación realista de la digitalización urbana.

Se procesan nubes de puntos formadas aproximadamente por $100k$ puntos y las imágenes panorámicas correspondientes. La Figura 9 muestra una comparación de nuestra metodología. La escena muestra un camión estacionado, un cartel publicitario e infraestructura vial. Se observa que nuestra metodología tiene una mayor definición del entorno digitalizado debido a la menor propagación de errores. Aunque nuestro trabajo tiene una cobertura más baja en comparación con los otros, es el único que genera las superficies directamente a partir de los datos tridimensionales.

La Tabla 1 muestra los resultados de RMSE y cobertura al

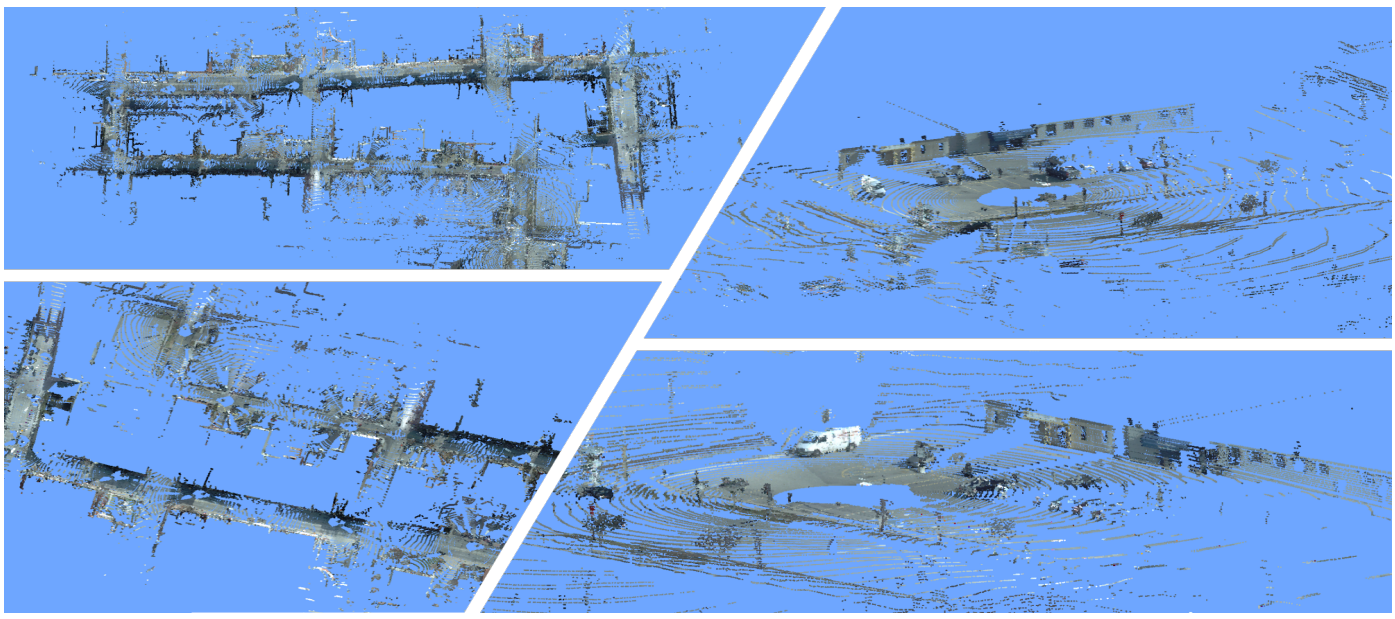

Figura 8: Construcción de un mapa global (*visión profunda*) texturizado. Se digitalizó un recorrido a través de 3 calle.

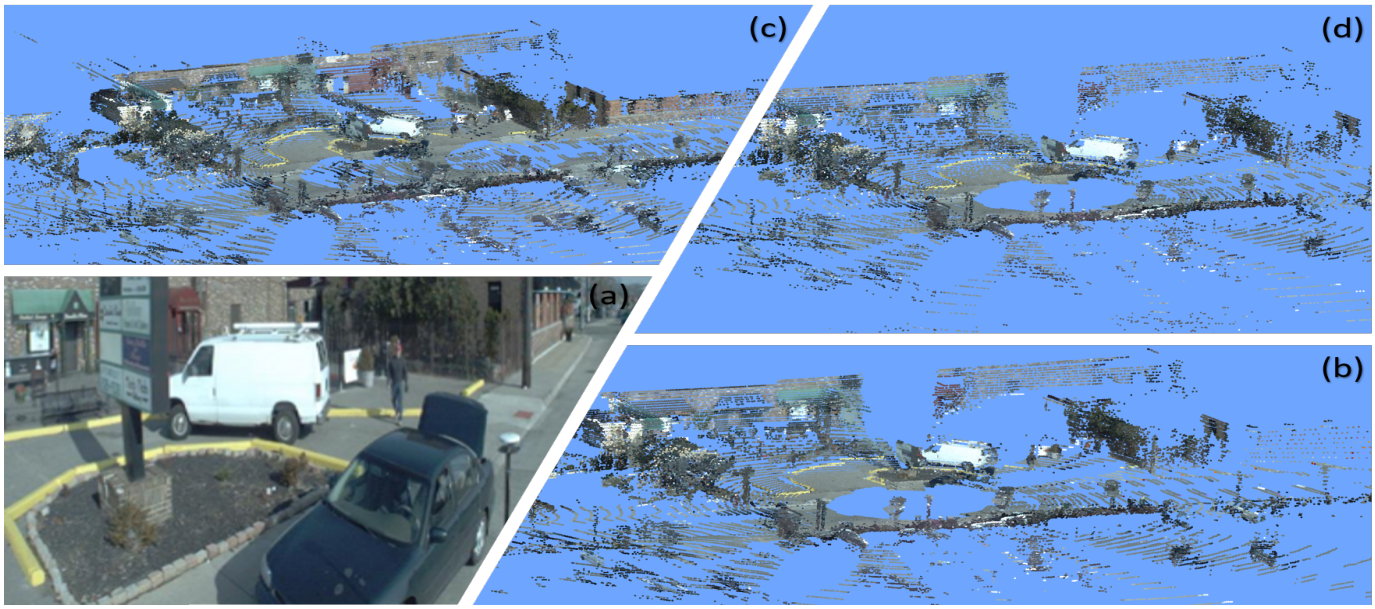

Figura 9: Se muestran los resultados de correlación de datos cámara-LIDAR implementando 3 metodologías. Estos datos son usados para reconstruir tridimensionalmente ambientes urbanos. Las metodologías que se comparan con el trabajo aquí presentado (a) se describen en Pandey et al. (2014) (b) y Yi et al. (2017) (c). La escena muestra un camión estacionado, un cartel publicitario e infraestructura vial. Se observa que nuestra metodología tiene un error más bajo (RSME) y una mayor definición del entorno digitalizado. También tiene un tiempo de computación corto. Como se puede ver en la Tabla 1.

procesar las nubes de puntos correspondientes a las figuras 7 y 9. Se muestrearon 5 nubes de puntos con sus respectivas imágenes. Se observa que nuestra metodología tiene un error menor (RMSE), que genera mayor claridad en los resultados obtenidos. Aunque es cierto que nuestro factor de cobertura no es tan amplio, ya que se han establecido distancias mínimas y máximas para extraer los datos de textura. Por otro lado, se observa que nuestra metodología requiere menos tiempo para procesar los datos en comparación con la presentada por Pandey et al. (2014), 75 segundos en comparación con 144s. La propuesta presentada por Yi et al. (2017) es la más rápida pero no genera las superficies directamente desde las nubes de puntos, aproximan las formas a primitivas geométricas. Un punto a tener en cuenta es que gracias a los postprocesos que se implementaron en este trabajo, nuestra metodología requiere menos datos para generar una reconstrucción.

Tabla 1: Comparación cuantitativa entre diferentes metodologías. Se muestra la precisión de la reconstrucción, el factor de cobertura y el tiempo de computo para procesar las nubes de puntos. Se procesaron cinco nubes de puntos que constan aproximadamente de 100k puntos cada una y sus respectivas imágenes. Todas ellas parte del recorrido mostrado en la Figura 8.

	Métricas		
Metodología	RMSE	Cobertura %	Tiempo Comp. (s)
Metodología propuesta	0.85	88	≈ 75
Pandey et al. (2014)	0.86	98	≈ 157
Yi et al. (2017)	1.02	95	≈ 63

Estos resultados muestran extensas escenas urbanas tridimensionales con: exactitud deseada, alta precisión y buen aspecto visual. Además de requerir un tiempo de cálculo más corto, obteniendo buenos resultados.

5. Conclusiones

Utilizando los métodos que aquí se presentan para la reconstrucción geométrica y fotorrealista, se construyen modelos urbanos 3D de gran calidad. Los resultados cumplieron con los objetivos: generar modelos globales texturizados conservando la geometría de las escenas escaneadas, utilizar la información de incertidumbre y sensibilidad evaluada en trabajos anteriores y conseguir un buen aspecto visual.

Nuestro algoritmo de texturización es bastante rápido, toma alrededor de 14 *min* digitalizar un recorrido alrededor de 3 calles, y nos permite tener una navegación continua de toda la escena escaneada. A pesar de las limitaciones de la señal GPS en las zonas urbanas, la exactitud de los modelos reunió los requisitos esperados para obtener la información fotogramétrica. Se alcanzo una precisión de aproximadamente 1,56 *cm*, ya que al procesar los puntos tridimensionales únicamente en el rango de distancia ideal (según los resultados presentados en García-Moreno et al. (2016)), se conserva la precisión obtenida con el método de calibración presentado por Atanacio-Jiménez et al. (2011).

Sin embargo, hay dos factores que tienen gran impacto en la precisión y exactitud en la generación de nuestros modelos 3D. Uno de ellos es el resultado obtenido del proceso automatizado, que afecta al resultado final ya que no se tiene un control visual sobre lo que ocurre en ellos, es decir, no se controlan las excepciones. El otro es el proceso de preparación para la extracción la textura. Ya que no se utiliza toda la imagen panorámica para obtener la textura y los puntos tridimensionales tampoco se utilizan todos, muchos objetos quedan incompletos en su texturización, es decir, en la continuidad de la misma. Este problema se presenta en los objetos que están más cerca de la plataforma de reconstrucción.

Por otro lado, el valor del radio de la esfera usada para encontrar nuevos vértices para mallar afecta la calidad y el tiempo del teselado. El valor del radio de búsqueda se especifica como una fracción de la longitud del borde frontal: un radio de búsqueda igual a 1 indica que tiene la misma longitud que el borde frontal. Un radio de búsqueda bajo mantiene la región de búsqueda cerca del borde frontal, por lo que el método queda corto en el mallado de áreas submuestreadas. Un radio de búsqueda elevado, por otro lado, reduce la tasa de teselación y

a veces genera defectividad.

Modelos urbanos tridimensionales son exitosamente generados por nuestra plataforma terrestre de reconstrucción, utilizando un proceso completamente automático. Para cada uno de los sensores se analizó su sensibilidad e incertidumbre para trabajar con datos óptimos y hacer las correcciones adecuadas, obteniendo así un modelo 3D con una precisión alta. Nuestros modelos urbanos son visualmente correctos y tienen una exactitud adecuada, las zonas urbanas escaneadas están conformadas por casas, edificios, automóviles, señales urbanas, personas, etc. Y los modelos obtenidos muestran con gran detalle todas las características de los objetos. Cabe resaltar que nuestra plataforma es terrestre, y que tanto el ambiente como la plataforma están en movimiento, aumentando la complejidad en la extracción de textura y posterior fusión de datos.

Estos resultados muestran escenas urbanas 3D reales, con una exactitud deseada, precisión alta y buena apariencia visual.

En trabajos futuros se tendrán en cuenta las siguientes áreas de oportunidad: (1) aumentar el proceso de automatización, (2) ajustar la perspectiva y los campos de visión de los sensores. Esto con la idea de obtener menos falsos positivos y que los objetos con gran altura puedan ser escaneados en su totalidad, (3) optimizar el modelo de proyección y extracción de la textura, (4) procesar las oclusiones generadas durante el proceso de digitalización, (5) desarrollar algoritmos de segmentación de nubes de puntos 3D, para eliminar objetos no deseados en los mapas, por ejemplo personas en movimiento.

Referencias

Bernard O Abayowa, Alper Yilmaz, and Russell C Hardie. Automatic registration of optical aerial imagery to a lidar point cloud for generation of city models. *ISPRS Journal of Photogrammetry and Remote Sensing*, 106:68–81, 2015.

Gerardo Atanacio-Jiménez, José-Joel González-Barbosa, Juan B Hurtado-Ramos, Francisco J Ornelas-Rodríguez, Hugo Jiménez-Hernández, Teresa García-Ramirez, and Ricardo González-Barbosa. Lidar velodyne hdl-64e calibration using pattern planes. *International Journal on Advanced Robotics Systems*, 8(5):70–82, 2011.

Matthew Brown, Richard Szeliski, and Simon Winder. Multi-image matching using multi-scale oriented patches. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 510–517. IEEE, 2005.

Jonathan C Carr, Richard K Beatson, Jon B Cherrie, Tim J Mitchell, W Richard Fright, Bruce C McCallum, and Tim R Evans. Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 67–76. ACM, 2001.

Ke Chen, Weisheng Lu, Fan Xue, Pingbo Tang, and Ling Hin Li. Automatic building information model reconstruction in high-density urban areas: Augmenting multi-source data with architectural knowledge. *Automation in Construction*, 93:22–34, 2018.

Tamal K Dey and Samrat Goswami. Provable surface reconstruction from noisy samples. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 330–339. ACM, 2004.

Luca Di Angelo, Paolo Di Stefano, and Luigi Giaccari. A new mesh-growing algorithm for fast surface reconstruction. *Computer-Aided Design*, 43(6): 639–650, 2011.

Julie Digne. An analysis and implementation of a parallel ball pivoting algorithm. *Image Processing On Line*, 4:149–168, 2014.

Damien Garcia. Robust smoothing of gridded data in one and higher dimensions with missing values. *Computational statistics & data analysis*, 54(4):1167–1178, 2010.

Angel-Iván García-Moreno, José-Joel Gonzalez-Barbosa, Francisco-Javier Ornelas-Rodriguez, Juan B Hurtado-Ramos, and Marco-Neri Primo-Fuentes. Lidar and panoramic camera extrinsic calibration approach using a pattern plane. In *Pattern Recognition*. Springer, 2013.

Angel-Iván García-Moreno, Denis-Eduardo Hernandez-García, José-Joel Gonzalez-Barbosa, Alfonso Ramírez-Pedraza, Juan B Hurtado-Ramos, and Francisco-Javier Ornelas-Rodriguez. Error propagation and uncertainty analysis between 3d laser scanner and camera. *Robotics and Autonomous Systems*, 62(6):782–793, 2014.

Angel-Iván García-Moreno, José-Joel González-Barbosa, Alfonso Ramírez-Pedraza, Juan B Hurtado-Ramos, and Francisco-Javier Ornelas-Rodriguez. Accurate evaluation of sensitivity for calibration between a lidar and a panoramic camera used for remote sensing. *Journal of Applied Remote Sensing*, 10(2):024002–024002, 2016.

Jianwei Guo, Dong-Ming Yan, Li Chen, Xiaopeng Zhang, Oliver Deussen, and Peter Wonka. Tetrahedral meshing via maximal poisson-disk sampling. *Computer Aided Geometric Design*, 43:186–199, 2016.

Rostam Affendi Hamzah, A Fauzan Kadmin, M Saad Hamid, S Fakhar A Ghani, and Haidi Ibrahim. Improvement of stereo matching algorithm for 3d surface reconstruction. *Signal Processing: Image Communication*, 65:165–172, 2018.

Chris Harris. Geometry from visual motion. In *Active vision*, pages 263–284. MIT Press, 1993.

C. Hatger and C. Brenner. Extraction of road geometry parameters from laser scanning and existing databases. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 34(3/W13):225–230, 2003.

Dorota Iwaszczuk and Uwe Stilla. Camera pose refinement by matching uncertain 3d building models with thermal infrared image sequences for high quality texture extraction. *ISPRS Journal of Photogrammetry and Remote Sensing*, 132:33–47, 2017.

Hansung Kim and Adrian Hilton. Block world reconstruction from spherical stereo image pairs. *Computer Vision and Image Understanding*, 139:104–121, 2015.

Eyal Kushilevitz, Rafail Ostrovsky, and Yuval Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM Journal on Computing*, 30(2):457–474, 2000.

Maxime Lhuillier. Surface reconstruction from a sparse point cloud by enforcing visibility consistency and topology constraints. *Computer Vision and Image Understanding*, 175:52–71, 2018.

Lingyun Liu and Ioannis Stamos. A systematic approach for 2d-image to 3d-range registration in urban environments. *Computer Vision and Image Understanding*, 116(1):25–37, 2012.

Jules Morel, Alexandra Bac, and Cédric Véga. Surface reconstruction of incomplete datasets: A novel poisson surface approach based on csrbf. *Computers & Graphics*, 74:44–55, 2018.

Gaurav Pandey, James R McBride, and Ryan M Eustice. Ford campus vision and lidar data set. *The International Journal of Robotics Research*, 30(13): 1543–1552, 2011.

Gaurav Pandey, James R McBride, Silvio Savarese, and Ryan M Eustice. Automatic extrinsic calibration of vision and lidar by maximizing mutual information. *Journal of Field Robotics*, 2014.

Yun Shi, Shunping Ji, Xiaowei Shao, Peng Yang, Wenbin Wu, Zhongchao Shi, and Ryosuke Shibasaki. Fusion of a panoramic camera and 2d laser scanner data for constrained bundle adjustment in gps-denied environments. *Image and Vision Computing*, 40:28–37, 2015.

Miao Wang and Yi-Hsing Tseng. Automatic segmentation of lidar data into coplanar point clusters using an octree-based split-and-merge algorithm. *Photogrammetric Engineering & Remote Sensing*, 76(4):407–420, 2010.

Ruisheng Wang, Jeff Bach, Jane Macfarlane, and Frank P Ferrie. A new upsampling method for mobile lidar data. In *Applications of Computer Vision (WACV), 2012 IEEE Workshop on*, pages 17–24. IEEE, 2012.

Bin Wu, Bailang Yu, Qiusheng Wu, Shenjun Yao, Feng Zhao, Weiqing Mao, and Jianping Wu. A graph-based approach for 3d building model reconstruction from airborne lidar point clouds. *Remote Sensing*, 9(1):92, 2017.

Lin Yang, Yehua Sheng, and Bo Wang. 3d reconstruction of building facade with fused data of terrestrial lidar data and optical image. *Optik-International Journal for Light and Electron Optics*, 127(4):2165–2168, 2016.

Michael Ying Yang, Yanpeng Cao, and John McDonald. Fusion of camera images and laser scans for wide baseline 3d scene alignment in urban environments. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(6): S52–S61, 2011.

Cheng Yi, Yuan Zhang, Qiaoyun Wu, Yabin Xu, Oussama Remil, Mingqiang Wei, and Jun Wang. Urban building reconstruction from raw lidar point data. *Computer-Aided Design*, 93:1–14, 2017.

Fanyang Zeng and Ruofei Zhong. The algorithm to generate color point-cloud with the registration between panoramic image and laser point-cloud. In *IOP Conference Series: Earth and Environmental Science*, volume 17, page 012160. IOP Publishing, 2014.

SM Iman Zolanvari, Debra F Laefer, and Atteyeh S Natanzi. Three-dimensional building façade segmentation and opening area detection from point clouds. *ISPRS journal of photogrammetry and remote sensing*, 143:134–149, 2018.