Mapeo denso en tiempo real sobre sistemas SLAM basados en visión estéreo

15 de marzo de 2018

Ariel Roberto D'Alessandro
ariel.dalessandro@gmail.com

Director
Pire, Taihú
pire@cifasis-conicet.gov.ar

Co-Director
Baravalle, Rodrigo
baravalle@cifasis-conicet.gov.ar

Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Universidad de Rosario
Av. Pellegrini 250 - Planta Baja - (S2000BTP)
Rosario - Rep. Argentina.
Teléfono: +54 - 341 - 4802649/52 - interno 112
http://www.fceia.unr.edu.ar

Reconstrucción densa en tiempo real sobre sistemas SLAM basados en visión estéreo

Para poder navegar de manera segura en un entorno desconocido, un robot autónomo móvil debe poder construir una representación del ambiente (mapeo) en el cual se mueve, al mismo tiempo que estima su posición (localización). Este problema es conocido en la robótica móvil como SLAM por el acrónimo en inglés de *Simultaneous Localization and Mapping*.

Un mayor nivel de detalle del entorno (mapa) resultará en una posible mejora en los algoritmos de navegación del robot, permitiéndole esquivar obstáculos con una precisión mayor, obteniendo trayectorias más seguras y óptimas.

En este trabajo se propone un sistema de SLAM denso basado en visión estéreo que resulta de la inclusión de un módulo de densificación al sistema de SLAM S-PTAM (*Stereo Parallel Tracking and Mapping*) del estado del arte. El sistema de SLAM resultante es capaz de generar un mapa denso del entorno al mismo tiempo que estima la localización de la cámara. Los experimentos muestran que el método realiza una densificación local precisa del mapa en tiempo real.

El sistema se implementó utilizando el framework ROS (*Robot Operating* System). El código ha sido liberado bajo licencia GPLv3, con el objetivo de facilitar el uso y modificación del sistema por parte de la comunidad robótica.

Agradecimientos

Quiero agradecer...

en primer lugar a mis directores, Taihú Pire y Rodrigo Baravalle, por la paciencia y el tiempo que dedicaron a lo largo de esta tesina.

a Javier Civera, por sus sugerencias y colaboración en el presente trabajo.

a los jurados, Pablo Pilotti y Juan Carlos Gómez, por la rápida evaluación.

a todos los profesores que forman parte de la LCC, por las ganas y gran dedicación que ponen en la carrera.

a la FCEIA y UNR en general, por haber sido y ser un hermoso espacio para estudiar y desarrollarse.

con sinceridad, al pueblo argentino por defender y sostener la educación pública, siempre!

a los/as amigos/as y compañeros/as que hice en este extenso pasaje universitario...

al café, al mate y al negro Dolina, porque tantas noches de desvelo.

a Vir, mi compañera de vida todos estos años, quien me bancó incondicionalmente y brindó el amor y los ánimos necesarios para que finalmente llegue este día.

a mi Mamá Liliana, mi Papá Roberto y mi hermana Daniela, por ayudarme a estudiar y apoyarme constantemente, con paciencia, hasta el final.

una vez más, a mi Viejo... porque me encantaría poder estar compartiendo esto con vos...

Índice general

Capítulo 1

Introducción

Uno de los principales problemas presentes en aplicaciones relativas a robots autónomos móviles consiste en el planeamiento de una trayectoria segura hacia determinados objetivos, mientras el robot se desplaza a través de un entorno potencialmente desconocido. Para esto, resulta necesario contar con una representación del ambiente que se está transitando (mapa) y la ubicación del robot dentro del mismo. El mapa debe ser suficientemente detallado (denso) para que el robot pueda realizar un planeamiento seguro, evitando posibles obstáculos.

El problema de construir incrementalmente un mapa del entorno mientras, simultáneamente, la ubicación del robot se estima es conocido como SLAM (*Simultaneous Localization and Mapping*) [1, 2].

En la última década, ha habido un creciente interés en sistemas de SLAM visual que utilizan cámaras como sensor principal. Las cámaras proveen información detallada de la escena observada y presentan importantes ventajas: bajo costo, menor consumo energético —en comparación con otros sensores, como lásers—, portabilidad y alta disponibilidad en dispositivos móviles. Además, las cámaras son sensores pasivos por lo que no interfieren con otros sensores y pueden utilizarse en entornos interiores, donde el uso de GPS (*Global Positioning System*) no es posible. Así mismo, son menos restrictivos que los encoders, los cuales se encuentran limitados a robots terrestres y son considerablemente imprecisos en terrenos irregulares.

Existen dos enfoques predominantes en los sistemas de SLAM basados en cámaras, denominados monocular y estéreo [3]. Los sistemas de SLAM visual estéreo permiten realizar una reconstrucción de la escena mediante una única observación [4, 5, 6], mientras que un sistema monocular necesita al menos dos mediciones de la cámara para realizar la misma tarea [7, 8]. Además, la reconstrucción 3D que se puede estimar con un sistema monocular se encuentra a un factor de escala desconocido, mientras que la obtenida por un sistema estéreo se encuentra a escala real (debido a que se conoce el desplazamiento que existe entre ambas cámaras).

Determinados enfoques [9, 10, 11] han sido capaces de reconstruir representaciones semi-densas del entorno, que emplean los píxeles de la imagen con alto gradiente, incluyendo vértices, aristas y áreas de alta textura. Sin embargo, estas reconstrucciones pueden ocultar detalles importantes para la navegación, por lo que las reconstrucciones densas resultan claves para el planeamiento seguro de trayectorias para un robot.

Métodos completamente densos han aparecido recientemente en la literatura [11, 12]. Estos enfoques, a diferencia de las soluciones semi-densas, utilizan todos los píxeles de la imagen para generar reconstrucciones completas del entorno, aunque no resultan prácticos en tiempo real o requieren hardware dedicado (por ejemplo, GPU).

El presente trabajo se centra en dotar a un robot móvil con la capacidad de realizar una reconstrucción densa del entorno, al mismo tiempo que se estima su posición, utilizando como único sensor una cámara estéreo. Para esto se trabajó sobre el sistema de SLAM estéreo S-PTAM [6] que mantiene un mapa disperso del entorno —compuesto únicamente por características visuales detectadas en las imágenes— para obtener una estimación precisa de la posición de la cámara en tiempo real. Dada la localización del robot estimada por S-PTAM, una reconstrucción 3D completa es computada paralelamente por el módulo de densificación. Si bien nuestro trabajo se implementó sobre este sistema, puede ser empleado sobre otras implementaciones de SLAM estéreo basados en *keyframes*.

La solución propuesta se compone de tres hilos de ejecución paralelos: generación y reproyección de mapas de disparidad, transformación y fusión de nubes de puntos, y refinamiento del mapa global. Utilizamos la librería LIBELAS [13] para el cómputo de mapas de disparidad, cuyos píxeles a su vez son reproyectados para generar una nube de puntos tridimensional local. Transformamos la nube de puntos local usando la posición global proveniente del sistema de SLAM subyacente, refinando continuamente las nubes de puntos locales a lo largo de todo el proceso. Cada nube de puntos local es filtrada previamente —por medio de fusión de puntos redundantes y detección de outliers— empleando un método similar al propuesto en [14].

Finalmente, el método desarrollado se programó como un nodo del framework ROS (*Robot Operating System*) para ser ejecutado en paralelo y conjuntamente con el sistema S-PTAM, del que se obtienen las posiciones globales. Haremos referencia al sistema propuesto en esta tesina como S-PTAM Denso. Los experimentos realizados sobre datasets de dominio público muestran que el sistema es apto para generar en tiempo real una reconstrucción densa, sin requerir el uso de GPUs, con precisión similar a otros métodos de densificación presentes en el estado del arte.

1.1. Organización del trabajo

El capítulo 2 presenta el estado del arte en sistemas SLAM densos. En el mismo se comentan los distintos paradigmas para abordar la reconstrucción 3D o mapeo denso en sistemas (SLAM), y se analizan los métodos más relevantes de la literatura. En el capítulo 3 se presentan los fundamentos básicos de visión y robótica utilizados a lo largo de toda la tesina. En el capítulo 4 se detalla el método de SLAM denso desarrollado. En el capítulo 5 se evalúa S-PTAM Denso en datasets de dominio público y se discuten los resultados obtenidos. Por último, en el capítulo 6 se presentan las conclusiones y trabajo futuro que se derivan del presente trabajo de tesina.

1.2. Publicaciones

Publicaciones relevantes de la tesina:

- A. D'Alessandro, T. Pire, R. Baravalle: **Hacia una densificación de sistemas SLAM dispersos basados en visión estéreo**. En IX Jornadas Argentinas de Robótica (JAR) 2017. 15, 16 y 17 de Noviembre de 2017. Facultad Regional Córdoba, Córdoba, Universidad Tecnológica Nacional. Noviembre 2017.
- T. Pire, R. Baravalle, A. D'Alessandro, J. Civera: **Real-time and Locally Dense Stereo SLAM**. En Robotics and Autonomous Systems (RAS). (En revisión).

Capítulo 2

Trabajo relacionado

La presente tesina investiga y propone un método de reconstrucción 3D o mapeo denso en tiempo real sobre sistemas de localización y mapeo simultáneo (SLAM) basado en visión estéreo. En este capítulo, se exponen brevemente los trabajos relacionados al problema de mapeo denso en SLAM.

2.1. Reconstrucción 3D

La reconstrucción 3D se define como el proceso mediante el cual se captura y representa la forma y apariencia de entornos u objetos presentes en la realidad. Ha sido un tópico central en visión artificial durante décadas, siendo también una línea de investigación activa en diversos campos como el diseño asistido por computadora, imágenes médicas, realidad virtual y aumentada, robótica movil, entre otros. Dependiendo del campo de aplicación y los sensores utilizados, se han propuesto numerosos métodos para abordar las variantes de este problema.

Dado el enfoque de la presente tesina, resultan de interés principalmente aquellos métodos de reconstrucción 3D que utilizan cámaras como sensores primarios, es decir que la información del entorno es capturada a través de imágenes. En las últimas décadas numerosos trabajos se han publicado en esta área [15, 16, 8, 6, 17, 18], lo cual demuestra la relevancia actual del problema abordado en esta tesina.

2.2. SLAM

Complementariamente al problema de reconstrucción 3D, particularmente en el campo de la robótica móvil, se plantea el problema de localización y mapeo simultáneo, conocido como SLAM (*Simultaneous Localisation and Mapping*). SLAM es el proceso mediante el cual un robot móvil es capaz de construir incrementalmente un mapa del entorno que se encuentra transitando y, simultaneamente, usar este mapa para determinar su posición y orientación en el mismo.

El interés sobre el problema de SLAM ha crecido en gran medida debido a que su solución resulta útil en la navegación autónoma de robots. Recientemente, el problema de SLAM ha sido extensivamente estudiado utilizando diferentes sensores: IMUs (Unidades de Medición Inercial) [19], SONAR [20], infrarrojos [21], escáneres láser [22], GPS (Sistema de Posicionamiento Global) [23], encoders [24] y cámaras [25, 26, 27, 28].

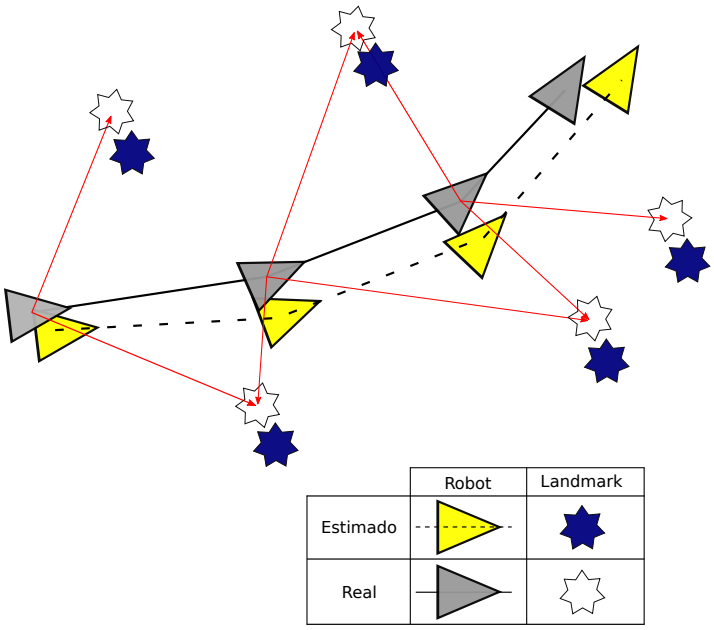


Figura 2.2.1: Representación del problema de SLAM. Las posiciones reales no son conocidas en ningún momento. El robot observa las *landmarks* desde cada posición a medida que se desplaza. Simultáneamente, las posiciones del robot y de las *landmarks* son estimadas en base a estas observaciones. Imagen adaptada de [1].

En la Figura 2.2.1 se ilustra el problema de SLAM. A partir de la detección y seguimiento de marcas naturales del ambiente (*landmarks*), los sistemas de SLAM pueden estimar tanto la posición del robot como la ubicación de estas marcas en el entorno. El mapa es construído incrementalmente con las posiciones estimadas de dichas marcas, las cuales son ajustadas a lo largo de la trayectoria a medida que son observadas.

Una completa introducción a SLAM, descripción del problema y soluciones al mismo pueden encontrarse en [1, 2, 29].

2.3. V-SLAM - Localización y mapeo simultáneo visual

Recientemente ha habido un creciente interés en sistemas de SLAM visual, que utilizan cámaras como sensor principal. Las cámaras proveen una rica información de la escena observada y presentan importantes ventajas: bajo costo, menor consumo energético —en comparación con otros sensores, como lásers—, portabilidad y alta disponibilidad en dispositivos móviles. Además, las cámaras son sensores pasivos por lo que no interfieren con otros sensores y pueden utilizarse en entornos interiores, donde el uso de GPS puede verse imposibilitado. Asímismo, son menos restrictivos que los encoders, los cuales se encuentran limitados a robots terrestres y resultan imprecisos en terrenos irregulares.

Existen dos enfoques predominantes en los sistemas de SLAM basados en cámaras, denominados monocular y estéreo [3]. Los sistemas de SLAM visual estéreo proveen información

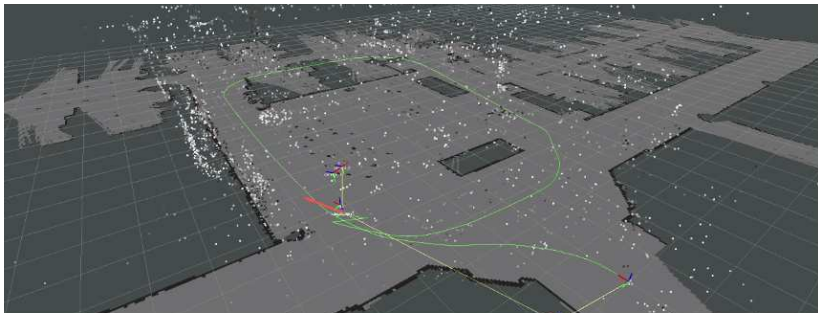


Figura 2.3.1: Mapa disperso generado por el sistema S-PTAM [6] sobre una secuencia del Dataset Level 7 [32] en un entorno de oficinas. En la imagen se observan el mapa de puntos (features) disperso y la trayectoria del robot (línea verde). Como referencia, el contorno de la habitación es proyectado sobre el fondo.

sobre la profundidad de los píxeles mediante una única observación [4, 5, 6], lo cual requiere mayor número de cómputos en el caso monocular [7, 8].

Los métodos de SLAM visual utilizan una secuencia de imágenes como entrada. Mediante la extracción de características visuales en la imágenes (*features*) es posible detectar y realizar un seguimiento de marcas naturales del ambiente a lo largo de la secuencia. De esta manera, los métodos de SLAM visual son capaces de estimar con precisión el desplazamiento de la cámara y computar un mapa disperso del entorno en tiempo real [17, 6, 30, 31]. En la Figura 2.3.1 puede observarse un mapa disperso generado por el sistema S-PTAM [6].

2.3.1. Características visuales

Los enfoques de SLAM visual basados en características dependen de la calidad y cantidad de características visuales locales (*features*) presentes en la imagen. Estos puntos salientes —o puntos claves— deben ser descriptos unívocamente para poder establecer correspondencias entre diferentes imágenes. En la literatura existe una gran variedad de extractores de features, siendo los más usados: Shi–Tomasi [33], SIFT [34], SURF [35], FAST [36], ORB [37] y KAZE [38]. En la Figura 2.3.2 se muestran los puntos detectados y asociados por el método ORB en un par de imágenes de ejemplo.

Para que las características visuales resulten útiles en la búsqueda de correspondencias en sucesivas imágenes, su descriptor debe poseer importantes propiedades:

- Independencia en la posición: si el mismo punto clave es extraído en diferentes posiciones de la imagen, el descriptor debe ser el mismo.
- Robustez frente a transformaciones en la imagen: reconocer el mismo feature frente a cambios de iluminación (por ejemplo efecto del sol en entornos exteriores) o perspectiva (observación del mismo objeto desde distintas posiciones).
- Invariante a la escala: el mismo descriptor debe ser asignado independiente de cambios en la escala de la imagen.

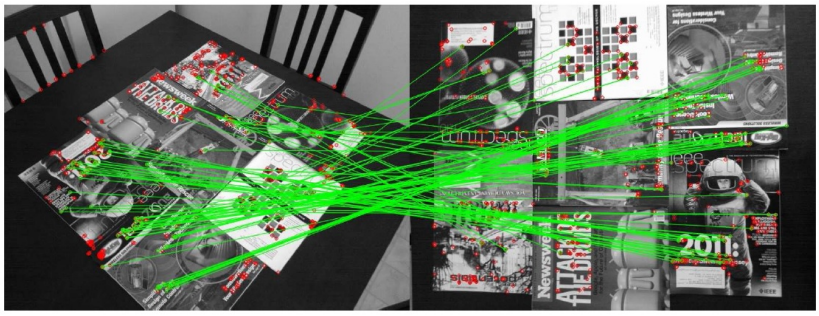


Figura 2.3.2: Asociación de características extraídas por el método ORB. Notar la robustez del método ante la rotación de la imagen.

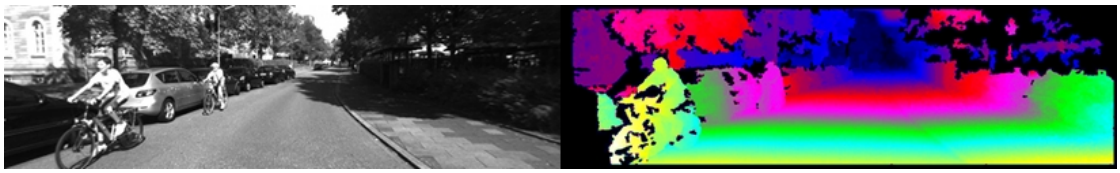


Figura 2.4.1: Izquierda: Imagen original. Derecha: Mapa de disparidad denso calculado mediante LIBELAS [13]. Color amarillo (mayor disparidad), azul (menor disparidad). Imagen obtenida de [13].

2.4. V-SLAM denso

Como se analizó en la sección 2.3, los métodos V-SLAM son capaces de generar un mapa disperso del entorno de manera precisa, aunque este contiene solamente una pequeña porción de los puntos 3D de la escena. Una reconstrucción 3D completa debe ser lo suficientemente densa para poder modelar la geometría y apariencia del entorno, lo que resulta clave para que el robot pueda navegar de manera segura. Por esto, dentro del área de V-SLAM se han desarrollado distintos enfoques para la obtención de mapas densos, que analizaremos a continuación.

2.4.1. Densificación de métodos dispersos

Aprovechando que los métodos de SLAM visual basados en features producen resultados precisos con respecto a la localización del robot, es posible agregar información densa al mapa disperso obtenido.

Mapas de disparidad densos

Un mapa de disparidad denso es aquel donde a cada píxel de una imagen se le asigna un valor de *disparidad*[1] y, por ende, su profundidad es conocida. Las definiciones y relaciones geométricas involucradas en la búsqueda de correspondencias entre un par de imágenes y el cálculo de disparidad se describen con detalle en la sección 3.2. La Figura 2.4.1 muestra un mapa de disparidad denso obtenido desde un par de imágenes estéreo.

[1] la disparidad se define como la distancia entre dos puntos correspondientes en la imagen izquierda y derecha de un par de imágenes estéreo

El problema de asociación densa de píxeles entre imágenes ha sido abordado en la última década por diversos trabajos [39, 40] obteniendo mapas precisos y densos en pocos segundos [41, 42, 43], o incluso pocos milisegundos [44, 45, 46]. En [47, 48] se expone una taxonomía y clasificación de algoritmos de búsqueda de correspondencias entre pares de imágenes.

Particularmente, el interés en los mapas de disparidad ha crecido con la introducción de cámaras estéreo ya que la línea base (distancia entre ambas cámaras) y las posiciones relativas entre las cámaras son fijas. Esto simplifica la búsqueda de correspondencias y el cálculo de disparidad, como se verá en las secciones 3.2.2 y 3.2.3.

Los algoritmos de búsqueda de correspondencias entre imágenes deben resolver algunos problemas inherentes, como son:

- Oclusión: Al observar una escena desde distintas posiciones posiblemente existan oclusiones entre los objetos. Por esta razón, ciertas regiones de una imagen pueden no tener correspondencias en la otra imagen.
- Especularidad: La percepción de los objetos puede variar a medida que la cámara se desplaza debido a la especularidad de estos (es decir variaciones en el brillo debido a la reflectancia de la superficie), dificultando la asociación basada en la apariencia.
- *Muro blanco*: Bajo este nombre se denomina al problema de encontrar correspondencias en regiones con poca textura —bajo gradiente en la función de intensidad—. Los píxeles que se encuentran en el interior de estas regiones no presentan suficiente información para ser distinguidos y asociados, por ejemplo una pared completamente blanca.

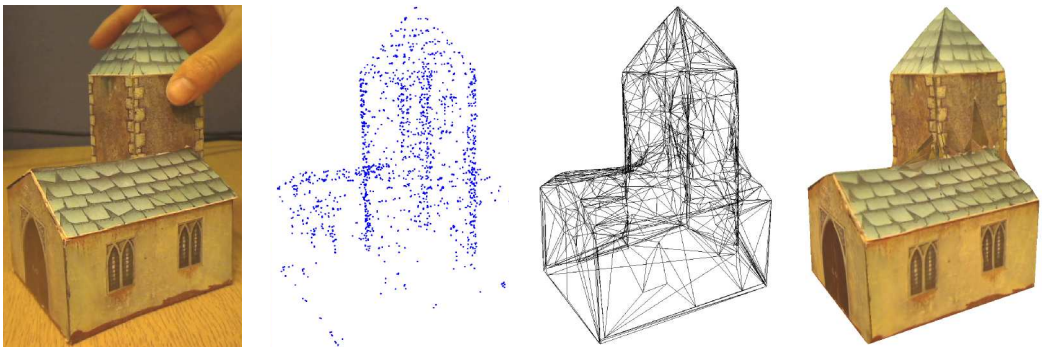
Distintos trabajos han aprovechado los mapas de disparidad densos para densificar la reconstrucción 3D de métodos dispersos. En el trabajo StereoScan [14] se parte de un sistema SLAM basado en *features* obteniendo un mapa disperso. Luego, los pares de imágenes provenientes de una cámara estéreo son procesados para generar mapas de disparidad densos mediante LIBELAS [13], aumentando la cantidad de puntos presentes en el mapa. De manera similar, los trabajos [49, 50] utilizan sistemas de SLAM basados en EKF (Filtro de Kallman Extendido) para realizar la localización y mapeo, densificando luego el mapa resultante mediante información de disparidad densa.

Reconstrucción de planos y polígonos

Aprovechando la precisión en la trayectoria y la nube de puntos dispersa generadas por sistemas SLAM basados en *features*, se han propuesto distintos métodos que procesan el mapa disperso para modelar planos y polígonos, agregando información a la reconstrucción 3D.

El enfoque propuesto en [51] utiliza el algoritmo de *j-linkage* [52] para agrupar los puntos del mapa reconstruyendo los planos dominantes de la escena. Siguiendo el mismo razonamiento, en [53] estos planos son estimados mediante el método iterativo *RANSAC* [54].

De manera similar, en [55, 56, 57] los puntos del mapa disperso son utilizados como vértices para obtener una *mesh* (malla) continua de triángulos. En la Figura 2.4.2 pueden observarse las etapas involucradas en [57]. Estos métodos aprovechan la precisión de la nube de puntos para generar una buena aproximación inicial de la superficie que componen, que luego puede ser refinada mediante consistencia en las diferentes imágenes.



(a) (b) (c) (d)

Figura 2.4.2: Etapas del método de reconstrucción 3D propuesto en [57]. (a) imagen del objeto. (b) nube de puntos dispersa. (c) modelo de superficie mediante malla de triángulos. (d) modelo 3D final con textura añadida.

2.4.2. Métodos basados en superpíxeles

En la sección anterior 2.4.1 vimos que el enfoque predominante en SLAM se basó tradicionalmente en encontrar correspondencias en áreas de la imagen fuertemente texturadas —características visuales de bajo nivel—. En consecuencia largas regiones sin textura, usualmente presentes en entornos interiores y urbanos, son difíciles de reconstruir por estos sistemas. Para superar esta dificultad y aumentar la densidad de estas reconstrucciones, se han desarrollado métodos que explotan características visuales de mediano nivel, como los superpíxeles.

Los superpíxeles representan regiones de la imagen con textura homogénea, es decir un conjunto de píxeles con valores similares. La segmentación de una imagen en superpíxeles puede variar en forma y tamaño de acuerdo al algoritmo [58, 59, 60, 61, 62] y configuración utilizados, como se ejemplifica en la Figura 2.4.3.

A diferencia de los *features*, los superpíxeles poseen baja repetibilidad y una marcada dependencia a oclusiones, como puede observarse en la Figura 2.4.4. La baja repetibilidad significa que una misma región, observada desde distintas posiciones de la cámara, puede dar lugar a diferentes segmentaciones dificultando la asociación de superpíxeles a lo largo de una secuencia de imágenes.

En [63] se segmentan las imágenes utilizando el algoritmo propuesto por Felzenszwalb et al. [58] y se asume que cada superpíxel corresponde a una superficie planar 3D denominada superpíxel 3D o parche. Los contornos de los superpíxeles son proyectados y mapeados a lo largo de la secuencia para validar y ajustar la posición y figura de estas superficies planares. En [63] este sistema es integrado a PTAM [30] y DTAM [11] densificando en tiempo real los mapas provenientes de estos sistemas.

De manera similar, en [64] se utiliza información semi-densa (es decir píxeles con alto gradiente en las imágenes) para realizar la reconstrucción del entorno agregando superpíxeles 3D.

Figura 2.4.3: Segmentación de imágenes en superpíxeles de diferente tamaño. Imagen obtenida de [62].

2.4.3. Métodos semánticos

En SLAM visual se han comenzado a utilizar características visuales de alto nivel de abstracción, que permiten reconocer y mapear objetos presentes en el entorno. Este área de investigación se encuentra relacionada a tópicos de aprendizaje automatizado e inteligencia artificial.

Dentro del campo de SLAM visual monocular, en [65] se logra reconocer un conjunto de objetos, determinado con anterioridad, en el mapa a través de los descriptores de bajo nivel (*features*) que presenta. En SLAM++ [66] se profundiza el enfoque usando directamente los objetos conocidos como los descriptores del mapa y en [67, 68] la estructura de una habitación es utilizada como descriptores de alto nivel.

En [12] se utiliza conocimiento previo sobre la forma de distintas categorías de objetos para clasificarlos y refinar su reconstrucción 3D. Recientemente, apuntando a reconstrucciones incrementales de gran escala, se publicó el trabajo [69] que realiza segmentación semántica en tiempo real para distintas categorías como se ilustra en la Figura 2.4.5.

2.4.4. Métodos directos

En los últimos años, los métodos directos propiciaron una nueva línea de investigación [9, 10, 11] que, en vez utilizar características visuales (*features*), operan directamente sobre las intensidades de la imagen para realizar la localización y mapeo. A su vez, estos métodos aprovechan mayor parte de la información disponible en las imágenes —píxeles de alto gradiente, incluyendo vértices, aristas y áreas de alta textura— obteniendo mapas 3D de mayor densidad denominados semi-densos, como puede observarse en la Figura 2.4.6.

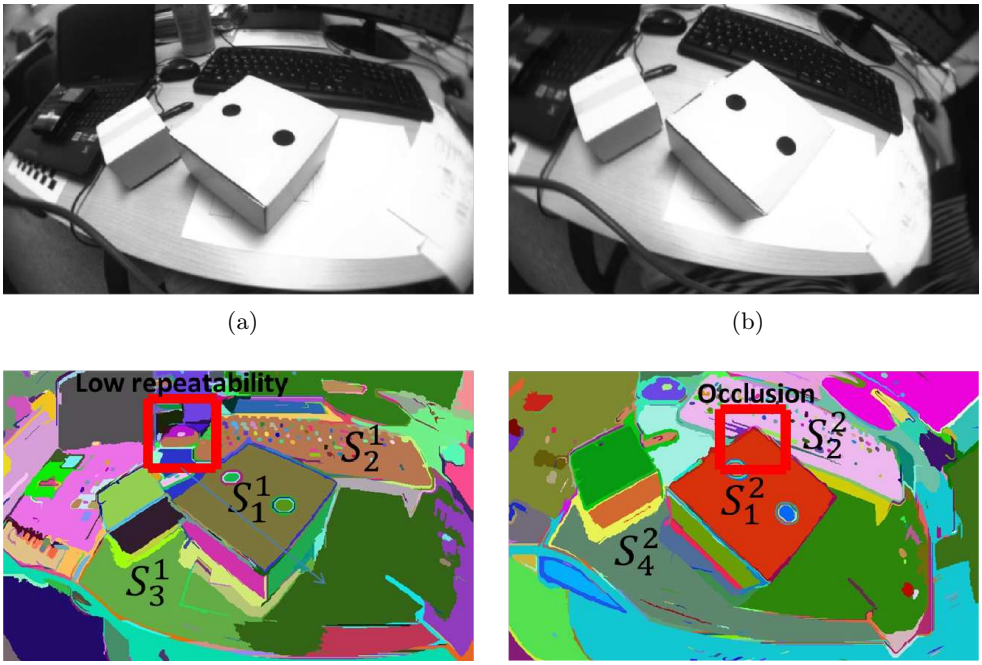


Figura 2.4.4: (a)(b) Par de imágenes pertenecientes a una misma secuencia. (c)(d) Segmentación en superpíxeles correspondiente al par de imágenes. El superpíxel s_1 es segmentado con precisión en ambas imágenes (s_1^1 y s_1^2). El superpíxel s_2, aunque es observado en ambas imágenes (s_2^1 y s_2^2), presenta baja repetibilidad: notar que, en el recuadro rojo en (c), s_2^1 se extiende por fuera de los límites del teclado, a diferencia de s_2^2. Además, como se observa en el recuadro rojo en (d), el superpíxel s_2 es segmentado de diferentes formas debido a que la oclusión varía según la posición de la cámara. Imagen adaptada de [63].

Sin embargo, el nivel de desarrollo de SLAM directo aún es bajo en comparación con los métodos basados en características visuales, resultando ser menos flexibles y precisos para la eliminación de outliers [72].

Uno de los trabajos más importantes es LSD-SLAM [73] (*Large Scale Direct SLAM*) que implementa una solución para secuencias de larga escala en tiempo real en CPU (sin necesidad de utilizar GPU) utilizando una cámara monocular. Posteriormente, este sistema se extiende para cámaras estéreo [71] y omnidireccionales [74].

Finalmente, en [75] se ha publicado un sistema de V-SLAM directo que fusiona información visual e inercial provenientes de una cámara monocular y una IMU respectivamente. La información inercial permite conocer la velocidad, orientación y fuerzas gravitacionales que afectan al robot, lo que contribuye a mejorar y acelerar la estimación de las posiciones de la cámara. Por otro lado, la fusión visual-inercial resuelve la ambigüedad en la escala, limitación inherente a los sistemas de SLAM monocular, que implica la existencia de un factor desconocido entre el mapa y la escala del mundo real.

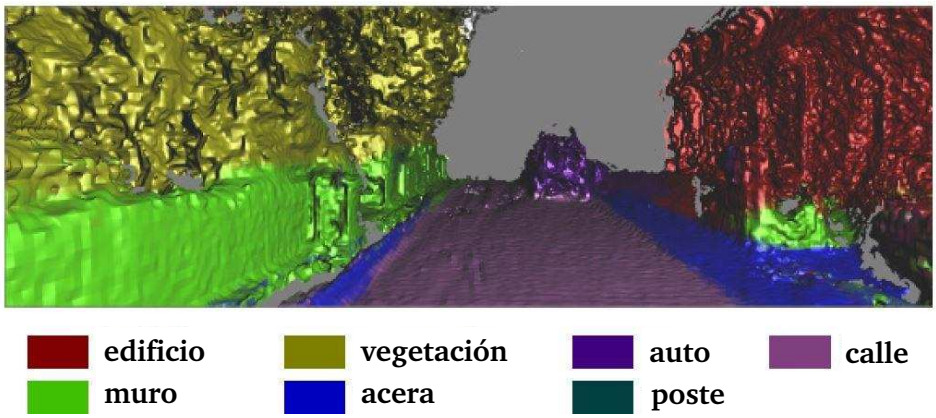


Figura 2.4.5: Reconstrucción 3D (arriba) y segmentación semántica (abajo) del dataset KITTI secuencia 05. Imagen obtenida de [69].

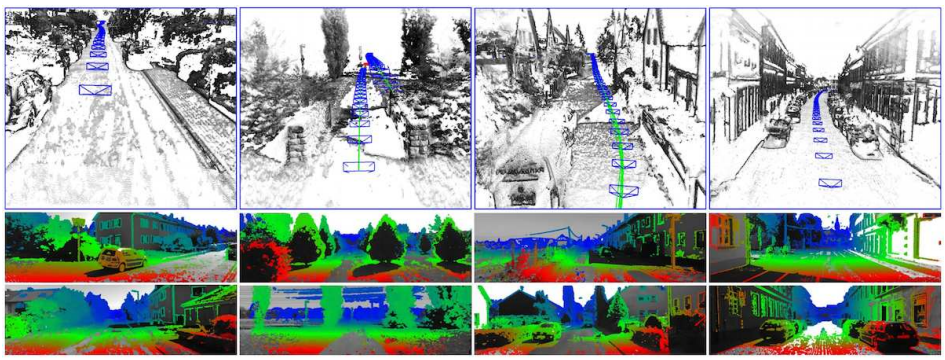


Figura 2.4.6: Arriba: Nubes de puntos 3D semi-densas del entorno reconstruído. Centro y abajo: Mapas de profundidad generados sobre el dataset KITTI [70]. Los mapas de profundidad son coloreados según la distancia de los puntos a la cámara utilizando el espectro entre los colores rojo (más cercano) y azul (más lejano). Imagen obtenida de [71].

Capítulo 3

Cámaras como sensores

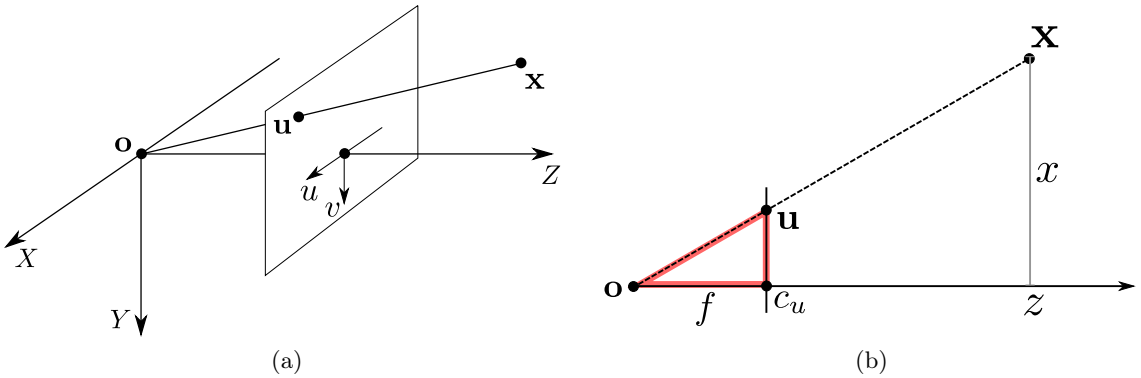
En este capítulo se presenta el marco conceptual utilizado a lo largo de la tesina. Los conceptos matemáticos y geométricos detallados en este capítulo son de común conocimiento en el campo de la robótica y visión computarizada, y han sido principalmente recopilados de [25], entre otras fuentes.

3.1. Modelo y geometría de una cámara monocular

Una cámara es definida matemáticamente como una correspondencia entre el mundo 3D y una imagen 2D. Existen diferentes modelos para representar una cámara, siendo el modelo de *cámara pinhole* (cámara estenopeica) aquel que resulta más simple y mayormente utilizado.

3.1.1. Modelo de cámara *pinhole*

En el modelo de cámara *pinhole,* el punto de la imagen $\mathbf{u} = \begin{bmatrix} u & v \end{bmatrix}^\top$ es determinado como la intersección entre el *plano de la imagen* —también denominado *plano focal*— y el rayo que une el punto del mundo 3D $\mathbf{x} = \begin{bmatrix} x & y & z \end{bmatrix}^\top$ con el *centro focal* $\mathbf{o}$ de la cámara. En la Figura 3.1.1 se ilustra esta definición y las relaciones geométricas involucradas en este modelo de cámara.



(a) (b)

Figura 3.1.1: Relaciones geométricas presentes en el modelo de cámara *pinhole*.

Llamamos *rayo principal* o *eje principal* de la cámara al rayo que se origina en el centro focal **o** y es perpendicular al plano de la imagen. El punto donde este rayo intersecta al plano de la imagen es denominado *punto principal.*

Consideremos que el centro focal se encuentra en el origen de coordenadas y el plano focal expresado como $Z = f$. Utilizando la propiedad de semejanza de triángulos puede verse facilmente que el punto 3D $\mathbf{x} = \begin{bmatrix} x & y & z \end{bmatrix}^\top$ es mapeado al punto $\mathbf{u} = \begin{bmatrix} fx/z & fy/z \end{bmatrix}^\top$ en el plano de la imagen. Esto da lugar a la siguiente proyección central:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \longmapsto \begin{bmatrix} fx/z \\ fy/z \end{bmatrix}. \tag{3.1.1}$$

Matriz de calibración intrínseca K

El mapeo de la ecuación (3.1.1) asume que el origen de coordenadas del plano de la imagen se encuentra en el punto principal. En la práctica, esto no siempre se cumple, por lo que en general la proyección se expresa como:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \longmapsto \begin{bmatrix} fx/z + c_u \\ fy/z + c_v \end{bmatrix}^\top, \tag{3.1.2}$$

siendo $\begin{bmatrix} c_u & c_v \end{bmatrix}^\top$ la posición del punto principal.

Notamos con $\dot{\mathbf{x}} = \begin{bmatrix} x & y & z & 1 \end{bmatrix}^\top$ la representación en coordenadas homogéneas del punto $\mathbf{x} = \begin{bmatrix} x & y & z \end{bmatrix}^\top$. De esta manera, la proyección central del modelo de cámara *pinhole* de la ecuación (3.1.2) puede expresarse como:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \longmapsto \begin{bmatrix} fx + zc_u \\ fy + zc_v \\ z \end{bmatrix} = \begin{bmatrix} f & & c_u & 0 \\ & f & c_v & 0 \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{3.1.3}$$

Definiendo la *matriz de calibración intrínseca* $\mathbf{K}$ como:

$$\mathbf{K} = \begin{bmatrix} f & 0 & c_u \\ 0 & f & c_v \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.1.4}$$

la ecuación (3.1.3) tiene la siguiente forma concisa:

$$\dot{\mathbf{u}} = \mathbf{K} \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \dot{\mathbf{x}}.$$

Matriz de proyección P

En general, los puntos del ambiente son expresados en referencia a un sistema de coordenadas conocido como *mundo.* Dado que la cámara no se encuentra necesariamente ubicada en el centro de este sistema, como se ilustra en la Figura 3.1.2, las ecuaciones previamente definidas deben extenderse agregando la transformación existente entre los sistemas de coordenadas de la cámara y el mundo.

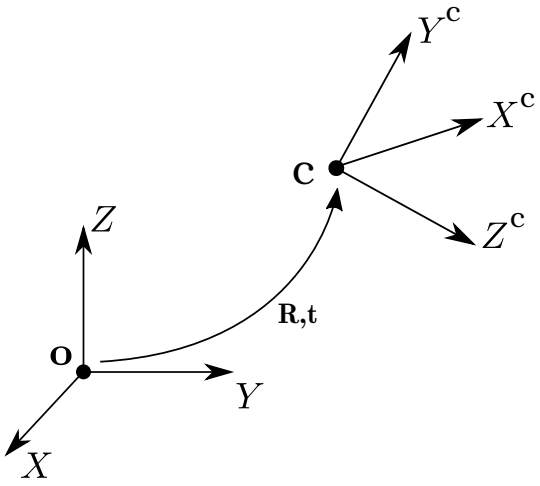


Figura 3.1.2: En general, el sistema de coordenadas de la cámara se encuentra relacionado con el sistema de coordenadas del mundo mediante una rotación $\mathbf{R}$ y una traslación $\mathbf{t}$.

Sea $\dot{\mathbf{x}}^{\mathrm{w}} = \begin{bmatrix} x & y & z & 1 \end{bmatrix}^{\top}$ un punto del espacio en referencia al sistema de coordenadas del mundo y $\dot{\mathbf{x}}^{\mathrm{c}}$ el mismo punto expresado en el sistema de coordenadas de la cámara, es posible definir una transformación $\mathbf{E}^{\mathrm{cw}}$ tal que:

$$\dot{\mathbf{x}}^{\mathrm{c}} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \mathbf{E}^{\mathrm{cw}} \dot{\mathbf{x}}^{\mathrm{w}} \ .$$

En particular, $\mathbf{E}^{\mathrm{cw}}$es una transformación perteneciente al grupo de movimientos de cuerpo rígido 3D (*Lie Group*, **SE(3)**) [76], donde $\mathbf{R}$ es una matriz de rotación y $\mathbf{t}$ un vector de traslación. Un cuerpo rígido 3D hace referencia a un conjunto de puntos en el espacio que se mueven de tal manera que no se alteran las distancias entre ellos.

Utilizando la matriz de calibración intrínseca $\mathbf{K}$ definida en (3.1.4), se define la matriz de proyección $\mathbf{P}$:

$$\mathbf{P} = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} . \tag{3.1.5}$$

De esta manera, es posible proyectar cualquier punto 3D $\dot{\mathbf{x}}^{\mathrm{w}}$ en el sistema de coordenadas del mundo al correspondiente punto $\dot{\mathbf{u}}$ en el plano de la imagen mediante:

$$\dot{\mathbf{u}} = \mathbf{P} \dot{\mathbf{x}}^{\mathrm{w}} \ . \tag{3.1.6}$$

3.1.2. Re-proyección

Dado un punto $\mathbf{u}$ en la imagen, es posible determinar el conjunto de puntos 3D en el espacio que son proyectados sobre este punto. El conjunto de puntos conforma un rayo en el espacio que pasa por el centro focal y el punto de la imagen $\mathbf{u}$. Así, el rayo $\mathbf{x}(\lambda)$ puede definirse como:

$$\mathbf{x}(\lambda) = \mathbf{P}^{+}\mathbf{u} + \lambda\mathbf{o}\ ,$$

donde $\mathbf{P}^{+}$ es la matriz *pseudo-inversa*[1] de $\mathbf{P}$ y $\mathbf{o}$ es el centro focal de la cámara. La Figura 3.1.3 muestra la re-proyección de un punto perteneciente a la imagen. Puede observarse que el rayo $\mathbf{x}(\lambda)$ es la linea que pasa por los puntos $\mathbf{P}^{+}\mathbf{u}$ y $\mathbf{o}$ en el espacio 3D dado que $\mathbf{P}\mathbf{P}^{+}\mathbf{u} = \mathbf{I}\mathbf{u} = \mathbf{u}$ y $\mathbf{P}\mathbf{o} = \mathbf{0}$.

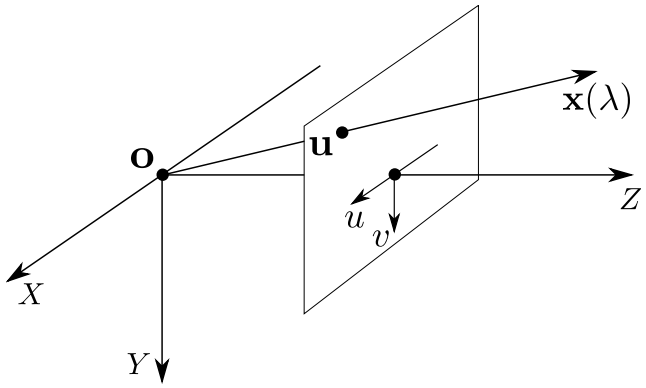


Figura 3.1.3: Re-proyección de un punto de la imagen.

De la anterior definición puede concluirse que no es posible realizar una reconstrucción 3D del entorno partiendo de una única imagen. En la sección 3.2 veremos cómo es posible obtener una reconstrucción 3D de la escena capturada desde dos puntos de vista diferentes, cuando el desplazamiento entre ambas cámaras es conocido.

3.1.3. Frustum culling

El campo visual de la cámara es la región 3D que contiene todos los puntos que son potencialmente visibles en el plano de la imagen. En el modelo de cámara *pinhole*, el campo visual tiene la forma de una pirámide trunca, como se ilustra en la Figura 3.1.4. Como se verá a continuación, esta región es determinada por los parámetros intrínsecos y extrínsecos de la cámara.

Frustum culling es el proceso mediante el cual se filtra el entorno descartando aquellos objetos que no se encuentran dentro del campo visual. De esta manera, se evita desperdiciar recursos renderizando objetos que no son directamente visibles por la cámara.

En el marco de la presente tesina, el entorno es una nube de puntos y mediante *frustum culling* es posible descartar aquellos puntos que no es necesario proyectar dado que su proyección se encuentra fuera de los límites del plano de la imagen.

[1]La matriz pseudo-inversa $\mathbf{P}^{+}$ está definida como $\mathbf{P}^{+} = \mathbf{P}^{\top}\left(\mathbf{P}\mathbf{P}^{\top}\right)^{-1}$.

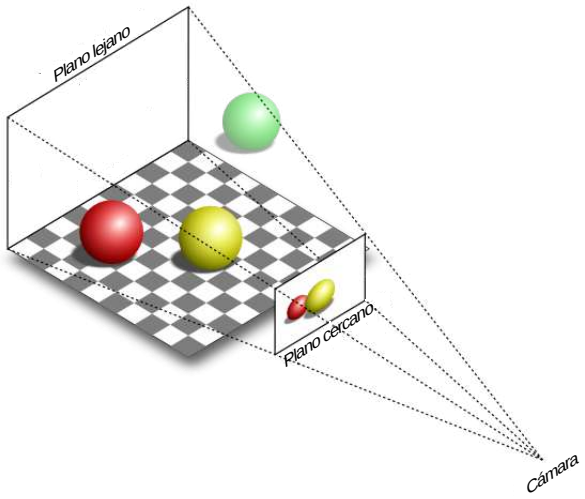


Figura 3.1.4: Representación del campo visual en un modelo de cámara *pinhole*. La esfera verde queda fuera de la pirámide trunca, por lo que no se verá proyectada sobre el plano de la imagen.

Vértices del campo visual de la cámara

El vértice de la pirámide es el centro focal $\mathbf{o}$ y su base está definida por el plano lejano $\boldsymbol{\pi}_{lejano}$, potencialmente infinito. A su vez, la pirámide es delimitada por un plano cercano $\boldsymbol{\pi}_{cercano}$, que da el nombre de pirámide trunca. Los planos $\boldsymbol{\pi}_{cercano}$ y $\boldsymbol{\pi}_{lejano}$ se encuentran ubicados a distancias conocidas $f_{cercano}$ y f_{lejano} respectivamente, y son perpendiculares al rayo principal.

El área rectangular del campo visual sobre estos planos puede ser delimitada en función de las dimensiones del plano focal $\boldsymbol{\pi}_{cam}$ y las distancias $f_{cercano}$ y f_{lejano}. Como se muestra en la Figura 3.1.5, aplicando la propiedad de triángulos semejantes pueden obtenerse facilmente las coordenadas de los vértices que definen el área rectangular del campo visual sobre el plano lejano $\boldsymbol{\pi}_{lejano}$. El caso del plano cercano $\boldsymbol{\pi}_{cercano}$ es análogo.

Sea $\mathbf{x} = \begin{bmatrix} x & y & f \end{bmatrix}^\top$ uno de los vértices del plano focal, expresado en el sistema de coordenadas de la cámara como se muestra en la Figura 3.1.5a. Por propiedad de triángulos semejantes, pueden derivarse las coordenadas del vértice $\mathbf{x}_{lejano}$ sobre el plano lejano $\boldsymbol{\pi}_{lejano}$:

$$\mathbf{x}_{lejano} = \begin{bmatrix} \frac{x\, f_{lejano}}{f} & \frac{y\, f_{lejano}}{f} & f_{lejano} \end{bmatrix}. \tag{3.1.7}$$

Utilizando la Ecuación (3.1.7) pueden derivarse las coordenadas de cada uno de los vértices que definen el área rectangular del campo visual de la cámara sobre los planos $\boldsymbol{\pi}_{cercano}$ y $\boldsymbol{\pi}_{lejano}$.

Extracción de los planos

Cada plano $\boldsymbol{\pi}_i$ que determina el campo visual de la cámara está conformado por 3 o más vértices no alineados, cuyas coordenadas fueron calculadas mediante las relaciones geométricas

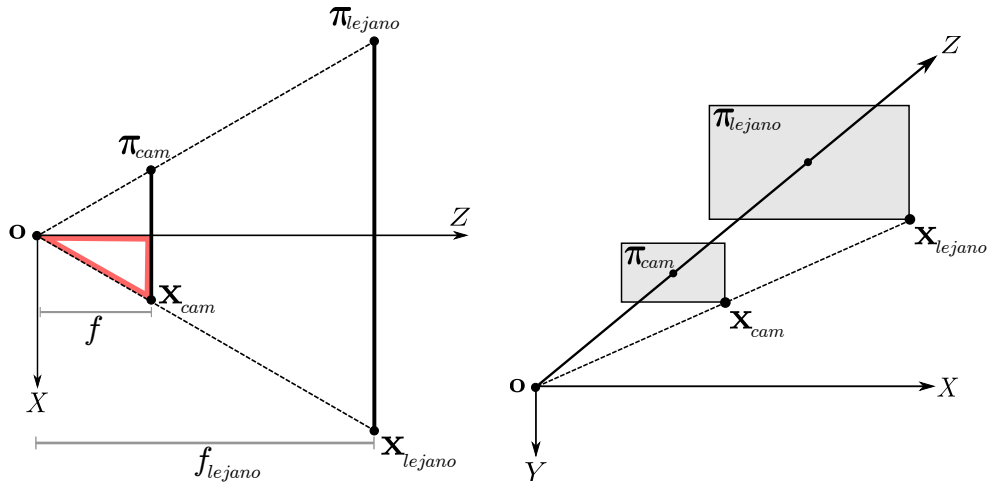


Figura 3.1.5: Relaciones geométricas presentes en el campo visual de una cámara *pinhole*. En ambas figuras se omite el plano cercano por simplificación. Nótese que reemplazando al plano lejano por el plano cercano, se obtienen las mismas propiedades geométricas en relación al plano focal.

enunciadas en la sección anterior 3.1.3. Por lo tanto, es posible definir el plano $\boldsymbol{\pi}_i$ y expresar su ecuación general como:

$$\boldsymbol{\pi}_i = \begin{bmatrix} A_i & B_i & C_i & D_i \end{bmatrix},$$

tal que

$$A_i x + B_i y + C_i z + D_i w = 0 \quad \text{para cada} \quad \dot{\mathbf{x}}^{\mathrm{w}} = \begin{bmatrix} x & y & z & w \end{bmatrix}^\top \in \boldsymbol{\pi}_i \,.$$

Por convención, la dirección del vector normal $\eta_i = \begin{bmatrix} A_i & B_i & C_i \end{bmatrix}^\top$ de cada plano $\boldsymbol{\pi}_i$ apunta hacia el interior de la pirámide. Por lo tanto, un punto 3D $\dot{\mathbf{x}}^{\mathrm{w}} = \begin{bmatrix} x & y & z & w \end{bmatrix}^\top$ se encuentra dentro del campo visual de la cámara si la siguiente ecuación se satisface para cada plano $\boldsymbol{\pi}_i$ que conforma la pirámide:

$$\boldsymbol{\pi}_i \, \dot{\mathbf{x}}^{\mathrm{w}} \geq 0 \,. \tag{3.1.8}$$

3.2. Modelo y geometría de una cámara estéreo

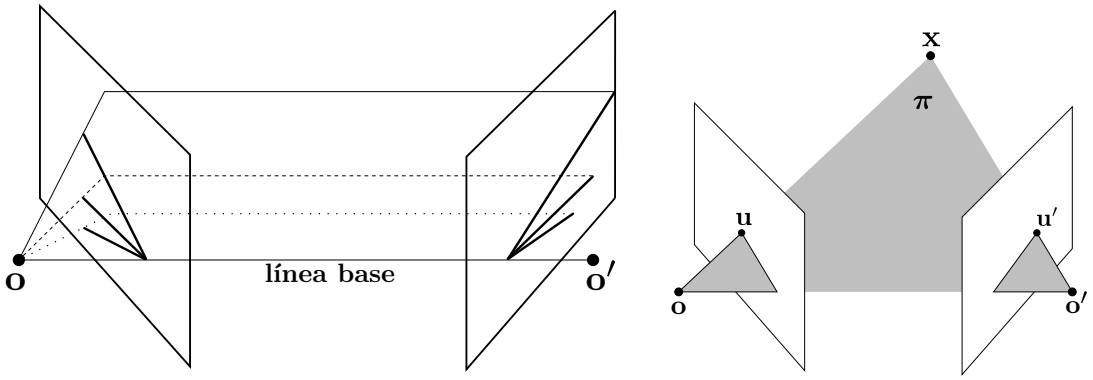
Como se vio en la sección 3.1.2, no es posible realizar una reconstrucción 3D del entorno utilizando una única imagen, sino que se requieren al menos dos imágenes tomadas desde diferentes puntos de vista. En esta sección se introducen los conceptos necesarios para abordar el proceso de reconstrucción 3D a partir de dos imágenes.

3.2.1. Geometría epipolar

La geometría proyectiva intrínseca entre dos cámaras es conocida como *geometría epipolar*. Esta es independiente de la escena observada y depende únicamente de los parámetros internos

y las posiciones relativas de las cámaras involucradas. La *geometría epipolar* entre dos imágenes es esencialmente la geometría determinada por la intersección de los planos de las imágenes con la familia de planos que contengan la *línea base* como eje, como puede observarse en la Figura 3.2.1a. La línea base es la línea que une los centros focales de las cámaras.

Como se ilustra en la Figura 3.2.1b, cualquier punto 3D $\mathbf{x}$ en el espacio, sus proyecciones $\mathbf{u}$ y $\mathbf{u}'$ en los planos de las imágenes y los centros focales de las cámaras, pertenecen a un mismo plano $\boldsymbol{\pi}$. De manera análoga, los rayos re-proyectados desde $\mathbf{u}$ y $\mathbf{u}'$ se intersectan en $\mathbf{x}$, son coplanares y yacen sobre $\boldsymbol{\pi}$.



(a) (b)

Figura 3.2.1: Relaciones presentes en la geometría epipolar de una cámara *pinhole*. Imágenes adaptadas de [25].

Búsqueda de correspondencias sobre la línea epipolar

Cuando solo es conocido el punto $\mathbf{u}$, la búsqueda del correspondiente punto $\mathbf{u}'$ en el segundo plano focal puede acotarse mediante la siguiente propiedad: partiendo del plano $\boldsymbol{\pi}$ que está determinado por la línea base y el rayo definido por $\mathbf{u}$, de la explicación precedente sabemos que el rayo correspondiente al (desconocido) punto $\mathbf{u}'$ también yace sobre el plano $\boldsymbol{\pi}$. Por lo tanto, como se ilustra en la Figura 3.2.2, el punto $\mathbf{u}'$ se encuentra en la línea $\mathbf{l}'$, definida por la intersección de $\boldsymbol{\pi}$ con el plano focal de la segunda imagen. Esta línea $\mathbf{l}'$ es la imagen del rayo re-proyectado desde $\mathbf{u}$ sobre el segundo plano de imagen y se la denomina *línea epipolar* correspondiente al punto $\mathbf{u}$.

Este resultado implica que para cualquier punto $\mathbf{u}$, la búsqueda de su correspondiente punto $\mathbf{u}'$ no necesita cubrir el plano focal completo sino que puede restringirse a la línea epipolar $\mathbf{l}'$. La Figura 3.2.2 muestra la línea epipolar generada por la proyección del rayo, re-proyectado desde $\mathbf{u}$, sobre el plano focal de la segunda imagen.

3.2.2. Rectificación estéreo

La *rectificación estéreo* consiste en proyectar el par de imágenes estéreo sobre un plano de imagen común, de manera que los *puntos correspondientes* (puntos que corresponden al mismo

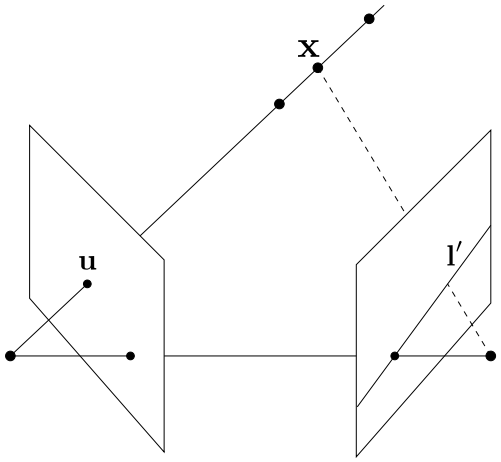


Figura 3.2.2: Línea epipolar $\mathbf{l}'$ resultante de la proyección del rayo, re-proyectado desde $\mathbf{u}$ en la primer imagen (izquierda), sobre el plano focal de la segunda imagen (derecha). Imagen adaptada de [25].

punto 3D) se encuentren alineados en la misma fila. Esta proyección permite considerar las cámaras involucradas como paralelas, es decir que la transformación entre ambas cámaras es únicamente una traslación en el eje horizontal. La Figura 3.2.3 ilustra una cámara estéreo rectificada.

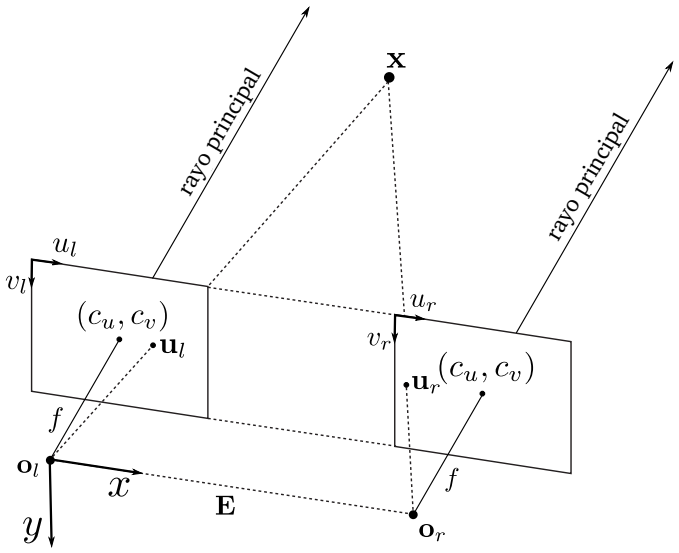
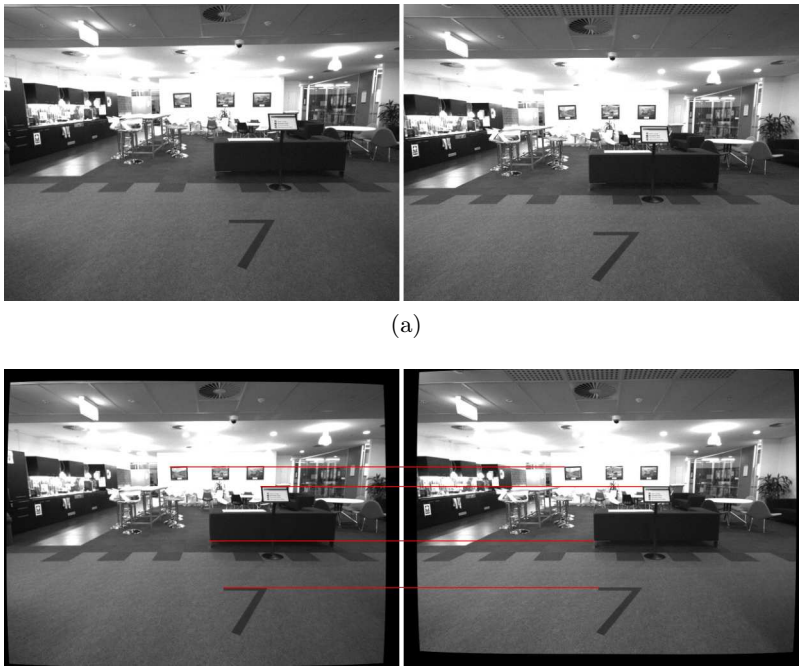


Figura 3.2.3: Rectificación estéreo. Imagen adaptada de [25].

De esta manera, luego de aplicar rectificación estéreo, la búsqueda de correspondencias entre las imágenes es reducida a una búsqueda uni-dimensional, sobre la misma fila. La Figura 3.2.4b muestra el resultado del proceso de rectificación estéreo sobre un par de imágenes. Este

resultado es de suma importancia en el cálculo de *disparidad*. La disparidad se define como la distancia entre dos puntos correspondientes en la imagen izquierda y derecha de un par de imágenes estéreo. Usando el valor de disparidad es posible calcular la posición del punto 3D asociado, como se verá a continuación.



(a)

(b)

Figura 3.2.4: Rectificación estéreo sobre un par de imágenes del Dataset Level 7 [32]. **(a)** Par de imágenes estéreo original provisto por la cámara estéreo. **(b)** El par de imágenes estéreo luego de ser rectificadas. Las líneas rojas asocian algunos puntos correspondientes entre ambas imágenes. Notar que estos puntos se encuentran en la misma fila en ambas imágenes.

3.2.3. Triangulación estéreo

A partir de un par de imágenes estéreo rectificadas es posible realizar una reconstrucción 3D de los puntos correspondientes en ambas imágenes. La Figura 3.2.5 muestra la geometría involucrada en la triangulación estéreo de un punto en el espacio.

Sea $\mathbf{x} = \begin{bmatrix} x & y & z \end{bmatrix}^\top$ un punto 3D cuyas coordenadas son desconocidas. Conociendo las proyecciones correspondientes $\mathbf{u}_l = \begin{bmatrix} u_l & v_l \end{bmatrix}$ y $\mathbf{u}_r = \begin{bmatrix} u_r & v_r \end{bmatrix}$ sobre los planos focales de la imagen izquierda y derecha respectivamente, la posición del punto $\mathbf{x}$ puede ser derivada. Aplicando la propiedad de triángulos semejantes entre el triángulo de línea negra punteada y el triángulo de línea roja en la Figura 3.2.5a, la siguiente ecuación puede ser formulada:

$$\frac{b}{z} = \frac{(b + u_r) - u_l}{z - f} \ ,$$

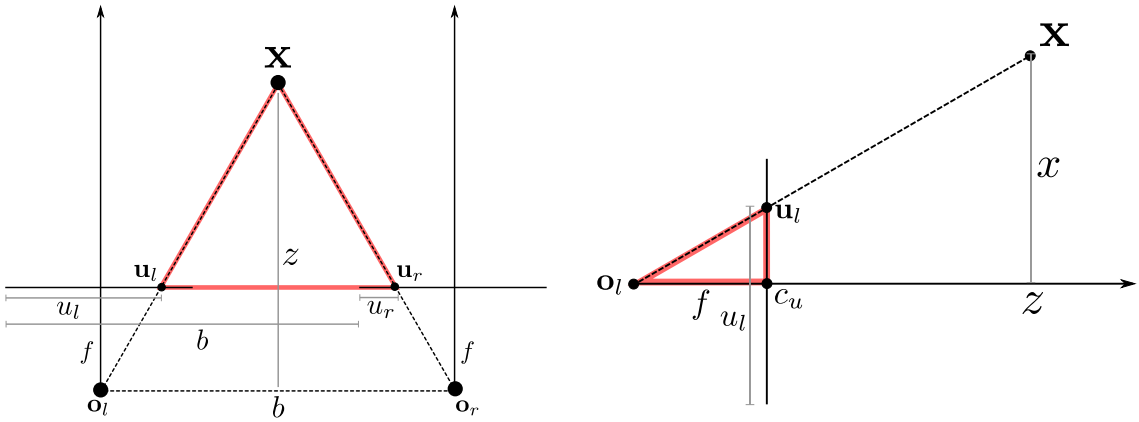


Figura 3.2.5: Triangulación estéreo.

por lo tanto:

$$d = u_l - u_r = \frac{bf}{z} \,, \tag{3.2.1}$$

donde d es la disparidad, definida como la distancia existente entre las proyecciones de las diferentes cámaras.

De la misma manera, utilizando propiedad de triángulos semejantes nuevamente sobre la Figura 3.2.5b, se obtienen las siguientes ecuaciones:

$$\frac{x}{z} = \frac{u_l - c_u}{f} \qquad \frac{y}{z} = \frac{v_l - c_v}{f} \,. \tag{3.2.2}$$

Finalmente, de las ecuaciones (3.2.1) y (3.2.2) podemos obtener las expresiones para x, y y z, dadas por:

$$\begin{aligned} x &= \frac{(u_l - c_u)z}{f} \,, \\ y &= \frac{(v_l - c_v)z}{f} \,, \\ z &= \frac{bf}{u_l - u_r} \,. \end{aligned} \tag{3.2.3}$$

Es interesante notar en la Ecuación (3.2.1) que la profundidad z es inversamente proporcional a la disparidad d. Cuando el valor de disparidad d es cercano a 0, pequeñas diferencias de disparidad producen un gran cambio en la profundidad del punto. En consecuencia, la reconstrucción 3D del ambiente mediante cámaras estéreo es más precisa para puntos cercanos a la cámara.

De manera compacta, es posible definir la matriz de re-proyección $\mathbf{Q}$ que transforma vectores de la forma $\begin{bmatrix} u_l & v_l & d & 1 \end{bmatrix}^\top$ en puntos 3D expresados en coordenadas homogéneas:

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \mathbf{Q} \begin{bmatrix} u_l \\ v_l \\ d \\ 1 \end{bmatrix},$$

donde

$$\mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 & -c_{u_l} \\ 0 & 1 & 0 & -c_{v_l} \\ 0 & 0 & 0 & f \\ 0 & 0 & \frac{1}{b} & 0 \end{bmatrix}. \tag{3.2.4}$$

En el presente capítulo se presentaron los fundamentos básicos de visión y robótica utilizados a lo largo de toda la tesina. A continuación se detallará el método de SLAM denso propuesto, utilizando los conceptos matemáticos y geométricos desarrollados.

Capítulo 4

Método propuesto de mapeo denso basado en disparidad estéreo

En este capítulo se expone en detalle la solución propuesta en esta tesina para el problema de mapeo denso basado en el uso de información de disparidad. El método propuesto, denominado como S-PTAM Denso, utiliza la localización estimada por el sistema de SLAM disperso y la información de disparidad, obtenida desde el par de imágenes estéreo, para realizar una reconstrucción 3D completa del entorno. Asímismo, se encarga de mejorar sucesivamente el mapa mediante la fusión de nubes de puntos 3D computadas a lo largo de la trayectoria de la cámara.

El sistema fue concebido para ejecutarse en paralelo con S-PTAM [6], por lo que fue diseñado tomando en consideración la interfaz y el funcionamiento del mismo. Sin embargo, no se restringe a ser ejecutado sobre este sistema de SLAM en particular. El módulo de densificación se implementó en un nodo ROS independiente, facilitando su reutilización e integración con diferentes implementaciones de SLAM de la comunidad robótica. En los siguientes subcapítulos, describimos en detalle los distintos componentes que forman parte de S-PTAM Denso.

4.1. S-PTAM

En [6] se presenta un sistema SLAM basado en *features* llamado S-PTAM (*Stereo Parallel Tracking and Mapping*). Utilizando una cámara estéreo como sensor principal, construye y mantiene un mapa disperso del entorno para obtener una localización precisa de la cámara. S-PTAM sigue el enfoque propuesto en PTAM [30] (*Parallel Tracking and Mapping*) que divide las tareas de localización y mapeo en dos hilos de ejecución diferentes —localización, y construcción del mapa (*tracking and mapping*)—, para aprovechar la capacidad de cómputo de procesadores con dos o más núcleos.

El sistema comienza con la cámara en el centro del sistema de referencia del mundo y un mapa vacío, que es inicializado inmediatamente mediante la triangulación de características en el primer par de imágenes estéreo. Sucesivamente, el hilo de *tracking* estima la posición actual para cada nuevo par de imágenes estéreo, minimizando el error de re-proyección entre los puntos del mapa proyectados y sus correspondencias en la imagen. Una vez que la posición actual de la cámara es estimada, el par estéreo puede ser seleccionado y agregado al mapa, en cuyo caso es denominado como *keyframe*. Al agregar el *keyframe* al mapa, aquellos features que no fueron asociados con puntos en el mapa son triangulados desde el par de imágenes estéreo.

Tanto la posición como los nuevos puntos son insertados al grafo de puntos y posiciones, mantenido por S-PTAM.

En paralelo al hilo de *tracking*, el hilo de *mapping* busca activamente observaciones de los puntos del mapa existentes en los *keyframes* con el objetivo de fortalecer las restricciones presentes en el grafo. Inmediatamente, el hilo de *mapping* realiza un *Local Bundle Adjustment* (LBA) sobre el subgrafo de puntos y posiciones definido por los últimos *keyframes* agregados. Para una mejor comprensión del sistema, en [6] puede encontrarse una descripción exhaustiva de S-PTAM. La Figura 4.1.1 muestra un escenario de ejemplo, donde se utiliza S-PTAM para la estimación de la trayectoria.

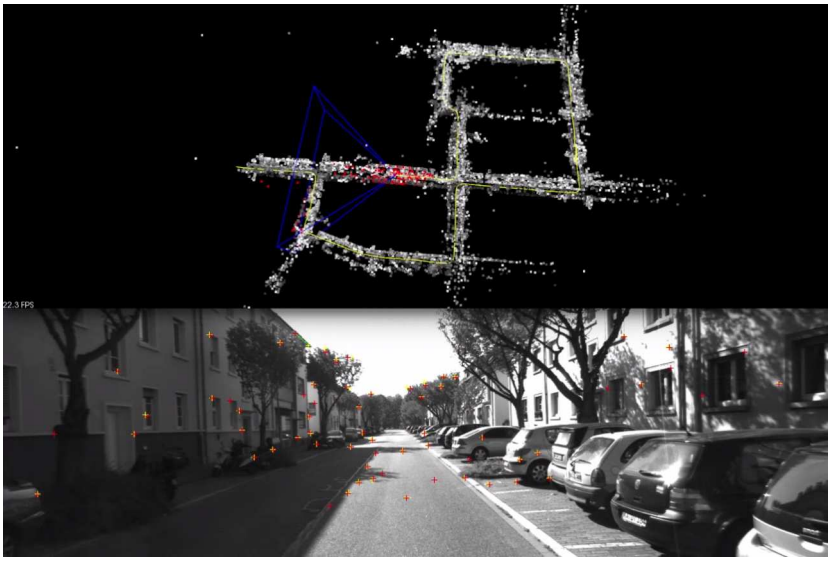


Figura 4.1.1: Reconstrucción dispersa obtenida por S-PTAM a partir de una secuencia de imágenes del recorrido de un vehículo. Arriba: mapa disperso 3D del entorno (*landmarks*), trayectoria estimada (línea amarilla). Abajo: imagen izquierda correspondiente al *keyframe* actual.

4.2. Esquema general

En la Figura 4.2.1 se muestra la estructura general del módulo de reconstrucción 3D densa. Los *keyframes* generados por el sistema S-PTAM son encolados para ser procesados secuencialmente por el hilo de *Cómputo de Disparidad*. Para cada par de imágenes en la cola, se genera un mapa de disparidad que, a su vez, es encolado para su tratamiento en el hilo de *Expansión y fusión de mapa*. El hilo de *Expansión y fusión de mapa* usa la posición del *keyframe* estimada por S-PTAM y el mapa de disparidad asociado para extender y ajustar el mapa. Los puntos existentes en el mapa 3D denso global se proyectan sobre los planos de imagen estéreo para buscar correspondencias con los puntos cercanos en la imagen. Si tal correspondencia existe, la posición del punto 3D se ajusta en base a su valor en el mapa de disparidad actual. Finalmente, aquellos píxeles que no poseen correspondencias son utilizados para triangular nuevos puntos y extender el mapa. Para mantener el mapa denso consistente a la trayectoria estimada por S-PTAM, el hilo de *Refinamiento de mapa* (*Mapping*) ajusta

constantemente la posición de los *keyframes* y las nubes de puntos asociadas cada vez que S-PTAM refina una posición.

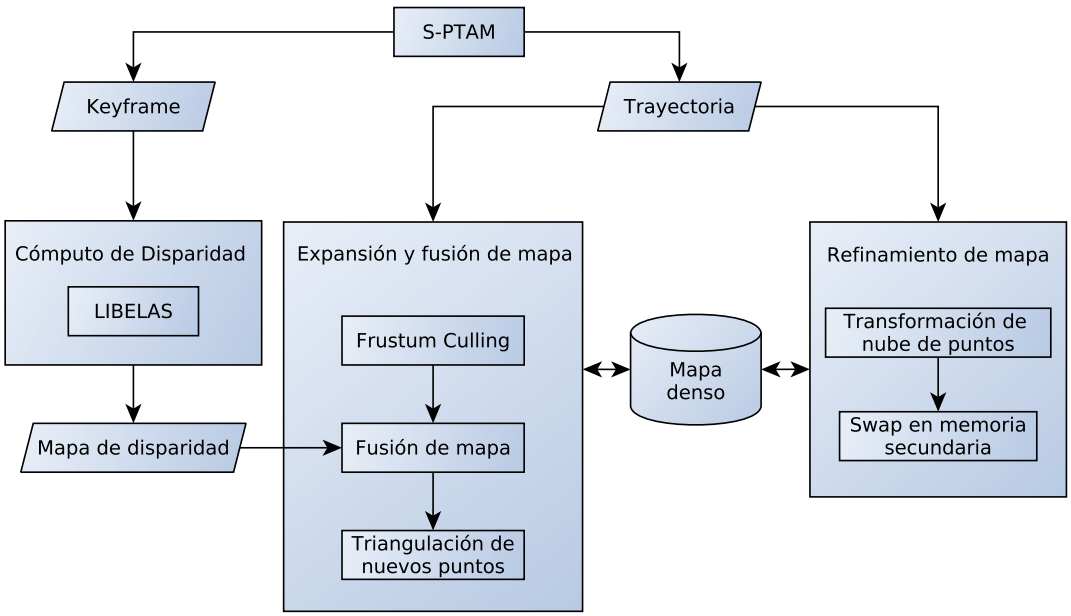
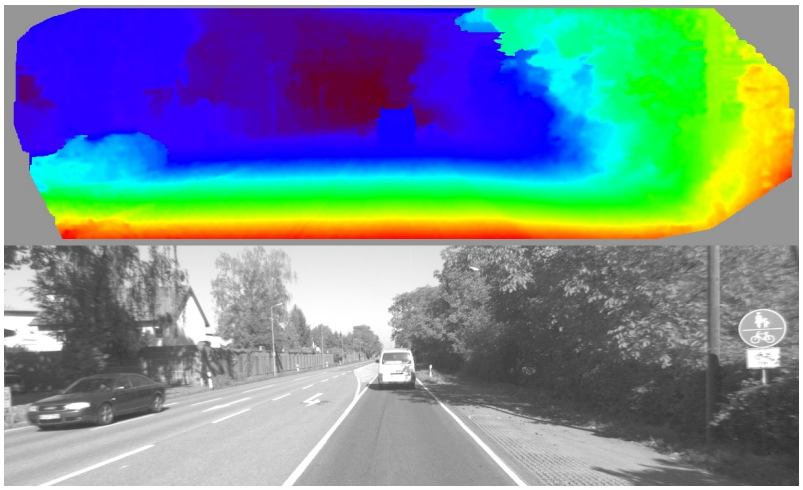


Figura 4.2.1: Estructura general del módulo de densificación 3D densa en tiempo real.

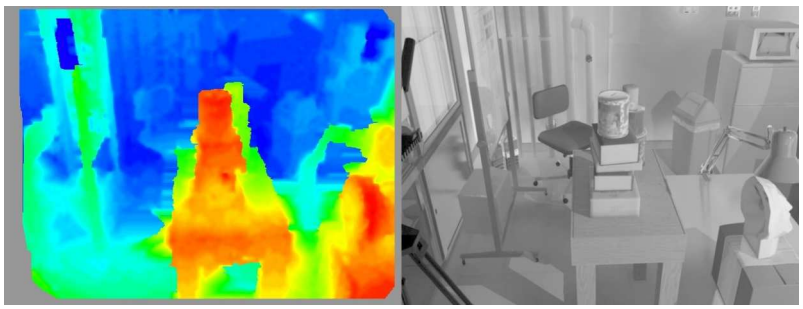
4.3. Cómputo de Disparidad

Como se analizó en las secciones 2.4.1 y 3.2, es posible procesar un par de imágenes estéreo mediante la búsqueda de correspondencias y computar una mapa denso de disparidad. Un *mapa de disparidad* se define como una función $\mathcal{D} : \mathbb{R}^2 \rightarrow \mathbb{R}^+$ que mapea los píxeles en el plano de la imagen a su correspondiente valor de disparidad. Denominaremos $\mathcal{D}_j$ a la función que representa el mapa de disparidad generado a partir del par de imágenes estéreo del *keyframe* $\mathbf{K}_j$.

Cada vez que un nuevo par de imágenes estéreo es seleccionado por S-PTAM como *keyframe*, el hilo de *Cómputo de Disparidad* utiliza la librería LIBELAS para generar un mapa de disparidad. LIBELAS (*Library for Efficient Large-scale Stereo Matching*) [14] es una librería multiplataforma de código abierto para computar mapas de disparidad a partir de pares de imágenes estéreo rectificadas de alta resolución. El método empleado se basa en que, pese al hecho de que muchas correspondencias estéreo son altamente ambiguas, algunas de ellas pueden ser robustamente matcheadas. Asumiendo variaciones suaves en la disparidad es posible sectorizar la imagen, mediante la triangulación de Delaunay [77], sobre estos puntos de soporte para la estimación de las restantes disparidades. En las Figuras 4.3.1a y 4.3.1b pueden observarse resultados de su uso.



(a) Mapa de disparidad obtenido mediente LIBELAS para una imagen izquierda del Dataset KITTI [70].



(b) Mapa de disparidad obtenido mediente LIBELAS para una imagen izquierda del Dataset Tsukuba [78].

Figura 4.3.1: Imágenes pertenecientes a diferentes datasets, junto a sus correspondientes mapas de disparidad (rojo: mayor disparidad, azul: menor disparidad) computados utilizando la librería LIBELAS.

4.4. Expansión y fusión de mapa

Luego que el hilo de *Cómputo de Disparidad* computa el mapa de disparidad del *keyframe* actual $\mathbf{K}_j$, el hilo de *Expansión y fusión de mapa* utiliza este mapa y la posición correspondiente a $\mathbf{K}_j$ para mejorar y agregar nuevos puntos a la reconstrucción 3D densa. En base al mapa de disparidad $\mathcal{D}_j$ es posible calcular un mapa de profundidad inversa según las ecuaciones de geometría estéreo. La profundidad inversa ρ_i para cada píxel i está dada por

$$\rho_i = \frac{\mathcal{D}_j}{bf}, \tag{4.4.1}$$

siendo f la distancia focal y b el baseline del par de imágenes estéreo rectificadas.

Podemos estimar una nube de puntos densa $\mathcal{P} = \{\mathbf{x}_1, \ldots, \mathbf{x}_P\}$ acumulando las reconstrucciones densas para los últimos J *keyframes* estéreo $\{\mathcal{P}_{j-J}, \ldots, \mathcal{P}_j\}$. Sobre esta nube de puntos $\mathcal{P}$, se aplica *Frustum culling* para obtener solo aquellos puntos visibles desde $\mathbf{K}_j$, como se explicó en la sección 3.1.3. Denominamos a la nube de puntos resultante *mapa local* al *keyframe* $\mathbf{K}_j$.

A medida que la cámara se traslada, es posible que ciertas áreas del mapa local $\mathcal{P}$ se encuentren duplicadas. Asímismo, la reconstrucción ha de ser más precisa desde cierto puntos de vista cercanos a la escena. Proponemos un algoritmo de fusión eficiente para estas áreas duplicadas. Dado dos puntos $\mathbf{x}_{previo} \in \mathcal{P}$ y $\mathbf{x}_{actual} \in \mathcal{P}_j$ de la nube de puntos local y del *keyframe* actual respectivamente, consideramos que corresponden al mismo punto 3D si:

- Sus proyecciones en *keyframes* estéreo cercanos coinciden en las mismas coordenadas del píxel en la imagen.
- La distancia euclideana entre sus coordenadas 3D se encuentra por debajo de un cierto *threshold*.

La Figura 4.4.1 ilustra la fusión de un par de puntos 3D correspondientes $\mathbf{x}_{previo}$ y $\mathbf{x}_{actual}$. El resultado de la fusión es

$$\mathbf{x}_{fusión} = \frac{1}{\rho_{fusión}} \frac{\mathbf{x}_{actual}}{\|\mathbf{x}_{actual}\|}, \tag{4.4.2}$$

donde $\rho_{fusión}$ es la profundidad inversa promedio

$$\rho_{fusión} = \frac{\rho_{actual} + \rho_{previo}}{2}, \tag{4.4.3}$$

y ρ_{actual} y ρ_{previo} son las profundidades inversas de $\mathbf{x}_{actual}$ y $\mathbf{x}_{previo}$ respectivamente.

Nótese en la Ecuación 4.4.1 que la profundidad inversa tiene una relación lineal con la disparidad. Es posible expresar la desviación estándar de la profundidad inversa con respecto a la disparidad como

$$\sigma_{\rho_i} = \frac{\partial \rho_i}{\partial \mathcal{D}_i} \sigma_{\mathcal{D}_i} = \frac{1}{bf} \sigma_{\mathcal{D}_i}. \tag{4.4.4}$$

Asumiendo que la variación estándar en la disparidad es constante, la Ecuación 4.4.4 expresa que la incertidumbre en la profundidad inversa también resulta constante.

En este algoritmo de fusión, estamos asumiendo que la profundidad inversa de un punto es similar desde dos puntos de vista diferentes. Esta es una suposición razonable mientras los *keyframes* se encuentren cercanos entre sí, lo cual se cumple en nuestro caso, dado que incrementalmente fusionamos el *mapa local* al *keyframe* $\mathbf{K}_j$ con la nube de puntos $\mathcal{P}_j$ generada desde el mismo.

Siendo $\rho_i^{c_j}$ y $n_i^{c_j}$ la profundidad inversa y el rayo de proyección del punto i en el sistema de coordenadas de la cámara c_j y $\mathbf{t}_{c_1 c_2}$ y $\mathbf{R}_{c_1 c_2}$ el vector de traslación y matriz de rotación entre la cámara c_1 y c_2. La siguiente igualdad se cumple:

$$\frac{1}{\rho_1^{c_1}} n_1^{c_1} = \mathbf{t}_{c_1 c_2} + \mathbf{R}_{c_1 c_2} \frac{1}{\rho_1^{c_2}} n_1^{c_2}. \tag{4.4.5}$$

Aplicando la norma euclideana lado a lado en la Ecuación 4.4.5 y asumiendo que los *keyframes* se encuentran cerca $\mathbf{t}_{c_1 c_2} \approx 0$, vemos que

$$
\begin{aligned}
\left\| \frac{1}{\rho_1^{c_1}} n_1^{c_1} \right\| &= \left\| \mathbf{t}_{c_1 c_2} + \mathbf{R}_{c_1 c_2} \frac{1}{\rho_1^{c_2}} n_1^{c_2} \right\| && \text{despreciando el término } \mathbf{t}_{c_1 c_2} \approx 0 \\
\left\| \frac{1}{\rho_1^{c_1}} n_1^{c_1} \right\| &= \left\| \mathbf{R}_{c_1 c_2} \frac{1}{\rho_1^{c_2}} n_1^{c_2} \right\| && \text{dado que } \mathbf{R}_{c_1 c_2} \text{ es ortonormal} \\
\left\| \frac{1}{\rho_1^{c_1}} n_1^{c_1} \right\| &= \left\| \frac{1}{\rho_1^{c_2}} n_1^{c_2} \right\| \\
\frac{1}{\rho_1^{c_1}} \left\| n_1^{c_1} \right\| &= \frac{1}{\rho_1^{c_2}} \left\| n_1^{c_2} \right\| && \text{dado que } n_1^{c_1} \text{ y } n_1^{c_1} \text{ son versores} \\
\frac{1}{\rho_2^{c_1}} &= \frac{1}{\rho_1^{c_2}} \\
\rho_2^{c_1} &= \rho_1^{c_2}.
\end{aligned}
$$

Finalmente, este hilo expande el mapa utilizando los restantes píxeles que no encontraron correspondencias y que poseen un valor de disparidad válido. El método distingue dos casos: píxeles sin correspondencia en el mapa, y píxeles que no satisfacen la restricción de distancia euclideana. En el primer caso, el hilo simplemente triangula un nuevo punto según la disparidad del píxel. En el último caso, se triangula un nuevo punto si el punto actual $\mathbf{x}_{actual}$ se encuentra más cercano a la cámara que el punto existente $\mathbf{x}_{previo}$, indicando que pertenecen a diferentes objetos.

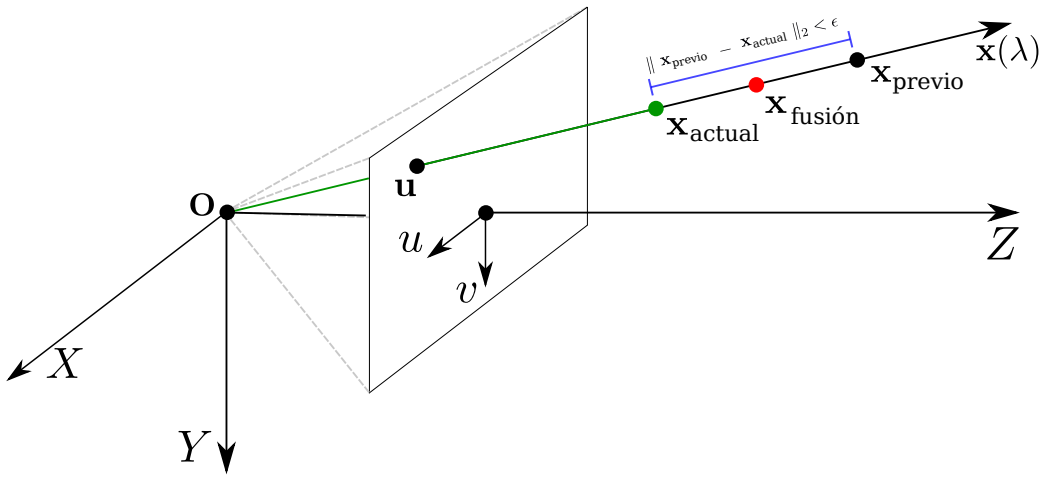


Figura 4.4.1: Fusión de un par de puntos correspondientes. $\mathbf{x}_{actual}$, $\mathbf{x}_{previo}$ y $\mathbf{x}_{fusión}$ son el punto triangulado desde el mapa de disparidad actual, el punto existente y el nuevo punto estimado (resultado de la fusión), respectivamente. En la imagen también se muestra el criterio de fusión donde la distancia euclideana entre las coordenadas 3D se encuentra por debajo de un cierto *threshold*.

4.5. Refinamiento de mapa

El sistema propuesto es capaz de actualizar el mapa denso 3D al recibir actualizaciones de posiciones, cuando los *keyframes* asociados ya hayan sido procesados. Notar que este compor-

tamiento es esperable en sistemas de SLAM, ya que el mapa disperso y la trayectoria de la cámara son refinados a lo largo de toda la secuencia. En nuestro caso, cada vez que S-PTAM ajusta un *keyframe* en la trayectoria, el hilo de *Refinamiento* ajusta el mapa. Esto ayuda a mantener una reconstrucción más precisa.

Cabe recordar que las nubes de puntos son almacenadas en el sistema de coordenadas del mundo. Por lo tanto, dada la nube de puntos $\mathcal{P}_i$, generada desde el *keyframe* $\mathbf{K}_i$ utilizando la posición de la cámara c_i, notamos la transformación existente en el mundo y esta cámara como:

$$\mathbf{E}^{c_i \mathrm{w}} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \quad .$$

De la misma manera, siendo $\hat{\mathrm{c}}_i$ la posición actualizada de la cámara en cuestión, notamos:

$$\mathbf{E}^{\hat{\mathrm{c}}_i \mathrm{w}} = \begin{bmatrix} \hat{\mathbf{R}} & \hat{\mathbf{t}} \\ \mathbf{0} & 1 \end{bmatrix} \quad .$$

Luego, la actualización de la nube de puntos $\mathcal{P}_i$ está dada por la siguiente matriz de transformación $\mathbf{E}_{ref}$:

$$\begin{aligned} \mathbf{E}_{ref} &= \mathbf{E}^{\mathrm{w}\hat{\mathrm{c}}_i} \mathbf{E}^{\mathrm{c}_i \mathrm{w}} \\ &= (\mathbf{E}^{\hat{\mathrm{c}}_i \mathrm{w}})^{-1} \mathbf{E}^{\mathrm{c}_i \mathrm{w}} \\ &= \begin{bmatrix} \hat{\mathbf{R}} & \hat{\mathbf{t}} \\ \mathbf{0} & 1 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \\ &= \begin{bmatrix} \hat{\mathbf{R}}^\top & -\hat{\mathbf{R}}^\top \hat{\mathbf{t}} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \\ &= \begin{bmatrix} \hat{\mathbf{R}}^\top \mathbf{R} & \hat{\mathbf{R}}^\top (\mathbf{t} - \hat{\mathbf{t}}) \\ \mathbf{0} & 1 \end{bmatrix} . \end{aligned}$$

Por otro lado, el hilo de *Refinamiento de mapa* se encarga permanentemente de mover nubes de puntos no utilizadas a memoria secundaria, permitiendo ejecutar el sistema en secuencias de extensa longitud, reconstruyendo millones de puntos 3D. Dado que el hilo de *Expansión y fusión de mapa* utiliza las nubes de puntos recientes, estas se conservan en memoria RAM hasta que salen de dicho *mapa local*.

4.6. Detalles de implementación

El sistema de mapeo denso se implementó como un nodo del framework de código abierto ROS [79], con el objetivo de promover su uso dentro de la comunidad de investigación en robótica. El código fuente del sistema de mapeo denso propuesto en esta tesina se encuentra disponible públicamente[1] bajo licencia GPLv3.

Notar que el sistema está íntegramente desacoplado de S-PTAM, generalidad que permite fácilmente re-utilizar el nodo de densificación con otros métodos de SLAM dispersos que presenten una interfaz similar a S-PTAM. La biblioteca libre de visión por computadora

[1] https://github.com/CIFASIS/dense-sptam

OpenCV [80] fue utilizada para el manejo y codificación de imágenes y la librería de código abierto *Point Cloud Library* (PCL) [81] se empleó en el manejo de nubes de puntos.

La implementación del sistema de mapeo denso se constituye de dos hilos principales que realizan las tareas de *Cómputo de disparidad* (sección 4.3) *y Expansión y fusión de mapa* (sección 4.4). Un tercer hilo, con menor prioridad, se encarga del *Refinamiento de mapa* (sección 4.5). Nótese que los hilos de *Cómputo de disparidad* y *Expansión y fusión de mapa* son *CPU-bound* (requieren uso intensivo de CPU) y, en cambio, el hilo de *Refinamiento de mapa* es *I/O-bound* (limitado por operaciones de entrada-salida).

Capítulo 5

Experimentación y resultados

En este capítulo se presentan los experimentos diseñados para evaluar el desempeño y precisión de la solución propuesta para el problema de reconstrucción densa basada en disparidad. En todos los casos, el hardware utilizado para el procesamiento de los experimentos corresponde a una computadora estándar de escritorio con procesador Intel Core i7 de 2.6 GHz y 16 GB de memoria RAM.

Los experimentos se llevan a cabo sobre el entorno de desarrollo ROS (*Robot Operating System*) Kinetic, el cual provee las herramientas necesarias para comprobar el desempeño del sistema en distintos tipos de escenarios. El método propuesto se implementó como un nodo ROS que es ejecutado en simultáneo al nodo ROS correspondiente al sistema S-PTAM.

5.1. Datasets

Para la experimentación y evaluación se utilizan colecciones de datos de dominio público, denominados *datasets*. Los datos recolectados en estas colecciones corresponden a mediciones efectuadas mediante diferentes sensores a lo largo de un trayecto realizado por robots móviles o vehículos. ROS posibilita la confección y posterior reproducción de estos *datasets*, permitiendo evaluar el desempeño del sistema en diferentes condiciones.

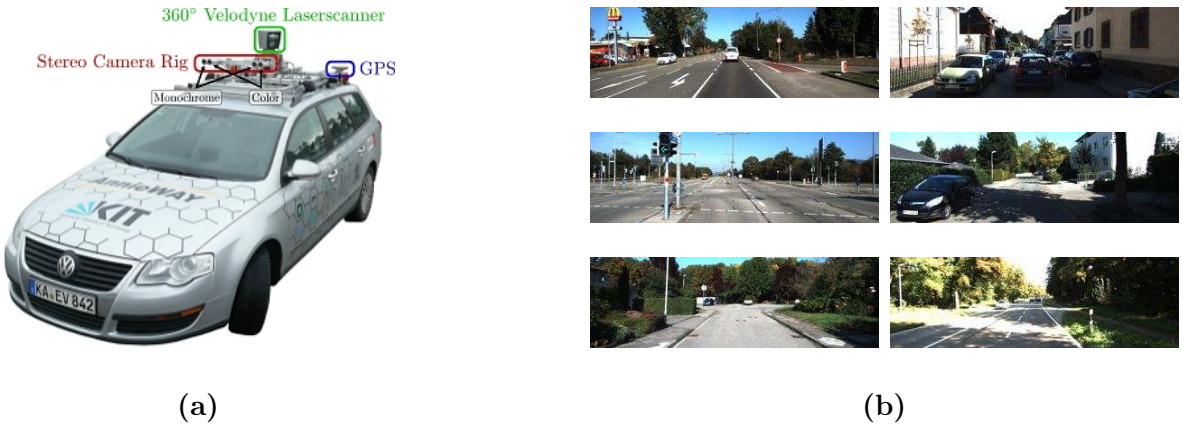
Evaluamos nuestro sistema S-PTAM Denso sobre los datasets públicos KITTI [70] y Tsukuba [78].

5.1.1. Dataset KITTI

KITTI Vision Benchmark Suite [70] es un *dataset* compuesto por 23 secuencias capturadas por un vehículo transitando un entorno urbano por las calles de la ciudad de Karlsruhe, Alemania. La longitud promedio de cada secuencia está en el orden de varios kilómetros.

El vehículo posee una cámara estéreo montada en su frente, con un *baseline* de ∼60cm y una resolución de 1344 × 391 píxeles, que corre a una frecuencia de 10 *frames* por segundo (10Hz). En la Fig. 5.1.1 se muestran los sensores montados en el vehículo utilizado en el dataset KITTI y un conjunto de imágenes de ejemplo seleccionadas de distintas secuencias.

Para la evaluación del método propuesto se utilizaron las primeras 11 secuencias del *dataset* (secuencias 00 a 10) dado que, además de las imágenes estéreo, proporcionan información del *ground truth*, con la localización precisa del vehículo en cada momento y nubes de puntos 3D tomadas con un láser Velodyne.



(a) **(b)**

Figura 5.1.1: **(a)** Sensores utilizados en el dataset KITTI. **(b)** Imágenes pertenecientes a secuencias del dataset KITTI.

5.1.2. Dataset Tsukuba

Tsukuba Stereo [78] es un *dataset* generado completamente por computadora que consiste en una cámara estéreo recorriendo una trayectoria por un entorno interior. El dataset consta de una secuencia de video de 1 minuto, formada por 1800 pares de imágenes estéreo, a una frecuencia de 30 *frames* por segundo (30Hz). La secuencia es capturada por una cámara estéreo con un *baseline* de ∼10cm y una resolución de 640 x 480 píxeles.

Al tratarse de una dataset artificial, provee información exacta del *ground truth*: posición 3D y orientación de la cámara y mapas de disparidad para cada *frame*. Combinando esta información es posible reconstruir el entorno 3D a medida que la cámara recorre la trayectoria, determinando con exactitud la distancia entre el lente de la cámara y cualquier objeto o punto 3D en la escena. En la Fig. 5.1.2 se muestran algunas imágenes de ejemplo seleccionadas del dataset, con sus respectivos mapas de disparidad.

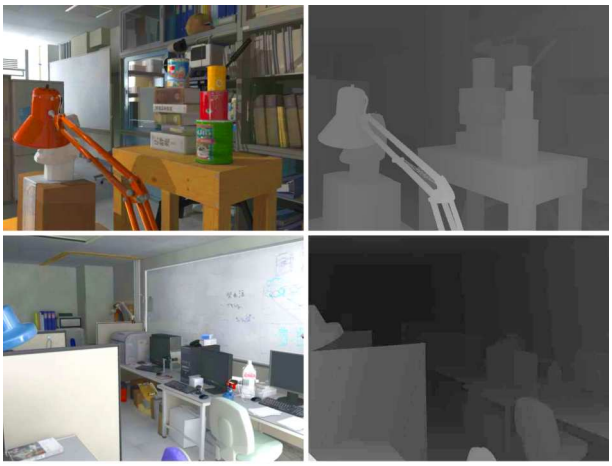


Figura 5.1.2: Imágenes pertenecientes al dataset Tsukuba con sus respectivos mapas de disparidad (blanco: mayor disparidad, negro: menor disparidad).

5.2. Evaluación

La precisión de la reconstrucción 3D obtenida se evaluó mediante la comparación de *mapas de profundidad* generados por el método propuesto y en base a información de *ground truth*. Un *mapa de profundidad* es una imagen donde a cada píxel se le asigna el valor determinado por la profundidad (distancia en el eje Z) del punto 3D proyectado. En caso que un píxel no corresponda a la proyección de ningún punto del entorno 3D, se considera que el píxel tiene un valor de profundidad inválido.

Los mapas de profundidad asociados al mismo *keyframe* son comparados píxel a píxel, calculando el error absoluto existente en la profundidad, omitiendo aquellos píxeles que no posean un profundidad válida. En el dataset Tsukuba, los autores proveen los mapas de disparidad del *ground truth* para cada par de imágenes estéreo, desde los que extrajimos los mapas de profundidad. Para el caso del dataset KITTI, generamos los mapas de profundidad del *ground truth* para cada par de imágenes proyectando las nubes de puntos del láser Velodyne en cada plano de imagen, utilizando las correspondientes posiciones del *ground truth*.

Por otro lado, utilizamos los mapas de profundidad generados a partir de la librería de disparidad LIBELAS [13], sobre distintos pares de imágenes estéreo, para mostrar los beneficios de nuestro algoritmo de fusión.

5.3. Resultados

Las Figuras 5.3.1 y 5.3.2 muestran distintas reconstrucciones tridimensionales producidas por S-PTAM Denso en ambos datasets, donde puede notarse la alta precisión del método.

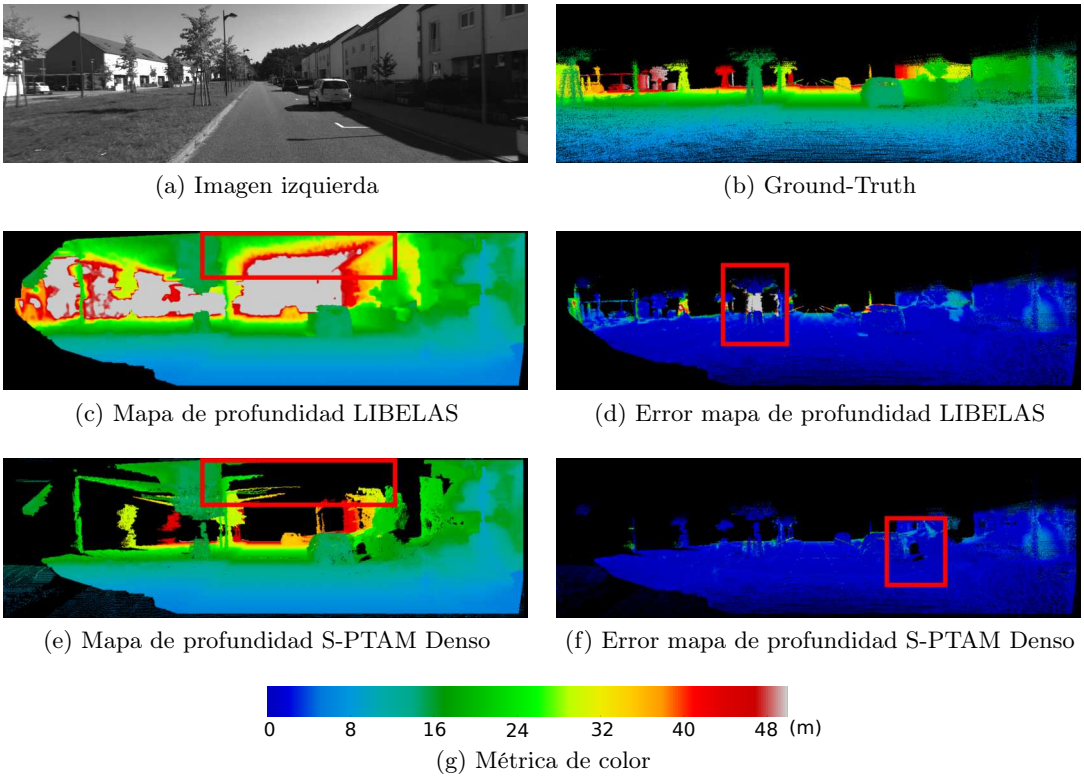
Figura 5.3.1: Distintas reconstrucciones 3D generadas mediante nuestro algoritmo en el dataset KITTI (secuencia 06).

Figura 5.3.2: Distintas reconstrucciones 3D generadas mediante nuestro algoritmo en el dataset sintético Tsukuba.

Las Figuras 5.3.3 y 5.3.4 muestran una comparación entre los mapas de profundidad estimados por LIBELAS, a partir de un par de imágenes estéreo, y el mapa de profundidad calculado por nuestro algoritmo. Además mostramos los errores en la reconstrucción.

Obsérvese que, para la secuencia KITTI, la profundidad obtenida por S-PTAM Denso 5.3.3e se aproxima mejor al *ground truth* 5.3.3b que el obtenido por LIBELAS 5.3.3c. Esto también puede ser apreciado en las figuras de error 5.3.3d y 5.3.3f. Esta notable mejora en la precisión es principalmente el resultado de la fusión desde diferentes puntos de vista realizada por el método propuesto. La imágenes del dataset KITTI contienen áreas distantes, que resultan en un mayor error estéreo. Para tales áreas el error puede ser reducido si la profundidad estéreo es fusionada con una observación más cercana.

En la Figura 5.3.3c puede observarse el área del cielo remarcada, a la que LIBELAS asigna valores de disparidad incorrectos —el cielo se considera que tiene un valor de disparidad nulo, equivalente a una profundidad infinita—. El método propuesto mejora la reconstrucción descartando estos puntos como puede observarse en la Figura 5.3.3e. Otra mejora puede observarse en las Figura 5.3.3d donde la región remarcada presenta error alto en LIBELAS, a diferencia del método propuesto. Como contraparte, el mapa de profundidad de S-PTAM Denso presenta un mayor número de *huecos* —píxeles sin valor de profundidad— como se observa en la Figura 5.3.3f debido a que estos puntos no pudieron ser validados.



(a) Imagen izquierda

(b) Ground-Truth

(c) Mapa de profundidad LIBELAS

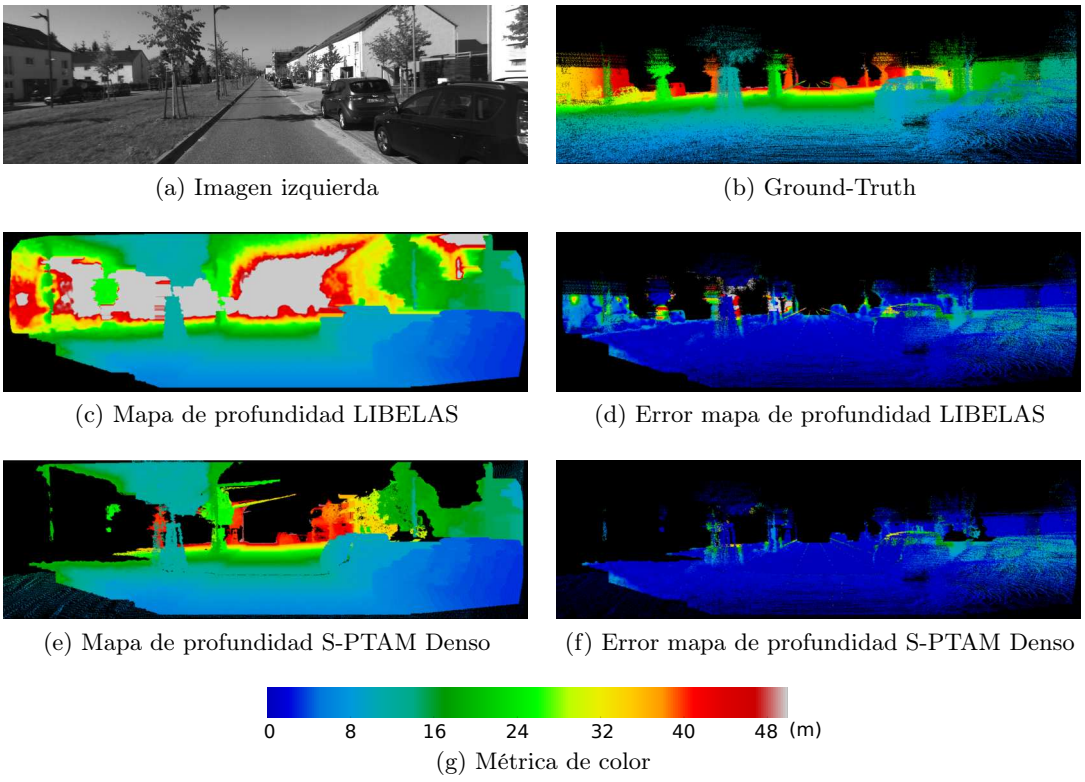
(d) Error mapa de profundidad LIBELAS

(e) Mapa de profundidad S-PTAM Denso

(f) Error mapa de profundidad S-PTAM Denso

(g) Métrica de color

Figura 5.3.3: Comparación entre mapas de profundidad obtenidos mediante LIBELAS y S-PTAM en relación al *ground truth*, para una imagen de la secuencia 06 del Dataset KITTI.



(a) Imagen izquierda

(b) Ground-Truth

(c) Mapa de profundidad LIBELAS

(d) Error mapa de profundidad LIBELAS

(e) Mapa de profundidad S-PTAM Denso

(f) Error mapa de profundidad S-PTAM Denso

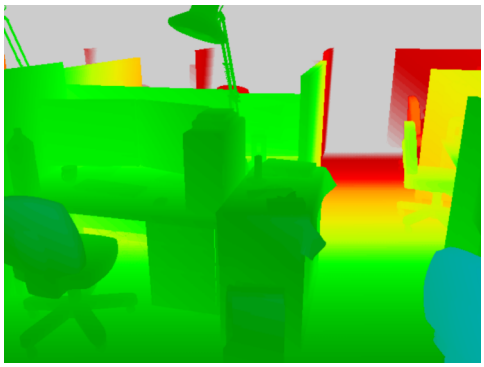
(g) Métrica de color

Figura 5.3.4: Comparación entre mapas de profundidad obtenidos mediante LIBELAS y S-PTAM en relación al *ground truth*, para un par de imágenes estéreo de la secuencia 06 del Dataset KITTI.

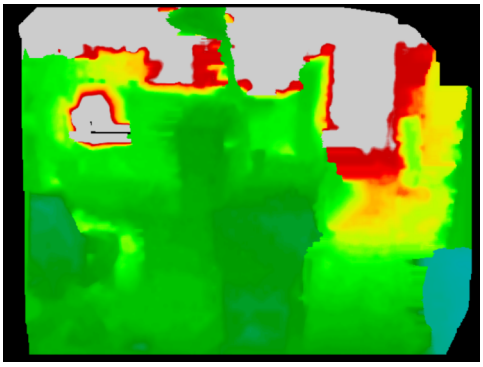
En las Figuras 5.3.5 y 5.3.6 puede observarse que los errores de LIBELAS y S-PTAM Denso son similares en el dataset Tsukuba. La razón es que la escena de interior, renderizada por el dataset Tsukuba, no contiene áreas distantes. La profundidad desde un par de imágenes estéreo es entonces suficientemente precisa, y la mejora obtenida por el algoritmo de fusión desde diferentes puntos de vista no resulta tan evidente.

Nótese que en términos generales el mapa de profundidad de S-PTAM Denso 5.3.5e presenta un mayor número de *huecos* y es menos denso que el mapa de LIBELAS 5.3.5c. Por otro lado, el mapa de profundidad de LIBELAS presenta áreas con mucho error —como la región remarcada en la Figura 5.3.5d— que son refinadas por el método propuesto, mejorando considerablemente la reconstrucción. Sin embargo, el área remarcada en la Figura 5.3.5f muestra que S-PTAM Denso descartó algunos puntos que presentaban un error relativamente bajo en el mapa generado por LIBELAS, lo cual tiene un impacto negativo en la reconstrucción.

(a) Imagen izquierda

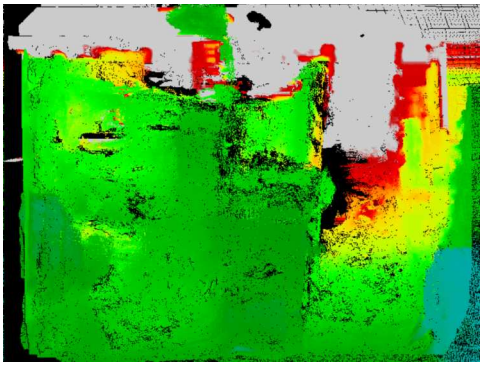


(b) Ground-Truth

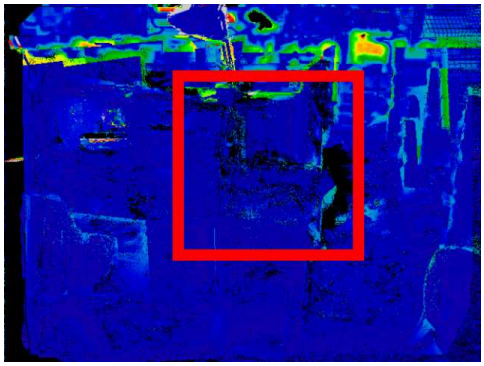


(c) Mapa de profundidad LIBELAS

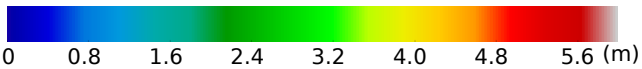
(d) Error mapa de profundidad LIBELAS



(e) Mapa de profundidad S-PTAM Denso



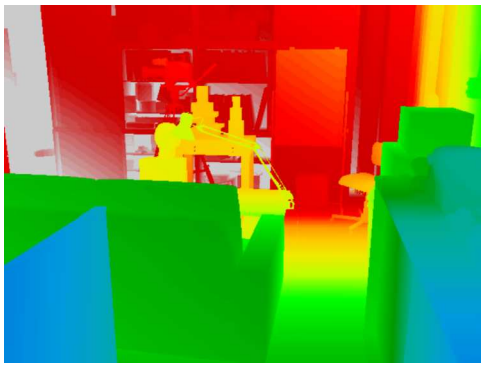
(f) Error mapa de profundidad S-PTAM Denso



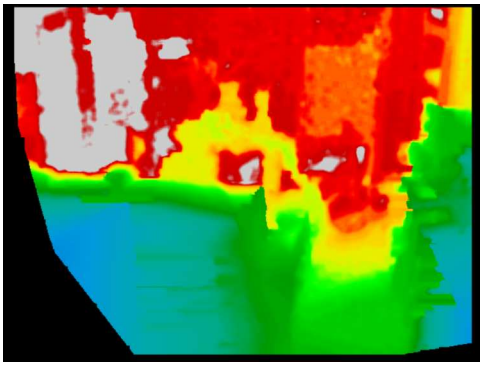
(g) Métrica de color

Figura 5.3.5: Comparación entre mapas de profundidad obtenidos mediante LIBELAS y S-PTAM en relación al *ground truth*, para una imagen del Dataset Tsukuba.

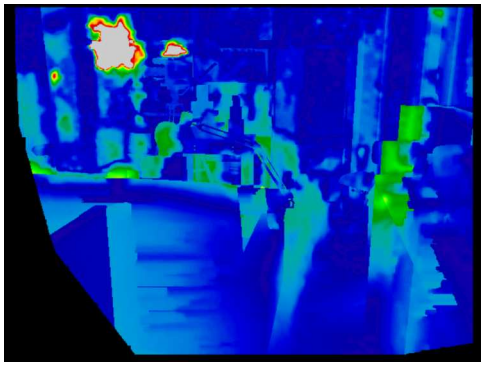
(a) Imagen izquierda



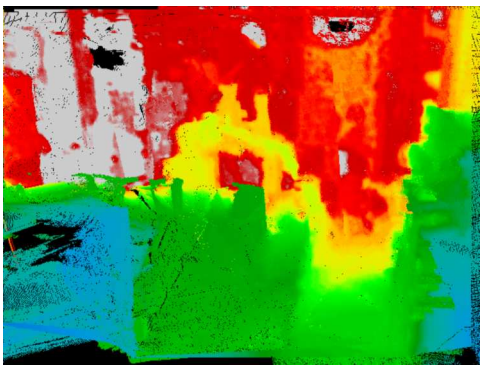
(b) Ground-Truth



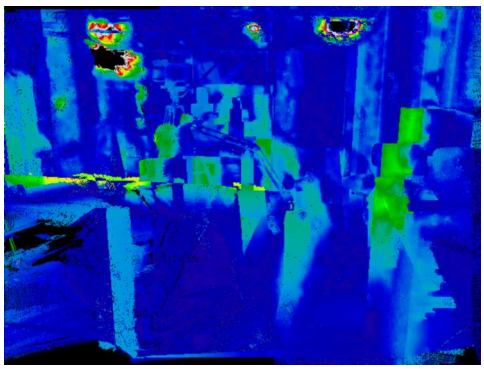
(c) Mapa de profundidad LIBELAS



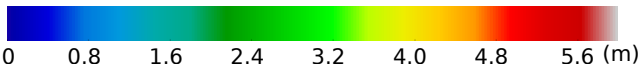
(d) Error mapa de profundidad LIBELAS



(e) Mapa de profundidad S-PTAM Denso



(f) Error mapa de profundidad S-PTAM Denso



(g) Métrica de color

Figura 5.3.6: Comparación entre mapas de profundidad obtenidos mediante LIBELAS y S-PTAM en relación al *ground truth*, para una imagen del Dataset Tsukuba.

5.3.1. Análisis del error

Para obtener un análisis cuantitativo de nuestro algoritmo, también evaluamos el error de la profundidad a lo largo de toda la secuencia 06 del dataset KITTI. Las Figuras 5.3.7 y 5.3.8 muestran los errores de la reconstrucción a distintas distancias de la cámara, en la secuencia 06 del dataset KITTI, empleando los mapas de profundidas de LIBELAS y S-PTAM Denso. Los errores se muestran a partir de los 5 metros de distancia, ya que esta es la mínima distancia del sensor Velodyne que utilizamos como *ground truth*.

Para una mejor comparación, la Figura 5.3.9 muestra el error medio para cada profundidad tanto en S-PTAM Denso como en LIBELAS. Nótese que los errores son muy similares para pequeñas profundidades. Dado que el *parallax*[1] es grande, la profundidad estimada es lo suficientemente precisa, por lo tanto las triangulaciones estéreo adicionales no agregan demasiada información. La ganancia obtenida mediante la fusión estéreo es apreciada a grandes profundidades, donde una sola triangulación estéreo produce resultados ruidosos y la fusión de múltiples vistas es capaz de reducir el error.

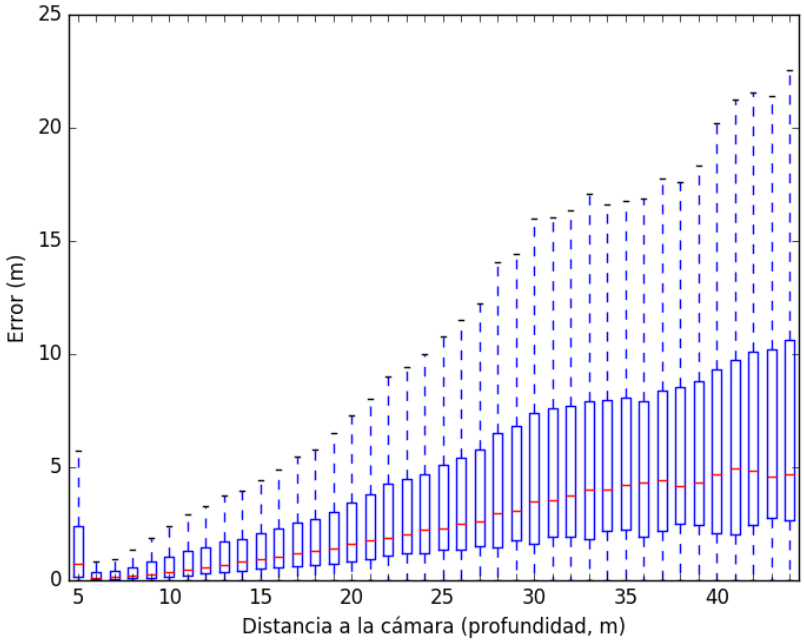


Figura 5.3.7: Error en LIBELAS (profundidad vs. error) en la secuencia 06 del Dataset KITTI.

[1] *Parallax*: desviación en la posición aparente de un objeto dependiendo del punto de vista elegido.

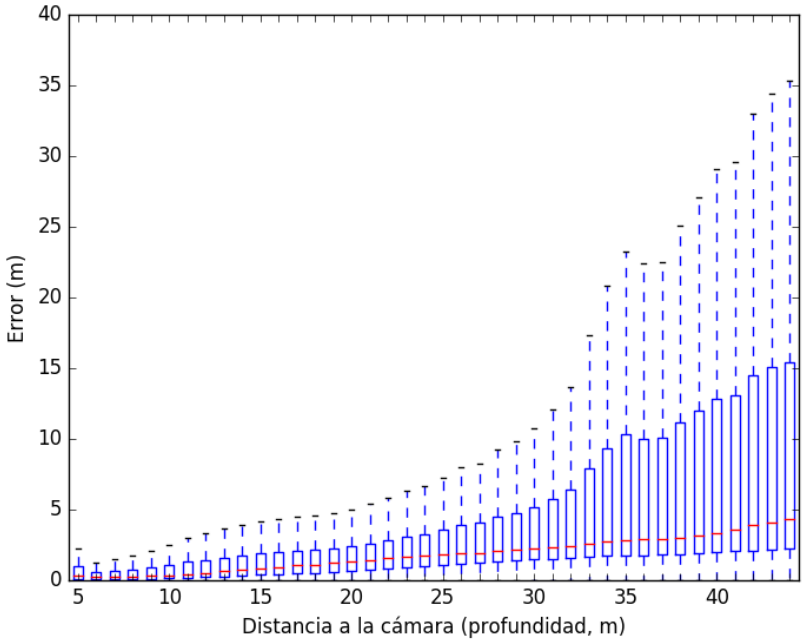


Figura 5.3.8: Error en S-PTAM Denso (profundidad vs. error) en la secuencia 06 del Dataset KITTI.

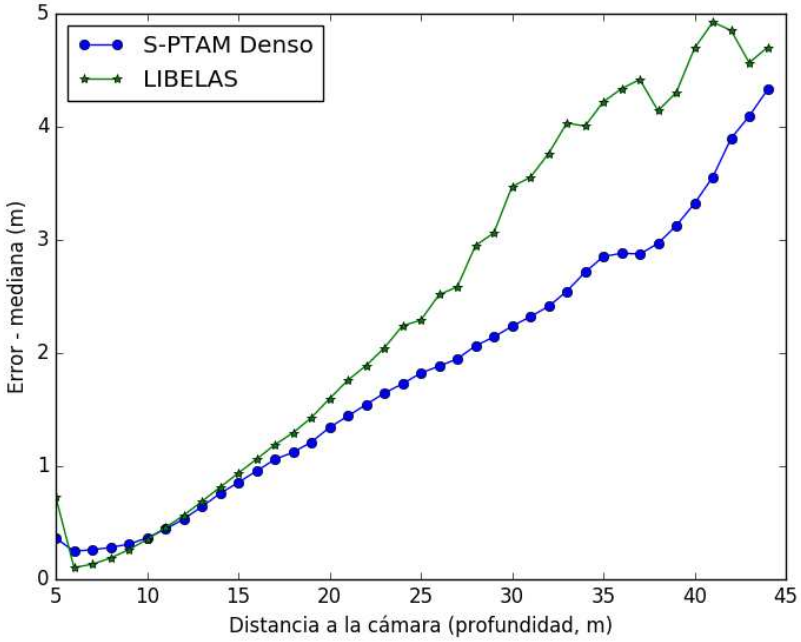


Figura 5.3.9: Comparación de mediana de error entre LIBELAS y S-PTAM Denso (profundidad vs. error) en la secuencia 06 del Dataset KITTI.

En las Figuras 5.3.10 y 5.3.11 se grafica el error en la profundidad de LIBELAS y S-PTAM Denso, respectivamente, para la secuencia del dataset Tsukuba. Al igual que en el Dataset KITTI, mostramos el error medio en la profundidad para ambos métodos en la Figura 5.3.12. En contraste con los resultados del Dataset KITTI, en este caso la precisión de S-PTAM Denso resulta en una mejora mínima frente a la obtenida por LIBELAS.

El motivo subyacente fue detallado anteriormente. Las profundidades del escenario de interior generado en el dataset Tsukuba son pequeñas (en comparación con las calles en exterior del dataset KITTI). El baseline de la cámara estéreo es lo suficientemente grande para producir estimaciones precisas de la profundidad, y por lo tanto las sucesivas fusiones no implican una mejora significativa.

Como conclusión, las triangulaciones desde múltiple pares de imágenes estéreo mejoran sustancialmente la precisión de la reconstrucción cuando el cociente de relación profundidad-baseline es alto. En caso contrario, la ganancia en el método se ve limitada.

Por completitud, en las Figuras 5.3.13 y 5.3.14 se expone un análisis comparativo de las medianas de error correspondientes a las restantes secuencias del dataset KITTI.

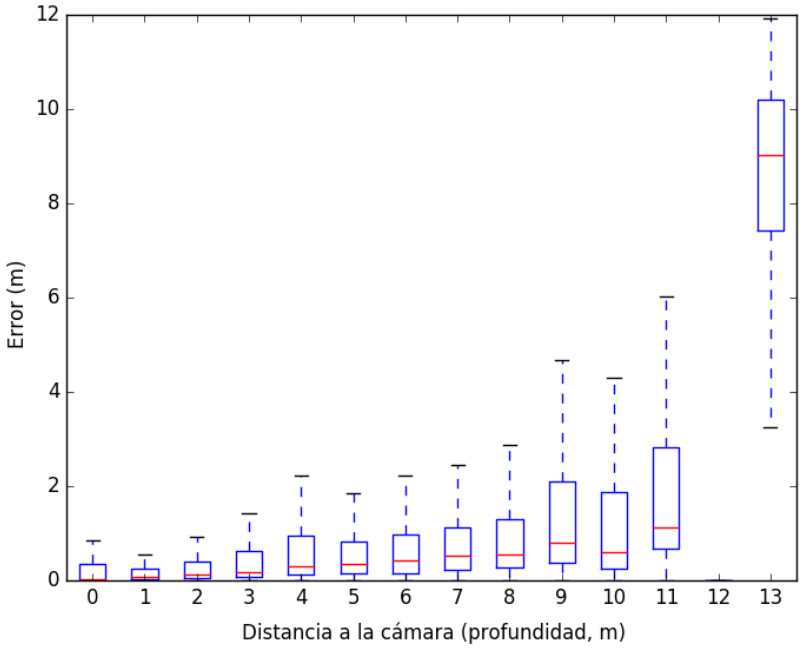


Figura 5.3.10: Error en LIBELAS (profundidad vs. error) en el Dataset Tsukuba.

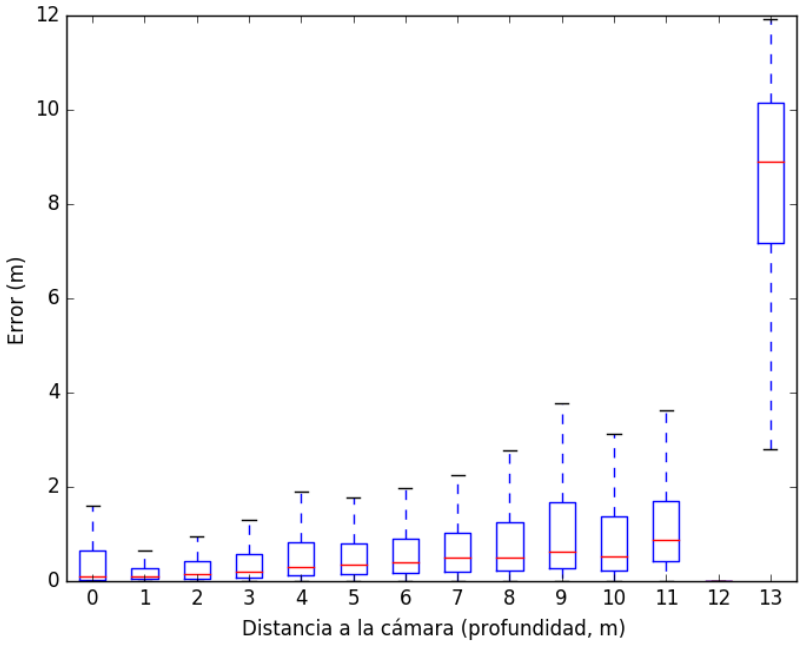


Figura 5.3.11: Error en S-PTAM Denso (profundidad vs. error) en el Dataset Tsukuba.

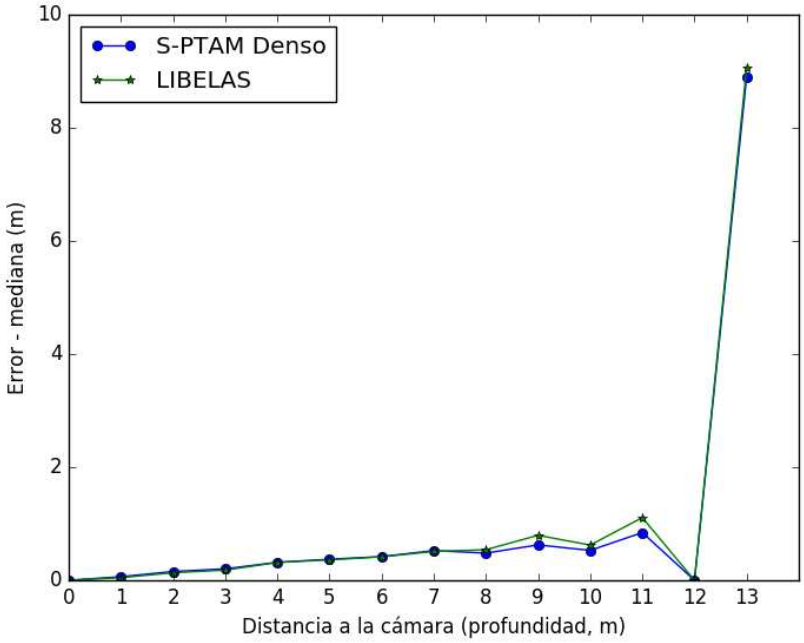
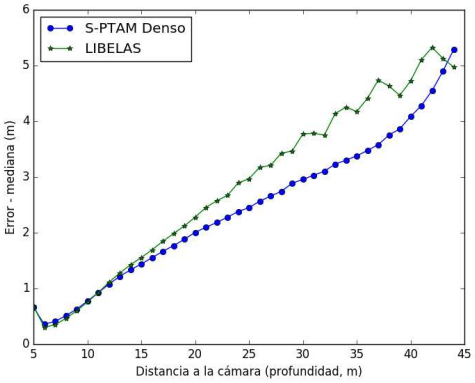
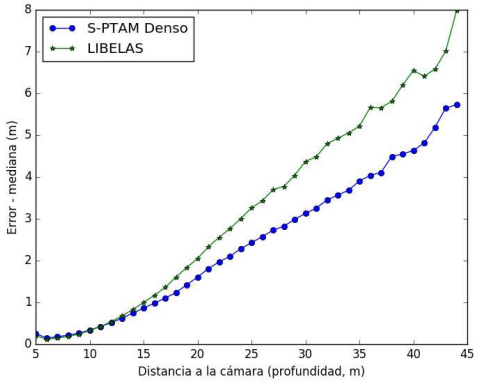


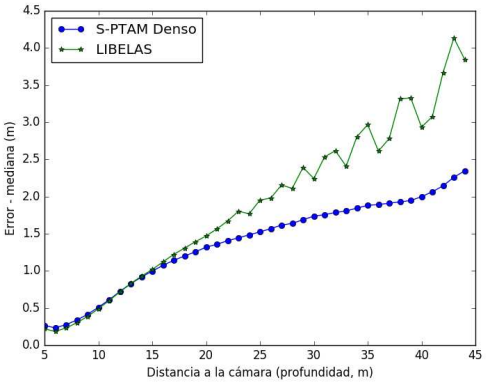
Figura 5.3.12: Comparación de mediana de error entre LIBELAS y S-PTAM Denso (profundidad vs. error) en el Dataset Tsukuba.



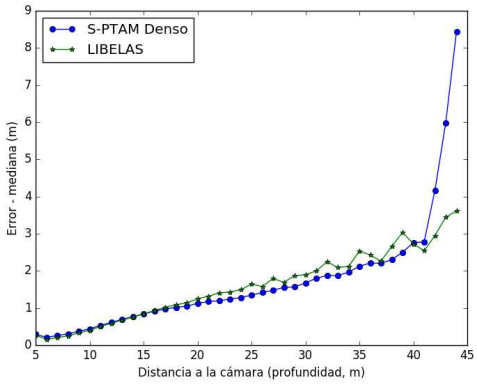
(a) Secuencia 00 del Dataset KITTI.



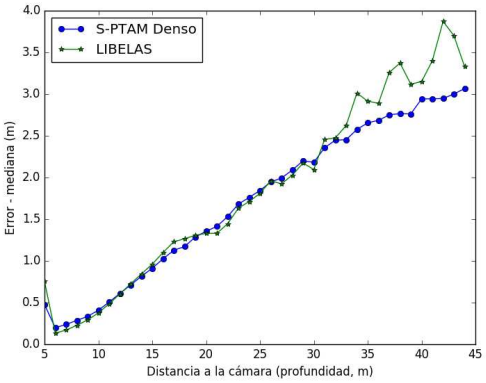
(b) Secuencia 01 del Dataset KITTI.



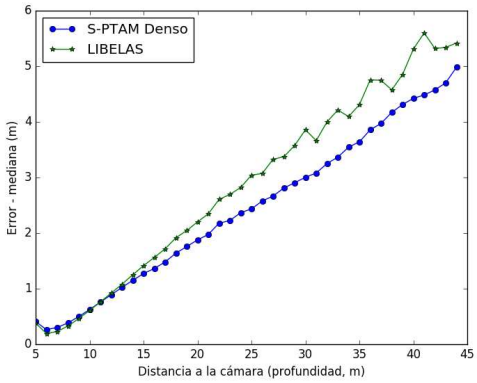
(c) Secuencia 02 del Dataset KITTI.



(d) Secuencia 03 del Dataset KITTI.

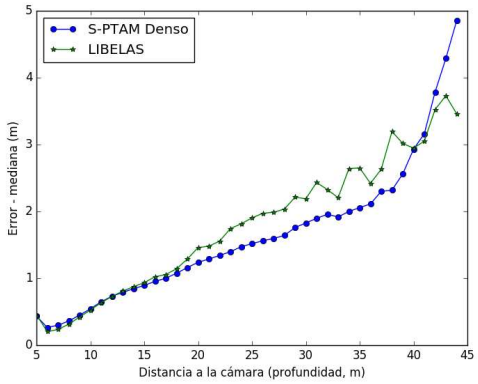


(e) Secuencia 04 del Dataset KITTI.

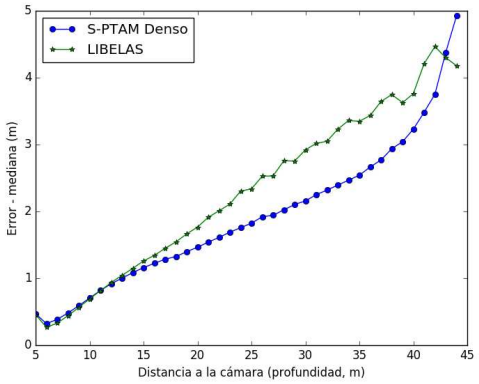


(f) Secuencia 05 del Dataset KITTI.

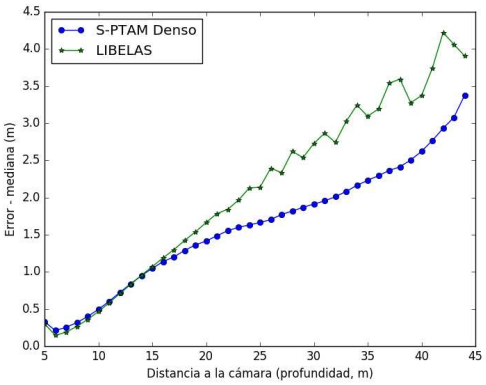
Figura 5.3.13: Comparación de mediana de error entre LIBELAS y S-PTAM Denso para el dataset KITTI (profundidad vs. error).



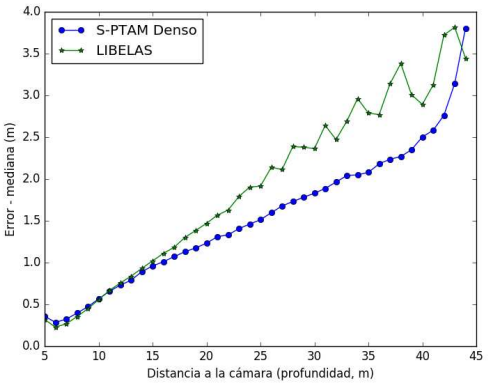
(a) Secuencia 07 del Dataset KITTI.



(b) Secuencia 08 del Dataset KITTI.



(c) Secuencia 09 del Dataset KITTI.



(d) Secuencia 10 del Dataset KITTI.

Figura 5.3.14: Comparación de mediana de error entre LIBELAS y S-PTAM Denso para el dataset KITTI (profundidad vs. error).

5.3.2. Costo computacional

Las Figuras 5.3.15 y 5.3.16 dividen el número total de puntos al final de cada secuencia entre dos clases: aquellos que fueron vistos una única vez —*hipótesis*—, o en múltiples ocasiones —*validados*—, y finalmente se detalla el número total de fusiones de profundidad realizadas. Estos valores muestran cómo el método propuesto reduce considerablemente el tamaño del mapa comparado con un enfoque *naive* donde simplemente se concatenan las distintas nubes de puntos estéreo generadas.

Nótese en las figuras que el número de puntos es mayor en las secuencias del dataset KITTI, ya que la cámara recorre una distancia mucho mayor que en la secuencia Tsukuba. Por otro lado, el total de fusiones resulta mayor en la secuencia Tsukuba dado que la escena es observada desde múltiples puntos de vista, a diferencia del dataset KITTI donde la trayectoria resulta puramente exploratoria.

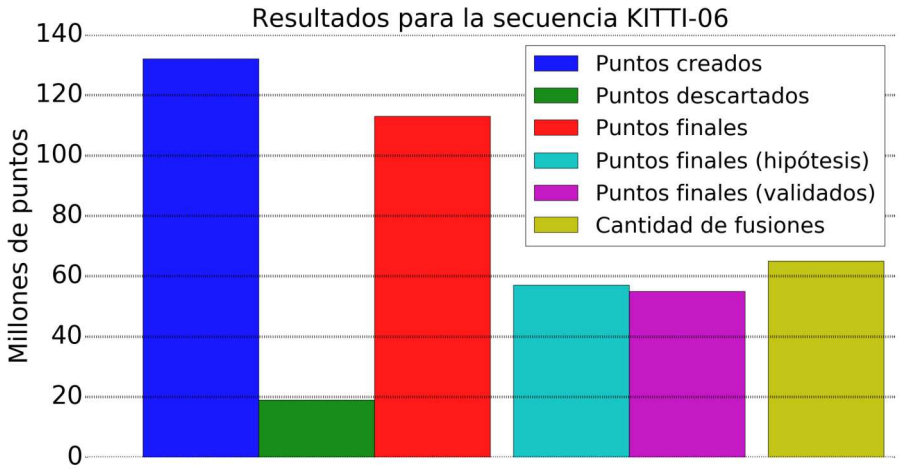


Figura 5.3.15: Cantidad total de puntos a lo largo de una ejecución de S-PTAM Denso (creados, descartados, presentes en la reconstrucción final —hipótesis (vistos una única vez) y validados (fusionados múltiples veces)—, y el número total de fusiones) en la secuencia 06 del dataset KITTI.

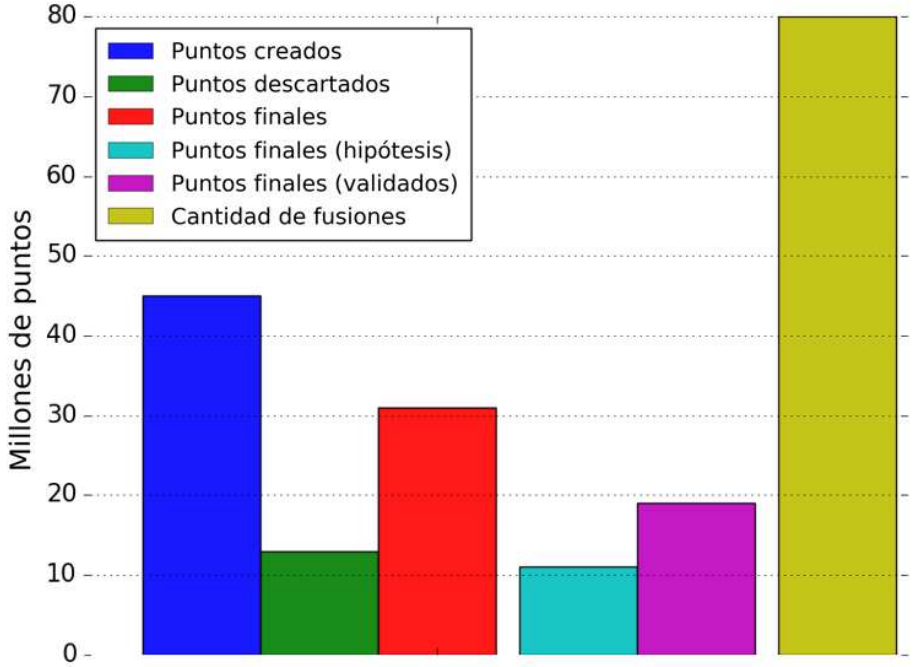


Figura 5.3.16: Cantidad total de puntos a lo largo de una ejecución de S-PTAM Denso (creados, descartados, presentes en la reconstrucción final -hipótesis (vistos una única vez) y validados (fusionados múltiples veces)-, y el número total de fusiones) en la secuencia del dataset Tsukuba.

La Tabla 5.3.17 contiene los pasos más relevantes de nuestro algoritmo -cómputo de disparidad, fusión de mapa y refinamiento de mapa- y el tiempo promedio de cómputo por *keyframe*. La Tabla muestra que el método de S-PTAM Denso resulta apto para la navegación de robots móviles, dado que puede correr en tiempo real a 5 fps en las secuencias del dataset KITTI y a 18 fps en el caso de Tsukuba. Cabe destacar que el tiempo computacional empleado en la fusión y expansión de mapa es prácticamente el mismo en ambos casos. El tiempo consumido por el Refinamiento de mapa es despreciable en comparación con las restantes tareas.

Secuencia	Disparidad (ms)	Expansión y fusión (ms)	Refinamiento (ms)
Tsukuba	118.20	132.72	4.80
KITTI 00	156.92	115.40	6.53
KITTI 01	156.10	69.12	5.21
KITTI 02	160.63	108.13	6.58
KITTI 03	153.66	83.01	4.83
KITTI 04	149.15	72.99	5.00
KITTI 05	165.31	108.97	5.97
KITTI 06	154.89	83.21	4.84
KITTI 07	155.74	113.15	6.10
KITTI 08	152.10	101.01	5.22
KITTI 09	159.84	101.20	5.76
KITTI 10	158.81	117.42	6.35

Figura 5.3.17: Tiempo de computación promedio (en ms) para cada fase de nuestro algoritmo.

Capítulo 6

Conclusiones

En este trabajo de tesina se presentó un método capaz de realizar una reconstrucción densa del entorno, integrado a un sistema de SLAM disperso de visión estéreo basado en *keyframes*. El método es denominado S-PTAM Denso debido a que la implementación se realizó para ser ejecutada conjuntamente y de manera desacoplada con el sistema de SLAM estéreo S-PTAM [6].

El sistema se dividió en tres hilos de ejecución concurrentes: generación y reproyección de mapas de disparidad, transformación y fusión de nubes de puntos, y refinamiento del mapa global. Este diseño fuertemente paralelizable permitió aprovechar la capacidad de cómputo de procesadores multinúcleo, mejorando considerablemente el desempeño del método. Los *keyframes* generados por el sistema S-PTAM son procesados secuencialmente por el hilo de *Cómputo de Disparidad* generando un mapa de disparidad que, a su vez, es tratado posteriormente en el hilo de *Expansión y fusión de mapa*. El hilo de *Expansión y fusión de mapa* combina el mapa de disparidad y la posición del *keyframe* asociado para extender y ajustar el mapa buscando correspondencias con los puntos 3D existentes en el mapa denso. Un tercer hilo de *Refinamiento de mapa* (*Mapping*) ajusta constantemente la posición de los *keyframes* y las nubes de puntos asociadas cada vez que S-PTAM refina una posición, comportamiento esperable en sistemas de SLAM ya que el mapa disperso y la trayectoria de la cámara son refinados a lo largo de toda la secuencia.

Para el cómputo de mapas de disparidad se decidió utilizar una librería externa. Se optó por la librería de código abierto LIBELAS [13], que demostró satisfacer los requeriminetos del método de densificación generando mapas de disparidad densos abarcando casi la totalidad de la imagen. Contar con suficiente información de disparidad de la escena observada permitió que la heurística empleada por S-PTAM Denso mejore considerablemente la reconstrucción del entorno —mediante la fusión de puntos y descarte de *outliers*— en comparación con un enfoque *naive,* donde simplemente se concatenan las distintas nubes de puntos.

El método desarrollado se programó como un nodo del framework ROS (*Robot Operating System*) para ser ejecutado en paralelo y conjuntamente con el nodo correspondiente al sistema S-PTAM. Evaluamos el sistema en un entorno real (dataset KITTI) [70] y sintético (dataset Tsukuba) [78]. Los experimentos realizados sobre datasets de dominio público muestran que el sistema es apto para generar en tiempo real una reconstrucción densa, sin requerir el uso de GPUs, con precisión similar a otros métodos de densificación presentes en el estado del arte. La precisión del mapa denso y el rendimiento en la reconstrucción hacen que el método sea apto para la navegación de robots autónomos, tanto en entornos de interior y exterior como

en secuencias de extensa longitud. Sin embargo, los experimentos muestran que la precisión de la reconstrucción en entornos de exterior —Dataset KITTI— superaron ampliamente a los de interior —Dataset Tsukuba— debido principalmente a que el cociente de relación profundidad-baseline es alto en el primer caso, a diferencia del segundo.

6.1. Trabajo futuro

Como trabajo futuro derivado de la presente tesina identificamos varias mejoras que resultan interesantes de ser investigadas.

Actualmente, la heurística de fusión planteada utiliza solamente información geométrica para la búsqueda de correspondencias entre puntos del mapa. Es posible mejorar la heurística incluyendo información de apariencia que ayudaría a determinar si dos puntos pertenecen al mismo objeto, lo que aumentaría la precisión del método. Por otro lado, teniendo en cuenta el factor de error existente tanto en las posiciones como en el mapa denso, el criterio para determinar correspondencias puede ampliarse a considerar no solo el píxel del punto 3D proyectado sino un entorno alrededor del mismo.

Otra línea de trabajo planteada a futuro consiste en simplificar las nubes de puntos mediante la detección de regiones planares, reduciendo el uso de CPU y memoria. De la misma manera, el uso de información semántica y técnicas de *machine-learning* puede permitir detectar características visuales de alto nivel que pueden emplearse para mejorar la precisión del mapa y reducir su complejidad.

Como se describió anteriormente, la implementación actual del sistema se realizó utilizando LIBELAS [13] para el cálculo de mapas de disparidad y el sistema de SLAM estéreo S-PTAM [6] para posicionamiento. La versatilidad del método permite fácilmente utilizar otros componentes para la realización de estas tareas, por lo que un análisis comparativo de diversos proyectos resultaría en un aporte que podría repercutir considerablemente en la precisión y el desempeño de S-PTAM Denso.

Los experimentos realizados en este trabajo pueden extenderse a numerosos datasets que se encuentran disponibles públicamente dentro de la comunidad robótica. Un análisis de la reconstrucción obtenida en diversos escenarios permitiría conocer con mayor certeza la precisión y las limitaciones del método propuesto. Así mismo, se planea correr el método sobre un robot móvil en tiempo real, para lo cual puede resultar necesario modificar y optimizar la implementación de modo que se adapte a las limitaciones de hardware de la plataforma. En este sentido, el soporte y uso de GPU puede resultar de suma importancia y debe ser considerado como una línea de trabajo a futuro.

Por último, para demostrar la utilidad del sistema propuesto, la salida generada por el método podría ser utilizada por sistemas de navegación y detección de obstáculos que requieran de un mapa denso para su funcionamiento.

Bibliografía

[1] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part I. *IEEE Robotics Automation Magazine*, 13(2):99–110, June 2006.

[2] Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (SLAM): part II. *IEEE Robotics Automation Magazine*, 13(3):108–117, September 2006.

[3] Thomas Lemaire, Cyrille Berger, Il-Kyun Jung, and Simon Lacroix. Vision-Based SLAM: Stereo and Monocular Approaches. *International Journal of Computer Vision*, 74(3):343–364, 2007.

[4] M. Tomono. Robust 3D SLAM with a stereo camera based on an edge-point ICP algorithm. In *2009 IEEE International Conference on Robotics and Automation*, pages 4306–4311, May 2009.

[5] Kuen-Han Lin and Chieh-Chih Wang. Stereo-based simultaneous localization, mapping and moving object tracking. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3975–3980, Oct 2010.

[6] Taihú Pire, Thomas Fischer, Javier Civera, Pablo De Cristóforis, and Julio Jacobo Berlles. Stereo parallel tracking and mapping for robot localization. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1373–1378, September 2015.

[7] Etienne Mouragnon, Maxime Lhuillier, Michel Dhome, Fabien Dekeyser, and Patrick Sayd. Monocular Vision Based SLAM for Mobile Robots. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 3, pages 1027–1031, 2006.

[8] Seo-Yeon Hwang and Jae-Bok Song. Monocular Vision-Based SLAM in Indoor Environment Using Corner, Lamp, and Door Features From Upward-Looking Camera. *IEEE Transactions on Industrial Electronics*, 58(10):4804–4812, Oct 2011.

[9] Jan Stühmer, Stefan Gumhold, and Daniel Cremers. *Real-Time Dense Geometry from a Handheld Camera*, pages 11–20. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[10] Gottfried Graber, Thomas Pock, and Horst Bischof. Online 3D reconstruction using convex optimization. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 708–711, Nov 2011.

[11] Richard Newcombe, Steven Lovegrove, and Andrew Davison. DTAM: Dense tracking and mapping in real-time. In *2011 International Conference on Computer Vision*, pages 2320–2327, Nov 2011.

[12] Sid Yingze Bao, Manmohan Chandraker, Yuanqing Lin, and Silvio Savarese. Dense Object Reconstruction with Semantic Priors. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1264–1271, June 2013.

[13] Andreas Geiger, Martin Roser, and Raquel Urtasun. *Efficient Large-Scale Stereo Matching*, pages 25–38. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[14] Andreas Geiger, Julius Ziegler, and Christoph Stiller. StereoScan: Dense 3d reconstruction in real-time. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 963–968, June 2011.

[15] Marc Pollefeys, Luc Van Gool, Maarten Vergauwen, Frank Verbiest, Kurt Cornelis, Jan Tops, and Reinhard Koch. Visual Modeling with a Hand-Held Camera. *International Journal of Computer Vision*, 59(3):207–232, 2004.

[16] Steven Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 519–528, June 2006.

[17] Raúl Mur-Artal, José M. M. Montiel, and Juan D. Tardós. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics*, 31(5):1147–1163, Oct 2015.

[18] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct Sparse Odometry. In *arXiv:1607.02565*, July 2016.

[19] Gabriel Nützi, Stephan Weiss, Davide Scaramuzza, and Roland Siegwart. Fusion of IMU and Vision for Absolute Scale Estimation in Monocular SLAM. *Journal of Intelligent & Robotic Systems*, 61(1):287–299, 2011.

[20] Lindsay Kleeman. Advanced sonar and odometry error modeling for simultaneous localisation and map building. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, volume 1, pages 699–704 vol.1, Oct 2003.

[21] Fabrizio Abrate, Basilio Bona, and Marina Indri. Experimental EKF-based SLAM for Mini-rovers with IR Sensors Only. In *EMCR*, 2007.

[22] David Cole and Paul Newman. Using laser range data for 3D SLAM in outdoor environments. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 1556–1563, May 2006.

[23] Justin David Carlson. *Mapping Large, Urban Environments with GPS-Aided SLAM*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 2010.

[24] Leopoldo Armesto and Josep Tornero. SLAM based on Kalman filter for multi-rate fusion of laser and encoder measurements. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 2, pages 1860–1865 vol.2, Sept 2004.

[25] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2 edition, 2003.

[26] Emanuele Trucco and Alessandro Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1998.

[27] Yi Ma, Stefano Soatto, Jana Kosecka, and Shankar Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. SpringerVerlag, 2003.

[28] Hao Li, Bart Adams, Leonidas Guibas, and Mark Pauly. Robust Single-view Geometry and Motion Reconstruction. In *ACM SIGGRAPH Asia 2009 Papers*, SIGGRAPH Asia '09, pages 175:1–175:10, New York, NY, USA, 2009. ACM.

[29] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. *IEEE Transactions on Robotics*, 32(6):1309–1332, Dec 2016.

[30] Georg Klein and David Murray. Parallel Tracking and Mapping for Small AR Workspaces. In *ISMAR*, pages 1–10, Washington, DC, USA, 2007. IEEE Computer Society.

[31] Wei Tan, Haomin Liu, Zilong Dong, Guofeng Zhang, and Hujun Bao. Robust monocular SLAM in dynamic environments. In *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 209–218, Oct 2013.

[32] Liz Murphy, Timothy Morris, Ugo Fabrizi, Michael Warren, Michael Milford, Ben Upcroft, Michael Bosse, and Peter Corke. Experimental Comparison of Odometry Approaches. In Jaydev P. Desai, Gregory Dudek, Oussama Khatib, and Vijay Kumar, editors, *Experimental Robotics*, volume 88 of *Springer Tracts in Advanced Robotics*, pages 877–890. Springer International Publishing, 2013.

[33] Jianbo Shi and Carlo Tomasi. Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 593–600, June 1994.

[34] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[35] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded Up Robust Features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 3951 of *Lecture Notes in Computer Science*, pages 404–417. Springer Berlin Heidelberg, 2006.

[36] Edward Rosten and Tom Drummond. Machine Learning for High-Speed Corner Detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 3951 of *Lecture Notes in Computer Science*, pages 430–443. Springer Berlin Heidelberg, 2006.

[37] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2564–2571, November 2011.

[38] Pablo Fernández Alcantarilla, Adrien Bartoli, and Andrew J. Davison. KAZE Features. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part VI*, Proceedings of the European Conference on Computer Vision (ECCV), pages 214–227, Berlin, Heidelberg, 2012. Springer-Verlag.

[39] Jangheon Kim and Thomas Sikora. Gaussian scale-space dense disparity estimation with anisotropic disparity-field diffusion. In *Fifth International Conference on 3-D Digital Imaging and Modeling (3DIM'05)*, pages 556–563, June 2005.

[40] Christoph Strecha and Luc Van Gool. PDE-based multi-view depth estimation. In *Proceedings. First International Symposium on 3D Data Processing Visualization and Transmission*, pages 416–425, 2002.

[41] Ayan Chakrabarti, Ying Xiong, Steven Gortler, and Todd Zickler. Low-level vision by consensus in a spatial hierarchy of regions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4009–4017, June 2015.

[42] Koichiro Yamaguchi, David McAllester, and Raquel Urtasun. Robust Monocular Epipolar Flow Estimation. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1862–1869, June 2013.

[43] Koichiro Yamaguchi, David McAllester, and Raquel Urtasun. *Efficient Joint Segmentation, Occlusion Labeling, Stereo and Flow Estimation*, pages 756–771. Springer International Publishing, Cham, 2014.

[44] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-End Learning of Geometry and Context for Deep Stereo Regression. *arXiv preprint arXiv:1703.04309*, 2017.

[45] W. Luo, A. G. Schwing, and R. Urtasun. Efficient Deep Learning for Stereo Matching. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5695–5703, June 2016.

[46] Nikolaus Mayer, Eddy Ilg, Philip Häusser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4040–4048, June 2016.

[47] Daniel Scharstein and Richard Szeliski. A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. *International Journal of Computer Vision*, 47(1):7–42, 2002.

[48] Nalpantidis Lazaros, Georgios Christou Sirakoulis, and Antonios Gasteratos. Review of Stereo Vision Algorithms: From Software to Hardware. *International Journal of Optomechatronics*, 2(4):435–462, 2008.

[49] Il-Kyun Jung and Simon Lacroix. High resolution terrain mapping using low attitude aerial stereo imagery. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 946–951 vol.2, Oct 2003.

[50] Pekka Paalanen, Ville Kyrki, and Joni-Kristian Kamarainen. Towards Monocular On-Line 3D Reconstruction. In *Workshop on Vision in Action: Efficient strategies for cognitive agents in complex environments*, Marseille, France, Oct 2008. Markus Vincze and Danica Kragic and Darius Burschka and Antonis Argyros.

[51] Sarthak Upadhyay, Ayush Dewan, Arun Kumar Singh, and Madhava Krishna. Trajectory planning for monocular slam based exploration system. In *Proceedings of the 2015 Conference on Advances In Robotics*, page 27. ACM, 2015.

[52] Roberto Toldo and Andrea Fusiello. *Robust Multiple Structures Estimation with J-Linkage*, pages 537–547. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[53] Alberto Argiles, Javier Civera, and Luis Montesano. Dense multi-planar scene estimation from a sparse set of images. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4448–4454, Sept 2011.

[54] Martin Fischler and Robert Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[55] Richard Newcombe and Andrew Davison. Live dense reconstruction with a single moving camera. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1498–1505, June 2010.

[56] Lucas Teixeira and Margarita Chli. Real-time mesh-based scene estimation for aerial inspection. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4863–4869, Oct 2016.

[57] Qi Pan, Gerhard Reitmayr, and Tom Drummond. ProFORMA: Probabilistic Feature-based On-line Rapid Model Acquisition. In *BMVC*, volume 2, page 6. Citeseer, 2009.

[58] Pedro Felzenszwalb and Daniel Huttenlocher. Efficient Graph-Based Image Segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[59] Greg Mori. Guiding model search using segmentation. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1417–1423 Vol. 2, Oct 2005.

[60] Alex Levinshtein, Adrian Stere, Kiriakos Kutulakos, David Fleet, Sven Dickinson, and Kaleem Siddiqi. TurboPixels: Fast Superpixels Using Geometric Flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2290–2297, Dec 2009.

[61] Andrea Vedaldi and Stefano Soatto. *Quick Shift and Kernel Methods for Mode Seeking*, pages 705–718. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[62] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, Nov 2012.

[63] Alejo Concha and Javier Civera. Using superpixels in monocular SLAM. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 365–372, May 2014.

[64] Alejo Concha and Javier Civera. DPPTAM: Dense piecewise planar tracking and mapping from a monocular sequence. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5686–5693, Sept 2015.

[65] Javier Civera, Dorian Gálvez-López, Luis Riazuelo, Juan Tardós, and José Montiel. Towards semantic SLAM using a monocular camera. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1277–1284, Sept 2011.

[66] Renato Salas-Moreno, Richard Newcombe, Hauke Strasdat, Paul Kelly, and Andrew Davison. SLAM++: Simultaneous Localisation and Mapping at the Level of Objects. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1352–1359, June 2013.

[67] Varsha Hedau, Derek Hoiem, and David Forsyth. Recovering the spatial layout of cluttered rooms. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1849–1856, Sept 2009.

[68] Alex Flint, David Murray, and Ian Reid. Manhattan scene understanding using monocular, stereo, and 3D features. In *2011 International Conference on Computer Vision*, pages 2228–2235, Nov 2011.

[69] Vibhav Vineet, Ondrej Miksik, Morten Lidegaard, Matthias Nießner, Stuart Golodetz, Victor Prisacariu, Olaf Kähler, David Murray, Shahram Izadi, Patrick Pérez, et al. Incremental dense semantic stereo fusion for large-scale semantic scene reconstruction. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 75–82, May 2015.

[70] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision Meets Robotics: The KITTI Dataset. *The International Journal of Robotics Research, IJRR*, 32(11):1231–1237, September 2013.

[71] Jakob Engel, Jörg Stückler, and Daniel Cremers. Large-scale direct slam with stereo cameras. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1935–1942, Sept 2015.

[72] Raul Mur-Artal and Juan Tardós. Probabilistic Semi-Dense Mapping from Highly Accurate Feature-Based Monocular SLAM. In *Robotics: Science and Systems XI, Sapienza University of Rome, Rome, Italy, July 13-17, 2015*, 2015.

[73] Jakob Engel, Thomas Schöps, and Daniel Cremers. *LSD-SLAM: Large-Scale Direct Monocular SLAM*, pages 834–849. Springer International Publishing, Cham, 2014.

[74] David Caruso, Jakob Engel, and Daniel Cremers. Large-scale direct SLAM for omnidirectional cameras. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 141–148, Sept 2015.

[75] Alejo Concha, Giuseppe Loianno, Vijay Kumar, and Javier Civera. Visual-inertial direct SLAM. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1331–1338, May 2016.

[76] Veeravalli S. Varadarajan. *Lie Groups, Lie Algebras, and Their Representations*. Graduate Texts in Mathematics. Springer-Verlag New York, 1984.

[77] Boris Delaunay. Sur la sphère vide. *Otdelenie Matematicheskikh i Estestvennykh Nauk*, 8:793–800, 1934.

[78] M. Peris, S. Martull, A. Maki, Y. Ohkawa, and K. Fukui. Towards a simulation driven stereo vision system. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR)*, pages 1038–1042, November 2012.

[79] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*, 2009.

[80] Gary Bradski et al. The opencv library. *Doctor Dobbs Journal*, 25(11):120–126, 2000.

[81] R. B. Rusu and S. Cousins. 3d is here: Point cloud library (pcl). In *2011 IEEE International Conference on Robotics and Automation*, pages 1–4, May 2011.