Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California

Maestría en Ciencias en Ciencias de la Computación

Diseño de algoritmos adaptativos para sistemas SLAM con cámara RGB-D

Tesis

para cubrir parcialmente los requisitos necesarios para obtener el grado de Maestro en Ciencias

Presenta:

Antonio de Jesús Ortiz González

Ensenada, Baja California, México

2019

Tesis defendida por

Antonio de Jesús Ortiz González

y aprobada por el siguiente Comité

Dr. Vitaly Kober

Director de tesis

Dr. Hugo Homero Hidalgo Silva

Dr. Luis Alonso Gallardo Delgado

Dr. Ubaldo Ruiz López

Coordinador del Posgrado en Ciencias de la Computación

Dra. Rufina Hernández Martínez

Directora de Estudios de Posgrado

Resumen de la tesis que presenta Antonio de Jesús Ortiz González como requisito parcial para la obtención del grado de Maestro en Ciencias en Ciencias de la Computación .

Diseño de algoritmos adaptativos para sistemas SLAM con cámara RGB-D

Resumen aprobado por:

Dr. Vitaly Kober
Director de tesis

La localización y el mapeo simultáneo (SLAM) visual es un problema de investigación muy activo en las áreas de la robótica móvil autónoma y visión por computadora, donde un robot necesita localizarse en entornos desconocidos procesando la información de cámaras a bordo sin sistemas de referencia externos como el Sistema de Posicionamiento Global (GPS). En este trabajo, se presenta un SLAM visual basado en características visuales, que es capaz de producir mapas tridimensionales de alta calidad en tiempo real con una cámara RGB-D de bajo costo como el Microsoft Kinect. El mapa generado es adecuado para la planificación futura o tareas comunes de navegación de robots. Primero, se presenta una evaluación integral del rendimiento de la combinación de diferentes detectores y descriptores de características visuales más usados en SLAM. El propósito principal detrás de estas evaluaciones es determinar la mejor combinación de algoritmo de descriptor-detector para usar en la navegación del robot. En segundo lugar, utilizamos el algoritmo RANSAC en combinación con el algoritmo del punto más cercano iterativo (ICP) para obtener el movimiento relativo entre cuadros consecutivos y luego refinar la estimación de pose siguiendo la regla de composición. Sin embargo, la distribución espacial y la resolución de los datos de profundidad afectan el rendimiento de la reconstrucción de escenas 3D basada en RANSAC e ICP. Debido a esto, proponemos una arquitectura adaptativa que calcule la estimación de pose a partir de las mediciones más confiables en un entorno dado. Evaluamos nuestro enfoque ampliamente en los conjuntos de datos de referencia disponibles comúnmente usados en la literatura. Los resultados experimentales demuestran que nuestro sistema puede lidiar robustamente con escenarios desafiantes mientras es lo suficientemente rápido para aplicaciones en línea.

Palabras clave: Localización, mapeo, SLAM, cámara RGB-D, característica visual, algoritmo adaptativo, RANSAC, ICP

Abstract of the thesis presented by Antonio de Jesús Ortiz González as a partial requirement to obtain the Master of Science degree in Computer Science .

Design of adaptive algorithms for SLAM systems with RGB-D camera

Abstract approved by:

Dr. Vitaly Kober
Thesis Director

Visual Simultaneous localization and mapping (SLAM) is a very active research problem in the fields of autonomous mobile robotics and computer vision, where a robot needs to localize itself in unknown environments by processing onboard camera information without external referencing systems such as Global Positioning System (GPS). In this work, we present a feature-based visual SLAM, which is able to produce high-quality tridimensional maps in real time with a low-cost RGB-D camera such as the Microsoft Kinect. It is suitable for future planning or common robot navigation tasks. First, a comprehensive performance evaluation of the combination of different state-of-the-art feature detectors and descriptors is presented. The main purpose behind these evaluations is to determine the best detector-descriptor combination to use for robot navigation. Second, we use the RANSAC algorithm in combination with the Iterative Closest Point (ICP) algorithm to get the relative motion between consecutive frames and then to refine the pose estimate following the composition rule. However, the spatial distribution and resolution of depth data affect the performance of 3D scene reconstruction based on the RANSAC and ICP. Due to this, we propose an adaptive architecture, which computes the pose estimate from the most reliable measurements in a given environment. We evaluate our approach extensively on common available benchmark datasets. The experimental results demonstrate that our system can robustly deal with challenging scenarios while being fast enough for online applications.

Keywords: Localization, mapping, SLAM, RGB-D camera, visual feature, adaptive algorithm, RANSAC, ICP

Dedicatoria

A Berenice y Ana Victoria

A mis padres y mis hermanos

Agradecimientos

Al Centro de Investigación Científica y de Educación Superior de Ensenada por la oportunidad y apoyo brindado para realizar este trabajo.

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) por brindarme el apoyo económico para realizar mis estudios de maestría. No. de becario: 634318

A mi asesor de tesis Dr. Vitaly Kober por darme la confianza de trabajar con él y por compartir sus conocimientos, experiencias y tiempo para la relización de este trabajo.

A mis padres, por sus consejos y valores que me han influido siempre. Gracias por su infinito amor que me han brindado incodicionalmente

A Berenice por estar a mi lado durante todo este tiempo, sobre todo en los momentos más estresantes, por su apoyo incondicional y su cariño.

A Ana Victoria, la razón de mi felicidad, de mi esfuerzo y de mis ganas de salir adelante para ser mejor cada día, nada me ha motivado más estos meses.

A mis compañeros de generación, por compartir sus conocimientos pero sobre todo por su amistad, les deseo mucho éxito.

Tabla de contenido

Página

Tabla de contenido (continuación)

Tabla de contenido (continuación)

Lista de figuras

Figura Página

Lista de figuras (continuación)

Figura Página

Lista de figuras (continuación)

Lista de tablas

Tabla Página

Capítulo 1. Introducción

1.1. Localización y mapeo simultáneo (SLAM)

En los últimos años, la disponibilidad de robots móviles ha incrementado, en consecuencia, cada vez es más común encontrarlos en casas, oficinas e industrias, donde cumplen con tareas importantes como transporte, limpieza, telepresencia, brazos robóticos en líneas de ensamblaje, vehículos autónomos y manipuladores que asisten operaciones. Para realizar dichas tareas, los robots tienen que ser capaces de adaptarse a la enorme incertidumbre que existe en el mundo físico. Existe un número considerable de factores que contribuyen a la incertidumbre de un robot. Los ambientes por donde navega el robot son inciertos, especialmente cuando opera cercano a personas. Los sensores están limitados en lo que pueden percibir, ya que, el rango y la resolución espacial de un sensor está sujeto a limitaciones físicas. Por ejemplo, las cámaras no pueden ver a través de las paredes y la resolución espacial de la imagen capturada por la cámara es limitada. Los sensores también están sujetos a ruido, lo que perturba las mediciones del sensor de manera impredecible y, por lo tanto, limita la información que puede ser extraída.

Poder navegar es comúnmente considerado como uno de los requisitos básicos para robots verdaderamente autónomos. Esta tarea permite que el robot pueda recoger productos y los envíe a un lugar determinado, limpie un área de manera confiable, ofrezca vigilancia, ayude a los humanos en situaciones potencialmente peligrosas, como los escenarios de búsqueda y rescate. Por lo tanto, la navegación es uno de los principales temas de investigación en la comunidad robótica.

Cuando un robot desempeña sus tareas en entornos interiores como casas u oficinas, por lo general, no es factible equipar al robot con un modelo preciso de ese entorno ni con un sistema de posicionamiento global (GPS). Por lo tanto, el robot primero necesita crear un modelo del mundo. Para un robot móvil, este modelo debe comprender un mapa que permita al robot localizarse y planificar rutas libres de colisiones donde es deseable que la localización se realice como lo hacen los seres humanos y animales, al procesar la información de los sensores a bordo. El método más simple para localizar un robot es procesando la información del sensor para calcular movimiento incremental, el cual es llamado odometría. Esto nos permite recuperar la trayectoria

del robot, el cual inevitablemente acumulará error haciendo que la trayectoria estimada se aleje de la trayectoria real del robot. La técnica de odometría es adecuada para la estimación de movimientos a corto plazo, pero para operaciones a largo plazo en el mismo entorno, lo más conveniente sería tener un mapa que permita una localización más precisa. Construir dicho mapa a partir de sensores a bordo es un problema complejo, ya que el robot necesita localizarse para crear un mapa a partir de las mediciones del sensor, es decir, es necesario localizar al robot en relación a un mapa que es desconocido y que tiene que ser estimado incrementalmente. Por lo tanto, estas tareas deben realizarse simultáneamente, este problema es llamado Localización y Mapeo Simultáneo o SLAM por sus siglas en inglés (Simultaneous Localization and Mapping) (Thrun *et al.*, 2005). Surge cuando el robot no tiene acceso al mapa del ambiente donde se desenvuelve, ni conoce su posición, en cambio, todo lo que recibe son observaciones y controles. La combinación de ambas tareas resulta en un problema complejo de estimación de estados el cual involucra miles de variables, por ejemplo, la pose del robot en cualquier instante de tiempo y todas las observaciones adquiridas por el robot. Errores en la estimación de la posición del robot introduce errores en la estimación del mapa y viceversa, además estos errores tienden a acumularse.

Una formulación muy popular hoy en día para abordar el problema de SLAM es la basada en grafos. En dicho grafo, las poses del robot se codifican como nodos y la información del sensor ruidoso se modela como aristas o restricciones entre los nodos. La solución es obtenida determinando la configuración de las poses del robot que mejor satisface todas las restricciones.

El problema de SLAM ha sido sujeto de mucha investigación de la cual han sido desarrolladas soluciones factibles para diferentes escenarios. Cuando se utilizan cámaras, se conoce como SLAM visual. Las principales ventajas del uso de cámaras son que estos dispositivos proporcionan una mayor cantidad de información del entorno, como color y texturas, lo que permite integrar tareas de alto nivel como detección y reconocimiento de personas y lugares. Comparando con láseres, las cámaras son más económicas, ligeras y consumen menos energía. Dentro de las desventajas que se tienen al emplear cámaras como único sensor cabe mencionar: resolución insuficiente, sensibilidad a los cambios de iluminación, imágenes pobres por falta de textura o borrosas por movimientos rápidos del robot. A pesar que el uso de cámaras es una

alternativa viable, extraer la información tridimensional para sistemas monoculares no es trivial. Por esto, a menudo se desarrollan sistemas estéreo que aprovechan las características geométricas entre ambas cámaras para calcular la información tridimensional, sin embargo, estos sistemas deben ser calibrados minuciosamente para poder obtener resultados precisos.

Recientemente, se ha desarrollado un nuevo tipo de cámara 3D llamada cámara RGB-D. Son una combinación de una cámara monocular RGB y un sensor de profundidad, basadas en luz estructurada, es decir, las cámaras RGB-D iluminan la escena con un patrón de luz conocido como speckle (Smisek *et al.*, 2011). Conociendo los parámetros de calibración intrínsecos de la cámara y la calibración extrínsecos entre la cámara y el sensor de profundidad, la profundidad medida se puede registrar en un mapa de profundidad con correspondencia 1:1 píxeles en la imagen RGB. Esto implica que para cada pixel en la imagen sabemos su profundidad sin necesidad de realizar una comparación estéreo. Sin embargo, debido a la naturaleza del sensor de profundidad, su uso está restringido a ambientes interiores donde el rango de profundidad está limitado. Para explotar el potencial de este sensor, en este trabajo se creó un sistema SLAM adaptativo para cámaras RGB-D basado en grafos, el cual permite al robot crear un mapa tridimensional de su ambiente en tiempo real durante su operación.

1.2. Antecedentes

El problema de SLAM tiene una larga historia que se remonta al trabajo de Harris y Pike (1988) donde fue posible estimar el movimiento de una cámara y crear un mapa tridimensional en tiempo real. Ellos usaron inferencia iterativa mediante el filtro de Kalman extendido (EKF). En la década de 1990, la formulación del EKF-SLAM emergió como el enfoque estándar para aplicaciones en robots que operaban en tiempo real (Leonard y Durrant-Whyte, 1991; Betge-Brezetz *et al.*, 1996). Hasta principios del siglo XXI, el EKF aún ocupaba la posición más importante en los sistemas SLAM, pero debido a limitaciones, en particular la complejidad cuadrática en tiempo y memoria del algoritmo con respecto al número de puntos en el mapa, así como la aparición de inconsistencias debido a la linealización de los modelos no lineales de observación

(sensores) y movimiento (cinemática del robot), fue desarrollado un nuevo método basado en grafos tomando gradualmente su lugar. La formulación basada en grafos fue propuesta por Lu y Milios (1997). Sin embargo, tomó muchos años en volverse popular debido a la alta complejidad para resolver el problema de minimización del error en comparación con las técnicas estándar de la época. Ideas recientes sobre la estructura del SLAM y los avances en el campo del álgebra lineal dispersa dieron como resultado enfoques eficientes para resolver el problema de optimización, es por ello que, el enfoque basado en grafos se ha convertido en una de las técnicas más avanzadas con respecto a tiempo de procesamiento y precisión.

Diferentes sensores han sido usados para SLAM, tales como: sonar, láser, cámaras y enfoques multi-sensor. El uso de cámaras se remonta al trabajo de Moravec (1980) implementado en un carro robótico. El algoritmo básico usado por Moravec consiste en identificar puntos de interés en cada imagen proporcionada por la cámara, estimar la profundidad de cada punto, encontrar correspondencias entre puntos de distintas imágenes y luego estimar la transformación de movimiento que mejor alinea los puntos a lo largo del tiempo. Desde entonces, se ha avanzado mucho en todos los aspectos de odometría visual y SLAM. Davison (1998) empleó un robot móvil con un sistema de visión estéreo basado en características visuales, este enfoque integró la información visual con la odometría del robot para construir mapas bidimensionales. La información de odometría fue usada como aproximación inicial de la pose del robot y posteriormente refinada mediante optimización por mínimos cuadrados. Nister *et al.* (2004) aplicaron por primera vez una técnica de optimización que tiene como objetivo minimizar la distancia entre las reproyecciones del modelo tridimensional y los puntos asociados en la imagen. Esta técnica es llamada bundle adjustment (BA) (Triggs *et al.*, 2000) y fue implementada en una ventana deslizante, de esta manera es posible calcular una estimación precisa del movimiento incremental tridimensional en tiempo real, sin embargo, la consistencia global no está asegurada. Grisetti *et al.* (2007) proponen un método eficiente basado en la optimización de un grafo de poses, el cual es muy útil para realizar correcciones cuando se detectan ciclos y así lograr una consistencia global de la estimación.

Un trabajo muy importante en el desarrollo de los sistemas SLAM visual fue PTAM (Parallel Tracking and Mapping) desarrollado por Klein y Murray (2007). Fué el primer

trabajo en introducir la idea de separar el seguimiento de la cámara y el mapeo en hilos paralelos. Un aspecto clave de este sistema es que se extraen cuidadosamente varias imágenes clave (keyframes), los cuales son un subconjunto de todas las imágenes procesadas por el sistema, de tal manera que cubran toda el área de operación del robot. El hilo de mapeo aplica BA minimizando el error de reproyección con el objetivo de optimizar las poses de todos los puntos del mapa y de los keyframes seleccionados. Debido al uso de keyframes, el proceso de optimización solo se realiza cada que un nuevo keyframe es insertado en el hilo de mapeo, de esta manera PTAM puede trabajar en tiempo real solo en áreas pequeñas, dado que, la complejidad del proceso de optimización crece con el número de keyframes.

Strasdat *et al.* (2011) adaptaron el sistema PTAM para realizar el seguimiento de la cámara en un mapa local recuperado de un grafo de covisibilidad en lugar del mapa completo. Propusieron un método de optimización basado en una ventana doble (DWO) que continuamente realiza BA en la ventana interna y optimización del grafo de poses en la ventana externa. Esta idea fue clave para poder usar SLAM en ambientes más grandes.

El primer sistema SLAM que utilizó un sensor RGB-D fue propuesto por Henry *et al.* (2012) quienes desarrollaron RGBD-ICP, un novedoso algoritmo de optimización para calcular la pose del robot. Ellos aprovecharon la ventaja de tener la combinación de la información RGB y de profundidad para poder crear un mapa tridimensional basado en surfels (Pfister *et al.*, 2000). Su algoritmo consiste en extraer características visuales de la imagen RGB usando el algoritmo SIFT (Scale Invariant Feature Transform) (Lowe, 2004) en combinación con RANSAC (Random Sample Consensus) (Fischler y Bolles, 1981) y GICP (Generalized Iterative Closest Point) (Segal *et al.*, 2009) para crear y optimizar un grafo de poses y así lograr una consistencia global del mapa, al igual que Klein y Murray (2007) utilizaron el concepto de keyframe para tener un grafo relativamente disperso. Demostraron que los sensores RGB-D podían ser usados para construir robustos sistemas de mapeo, navegación y sistemas de interacción robóticos. Sin embargo, su implementación no podía correr en tiempo real, probablemente debido al uso de SIFT, ya que es un algoritmo computacionalmente costoso. Posteriormente, este sistema fue mejorado e implementado a bordo de un quadrotor (Huang *et al.*, 2017). Una de las mejoras fue utilizar un detector de características visuales llamado

FAST (Rosten y Drummond, 2006), el cuál es menos costoso computacionalmente en comparación con SIFT, esto permitió que el sistema pudiera operar en tiempo real a bordo del quadrotor. Además, aplicaron un algoritmo distinto a RANSAC para clasificar verdaderas y falsas correspondencias, este nuevo enfoque consiste en generar un grafo de correspondencias consistentes y por medio de un algoritmo voraz aproximar el cliqué máximo en el grafo (Hirschmuller *et al.*, 2002; Howard, 2008). Para escenas estáticas, el conjunto de correspondencias verdaderas genera el cliqué máximo en el grafo. Esta técnica fue comparada con RANSAC y el enfoque del cliqué máximo arrojó mejores resultados.

Paton y Kosecka (2012) mejoraron el algoritmo RGBD-ICP al usarlo de una manera adaptativa y lo llamaron localización RGB-D adaptativa. Los algoritmos adaptativos son aquellos que modifican su conducta durante su ejecución, atendiendo a los cambios que se producen en su entorno. Este enfoque hace uso de la información disponible como: información de ruido de los sensores, error residual en la estimación de movimiento, parámetros estadísticos y modelo de la imagen. Esta información se incorpora en la construcción del algoritmo para ajustarlo en tiempo de ejecución y así alcanzar un mejor desempeño que con algoritmos desarrollados bajo otros enfoques. Este nuevo algoritmo adaptativo primero aplica RANSAC para obtener una estimación inicial de la pose del robot. Después de este paso, el rendimiento de RANSAC es analizado en términos del error de la estimación, así como el total de correspondencias visuales que fueron consideradas como correctas. Si el resultado es satisfactorio entonces la estimación es considerada como correcta. En caso de que el error sea grande se aplica GICP para refinar la estimación. Si el error es muy grande, la estimación es solo realizada por GICP usando un subconjunto aleatorio de los datos RGB-D. Este enfoque resulta en mejores estimaciones ya que aprovecha las ventajas y desventajas de RANSAC y GICP al usarlos alternadamente, además de ser más eficiente ya que no en todos los casos GICP es usado. Sin embargo, su algoritmo tampoco corría en tiempo real, dado que también usaron SIFT para extraer características visuales.

El primer sistema popular de código abierto en usar sensores RGB-D fue el RGB-D SLAM de Endres *et al.* (2014). Este es un sistema basado en características visuales. La arquitectura del sistema consta de tres módulos principales, frontend, backend y representación final del mapa. El módulo frontend calcula el movimiento incremental

de la cámara mediante asociación de características visuales e ICP (Iterative Closest Point) (Besl y McKay, 1992). El módulo backend realiza la optimización del grafo de poses con restricciones de ciclos mediante una búsqueda heurística. El mapa final fue representado mediante vóxeles 3D basados en octrees implementado mediante la librería Octomap (Hornung *et al.*, 2013). Los vóxeles son almacenados en una eficiente estructura de árbol que permite una representación compacta de memoria y crear mapas a diferentes resoluciones. Una ventaja crucial de esta representación en contraste con la basada en puntos, es la representación explícita del espacio libre y áreas no mapeadas lo cual es esencial para evitar colisiones en tareas de exploración y planear rutas en tareas de navegación.

Recientemente, Mur-Artal *et al.* (2015); Mur-Artal y Tardós (2016) introdujeron ORB-SLAM, un eficiente algoritmo basado en características ORB (Oriented FAST and rotated BRIEF) (Rublee *et al.*, 2011) e innovaciones como reutilización del mapa con capacidad para trabajar con cámaras monoculares, sistemas estéreo y sensores RGB-D. Este sistema ha ganado mucha popularidad debido a su alta precisión y robustez, por lo que se encuentra entre los métodos más avanzados para SLAM visual. ORB-SLAM está construido sobre las principales ideas de PTAM (Klein y Murray, 2007) y DWO (Strasdat *et al.*, 2011). El módulo frontend emplea BA para calcular la posición del robot en cada instante del tiempo y lograr una consistencia local del mapa. En el módulo backend, el grafo de poses es optimizado cada vez que se encuentra un ciclo para así lograr una consistencia global de la estimación del mapa.

1.3. Justificación

El problema de SLAM ha tenido un gran progreso en los últimos 30 años. A lo largo de este tiempo se han respondido preguntas importantes, mientras que se han planteado algunas nuevas e interesantes con el desarrollo de nuevas aplicaciones, sensores y herramientas de cómputo.

SLAM y técnicas relacionadas como odometría visual se están desplegando cada vez más en una variedad de entornos del mundo real, desde automóviles sin conductor hasta dispositivos móviles. Se recurrirá cada vez más a las técnicas de SLAM para

proporcionar un posicionamiento métrico confiable en situaciones donde las soluciones brindadas por otros medios como GPS no estén disponibles o no brindan suficiente exactitud. Para muchas aplicaciones y entornos quedan abiertos numerosos desafíos y preguntas importantes. Para lograr una percepción y navegación verdaderamente robusta para robots autónomos se necesita más investigación en SLAM.

Hoy en día, el problema de SLAM se considera resuelto cuando se utilizan sensores de alcance como láser o sonar para construir mapas 2D de pequeños entornos estáticos. Sin embargo, SLAM para entornos dinámicos, complejos y de gran escala, utilizando la visión como único sensor externo, es un área activa de investigación. Las técnicas de visión por computadora empleadas en SLAM visual como la detección, descripción, comparación de características visuales, reconocimiento y recuperación de imágenes, entre otras, aún son susceptibles de mejoras.

De lo anterior se puede concluir que aún queda mucho por hacer en SLAM visual, ya que, constituye un elemento indispensable para la mayoría de aplicaciones robóticas y, a pesar del increíble progreso en las últimas décadas, los sistemas SLAM existentes aún fallan en situaciones desafiantes. Por lo tanto, la investigación llevada a cabo por este proyecto resultará útil para desarrollar sistemas SLAM visuales más robustos, precisos y confiables.

1.4. Hipótesis

El uso de métodos adaptativos para la extracción de características visuales en el problema SLAM visual, así como en el proceso de estimación de movimiento de la cámara, incrementarán la precisión de localización y construcción del mapa del entorno en sistemas SLAM basados en sensores RGB-D, ya que, al incorporar la información del modelo del ruido del sensor RGB-D, error residual de estimación de movimiento y propiedades estadísticas de la imagen, se contribuirá a una mejor toma de decisiones dentro de los algoritmos desarrollados.

1.5. Objetivos

1.5.1. Objetivo general

Desarollar algoritmos adaptativos para mejorar diversas etapas del algoritmo SLAM visual basado en cámaras RGB-D.

1.5.2. Objetivos específicos

- Realizar un estudio comparativo entre los diferentes algoritmos de extracción/descripción de características visuales en imágenes para seleccionar las combinaciones más adecuadas para el problema de SLAM visual.
- Desarrollar un algoritmo adaptativo para mejorar la etapa de extracción y correspondencias de características visuales.
- Desarrollar un algoritmo adaptativo para mejorar el proceso de estimación de movimiento entre imágenes.
- Implementar los algoritmos desarrollados en un sistema SLAM que opere en tiempo real.
- Realizar un estudio comparativo entre los algoritmos desarrollados y el estado del arte, tanto de rendimiento como de precisión y tiempo de procesamiento.

Capítulo 2. Fundamentos

2.1. Conceptos básicos de probabilidad

En SLAM, cantidades como mediciones del sensor, controles, estados del robot y su entorno son modelados como variables aleatorias. Las variables aleatorias pueden tomar múltiples valores y lo hacen de acuerdo a leyes probabilísticas específicas. La inferencia probabilística es el proceso de calcular estas leyes para variables aleatorias que se derivan de otras variables aleatorias y los datos observados. Sea X una variable aleatoria y x un valor específico que X puede tomar. Un ejemplo sencillo de una variable aleatoria es el lanzamiento de una moneda, donde X puede tomar dos valores *sol* o *cruz*. Si el espacio de todos los valores que X puede tomar es discreto, como es el caso si X es el resultado de un lanzamiento de moneda, entonces

$$p(X = x) \tag{1}$$

denota la probabilidad que la variable aleatoria X tome el valor x. Por ejemplo, un lanzamiento de moneda es caracterizado por $p(X = sol) = p(X = cruz) = \frac{1}{2}$. Las probabilidades discretas suman uno, es decir,

$$\sum_x p(X = x) = 1. \tag{2}$$

Las probabilidades son siempre positivas $p(X = x) \geq 0$. Para simplificar la notación una común abreviación es escribir $p(x)$ en lugar de $p(X = x)$.

La mayoría de problemas de estimación y decisión se hacen en el espacio continuo. Estos espacios son caracterizados por variables aleatorias que pueden tener un continuo de valores. Se asume que todas las variables aleatorias continuas poseen una función de densidad de probabilidad (PDF). Una PDF común es la distribución normal con media μ y varianza σ^2, la cual está dada por la siguiente función gausiana

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right), \tag{3}$$

usualmente abreviadas como $x \sim \mathcal{N}(\mu, \sigma^2)$, lo cual indica que x es variable aleatoria está distribuida normalmente con media μ y varianza σ^2. La distribución normal (Ec. 3)

asume que x es un valor escalar, sin embargo, en SLAM x es frecuentemente un vector n-dimensional, denotado por $\mathbf{x}$. Distribuciones normales sobre vectores son llamadas multivariadas, las cuales son caracterizadas por la siguiente PDF

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{n/2}|\,\Sigma\,|^{1/2}}\ \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right), \tag{4}$$

donde μ es el vector de media y Σ es la matriz de covarianza y $|\,\Sigma\,|$ su determinante.

Al igual que en el caso discreto, la PDF en el caso continuo siempre integra a 1

$$\int p(x)\ dx = 1. \tag{5}$$

La distribución conjunta de dos variables aleatorias X y Y está dada por

$$p(x, y) = p(X = x \text{ y } Y = y). \tag{6}$$

Esta expresión describe la probabilidad de que la variable aleatoria X tome el valor x y que Y tome el valor y. Si X y Y son independientes, entonces

$$p(x, y) = p(x)p(y). \tag{7}$$

A menudo, las variables aleatorias contienen información sobre otras variables aleatorias. Por ejemplo, si se sabe que el valor de Y es y y se quiere conocer la probabilidad de que el valor de X sea x condicionado a ese hecho. Tal probabilidad se denota

$$p(x\,|\,y) = p(X = x\,|\,Y = y) \tag{8}$$

y es llamada probabilidad condicional. Si $p(y) > 0$, entonces la probabilidad condicional se define como

$$p(x\,|\,y) = \frac{p(x, y)}{p(y)}. \tag{9}$$

Por lo que, si X y Y son independientes, entonces

$$p(x\,|\,y) = \frac{p(x)p(y)}{p(y)} = p(x). \tag{10}$$

En otras palabras, si X y Y son independientes, Y no brinda ningún tipo de información acerca del valor de X. Un hecho interesante que se deriva de la definición de probabilidad condicional se denomina teorema de probabilidad total

$$p(x) = \sum_{y} p(x \mid y)p(y) \quad \text{Caso discreto} \tag{11}$$

$$p(x) = \int p(x \mid y)p(y)\, dy \quad \text{Caso continuo.} \tag{12}$$

Igualmente importante es la regla de Bayes, la cual relaciona una probabilidad condicional del tipo $p(x \mid y)$ con su inversa $p(y \mid x)$, esta regla requiere que $p(y) > 0$

$$p(x \mid y) = \frac{p(y \mid x)p(x)}{p(y)} = \frac{p(y \mid x)p(x)}{\sum_{x'} p(y \mid x')p(x')} \quad \text{Discreto} \tag{13}$$

$$p(x \mid y) = \frac{p(y \mid x)p(x)}{p(y)} \frac{p(y \mid x)p(x)}{\int p(y \mid x')p(x')\, dx'} \quad \text{Continuo.} \tag{14}$$

La regla de Bayes juega un papel dominante en inferencia probabilística. Si x es una cantidad que se desea inferir a partir de y, la probabilidad $p(x)$ es conocida como distribución de probabilidad a priori y y es llamada datos (por ejemplo, mediciones del sensor). La distribución $p(x)$ resume el conocimiento que se tiene sobre X antes de incorporar los datos y. La probabilidad $p(x \mid y)$ es conocida como distribución de probabilidad a posteriori sobre X. La regla de Bayes provee una manera de calcular la distribución a posteriori $p(x \mid y)$ usando su probabilidad condicional inversa $p(y \mid x)$ junto con la distribución a priori $p(x)$. En otras palabras, si se desea inferir la cantidad x a partir de los datos del sensor y, la regla de Bayes permite hacerlo a través de la probabilidad inversa, la cual específica la probabilidad de los datos y asumiendo que x es el caso. En robótica, la probabilidad $p(y \mid x)$ es el modelo generativo, ya que, describe como las variables de estado X causan la medición del sensor Y. Una importante observación es que el denominador de la regla de bayes, $p(y)$, no depende de x. Por lo tanto, el factor $p(y)^{-1}$ en las Ec. 13 y 14 será el mismo para cualquier valor x en la distribución a posterior $p(x \mid y)$. Por esta razón, $p(y)^{-1}$ es denotado como una variable normalizadora y genericamente denotada por η

$$p(x \mid y) = \eta\ p(x \mid y)p(x). \tag{15}$$

Las reglas discutidas hasta ahora pueden ser aplicadas a variables aleatorias arbitrarias, como la variable Z. Por ejemplo, condicionar la regla de Bayes en $Z = z$ resulta

$$p(x \mid y, z) = \frac{p(y \mid x, z)p(x \mid z)}{p(y \mid z)} \tag{16}$$

mientras que $p(y \mid z) > 0$. De manera similar se puede condicionar la regla para combinar probabilidades de variables aleatorias independientes (Ec. 7) en otras variables z

$$p(x, y \mid z) = p(x \mid z)p(y \mid z). \tag{17}$$

Esta relación es conocida como independencia condicional y juega un rol muy importante en estimación probabilística. Aplica cuando una variable y no contiene información sobre una variable x si el valor de otra variable z es conocido.

Muchos algoritmos probabilísticos requieren calcular propiedades estadísticas sobre distribuciones de probabilidad. El valor esperado de una variable aleatoria X está dado por

$$\begin{aligned} E[X] &= \sum_x xp(x) \qquad (Discreto) \\ E[X] &= \int xp(x)\, dx \quad (Continuo). \end{aligned} \tag{18}$$

El valor esperado es una función lineal de una variable aleatoria, por lo tanto

$$E[aX + b] = a\, E[X] + b \tag{19}$$

para cualquier par de valores a y b. La covarianza de X es obtenida de la siguiente manera

$$Cov[X] = E[X - E[X]]^2 = E[X^2] - E[X]^2. \tag{20}$$

La covarianza mide la desviación cuadrática esperada de la media.

2.2. Optimización

Un problema de optimización es generalmente definido como encontrar el mínimo de una función objetivo o función de costo $F(\mathbf{x})$, es decir, encontrar el valor de la

variable $\mathbf{x}^*$ donde ese mínimo ocurre. Formalmente un problema de optimización se define como

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} F(\mathbf{x}). \tag{21}$$

Un valor crítico (mínimo o un máximo) de una función F ocurre cuando su primera derivada F' es igual a cero. En general, la función objetivo es una función: $F : \mathbb{R}^n \rightarrow \mathbb{R}$, donde $n \geq 1$. Los problemas de optimización se puede clasificar de acuerdo a las siguientes propiedades:

- **Global vs local**. En el mejor de los casos, la función objetivo $F(\mathbf{x})$ tiene un mínimo único, el cual es llamado mínimo global. Formalmente, $\mathbf{x}^*$ es un mínimo global de F si

  $$F(\mathbf{x}^*) < F(\mathbf{y}) \forall \mathbf{y}. \tag{22}$$

  Las funciones también pueden tener múltiples mínimos locales. $\mathbf{x}^*$ es un mínimo local de F si minimiza la función dentro de una región ϵ

  $$F(\mathbf{x}^*) \leq F(\mathbf{y}) \forall \mathbf{y} : \| \mathbf{y} - \mathbf{x}^* \| < \epsilon. \tag{23}$$

- **Convexo vs no convexo**. La convexidad de una función de costo es un criterio importante sobre qué tan fácil puede resolverse el problema de optimización asociado. Una función convexa tiene un único mínimo, la cual es una propiedad muy deseable.

- **Restringido vs no restringido**. El problema de optimización definido en la Ec. 21 es un problema no restringido, ya que, la búsqueda de la solución $\mathbf{x}^*$ no está sujeta a ninguna condición.

- **Continuo vs discreto**. Muchos problemas importantes en ciencias de la computación son problemas de optimización discretos, dado que, las variables $\mathbf{x}$ están definidas sobre un conjunto discreto.

2.2.1. Mínimos cuadrados

En optimización por mínimos cuadrados, la función objetivo es una suma sobre los términos cuadrados, es decir

$$F(\mathbf{x}) = \frac{1}{2}\sum_{i}^{n} f_i(\mathbf{x})^2. \tag{24}$$

Esta formulación de la función objetivo es muy común en problemas de róbotica y visión por computadora. El problema de SLAM se puede describir como un problema de optimización por mínimos cuadrados. La ecuación anterior se puede reformular usando notación vectorial

$$F(\mathbf{x}) = \frac{1}{2}\mathbf{f}(\mathbf{x})^T\mathbf{f}(\mathbf{x}). \tag{25}$$

Entonces el problema de optimización por mínimos cuadrados es

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}}\, F(\mathbf{x}) = \underset{\mathbf{x}}{\operatorname{argmin}}\, \frac{1}{2}\mathbf{f}(\mathbf{x})^T\mathbf{f}(\mathbf{x}). \tag{26}$$

2.2.2. Mínimos cuadrados lineal

En el caso de un problema de mínimos cuadrados lineal, todos los f_i depende linealmente de $\mathbf{x}$, es decir, el valor de la función es una combinación lineal de las variables en $\mathbf{x}$

$$f_i(\mathbf{x}) = a_{i,1}x_1 + a_{i,2}x_2 + \ldots + a_{i,n}x_n - b_i. \tag{27}$$

Dada la ecuación anterior, es posible recopilar cada f_i en un vector columna $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_n(\mathbf{x}))^T$. Entonces, recolectando todos los coeficientes $a_{i,j}$ en la matriz $\mathbf{A}$ y todos los b_i en el vector $\mathbf{b}$, la ecuación anterior se puede se simplificar como

$$\mathbf{f}(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b}. \tag{28}$$

Para poder resolver el problema, primero se determina la ubicación de los puntos críticos para $F(\mathbf{x})$ calculando su primer derivada $F'(\mathbf{x})$:

$$F'(\mathbf{x}) = (\mathbf{A}\mathbf{x} - \mathbf{b})^T \cdot \mathbf{A}. \tag{29}$$

La posición del punto crítico $\mathbf{x}^*$ se encuentra al igualar la primer derivada a cero

$$\begin{aligned} 0 &= F'(\mathbf{x}^*) \\ \mathbf{A}^T\mathbf{A}\mathbf{x}^* &= \mathbf{A}^T\mathbf{b}. \end{aligned} \tag{30}$$

La ecuación anterior es llamada ecuación normal, la cual puede ser resuelta usando métodos diferentes:

1. Si $\mathbf{A}^T\mathbf{A}$ es invertible, la solución es directamente $\mathbf{x}^* = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$.
2. Una mejor alternativa usa descomposición QR de $\mathbf{A}$ en un matriz triangular superior $\mathbf{R}$ y una matriz ortogonal $\mathbf{Q}$, de modo que $\mathbf{A} = \mathbf{QR}$. La solución $\mathbf{x}^*$ está dada por $\mathbf{x}^* = \mathbf{R}^{-1}\mathbf{Q}^T\mathbf{b}$. Este método evita calcular la inversa, lo cual puede ser numéricamente inestable.
3. Mediante descomposición de Cholesky. Este método descompone $\mathbf{A}^T\mathbf{A}$ en $\mathbf{LL}^T$, después resuelve $\mathbf{Ly} = \mathbf{A}^T\mathbf{b}$ para $\mathbf{y}$ y encuentra $\mathbf{x}^*$ resolviendo $\mathbf{L}^T\mathbf{x} = \mathbf{y}$. Comparado con descomposición QR, Cholesky es más rápido pero numéricamente más inestable dado que $\mathbf{A}^T\mathbf{A}$ tiene que ser calculado.

2.2.3. Mínimos cuadrados no lineal

Cuando las funciones f_i no pueden ser expresadas como una combinación lineal de $\mathbf{x}$, el problema no puede ser resuelto directamente mediante la ecuación normal 30, por lo que requiere el uso de técnicas iterativas. Empezando por un valor inicial de la solución $\mathbf{x}_0$, estos métodos convergen hacia $\mathbf{x}^*$, la cual se espera que sea la solución óptima global. Sin embargo, esto solo puede ser garantizado si la función es convexa o quasi-convexa. En el caso general, la solución encontrada $\mathbf{x}^*$ puede ser solo un mínimo local de la función objetivo, y a partir de diferentes valores iniciales $\mathbf{x}_0$, se puede encontrar diferentes mínimos locales $\mathbf{x}^*$.

Existen muchos enfoques iterativos para resolver el problema no lineal de mínimos cuadrados. La primer derivada (Jacobiano) de la función objetivo F es

$$F(\mathbf{x})' = \mathbf{J}_F = \mathbf{J}_\mathbf{f}^T\mathbf{f}(\mathbf{x}). \tag{31}$$

La segunda derivada de F (Hesiano) está dada por

$$F(\mathbf{x})'' = \mathbf{H}_F = \mathbf{J}_F' = \mathbf{H}_\mathbf{f}^T \mathbf{f}(\mathbf{x}) + \mathbf{J}_\mathbf{f}^T \mathbf{J}_\mathbf{f}. \tag{32}$$

2.2.3.1. Descenso de gradiente

Descenso de gradiente es una de las estrategias iterativas más sencillas para encontrar un mínimo local de una función de costo diferenciable $F(\mathbf{x})$. De la definición de gradiente

$$\nabla F(\mathbf{x}) = F'(\mathbf{x}) = \frac{\partial F(\mathbf{x})}{\partial \mathbf{x}} = \mathbf{J}_F. \tag{33}$$

La función F decrece más rápidamente en la dirección del gradiente negativo, por lo tanto

$$\Delta \mathbf{x} = -\mathbf{J}_F \tag{34}$$

es la dirección de descenso. La idea de la estrategia de descenso de gradiente es moverse repetidamente en la dirección del gradiente negativo hasta la convergencia. Sin embargo, usar directamente el gradiente como paso de actualización $\mathbf{x}_{n+1} = \mathbf{x}_n - \mathbf{J}_F$ no garantiza que $F(\mathbf{x}_{n+1}) < F(\mathbf{x}_n)$. Por lo tanto, la longitud del paso α debe determinarse de modo que $F(\mathbf{x}_n + \alpha \Delta \mathbf{x}) < F(\mathbf{x}_n)$. Esto se puede hacer buscando a lo largo de la línea determinada por la dirección del gradiente hasta que se encuentre un valor adecuado de α. Finalmente, el paso del descenso de gradiente es

$$\Delta \mathbf{x} = -\alpha \mathbf{J}_F = -\alpha \mathbf{J}_\mathbf{f}^T \mathbf{f}(\mathbf{x}). \tag{35}$$

El algoritmo de descenso de gradiente se puede resumir en los siguientes pasos:

1. Encontrar la dirección de descenso $-\mathbf{J}_F$ al actual $\mathbf{x}_n$.
2. Determinar la longitud del paso α.
3. Repetir $\mathbf{x}_{n+1} = \mathbf{x}_n - \alpha \mathbf{J}_F$ hasta la convergencia.

Si bien la estrategia de descenso de gradiente es fácil de implementar y garantiza

la convergencia a un mínimo de la función objetivo, existen métodos más sofisticados que pueden acelerar significativamente la convergencia.

2.2.3.2. Método de Newton

El método de Newton puede ser aplicado a cualquier problema de optimización no lineal siempre y cuando la función objetivo F sea doble diferenciable[1]. Para el caso multi-dimensional, el paso de Newton puede ser calculado mediante la solución a la siguiente ecuación

$$\begin{aligned} \mathbf{H}_F \Delta\mathbf{x} &= -\mathbf{J}_F \\ (\mathbf{H}_\mathbf{f}^T \mathbf{f}(\mathbf{x}) + \mathbf{J}_\mathbf{f}^T \mathbf{J}_\mathbf{f}) \Delta\mathbf{x} &= -\mathbf{J}_\mathbf{f}^T \mathbf{f}(\mathbf{x}). \end{aligned} \tag{36}$$

El método de Newton converge más rápido que el descenso de gradiente, ya que, hace uso de la información de curvatura codificada en el Hesiano $\mathbf{H_f}$. Aunque converge rápido hacia la solución, requiere de la segunda derivada, la cual puede ser difícil de calcular.

2.2.3.3. Gauss-Newton

El algoritmo se deriva aplicando una linealización de primer orden mediante la serie de Taylor de $\mathbf{f(x)}$ alrededor de $\mathbf{x}$

$$\mathbf{f}(\mathbf{x} + \Delta\mathbf{x}) \approx \mathbf{f}(\mathbf{x}) + \mathbf{J_f} \Delta\mathbf{x}, \tag{37}$$

sustituyendo se obtiene

$$F(\mathbf{x} + \Delta\mathbf{x}) \approx L_\mathbf{x}(\Delta\mathbf{x}) = F(\mathbf{x}) + \mathbf{f}(\mathbf{x})^T \mathbf{J_f} \Delta\mathbf{x} + \frac{1}{2} \Delta\mathbf{x}^T \mathbf{J}_\mathbf{f}^T \mathbf{J_f} \Delta\mathbf{x}. \tag{38}$$

Dada esta aproximación a la función objetivo en la vecindad de $\mathbf{x}$, se busca el $\Delta\mathbf{x}$ que la minimice. Debido a que $L_\mathbf{x}(\Delta\mathbf{x})$ es cuadrática en $\Delta\mathbf{x}$ solo tiene un mínimo global que

[1] Una función doble diferenciable es aquella que tiene primera y segunda derivada.

se puede encontrar directamente al igualar la primer derivada a cero

$$\begin{aligned} 0 &= L'_{\mathbf{x}}(\Delta\mathbf{x}) \\ \mathbf{J}_{\mathbf{f}}^T\mathbf{J}_{\mathbf{f}}\Delta\mathbf{x} &= -\mathbf{J}_{\mathbf{f}}^T\mathbf{f}(\mathbf{x}). \end{aligned} \tag{39}$$

Esta es llamada ecuación normal (Ec. 30) y puede ser resuelta usando alguno de los métodos presentados para problemas lineales. Es interesante notar que el lado derecho de la ecuación anterior, $\mathbf{J}_{\mathbf{f}}^T\mathbf{f}(\mathbf{x})$, es igual a la derivada de la función objetivo F (Ec. 31). Además, comparando el lado izquierdo con el método de Newton (Ec. 36), se observa que la diferencia principal entre ambos enfoques es que Gauss-Newton aproxima el Hesiano de la función objetivo mediante $\mathbf{J}_{\mathbf{f}}^T\mathbf{J}_{\mathbf{f}}$. De esta manera se evita el cálculo de segundas derivadas lo cual puede ser difícil para muchos problemas no triviales.

2.2.3.4. Levenberg-Marquardt

De los tres métodos anteriores, solo el paso $\Delta\mathbf{x} = \alpha\mathbf{J}_F$ de descenso de gradiente garantiza disminuir el valor de la función objetivo $F(\mathbf{x})$. Aunque el Método de Newton converge muy rápido cuando se está cerca de la solución, puede fallar cuando se está lejos del mínimo. Lo mismo aplica para Gauss-Newton, cuya única diferencia es la aproximación del Hesiano a través del cuadrado del Jacobiano.

El método conocido como Levenberg-Marquardt aborda este problema cambiando gradualmente entre descenso de gradiente y la estrategia de Gauss-Newton. Esto se logra introduciendo un parámetro λ, llamado factor de amortiguación y resolviendo el siguiente problema para encontrar el paso de descenso

$$(\mathbf{J}_{\mathbf{f}}^T\mathbf{J}_{\mathbf{f}} + \lambda\mathbf{I})\Delta\mathbf{x} = -\mathbf{J}_{\mathbf{f}}^T\mathbf{f}(\mathbf{x}). \tag{40}$$

Si la solución $\Delta\mathbf{x}$ al problema anterior resulta en un decremento de la función objetivo, se acepta la solución, disminuye λ y el algoritmo continúa $\mathbf{x}_{n+1} = \mathbf{x}_n + \Delta\mathbf{x}$. En caso contrario se incrementa λ y (Ec. 40) se resuelve de nuevo hasta que se encuentre un valor de λ que resulte en un decremento.

Para valores muy grandes de λ, el algoritmo se comporta como descenso de gra-

diente, ya que el término izquierdo en la Ec. 40 es dominado por $\lambda \mathbf{I}$, por lo tanto

$$\Delta \mathbf{x} \approx -\frac{1}{\lambda} \mathbf{J}_{\mathbf{f}}^T \mathbf{f}(\mathbf{x}) = -\frac{1}{\lambda} \mathbf{J}_F, \tag{41}$$

lo que resulta en un pequeño paso a lo largo del gradiente negativo de F. Por otro lado, si λ es pequeño

$$\mathbf{J}_{\mathbf{f}}^T \mathbf{J}_{\mathbf{f}} \Delta \mathbf{x} = -\mathbf{J}_{\mathbf{f}} \mathbf{f}(\mathbf{x}), \tag{42}$$

es igual al paso de Gauss-Newton (Ec. 39). De esta manera, adaptando el parámetro λ, se puede cambiar el comportamiento de Levenberg-Marquardt, esto lo hace un enfoque de optimización dominante. Para valores iniciales de λ, Hartley y Zisserman (2003) proponen usar 10^{-3} veces el promedio de $diag(\mathbf{J}_{\mathbf{f}}^T \mathbf{J}_{\mathbf{f}})$.

Todas las estrategias anteriores son incrementales, pero cada una de ellas aplica diferentes pasos de actualización para encontrar la siguiente mejor configuración de variables en el descenso de la función objetivo. En la siguiente tabla se muestra los pasos de actualización para los diferentes enfoques.

Tabla 1. Pasos para las diferentes estrategias iterativas para problemas no lineales de mínimos cuadrados.

Método	Paso $\Delta \mathbf{x}$
Descenso de gradiente	$\Delta \mathbf{x} = -\alpha \mathbf{J}_{\mathbf{f}}^T \mathbf{f}(\mathbf{x})$
Método de Newton	$(\mathbf{H}_{\mathbf{f}}^T \mathbf{f}(\mathbf{x}) + \mathbf{J}_{\mathbf{f}}^T \mathbf{J}_{\mathbf{f}}) \Delta \mathbf{x} = -\mathbf{J}_{\mathbf{f}} \mathbf{f}(\mathbf{x})$
Gauss-Newton	$\mathbf{J}_{\mathbf{f}}^T \mathbf{J}_{\mathbf{f}} \Delta \mathbf{x} = -\mathbf{J}_{\mathbf{f}}^T \mathbf{f}(\mathbf{x})$
Levenberg-Marquardt	$(\mathbf{J}_{\mathbf{f}}^T \mathbf{J}_{\mathbf{f}} + \lambda \mathbf{I}) \Delta \mathbf{x} = -\mathbf{J}_{\mathbf{f}}^T \mathbf{f}(\mathbf{x})$

2.2.4. Mínimos cuadrados no lineal con pesos

En los enfoques de mínimos cuadrados presentados anteriormente, todos los datos y funciones objetivo tienen la misma influencia en el resultado de la optimización. Sin embargo, en muchas aplicaciones, se proporciona información adicional que permite ponderar los datos individuales de manera diferente, por lo que algunos tendrán más influencia que otros en el resultado de la optimización.

Los pesos vienen codificados en la matriz de covarianza Σ_i para las mediciones $\mathbf{y}_i$,

entonces se puede usar la distancia de Mahalanobis en lugar de la distancia Euclidiana cuadrática para definir la función objetivo. La distancia cuadrática de Mahalanobis se define como

$$\| \mathbf{x}_i - \mathbf{y}_i \|^2_{\Sigma_i} = (\mathbf{x}_i - \mathbf{y}_i)^T \Sigma_i^{-1} (\mathbf{x}_i - \mathbf{y}_i). \tag{43}$$

Aplicando la definición de distancia de Mahalanobis se puede definir la función objetivo ponderada

$$F = \frac{1}{2} \sum_i \| \mathbf{x}_i - \mathbf{y}_i \|^2_{\Sigma_i} = \frac{1}{2} \mathbf{f}(\mathbf{x})^T \Sigma^{-1} \mathbf{f}(\mathbf{x}) = \frac{1}{2} \mathbf{f}(\mathbf{x})^T \Omega \mathbf{f}(\mathbf{x}), \tag{44}$$

donde Ω es la inversa de la matriz de covarianza (llamada matriz de información), la cual sirve como una matriz de pesos que pondera cada uno de los términos f_i. El procedimiento para encontrar la solución óptima es similar al de mínimos cuadrados no lineal, en particular, si se aplica la estrategia de Levenberg-Marquard, se obtiene

$$(\mathbf{J}_\mathbf{f}^T \Omega \mathbf{J}_\mathbf{f} + \lambda \mathbf{I}) \Delta \mathbf{x} = -\mathbf{J}_\mathbf{f}^T \Omega^T \mathbf{f}(\mathbf{x}). \tag{45}$$

El problema de SLAM visual puede representarse como un problema de optimización no lineal ponderado, por lo tanto, en este trabajo se hace uso del método de Levenberg-Marquardt para llevar a cabo todos los procesos de optimización, ya que, el factor de amortiguamiento λ se ajusta en cada iteración para comportarse como el algoritmo de Gauss-Newton o descenso de gradiente. Esta característica lo hace más apropiado para aplicaciones prácticas ya que aprovecha la ventaja de ambas técnicas. Además se utilizó descomposición de Cholesky para resolver la ecuación normal, debido a su rapidez de cómputo.

2.3. Formulación probabilística de SLAM

Muchas tareas en robótica involucran problemas de naturaleza estadística. Los robots perciben el mundo a través de sensores e interactúan en el mediante de actuadores, estos procesos están inherentemente sujetos al ruido, por lo tanto, tratar con la incertidumbre es fundamental para muchos algoritmos. Los métodos probabilísticos

nos permiten modelar la naturaleza estadística de las mediciones del robot, así como la incertidumbre sobre el estado del entorno.

Dado que las poses[2] del robot y el mapa no son conocidos a priori, tienen que ser estimados usando la información de los sensores, la cual inevitablemente contendrá ruido. A continuación, se presenta una caracterización de SLAM por su formulación probabilística (Thrun *et al.*, 2005).

La pose del robot al instante de tiempo t es descrita como $\mathbf{x}_t$. Para un robot que opera en un plano bidimensional, la pose consiste de $\mathbf{x} = (x, y, \theta)^T$, que describe la ubicación actual y la orientación del robot en relación con un marco de referencia global. Para el caso tridimensional, una posible parametrización es, $\mathbf{x} = (x, y, z, \phi, \vartheta, \psi)$, donde (ϕ, ϑ, ψ) son los ángulos de Euler que representan la orientación del robot en el espacio. Toda la trayectoria del robot hasta el instante de tiempo T está dada por

$$\mathbf{x}_{0:T} = \{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T\} \tag{46}$$

la cual describe la ruta tomada por el robot como una secuencia de poses, donde T denota el fin de la secuencia. Mientras el robot se mueve por ese camino, sus odómetros le permiten medir su movimiento $\mathbf{u}_t$, la cual representa el movimiento entre $t-1$ y t. La secuencia de mediciones de odometría está dada por

$$\mathbf{u}_{1:T} = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_T\}. \tag{47}$$

La relación entre dos poses sucesivas y su respectiva medición de odometría está dada por una función generalmente no lineal llamada modelo de movimiento $g(\cdot)$

$$\mathbf{x}_t = g(\mathbf{x}_{t-1}, \mathbf{u}_t) + \mathbf{w}_t. \tag{48}$$

El ruido del odómetro es usualmente modelado como una distribución gausiana con media cero y covarianza $\mathbf{\Lambda}_t$, el cual es capturado por el término $\mathbf{w}_t$

$$\mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{\Lambda}_t), \tag{49}$$

[2] El termino *pose* significa *posición* junto con *orientación*.

por lo tanto, $\mathbf{x}_t$ también es modelado como una distribución gausiana

$$\mathbf{x}_t \sim \mathcal{N}(g(\mathbf{x}_{t-1}, \mathbf{u}_t), \mathbf{\Lambda}_t). \tag{50}$$

Mientras que $\mathbf{u}_t$ almacena la entrada de control o lectura de odometría, los robots normalmente están equipados con un sensor que le proporciona información sobre el entorno. Se denota el mapa del entorno con $\mathbf{m}$ y las mediciones del sensor por $\mathbf{s}_t$. Como anteriormente, la secuencia de mediciones obtenidas por el sensor se denota por

$$\mathbf{s}_{1:T} = \{\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_T\}. \tag{51}$$

El modelo del sensor $f(\cdot)$ permite predecir las mediciones del sensor dada la estimación actual del mapa y las poses del robot, al igual que el modelo de movimiento, usualmente es una función no lineal en aplicaciones prácticas

$$\mathbf{s}_t = f(\mathbf{x}_t, \mathbf{m}) + \mathbf{v}_t, \tag{52}$$

donde $\mathbf{v}_t$ captura el ruido del sensor y la incertidumbre, usualmente modelada por una distribución gausiana con media cero

$$\mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{\Gamma}_t) \tag{53}$$

y, por lo tanto

$$\mathbf{s}_t \sim \mathcal{N}(f(\mathbf{x}_t, \mathbf{m}), \mathbf{\Gamma}_t). \tag{54}$$

Desde una perspectiva probabilística, existen dos formas principales del problema de SLAM. Una es conocida como el problema en línea, el cual estima la distribución conjunta de la pose actual y el mapa

$$p(\mathbf{x}_t, \mathbf{m} \mid \mathbf{s}_{1:t}, \mathbf{u}_{1:t}). \tag{55}$$

Este problema es llamado SLAM en línea ya que solo estima las variables que persisten al tiempo t. Muchos algoritmos para el problema en línea son incrementales, es decir, descartan las mediciones pasadas una vez que han sido procesadas. Las técnicas

basadas en filtros son ampliamente usadas para resolver este problema.

El segundo problema es llamado SLAM completo. Resolverlo consiste en estimar la probabilidad conjunta sobre toda la trayectoria del robot $\mathbf{x}_{0:T}$ y el mapa del entorno $\mathbf{m}$ dadas todas las mediciones del sensor $\mathbf{s}_{1:T}$ y las mediciones de odometría $\mathbf{u}_{1:T}$

$$p(\mathbf{x}_{0:T}, \mathbf{m} \mid \mathbf{s}_{1:T}, \mathbf{u}_{1:T}). \tag{56}$$

Las técnicas basadas en optimización son usadas para resolver el problema completo.

La diferencia principal entre enfoques basados en filtros y optimización es la manera de representar los estados internos.

2.3.1. SLAM basado en filtros

Los métodos basados en filtros estiman y continuamente actualizan la distribución conjunta de la Ec. 55. Nuevas mediciones son usadas para actualizar dicha distribución (reducir incertidumbre), mientras que la progresión del tiempo agrega nuevos parámetros con una gran incertidumbre inicial o aumenta la incertidumbre de los existentes. Este enfoque es conocido como filtrado de Kalman. Históricamente, el algoritmo de SLAM más antiguo y quizás el más influyente, se basa en el filtro de Kalman Extendido o EKF-SLAM. Este enfoque permite marginalizar las variables de estado antiguas, como consecuencia, solo se mantiene la posición actual del robot en el vector de estados.

Los mapas, en EKF-SLAM, son basados en características, están compuestos de puntos de referencia. Por razones computacionales, el número de puntos de referencia suele ser pequeño (menor a 1000), debido a que la complejidad de este enfoque crece de manera cuadrática con el número de puntos de referencia. Además, el enfoque EKF tiende a funcionar bien mientras menos ambiguedad presenten los puntos de referencia, por esta razón requiere algoritmos robustos para detectar características, a veces usando puntos de referencia artificiales bien definidos en el mapa como características.

Como cualquier algoritmo EKF, EKF-SLAM asume ruido gausiano para el movimiento

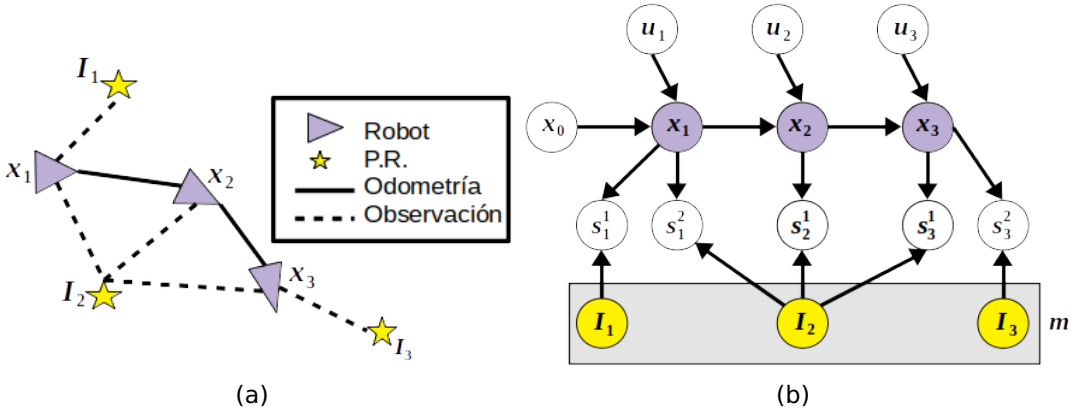


Figura 1. Ejemplo del problema SLAM y su correspondiente red Bayesiana. (a) El robot navega sobre un entorno y detecta puntos de referencia (P.R.) con su sensor a bordo. (b) La red Bayesiana correspondiente al problema de SLAM.

y percepción del robot. La cantidad de incertidumbre en la distribución a posteriori debe ser relativamente pequeña, ya que, de lo contrario, la linealización en el EFK tiende a introducir errores intolerables.

En práctica, EKF SLAM ha sido aplicado con éxito a un número considerable de problemas de mapeo en robótica. Cuando los puntos de referencia son lo suficientemente distintos, este enfoque aproxima muy bien la distribución a posteriori. Sin embargo, el algoritmo EKF SLAM adolece de su enorme complejidad de actualización y la limitación a mapas dispersos. Esto hace que el problema de asociación de datos sea difícil y EKF SLAM tiende a funcionar mal en situaciones donde los puntos de referencia son altamente ambiguos.

2.3.2. SLAM basado en grafos

Los métodos basados en optimización modelan la información como una función de energía no lineal, la cual es continuamente optimizada. La idea de este enfoque se inspira en la representación por medio de grafos del problema SLAM. La mayoría de estas técnicas abordan el problema de SLAM completo. La idea básica es la siguiente, puntos de referencia y poses del robot representan nodos del grafo. Cada par consecutivo de estados está conectado entre si por una arista que representa la información de odometría $\mathbf{u}_t$. Existen otras aristas entre los puntos de referencia $\mathbf{s}_t^i$ y posiciones

del robot $\mathbf{x}_t$, si en el tiempo t, el robot detecta el punto de referencia i. Las aristas son consideradas como restricciones, al ajustar estas restricciones se obtiene una mejor estimación de los estados internos. Un ejemplo de esta representación se muestra en la Fig. 1.

En el trabajo de Strasdat *et al.* (2010), los autores llegaron a la conclusión que el enfoque basado en filtros presenta mejor precisión en problemas pequeños, mientras que los métodos de optimización logran una mejor precisión para problemas más grandes, usando significativamente más puntos de referencia.

El enfoque basado en grafos resuelve el problema de SLAM completo, donde la distribución conjunta forma un grafo disperso, el cual conduce a una suma de restricciones no lineales. Optimizar estas restricciones resulta en un problema máximo a posteriori (MAP) mediante minimización por mínimos cuadrados.

La aplicación de la regla de Bayes a la Ec. 56 permite factorizar la probabilidad al instante de tiempo t:

$$p(\mathbf{x}_{0:t}, \mathbf{m} \mid \mathbf{s}_{1:t}, \mathbf{u}_{1:t}) = \eta p(\mathbf{s}_t \mid \mathbf{x}_{0:t}, \mathbf{m}, \mathbf{s}_{1:t-1}, \mathbf{u}_{1:t}) p(\mathbf{x}_{0:t}, \mathbf{m} \mid \mathbf{s}_{1:t-1}, \mathbf{u}_{1:t}), \tag{57}$$

donde η es un normalizador que toma en cuenta el denominador omitido en el lado derecho de la ecuación. Además de la suposición Gausiana, se asume que el estado $\mathbf{x}_t$ contiene toda la información relevante hasta el tiempo t, esto se conoce como supuesto de Markov y permite simplificar el primer término en la Ec. 57

$$p(\mathbf{s}_t \mid \mathbf{x}_{0:t}, \mathbf{m}, \mathbf{s}_{1:t-1}, \mathbf{u}_{1:t}) \overset{Markov}{=} p(\mathbf{s}_t \mid \mathbf{x}_t, \mathbf{m}). \tag{58}$$

Además, separando $\mathbf{x}_{0:t}$ en $\mathbf{x}_{0:t-1}$ y $\mathbf{x}_t$ en el segundo término de la Ec. 57, se obtiene

$$p(\mathbf{x}_{0:t}, \mathbf{m} \mid \mathbf{s}_{1:t-1}, \mathbf{u}_{1:t}) = p(\mathbf{x}_t \mid \mathbf{x}_{0:t-1}, \mathbf{m}, \mathbf{s}_{1:t-1}, \mathbf{u}_{1:t}) p(\mathbf{x}_{0:t-1}, \mathbf{m} \mid \mathbf{s}_{1:t-1}, \mathbf{u}_{1:t}) \tag{59}$$

$$\overset{Markov}{=} p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t) p(\mathbf{x}_{0:t-1}, \mathbf{m} \mid \mathbf{s}_{1:t-1}, \mathbf{u}_{1:t-1}). \tag{60}$$

Sustituyendo las Ec. 58 y 60 en la Ec. 57 se obtiene

$$p(\mathbf{x}_{0:t}, \mathbf{m} \mid \mathbf{s}_{1:t}, \mathbf{u}_{1:t}) = \eta p(\mathbf{s}_t \mid \mathbf{x}_t, \mathbf{m}) p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t) p(\mathbf{x}_{0:t-1}, \mathbf{m} \mid \mathbf{s}_{1:t-1}, \mathbf{u}_{1:t-1}), \tag{61}$$

la cual es una definición recursiva, donde la primer posición se especifica mediante la probabilidad $p(\mathbf{x}_0)$, la cual es la posición del robot en el marco de referencia global. La distribución de probabilidad a posteriori en forma cerrada está dada por

$$p(\mathbf{x}_{0:t}, \mathbf{m} \mid \mathbf{s}_{1:t}, \mathbf{u}_{1:t}) = \eta p(\mathbf{x}_0) \prod_t p(\mathbf{s}_t \mid \mathbf{x}_t, \mathbf{m}) p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t). \tag{62}$$

En la Fig. 1 se muestra un ejemplo del problema SLAM basado en grafos. El robot se mueve a través de un entorno y percibe puntos de referencia (P.R.) por medio de su sensor. En este ejemplo, el mapa $\mathbf{m} = (\mathbf{l}_1, \mathbf{l}_2, \mathbf{l}_3)$ consiste de tres P.R. detectables por el sensor del robot, además, las mediciones de odometría proporcionan información sobre el movimiento del robot desde un instante de tiempo hasta el siguiente. En la Fig. 1b se muestra la red de Bayes correspondiente a la situación en la Fig. 1a. Para indicar observaciones individuales del sensor y su relación con el mapa, se aplican índices superiores a las observaciones del sensor $\mathbf{s}$. Por ejemplo, al instante de tiempo $t = 1$ el robot detecta $\mathbf{s}_1 = (\mathbf{s}_1^1, \mathbf{s}_1^2)$ correspondiente a dos P.R diferentes en el entorno.

En la Ec. 62 aparecen las distribuciones de probabilidad del modelo del sensor $p(\mathbf{s}_t \mid \mathbf{x}_t, \mathbf{m})$ y modelo de movimiento $p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t)$. El modelo del sensor modela la probabilidad de realizar una observación $\mathbf{s}_t$ dado que el robot se encuentra en la posición $\mathbf{x}_t$ en el mapa. De igual forma, el modelo de movimiento representa la probabilidad de que al instante de tiempo t el robot esté en la posición $\mathbf{x}_t$ dado que en el tiempo $t-1$ estaba en la posición $\mathbf{x}_{t-1}$ y adquirió una medición de odometría $\mathbf{u}_t$. En la Fig. 2 se muestra una representación de las variables involucradas en estos modelos. Suponinendo que la restricción en Ec. 50 se mantiene, y por lo tanto, la probabilidad condicional del modelo de movimiento es Gausiana, entonces

$$\begin{aligned} \mathbf{x}_t &\sim \mathcal{N}(g(\mathbf{x}_{t-1}, \mathbf{u}_t), \mathbf{\Lambda}_t) \\ p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t) &= \eta \ \exp\left(-\frac{1}{2}(g(\mathbf{x}_{t-1}, \mathbf{u}_t) - \mathbf{x}_t)^T \Lambda_t^{-1} (g(\mathbf{x}_{t-1}, \mathbf{u}_t) - \mathbf{x}_t)\right), \end{aligned} \tag{63}$$

donde el normalizador $\eta = 1/\sqrt{2\pi|\Lambda_t|}$. Aplicando la definición de la distancia de Mahalanobis la ecuación anterior se puede reescribir de una manera más compacta

$$p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t) = \eta \ \exp -\frac{1}{2} \| \ g(\mathbf{x}_{t-1}, \mathbf{u}_t) - \mathbf{x}_t \ \|^2_{\Lambda_t}. \tag{64}$$

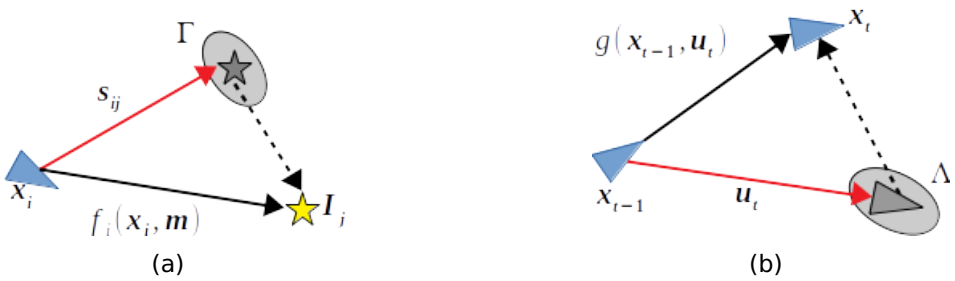


Figura 2. Representación de los modelos del sensor y movimiento. (a) El robot observa un P.R. (estrella gris), por medio del modelo del sensor $f(\cdot)$ se calcula la observación esperada (estrella amarilla), la flecha punteada indica el error. (b) De igual manera, en caso de restricciones pose-pose, mediante odometría el robot estima su posición (triángulo gris), con la función de movimiento se calcula la posición esperada (triángulo azul), la fecha punteada indica el error en la posición.

De la misma manera, la probabilidad condicional del modelo del sensor está dada por

$$\begin{aligned} \mathbf{s}_t &\sim \mathcal{N}(f(\mathbf{x}_t, \mathbf{m}), \mathbf{\Gamma}_t) \\ p(\mathbf{s}_t \mid \mathbf{x}_t, \mathbf{m}) &= \eta \ \exp\left(-\frac{1}{2}(f(\mathbf{x}_t, \mathbf{m}) - \mathbf{s}_t)^T \mathbf{\Gamma}_t^{-1}(f(\mathbf{x}_t, \mathbf{m}) - \mathbf{s}_t)\right) \\ &= \eta \exp -\frac{1}{2}\| f(\mathbf{x}_t, \mathbf{m}) - \mathbf{s}_t \|^2_{\Gamma_t}. \end{aligned} \tag{65}$$

Remplazando la Ec. 65 y Ec. 63 en la Ec. 62 y tomando el logaritmo negativo, se obtiene

$$-\log\ p(\mathbf{x}_{0:t}, \mathbf{m} \mid \mathbf{s}_{1:t}, \mathbf{u}_{1:t}) \approx c + \frac{1}{2}\sum_t \| f(\mathbf{x}_t, \mathbf{m}) - \mathbf{s}_t \|^2_{\Gamma_t} + \| g(\mathbf{x}_{t-1}, \mathbf{u}_t) - \mathbf{x}_t \|^2_{\Lambda_t}, \tag{66}$$

donde c es una constante que refleja los diversos normalizadores utilizados en las ecuaciones anteriores. El estado óptimo $\mathbf{x}^*$ es obtenido mediante una estimación máxima a posteriori (MAP). Dado que maximizar el logaritmo de una función es equivalente a la maximización de la función en si (Bishop, 2006), se obtiene

$$\begin{aligned} \mathbf{x}^* &= \underset{\mathbf{x}}{\text{argmax}}\ p(\mathbf{x}_{0:t}, \mathbf{m} \mid \mathbf{s}_{1:t}, \mathbf{u}_{1:t}) \\ &= \underset{\mathbf{x}}{\text{argmin}}\ -\log p(\mathbf{x}_{0:t}, \mathbf{m} \mid \mathbf{s}_{1:t}, \mathbf{u}_{1:t}). \end{aligned} \tag{67}$$

Para simplificar la notación, los datos de odometría **u** y las observaciones del sensor **s** se expresan en una sola variable denotada por **z** y se añade una nueva función **e**(**x**, **z**) la cual llama a las dos funciones g y f definidas anteriormente. De la misma manera, Ω_k^{-1} representa Λ_t^{-1} ó Γ_t^{-1}. Por último, se combinan todos los pares de índices para las

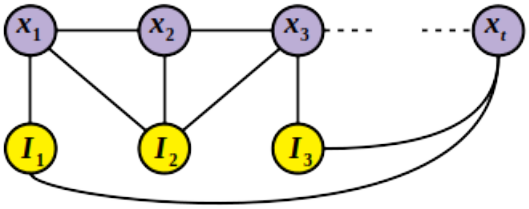


Figura 3. Representación del problema de SLAM mediante un grafo. Los nodos pueden representar la posición del robot ó de las observaciones, mientras que las aristas modelan las restricciones pose-pose (odometría) ó pose-observación (sensor)

mediciones disponibles en el conjunto $\mathcal{G}$. De esta manera, la Ec. 67 puede ser reescrita como

$$\mathbf{x}^* = F(\mathbf{x}) = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{k \in \mathcal{G}} \| \mathbf{e}(\mathbf{x}_k, \mathbf{z}_k) \|^2_{\Omega_k}. \tag{68}$$

La ecuación anterior resulta en un problema de estimación por mínimos cuadrados, el cual puede ser resuelto de manera muy eficiente al aprovechar las características de SLAM. En el contexto de mínimos cuadrados, las mediciones **z** representan restricciones al problema que puede ser representado mediante un grafo. Dicho grafo describe las relaciones entre las variables de estado, un nodo representa una variable de estado $\mathbf{x}_i$. La Fig. 3 muestra un ejemplo de dicho grafo. En este caso, las restricciones afectan las posiciones consecutivas del robot mediante mediciones de odometría y relacionan el robot con los P.R. observados con el sensor. La clave para una inferencia eficiente reside en la forma de la Ec. 68. Esta función tiene una forma general de mínimos cuadrados y, por lo tanto, puede resolverse mediante algoritmos iterativos descritos anteriormente: Descenso de Gradiente, Levenberg-Marquardt, Método de Newton o Gauss-Newton.

La solución al problema de SLAM basado en grafos consiste en resolver un problema de optimización por mínimos cuadrados (Ec. 68), donde:

- $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$ es el vector de estado donde cada $\mathbf{x}_k$ representa una variable de estado que modela la posición del robot ó la ubicación de un P.R.
- $\mathbf{z}_k$ es una medición que depende de las variables de estado $\mathbf{x}_k$. Dada una configuración de $\mathbf{x}_k$, el valor esperado $\hat{\mathbf{z}}_k = \mathbf{h}_k(\mathbf{x}_k)$ para una medición puede ser calculada usando el modelo determinístico $\mathbf{h}_k(\mathbf{x}_k)$, esta función puede ser el modelo del sensor, modelo de movimiento o una composición de ambas.
- $\mathbf{e}_k(\mathbf{x}_k, \mathbf{z}_k)$ es la función de error que calcula la diferencia entre la medición espe-

rada $\hat{\mathbf{z}}_k$ y la medición real $\mathbf{z}_k$, es decir, $\mathbf{e}(\mathbf{x}_k, \mathbf{z}_k) = \mathbf{h}_k(\mathbf{x}_k) - \mathbf{z}_k \overset{\text{def.}}{=} \mathbf{e}_k(x) \overset{\text{def.}}{=} \mathbf{e}_k$.

- Ω_k es la matriz de información que modela la incertidumbre del error.

El objetivo general de la minimización por mínimos cuadrados es determinar el vector de estado $\mathbf{x}^*$ que mejor se ajuste al conjunto de mediciones, es decir, se busca la solución de la siguiente expresión

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}}\ F(\mathbf{x}). \tag{69}$$

En aplicaciones prácticas, el modelo de medición $\mathbf{h}_k(\mathbf{x}_k)$ es en general una función no lineal de los estados. Sin embargo, si es lo suficientemente suave, se puede aproximarla mediante una linealización en la vecindad de un punto $\check{\mathbf{x}}$ por su expansión de Taylor

$$\mathbf{h}_k(\check{\mathbf{x}}_k + \Delta\mathbf{x}) = \mathbf{h}_k(\check{\mathbf{x}}_k) + \mathbf{J}_k\Delta\mathbf{x} + \mathcal{O}(\| \Delta\mathbf{x} \|^2), \tag{70}$$

donde $\mathbf{J}_k$ es el Jacobiano de $\mathbf{h}_k$ evaluada en $\check{\mathbf{x}}$. Si $\mathbf{J}_k$ es una aproximación suficiente al modelo $\mathbf{h}_k$ y $\Delta\mathbf{x}$ es pequeño, entonces se pueden ignorar los términos de mayor orden, lo que resulta en

$$\mathbf{h}_k(\check{\mathbf{x}}_k + \Delta\mathbf{x}) \approx \mathbf{h}_k(\check{\mathbf{x}}_k) + \mathbf{J}_k\Delta\mathbf{x}. \tag{71}$$

Sustituyendo esta aproximación de Taylor en la Ec. 68 resulta en una aproximación lineal a la función de error

$$\begin{aligned} \mathbf{e}_k(\check{\mathbf{x}} + \Delta\mathbf{x}) &= \mathbf{h}_k(\check{\mathbf{x}} + \Delta\mathbf{x}) - \mathbf{z}_k \\ &\approx \mathbf{h}_k(\check{\mathbf{x}}) + \mathbf{J}_k\Delta\mathbf{x} - \mathbf{z}_k \\ &= \mathbf{e}_k + \mathbf{J}_k\Delta\mathbf{x}. \end{aligned} \tag{72}$$

Sustituyendo la Ec. 72 en la Ec. 68, se obtiene

$$\begin{aligned} F_k(\check{\mathbf{x}} + \Delta\mathbf{x}) &= \mathbf{e}_k(\check{\mathbf{x}} + \Delta\mathbf{x})^T \Omega_k \mathbf{e}_k(\check{\mathbf{x}} + \Delta\mathbf{x}) \\ &\approx (\mathbf{e}_k + \mathbf{J}_k\Delta\mathbf{x})^T \Omega_k (\mathbf{e}_k + \mathbf{J}_k\Delta\mathbf{x}) \\ &= \underbrace{\mathbf{e}_k^T\Omega_k\mathbf{e}_k}_{\mathbf{c}_k} + 2\underbrace{\mathbf{e}_k^T\Omega_k\mathbf{J}_k}_{\mathbf{b}_k^T}\Delta\mathbf{x} + \Delta\mathbf{x}^T\underbrace{\mathbf{J}_k^T\Omega_k\mathbf{J}_k}_{\mathbf{H}_k}\Delta\mathbf{x} \\ &= \mathbf{c}_k + 2\mathbf{b}_k^T\Delta\mathbf{x} + \Delta\mathbf{x}^T\mathbf{H}_k\Delta\mathbf{x}. \end{aligned} \tag{73}$$

Con esta aproximación lineal, la Ec. 68 se puede reescribir de la siguiente manera

$$\begin{aligned} F(\check{\mathbf{x}} + \Delta\mathbf{x}) &= \sum_{k\in\mathcal{G}} F_k(\check{\mathbf{x}} + \Delta\mathbf{x}) \\ &\approx \sum_{k\in\mathcal{G}} \left(\mathbf{c}_k + 2\mathbf{b}_k^T\Delta\mathbf{x} + \Delta\mathbf{x}^T\mathbf{H}_k\Delta\mathbf{x}\right) \\ &= \underbrace{\sum_{k\in\mathcal{G}} \mathbf{c}_k}_{\mathbf{c}} + 2\underbrace{\sum_{k\in\mathcal{G}} \mathbf{b}_k^T}_{\mathbf{b}^T}\Delta\mathbf{x} + \Delta\mathbf{x}^T\underbrace{\sum_{k\in\mathcal{G}} \mathbf{H}_k}_{\mathbf{H}}\Delta\mathbf{x} \\ &= \mathbf{c} + 2\mathbf{b}^T\Delta\mathbf{x} + \Delta\mathbf{x}^T\mathbf{H}\Delta x \\ &=: G(\Delta\mathbf{x}). \end{aligned} \tag{74}$$

De la ecuación anterior se observa que esta aproximación local conduce a una función $G(\Delta\mathbf{x})$ que es cuadrática en los incrementos $\Delta\mathbf{x}$, por lo tanto, solo tiene un único mínimo global. Dado que $G(\Delta\mathbf{x})$ es una aproximación lineal de $F(\check{\mathbf{x}} + \Delta\mathbf{x})$ es necesario determinar el incremento $\Delta\mathbf{x}^*$ que aplicado a la estimación actual proporciona la mejor solución

$$\Delta\mathbf{x}^* = \underset{\Delta\mathbf{x}}{\operatorname{argmin}}\ G(\Delta\mathbf{x}). \tag{75}$$

Para determinar $\Delta\mathbf{x}^*$ se toma la primer derivada y se iguala a cero $G(\Delta\mathbf{x})$:

$$\frac{\partial(G(\Delta\mathbf{x}))}{\partial(\Delta\mathbf{x})} = 2\mathbf{H}\Delta\mathbf{x} + 2\mathbf{b} = 0, \tag{76}$$

lo que resulta en el sistema lineal

$$\mathbf{H}\Delta\mathbf{x}^* = -\mathbf{b}. \tag{77}$$

Resolviendo este sistema se encuentra el valor del incremento $\Delta\mathbf{x}^*$, el cual es añadio al valor inicial

$$\mathbf{x}^* = \check{\mathbf{x}} + \Delta\mathbf{x}^*. \tag{78}$$

La solución típica es iterar la linealización en la Ec. 74, la solución en la Ec. 77 y el paso de actualización en la Ec. 78 hasta que se cumpla algún criterio dado. En cada iteración, la solución anterior se utiliza como el punto de linealización. Los algoritmos de Gauss-Newton y Levenberg-Marquardt son métodos muy populares para llevar a cabo estas iteraciones. La matriz $\mathbf{H}$ es dispersa por construcción, dado que se obtiene

proyectando el error de la medición a través de los Jacobianos, solo tiene valores diferentes de cero entre estados conectados por una restricción. Esto permite resolver la Ec. 77 de manera muy eficiente mediante factorización de Cholesky (Grisetti *et al.*, 2010).

2.4. Geometría 3D

El movimiento de un cuerpo rígido en espacio tridimensional Euclidiano preserva la distancia y la orientación entre cualquier par de puntos del objeto. Formalmente un movimiento de cuerpo rígido es un mapeo $\tau : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ de modo que la norma y el producto cruz entre cualquier par de puntos $\mathbf{p}$ y $\mathbf{q}$ del objeto se conserva

$$\begin{aligned} \| \mathbf{p} - \mathbf{q} \| = \| \tau(\mathbf{p}) - \tau(\mathbf{q}) \| \qquad & \forall \mathbf{p}, \mathbf{q} \in \mathbb{R}^3 \\ \tau(\mathbf{p}) \times \tau(\mathbf{q}) = \tau(\mathbf{p} \times \mathbf{q}) \qquad & \forall \mathbf{p}, \mathbf{q} \in \mathbb{R}^3. \end{aligned} \tag{79}$$

Este mapeo es llamado transformación especial Euclidiana. El conjunto de todas estas transformaciones en el espacio tridimensional Euclidiano forman el grupo de Lie ó grupo especial Euclidiano $SE(3)$. Las propiedades de preservación de la distancia y orientación se pueden usar para representar el movimiento de un cuerpo rígido de manera compacta. La transformación de un punto que tiene un sistema de referencia global es suficiente para especificar el movimiento de todo el objeto. Esta transformación es siempre con respecto a un sistema de coordenadas de referencia y puede descomponerse en partes rotacional y traslacional.

Se define una transformación 3D del marco de coordenadas B al marco A en términos de su traslación relativa $\mathbf{t}_{AB} \in \mathbb{R}^3$ y rotación $\mathbf{R}_{AB} \in SO(3)$, los cuales forman la matriz de transformación $\mathbf{T}_{AB} \in SE(3)$. Todas las matrices de rotación pertenecen al grupo especial ortogonal ó simplemente grupo de rotación $SO(3)$, definido como

$$SO(3) = \{ \mathbf{R} \in \mathbb{R}^{3\times 3} \mid \mathbf{R}^T \cdot \mathbf{R} = \mathbf{I} \wedge \det(\mathbf{R}) = \mathbf{1} \}. \tag{80}$$

Con esta notación es posible definir formalmente el grupo $SE(3)$ de la siguiente mane-

ra:

$$SE(3) = \{\tau = (\mathbf{R}, \mathbf{t}) \mid \mathbf{R} \in SO(3), \mathbf{t} \in \mathbb{R}^3\} = SO(3) \times \mathbb{R}^3. \tag{81}$$

Un punto $\mathbf{x}_B$ en el marco B puede ser transformado en el marco A multiplicando con la matriz de transformación

$$\tilde{\mathbf{x}}_A = \mathbf{T}_{AB} \cdot \tilde{\mathbf{x}}_B, \tag{82}$$

donde la tilde $\tilde{.}$ representa un punto en coordenadas homogéneas. Usando esta representación, las matrices de transformación puede ser concatendas con una simple multiplicación

$$\mathbf{T}_{CA} = \mathbf{T}_{CB} \cdot \mathbf{T}_{BA}. \tag{83}$$

La matriz de transformación inversa está dada por

$$\mathbf{T}_{BA} = (\mathbf{T}_{AB})^{-1} = \begin{bmatrix} \mathbf{R}_{AB}^T & -\mathbf{R}_{AB}^T \cdot \mathbf{t}_{AB} \\ 0 & 1 \end{bmatrix}. \tag{84}$$

En algunos casos no es conveniente trabajar con esta representación matricial, ya que, la matriz de rotación tiene nueve parámetros y solo tres grados de libertad. Sim embargo, existen otras representaciones más eficientes como quaterniones, ángulos de Euler y representación de Lie, las cuales son mayormente usadas en problemas de optimización.

2.4.1. Quaterniones

Los quaterniones representan rotaciones 3D como un punto en una esfera unitaria 4D. Son la representación más compacta para rotaciones libres de singularidades. En este trabajo, se utilizan quaterniones para representar los estados internos. Dado un quaternión $\mathbf{q} = (x, y, z, w)^T \in \mathbb{S}^3$, la correspondiente matriz de rotación puede ser obtenida como (Ma *et al.*, 2003)

$$\mathbf{R} = \begin{bmatrix} 1-2y^2-2z^2 & 2xy-2zw & 2xz+2yw \\ 2xy+2zw & 1-2x^2-2z^2 & 2yz-2xw \\ 2xz-2yw & 2yz+2xw & 1-2x^2-2y^2 \end{bmatrix}. \tag{85}$$

2.4.2. Ángulos de Euler

Los ángulos de Euler representan rotaciones 3D como tres ángulos, denotando tres rotaciones secuenciales sobre ciertos ejes. Hay muchas posibilidades de cómo se pueden definir los ejes y el orden en que giran, lo que en práctica hace que esta representación sea ambigua. Como una representación mínima de $SO(3)$, los ángulos de Euler sufren de singularidades. Un problema particular es que, en ciertas configuraciones, pierden un grado de libertad, es decir, pierden la propiedad de poder ser perturbados en las tres dimensiones. Debido a esto, esta representación es inadecuada para problemas de optimización.

Uno de los casos de uso más común es el contexto de navegación aérea, donde los tres ángulos son llamados *roll*, *pitch* y *yaw*.

2.4.3. Representación de Lie

Una representación mínima para poses 3D puede ser derivada usando la teoría de grupos de Lie. Existen muchos grupos de Lie, los más importantes en el contexto de esta tesis son el Grupo Especial Euclidiano (Ec. 81) y el Grupo Especial Ortogonal (Ec. 80).

Cada grupo de Lie da lugar a su correspondiente álgebra de Lie, que define la estructura del espacio tangente alrededor de la identidad. Está dada por todas las combinaciones lineales de sus matrices generadores, que son las derivadas con respecto a la rotación y traslación elemental evaluadas alrededor de la identidad. Están dadas por

$$\mathbf{G}_{r_x} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{G}_{r_y} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{G}_{r_z} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$
$$\mathbf{G}_{t_x} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{G}_{t_y} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{G}_{t_z} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{86}$$

Se define el álgebra de Lie correspondiente a los grupos de Lie $SO(3)$ y $SE(3)$ de la siguiente manera

$$\mathfrak{so}(3) = \{\omega_1 \mathbf{G}_{r_x} + \omega_2 \mathbf{G}_{r_y} + \omega_3 \mathbf{G}_{r_z} \mid (\omega_1, \omega_2, \omega_3) \in \mathbb{R}^3\}, \tag{87}$$

$$\mathfrak{se}(3) = \{\mathbf{A} + \upsilon_1 \mathbf{G}_{t_x} + \upsilon_2 \mathbf{G}_{t_y} + \upsilon_3 \mathbf{G}_{t_z} \mid \mathbf{A} \in \mathfrak{so}(3), (\upsilon_1, \upsilon_2, \upsilon_3) \in \mathbb{R}^3\}. \tag{88}$$

Los elementos del álgebra de Lie puede ser transformados a su respectivo grupo de Lie usando el mapeo exponencial, y viceversa usando el mapeo logarítmico. Por simplicidad, se denotan los elementos del algebra de Lie directamente con vectores que contengan los coeficientes de la matriz generadora respectiva.

La representación resultante es ideal para cantidades diferenciales en un grupo de Lie (incertidumbre o derivadas), ya que,

- es una representación mínima con un mapeo exacto $exp : \mathfrak{se}(3) \rightarrow SE(3)$
- la adjunta puede ser usada para mapear de manera exacta y lineal elementos entre diferentes espacios tangentes. Esto significa que todos los espacios tangentes tiene la misma estructura lineal.
- está definida en términos de derivadas con respecto a los tipos de movimiento subyacentes y, por lo tanto, es intuitivo usarla en el contexto de opimización.

2.4.3.1. La adjunta

La adjunta es una función que mueve elementos entre diferentes espacios tangente de un grupo de Lie. Dado un elemento de un grupo $\mathbf{T}$ y un elemento de algebra ξ con

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \in SE(3), \qquad \xi = \begin{bmatrix} \mu \\ \omega \end{bmatrix} \in \mathfrak{se}(3). \tag{89}$$

La adjunta $\text{Adj}_{\mathbf{T}} \in \mathbb{R}^{6\times 6}$ es una función lineal que mueve ξ del espacio tangente derecho de $\mathbf{T}$ al espacio tangente izquierdo de $\mathbf{T}$, es decir

$$\mathbf{T} \cdot e^{\widehat{\xi}} = e^{\widehat{Adj_{\mathbf{T}} \cdot \xi}} \cdot \mathbf{T}. \tag{90}$$

Es posible obtener en forma cerrada la expresión para la adjunta usando $\mathbf{A}e^{\mathbf{X}}\mathbf{A}^{-1} = e^{\mathbf{AXA}^{-1}}$ para cualquier matriz invertible $\mathbf{A}$, es decir

$$\begin{aligned} e^{\widehat{Adj_{\mathbf{T}} \cdot \xi}} &= \mathbf{T} \cdot e^{\widehat{\xi}} \cdot \mathbf{T}^{-1} \\ &= e^{\mathbf{T} \cdot \widehat{\xi} \cdot \mathbf{T}^{-1}} \\ \widehat{Adj_{\mathbf{T}} \cdot \xi} &= \mathbf{T} \cdot \widehat{\xi} \cdot \mathbf{T}^{-1} \end{aligned} \tag{91}$$

que se puede resolver para la $Adj_{\mathbf{T}}$

$$Adj_{\mathbf{T}} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \cdot \mathbf{R} \\ \mathbf{0}_{3\times 3} & \mathbf{R} \end{bmatrix} \in \mathbb{R}^{6\times 6}. \tag{92}$$

2.4.3.2. Jacobianos

La definición de un elemento de álgebra de Lie ξ como vector coeficiente de las matrices generadoras $\mathbf{G}_1$ a $\mathbf{G}_6$

$$\left.\frac{\partial e^{\widehat{\xi}}}{\partial \xi_i}\right|_{\xi=0} \qquad \text{para} \;\; i = 1 \ldots 6. \tag{93}$$

En el contexto de reconstrucción 3D y SLAM, casi siempre se requieren derivadas de funciones que involucran puntos de rotación y traslación, sea

$$\tilde{\mathbf{y}}(\xi, \mathbf{x}) := e^{\widehat{\xi}}\tilde{\mathbf{x}} \tag{94}$$

la transformación de un punto 3D (por simplicidad se incluyen las coordenadas homogéneas, en practica siempre es uno y su respectiva derivada siempre es cero). Dado que $\tilde{\mathbf{y}}$ es lineal en $e^{\widehat{\xi}}$ y $\tilde{\mathbf{x}}$, la derivada con respecto a ξ alrededor de la identidad puede ser obtenida por los generadores

$$\left.\frac{\partial \tilde{\mathbf{y}}(\xi, \mathbf{x})}{\partial \xi_i}\right|_{\xi=0} = \left.\frac{\partial e^{\widehat{\xi}}}{\partial \xi_i}\right|_{\xi=0} \tilde{\mathbf{x}} = \mathbf{G}_i\tilde{\mathbf{x}} \qquad \text{para} \quad i = 1\ldots6. \tag{95}$$

La adjunta puede ser usada para derivar las funciones que implican concatenaciones de pose: dada una función

$$\tilde{\mathbf{y}}'(\xi, \mathbf{x}) := \mathbf{T}e^{\widehat{\xi}}\tilde{\mathbf{x}}. \tag{96}$$

Usando la Ec. 90 se puede mover ξ completamente a la izquierda

$$\tilde{\mathbf{y}}'(\xi, \mathbf{x}) = e^{\widehat{Adj_{\mathbf{T}}\xi}}\mathbf{T}\tilde{\mathbf{x}} \tag{97}$$

y usando la Ec. 95 para calcular la derivada como

$$\left.\frac{\partial \tilde{\mathbf{y}}'(\xi, \mathbf{x})}{\partial \xi}\right|_{\xi=0} = \left.\frac{e^{\widehat{\xi}'}\mathbf{T}\tilde{\mathbf{x}}}{\partial \xi'}\right|_{\xi'=0} \left.\frac{\partial Adj_{\mathbf{T}}\xi}{\partial \xi}\right|_{\xi=0} = [\mathbf{G}_1\mathbf{T}\tilde{\mathbf{x}}|\ldots|\mathbf{G}_6\mathbf{T}\tilde{\mathbf{x}}] \cdot Adj_{\mathbf{T}}, \tag{98}$$

donde $[|\ldots|]$ denota el Jacobiano completo obtenido de las derivadas parciales de los vectores (Ec. 95) en forma matricial. Es importante señalar que estas derivaciones solo se mantienen si el mapa exponencial es evaluado alrededor de cero.

2.4.3.3. Representación de incertidumbre en poses

En el contexto gausiano, las incertidumbres sobre un elemento del grupo de Lie $\mathbf{T} \in SE(3)$ se representan mejor en el espacio tangente respectivo, es decir

$$\mathbf{T} =: e^{\widehat{\epsilon}}\mathbf{T}_0 \qquad \text{con} \quad \epsilon \sim \mathcal{N}(0, \Sigma_\epsilon). \tag{99}$$

Usando la derivación anterior de Jacobianos, la adjunta puede ser usada para propagar las incertidumbres a través de la concatenación de las poses

$$\mathbf{T}' = \mathbf{T}_a\mathbf{T}\mathbf{T}_b =: e^{\widehat{\epsilon}'}\underbrace{\mathbf{T}_a\mathbf{T}_0\mathbf{T}_b}_{=\mathbf{T}'_0} \qquad \text{con} \quad \epsilon' \sim \mathcal{N}(0, \underbrace{Adj_{\mathbf{T}_a}\Sigma_\epsilon Adj_{\mathbf{T}_a}^T}_{\Sigma_{\epsilon'}}) \tag{100}$$

y, a través de la inversión de pose como

$$\mathbf{T}' = \mathbf{T}^{-1} =: e^{\widehat{\epsilon}'}\underbrace{\mathbf{T}_0^{-1}}_{=\mathbf{T}'_0} \qquad \text{con} \qquad \epsilon' \sim \mathcal{N}(0, \underbrace{Adj_{\mathbf{T}^{-1}}\Sigma_\epsilon Adj_{\mathbf{T}^{-1}}^T}_{\Sigma_{\epsilon'}}). \tag{101}$$

Debido a la linealidad de la adjunta, la propagación de incertidumbres de esta manera es exacta, es decir, no implica linealizaciones adicionales. En particular, esto implica que la precisión de las linealizaciones requeridas para ajustar la distribución verdadera a un gausiano no depende del espacio tangente específico elegido para representarlo.

Capítulo 3. SLAM Visual

Cuando se usan cámaras como único sensor exteroceptivo para extraer información del ambiente por donde navega el robot, se le conoce como SLAM visual (Fuentes-Pacheco *et al.*, 2015). En la última década, los enfoques basados en visión han ganado mucha popularidad debido a las mejoras en la tecnología y a la cantidad de información útil que las cámaras pueden proveer (apariencia del entorno, color y textura) dando a los robots la capacidad de integrar tareas de alto nivel como detección y reconocimiento de personas y lugares. Además, las cámaras son menos costosas, más ligeras y consumen menos energía en comparación con láseres. Desafortunadamente, pueden existir errores en los datos debido a las siguientes razones: resolución insuficiente, cambios de iluminación, superficies con falta de textura, imágenes borrosas debido a movimientos rápidos, entre otros.

En SLAM visual se busca estimar el movimiento de la cámara que se mueve en el espacio tridimensional a través del conjunto de imágenes que captura, y al mismo tiempo, estimar la estructura del entorno. El problema es discretizado en espacio y tiempo, como se observa en la Fig. 4, donde se representa la trayectoria de la cámara mediante las poses consecutivas $\ldots, \mathbf{T}_{t-1}, \mathbf{T}_t, \mathbf{T}_{t+1}, \ldots$ en intervalos de tiempo discreto y el entorno usando un conjunto de puntos. Donde $\mathbf{T} \in SE(3)$ es una matriz de transformación de movimiento de cuerpo rígido que describe la pose de la cámara en en relación con un sistema de coordenadas global.

En general, el problema de SLAM visual basado en grafos puede ser descompuesto en dos módulos (Durrant-Whyte y Bailey, 2006; Endres *et al.*, 2014; Bailey y Durrant-Whyte, 2006). El primer módulo es llamado Frontend, el cual consiste en asociar puntos o regiones de una imagen con puntos o regiones de las imágenes siguientes y así extraer relaciones geométricas entre la cámara y las observaciones visuales. Esto se conoce como odometría visual (VO). El concepto de odometría viene del campo de robótica móvil donde sensores acoplados a las ruedas del robot cuentan el número de giros, esto resulta en una estimación incremental de la pose, donde la pose actual $\mathbf{T}_t$ solo depende de la pose anterior $\mathbf{T}_{t-1}$ y su correspondiente medición de odometría $\mathbf{u}_{t-1}$. De manera similar, odometría visual se refiere al hecho de estimar la pose actual $\mathbf{T}_t$ dada la anterior $\mathbf{T}_{t-1}$ y sus correspondientes mediciones visuales $\mathbf{z}_{t-1}$ a partir de imágenes temporalmente adyacentes $\mathbf{I}_{t-1}$, $\mathbf{I}_t$. El segundo módulo es llamado Bac-

(a) (b)

Figura 4. SLAM visual. (a) Secuencia de imágenes adquiridas por la cámara. (b) Estimación de la pose de la cámara y de la estructura del entorno.

kend, el cual estima la estructura del mapa y trayectoria de la cámara mediante la optimización del grafo para lograr una consistencia global.

3.1. Clasificación

Localización y mapeo simultáneo (SLAM) y odometría visual (VO) son bloques de construcción fundamentales para robots autónomos y sistemas de realidad virtual. Muchos métodos para SLAM y VO han progresado significativamente en los últimos años. El campo ha sido dominado por métodos basados en características (indirectos), sin embargo, en los últimos años nuevos enfoques han emergido ganando popularidad debido a su precisión y robustez, llamados métodos directos (Engel *et al.*, 2018).

3.1.1. Métodos directos

Los métodos directos usan directamente las mediciones del sensor, en este caso la intensidad de los píxeles en la imagen. Un método directo puede ser denso (Newcombe *et al.*, 2011) si todos los píxeles son usados, semi-denso (Engel y Cremers, 2014) si solo se consideran píxeles con alto gradiente, o disperso (Engel *et al.*, 2018) si solo se usa un conjunto pequeño de píxeles. El enfoque directo minimiza el error fotométrico obtenido por la alineación directa de la imagen en los píxeles usados, basados en el supuesto de constancia de brillo. Las poses de la cámara y la profundidad de los píxeles son estimadas minimizando el error fotométrico usando algoritmos de optimización

no lineales. Dado que se puede utilizar mucha información de la imagen, los métodos directos son robustos en escenas con poca textura y pueden ofrecer reconstrucciones relativamente densas. En consecuencia, debido a la formulación de alineación directa de la imagen, los métodos directos son muy sensibles al efecto obturador, la exposición automática de la cámara y al control de ganancia. Más importante aún, el supuesto de constancia de brillo no siempre se cumple en la práctica, lo que reduce drásticamente el rendimiento de los métodos directos en entornos con cambios rápidos de iluminación.

El método directo del estado del arte es DSO (Engel *et al.*, 2018). DSO realiza un novedoso muestreo de puntos dispersos en áreas de imagen con gradiente de intensidad suficiente. La reducción de la cantidad de datos permite aplicar BA en una ventana en tiempo real. La información obsoleta y redundante se margina con el complemente de Schur (Leutenegger *et al.*, 2015). Como método directo, DSO se basa fundamentalmente en el supuesto de constancia de brillo, por lo que los autores propusieron un método de calibración fotométrica de la cámara para recuperar las imágenes de irradiación, lo que incrementa drásticamente la precisión del seguimiento de la cámara. En la Fig. 5 se muestran los resultados del sistema DSO.

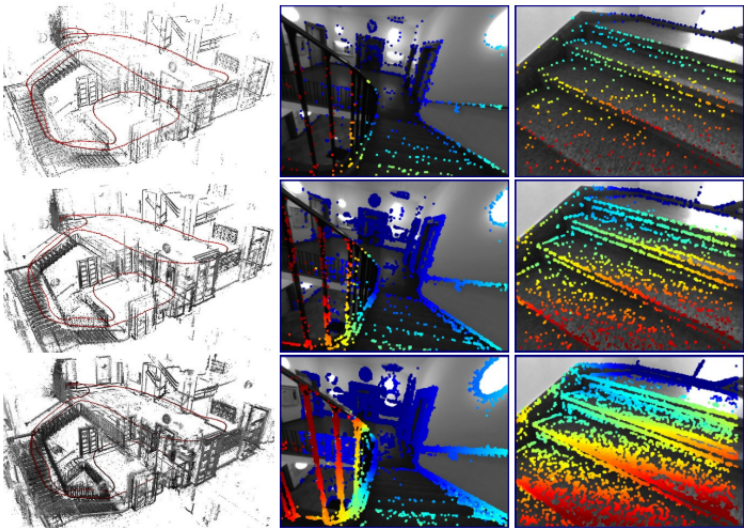


Figura 5. Mapa de nube de puntos 3D y subconjunto de píxeles usados para optimizar el error fotométrico en el sistema DSO.

3.1.2. Métodos indirectos (basados en características)

Los métodos indirectos operan en dos etapas: Primero se extraen puntos distintivos de interés de las imágenes, estos puntos forman una representación dispersa de la imagen. Segundo, estos puntos son interpretados como mediciones ruidosas en un modelo probabilístico para estimar la geometría y el movimiento de la cámara. Estos métodos optimizan el error geométrico, ya que, los puntos extraídos son entidades geométricas. La precisión de estos métodos depende directamente de las propiedades de invarianza del algoritmo extractor utilizado. La ventaja es que estas entidades geométricas son fáciles de manipular para calcular soluciones iniciales a problemas geométricos que son importantes en SLAM visual, como triangulación, geometría epipolar y transformaciones de cuerpo rígido entre sistemas de referencia. La principal limitante de estos enfoques es que solo pueden explotar la información visual donde se puedan extraer las características visuales. La falta de textura o imágenes borrosas ocasiona que los métodos indirectos fallen, además, el mapa generado por estos enfoques consiste de un conjunto disperso de puntos con poca utilidad para tareas robóticas distintas de la localización. Mur-Artal y Tardós (2015) demostraron que los métodos basados en características son más precisos que los métodos directos, posiblemente debido a la falta de madurez de los métodos directos. En la Fig. 6 se muestra un ejemplo de un sistema indirecto llamado ORB-SLAM propuesto por Mur-Artal *et al.* (2015).

ORB-SLAM se ha convertido en uno de los métodos basados en características más populares y ha sido ampliamente adoptado para una variedad de aplicaciones. Usa características ORB para todas las tareas incluyendo seguimiento de la cámara, mapeo, relocalización y cierre de ciclos. Para estimar la pose inicial de la cámara aplica BA, que luego se refina usando todas las correspondencias de características en el mapa local y realizando nuevamente la optimización de la pose. Se utiliza un grafo de covisibilidad para mejorar la eficiencia del sistema al limitar BA a un área local visible. A diferencia de DSO, en ORB-SLAM los puntos antiguos y keyframes se eliminan directamente de la ventana activa sin marginación. En esta tesis se hace uso del enfoque indirecto.

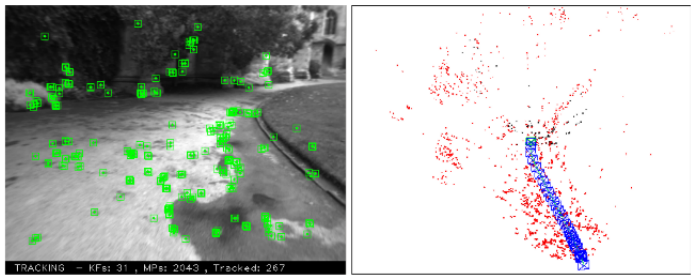


Figura 6. Puntos de intéres extraidos usando el algoritmo ORB y mapa de puntos dispersos resultado del algoritmo ORB-SLAM.

3.1.3. Métodos semi-directos

Los métodos semi-directos han sido considerados como métodos híbridos entre las dos formulaciones mencionadas anteriormente. SVO (Forster *et al.*, 2014) extrae esquinas FAST y realiza alineación directa de imágenes en estas zonas para una estimación inicial de la pose. La extracción de características se extendió más tarde para incluir bordes o segmentos de línea y mejorar la robustez (Gomez-Ojeda *et al.*, 2016; Forster *et al.*, 2017). Las profundidades de los píxeles seleccionados son estimadas a partir de múltiples observaciones usando un filtro de profundidad Bayesiano recursivo. Para reducir el error acumulado causado por las estimaciones incrementales, las poses y profundidades son refinadas mediante BA. Debido a su excepcional alta eficiencia (alrededor de 400 fps en una laptop) y su bajo costo, SVO puede ser implementado en dispositivos con recursos computaciones limitados, por lo que ha ganado una gran popularidad en una amplia gama de aplicaciones robóticas.

3.2. Sensores

Actualmente existen tres configuraciones para SLAM visual: cámara monocular, cámara estereoscópica y cámara RGB-D. Independientemente de la configuración usada, las cámaras tienen que ser calibradas. Este proceso de calibración estima los parámetros intrínsecos y extrínsecos, los primeros dependen de la geometría interna de la cámara, mientras que los segundos dependen de la posición de la cámara en el espacio con respecto a un sistema de coordenadas. Los parámetros necesarios generalmente

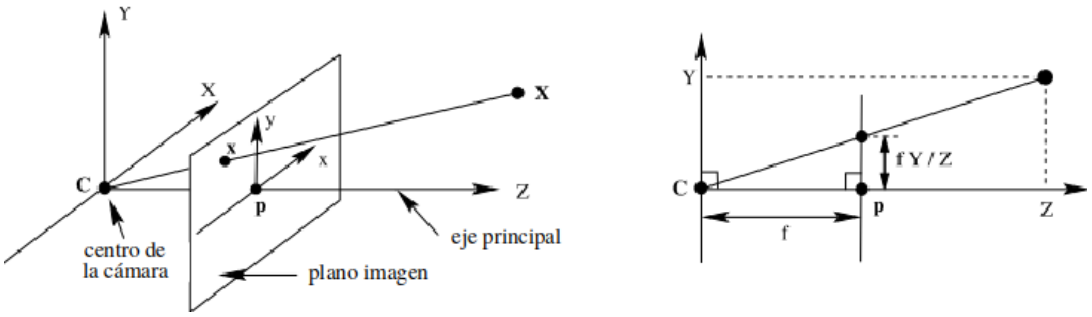


Figura 7. Modelo de la camara pinhole.

son estimados a partir de un conjunto de imágenes que contienen múltiples vistas de un patrón de calibración de tablero de ajedrez para relacionar las coordenadas de la imagen con las coordenadas del mundo real (Hartley y Zisserman, 2003).

3.2.1. Cámara monocular

La idea de utilizar una sola cámara se ha vuelto muy popular debido a que son baratas, consumen poca energía, son ideales para ir montadas en un robot ya que son pequeñas y ligeras y además su proceso de calibración es muy sencillo. Por otro lado, debido a la naturaleza proyectiva de una sola cámara la estimación del movimiento y estructura del mapa solo puede ser recuperada hasta cierta escala (Strasdat *et al.*, 2010). Esto ocasiona que el robot tenga errores de escala en la estimación de la trayectoria y mapa. A pesar de esto, en enfoque monocular ofrece una solución simple, flexible y económica en términos de hardware y tiempo de procesamiento.

Este tipo de cámara puede ser modelada con precisión como una cámara pinhole (Hartley y Zisserman, 2003). En esta tesis se usa el modelo de la camara pinhole mostrado en la Fig. 7. Este modelo es una abstracción de una cámara completa a un agujero infinitamente pequeño, el pinhole, y un plano de imagen. La proyección descrita por el modelo es conocida como proyección perspectiva. Solo se proyectan los rayos de luz que caen a través del orificio y se intersectan con el plano de la imagen. La ubicación del pinhole es el centro óptico c de la cámara y la distancia entre el centro óptico y el plano imagen es la distancia focal f.

Este modelo describe la relación matemática entre las coordenadas de un punto

en el espacio tridimensional y su proyección en el plano de la imagen. Un punto $\mathbf{x}_C = (x, y, z)^T \in \mathbb{R}^3$ en el sistema de coordenadas de la cámara C es proyectado a un píxel $\mathbf{x}_i = (u, v)^T$ en el plano de la imagen mediante la función de proyección $\pi_m : \mathbb{R}^3 \rightarrow \mathbb{R}^2$:

$$\mathbf{x}_i = \pi_m(\mathbf{x}_C) = \begin{bmatrix} f_x \frac{x}{z} + c_x \\ f_y \frac{y}{z} + c_y \end{bmatrix}. \tag{102}$$

Dado que los píxeles en el sensor no tienen que ser cuadráticos, se utilizan dos distancias focales diferentes f_x y f_y, las cuales se obtiene a través de diferentes factores de escala s_x y s_y en la dirección respectiva, entonces $f_x = s_x \cdot f$ y $f_y = s_y \cdot f$. c_x y c_y son las coordenadas del punto principal que toman en cuenta el hecho de que el centro óptico c no coincide con el origen del sistema de coordenadas de la imagen.

Si la profundidad z y los parámetros de la cámara son conocidos, un punto $\mathbf{x}_C$ el sistema de coordenadas de la cámara se puede recuperar a partir de un punto en la imagen $\mathbf{x}_i$ usando la inversa de la Ec. 102

$$\mathbf{x}_C = \pi_m^{-1}(\mathbf{x}_i) = \begin{bmatrix} \frac{z}{f_x}(u - c_x) \\ \frac{z}{f_y}(v - c_y) \\ z \end{bmatrix}. \tag{103}$$

3.2.2. Cámara estereoscópica

Las configuraciones estereoscópicas ofrecen las ventajas de ser fáciles de calibrar y de calcular de manera precisa la posición tridimensional real de las observaciones contenidas en la escena por medio de triangulación, esta información es muy útil para el problema SLAM visual. Entre las desventajas se encuentran la necesidad de una calibración extrínseca y mayor de tiempo de procesamiento.

Las cámaras estéreo están compuestas por dos cámaras fijas sincronizadas por hardware para que la captura de imágenes se haga al mismo tiempo. La profundidad puede ser estimada a partir de un solo frame estéreo encontrando correspondencias entre ambas imágenes, ya que un punto en el mundo se puede proyectar en la imagen

Figura 8. Microsoft Kinect: 1) Proyector IR. 2) Cámara RGB. 3) Cámara IR.

izquierda I_L así como en la imagen derecha I_R. Se asume que ambas imágenes se encuentran rectificadas, es decir, una fila en la imagen I_R corresponde a una fila en la imagen I_L, por lo tanto, dadas dos observaciones del mismo punto en el mundo (u_L, v_L) y (u_R, v_R), entonces, $v_L = v_R$. La distancia horizontal entre el origen de I_L y el origen de I_R se le conoce como línea base b. La función de proyección para una cámara estéreo rectificada $\pi_s : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ es la siguiente

$$\mathbf{x}_i = \pi_s(\mathbf{x}_C) = \begin{bmatrix} f_x \dfrac{x}{z} + c_x \\ f_y \dfrac{y}{z} + c_y \\ f_x \dfrac{x - b}{z} + c_x \end{bmatrix}, \tag{104}$$

donde $\mathbf{x}_i = (u_L, v_L, u_R)$, (u_L, v_L) son las coordenadas en la imagen izquierda y u_R es la coordenada horizontal en la imagen derecha.

3.2.3. Cámara RGB-D

Las cámaras RGB-D como el Kinect se han convertido en un popular dispositivo de medición tridimensional para aplicaciones de robótica en entornos interiores, reconstrucción 3D y reconocimiento de objetos, debido a su bajo costo, confiabilidad y velocidad de operación (Smisek *et al.*, 2011).

El Kinect está compuesto por un proyector infrarrojo y una cámara infrarroja, los cuales son usados para triangular puntos en el espacio, y una cámara RGB, la cual puede ser usada para extraer color y textura de los puntos 3D (Fig. 9). Esto implica que para cada píxel en la imagen RGB se conoce su profundidad. Debido a la naturaleza de los dispositivos infrarrojos del Kinect, su operación está restringida a entornos interiores. Las cámaras RGB e IR ofrecen una resolución de 640×480 píxeles a una frecuencia de 30 Hz y su rango de operación está entre los $0.5m - 5m$.

El Kinect tiene un sensor de profundidad compuesto por el proyector y la cámara IR, el cual estima la profundidad usando la tecnología de codificación de luz estructurada. Su proyector IR emite un patrón de luz en la escena conocido como speckle. La cámara IR captura el patrón speckle reflejado y lo correlaciona con un patrón de referencia almacenado de un plano. Dependiendo si el objeto está cercano o lejano del plano de referencia, el patrón de speckle se aleja o se desplaza hacia el centro del sensor a lo largo de la línea base que une el proyector IR y la cámara IR. Una estimación de esta desplazamiento o disparidad se obtiene haciendo coincidir los patrones reflejados y de referencia mediante el algoritmo de triangulación estéreo. En la Fig. 9 se muestran las imágenes de color y de profundidad obtenidas con el Microsoft Kinect.

Figura 9. Imágenes de color y profundidad obtenidas con un Microsoft Kinect. Los niveles de gris en la imagen de profundidad representan la distancia.

Khoshelham y Elberink (2012) calcularon la profundidad a partir de la disparidad usando triángulos semejantes y determinaron que la precisión y resolución de la profundidad decrecen de forma cuadrática al aumentar la profundidad del sensor.

3.3. Extracción de características visuales

En SLAM visual basado en características (indirecto), la idea principal es procesar las imágenes para extraer puntos de interés distintivos que pueden detectarse de manera confiable y repetida en imágenes de la misma escena. Típicamente, estos puntos deben ser invariantes a rotaciones, traslaciones, escala, punto de vista y cambios de

iluminación. Sin embargo, en la práctica, la estabilidad de los puntos de interés no se mantiene, haciendo difícil el proceso estimación de correspondencias. El proceso de extracción de características visuales está compuesto por dos fases: detección y descripción. La detección consiste en procesar la imagen para obtener un número de características. La descripción consiste en construir un vector de características basado en la apariencia visual de la imagen. La invariabilidad del descriptor con respecto a los cambios de posición y orientación permitirá mejorar la correspondencia y asociación de datos.

En la literatura, han sido propuestos muchos detectores y descriptores de características. En este trabajo solo se contemplaron aquellos que han sido exitosamente usados en VO/SLAM (Garcia-Fidalgo y Ortiz, 2015).

3.3.1. Detección

Ser capaz de representar una imagen de forma invariante que sea la misma, o muy similar, en otras imágenes de la misma escena es un requerimiento fundamental para resolver el problema indirecto de SLAM visual. Las representaciones de esquina son métodos poderosos para hacer esto. Una esquina es una pequeña porción de una imagen que es rica en información local y, por lo tanto, es probable que se reconozca en otra imagen. A continuación se describen los algoritmos detectores evaluados en esta tesis.

- **Harris-Shi-Tomasi**. Este detector de características fue desarrollado por Harris y Stephens (1988). El algoritmo detecta esquinas como ubicaciones en la imagen donde la señal cambia en dos dimensiones. Esto se logra utilizando la función de autocorrelación dada por

$$c(x, y) = [\Delta x, \Delta y] M [\Delta x, \Delta y]^T, \tag{105}$$

  donde Δx y Δy son los desplazamientos de una pequeña ventana centrada en (x, y). La matriz M representa la estructura de intensidad de un vecindario local.

Es una matriz de 2 × 2 calculada a partir de las derivadas de la imagen

$$M = \begin{bmatrix} \frac{\partial^2 I(i,j)}{\partial x^2} & \frac{\partial^2 I(i,j)}{\partial xy} \\ \frac{\partial^2 I(i,j)}{\partial xy} & \frac{\partial^2 I(i,j)}{\partial y^2} \end{bmatrix}, \tag{106}$$

donde (i, j) son los índices de los valores en la ventana W sobre la imagen I. La ubicación de un punto se obtiene haciendo la supresión máxima en una región de 3 × 3 usando la siguiente función

$$\text{esquinas} = \det(M) \alpha \text{traza}(M)^2. \tag{107}$$

- **FAST**. El detector de características FAST (Rosten y Drummond, 2006) analiza los valores de intensidad de los píxeles en un círculo de radio r alrededor de un punto candidato p. p es clasificado como esquina si existe un arco contiguo de al menos n píxeles que son más brillantes o más oscuros que p por un umbral t. Los autores usaron $r = 3$ y $n = 9$. El algoritmo fue optimizado entrenando un árbol de decisiones para probar la menor cantidad de píxeles posibles para clasificar un píxel candidato como esquina o no esquina. Con este árbol de decisión, solo se prueban 2.26 píxeles para cada candidato en promedio, mientras que con el algoritmo original se prueban 2.8. El algoritmo FAST no proporciona inherentemente una medida de aptitud para una característica detectada. Para aplicar la supresión no-máxima, se calcula la siguiente función de aptitud para cada punto candidato

$$c(p) = \max\{\sum_{q \in S_+} |I_q - I_p| - t, \sum_{q \in S_-} |I_q - I_p| - t\}, \tag{108}$$

donde S_+ es el subconjunto de los píxeles dentro del círculo que son más brillantes que p (por t) y S_- el subconjunto de píxeles más oscuros que p (por t).

- **SIFT**. Algoritmo desarrollado por Lowe (2004) que ha sido muy popular debido a su repetibilidad e invarianza a escala y rotación. Para detectar puntos, SIFT obtiene máximos locales 3D en el espacio DoG (Diferencia de Gausianas), el cual se puede obtener mediante la resta de escalas sucesivas en la imagen original. Para mejorar el tiempo de ejecución del método se construye una representación piramidal del espacio de escalas. La precisión de subpíxeles se realiza ajustando una función cuadrática en los puntos detectados $x = (x, y, \sigma)$. La ubicación de los

subpíxeles se obtiene aplicando la expansión de la serie de Taylor en la imagen alrededor del punto y luego diferenciando e igualando a cero

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T\frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x}. \tag{109}$$

Para rechazar los puntos que se encuentran a lo largo de los bordes, la traza y el determinante de la matriz Hesiana se utilizan de tal manera que un punto que es menor a un umbral r es rechazado

$$\frac{\mathrm{Tr}(H^2)}{\mathrm{Det}(H)} < \frac{(r+1)^2}{r}. \tag{110}$$

El siguiente paso es asignar una orientación a cada característica, calculado los gradientes dentro de una ventana alrededor de ella y construir un histograma de gradientes. El histograma es usado para detectar la orientación más prominente.

- **SURF**. Al algoritmo SURF (Bay *et al.*, 2008) está inspirado en SIFT. La principal motivación para su desarrollo fue superar la debilidad principal de SIFT: su complejidad computacional y su baja velocidad de ejecución. Se ha demostrado que SURF es más rápido que SIFT sin comprometer el rendimiento. Está basado en una representación en el espacio de escalas de la matriz Hesiana, la cual es eficientemente aproximada usando imágenes integrales. Por lo tanto, las características distintivas de una imagen corresponden a ubicaciones donde el determinante de la matriz Hesiana alcanza un máximo en una vecindad de $(3 \times 3 \times 3)$. Los máximos detectados se interpolan para obtener ubicaciones precisas de subíxeles en el espacio de escalas.

- **STAR**. El detector de características STAR es una derivación del detector CenSurE (Agrawal *et al.*, 2008). El algoritmo usa una aproximación del filtro Laplaciano de Gausianas (LoG). La forma circular de la máscara en el detector CenSurE se remplaza por una aproximación que permite preservar la invarianza rotacional y permite el uso de imágenes integrales para un cálculo eficiente. El espacio de escalas es construido sin interpolación al aplicar máscaras de diferentes tamaños.

- **ORB**. El algoritmo ORB (Rublee *et al.*, 2011) hace una modificación a FAST para detectar características invariantes a escala por medio de una construcción de una escala piramidal de la imagen. En cada escala características FAST son

detectadas, la medida de esquina de Harris se emplea para clasificarlas y solo retener las *n* mejores basadas en un umbral. Para obtener invarianza a la rotación se utilizan momentos de primer orden para calcular la orientación local a través de la intensidad del centroide, el cual resulta del promedio ponderado de la intensidad de los píxeles en la ventana local.

- **BRISK**. El detector BRISK (Leutenegger *et al.*, 2011) está basado en FAST. En general, el algoritmo consiste de tres partes: un patrón de muestreo, compensación de orientación y pares de muestreo. Tomar un patrón de muestreo alrededor de un punto candidato se refiere a puntos dispersos en un conjunto de círculos concéntricos, que se utilizan para determinar si el punto es una esquina o no. Entonces estos pares se separan en dos subconjuntos: pares de distancia corta y pares de distancia larga. Para lograr invarianza a rotación, la dirección de cada punto se determina tomando la suma del gradiente local calculado entre los pares de distancia larga, los pares de distancia corta se giran según las orientaciones obtenidas.

3.3.2. Descripción

La información descriptiva sobre una característica visual se codifica en su descriptor. El descriptor representa la ventana local de la característica para que sea mucho más fácil de reconocer cuando aparece en otra imagen diferente. A continuación de describen los algoritmos detectores evaluados en esta tesis.

- **SIFT**. Para obtener descriptores que sean invariantes a rotación, se calcula un histograma de orientaciones a partir de la orientación del gradiente de cada máximo local de la función DoG dentreo de una región alrededor de la característica visual. El vector descriptor considera la dirección de la característica visual en que el gradiente es máximo. Típicamente, una región de 16×16 se determina colocando la característica visual en el centro. Esta región se divide en 4×4 subregiones con 8 bins de orientación en cada una. Finalmente el vector contiene 128 elementos, los descriptores significativos que se extraen de las imágenes son

compactos, altamente distintivos y robustos a cambios de iluminación, escala y puntos de vista.

- **SURF**. Para calcular el descriptor, primero la región de vecindario de cada característica detectada se divide en varias regiones cuadradas de 4×4. Luego, se calcula la respuesta de onda de Haar 2D en cada subregión. Este procedimiento puede calcularse con ayuda de la imagen integral. Cada respuesta aporta 4 valores al vector descriptor, por lo que cada característica se describe con un vector de 64 dimensiones.

- **BRIEF**. BRIEF fue desarrollado por Calonder *et al.* (2010) y fué el primer descriptor binario binario propuesto. Es calculado directamente de los valores de intensidad para evitar altos costos computacionales. Dada una característica visual y una ventana w de un tamaño determinado alrededor de ella. El descriptor B_w se construye a partir de un conjunto S de n pares de píxeles definidos como: $S = \{s_i\}_{i=1\ldots n} = \{P_{i,1}, P_{i,2}\}_{i=1\ldots n}$. Donde $P_{i,1} = (x_{i,1}, y_{i,1})$ y $P_{i,2} = (x_{i,2}, y_{i,2})$ se refieren a un par de coordenadas de muestreo en la ventana w. Después de suavizar la ventana usando un kernel gausiano, para cada par $(P_{i,1}, P_{i,2})$, la intensidad suavizada de $P_{i,1}$ se compara con $P_{i,2}$ para obtener un solo bit de acuerdo con la siguiente regla

$$T(w, s_i) = \begin{cases} 1 & \text{si } P_{i,1} > P_{i,2} \\ 0 & \text{de otro modo} \end{cases}. \tag{111}$$

  El descriptor binario B_w de n bits se define como

$$B_w = \sum_i 2^{i-1} T(w, s_i). \tag{112}$$

  Los autores usaron $n = 128, 256, 512$ para tener un compromiso entre velocidad, precisión y memoria. BRIEF no tiene un patrón de muestreo, es decir, los pares muestreados son tomados de manera aleatoria usando una distribución gausiana isotrópica. BRIEF trabaja bien en presencia de transformaciones fotométricas como desenfoque, cambios de iluminación y transformaciones perspectivas.

- **ORB**. El descriptor de ORB está basado en BRIEF. Para obtener invarianza a la rotación se calculan los momentos de primer orden para calcular la orientación

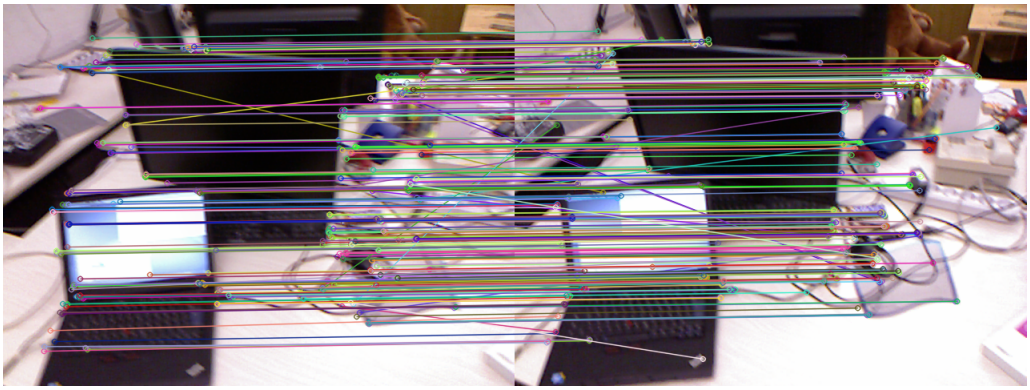
local a través de un centroide de intensidad que se refiere al promedio ponderado de las magnitudes de píxeles en el área local. El descriptor es calculado sobre las áreas rotadas y representado como una cadena binaria.

- **BRISK**. Para todos los pares, se comparan valores de la intensidad del primer y segundo punto del par, es decir, si el valor del primer punto es mayor que el del segundo, la salida es 1, de lo contrario 0. Por lo tanto después de procesar los 512 pares, conduce a un descriptor de 512 bits de longitud.
- **FREAK**. Propuesto por Ortiz (2012), comparte similitud con BRISK al utilizar un patrón de muestreo inspirado en el sistema visual humano y, más precisamente, en la estructura de la retina con más puntos de muestreo cerca del centro. Además, el uso de diferentes tamaños de kernels para suavizar cada punto de muestreo y reducir la sensibilidad al ruido. Sin embargo, utiliza círculos superpuestos que cambia de tamaño exponencialmente donde cada círculo representa las desviaciones estándar del kernel gausiano. FREAK utiliza el aprendizaje no supervisado para aprender del conjunto de pares de muestreo. Es rápido de calcular y proporciona robustez contra la rotación y los cambios de escala.
- **LATCH**. Propuesto por Levi y Hassner (2016), es un descriptor binario que utiliza pequeñas comparaciones de regiones en lugar de comparaciones de un solo píxel. Usa tres pares de puntos en lugar de dos para el muestreo. Hace uso del aprendizaje no supervisado, para aprender del conjunto óptimo de tripletas. El valor binario del descriptor resulta de comparar la norma Frobenious entre cada triplete.

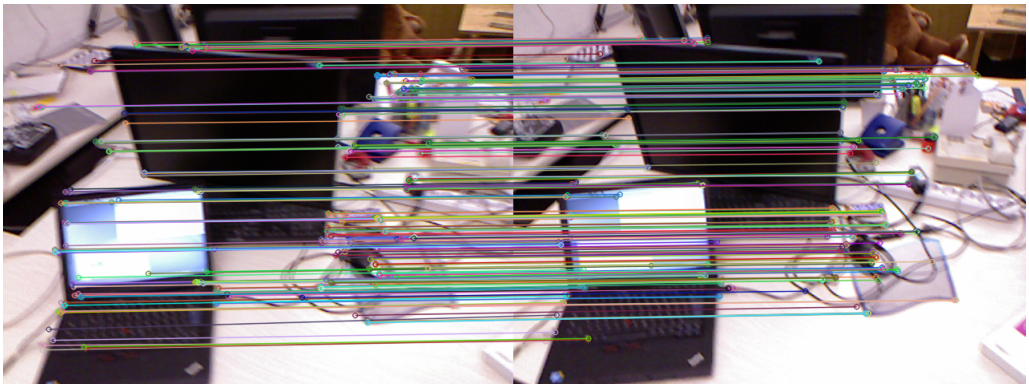
3.3.3. Asociación de datos

La correspondencia entre imágenes consiste en la búsqueda de características extraídas por un detector en una imagen y su correspondiente característica en otra imagen. Una correcta asociación de datos es esencial para una estimación precisa del movimiento de la cámara.

La técnica general consiste en emplear un algoritmo de fuerza bruta, es decir, comparar el descriptor de cada característica de la imagen base I_b con cada descriptor en



(a)



(b)

Figura 10. Cálculo de la matriz de homografía usando el enfoque RANSAC. (a) Las posibles correspondencias son encontradas mediante un algoritmo de fuerza bruta. (b) El conjunto de outliers es removido minimizando el error de reproyección.

la imagen de referencia I_r y relacionar aquellas cuya distancia sea menor. Las métricas de distancia más común son la distancia de Hamming para descriptores binarios y la distancia Euclidiana para descriptores de punto flotante. Sin embargo, la distancia no es criterio suficientemente fuerte para una asociación de datos correcta debido a la alta dimensionalidad del espacio de características, por lo tanto, es necesario emplear algoritmos de estimación robusta como RANSAC (Fischler y Bolles, 1981) para rechazar falsas correspondencias (outliers).

3.3.3.1. Random Sample Consensus (RANSAC)

RANSAC (Random Sample Consensus) (Fischler y Bolles, 1981) es uno de los esquemas de estimación robusta más usados. La idea principal del algoritmo es relativamente simple:

1. Seleccionar n puntos de un conjunto de datos de manera aleatoria. n es el número de puntos que son suficientes para determinar de manera única los parámetros del modelo buscado. Por ejemplo, si se busca calcular la matriz de homografía entre dos imágenes 4 puntos son necesarios. Los n puntos seleccionados forman el conjunto mínimo.

2. Calcular los parámetros únicos del modelo a partir del conjunto mínimo. Estos parámetros forman la hipótesis actual.

3. Determinar el conjunto de inliers para la hipótesis actual. El conjunto de inliers consiste de aquellos puntos que tienen una distancia por debajo de un umbral τ del modelo determinado en el último paso. Por ejemplo, para el caso de la matriz de homografía, la métrica de distancia sería el error de reproyección, es decir, proyectar los puntos de I_b en I_r usando la matriz de homografía actual (hipótesis) y rechazar aquellos que tengan una distancia Euclidiana mayor a τ píxeles.

4. Repetir los pasos 1-3 k veces.

5. Regresar los parámetros de la hipótesis con el conjunto de inliers más grande.

En la Fig. 10 se muestra el resultado de calcular la matriz de homografía usando el enfoque de RANSAC. El número de iteraciones k es calculando en base a la idea que el algoritmo debería con probabilidad p elegir al menos un conjunto mínimo que no contenga outliers. En este caso, se espera que el conjunto de inliers resultante sea más grande que los demás. Si se conociera ϵ, la relación entre outliers e inliers en el conjunto de datos, un punto elegido al azar sería un inlier con probabilidad $1-\epsilon$. Entonces, la probabilidad de que todos los n puntos en el conjunto mínimo actual sea inliers es $(1-\epsilon)^n$. Po lo tanto, la probabilidad de que al menos un punto sea un outlier es $1-(1-\epsilon)^n$ y finalmente la probabilidad de que cada uno de los k conjuntos mínimos contenga al menos un outlier es $(1-(1-\epsilon)^n)^k$. Dado que se requiere que esta probabilidad esté por debajo de $1-p$

$$k = \frac{\log(1-p)}{\log(1-(1-\epsilon)^n)} \tag{113}$$

Recientemente, se han propuesto una serie de extensiones al algoritmo RANSAC para incrementar el rendimiento. Mientras RANSAC simplemente cuenta el número

de inliers y elige la hipótesis que obtuvo el mayor número de inliers, M-SAC elige la hipótesis para la cual la suma de todos los errores residuales (la distancia desde un punto hasta el modelo actual) fue mínima.

Otro enfoque es MLESAC (Torr y Zisserman, 2000) el cual escoge la hipótesis con la máxima verosimilitud en lugar de la que tiene la suma mínima de errores residuales. Nistér (2005) propuso P-RANSAC para reducir el tiempo empleado en la evaluación de hipótesis posiblemente malas. PROSAC (Chum y Matas, 2005) propone utilizar un proceso de muestreo guiado (semi-aleatorio). Raguram *et al.* (2009) incorporó explícitamente información de incertidumbre en el esquema RANSAC y propuso un algoritmo llamado Cov-RANSAC.

3.4. Registro 3D

Los algoritmos de registro 3D tratan de encontrar la transformación óptima entre dos nubes de puntos. La transformación óptima es definida como la transformación que mejor alinea las nubes de puntos dentro de un sistema de coordenadas global (Fig. 11). Dada una nube de puntos origen $A = \{a_i\}$, donde $a_i \in \mathbb{R}^3$ para $i = 1, \ldots, N_A$, y una nube de puntos destino $B = \{b_j\}$, donde $b_j \in \mathbb{R}^3$ para $j = 1, \ldots, N_B$, la matriz de transformación óptima $\mathbf{T}$ es definida como

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix}, \tag{114}$$

donde $\mathbf{R} \in SO(3)$ es la matriz de rotación y $\mathbf{t}$ es el vector de traslación. $\mathbf{T}$ transforma puntos del sistema de coordenadas de la nube A al sistema de coordenadas de la nube B. Este problema es comúnmente modelado como un problema de optimización de una función de aptitud $\Gamma : SE(3) \rightarrow \mathbb{R}$ de la forma

$$\mathbf{T}^* = \underset{\mathbf{T}}{\operatorname{argmin}}\ \Gamma(\mathbf{T}). \tag{115}$$

La función anterior es específica de cada algoritmo y, en muchos casos, es optimizada por un método iterativo antes de arrojar la solución final. A continuación se describen los algoritmos utilizados en esta tesis.

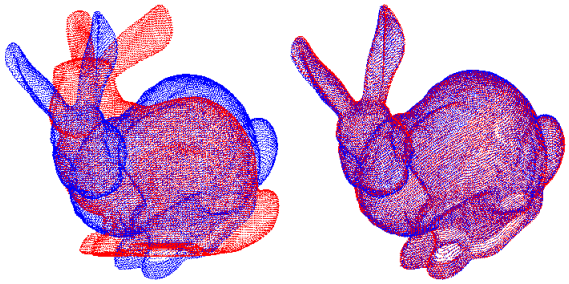


Figura 11. Ejemplo de Registro 3D. Antes y después de la alineación.

3.4.1. Iterative Closest Point (ICP)

Un algoritmo clásico de registro 3D es ICP (Iterative Closest Point) (Besl y McKay, 1992). El algoritmo está compuesto por dos etapas fundamentales. Primero se estiman las correspondencias entre las dos nubes de puntos basadas en una cierta métrica. Después se estima la transformación relativa con una solución cerrada para minimizar las distancias entre las correspondencias. Los dos pasos anteriores son realizados de manera iterativa hasta que se cumpla un criterio de convergencia o se alcance un número máximo de iteraciones. ICP sufre el riesgo de quedar atrapado en mínimos locales, especialmente en casos de grandes movimientos de la cámara o por la falta de estructura 3D en la escena observada. Una alternativa para evitar mínimos locales es usar una alineación basada en características (como SIFT, ORB) y usar ICP como paso de refinamiento (Henry *et al.*, 2012; Endres *et al.*, 2014), ya que, los métodos basados en características mejoran la probabilidad de converger al mínimo global.

En la Fig. 12 se muestran los puntos observados por un cámara desde dos posiciones diferentes A (origen) y B (destino), ambas en relación a un sistema de coordenadas global W. El algoritmo ICP convencional usa el par de nubes de puntos P_A y P_B formadas por los sistemas de coordenadas A y B para encontrar de manera iterativa la matriz de transformación relativa de cuerpo rígido óptima $\mathbf{T}^*$, de tal manera que después de transformar P_A con $\mathbf{T}^*$, las observaciones de A se alineen con las observaciones en B (Fig. 12). ICP es un método iterativo, la matriz $\mathbf{T}^*$ se inicializa con la matriz identidad, la kt-ésima iteración del algoritmo se describe a continuación:

1. Para cada punto en la nube P_A buscar el punto más cercano en la nube P_B. Para el i-ésimo punto en P_A, el índice del punto correspondiente en P_B se denota como

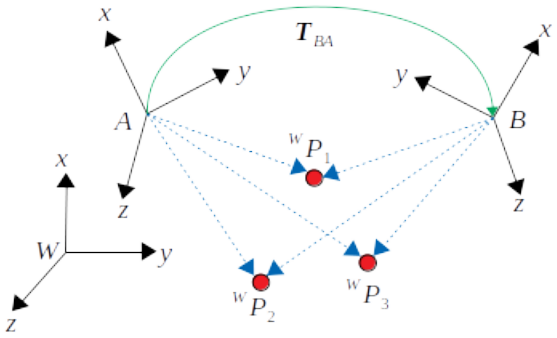


Figura 12. Observación de puntos desde dos posiciones diferentes de la cámara.

$c(i)$, donde

$$c(i) = \underset{j}{\operatorname{argmin}} \| \mathbf{T}^* P_A(i) - P_B(j) \| . \tag{116}$$

2. Calcular la matriz de transformación óptima incremental $\mathbf{T}_k$ que minimiza la distancia entre las correspondencias establecidas

$$\mathbf{T}_k = \underset{\mathbf{T}}{\operatorname{argmin}} \sum_i \| \mathbf{T}\mathbf{T}^* P_A(i) - P_B(c(i)) \|^2 . \tag{117}$$

   La solución a esta ecuación se encuentra de forma cerrada (Eggert *et al.*, 1997), por ejemplo, usando Singular Value Decomposition (SVD) (Arun *et al.*, 1987).

3. Actualizar $\mathbf{T}^*$

$$\mathbf{T}^* \leftarrow \mathbf{T}_k \mathbf{T}^* . \tag{118}$$

4. Los pasos 1 y 2 se realizan de manera iterativa hasta que la transformación incremental es más pequeña que un umbral o se alcance el número máximo de iteraciones.

3.4.2. Generalized Iterative Closest Point (GICP)

GICP (Segal *et al.*, 2009) se basa en adjuntar un modelo probabilístico al paso de minimización. La técnica mantiene el resto del algoritmo sin cambios para reducir la complejidad y mantener la velocidad.

En el modelo probabilístico se asume la existencia de un conjunto subyacente de puntos $\hat{A} = \{\hat{a}_i\}$ y $\hat{B} = \{\hat{b}_i\}$ que generan A y B de acuerdo con $a_i \sim \mathcal{N}(\hat{a}_i, C_i^A)$ y $b_i \sim$

$\mathcal{N}(\hat{b}_i, C_i^B)$. En este caso, C_i^A y C_i^B son las matrices de covarianza asociadas a los puntos medidos. Si se asume correspondencias perfectas (geométricamente consistentes sin errores de oclusión o muestreo), y la transformación correcta $\mathbf{T}^*$, entonces

$$\hat{b}_i = \mathbf{T}^* \hat{a}_i. \tag{119}$$

Para una transformación arbitraria $\mathbf{T}$, se define la distancia entre los puntos como $d_i^{(\mathbf{T})} = b_i - \mathbf{T}a_i$. Dado que a_i y b_i provienen de distribuciones gausiana independientes, se obtiene

$$\begin{aligned} d_i^{(\mathbf{T}^*)} &\sim \mathcal{N}(\hat{b}_i - \mathbf{T}^* \hat{a}_i, C_i^B + \mathbf{T}^* C_i^A (\mathbf{T}^*)^T) \\ &= \mathcal{N}(0, C_i^B + \mathbf{T}^* C_i^A (\mathbf{T}^*)^T). \end{aligned} \tag{120}$$

La transformación $\mathbf{T}$ es encontrada iterativamente usando un estimador de máxima verosimilitud (MLE)

$$\mathbf{T} = \underset{\mathbf{T}}{\operatorname{argmin}} \sum_i d_i^{(\mathbf{T})^T} (C_i^B + \mathbf{T} C_i^A \mathbf{T}^T)^{-1} d_i^{\mathbf{T}}. \tag{121}$$

El algoritmo ICP convencional puede ser visto como un caso especial estableciendo $C_i^B = I$ y $C_i^A = 0$. Esta formulación puede ser usada para representar cualquier variante de ICP como la métrica punto a punto y punto a plano. Sin embargo, GICP propone una métrica plano a plano donde se asume que los puntos son muestreados de superficies localmente planas. En este modelo se supone que la covarianza de un punto es pequeña en la dirección de la normal en ese punto y grande en todas las demás direcciones. En el caso de un punto con e_1 como su superficie normal, la matriz de covarianza se convierte en

$$\begin{bmatrix} \epsilon & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{122}$$

donde ϵ es una constante pequeña que representa la covarianza a lo largo de la normal. Sean u_i y v_i las normales respectivas de b_i y a_i; C_i^B y C_i^A se calculan rotando la matriz de covarianza anterior para que el término ϵ represente la incertidumbre a lo

largo de la superficie normal

$$C_i^B = \mathbf{R}_{u_i} \cdot \begin{bmatrix} \epsilon & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \mathbf{R}_{u_i}^T, \tag{123}$$

$$C_i^A = \mathbf{R}_{v_i} \cdot \begin{bmatrix} \epsilon & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \mathbf{R}_{v_i}^T. \tag{124}$$

La transformación **T**, es calculada mediante la Ec. 121. La naturaleza plano a plano del algoritmo GICP permite alinear nubes de puntos correctamente sin caer en mínimos locales, algo muy común en el algoritmo ICP convencional, ya que, las normales locales no son incluidas.

3.5. Representaciones de mapas

Los métodos de SLAM utilizan diferentes representaciones de mapas para recopilar información sobre el entorno. Dichas representaciones tienen diferentes características y su uso depende de la aplicación final, sin embargo, todas comparten la misma idea de proveer una referencia para la información obtenida por las cámaras del robot.

Crear un mapa desde cero a partir del aprendizaje del entorno del robot aumenta su autonomía y permite que el robot se adapte a los cambios en el entorno. Esto involucra problemas como:

- Alta dimensionalidad en el espacio de todos los posibles mapas y observaciones.
- Incertidumbre de la pose del robot y observaciones.
- Tamaño del mapa y resolución.
- Diferentes lugares con apariencia muy similar (Perceptual aliasing).

A continuación, se detallan los tipos de mapas más usados en SLAM visual: basados en características, topológicos y métricos. Algunas veces también se utilizan mapas híbridos que involucran alguna combinación de ellos.

3.5.1. Mapas de características

Este tipo de mapas es probablemente uno de los más usados en SLAM visual. El mapa es una representación dispersa de la escena formada por puntos de referencia presentes en el entorno (Fig. 13). Cada punto de referencia contiene su posición 3D y una descripción única. Cada vez que la cámara obtiene una observación, se comparan las características en el mapa con las características obtenidas al tiempo *t* realizando una asociación de datos, la cual hace coincidir las características en el mapa con su correspondiente observación. Este es un paso crucial para el rendimiento del sistema, ya que una incorrecta asociación de datos puede llevar a consecuencias catastróficas en el mapa.

Las características en el mapa sirven para localizar al robot dentro del mapa a través de mediciones de la diferencia entre las posiciones de características estimadas y las características correspondientes en el mapa. Estos mapas son poco útiles para tareas diferentes a la localización debido a que no es posible representar volúmenes ocupados y libres, requerimientos necesarios para tareas como evasión de obstáculos y planificación de rutas.

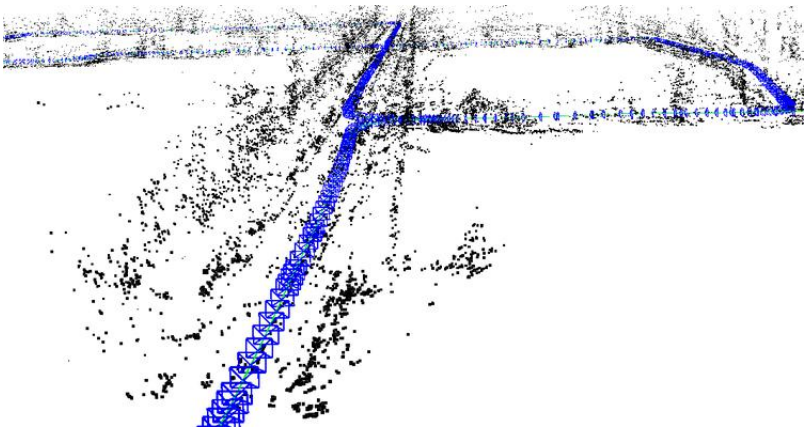


Figura 13. Mapa de características.

3.5.2. Mapas topológicos

Un mapa topológico puede ser representado como un grafo de poses donde los nodos almacenan información sobre lugares específicos en el entorno y las aristas son usualmente una transformación que unen un nodo con otro (Fig. 14). Cada nodo almacena observaciones de la cámara representando el entorno en una ubicación especifica junto con la posición del nodo en el mapa. Las aristas contienen información que permite al robot calcular el costo de ir de un nodo a otro. De esta manera, se pueden utilizar algoritmos para recorrer grafos como Dijkstra para obtener la ruta más corta entre dos nodos.

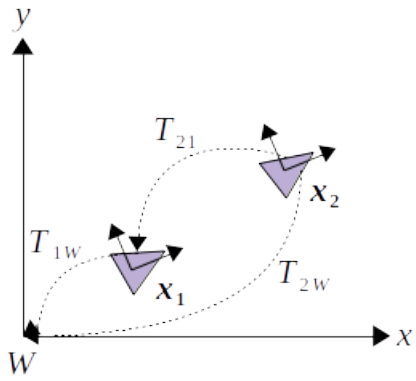


Figura 14. Poses del robot representadas en un mapa topológico.

Un problema importante es cuando crear nodos, por ejemplo, una estrategia podría estar basada en la distancia, es decir, crear un nodo cada vez que el robot haya avanzado 1 metro desde el nodo anterior. Sin embargo, esta estrategia podría crear un grafo con información redundante y con muchos nodos. Otra estrategia es elegir nodos en relación con la diferencia en su apariencia, de manera que se minimicen el número de nodos y se maximice la información contenida en cada nodo. Por lo tanto, el reconocimiento visual de lugares juega un papel fundamental para reducir el error en el mapa.

3.5.3. Mapas métricos

Los mapas métricos representan el entorno de la manera más precisa posible manteniendo mucha información acerca de los detalles del entorno, tales como: distancias,

mediciones ó tamaños. Usualmente están en base un sistema de coordenadas global. Esta representación es apropiada para localización y navegación de robots, así como para evitar obstáculos. Sin embargo, estos mapas son más difíciles de construir y mantener, además son computacionalmente más exigentes. En la Fig. 15 se muestra un ejemplo de este tipo de mapas creado con el sistema KinectFusion (Izadi *et al.*, 2011).

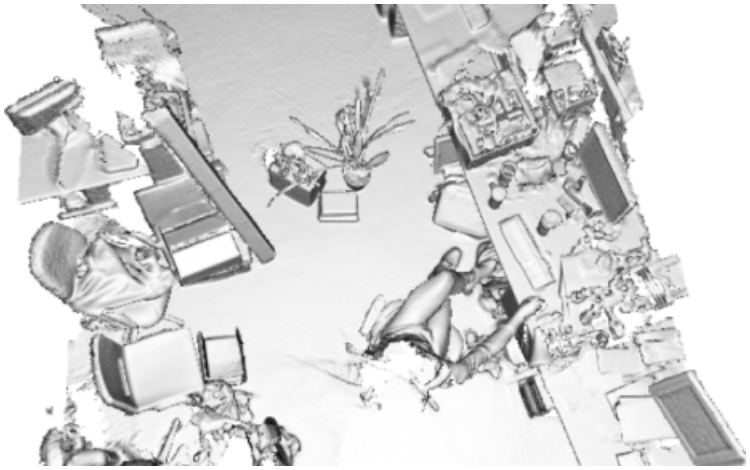


Figura 15. Mapa métrico representado mediante surfeles. En este tipo de mapas es posible representar objetos y diferenciar entre volúmenes libres y ocupados.

La resolución es un factor muy importante para la complejidad de los algoritmos, una mayor resolución lleva a más costo computacional. Como en cada representación del mapa, la detección de ciclos es muy importante para crear mapas consistentes.

3.6. Optimización

3.6.1. Bundle adjustment

Bundle adjustment (BA) es el problema de refinar una reconstrucción visual para producir estimaciones de la posición de puntos 3D $\mathcal{P}$ y posiciones de la cámara $\mathcal{C}$ óptimas. Óptimo significa que los parámetros estimados son encontrados mediante minimización de alguna función de costo que cuantifica el error de ajuste del modelo y que la solución es óptima con respecto a las variaciones de la estructura y la cámara (Triggs *et al.*, 2000). El nombre se refiere a los 'bundles' de rayos de luz que salen de

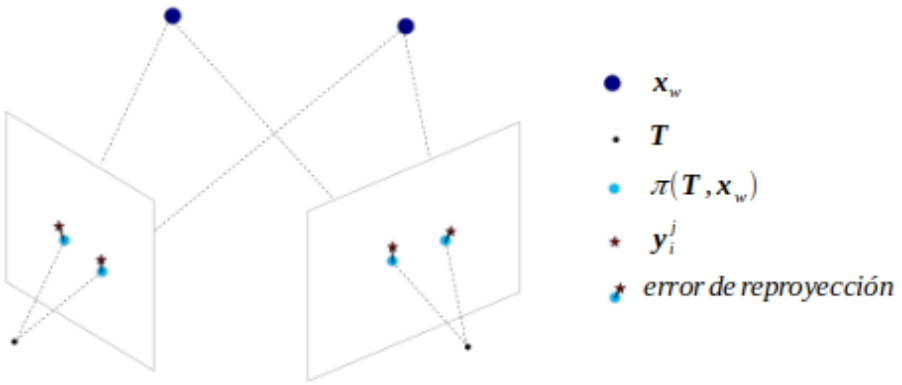


Figura 16. Error de reproyección.

cada punto 3D y convergen en cada centro de la cámara, que se ajustan ’adjusted’ de manera óptima con respecto a las poses de los puntos y la cámara.

Dadas correspondencias entre puntos 3D en el sistema de coordenadas global $\mathbf{x}_w$ y una característica visual 2D $\mathbf{y}_i$ en una cámara monocular, el error de reproyección $\mathbf{e}_{rep}$ (Fig. 16) es calculado como

$$\mathbf{e}_{rep} = \mathbf{y}_i - \pi(\mathbf{T}, \mathbf{x}_w), \tag{125}$$

donde $\mathbf{T}$ es la pose de la cámara que transforma puntos de sistema de coordenadas global al sistema de coordenadas de la cámara y π es la función de proyección del modelo de la cámara que puede ser monocular $\pi_m(\cdot)$ o estéreo $\pi_s(\cdot)$. BA minimiza el error de reproyección y es la técnica de optimización estándar en sistemas SLAM visual modernos

$$\{\mathbf{x}_w^j, \mathbf{T}_i | \forall j \in \mathcal{P}, \forall i \in \mathcal{C}\} = \underset{\mathbf{x}_w^j, \mathbf{T}_i}{\operatorname{argmin}} \sum_{i,j} \rho(\| \mathbf{y}_i^j - \pi(\mathbf{T}_i, \mathbf{x}_w^j) \|_{\Sigma_i^j}^2), \tag{126}$$

donde $\mathbf{y}_i^j$ es la característica visual asociada a un punto 3D $\mathbf{x}_w^j$ en la cámara i, Σ_i^j es la matriz de covarianza de la ubicación de la característica $\mathbf{y}_i^j$ en la imagen de la cámara i y ρ es una función robusta de costo para reducir las falsas correspondencias. BA es un problema de optimización de mínimos cuadrados no lineal que puede ser resuelto por algún método iterativo explicado en la sección anterior.

3.6.2. Grafo de poses

BA puede ser demasiado costoso en mapas grandes, y más aún si la solución inicial está lejos del óptimo. Este es el caso de cierres de ciclo, donde el error acumulado en la trayectoria hace que el estado del mapa esté lejos del mapa óptimo y globalmente consistente. Una aproximación a la solución de BA son los mapas llamados grafos de poses. El grafo de poses solo contiene la trayectoria de la cámara a través del mundo 3D, mientras que los distintos puntos de referencia no son parte del mapa, es decir, se descarta la estructura y solo se optimizan las posiciones de la cámara minimizando el error de transformación relativa. Este mapa es expresado mediante un grafo donde los vértices representan las poses del robot y las aristas representan las restricciones espaciales entre las poses.

Existen dos tipos de restricciones en el grafo de poses. Restricciones de odometría que conectan dos estados consecutivos (Ec. 50). Restricciones de cierre de ciclo que conectan nodos no necesariamente consecutivos cuando el robot reconoce un lugar previamente visitado.

Mientras el robot se mueve a través del entorno, usa odometría visual para estimar su posición. Por lo general, debido a los errores acumulados de odometría, las discrepancias grandes empiezan a mostrarse incluso después de un tiempo pequeño. Estos errores en el mapa son más evidentes cuando el robot cierra ciclos, es decir, cuando vuelve a visitar áreas que había recorrido antes (Fig. 17).

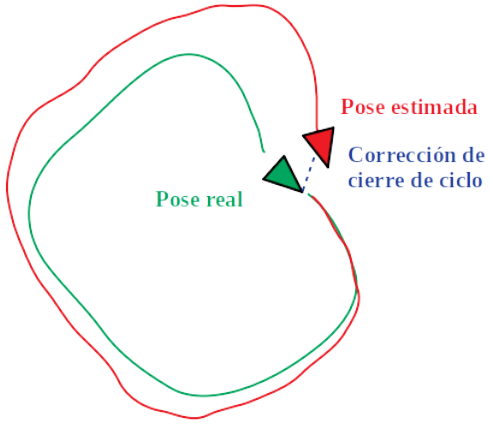


Figura 17. Ejemplo de cierre de ciclo donde se observa el error acumulado en la trayectoria.

Debido a la acumulación de errores, es necesario optimizar el grafo de poses cuan-

do se detectan ciclos para obtener un mapa global óptimo. Esto se logra minimizando el error de transformación relativa. Dadas dos poses absolutas de cámara $\mathbf{T}_i \in SE(3)$ y $\mathbf{T}_j \in SE(3)$, y una medición de la transformación relativa $\hat{\mathbf{T}}_{ij} \in SE(3)$, el error de transformación relativa $\mathbf{e}_{rel}$ se define en el espacio tangente como (Strasdat, 2012)

$$\mathbf{e}_{rel}(i,j) = \text{Log}_{SE(3)}(\hat{\mathbf{T}}_{ij}\mathbf{T}_j\mathbf{T}_i^{-1}). \tag{127}$$

Por lo tanto, dado un conjunto de aristas E en el grafo de poses, la función de costo a ser minimizada es

$$F = \sum_{(i,j)\in E} \rho(\| \mathbf{e}_{rel}(i,j) \|^2_{\Sigma_{i,j}}), \tag{128}$$

el cual resulta en un problema de optimización por mínimos cuadrados no lineal que puede ser resuelto por algún método explicado en la sección anterior.

Después de aplicar la optimización del grafo de poses es posible realizar algunas iteraciones de BA para obtener la solución óptima, el cual convergerá más rápido dado que el resultado de la optimización del grafo de poses es cercano al óptimo.

Capítulo 4. Evaluación de algoritmos extractores

Uno de los problemas principales en SLAM visual basado en características es como seleccionar puntos distintivos de una imagen para usarlos como puntos de referencia confiables. Típicamente, estos puntos son invariantes a rotación, traslación, escala y cambios de iluminación, sin embargo, en la práctica, la estabilidad de los puntos no siembre se mantiene, haciendo difícil el problema de búsqueda de correspondencias. Como se describió en el capítulo anterior, el proceso de extracción de características está compuesto por dos etapas: detección y descripción.

Se han propuesto muchos algoritmos detectores y descriptores de características, sin embargo, la pregunta de cuál combinación detector-descriptor es más adecuado para SLAM aún está abierta. Por lo tanto, se realizó un estudio comparativo entre los algoritmos más utilizados en SLAM visual para seleccionar los mejores candidatos. Los algoritmos evaluados se describen en la Tabla 2 (Garcia-Fidalgo y Ortiz, 2015).

El conjunto de datos usado para llevar a cabo los experimentos fue el propuesto por Mikolajczyk y Schmid (2005), ya que, incluye deformaciones en las imágenes comunes en SLAM visual, además incluye las matrices de homografía entre la imagen base y cada una de las imágenes de referencia. En la Fig. 18 se muestran algunas imágenes del conjunto de datos.

4.1. Tiempo de Ejecución

El procesamiento en tiempo real es un requisito clave para los sistemas SLAM, debido a esto, la velocidad de los algoritmos extractores de características es un factor importante para la operación en tiempo real. Se probaron todas las posibles combinaciones de detectores y descriptores mostrados en la Tabla 2. El tiempo que tarda cada algoritmo varía según el número de características encontradas en la imagen, por esta razón se calcula el tiempo promedio de extracción por característica. En la Tabla 3 se muestran los resultados en microsegundos.

El Kinect de Microsoft opera a 30 Hz, es decir, cada 33 ms se captura una nueva imagen, por lo que se necesita una combinación detector-descriptor que sea capaz de

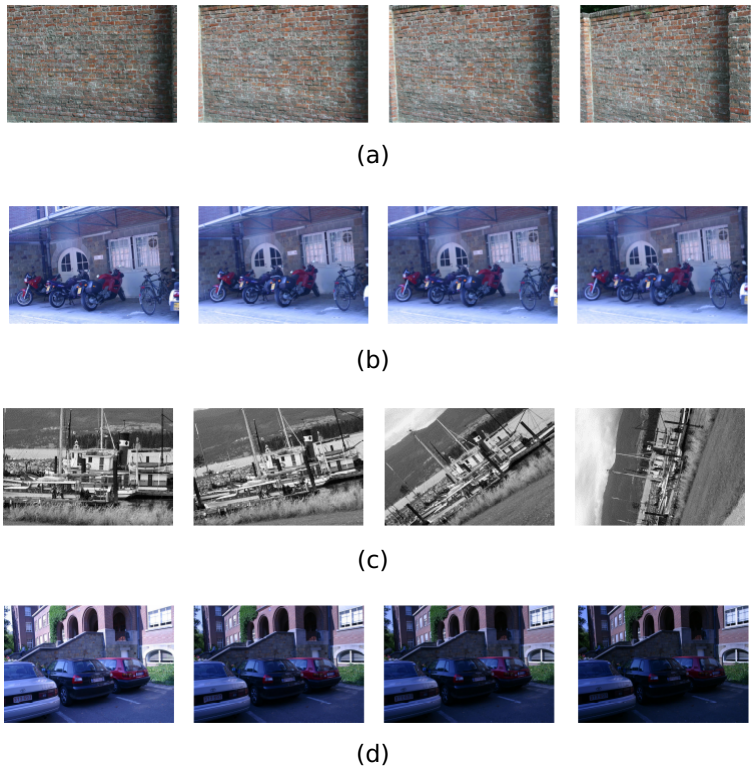


Figura 18. Imágenes del conjunto de datos usado para la evaluación. (a) Cambio de vista. (b) Blur. (c) Rotación y escala. (d) Cambios de iluminación.

extraer 1000 puntos de interés en mucho menos de 33 ms para lograr un seguimiento de la cámara en tiempo real (Mur-Artal *et al.*, 2015). En la Tabla 3 se muestran los resultados de la prueba de velocidad, todas las combinaciones con tiempo de extracción superior a 20 *us* son descartadas y se muestran en color rojo. Los experimentos fueron realizados en un Laptop Intel Core i5-4200U con 8 Gb de RAM.

4.2. Repetibilidad

Un requerimiento importante para SLAM visual es la capacidad de extraer puntos de interés que sean estables y puedan extraerse en una serie de imágenes consecutivas; esto se conoce como criterio de repetibilidad. La repetibilidad se calcula estimando las correspondencias entre la imagen base I_b con cada una de las imágenes de referencia I_r. Las correspondencias se estiman usando la relación de distancia del vecino más cercano (Lowe, 2004) (NNDR). Para cada descriptor de I_b, se buscan los dos descriptores de vecinos más cercanos en I_r para calcular la relación de distancia entre ellos. Si

Tabla 2. Algoritmos de extracción de características.

Algoritmo	Detector	Descriptor	Tipo de componente	Invarianza	
				Rotación	Escala
GFTT (Shi y Tomasi, 1993)	✓			✓	
FAST (Rosten y Drummond, 2006)	✓			✓	
ORB (Rublee *et al.*, 2011)	✓	✓	Bit	✓	✓
BRISK (Leutenegger *et al.*, 2011)	✓	✓	Bit	✓	✓
SIFT (Lowe, 2004)	✓	✓	Flotante	✓	✓
SURF (Bay *et al.*, 2008)	✓	✓	Flotante	✓	✓
STAR (Konolige *et al.*, 2010)	✓			✓	✓
BRIEF (Calonder *et al.*, 2010)		✓	Bit		
FREAK (Ortiz, 2012)		✓	Bit	✓	✓
LATCH (Levi y Hassner, 2016)		✓	Bit	✓	✓

la distancia es menor a un valor de umbral específico, Lowe (2004) sugiere un umbral de 0.8, ya que, ayuda a descartar el 90 % de las falsas correspondencias mientras se descartan solo el 5 % de las correctas), se selecciona el vecino más cercano como correcta correspondencia, de lo contrario se rechazan ambos. Sin embargo, la distancia no es un criterio suficientemente fuerte para asociaciones, ya que, la distancia de los descriptores coincidentes puede variar mucho debido a la alta dimensionalidad del espacio de características lo que ocasiona que aún existan falsas correspondencias, las cuales debe ser eliminadas. Una alternativa común es calcular la matriz de homografía mediante un enfoque RANSAC, sin embargo, el conjunto de datos usado (Mikolajczyk y Schmid, 2005) ya proporciona la matriz de homografía, la cual es usada para proyectar los puntos de interés de I_r en I_b, si la distancia Euclidiana entre el par de puntos es menor a un umbral (2.5 píxeles) la correspondencia se considera correcta, sino incorrecta. Esto se conoce como validación geométrica. En la Fig. 19 se muestra un ejemplo del proceso de estimación de correspondencias.

La repetibilidad es calculada usando la siguiente formula

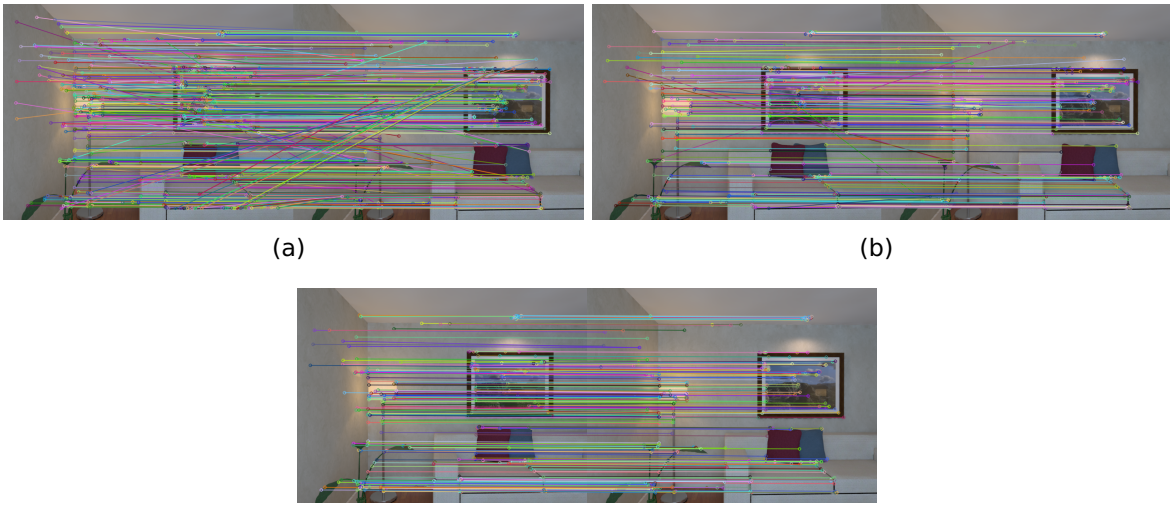
$$\text{repetibilidad} = \frac{|\{x_b \in S_b, x_r \in S_r| \parallel x_b - H^{-1}x_r \parallel < \epsilon\}|}{|S_b|} \cdot 100, \tag{129}$$

Tabla 3. Tiempo promedio de extracción en microsegundos (*us*) por característica visual. Algoritmos con un tiempo promedio mayor que 20 *us* son descartados y marcados en color rojo.

Detectores	Descriptores						
	ORB	BRISK	BRIEF	FREAK	LATCH	SURF	SIFT
GFTT	16	**23**	15	**21**	**53**	**23**	**29**
FAST	6	16	5	10	**46**	**21**	**60**
ORB	14	**21**	11	**28**	**50**	**199**	**309**
BRISK	**40**	**43**	**38**	**52**	**75**	**93**	**146**
SIFT	∞	**144**	**130**	**136**	**162**	**131**	**220**
SURF	**57**	**70**	**54**	**85**	**92**	**139**	**417**
STAR	**102**	**95**	**79**	**87**	**124**	**107**	**424**

donde S_r es el conjunto de características marcadas como verdaderas correspondencias en I_r, S_b es el conjunto de características que se encuentran en I_b, H es la matriz de homografía y ϵ es el umbral en píxeles. Una combinación detector-descriptor perfecta debería detectar los mismos puntos en la imagen base y todas las imágenes de referencia, es decir, repetibilidad = 100 % para cada imagen. Las combinaciones que pasaron la prueba de velocidad fueron evaluadas bajo las siguientes condiciones: cambios de puntos de vista, escala, rotación, iluminación y blur. En la Fig. 20 se muestran los resultados de la prueba de repetibilidad.

De la Fig. 20 se observa el comportamiento de las combinaciones detector-descriptor bajo ciertas condiciones. Debido a la baja repetibilidad se descartan las combinaciones: FAST-FREAK, FAST-BRISK y ORB-BRIEF. Una observación importante es que las combinaciones GFTT-ORB y GFTT-BRIEF tienen la mejor repetibilidad en la mayoría de los casos, excepto para rotación + escala donde ORB-ORB es la mejor combinación. Las combinaciones FAST-ORB y FAST-BRIEF tienen un comportamiento similar a las combinaciones con el detector GFTT.



(a) (b)

(c)

Figura 19. Estimación de correspondencias. (a)Algoritmo de fuerza bruta, se observan muchas falsas correspondencias. (b) Algoritmo NNDR donde el número de falsas correspondencias es menor. (c) Despúes de aplicar la validación geométrica.

4.3. Precision y Recall

Precision y recall son métricas comúnmente usadas para caracterizar la calidad del proceso de estimación de correspondencias, por lo tanto, en este trabajo se usan para determinar la calidad de las combinaciones detector-descriptor. Dadas dos imágenes I_i y I_j se calcula la distancia entre todos los descriptores de los puntos de interés en I_i y I_j. Un par de descriptores coincidentes es el par de descriptores donde la distancia entre ellos es mínima. Precision y recall se definen de la siguiente manera

$$\text{recall} = \frac{\text{verdaderos positivos}}{\text{total de positivos}}, \tag{130}$$

$$1 - \text{precision} = \frac{\text{falsos positivos}}{\text{total de correspondencias}}. \tag{131}$$

El total de verdaderos positivos se refiere a los puntos de interés coincidentes después de aplicar la validación geométrica mediante la matriz de homografía H. Variando el umbral ϵ entre lo que se considera verdaderos y falsos positivos se pueden obtener diferentes valores de precision y recall para general la curva RP. El área bajo la curva RP (AUC) es usada como medida para evaluar el rendimiento de una combinación detector-descriptor. El AUC está delimitada entre 0 y 1, donde 1 indica un rendimiento

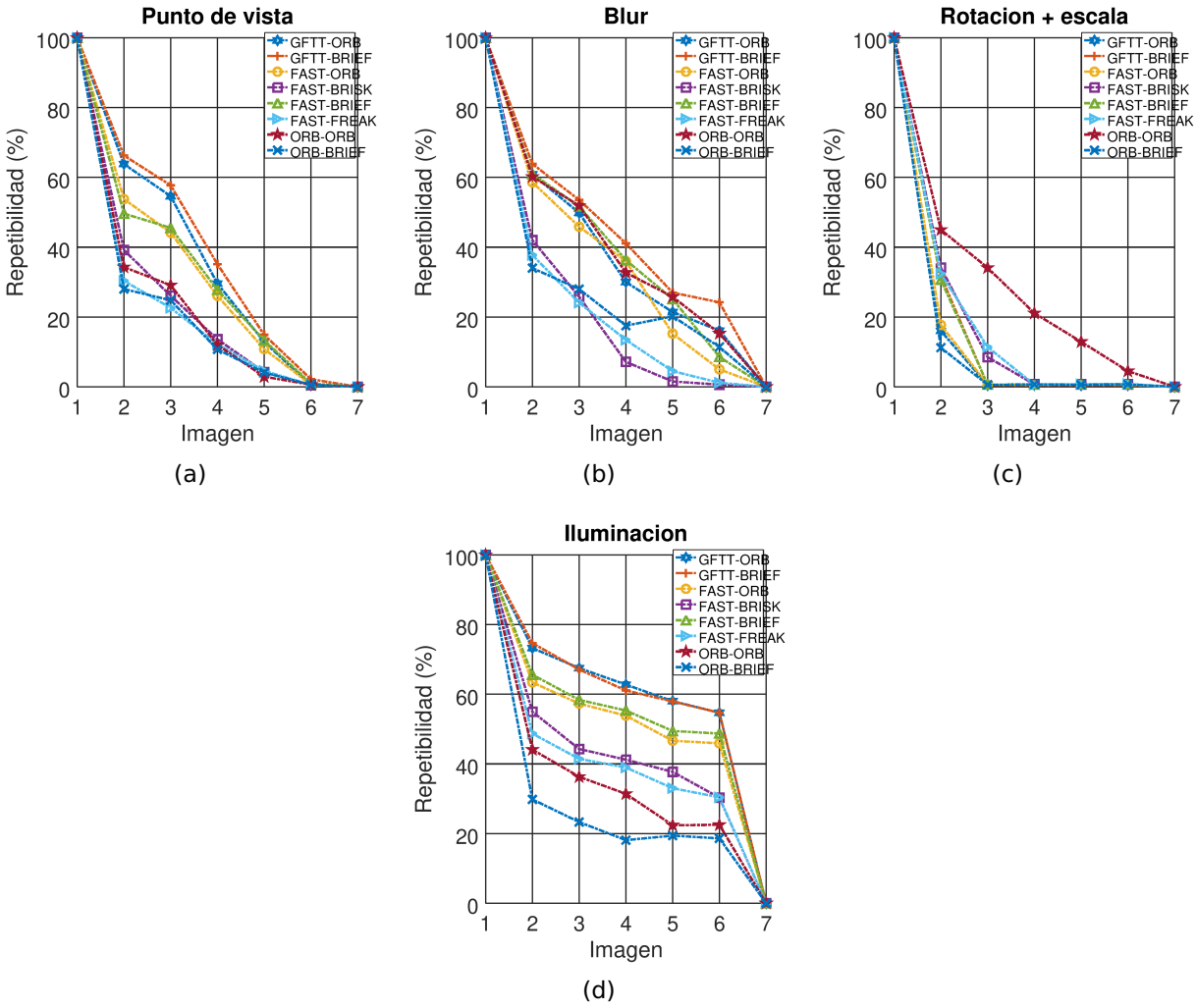


Figura 20. Prueba de repetibilidad.

perfecto. Un descriptor ideal tendría un valor de recall de 100 % sin importar el valor de la precision, es decir, devolvería todas las correspondencias correctas, pero no las incorrectas.

En la Fig. 21 se muestran los resultados, donde se observa que la combinación ORB-BRIEF tiene la AUC más baja en la mayoría de los casos, por lo tanto, se descarta. El resto de combinaciones tiene un rendimiento similar.

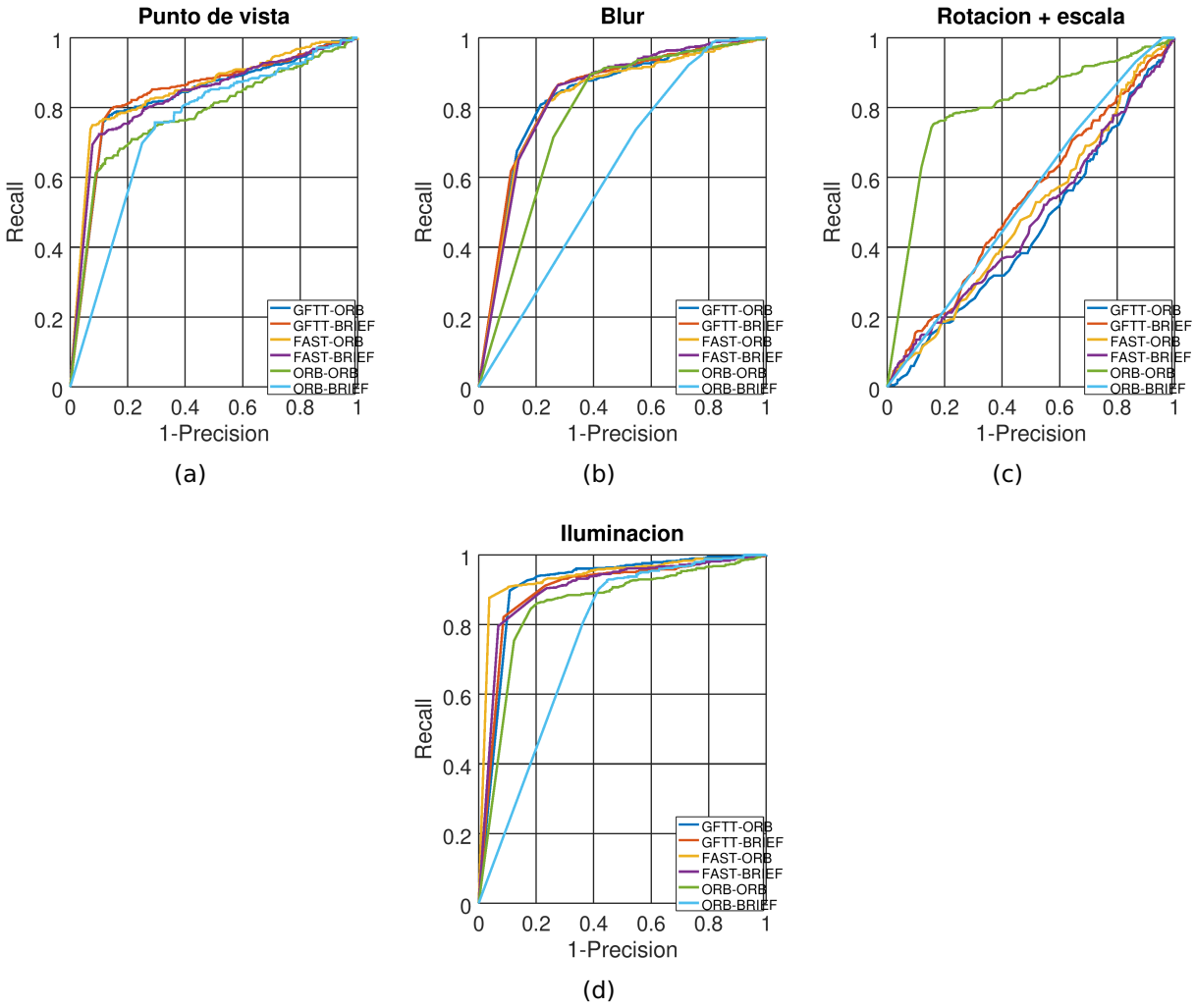


Figura 21. Prueba de Precision y Recall.

4.4. Selección del algoritmo

Los algoritmos que pasaron las pruebas anteriores fueron: GFTT-ORB, GFTT-BRIEF, FAST-ORB, FAST-BRIEF, ORB-ORB. Cualquiera de estas combinaciones son buenos candidatos para lograr un seguimiento de la cámara en tiempo real y arrojar buenos resultados en SLAM visual usando un sensor Kinect. De los experimentos se hacen las siguientes observaciones.

- Se observa que ORB-ORB tiene una buena repetibilidad y AUC bajo condiciones de rotación y cambios de escala debido a que tanto el detector como el descriptor ORB son invariantes a dichas condiciones.

- Las combinaciones GFTT-ORB y GFTT-BRIEF tienen la mejor repetibilidad en la mayoría de los casos y una AUC aceptable, excepto para rotación + escala, esto debido a que el detector GFTT no es invariante a cambios de escala y el descriptor BRIEF no es invariante ni a rotación ni a escala.

- Las combinaciones FAST-ORB y FAST-BRIEF tienen un rendimiento similar que las combinaciones con el detector GFTT con la ventaja de un tiempo de extracción mucho menor.

- Para movimientos pequeños que se encuentra en imágenes sucesivas de cámaras a alta frecuencia, las combinaciones con los detectores FAST y GFTT deben lograr un buen rendimiento en menos tiempo que ORB-ORB. Para cámaras que operan a frecuencias más bajas es recomendable usar ORB-ORB debido a que el movimiento entre imágenes será más grande. Para nuestro sistema se decidió usar FAST-BRIEF debido a que tiene buena repetibilidad, aceptable curva RP y el mejor tiempo de extracción.

Capítulo 5. RGB-D SLAM adaptativo

En este capítulo se describe el sistema SLAM propuesto usando un sensor RGB-D. El sistema desarrollado genera mapas topo-métricos usando algoritmos del estado del arte y se introduce el enfoque adaptativo para mejorar la precisión y robustez. En enfoque adaptativo se usa en el módulo frontend para mejorar el proceso de odometría visual. El sistema fue evaluado usando conjuntos de datos diseñados para evaluar sistemas SLAM con sensores RGB-D.

5.1. Sistema propuesto

En este trabajo se presenta un sistema SLAM indirecto (basado en características) que opera en tiempo real usando un sensor RGB-D, el cual hace uso del enfoque adaptativo para mejorar diversas etapas de odometría visual llamado ASLAM. El sistema fue desarrollado sobre las ideas principales de PTAM (Klein y Murray, 2007), el trabajo de reconocimiento de lugares de Galvez-López y Tardos (2012), el uso de un grafo de covisibilidad (Strasdat *et al.*, 2011), el algoritmo de localización RGB-D adaptativa (Paton y Kosecka, 2012) y la generación de mapas probabilísticos 3D (Hornung *et al.*, 2013).

Siguiendo el enfoque de PTAM, ASLAM divide el problema en dos tareas principales: seguimiento de la cámara (frontend) y optimización del mapa (backend). Estas tareas se ejecutan en dos hilos diferentes, solo comparten el mapa entre ellos. El módulo frontend calcula las correspondencias entre características, crea nuevos puntos y estima la pose de la cámara para cada nuevo frame, y el hilo backend refina iterativamente los puntos de referencia cercanos que componen el mapa, busca cierres de ciclos y actualiza el mapa de vóxeles 3D.

ASLAM fue desarrollado para lograr un sistema SLAM RGB-D flexible, robusto y preciso. Sus características principales se resumen a continuación:

- Uso del mismo extractor de características para todas las tareas: odometría visual, mapeo y cierre de ciclos. Esto hace al sistema más eficiente, simple y confiable. Se hace uso del extractor FAST-BRIEF el cual permite una extracción en

tiempo real sin la necesidad de GPUs y provee una buena invarianza a cambios de vista e iluminación.

- Los puntos de interés visuales son extraídos en cada nivel de una pirámide gausiana usando el detector FAST. El umbral para el detector FAST se elige de forma adaptativa utilizando un control proporcional simple para garantizar que se detecten un número suficiente de características en cada cuadro.
- Estimación de la pose de la cámara usando el algoritmo de localización RGB-D adaptativa para mejorar la precisión.
- Operación en tiempo real en ambientes grandes. Debido al uso de un grafo de covisibilidad, el seguimiento y mapeo se centran en un área local covisible, independiente del tamaño del mapa global.
- Detección y corrección de ciclos en tiempo real. La detección de ciclos se realiza mediante métodos basados en apariencia de imágenes y la corrección mediante la optimización de un grafo de poses.
- Un proceso de mantenimiento que se ejecuta en un hilo independiente refina iterativamente el mapa (BA) en un área covisible local, mejorando la consistencia global.

En la Fig. 22 se muestra un esquema de los componentes principales y el flujo de datos de ASLAM.

El sistema incorpora dos módulos principales que se ejecutan en paralelo: frontend y backend. El módulo frontend se encarga de la localización de la cámara con cada frame y decide cuando insertar un nuevo keyframe. Primero se buscan correspondencias con el frame anterior y se refina la pose usando el algoritmo de localización RGB-D adaptativa. Una vez que hay una estimación de la pose de la cámara y las correspondencias, se recupera un mapa visible local utilizando el grafo de covisibilidad de los keyframe que mantiene el sistema. El mapa es un recurso compartido entre los módulos backend y frontend.

El módulo backend constantemente minimiza el error de reproyección para refinar todos los puntos en el mapa y las poses de la cámara aplicando BA en el mapa local para lograr una reconstrucción óptima en la vecindad de la pose de la cámara. Se usa un

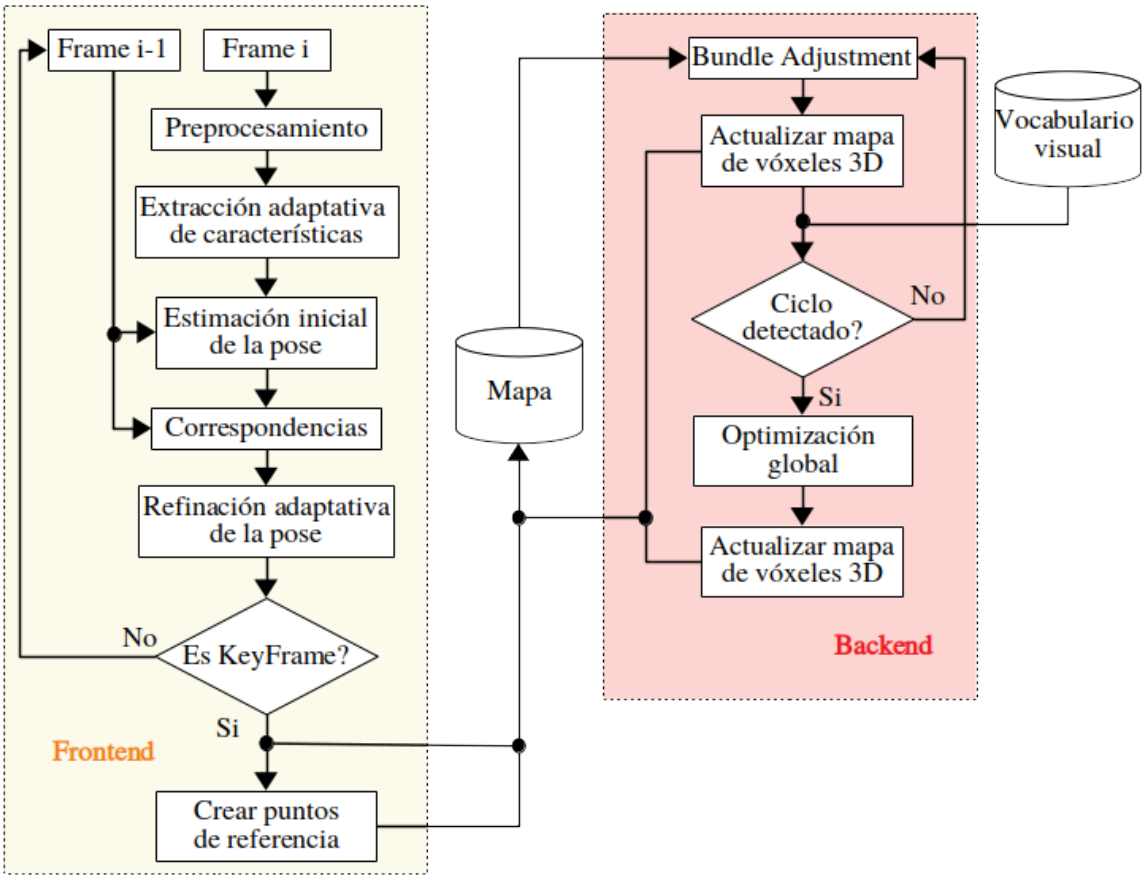


Figura 22. Esquema general del sistema ASLAM que muestra todos los componentes y el flujo de datos.

grafo de poses para mantener una consistencia global del mapa. Las correspondencias entre puntos se buscan activamente entre los keyframes para fortalecer las restricciones del grafo de poses. Basado en la información adquirida por el módulo frontend se aplica una política de selección de puntos exigente para conservar solo puntos de alta calidad. También se encarga de eliminar keyframes redundantes. Para lidiar con grandes errores en trayectorias largas se detectan ciclos. Si un ciclo es detectado, se calcula la transformación de cuerpo rígido que informa sobre el error acumulado en el ciclo. Ambas partes del ciclo se alinean y se fusionan puntos duplicados. Finalmente se optimiza el grafo de poses para logar una consistencia global. Se usa el algoritmo Levenberg-Marquardt para llevar a cabo todas las optimizaciones.

A continuación se describen a detalle cada módulo y componentes del sistema.

5.2. Mapa

El mapa está compuesto por un grafo de covisibilidad que almacena toda la información sobre los keyframes y puntos de referencia procesados por el sistema, donde las restricciones pose-pose están definidas por covisibilidad (Hornung *et al.*, 2013), es decir, dos poses están conectadas si comparten suficientes puntos en común. Además almacena una representación 3D basada en vóxeles.

5.2.1. Grafo de covisibilidad

El grafo de covisibilidad consiste del conjunto de vértices keyframes $\mathcal{V}$, un conjunto de puntos 3D $\mathcal{P}$ y un conjunto de aristas relativas $\mathcal{E}$. Cada keyframe V_i almacena su pose absoluta $\mathbf{T}_i$, los puntos $\mathbf{x}_k \in \mathcal{P}$ visibles desde su posición y todas las observaciones correspondientes $\mathbf{z}_{ik}$. Una arista E_{ij} entre dos vértices V_i y V_j tiene un peso de covisibilidad θ_{ij}, el cual es el número de puntos que son visibles en ambos vértices. La arista puede ser marcada como marginada o no, si es marginada almacena la restricción de pose relativa $\mathbf{T}_{ij}$ entre $\mathbf{T}_i$ y $\mathbf{T}_j$, de otro modo, la pose relativa se define implícitamente como $\mathbf{T}_{ij} = \mathbf{T}_i \cdot \mathbf{T}_j^{-1}$. En todo momento hay exactamente un keyframe de referencia V_{ref}. En la Fig. 23 se muestran ejemplos del grafo.

Para corregir un ciclo se realiza una optimización del grafo de poses que distribuye el error a lo largo del grafo. Dado que el grafo de covisibilidad puede ser muy denso, el grafo de poses solo contiene aquellas aristas de poses consecutivas y de cierres de ciclo.

5.2.2. Mapa probabilístico denso

Para representar entornos tridimensionales se hace uso de una representación basada en octrees para obtener la máxima flexibilidad con respecto al área mapeada y a la resolución. Se realiza una estimación probabilística de la ocupación para garantizar la capacidad de actualización y hacer frente al ruido del sensor.

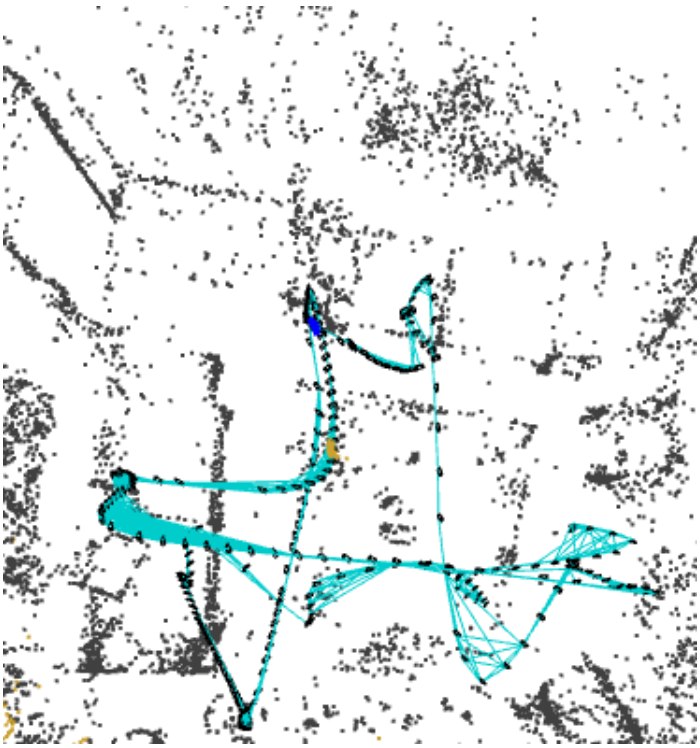


Figura 23. Ejemplo del grafo de covisibilidad. Las lineas azules representan las aristas entre las poses de la cámara. El conjunto de puntos $\mathcal{P}$ se muestra en color negro.

Un octree es una estructura de datos jerárquica usado para la subdivisión espacial 3D. Cada nodo en un octree representa el espacio contenido en un volumen cúbico, usualmente llamado vóxel. Este volumen se subdivide recursivamente en ocho subvolúmenes hasta que se alcanza un tamaño mínimo de vóxel, como se muestra en la Fig. 24. El tamaño mínimo de vóxel determina la resolución del octree. En su forma más básica, los octrees modelan propiedades booleanas. En el contexto de mapeo robótico, esto suele ser la ocupación de un volumen. Si cierto volumen se mide como ocupado, se inicializa el nodo correspondiente en el octree. Cualquier nodo no inicializado podría ser libre o desconocido en esta configuración booleana. Para resolver esta ambiguedad se representan explícitamente los volúmenes libres en el árbol. Las áreas que no se inicializan modelan implícitamente el espacio desconocido. El uso de estados de ocupación booleanos o etiquetas discretas permite representaciones compactas del octree: si todos los hijos de un nodo tienen el mismo estado (ocupado o libre), se pueden eliminar. Esto conduce a una reducción en el número de nodos que deben mantenerse en el árbol.

Debido a que se tiene que lidiar con el ruido del sensor y entornos que cambian de manera temporal o permanente, una etiqueta de ocupación discreta no es suficiente,

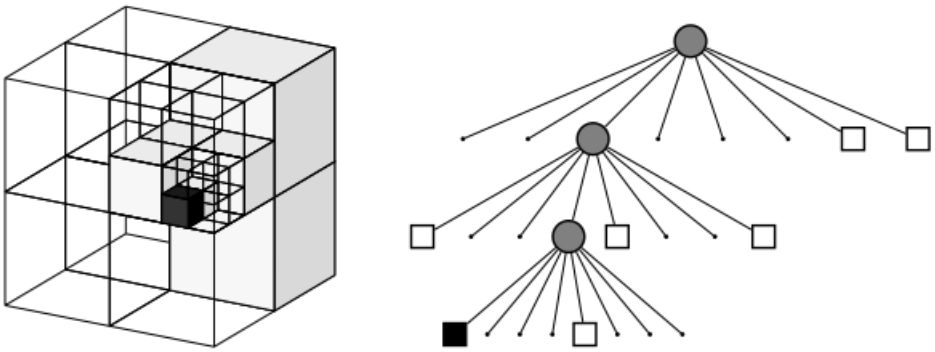


Figura 24. Ejemplo de un octree con celdas libres (blanco sombreado) y ocupadas (negro). La representación volumétrica se muestra en el lado izquierdo y su correspondiente árbol a la derecha

en cambio, la ocupación debe modelarse probabilísticamente usando un mapeo de cuadrícula de ocupación. La probabilidad $P(n \mid z_{1:t})$ de un nodo n para ser ocupado dadas las mediciones del sensor $z_{1:t}$ se estima de acuerdo con

$$P(n \mid z_{1:t}) = \left[1 + \frac{1 - P(n \mid z_t)}{P(n \mid z_t)} \frac{1 - P(n \mid z_{1:t-1})}{P(n \mid z_{1:t-1})} \frac{P(n)}{1 - P(n)}\right]^{-1}. \tag{132}$$

Esta fórmula de actualización depende de la medición actual z_t, la distribución a priori $P(n)$ y la estimación anterior $P(n \mid z_{1:t-1})$. El término $P(n \mid z_t)$ denota la probabilidad de que se ocupe el vóxel n dada la medición z_t. Este valor es específico del sensor que generó z_t.

Una suposición común es una distribución uniforme a priori $P(n) = 0.5$, la Ec. 132 se puede reescribir usando notación logarítmica

$$L(n \mid z_{1:t}) = L(n \mid z_{1:t-1}) + L(n \mid z_t), \tag{133}$$

donde

$$L(n) = \log\left[\frac{P(n)}{1 - P(n)}\right]. \tag{134}$$

Esto permite actualizaciones más rápidas ya que las multiplicaciones se remplazan por adiciones.

Un vóxel se considera ocupado si la probabilidad $P(n \mid z_{1:t})$ es mayor a un umbral y libre en otro caso.

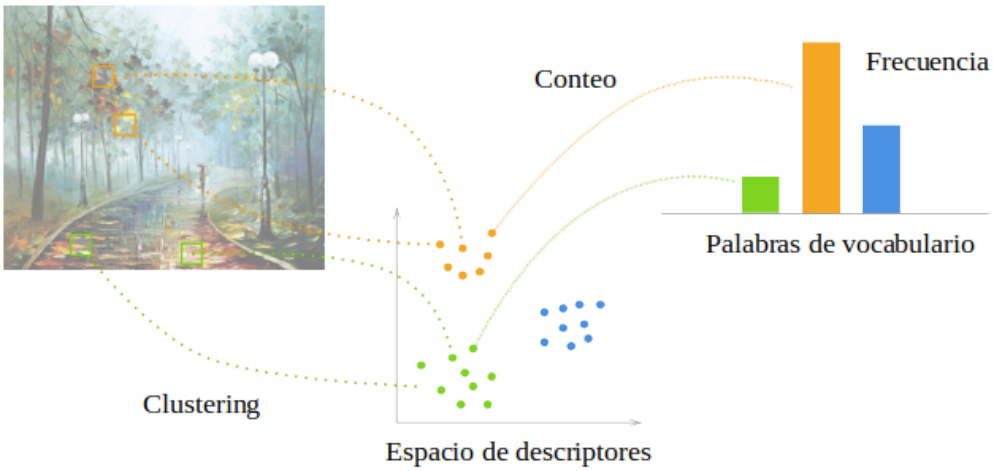


Figura 25. Proceso de creación del vocabulario visual. Los descriptores en la imagen son agrupados mediante el algoritmo *k*-medias. Cada grupo es cuatificado mediante un histograma de frecuencias.

5.3. Vocabulario visual

El sistema tiene embebido un módulo de reconocimiento de lugares basado en los árboles de vocabulario (Nister y Stewenius, 2006) para detectar ciclos, los cuales tiene un buen rendimiento para insertar y recuperar imágenes basadas en descriptores. Este paso es llamado prefiltro, ya que, solo se obtienen candidatos que podrían coincidir con la imagen actual, que luego debe ser sujeta a la verificación de consistencia geométrica para su confirmación y estimación de la pose. El prefiltro es una técnica de bolsa de palabras jerárquico que funciona con vistas monoculares (la imagen RGB del sensor RGB-D).

Las palabras visuales son solamente una discretización del espacio de descriptores, el cual es conocido como vocabulario visual (Fig. 25). El árbol de vocabulario visual es una estructura jerárquica que define simultáneamente las palabras visuales y un procedimiento de búsqueda para encontrar palabras más cercanas a cualquier características visual dada. El árbol se construye fuera de linea mediante una agrupación jerárquica de *k*-medias sobre un gran conjunto de descriptores de características visuales de entrenamiento. El conjunto de descriptores de entrenamiento se agrupa en *k* centros. Cada centro se convierte en una nueva rama del árbol, y el subconjunto de descriptores más cercanos a él se agrupa nuevamente. El proceso se repite hasta alcanzar el número deseado de niveles.

La capacidad discriminativa del vocabulario aumenta con el número de palabras, a un costo de mayor error de cuantificación y mayores requisitos de memoria (Konolige *et al.*, 2010). Nister y Stewenius (2006) demostraron que el rendimiento mejora con la cantidad de palabras, hasta vocabularios muy grandes (> 1 millón). En este trabajo se usaron alrededor de un 1 millón de características visuales de 500 imágenes seleccionadas de los conjuntos de datos usados (Sturm *et al.*, 2012; Handa *et al.*, 2014), con $k = 10$, árbol con una profundidad de 5, resultando en 10000 palabras visuales.

El vocabulario es llenado con las imágenes de referencia colocando cada uno de sus descriptores en una hoja y registrando la imagen en una lista o archivo invertido en la hoja. Para consultar el árbol, los descriptores de la imagen de consulta se colocan de manera similar en los nodos hoja y se recuperan imágenes de referencia potencialmente similares de la unión de los archivos invertidos. El vocabulario describe la imagen como un vector de frecuencia de palabras determinadas por las rutas tomadas por los descriptores a través del árbol. Cada imagen de referencia se puntúa por relevancia para la imagen de consulta calculando la distancia L1 entre sus vectores de frecuencia. La puntuación está ponderada por la entropía para descartar palabras muy comunes usando el enfoque de Term Frequency Inverse Document Frequency (TF-IDF) descrito en Nister y Stewenius (2006); Sivic y Zisserman (2003)

5.4. Frontend

5.4.1. Preprocesamiento de la imagen

La imagen RGB-D adquirida por la cámara RGB-D es preprocesada para mejorar el proceso de extracción de características. El componente RGB de la imagen es convertido a escala de grises y suavizado con un kernel gausiano de $\sigma = 0.85$. Se construye una pirámide gausiana con 5 niveles de escala para permitir una extracción de características más robusta a diferentes escalas con un factor de 1.5. Cada nivel de la pirámide corresponde a una octava en el espacio de escalas. Características a escalas más altas generalmente corresponden a estructuras de imagen más grandes en la escena, lo cual las hace más repetibles y resistentes al desenfoque de movimiento. En la Fig. 26 se muestra el proceso de preprocesamiento de la imagen RGB.

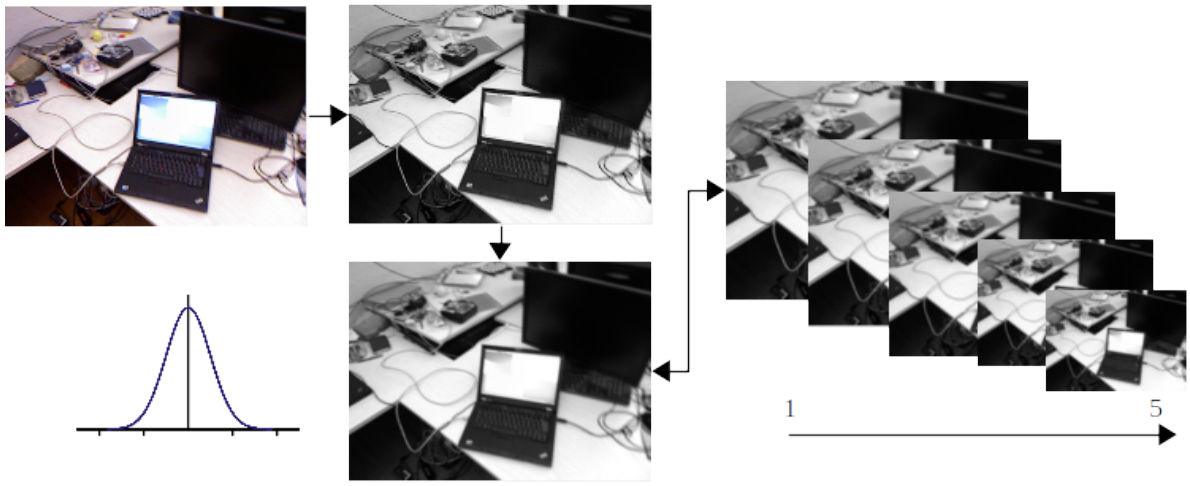


Figura 26. Proceso de preprocesamiento de la imagen RGB.

5.4.2. Extracción adaptativa de características

Características visuales son extraídas en cada nivel de la pirámide gausiana usando el detector FAST. El umbral para el detector FAST se elige de manera adaptativa usando un control proporcional simple (Fig. 27) para garantizar que sean detectadas un número suficiente de características en cada imagen. El umbral del detector de esquinas FAST se establece inicialmente a un valor que proporciona un compromiso entre el número de puntos y robustez al ruido. Este umbral se reduce o aumenta a cada paso de tiempo para garantizar un número mínimo de puntos. Esto resulto suficiente para adaptarse a los cambios de iluminación presentes en las secuencias evaluadas. La profundidad correspondiente de cada característica es extraída de la imagen de profundidad. Características que no tienen una profundidad asociada son descartadas. Cada nivel de la pirámide es discretizada en una rejilla para mantener una distribución uniforme de las características, en cada celda se mantienen las 20 características con mayor puntaje de esquina FAST. Por último se calculan los descriptores de las características extraídas mediante el algoritmo BRIEF.

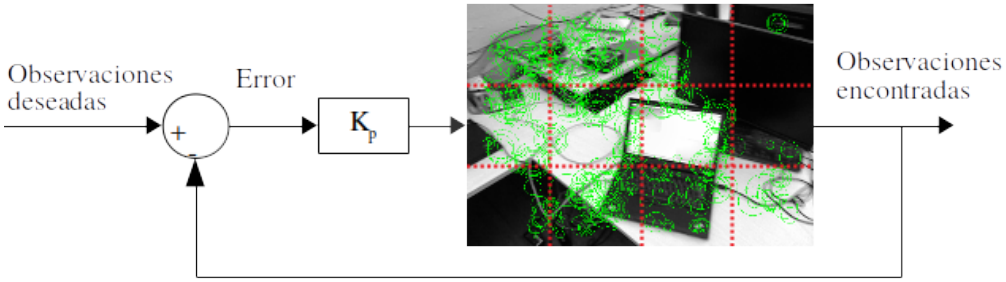


Figura 27. Control proporcional para extraer características visuales.

5.4.3. Estimación inicial de la pose

Para movimientos pequeños como aquellos encontrados en imágenes sucesivas, la mayor parte del movimiento aparente de una característica en el plano de la imagen es causado por la rotación 3D (Mei *et al.*, 2009). Estimar esta rotación permite restringir la ventana de búsqueda de correspondencias entre imágenes. Se usó un modelo de velocidad constante, es decir, se asume que el movimiento entre las imágenes i y $i+1$ es similar al de las imágenes $i-1$ y i: También se podría usar una Unidad de Medición Inercial (IMU), un modelo dinámico del robot ó técnicas basadas en imágenes como el propuesto por Mei *et al.* (2008) para calcular la rotación inicial minimizando directamente la suma de los errores cuadrados de los píxeles entre versiones reducidas de la imagen actual y anterior.

5.4.4. Asociación de datos

La asociación de datos es un problema fundamental en los sistemas de SLAM visual, ya que, una incorrecta asociación de datos puede resultar en consecuencias catastróficas para todo el sistema.

Para reducir la ventana de búsqueda de correspondencias se hace uso del modelo de velocidad constante explicado anteriormente para predecir la pose actual de la cámara y realizar una búsqueda guiada de los puntos observados en la última imagen.

A cada característica visual se le asigna un vector descriptor mediante el algorit-

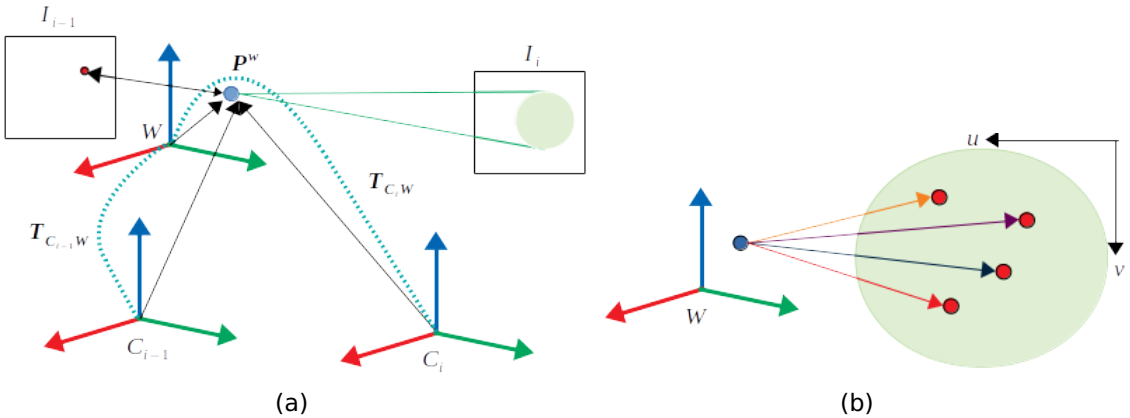


Figura 28. Modelo de velocidad. El punto $\mathbf{P}^w$ en el marco de coordenadas global fue observado en la imagen I_{i-1} y muy probablemente también aparezca en la imagen consecutiva I_i. La pose de la cámara C_i se predice usando el modelo de velocidad constante, gracias a esto es posible expresar el punto $\mathbf{P}^w$ en el sistema de la cámara $\mathbf{P}^{C_i}$ y mediante el modelo de la cámara pinhole proyectar el punto 3D a la imagen 2D I_i. Dado que la posición del punto en la imagen es una aproximación, es necesario definir un radio de búsqueda (circulo color verde), calcular su distancia de Hamming con cada punto dentro de la ventana y emparejarlo con aquel que tenga la menor distancia. Si la distancia es menor a un umbral se acepta como verdadera correspondencia, de otro modo es descartada.

mo BRIEF. Se buscan correspondencias entre imágenes comparando la distancia de sus descriptores usando un algoritmo de fuerza bruta, es decir, se calcula la distancia de cada descriptor en la imagen anterior I_{i-1} con todos los descriptores de la imagen actual I_i que se encuentren dentro de la ventana de búsqueda impuesta por el modelo de velocidad constante y se marca como correspondencia aquel par que tenga la distancia menor y que sea menor a un umbral, el proceso se repite para todos las características de la imagen base (Fig. 28). Dado que BRIEF es un descriptor binario, se utiliza la distancia de Hamming como métrica de distancia.

Si no se encuentran suficientes correspondencias (el modelo de velocidad es claramente violado), se compara cada descriptor de la imagen base con todos los descriptores de la imagen de referencia y se marca como correspondencia aquel par que tenga la distancia menor, el proceso se repite para todas las características de la imagen base. Sin embargo, por sí misma, la distancia no es un criterio suficientemente fuerte de asociación, ya que, la distancia entre los descriptores correspondientes puede variar mucho. Debido a la alta dimensionalidad del espacio de características, generalmente no es posible establecer un umbral de rechazo. Como lo propuso (Lowe, 2004), se calcula la relación entre la distancia del vecino más cercano y la distancia al segundo vecino más cercano en el espacio de características. Bajo el supuesto de que una característica solo corresponde exactamente a otra característica en otra imagen,

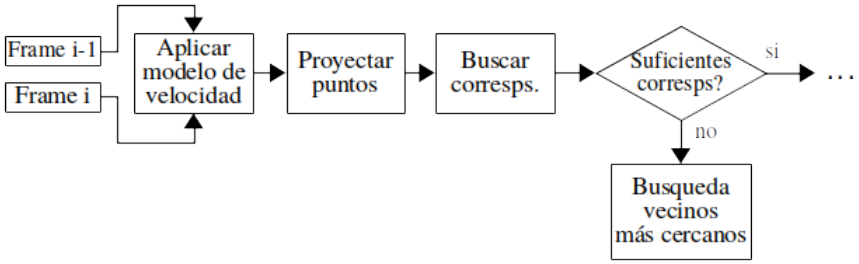


Figura 29. Diagrama de bloques para el proceso de asociación de datos.

la distancia del segundo vecino más cercano debería ser mucho mayor. Por lo tanto, un umbral en la relación entre las distancias al vecino más cercano se puede usar de manera efectiva para controlar la relación entre falsos negativos y falsos positivos. En la Fig. 29 se muestra el diagrama de bloques del algoritmo de asociación de datos.

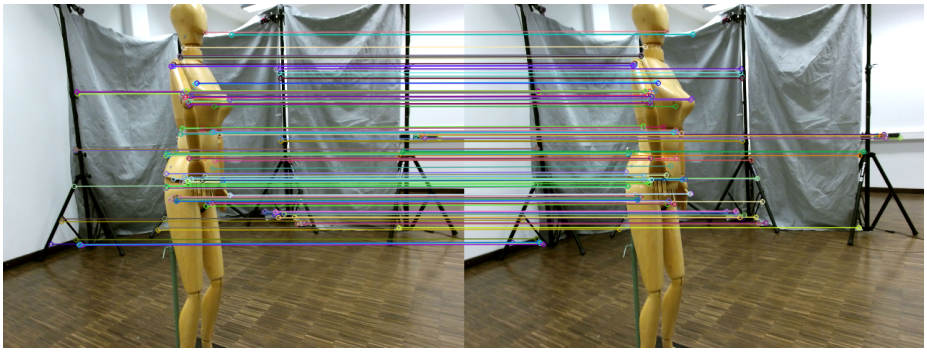
5.4.5. Detección de inliers

A pesar que la restricción impuesta por la estimación inicial de la pose reduce considerablemente la tasa de falsas correspondencias entre imágenes, un paso adicional es necesario para eliminar las falsas correspondencias restantes. Se utiliza RANSAC para encontrar la mejor transformación de cuerpo rigido $\mathbf{T}^*$ entre el conjunto de correspondencias. La métrica de error es usada para encontrar la alineación optima depende de cómo la cámara RGB-D determina los valores de profundidad. Para cámaras estéreo activas RGB-D como las utilizados en el Kinect, la medición del error de reproyección es la métrica más apropiada (Henry *et al.*, 2012)

$$\mathbf{T}^* = \underset{\mathbf{T}}{\operatorname{argmin}} \left(\frac{1}{A_f} \sum_{i \in A_f} |\pi_s(\mathbf{T}(f_s^i)) - \pi_s(f_t^i)|^2 \right), \tag{135}$$

donde, A_f contiene las correspondencias entre ambas imágenes, f_s^i es el punto en la imagen fuente y f_t^i su correspondiente punto en la imagen destino. Esta métrica mide el error en el espacio de píxeles al proyectar los puntos asociados en el marco de la cámara. La función de proyección $\pi_s : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ proporciona la proyección del punto $f = (x, y, z) \in \mathbb{R}^3$ desde su posición Euclidiana 3D con respecto a la cámara en el espacio de la imagen, resultando $(u, v, d) \in \mathbb{R}^3$, donde u y v son las coordenadas del píxel RGB y d es la disparidad, la cual representa el valor de profundidad en el espacio

(a)



(b)

Figura 30. (a) Estimación de correspondencias. (b) Resultado de aplicar el algoritmo RANSAC para descartar outliers.

de píxeles. Al igual que una cámara estéreo pasiva estándar, la profundidad de un píxel se mide mediante triangulación. En el caso del Kinect, esto se hace utilizando la línea base B entre el proyector infrarrojo y la cámara infrarroja. Una suposición implícita en la Ec. 135 es que los errores en u, v y d son independientes e igualmente ponderados (Konolige, 2010).

Para optimizar conjuntamente la asociación de datos A_f y la transformación $\mathbf{T}^*$ se aplica un enfoque RANSAC. Primero se encuentran características coincidentes entre dos imágenes. Luego se toman de manera aleatoria tres correspondencias y se determina la transformación óptima para esta muestra. Las transformaciones propuestas se evalúan de acuerdo con el número de inliers entre los puntos 3D restantes. Cuando se optimiza el error de reproyección (Ec. 135), los inliers son determinados mediante un umbral de distancias de píxeles. Se utiliza la muestra que da como resultado el mayor número de inliers para calcular eficientemente una transformación más precisa teniendo en cuenta todos los inliers. En la Fig. 30 se muestra el resultado de aplicar este enfoque para detectar inliers.

5.4.6. Localización RGB-D adaptativa

La estimación final del movimiento es calculada usando RANSAC+ICP bajo un enfoque adaptativo. El esquema general de mapeo RGB-D consiste en a) detección de características y correspondencias b) estimación inicial del movimiento de un cuerpo rígido con RANSAC y c) refinamiento de la estimación de movimiento con un algoritmo ICP. Hay dos casos en los que el paso b) puede fallar: la falta de características visuales en el entorno o la falta de características comunes entre dos imágenes. El primer caso se puede ver en ambientes interiores comunes como pasillos de oficinas. El último puede atribuirse a movimientos rápidos de la cámara, pequeña superposición entre vistas consecutivas o la posible pérdida de datos debido a problemas de comunicación y subprocesos. En todos estos escenarios, la estimación de movimiento derivada de RANSAC puede no ser representativa del movimiento verdadero e inicializar ICP con este movimiento puede hacer que ICP no converga. En este trabajo se propone un método para superar estas debilidades mediante el uso del algoritmo ICP generalizado (Segal *et al.*, 2009; Paton y Kosecka, 2012) no solo como una herramienta de refinamiento de movimiento sino también como un método alternativo si RANSAC proporciona un resultado insatisfactorio. El flujo general del algoritmo se detalla en la Fig. 31.

Hay tres posibles rutas que puede tomar el algoritmo. Las primeras dos empiezan con la extracción de características y estimación de correspondencias, la estimación de movimiento y el rechazo de outliers utilizando correspondencias visuales y el algoritmo RANSAC. Después de este paso se analiza el rendimiento de RANSAC en términos del error devuelto por la estimación, así como la cantidad de correspondencias visuales que se consideraron como inliers. Si el resultado es satisfactorio, se acepta la estimación de movimiento. En el caso de que el error de la estimación sea alto, se utiliza GICP para refinar la estimación. En el tercer caso, el movimiento se estima únicamente utilizando GICP en un subconjunto aleatorio de los datos de profundidad RGB-D. Esto se activa cuando ocurre cualquiera de los siguientes casos:

1. No hay suficientes correspondencias visuales para calcular de manera confiable una estimación de movimiento con RANSAC.

2. La estimación de movimiento generada por RANSAC tiene un error residual alto

que no es adecuado para la inicialización de GICP.

3. RANSAC excedió el número máximo de iteraciones indicando una alta proporción de outliers.

En cualquiera de estos casos, la estimación del movimiento generada por RANSAC dificultaría la convergencia de GICP a la solución correcta. En este caso GICP estima el movimiento sin la inicialización de RANSAC, usando como inicialización el modelo de velocidad constante. Esta ruta también descarta las correspondencias visuales de la etapa de detección de características y se usa un subconjunto aleatorio de puntos en la imagen y sus respectivas coordenadas 3D en imágenes consecutivas. GICP no requiere correspondencias exactas para converger a una estimación de movimiento correcta. Esta inicialización permite el registro correcto, incluso cuando hay una falta de superposición completa entre las nubes de puntos.

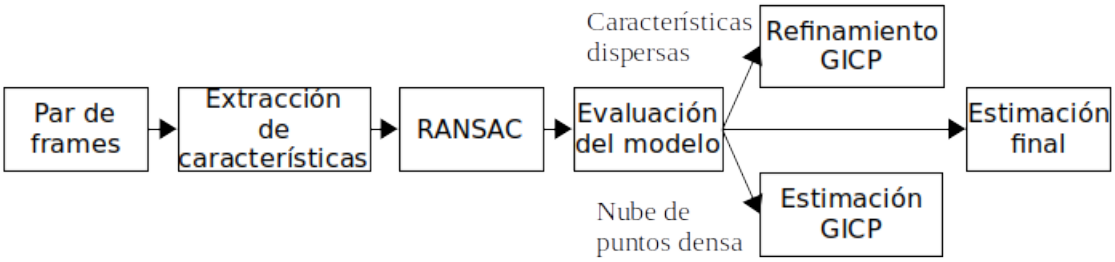


Figura 31. Esquema del algoritmo de estimación de movimiento adaptativa.

Los umbrales τ_1, τ_2 y μ son los parámetros del algoritmo. Si el error residual promedio de la estimación de movimiento generada con RANSAC es menor o igual que τ_1 y el número de inliers es mayor que μ, entonces la estimación de RANSAC se considera correcta y se usa como la respuesta final, de otra manera se usa GICP. Si el error de RANSAC es mayor que τ_2, entonces la estimación de movimiento que arroja RANSAC se considera inestable y GICP comienza desde cero usando un subconjunto aleatorio del conjunto de datos 3D completo. En este caso GICP registra la nueva nube de puntos usando el modelo de velocidad como inicialización si no fue violada, en caso de violación del modelo de velocidad se inicializa con la identidad. Si el error es menor que τ_2 pero mayor que τ_1, GICP usa la estimación de movimiento y los inliers obtenidos con RANSAC para mejorar la transformación **T**. Los valores usados son $\tau_1 = 5$, $\tau_2 = 20$ y $\mu = 20$ (Paton y Kosecka, 2012).

5.4.7. Técnica de keyframes

Para mantener el grafo relativamente disperso se hace uso de keyframes, los cuales son un subconjunto de los frames consecutivos procesados. Estos pueden ser seleccionados de diferentes maneras. El método más simple es seleccionar cada nth frame. Otra opción es determinar los keyframes en función de la superposición visual. En este trabajo, se añade un nuevo keyframe V_i al mapa cuando el movimiento acumulado desde el keyframe anterior excede 10° en rotación o 20 cm en traslación. Fara todos los keyframes en el grafo $V_j \in \mathcal{V}$ que comparten al menos φ puntos con el keyframe actual, se añade una nueva arista E_{ij}, se marca como no marginada y se le asigna un peso de covisibilidad θ_{ij}. Por último, el nuevo keyframe V_i se elige como el nuevo keyframe de referencia $V_{ref} = V_i$.

Para reducir el error se hace uso de la técnica de keyframes. El movimiento es estimado comparando el frame más reciente con un frame de referencia. Si el movimiento de la cámara relativo al frame de referencia se calcula de manera satisfactoria con un número suficiente de inliers, entonces el frame de referencia no cambia. De lo contrario, el frame más reciente remplaza el frame de referencia después de que finalice la estimación. Si la estimación de movimiento con el frame de referencia falla, entonces la estimación de movimiento se intenta nuevamente con el segundo frame más reciente. Esta simple heurística sirve para reducir el error en situaciones donde el punto de vista de la cámara no varía significativamente.

5.5. Backend

5.5.1. Bundle adjustment

Debido a los errores de estimación de movimiento presentes en el módulo frontend, las aristas no forman una trayectoria consistente, por lo tanto, es necesario refinar las poses. Se aplica BA a una ventana local de keyframes $\mathcal{V}_L$ formada por el keyframe procesado actualmente V_i y todos los keyframes conectados a el en el grafo de covisibilidad $\mathcal{V}_C$, y todos los puntos $\mathcal{P}_L$ vistos por esos keyframes. Todos los demás keyframes $\mathcal{V}_F$ que ven esos puntos pero no están conectados al keyframe actual son incluidos en

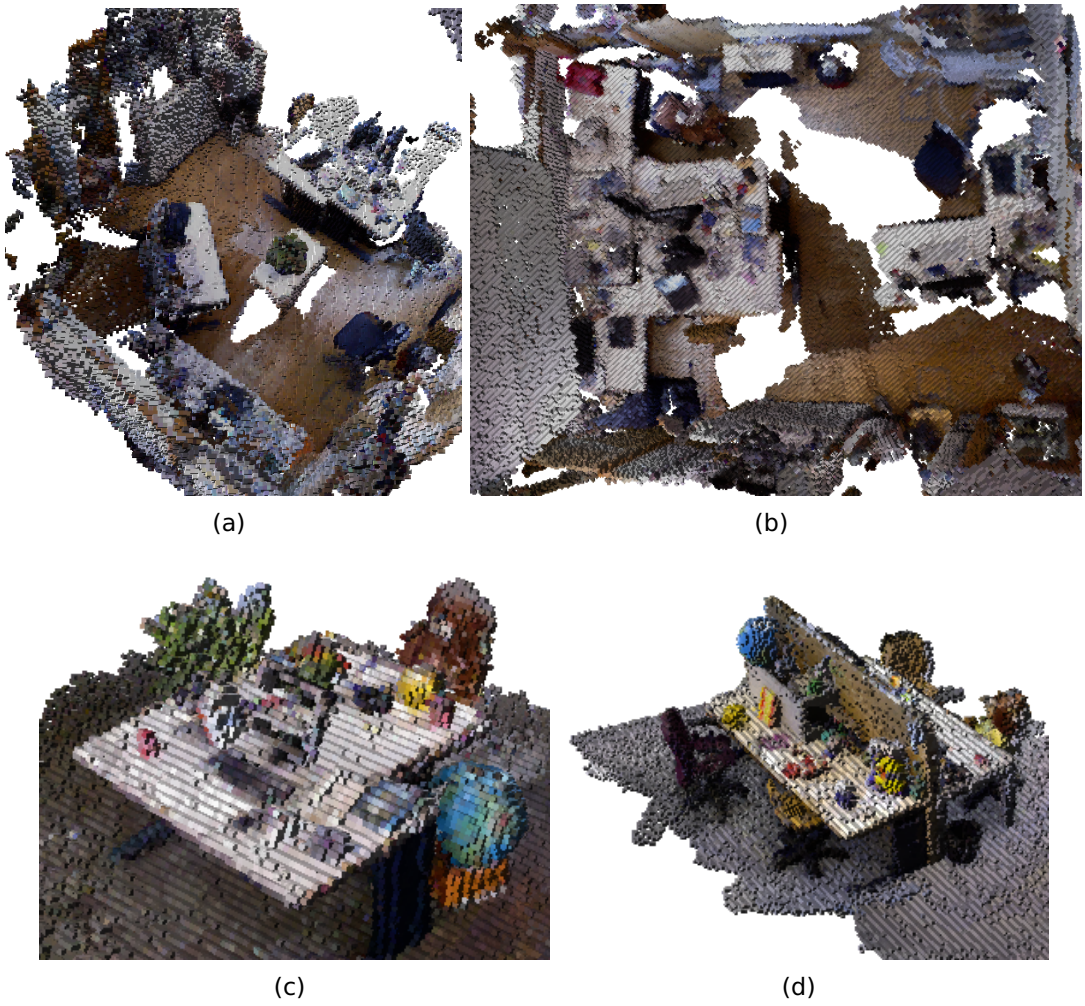
la optimización pero permanecen fijos. La optimización local de BA es un caso especial del problema genéral de BA

$$\{\mathbf{x}_p, \mathbf{T}_l | p \in \mathcal{P}_L, l \in \mathcal{V}_L\} = \underset{\mathbf{x}_p, \mathbf{T}_l}{\operatorname{argmin}} \sum_{v \in \mathcal{V}_L \cup \mathcal{V}_F} \sum_{j \in \mathcal{P}_v} \rho\left(\| \mathbf{y}_v^j - \pi_s(\mathbf{T}_v, \mathbf{x}_j) \|^2_{\Sigma_v^j} \right), \tag{136}$$

donde Σ_v^j es la covarianza de la posición de la jth característica visual correspondiente al punto 3D $\mathbf{x}_j$ en el keyframe V_v, la cual depende de la escala a la que fue detectada. $\mathcal{P}_v$ es el conjunto de puntos observados en el keyframe v. Las observaciones marcadas como outliers por la función de costo de Huber ρ son descartadas por el algoritmo de optimización.

Pueden existir keyframes en los cuales no fue posible establecer correspondencias visuales dado que RANSAC falló por algún motivo explicado anteriormente y la estimación de movimiento fue hecha sobre un subconjunto aleatorio de la nube de puntos completa usando solo GICP. Para estos keyframes, las correspondencias visuales se estiman muestreando puntos de la nube de puntos densa del keyframe a través de una cuadrícula en el espacio de la imagen y encontrando el punto más cercano en el grafo de covisibilidad dentro de un círculo de radio r a partir de la pose obtenida con GICP. Si la distancia al punto más cercano está por debajo de un umbral y las normales correspondientes se desvían en menos de un umbral angular, entonces se marca el par de puntos como correspondencia. Estos pares se consideran equivalentes a las coincidencias de puntos característicos y se incluyen como puntos adicionales dentro de sistema de BA. Sin esta modificación, los keyframes que contienen solo restricciones de GICP podrían dar como resultado un sistema de BA desconectado y sin solución.

Además de ser una valiosa herramienta de optimización, BA también ayuda a refinar los inliers estimados con el algoritmo de localización RGB-D adaptativa. Se usó la implementación de BA proporcionada por el paquete de optimización g2o (Kuemmerle *et al.*, 2011).



(a) (b)

(c) (d)

Figura 32. Ejemplos de OctoMaps creados por ASLAM.

5.5.2. Actualización de mapa de voxels

Usando la trayectoria generada por el sistema, es posible proyectar las observaciones de puntos en un sistema de coordenadas global, creando así una representación de nube de puntos del entorno. Estos modelos, sin embargo, son altamente redundantes y requieren demasiados recursos computacionales, es por esto que las nubes de puntos son comúnmente reducidas, por ejemplo, usando una rejilla de vóxeles.

Para superar las limitaciones de las representaciones de nubes de puntos, se utilizó mapas de cuadrícula de ocupación 3D, haciendo uso de la librería de mapeo volumétrico basado en octrees OctoMap (Hornung *et al.*, 2013). Los voxéles se gestionan en una estructura de árbol muy eficiente que conduce a una representación de memoria

compacta y permite inherentemente acceder a los valores de ocupación almacenados en múltiples resoluciones. El uso de estimación de ocupación probabilística proporciona un medio para hacer frente a mediciones ruidosas y errores en la estimación de la pose. Una ventaja crucial en contraste con una representación basada en puntos, es la representación explícita del espacio libre y las áreas no mapeadas, lo cual es esencial para evitar las colisiones y para realizar tareas de exploración.

De acuerdo con Endres *et al.* (2014), cada keyframe añadido a un mapa de nubes de puntos requiere aproximadamente 3.6 Mb en memoria. Un mapa sin filtro a partir de una escena del tamaño de una oficina requeriría entre 2 y 5 Gb. En contraste, los OctoMaps correspondientes con una resolución de 2 cm van desde solo 4.2 a 25 Mb. Por otro lado, la creación de un OctoMap requiere más recursos computacionales, ya que, cada medición de profundidad se proyecta en el mapa. El tiempo requerido para procesar cada keyframe depende en gran medida del tamaño del vóxel. Por lo tanto, para general un mapa en tiempo real es necesario reducir la resolución y la cantidad de puntos. Se utilizó una resolución de 5 cm y un submuestreo de 12 para poder lograr actualizaciones del mapa a 30 Hz. Sin embargo, un mapa de vóxel no puede actualizarse de manera eficiente en caso de correcciones importantes obtenidas por los cierres de ciclo. Dependiendo de la aplicación, puede ser razonable retrasar la voxelización, recrear el mapa en caso de un cierre de ciclo o crear submapas locales.

En la Fig. 32 se muestran ejemplos de OctoMaps creados por el sistema.

5.5.3. Detección de ciclos y optimización global

La alineación entre frames sucesivos es un buen método para seguir la posición de la cámara a distancias moderadas. Sin embargo, los errores en la alineación y el ruido del sensor hacen que la estimación de la posición de la cámara se desplace con el tiempo, lo que lleva a imprecisiones en el mapa. Esto es más notable cuando la cámara sigue un camino largo y finalmente regresa a una ubicación visitada anteriormente. El error acumulado en la alineación de frames da como resultado un mapa que tiene dos representaciones de la misma región en diferentes ubicaciones. Esto se conoce como el problema de cierre de ciclo, y la solución a esto consta de dos partes. Primero,

la detección del cierre de ciclo es necesario para reconocer cuando la cámara ha regresado a una ubicación visitada. Segundo, el mapa debe ser corregido para fusionar las regiones duplicadas (Strasdat *et al.*, 2011; Galvez-López y Tardos, 2012; Mur-Artal *et al.*, 2015).

Cada vez que se crea un keyframe se intenta detectar un cierre de ciclo con algún keyframe previo. No todos los keyframes previos necesitan ser considerados, por lo que solo se consideran aquellos keyframes que no forman parte de la ventana local del keyframe procesado. Posteriormente, se usa el vocabulario visual creado offline para identificar los 15 keyframes anteriores con apariencia similar. Los keyframes que pasaron esta prueba están sujetos a la alineación RANSAC con el keyframe actual. Se detecta un cierre de ciclo si RANSAC recupera suficientes correspondencias geométricamente consistentes, entonces, se añade la arista al grafo y los puntos coincidentes se fusionan en puntos únicos.

En cada cierre de ciclo exitoso se optimiza el grafo de poses para distribuir el error a lo largo del grafo y así lograr una consistencia global del mapa.

Capítulo 6. Evaluación

6.1. Métricas de error para SLAM

La precisión de algoritmos de VO/SLAM comúnmente se cuantifica evaluando la trayectoria estimada con respecto a la trayectoria real. Existen dos dificultades principales para realizar esto. Primero, la trayectoria estimada y la real generalmente se expresan en sistemas de referencia diferentes, y, por lo tanto, no se puede comparar directamente. En segundo lugar, una trayectoria consiste de los estados en muchos tiempos diferentes, por lo que son datos de alta dimensión.

Para abordar el primer problema, la trayectoria estimada requiere ser transformada al mismo sistema de referencia que la real, esto se denomina alineación de trayectoria y se puede realizar mediante diferentes algoritmos como el propuesto por Umeyama (1991). Para lidiar con el segundo problema, se deben usar métricas de error significativas donde sus propiedades sean muy descriptivas. Las métricas de error más utilizados son el error de trayectoria absoluto (ATE) y el error de trayectoria relativo (RTE) (Zhang y Scaramuzza, 2018).

6.1.1. Error absoluto

Una métrica muy común para calcular el error de estimación entre la trayectoria real $\mathbf{X}_{gt}$ y la estimación alineada $\hat{\mathbf{X}}'$ es el error de trayectoria absoluto (ATE).

Para un estado, el error entre $\hat{\mathbf{x}}'_i \in \hat{\mathbf{X}}'$ y la real $\mathbf{x}_i \in \mathbf{X}_{gt}$ puede ser parametrizado como

$$\Delta\mathbf{x}_i = \{\Delta\mathbf{R}_i, \Delta\mathbf{t}_i\}, \tag{137}$$

donde

$$\begin{aligned} \Delta\mathbf{R}_i &= \mathbf{R}_i(\hat{\mathbf{R}}'_i)^T \\ \Delta\mathbf{t}_i &= \mathbf{t}_i - \Delta\mathbf{R}_i\hat{\mathbf{t}}'_i \end{aligned} \tag{138}$$

Para cuantificar la calidad de toda la trayectoria, generalmente se usa el error cua-

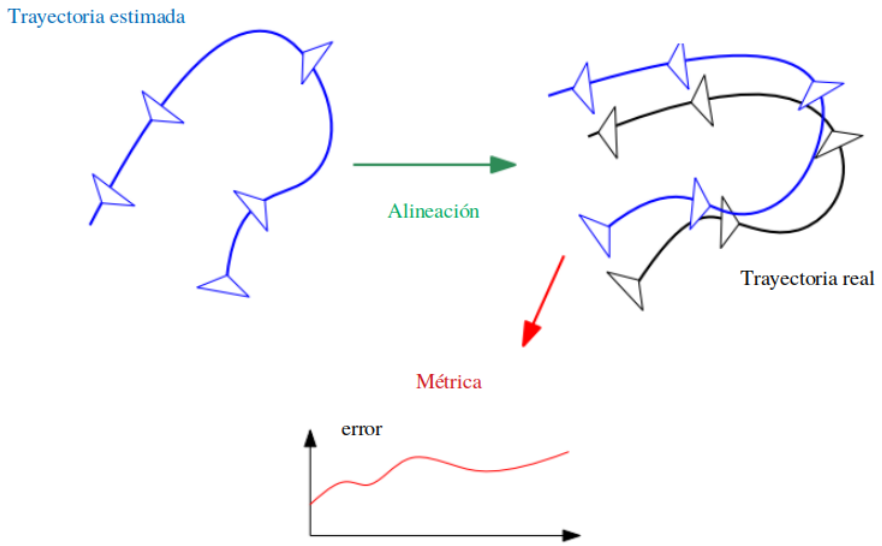


Figura 33. Proceso de evaluación cuantitativa de trayectoria. Primero, la trayectoria estimada (azul) debe alinearse con la real (negro). Luego, el error se calcula aplicando alguna métrica a las trayectorias alineada y real.

drático medio (RMSE)

$$ATE_{rot} = \left(\frac{1}{N} \sum_i \| \measuredangle(\Delta \mathbf{R}_i) \|^2 \right)^{1/2}$$
$$ATE_{tras} = \left(\frac{1}{N} \sum_i \| \Delta \mathbf{t}_i \|^2 \right)^{1/2}, \tag{139}$$

donde $\measuredangle(\cdot)$ significa convertir la matriz de rotación a una representación de ángulos y usar el ángulo de rotación como el error. La ventaja de ATE es que proporciona una métrica de número único para la estimación de la posición y rotación que es fácil de comparar. Sin embargo, ATE es sensible al momento en que el error ocurre. Por ejemplo, un error de estimación de rotación tiende a dar un ATE más grande cuando ocurre al comienzo de la trayectoria que si ocurriera al final. Por lo tanto, además de ATE, el error relativo también se usa ampliamente para proporcionar una evaluación más informativa.

6.1.2. Error relativo

La idea básica del error relativo es que, dado que los sistemas VO/SLAM no tienen una referencia global, la calidad de la estimación se puede evaluar midiendo las relaciones relativas entre los estados en diferentes tiempos. Formalmente, primero se

selecciona un conjunto de K pares de estados por algún criterio (por ejemplo, distancia a lo largo de la trayectoria) de $\hat{\mathbf{X}}$

$$\mathcal{F} = \{\mathbf{d}_k\}_{k=0}^{K-1}, \qquad \mathbf{d}_k = \{\hat{\mathbf{x}}_s, \hat{\mathbf{x}}_e\}, \tag{140}$$

donde $e > s$, cada par define una subtrayectoria. Para cada $\mathbf{d}_k$, el error relativo es calculado de manera similar que el error absoluto. Una transformación de alineación se calcula a partir del primer estado $\hat{\mathbf{x}}_s$ y su correspondiente valor real $\mathbf{x}_s$, y el segundo estado alineado $\hat{\mathbf{x}}_e'$. Entonces el error $\delta\mathbf{d}_k$ para el par $\mathbf{d}_k$ es

$$\begin{aligned} \delta\phi_k &= \measuredangle\delta\mathbf{R}_k = \measuredangle\mathbf{R}_e(\hat{\mathbf{R}}_e')^T \\ \delta\mathbf{t}_k &= \| \mathbf{t}_e - \delta\mathbf{R}_k\hat{\mathbf{t}}_e' \|_2 \ . \end{aligned} \tag{141}$$

Recolectar el error 141 para todos los pares de estado (subtrayectortias) en $\mathcal{F}$ resulta en

$$\begin{aligned} RE_{rot} &= \{\delta\phi_k\}_{k=0}^{K-1} \\ RE_{tras} &= \{\delta\mathbf{t}_k\}_{k=0}^{K-1}. \end{aligned} \tag{142}$$

Dado que el error relativo no genera un solo número, sino una colección de errores para todas las subtrayectorias que satisfacen ciertos criterios, se puede calcular estadísticas como la mediana, el promedio y los percentiles, lo que proporciona más información que ATE. Otra ventaja es que, al seleccionar los estados de acuerdo a diferentes criterios, RTE puede tener diferentes significados. Por ejemplo, una práctica común es seleccionar pares de estado que están separados por una cierta distancia a lo largo de la trayectoria. El RTE de los pares de estado que están espacialmente cercanos refleja la consistencia local mientras que el error para una distancia mayor releja más la precisión a largo plazo. La desventaja de RTE es que es relativamente complicado de calcular y es menos obvio clasificar la calidad de la estimación que usar una sola métrica numérica como ATE.

6.2. Recursos de hardware y software utilizados

El sistema fue desarrollado en una computadora marca ASUS, misma en la que fueron realizados todas las evaluaciones que a continuación se presentan. La computadora cuenta con un procesador Intel Core i5-4200U, 16 Gb de memoria RAM, sistema operativo Ubuntu 18.04 de 64 bits.

El proyecto ha sido exitosamente implementado en lenguaje C++ gracias a la existencia de librerías de código abierto específicas para cada uno de los elementos que lo componen. A continuación se describen las características más relevantes de las librerías de software utilizadas en el desarrollo del proyecto.

- **OpenCV (3.4.5)** (Open Source Computer Vision Library)[1]. Librería de visión por computadora y procesamiento digital de imágenes.
- **PCL (1.9.1)** (Point Cloud Library)[2]. Librería con un gran número de algoritmos de procesamiento de nubes de puntos y geometría 3D.
- **Eigen (3)**[3]. Biblioteca de alto nivel para álgebra lineal, operaciones con matrices, vectores, métodos numéricos y algoritmos relacionados.
- **g2o (1)** (General Graph Optimization)[4]. Librería para optimizar funciones de error no lineales basadas en grafos. La implementación proporciona soluciones a muchas variantes de SLAM y BA.
- **DBoW3 (1)**[5]. Librería para indexar y convertir imágenes en una representación de bolsa de palabras. Implementa árboles jerárquicos para aproximarse a los vecinos más cercanos en el espacio de características de la imagen y crear un vocabulario visual.
- **Pangolin (0.5)**[6]. Biblioteca de desarrollo rápido para administrar la visualización/interacción de OpenGL y abstraer la entrada de video.

[1] https://github.com/opencv/opencv
[2] https://github.com/PointCloudLibrary/pcl
[3] http://eigen.tuxfamily.org
[4] https://github.com/RainerKuemmerle/g2o
[5] https://github.com/rmsalinas/DBow3
[6] https://github.com/stevenlovegrove/Pangolin

- **OpenGV (1)**[7]. Provee una colección de métodos de visión por computadora para resolver problemas de visión geométrica.
- **OctoMap (1.9)**[8]. Implementa un enfoque de mapeo de cuadrícula de ocupación 3D, proporcionando estructuras de datos y algoritmos de mapeo particularmente adecuados para la robótica. La implementación del mapa se basa en octrees.

6.3. Conjunto de datos TUM RGB-D

El conjunto de datos TUM RGB-D (Sturm *et al.*, 2012) proporciona secuencias en entornos interiores capturadas usando un sensor Microsoft Kinect v.1 con resolución VGA junto con las trayectorias reales del sensor y un conjunto de herramientas para evaluar fácilmente la calidad de las trayectorias estimadas.

Las secuencias del conjunto de datos se dividen en categorías. Para evaluar el sistema se usaron secuencias de las siguientes categorías correspondientes a SLAM visual:

- Testing and debugging: Estas secuencias están destinadas para facilitar el desarrollo de nuevos algoritmos con movimientos separados a lo largo y alrededor de los ejes principales del kinect
- Handheld SLAM: Contiene 11 secuencias con un kinect de mano, es decir, movimientos de la cámara con 6 grados de libertad.
- Robot SLAM: El kinect se montó en un robot ActivMedia Pioneer 3. Con estas secuencias es posible demostrar la aplicabilidad del sistema SLAM a los robots con ruedas.

El sistema fue evaluado usando secuencias de las categorías anteriormente mencionadas con diferentes niveles de dificultad con resultados cuantitativos que se muestran en la Tabla 4. Se usó la métrica de error cuadrático medio del ATE para evaluar el sistema. El error relativo RTE traslacional fue calculado para diferentes longitudes

[7] https://github.com/laurentkneip/opengv
[8] https://github.com/OctoMap/octomap

de subtrayectorias como se muestra en la Fig. 34. En la Fig. 35 se muestra el error relativo rotacional para el ángulo yaw.

Se logró un rendimiento consistente en todas las secuencias evaluadas, con un error notablemente mayor en las secuencias fr1/360, fr2/pioneer_360 y fr2/pioneer_slam2. Esto puede explicarse por la alta velocidad angular promedio en estas secuencias que causa desenfoque de movimiento, aumenta el efecto obturador y viola la suposición de asociación de datos proyectiva. De los resultados se puede ver que el RMSE más alto se correlaciona con una velocidad angular promedio alta. Siempre que haya una desviación estándar baja en la velocidad de las imágenes y una buena superposición entre imágenes sucesivas, se puede lograr una buena estimación de la trayectoria. En la Fig. 36 se muestran gráficas bidimensionales de las diferencias entre la trayectoria estimada y la trayectoria real. En todos los conjuntos de datos del mundo real evaluados, se habilitaron las funciones de exposición automática y balance de blancos automático de la cámara RGB-D.

6.4. Evaluación comparativa

Se comparó la estimación de la trayectoria del sistema desarrollado con tres sistemas del estado del arte, DVO-SLAM (Kerl *et al.*, 2013), RGBD-SLAM (Endres *et al.*, 2014) y ORB-SLAM2 (Mur-Artal y Tardós, 2016). DVO-SLAM es un sistema directo-denso, es decir su frontend minimiza el error fotométrico usando gran parte de los píxeles de la imagen para estimar la posición de la cámara. RGBD-SLAM es un sistema indirecto basado en características, es decir, preprocesa las imágenes para extraer puntos de interés visuales y encontrar correspondencias con otras imágenes, su frontend estima la posición de la cámara usando un enfoque RANSAC+ICP de manera convencional. ORB-SLAM2 es un sistema indirecto basado en características visuales ORB, su frontend aplica BA para estimar la posición de la cámara.

A diferencia de los sistemas anteriormente mencionados, ASLAM es un sistema indirecto-disperso+denso. El frontend puede disponer solo de las características visuales FAST para calcular la trayectoria usando un enfoque RANSAC+GICP de manera adaptativa o usar la nube de puntos densa para estimar la trayectoria usando solo

Tabla 4. Estadísticas sobre ATE en el conjunto de datos TUM RGB-D. Los valores de trayectoria están en metros y representan el valor medio de ejecutar 10 veces el sistema.

Secuencia	RMSE	Promedio	Mediana	Dev. Std.	Min	Max
fr1/desk	0.021	0.016	0.013	0.01	0.001	0.061
fr1/desk2	0.039	0.030	0.024	0.024	0.002	0.184
fr1/room	0.036	0.034	0.0304	0.033	0.018	0.067
fr1/xyz	0.01	0.008	0.007	0.005	0.0004	0.029
fr1/plant	0.020	0.017	0.015	0.010	0.001	0.097
fr1/360	0.117	0.103	0.084	0.076	0.013	0.418
fr1/floor	0.028	0.019	0.014	0.021	0.008	0.120
fr2/desk	0.038	0.029	0.021	0.025	0.003	0.184
fr2/pioneer_360	0.195	0.180	0.177	0.073	0.038	0.549
fr2/pioneer_slam2	0.699	0.580	0.513	0.389	0.076	3.045
fr2/xyz	0.0049	0.0044	0.004	0.0021	0.0002	0.017
fr3/long_office	0.011	0.010	0.009	0.004	0.0005	0.035
fr3/nstl	0.021	0.018	0.016	0.010	0.0005	0.055
fr3/stf	0.010	0.009	0.008	0.005	0.0008	0.037
fr3/stn	0.011	0.010	0.009	0.005	0.0005	0.037

GICP si RANSAC falla.

La métrica de error usada para llevar a cabo la comparación de los sistemas fue el RMSE del ATE. En la Tabla 5 se muestran los resultados de la comparación de sistemas, donde los valores de ASLAM son el valor promedio de ejecutar 10 veces el sistema. Se observa que el rendimiento de nuestro sistema arroja resultados competentes con el de los demás enfoques y en algunos casos mejores.

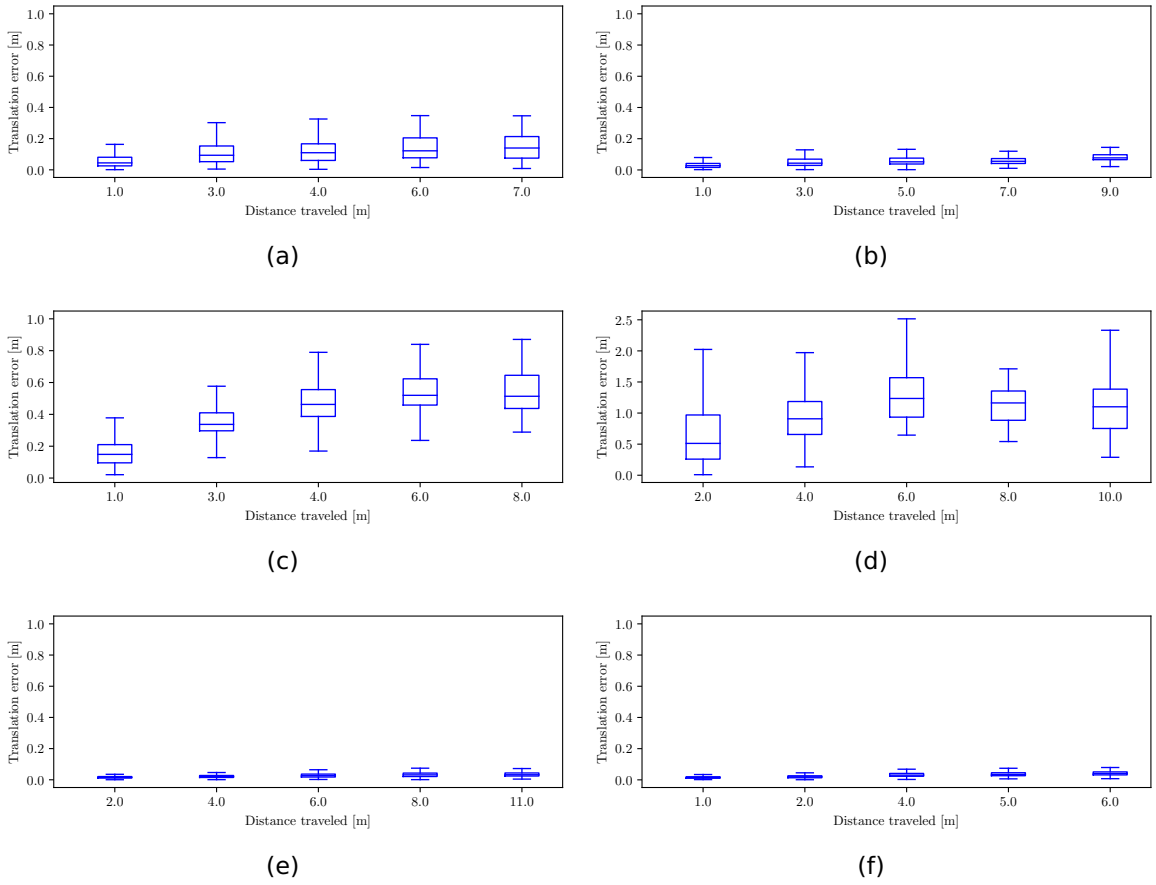


Figura 34. Error relativo traslacional para diferentes longitudes de subtrayectorias mostradas como una serie de diagrama de cajas. El recuadro en el medio indica los dos cuartiles de todos los errores de la estimación, la línea a través del recuadro la mediana y los bigotes los cuartiles superior e inferior. (a) fr1/room. (b) fr2/desk. (c) fr3/pioneer_360. (d) fr2/pioneer_slam2. (e) fr3/long_office. (f) fr3/nstl.

Tabla 5. Comparasión del RMSE del error absoluto de trayectoria (m).

Secuencia	ASLAM	ORB-SLAM2	RGBD-SLAM	DVO-SLAM
fr1/desk	0.021	**0.016**	0.026	0.021
fr1/desk2	0.039	**0.022**	-	0.046
fr1/room	**0.036**	0.047	0.087	0.043
fr2/desk	0.038	**0.009**	0.057	0.017
fr2/xyz	0.0049	**0.004**	-	0.018
fr3/long_office	0.011	**0.010**	-	0.035

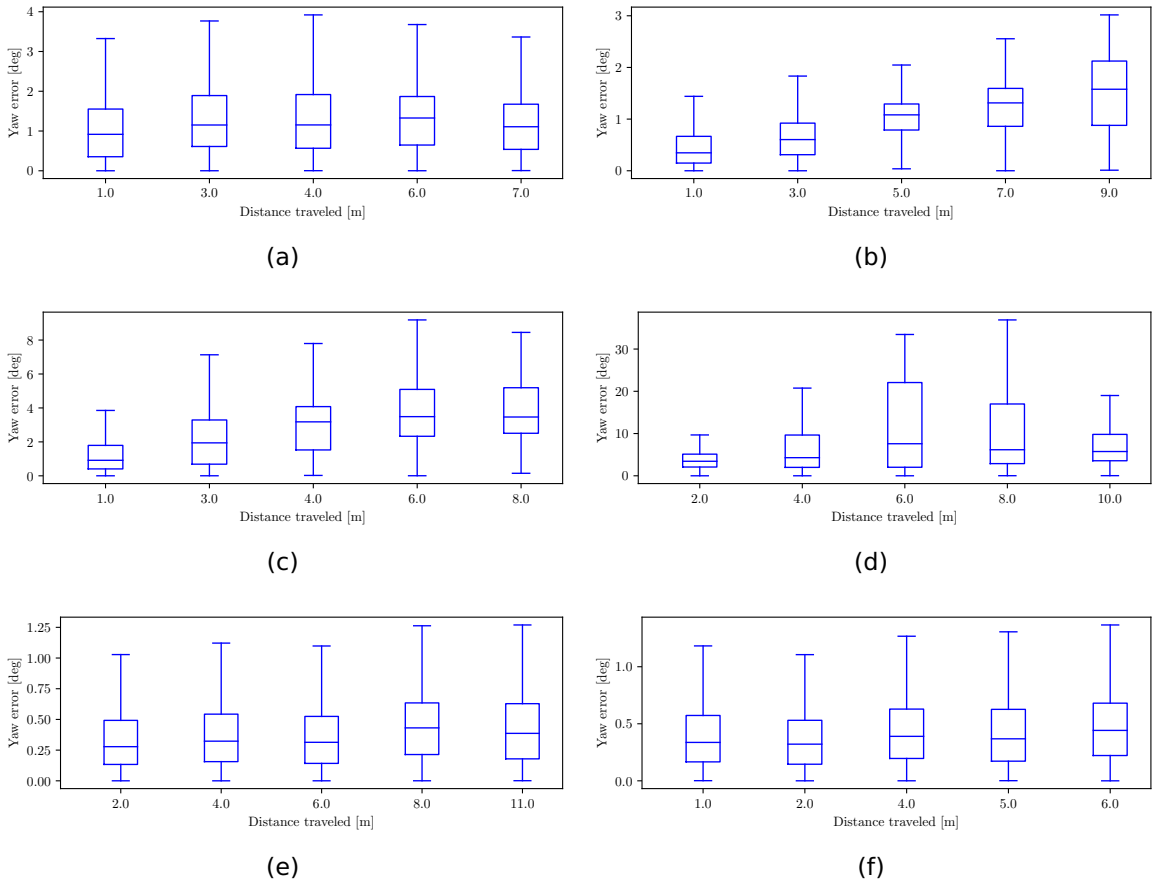


Figura 35. Error relativo rotacional para el ángulo yaw. (a) fr1/room. (b) fr2/desk. (c) fr3/pioneer_360. (d) fr2/pioneer_slam2. (e) fr3/long_office. (f) fr3/nstl.

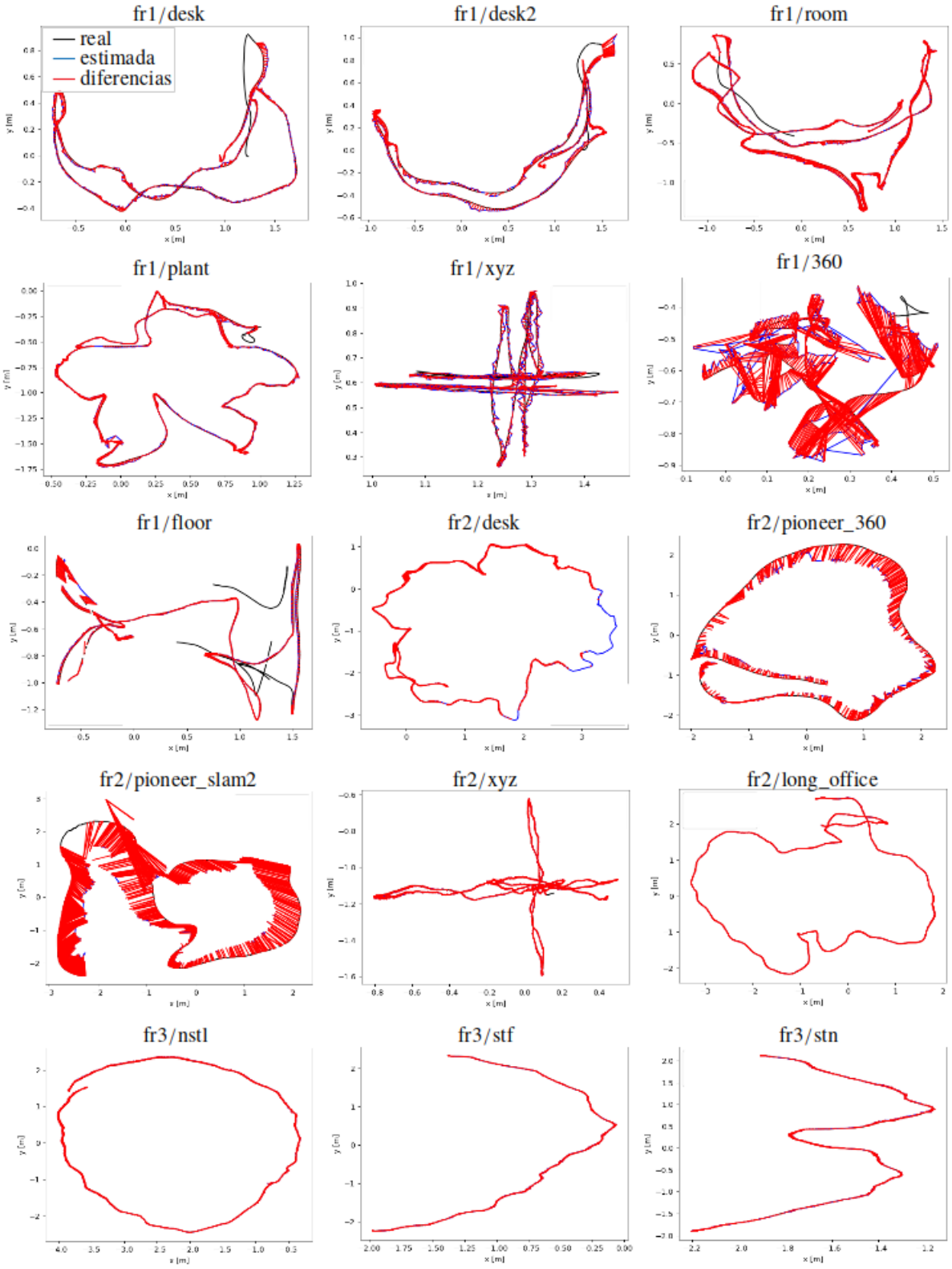


Figura 36. Gráficas bidimensionales de la trayectoria estimada vs. trayectoria real del conjunto de datos TUM RGB-D.

Capítulo 7. Conclusiones

En esta tesis se abordó el problema de localización y mapeo simultáneo utilizando cámaras RGB-D. El sistema propuesto hace uso de algoritmos adaptativos para brindar soluciones robustas y precisas que escalen bien en entornos grandes, trabaja en tiempo real en CPUs convencionales y proporciona una localización muy precisa cuando opera continuamente en el mismo entorno.

Se realizó un estudio comparativo entré los diferentes algoritmos de detección y descripción de características visuales más utilizados en SLAM y odometría visual para utilizar aquella combinación más adecuada para nuestro sistema. Se concluyó que las combinaciones GFTT-ORB, GFTT-BRIEF, FAST-ORB, FAST-BRIEF y ORB-ORB son buenos candidatos para lograr un seguimiento de la cámara en tiempo real y arrojar buenos resultados. Se observó que ORB-ORB tiene una buena repetibilidad y AUC bajo condiciones de rotación y cambios de escala. Las combinaciones GFTT-ORB y GFTT-BRIEF tienen la mejor repetibilidad en la mayoría de los casos y un AUC aceptable, excepto bajo condiciones de rotación y cambios de escala. Se observó que las combinaciones FAST-ORB y FAST-BRIEF tienen un rendimiento similar que las combinaciones con el detector GFTT con la ventaja de un tiempo de extracción mucho menor. A partir de estas observaciones, se puede llegar a la conclusión de que cualquiera de estas combinaciones es adecuadas para problemas de SLAM y odometría visual. Tomando en cuenta las características de la cámara RGB-D usada se decidió usar la combinación FAST-BRIEF, ya que tiene el mejor tiempo de extracción y además el movimiento que se encuentra en imágenes sucesivas a 30 Hz. es pequeño. Para cámaras que trabajen a frecuencias más bajas se recomienda usar la combinación ORB-ORB, ya que, el movimiento entre imágenes sucesivas será mayor.

Un componente clave para lograr un sistema robusto y preciso fue la incorporación del módulo de localización RGB-D adaptativa. Este módulo brinda una alternativa en situaciones donde el desenfoque causado por movimientos rápidos de la cámara o por falta de textura no permita extraer características visuales y por consiguiente no será posible estimar correspondencias visuales con RANSAC. Cuando esto sucede la estimación de movimiento entre imágenes se calcula usando el par de nubes de puntos densas mediante el algoritmo GICP donde las correspondencias se estiman usando un algoritmo de búsqueda de vecinos más cercanos. GICP no solo sirve como método

de recuperación si RANSAC falla, sino que también ayuda a refinar la pose estimada con RANSAC. La incorporación de este algoritmo adaptativo fue muy importante para hacer frente a la desventaja más importante en sistemas puramente basados en características, la falta de correspondencias visuales.

La principal mejora del mapa topológico es el uso de información de covisibilidad entre imágenes en lugar de orden temporal. La información de covisibilidad vincula las imágenes de la misma escena independientemente del momento en que se capturaron las imágenes. Esto es importante en SLAM visual donde los keyframes de la misma escena se pueden insertar en momentos muy diferentes cuando la cámara vuelve a visitar el entorno. Este enfoque ha demostrado que funciona en tiempo real para cierre de ciclos, logrando una precisión perfecta.

Un elemento fundamental para brindar soluciones globalmente consistentes es el reconocimiento visual de lugares para cerrar ciclos y reubicar la cámara RGB-D. El método de reconocimiento de lugares se basa en representar las imágenes como una bolsa de palabras, esta representación permite comparar la similitud entre imágenes de manera muy eficiente. Para todos los experimentos se usó el mismo vocabulario FAST de 1 millón de palabras, lo que demuestra que el vocabulario funciona bien en muchas situaciones. Si bien rara vez se observaron cierres de ciclo falsos (que tienen un efecto catastrófico en la estimación), suceden y constituyen un desafío abierto. Para reducir el número de cierre de ciclos falsos positivos se aplicó una validación geométrica mediante el algoritmo RANSAC.

Uno de las principales desventajas de los enfoques basados en características es su reconstrucción dispersa. Esta reconstrucción tiene la ventaja de ser eficiente en términos de memoria, sin embargo, para tareas como navegación o interacción robótica su uso es limitado. Por esto se propone el uso de mapas probabilísticos densos aprovechando todos los datos proporcionados por el sensor RGB-D. De esta manera, nuestro sistema combina la alta precisión de mapas topológicos, robustez de la localización adaptativa, capacidad de reconocimiento de lugares y reconstrucción de mapas métricos densos con amplias aplicaciones en el campo de la robótica.

7.1. Trabajo futuro

- Ampliar el sistema para trabajar con diferentes tipos de cámaras.
- Mapeo en entornos dinámicos.
- Implementación de mantenimiento del mapa denso en GPU para obtener mejores resoluciones.
- Implementar una solución híbrida que combine las ventajas de los métodos directos y los basados en características. Los métodos directos son robustos al desenfoque de movimiento y pueden explotar mejor la información de la imagen cuando hay poca textura. Por otro lado, los métodos basados en características son útiles para el reconocimiento de lugares, cierre de ciclos y reutilización de mapas.

Literatura citada

Agrawal, M., Konolige, K., y Blas, M. R. (2008). Censure: Center surround extremas for realtime feature detection and matching. En: *Computer Vision – ECCV 2008*, Berlin, Heidelberg. Springer Berlin Heidelberg, pp. 102–115.

Arun, K. S., Huang, T. S., y Blostein, S. D. (1987). Least-squares fitting of two 3-d point sets. *IEEE Trans. Pattern Anal. Mach. Intell.*, **9**(5): 698–700.

Bailey, T. y Durrant-Whyte, H. (2006). Simultaneous localization and mapping (slam): part ii. *IEEE Robotics Automation Magazine*, **13**(3): 108–117.

Bay, H., Ess, A., Tuytelaars, T., y Van Gool, L. (2008). Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, **110**(3): 346–359.

Besl, P. J. y McKay, N. D. (1992). A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, **14**(2): 239–256.

Betge-Brezetz, S., Hebert, P., Chatila, R., y Devy, M. (1996). Uncertain map making in natural environments. **2**: 1048–1053 vol.2.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag. Berlin, Heidelberg.

Cadena, C., Galvez-Lopez, D., Tardos, J. D., y Neira, J. (2012). Robust place recognition with stereo sequences. *IEEE Transactions on Robotics*, **28**(4): 871–885.

Calonder, M., Lepetit, V., Strecha, C., y Fua, P. (2010). Brief: Binary robust independent elementary features. En: *Proceedings of the 11th European Conference on Computer Vision: Part IV*, Berlin, Heidelberg. Springer-Verlag, ECCV'10, pp. 778–792.

Chum, O. y Matas, J. (2005). Matching with prosac "progressive sample consensus. En: *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, Washington, DC, USA. IEEE Computer Society, CVPR '05, pp. 220–226.

Davison, A. J. (1998). *Mobile robot navigation using active vision.*. Phd thesis, University of Oxford.

Durrant-Whyte, H. y Bailey, T. (2006). Simultaneous localization and mapping: part i. *IEEE Robotics Automation Magazine*, **13**(2): 99–110.

Eggert, D. W., Lorusso, A., y Fisher, R. B. (1997). Estimating 3-d rigid body transformations: A comparison of four major algorithms. *Mach. Vision Appl.*, **9**(5-6): 272–290.

Endres, F., Hess, J., Sturm, J., Cremers, D., y Burgard, W. (2014). 3d mapping with an rgb-d camera. En: *IEEE Transactions on Robotics*.

Engel, J. y Cremers, D. (2014). Lsd-slam: Large-scale direct monocular slam. En: *In ECCV*.

Engel, J., Koltun, V., y Cremers, D. (2018). Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **40**(3): 611–625.

Fischler, M. A. y Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, **24**(6): 381–395.

Forster, C., Pizzoli, M., y Scaramuzza, D. (2014). Svo: Fast semi-direct monocular visual odometry. En: *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May. pp. 15–22.

Forster, C., Zhang, Z., Gassner, M., Werlberger, M., y Scaramuzza, D. (2017). Svo: Semidirect visual odometry for monocular and multicamera systems. *Trans. Rob.*, **33**(2): 249–265.

Fuentes-Pacheco, J., Ruiz-Ascencio, J., y Rendón-Mancha, J. M. (2015). Visual simultaneous localization and mapping: A survey. *Artif. Intell. Rev.*, **43**(1): 55–81.

Galvez-López, D. y Tardos, J. D. (2012). Bags of binary words for fast place recognition in image sequences. *Trans. Rob.*, **28**(5): 1188–1197.

Galvez-Lopez, D. y Tardos, J. D. (2012). Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, **28**(5): 1188–1197.

Garcia-Fidalgo, E. y Ortiz, A. (2015). Vision-based topological mapping and localization methods. *Robot. Auton. Syst.*, **64**(C): 1–20.

Gomez-Ojeda, R., Briales, J., y Gonzalez-Jimenez, J. (2016). Pl-svo: Semi-direct monocular visual odometry by combining points and line segments. En: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. pp. 4211–4216.

Grisetti, G., Stachniss, C., Grzonka, S., y Burgard, W. (2007). A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. En: *In Proc. of Robotics: Science and Systems (RSS)*.

Grisetti, G., Kuemmerle, R., Stachniss, C., y Burgard, W. (2010). A tutorial on graph-based SLAM. *Intelligent Transportation Systems Magazine, IEEE*, **2**(4): 31–43.

Handa, A., Whelan, T., McDonald, J., y Davison, A. (2014). A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. En: *IEEE Intl. Conf. on Robotics and Automation, ICRA*, May, Hong Kong, China.

Harris, C. y Stephens, M. (1988). A combined corner and edge detector. En: *In Proc. of Fourth Alvey Vision Conference*. pp. 147–151.

Harris, C. G. y Pike, J. M. (1988). 3d positional integration from image sequences. *Image Vision Comput.*, **6**(2): 87–90.

Hartley, R. y Zisserman, A. (2003). *Multiple View Geometry in Computer Vision*. Cambridge University Press, segunda edición. New York, NY, USA.

Henry, P., Krainin, M., Herbst, E., Ren, X., y Fox, D. (2012). Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *The International Journal of Robotics Research*, **31**(5): 647–663.

Hirschmuller, H., Innocent, P. R., y Garibaldi, J. M. (2002). Fast, unconstrained camera motion estimation from stereo without tracking and robust statistics. En: *7th International Conference on Control, Automation, Robotics and Vision, 2002. ICARCV 2002.*, Dec. Vol. 2, pp. 1099–1104 vol.2.

Horn, B. K. P. (1987). Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, **4**(4): 629–642.

Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., y Burgard, W. (2013). OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*. Software disponible en `http://octomap.github.com`.

Howard, A. (2008). Real-time stereo visual odometry for autonomous ground vehicles. En: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. pp. 3946–3952.

Huang, A. S., Bachrach, A., Henry, P., Krainin, M., Maturana, D., Fox, D., y Roy, N. (2017). Visual odometry and mapping for autonomous flight using an rgb-d camera. En: H. I. Christensen y O. Khatib (eds.), *Robotics Research : The 15th International Symposium ISRR*, Cham. Springer International Publishing, pp. 235–252.

Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A., y Fitzgibbon, A. (2011). Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. En: *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, New York, NY, USA. ACM, UIST '11, pp. 559–568.

Kerl, C., Sturm, J., y Cremers, D. (2013). Robust odometry estimation for rgb-d cameras. En: *2013 IEEE International Conference on Robotics and Automation*, May. pp. 3748–3754.

Khoshelham, K. y Elberink, E. O. (2012). Accuracy and resolution of kinect depth data for indoor mapping applications. En: *Sensors 2012, 12, 1437–1454. 2013*. p. 8238.

Klein, G. y Murray, D. (2007). Parallel tracking and mapping for small ar workspaces. En: *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, Nov. pp. 225–234.

Konolige, K. (2010). Sparse sparse bundle adjustment. En: *Proc. BMVC*. pp. 102.1–11, doi:10.5244/C.24.102.

Konolige, K., Bowman, J., Chen, J., Mihelich, P., Calonder, M., Lepetit, V., y Fua, P. (2010). View-based maps. *The International Journal of Robotics Research*, **29**(8): 941–957.

Kuemmerle, R., Grisetti, G., Strasdat, H., Konolige, K., y Burgard, W. (2011). g2o: A general framework for graph optimization. En: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May, Shanghai, China. pp. 3607–3613.

Leonard, J. J. y Durrant-Whyte, H. F. (1991). Simultaneous map building and localization for an autonomous mobile robot. En: *Proceedings IROS '91:IEEE/RSJ International Workshop on Intelligent Robots and Systems '91*, Nov. pp. 1442–1447 vol.3.

Leutenegger, S., Chli, M., y Siegwart, R. Y. (2011). Brisk: Binary robust invariant scalable keypoints. En: *Proceedings of the 2011 International Conference on Computer Vision*, Washington, DC, USA. IEEE Computer Society, ICCV '11, pp. 2548–2555.

Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., y Furgale, P. (2015). Keyframe-based visual-inertial odometry using nonlinear optimization. *Int. J. Rob. Res.*, **34**(3): 314–334.

Levi, G. y Hassner, T. (2016). Latch: Learned arrangements of three patch codes. *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1–9.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, **60**(2): 91–110.

Lu, F. y Milios, E. (1997). Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, **4**(4): 333–349.

Ma, Y., Soatto, S., Kosecka, J., y Sastry, S. S. (2003). *An Invitation to 3-D Vision: From Images to Geometric Models*. SpringerVerlag.

Mallick, T., Das, P. P., y Majumdar, A. K. (2014). Characterizations of noise in kinect depth images: A review. *IEEE Sensors Journal*, **14**(6): 1731–1740.

Mei, C., Benhimane, S., Malis, E., y Rives, P. (2008). Efficient homography-based tracking and 3-d reconstruction for single-viewpoint sensors. *IEEE Transactions on Robotics*, **24**(6): 1352–1364.

Mei, C., Sibley, G., Cummins, M., Newman, P., y Reid, I. (2009). A constant-time efficient stereo slam system. En: *British Machine Vision Conference, BMVC*.

Mei, C., Sibley, G., y Newman, P. (2010). Closing loops without places. En: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. pp. 3738–3744.

Mikolajczyk, K. y Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.*, **27**(10): 1615–1630.

Moravec, H. (1980). *Obstacle avoidance and navigation in the real world by a seeing robot rover.*. Phd thesis, Stanford University.

Mur-Artal, R. y Tardós, J. D. (2016). ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *CoRR*, **abs/1610.06475**.

Mur-Artal, R. y Tardós, J. D. (2015). Probabilistic semi-dense mapping from highly accurate feature-based monocular slam. En: *Robotics: Science and Systems*.

Mur-Artal, R., Montiel, J. M. M., y Tardós, J. D. (2015). ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Trans. Robotics*, **31**(5): 1147–1163.

Newcombe, R. A., Lovegrove, S., y Davison, A. J. (2011). Dtam: Dense tracking and mapping in real-time. En: D. N. Metaxas, L. Quan, A. Sanfeliu, y L. J. V. Gool (eds.), *ICCV*. IEEE Computer Society, pp. 2320–2327.

Nistér, D. (2005). Preemptive ransac for live structure and motion estimation. *Mach. Vision Appl.*, **16**(5): 321–329.

Nister, D. y Stewenius, H. (2006). Scalable recognition with a vocabulary tree. En: *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, Washington, DC, USA. IEEE Computer Society, CVPR '06, pp. 2161–2168.

Nister, D., Naroditsky, O., y Bergen, J. (2004). Visual odometry. En: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, June. Vol. 1, pp. I–I.

Ortiz, R. (2012). Freak: Fast retina keypoint. En: *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Washington, DC, USA. IEEE Computer Society, CVPR '12, pp. 510–517.

Paton, M. y Kosecka, J. (2012). Adaptive rgb-d localization. En: *2012 Ninth Conference on Computer and Robot Vision*, May. pp. 24–31.

Pfister, H., Zwicker, M., van Baar, J., y Gross, M. (2000). Surfels: Surface elements as rendering primitives. En: *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co., SIGGRAPH '00, pp. 335–342.

Piniés, P., Paz, L., Gálvez-López, D., y Tardós, J. (2010). Ci-graph slam for 3d reconstruction of large and complex environments using a multicamera system. *International Journal of Field Robotics*, **27**(5): 561–586.

Raguram, R., Frahm, J., y Pollefeys, M. (2009). Exploiting uncertainty in random sample consensus. En: *2009 IEEE 12th International Conference on Computer Vision*, Sep. pp. 2074–2081.

Rosten, E. y Drummond, T. (2006). Machine learning for high-speed corner detection. En: *Proceedings of the 9th European Conference on Computer Vision - Volume Part I*, Berlin, Heidelberg. Springer-Verlag, ECCV'06, pp. 430–443.

Rublee, E., Rabaud, V., Konolige, K., y Bradski, G. (2011). Orb: An efficient alternative to sift or surf. En: *2011 International Conference on Computer Vision*, Nov. pp. 2564–2571.

Segal, A., Hähnel, D., y Thrun, S. (2009). Generalized-icp. En: *Robotics: Science and Systems (RSS)*. The MIT Press.

Shi, J. y Tomasi, C. (1993). Good features to track. Reporte técnico, Ithaca, NY, USA.

Sivic, J. y Zisserman, A. (2003). Video google: A text retrieval approach to object matching in videos. En: *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, Washington, DC, USA. IEEE Computer Society, ICCV '03, pp. 1470–.

Smisek, J., Jancosek, M., y Pajdla, T. (2011). 3d with kinect. En: *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, Nov. pp. 1154–1160.

Strasdat, H. (2012). *Local accuracy and global consistency for effiient visual SLAM.*. Phd thesis, Imperial College of London.

Strasdat, H., Montiel, J. M. M., y Davison, A. J. (2010). Real-time monocular slam: Why filter? En: *2010 IEEE International Conference on Robotics and Automation*, May. pp. 2657–2664.

Strasdat, H., Montiel, J. M. M., y Davison, A. J. (2010). Scale drift-aware large scale monocular slam. En: *In Proceedings of Robotics: Science and Systems*.

Strasdat, H., Davison, A. J., Montiel, J. M. M., y Konolige, K. (2011). Double window optimisation for constant time visual slam. En: *2011 International Conference on Computer Vision*, Nov. pp. 2352–2359.

Sturm, J., Engelhard, N., Endres, F., Burgard, W., y Cremers, D. (2012). A benchmark for the evaluation of rgb-d slam systems. En: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. pp. 573–580.

Thrun, S., Burgard, W., y Fox, D. (2005). *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press.

Torr, P. H. S. y Zisserman, A. (2000). MLESAC: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, **78**: 138–156.

Triggs, B., McLauchlan, P. F., Hartley, R. I., y Fitzgibbon, A. W. (2000). Bundle adjustment — a modern synthesis. En: B. Triggs, A. Zisserman, y R. Szeliski (eds.), *Vision Algorithms: Theory and Practice*, Berlin, Heidelberg. Springer Berlin Heidelberg, pp. 298–372.

Umeyama, S. (1991). Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, **13**(4): 376–380.

Zhang, Z. y Scaramuzza, D. (2018). A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry. En: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*.