UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

SISTEMA DE ODOMETRÍA VISUAL MONOCULAR PARA ROBOTS MÓVILES

TESIS

PARA OBTENER EL TÍTULO DE
INGENIERO MECATRÓNICO

PRESENTA
RAÚL PERALTA LOZADA

DIRECTOR DE TESIS
ING. STALIN MUÑOZ GUTIÉRREZ

CIUDAD UNIVERSITARIA, MÉXICO, MAYO DE 2013

Dedicado a mi familia,
y al amor de mi vida, Sara.

Agradecimientos

En primer lugar quiero agradercerle a mi familia, gracias por su amor y apoyo incondicional, gracias por creer en mí y por hacer de mí lo que soy ahora. A Sara, por ser tan linda y amorosa todos estos años, y por ayudar a mantener mi sentido de humanidad.
A la Universidad Nacional Autónoma de México, por brindarme la oportunidad de vivir una de las mejores experiencias de mi vida.
A mi director de tesis, Ing. Stalin Muñoz Gutiérrez, por su apoyo durante este trabajo, por sus astutas ideas y reflexiones, y por confiar en mí y en el proyecto que propuse. Al Profesor Yukihiro Minami Koyama, gracias por motivarnos a mí y a mis compañeros a seguir trabajando, por crear un ambiente de trabajo agradable dentro del Taller ORG y finalmente, por los chocolates. Al Dr. Walterio Mayol Cuevas por sus acertados consejos y sugerencias. A mis sinodales, Dr. Boris Escalante Ramírez, Dr. Jesús Savage Carmona, y Dr. Victor Javier González Villela, por su interés en el trabajo y sus observaciones.
Al Programa de Tecnología en Cómputo por darme ese conocimiento invaluable en programación y por brindarme un espacio de trabajo. Al Taller ORG (Open Robotics Group) por su apoyo durante el desarrollo de este proyecto. También quiero agradecerle a la DGAPA y al Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica (PAPIIT) por otorgarme un apoyo económico para el desarrollo de esta tesis.
Y a todos mis amigos: Toche, Bety, Pepe, Rojas, Serri, Cañe, W, Pedro, Buki, Laura, Arqui, Beto, Itzel, Neto, Leo, Toño, Brito, Ram, Rulo y todos los que ya no caben en esta hoja, gracias por los buenos momentos vividos a lo largo de la carrera.

Abstract

The development of visual navigation systems has become an important topic in robotics applications due to its high level of accuracy, great relocalization capability and simple implementation to a variety of work enviroments.
In this thesis we present a system that attacks the localization problem of a robot. Its only position sensor its a digital camera and, without any other information about the enviroment, we obtain a six degree of freedom representation of the state of the robot. We evaluate the state of the art techniques like: keypoints detection and matching, scene reconstruction using point triangulation, pose estimation using the differences between views, removal of outliers techniques, and optimization processes to minimize the estimation error. This is a Keyframe based system, so, it doesn't use all available images to update the camera state, only uses the ones that provide more information during the path estimation. Additionally, we propose several options that can improve the performance of the system making it more robust, efficent and accurate.

Resumen

El desarrollo de sistemas de navegación visual se ha convertido en un tema importante en aplicaciones de robótica debido a su alto nivel de precisión, gran capacidad de relocalización y fácil implementación en diversos entornos.
En esta tesis se presenta un sistema que ataca el problema de localización de un robot. Se hace uso de una cámara digital como único sensor de posición, sin información adicional sobre el entorno se obtiene una representación del estado del robot con seis grados de libertad. Se evalúan las técnicas usadas en el estado del arte como: la detección de puntos de interés y sus emparejamientos entre imágenes, la reconstrucción de la escena usando triangulación, la estimación de la pose a partir de los cambios de perspectiva, las técnicas de filtrado para eliminar asociaciones erróneas, y los procesos de optimización para minimizar el error en la estimación. Este es un sistema basado en Keyframes, es decir, no usa todas las imágenes disponibles para actualizar el estado de la cámara, sólo aquellas que brinden mayor información durante la trayectoria.
Además se plantean varias propuestas que podrían mejorar a futuro el desempeño del sistema en aspectos de robustez, rapidez y precisión.

Índice general

Capítulo 1

Introducción

Calcular la posición y orientación de un robot es un problema que muchos investigadores han enfrentado en las últimas décadas. Algunos sensores que son usados para realizar esta tarea son codificadores ópticos o de efecto Hall que miden el giro de las ruedas conforme avanzan, o unidades de medición inercial (IMU, por sus siglas en inglés) que pueden estimar la orientación y la velocidad de giro con precisión. También pueden usarse sistemas satelitales de localización como el GPS, y sensores de rango activo como el láser o los sensores ultrasónicos. En este trabajo se optó por usar un sensor más común: una cámara de vídeo. Una cámara de vídeo es un producto comercial de uso común, desde el punto de vista del agente es muy conveniente al brindarle una gran cantidad de información sobre el entorno.

La visión por computadora es una disciplina que tiene como objetivo permitir la toma de decisiones basada en la percepción de objetos físicos y escenas capturadas en imágenes digitales. Ésta representa el entorno visible a través de un conjunto de imágenes al que aplica algoritmos para estimar propiedades tales como la forma, la iluminación y la distribución de colores en una escena. Las aplicaciones de este campo contribuyen a áreas como a la medicina, la industria cinematográfica, la seguridad y vigilancia, y la robótica, por mencionar algunas.

La idea de usar la cámara como base para un estimador de posición fue propuesto por Movarec [30], su trabajo tenía como finalidad estimar la posición con seis grados de libertad de los Rovers que explorarían Marte usando una configuración estereoscópica. Posteriormente Nistér et al. [34] en su artículo "*Visual Odometry*" se plantean los pasos y los componentes necesarios para el desarrollo de dicho sistema. El término fue escogido por su similitud con la odometría por ruedas. Nistér et al.probaron su método usando una sola cámara y un sistema estéreo, pudiendo estimar una trayectoria de 184 m con un error de 4.1 m y logrando un desempeño más alto que la odometría con ruedas. Fue a partir de este trabajo que comenzaron a surgir sistemas autónomos con sistemas de navegación visual. En la actualidad existen robots que usan sistemas de odometría visual, ya sean robots terrestres, aéreos o submarinos (Figura 1.0.2).

Figura 1.0.1: Vehículo autónomo usado por Nistér en su trabajo de odometría visual. Cuenta adelante con dos sistemas de cámaras estéreo que se usan para aumentar la estabilidad del algoritmo.

1.1. Definición del problema

Una definición formal del término es la siguiente: *La odometría visual es el proceso de estimar la posición y orientación de un agente usando la información de una o más cámaras sujetas a él y examinando los cambios que el movimiento produce sobre las imágenes.* Algunas de las áreas en que la odometría visual más se utiliza son la robótica móvil y la realidad aumentada.

El problema se puede formular de la siguiente manera: un agente (por ejemplo, un robot) se mueve a través de un entorno tomando imágenes con una cámara en instantes discretos k, el conjunto de imágenes tomadas se denota como $I_{0:n} = \{I_0...I_n\}$. Dos posiciones contiguas de la cámara se relacionan por una transformación de cuerpo rígido de la siguiente manera:

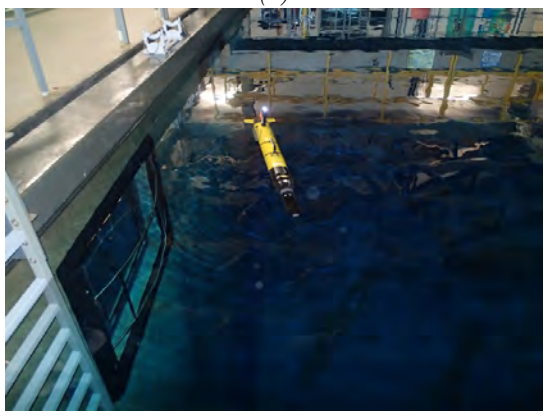
$$T_k = \begin{bmatrix} R_k & t_k \\ 0 & 1 \end{bmatrix} \tag{1.1.1}$$

donde R es la matriz de rotación y t es el vector de traslación. La tarea principal de la odometría visual consiste en estimar estas transformaciones T_k entre pares de imágenes. Estas transformaciones se concatenan de forma incremental hasta obtener la última posición de la cámara que corresponde con la imagen I_n. Hasta el momento es un sistema que cuenta con varias restricciones, es por eso que deben de asumir las siguientes condiciones para que funcione adecuadamente:

- Debe existir suficiente iluminación en el entorno de trabajo.
- Debe de haber suficiente textura que permita extraer movimiento aparente entre vistas.
- El entorno debe ser predominantemente estático.

(a)

(b)



(c)

Figura 1.0.2: Ejemplos de robots que cuentan con sistemas de odometría visual: *(a)* Cuadricóptero que pertenece al proyecto sFly (2009-2011) [2, 3]; *(b)* Automóvil con conducción autónoma desarrollado en la Universidad de Oxford (2012) [17]; y *(c)* robot submarino desarrollado en la Universidad de Tecnología de Queensland (2012) [27].

1.2. Objetivos

El objetivo de esta tesis es diseñar e implementar un sistema que un robot móvil pueda usar para estimar su posición de forma autónoma usando como información del entorno únicamente las imágenes que genera una cámara digital. Desde el punto de vista de la aplicación para robótica se deben de cumplir los siguientes puntos:

- La ejecución del sistema debe ser en tiempo real, es decir, se debe de estimar posiciones al menos 10 veces por segundo [40, 42].
- Las estimaciones deben tener un grado de error mínimo.
- El sistema debe funcionar en cualquier entorno, ya sea en espacios cerrados (oficinas, hogares, hospitales) o en espacios abiertos (ciudades, carreteras, bosques) siempre que se cumplan las restricciones mencionadas en la sección anterior.

Capítulo 2

Marco teórico

A continuación se describirán algunos conceptos preliminares que son necesarios para el entendimiento del algoritmo propuesto. El nivel de profundidad que se abordará será meramente introductorio, y cada apartado cuenta con referencias a documentos que profundizan en la teoría.

2.1. Modelo de la cámara y su calibración

2.1.1. Modelo de la cámara

Las imágenes se generan con cámaras digitales, las cámaras contienen sensores sensibles a la luz que capturaran los rayos que los objetos de la escena reflejan. Una imagen es la representación en dos dimensiones de una escena en tres dimensiones. La geometría proyectiva es la herramienta que describe y caracteriza la formación de una imagen.

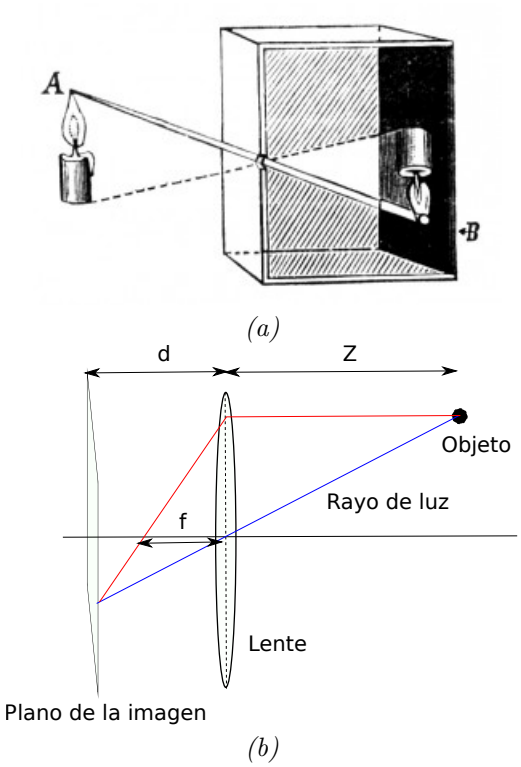
2.1.1.1. Geometría proyectiva

Un modelo de cámara que es sencillo y muy utilizado es el de *pinhole*, éste modelo ha sido usado desde los tiempos de los griegos. La idea principal es la siguiente, la luz reflejada por un objeto es capturada por una cámara a través de una apertura frontal, y la luz capturada se proyecta sobre un plano (sensores de la cámara). Los lentes ayudan a concentrar mayor luz proveniente de la escena. Esto se ve ilustrado en la Figura 2.1.1.

En la Figura 2.1.1 Z es la distancia que va desde el objeto al lente de la cámara, d es la distancia del lente al plano de la imagen, y f es la distancia focal del lente. Estos valores se relacionan con la siguiente ecuación:

$$\frac{1}{f} = \frac{1}{Z} + \frac{1}{d} \tag{2.1.1}$$

Al utilizar el modelo pinhole se simplifican algunos factores. Primero, no se considera el efecto de distorsión que el lente genera y se asume que la cámara



(a)

(b)

Figura 2.1.1: En *(a)* se muestra el concepto de *pinhole*, como un objeto en el espacio 3D es mapeado a un plano. Y en *(b)* se ilustra como este modelo es usado para describir la proyección en una cámara moderna.

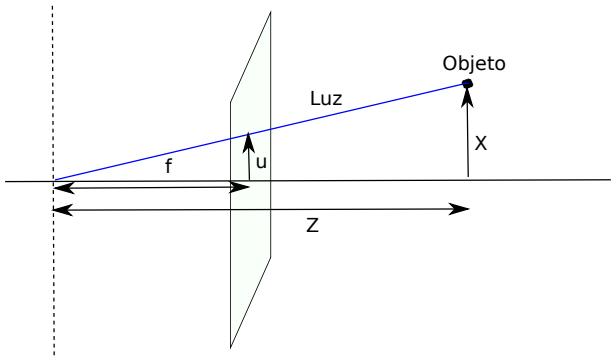


Figura 2.1.2: Modelo equivalente a la Figura 2.1.1. Este es usado para simplicar las operaciones de proyección.

tiene una apertura infinitesimal. Segundo, la mayoría de las veces sucede que $Z >> d$ asumimos que el plano de la imagen se localiza a la distancia focal. Y finalmente, notamos que en el plano la imagen se proyecta de manera invertida, se puede obtener una imagen idéntica y sin invertir simplemente colocando el plano de la imagen enfrente del lente. Esto no es físicamente posible pero desde el punto de vista matemático es equivalente (Figura 2.1.2).

De este modelo, y usando la ley de triángulos similares podemos derivar la siguiente ecuación.

$$u = f\frac{X}{Z} \tag{2.1.2}$$

Con esta ecuación podemos mapear un objeto del mundo al plano de la imagen. Este mapeo es un mapeo no lineal. Sin embargo, usando coordenadas homogéneas podemos obtener ecuaciones lineales como se muestran a continuación.

$$\begin{bmatrix} \lambda u \\ \lambda v \\ \lambda \end{bmatrix} = \begin{bmatrix} fx \\ fy \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.1.3}$$

Existen otras consideraciones que se deben de tomar en cuenta como: el centro óptico de la cámara con coordenadas (u_0, v_0) con respecto a la esquina superior izquierda, se debe notar que el centro óptico no corresponde al centro del sensor CCD; las coordenadas del punto en el plano de la imagen se miden en pixeles, es por eso que se debe introducir un factor de escala. Las siguientes ecuaciones realizan dichos cambios.

$$u = k_u\frac{f}{Z}X + u_0 \tag{2.1.4}$$

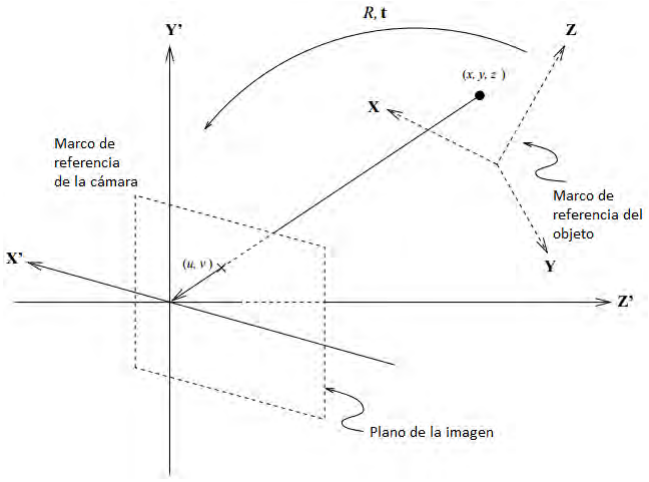


Figura 2.1.3: En esta figura se ilustra la transformación de cuerpo rígido que es necesaria para proyectar un objecto que se encuentra en un marco de referencia diferente.

$$v = k_v \frac{f}{Z} Y + v_0 \tag{2.1.5}$$

La ecuación de proyección queda de la siguiente manera:

$$\begin{bmatrix} \lambda u \\ \lambda v \\ \lambda \end{bmatrix} = \begin{bmatrix} fk_u & 0 & u_0 & 0 \\ 0 & fk_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.1.6}$$

Se puede expresar la distancia focal horizontal y vertical en pixeles con estas ecuaciones $\alpha_u = fk_u$, $\alpha_v = fk_v$.

Otro factor que se debe de tomar en cuenta es que la referencia del mundo no siempre coincidirá con el sistema de referencia de la cámara, es por tal motivo que se debe de introducir una transformación de cuerpo rígido entre los dos marcos de referencia mediante una matriz de rotación y un vector de traslación (Figura 2.1.3).

La ecuación completa de proyección queda de la siguiente manera:

$$\begin{bmatrix} \lambda u \\ \lambda v \\ \lambda \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.1.7}$$

Los valores α_u, α_v, u_0 y v_0 de la primera matriz se conocen como los parámetros intrínsecos de la cámara; mientras que los valores de la matriz de rotación y

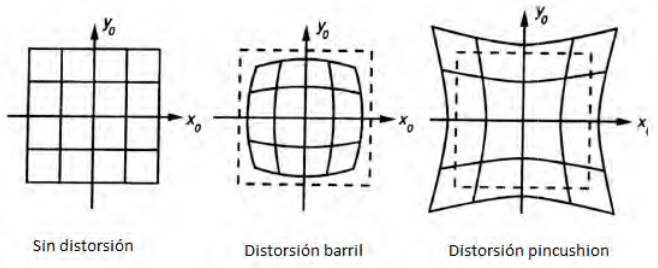


Figura 2.1.4: En esta figura se muestra como el lente de la cámara puede distorsionar el plano de la imagen. Adapatada de [35].

el vector de traslación son los parámetros extrínsecos de la cámara. Estos valores son determinados mediante el proceso de calibración.

2.1.1.2. Distorsión radial

El modelo de proyección de la imagen asume que la cámara obedece un modelo de proyección lineal en donde las líneas rectas en el entorno originan líneas rectas en la imagen. Sin embargo, muchas cámaras con lentes de ángulo amplio tienen cierta distorsión radial que se manifiestan en una curvatura en las líneas rectas.

El modelo estándar de la distorsión radial es una transformación de las coordenadas ideales a las coordenadas reales observadas. Dependiendo del tipo de distorsión radial la coordenadas de la imagen se dispersarán (barril) o concentrarán (pincushion) en el centro. La cantidad de distorsión de las coordenadas de las imágenes observadas es una función no lineal de la distancia radial r. Para la mayoría de los lentes, un simple modelo cuadrático de distorsión produce buenos resultados. La Figura 2.1.4 ilustra los posibles efectos de la distorsión.

$$\begin{bmatrix} u_d \\ v_d \end{bmatrix} = \left(1 + k_1 r^2\right) \begin{bmatrix} u - u_0 \\ v - v_0 \end{bmatrix} + \begin{bmatrix} u_0 \\ v_0 \end{bmatrix} \tag{2.1.8}$$

En donde,

$$r^2 = (u - u_0)^2 + (v - v_0)^2 \tag{2.1.9}$$

Y k_1 es el parámetro de la distorsión radial. Este puede ser estimado mediante la calibración de la cámara.

2.1.2. Calibración de la cámara

La calibración consiste en estimar precisamente los parámetros intrínsecos y extrínsecos del modelo de la cámara. Estos parámetros son los que determinan como los puntos en la escena son mapeados en sus correspondientes puntos en la imagen.

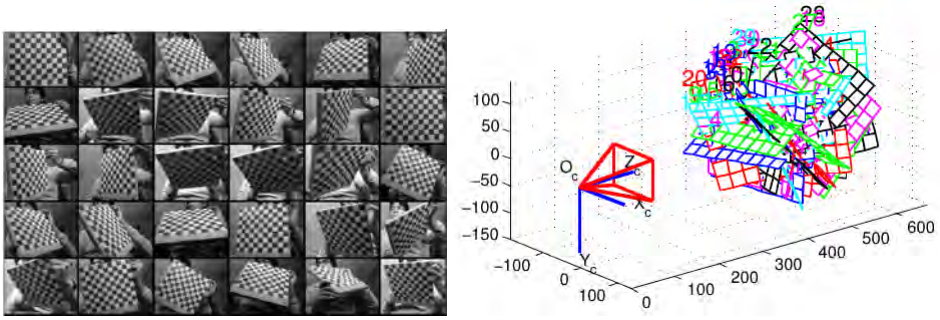


Figura 2.1.5: Calibración de la cámara usando el toolbox de MATLAB: conjunto de imágenes con el patrón de ajedrez (izquierda); posiciones del tablero estimadas por el software de calibración (derecha).

Uno de los métodos más populares es la calibración a partir de una cuadrícula plana, este método es muy sencillo de usar. La técnica requiere que el usuario tome muchas imágenes sobre un patrón conocido –por lo general, una cuadrícula de ajedrez; en distintas posiciones y orientaciones.

Conociendo la posición 2D de las esquinas en el patrón real y las coordenadas en la imagen los parámetros de la cámara son determinados imultáneamente resolviendo una optimización de mínimos cuadrados. La precisión en la calibración incrementa con el número de imágenes usadas; también es importante que las imágenes cubran, en su mayoría, el campo de visión y que el rango de orientaciones sea amplio.

Existen varias herramientas que han implementado este método para calibrar la cámara. Entre los más populares están el toolbox de MATLAB y la implementación en C/C++ de OpenCV [21]. La figura 2.1.5 muestra los resultados de la calibración con el toolbox de MATLAB.

2.2. Detección y descripcción de los puntos de interés en las imágenes

Encontrar la similitud que existe entre dos imágenes que contienen información sobre la misma escena es un problema común en visión por computadora. Existen dos técnicas para encontrar la similitud, la primera se basa en apariencias y la segunda se basa en extraer características sobresalientes. La primera aproximación es más conveniente cuando las imágenes son tomadas en intervalos pequeños; mientras que la segunda es más adecuada cuando existen grandes cambios en el movimiento. Anteriormente, la mayoría de los sistemas de odometría visual usaban técnicas basadas en apariencia; pero ahora, la mayoría se enfoca en buscar características sobresalientes o puntos de interés. Esto se debe a que los sistemas anteriores operaban en espacios reducidos, y lo que se desea en este momento es que los sistemas trabajen en grandes áreas.

2.2.1. Extracción de características

El proceso de detección de características consiste en buscar puntos clave (keypoints) en la imagen –también son llamados puntos de interés o puntos característicos locales. Estos puntos tienen una alta probabilidad de ser encontrados nuevamente en imágenes siguientes. En la Figura 2.2.1 se muestra un ejemplo de puntos clave encontrados en la imagen.

Un punto de interés es un patrón en la imagen que difiere con respecto a sus vecinos inmediatos en términos de intensidad, de color y de textura. Varios algoritmos realizan la tarea de determinar dichos puntos, los algoritmos detectores usualmente se basan en buscar esquinas o manchas en la imagen.

Las propiedades que se desean en un detector son:

- Precisión en la localización: los puntos detectados deben ser localizados precisamente en la imagen como en la escala. La precisión es especialmente importante en el paso de reconstrucción 3D.
- Cantidad de puntos: el número ideal de puntos encontrados depende de la aplicación. Para el caso de la odometría visual es importante una gran cantidad de puntos para tener una buena precisión durante la reconstrucción 3D y la estimación en la orientación de la cámara.
- Invariancia: las características encontradas deben ser invariantes a cambios de iluminación del ambiente, a cambios de escala (zoom) y a cambios de perspectiva.
- Eficiencia computacional: es deseable que el tiempo de ejecución para la detección de características sea eficiente. En robótica, la mayoría de las aplicaciones se deben de ejecutar en tiempo real. Sin embargo, el tiempo de ejecución está relacionado fuertemente con la invariancia deseada: entre mayor sea el nivel de invariancia, mayor es el tiempo de ejecución.
- Robustez: los puntos detectados deben ser robustos a ruido, a efectos de la discretización, a procesos de compresión, a invariancia a cambios fotométricos (iluminación) y cambios geométricos (rotación, escala y distorsión en la perspectiva).

2.2.2. Detectores de esquinas

Moravec inventó uno de los primeros detectores de esquinas [29]. El definió una esquina como un punto en donde hay una gran variación en la intensidad en todas las direcciones. Es decir, una esquina es la intersección entre dos o más bordes, a continuación se describirán los detectores de esquinas más populares en sistemas de navegación visual.

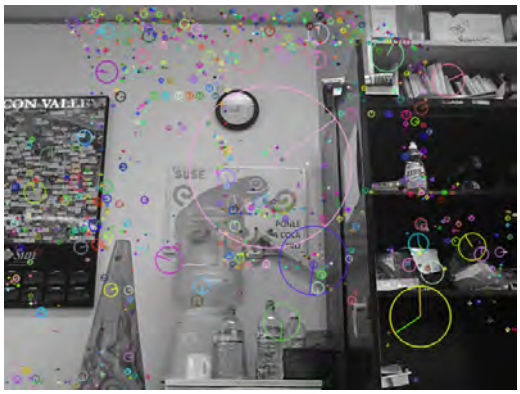


Figura 2.2.1: Ejemplo de la detección de puntos característicos usando el algoritmo SIFT. Este detector encuentra carácterísticas sobresalientes en la imagen a partir de cambio de intensidad; los círculos muestran la posición, orientación y tamaño del punto detectado.

2.2.2.1. Detector Harris

Este detector de esquinas [18] mejoró las ideas planteadas por Movarec; considera las derivadas parciales de la suma de las diferencias al cuadrado (SSD) entre dos regiones cuadradas de la imagen. Esto se hace usando la siguiente ecuación:

$$SSD(x,y) \approx \begin{bmatrix} x & y \end{bmatrix} M \begin{bmatrix} x \\ y \end{bmatrix}, \; donde \; M = \begin{bmatrix} \sum\sum I_x^2 & \sum\sum I_x I_y \\ \sum\sum I_x I_y & \sum\sum I_y^2 \end{bmatrix} \tag{2.2.1}$$

En la ecuación I_xy $I_y$son gradientes de la imagen en dirección horizontal y vertical, respectivamente. M es una matriz simétrica y puede ser representada como:

$$M = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R \tag{2.2.2}$$

Los valores λ_1 y $\lambda_2$son los eigenvalores de la matriz. Dada la función de autocorrelación $C = \lambda_1\lambda_2 - k\left(\lambda_1 + \lambda_2\right)^2 = det\left(M\right) - k\,traza^2\left(M\right)$, el punto en la imagen es seleccionado como esquina si C es positivo. El valor de k se determina empíricamente, pero suele estar entre los valores 0.04 y 0.15. Dadas Las magnitudes de los eigenvalores se pueden hacer las siguientes inferencias:

- Si λ_1y λ_2 son pequeños, la SSD es casi constante en todas las direcciones (se está en presencia de una región uniforme).
- Si $\lambda_1 >> \lambda_2$ o $\lambda_1 << \lambda_2$, la SSD sólo varía en una dirección (se encuentra sobre un borde).

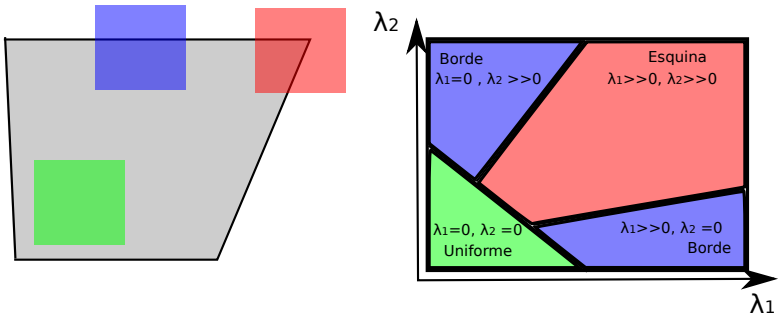


Figura 2.2.2: En esta figura se muestra la relación que existe entre los valores de λ_1 y λ_2, y la región de la imagen analizada. Cómo se puede observar, los valores de λ_1 y λ_2 son altos cuando el área analizada es una esquina.

- Si λ_1y $\lambda_2$son grandes, la SSD tiene variaciones en todas las direcciones (se está en presencia de una esquina).

Finalmente, se hace una eliminación de los puntos que no son máximos en una región, de esa manera, sólo los puntos con mayor respuesta permanecen. La Figura 2.2.2 ilustra como se determina si un punto es una esquina a partir de los valores λ_1 y λ_2.

2.2.2.2. Detector Shi-Tomasi

Este detector [41] se basa en el detector Harris. Los autores demostraron que la operación $min\,(\lambda_1, \lambda_2)$ es mucho más efectiva para determinar las esquinas. Da mejores resultados cuando suceden transformaciones afines.

2.2.2.3. Detector FAST

FAST (Features from Accelerated Segmented Test) [36] es un detector mucho más rápido que los otros métodos mencionados, es 20 a 30 veces más rápido que el detector Harris. El detector explícitamente busca esquinas; decide si un pixel es el centro de una esquina evaluando un círculo de pixeles que rodea al punto: si una sección continua grande del círculo es más oscura o brillante que el centro, el punto se clasifica como esquina. Los autores entrenaron un árbol de decisiones para que distinguiera de todos los pixeles de la imagen cuál es una esquina y cual no, con el menor número de comparaciones posibles. Este método no es robusto a altos niveles de ruido, pero debido a su velocidad es una de las técnicas más usadas en aplicaciones de tiempo real.

2.2.3. Detectores de manchas

Una mancha es un patrón en la imagen que difiere de sus vecinos inmediatos en términos de intensidad, color y textura. No es un borde ni una esquina. La

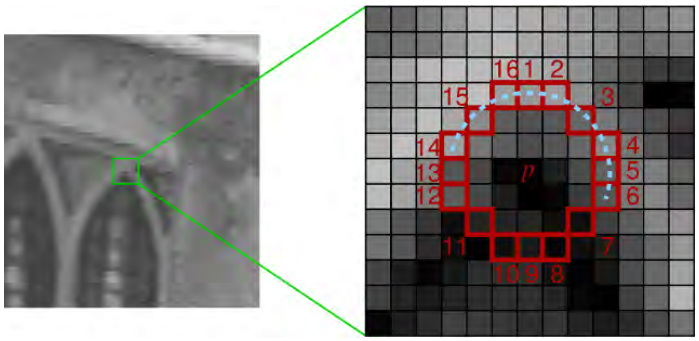


Figura 2.2.3: Representación gráfica de la evaluación que realiza FAST para determinar si el pixel p es una esquina. Adaptada de [36].

precisión en la localización de una mancha es menor que la de una esquina, pero la mancha está mejor definida en escala y forma A continuación se presentan algoritmos populares para detectar manchas.

2.2.3.1. Detector SIFT (DoG)

SIFT (Scale Invariant Feature Transform [26]) es un método para detectar y emparejar puntos de interés. Las características extraídas con SIFT son muy distintivas y pueden ser emparejadas aunque haya cambios muy bruscos en iluminación, rotación y escala.

Una de las ventajas de este algoritmo es que además de encontrar el punto de interés también genera un descriptor. El descriptor se calcula a partir de la información extraída de la región en donde se encuentra el punto de interés, de tal manera que el punto ahora tiene una firma particular que lo distingue del resto de los puntos de interés. El descriptor es lo que hace que SIFT sea robusto a cambios de rotación, iluminación, escala y perspectiva. Los descriptores serán estudiados en el siguiente apartado.

El primer paso del algoritmo es determinar los puntos de interés, esto lo hace generando diferencias de Gaussianas (DoG). Primero, se pasa un filtro Gaussiano (pasa bajos) a la imagen con escalas distintas (sigmas distintas), después se obtienen las diferencias restando las imágenes suavizadas que sean sucesivas. El proceso se repite submuestreando la imagen inicial en un factor de dos hasta tener las diferencias de la imagen original y tres submuestreos.

El segundo paso es la selección de los puntos. Cada pixel en las diferencias de Gaussianas es comparado con sus ocho vecinos en la misma escala, y además con sus nueve vecinos en escalas adyacentes. Si el pixel es un máximo o mínimo local, entonces se selecciona como candidato.

El último paso consiste en refinar la localización, tanto en espacio como en escala mediante un proceso de interpolación usando la información cercana a los puntos. Finalmente, los puntos con contraste bajo o que se encuentran sobre bordes o que sean esquinas son removidos debido a su poca distinción e

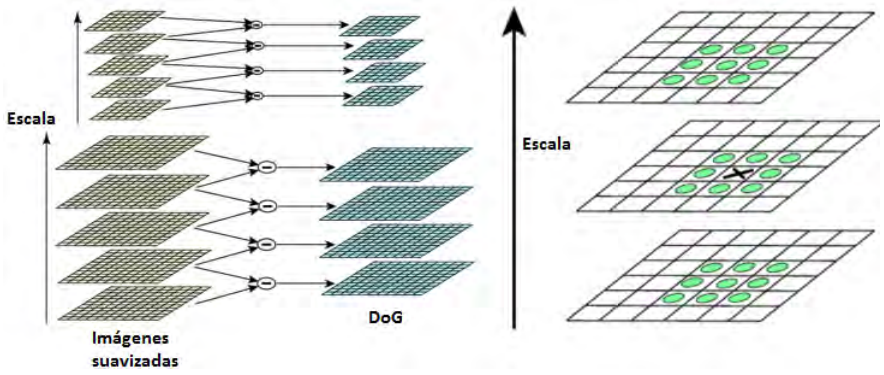


Figura 2.2.4: Representación gráfica de la extracción de características usando SIFT. Se observa como se obtienen las diferencias de Gaussianas y como se escoge el punto con mayor respuesta entre sus vecinos.

inestabilidad.

2.2.3.2. Detector SURF (Fast-Hessian)

SURF (Speed Up Robust Features [7]) es un detector fuertemente influenciado por SIFT y es mucho más rápido. Usa wavelets Haar para aproximar las diferencias de Gaussianas basándose en el determinante del Hessiano.

$$H(x,y,\sigma) = \begin{bmatrix} \frac{\delta^2}{\delta x^2} G(\sigma) * I(x,y) & \frac{\delta}{\delta x}\frac{\delta}{\delta y} G(\sigma) * I(x,y) \\ \frac{\delta}{\delta x}\frac{\delta}{\delta y} G(\sigma) * I(x,y) & \frac{\delta^2}{\delta y^2} G(\sigma) * I(x,y) \end{bmatrix} \tag{2.2.3}$$

En lugar de realizar una convolución con la segunda derivada de la Gaussiana, esta se aproxima usando filtros sencillos. Estos filtros se pueden ver en la Figura 2.2.5. Se usa la siguiente función de respuesta para determinar si un pixel es un punto característico.

$$c(x,y,\sigma) = D_{xx}(\sigma)D_{yy}(\sigma) - (0.9D_{xy}(\sigma))^2 \approx det\left[H(x,y,\sigma)\right] \tag{2.2.4}$$

Aquí D_{xx}, D_{yy} y D_{xy} son los resultados de aplicar la convolución con los filtros mencionados. Si c es positivo se considera como un punto de interés, en caso contrario se remueve. Al igual que SIFT, se comparan los pixeles con sus ocho vecinos de imagen, y con sus vecinos en escalas adyacentes para determinar a los máximos y mínimos. También realiza la interpolación para refinar la localización del punto.

2.2.3.3. Detector CenSurE

CenSurE (Center Surround Extrema [5]) es un algoritmo que intenta aproximar el método de diferencias de Gaussianas usando filtros binivel con valores de

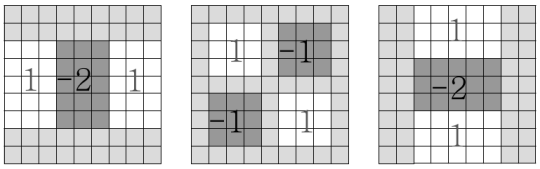


Figura 2.2.5: Filtros usados para aproximar la convolución en SURF.

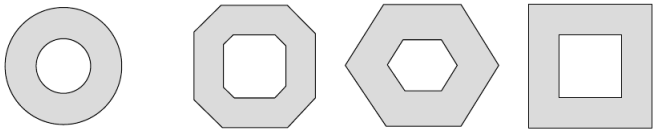


Figura 2.2.6: Filtros bi-nivel usados en CenSurE para aproximar el laplaciano. El círculo es la figura ideal que mejor aproxima al laplaciano, pero su cálculo es muy costoso; es más eficiente el cálculo de las imágenes a la derecha, pero se sacrifica un poco de precisión.

-1 y 1. La forma del filtro que más se parece a un laplaciano de Gaussiana es un círculo, pero debido a que es computacionalmente costoso calcularlo se usan distintas formas como octágono, hexágonos y cuadrados (Figura 2.2.6). Estas cada vez menos simétricas son más fáciles de calcular, pero sus resultados son menos robustos. La forma en que CenSurE obtiene invariancia a la escala es aplicando el filtro sobre la imagen con distintos tamaños, en vez de submuestrear la imagen. La ventaja de usar los filtros con distintos tamaños en vez de submuestrear es que no se pierde precisión en la localización del punto de interés, ya que no es necesaria ninguna interpolación entre escalas porque siempre se usa la imagen original. La forma en que se seleccionan y se remueven los puntos de interés es similar a SIFT y a SURF.

2.2.4. Evaluación

Los algoritmos detectores de esquinas se caracterizan por ser muy eficientes tanto en velocidad como en el número de puntos que encuentran, además de que tienen muy buena precisión en la localización; pero tienen el problema de que muchos de los puntos que encuentran en una toma pueden no ser encontrados de nuevo en otra toma (repetibilidad) –esto sucede principalmente cuando existen transformaciones afín. Específicamente, los detectores de esquinas no tienen problema cuando suceden movimientos de rotación porque las esquinas no se ven afectadas por este; sin embargo su problema principal son los cambios de escala, una esquina puede dejar de ser una y convertirse en un borde o desaparecer en diferentes escalas.

Por otro lado, los algoritmos basados en manchas tienen velocidades de ejecución más lentas, pero encuentran puntos que el algoritmo puede volver a encontrar de nuevo con una alta probabilidad a pesar de cambios de perspectiva

	Esquinas	Manchas	Inv. Rotación	Inv. Escala	Inv. transf. Afín	Repetibilidad	Localización	Robustez	Eficiencia
Harris	✔		✔			☆☆☆	☆☆☆	☆☆	☆☆
Shi-Tomasi	✔		✔			☆☆☆	☆☆☆	☆☆	☆☆
FAST	✔		✔	✔		☆☆	☆☆	☆☆	☆☆☆☆
SIFT		✔	✔	✔	✔	☆☆☆	☆☆	☆☆☆	☆
SURF		✔	✔	✔	✔	☆☆☆	☆☆	☆☆	☆☆
CenSurE		✔	✔	✔	✔	☆☆☆	☆☆	☆☆☆	☆☆☆

Figura 2.2.7: Se muestran y se comparan las características de los detectores de puntos de interés. Tabla adaptada de [39]

y de iluminación. Las manchas no dejan de ser manchas si existen rotaciones; también las manchas son fácilmente localizadas en distintas escalas. El problema principal de las manchas reside en su localización, una mancha no es sólamente un punto sino un conjunto; es por tal motivo que no se tiene mucha certeza de su posición.

Adicionalmente, se debe de tomar en cuenta el entorno de trabajo, porque algunos detectores de manchas rechazan totalmente a las esquinas –y la mayoría de los entornos urbanos son ricos en esquinas. En aplicaciones de odometría visual o SLAM visual se han usado detectores de puntos basados en esquinas (Harris, Shi-Tomasi y FAST) y basados en manchas (SIFT, SURF y CenSurE). En la tabla 2.2.7 se muestra una comparación de características para los detectores presentados.

2.2.5. Descriptores de características

La descripción de características consiste en extraer información sobre la región que rodea el punto de interés de manera simplificada y así poder compararla fácilmente con otros descriptores.

Los descriptores más sencillos se basan en la apariencia, en la intensidad que tienen los pixeles en un cuadro que rodea al punto. En este caso, se usan ciertas operaciones que miden la similitud. A continuación se explicarán los tres métodos más populares para medir la similitud. Estos métodos se basan en describir área. Supongamos que queremos comparar un cuadro de la imagen I_1 con dimensiones m x n con centro en (u, v) con otro cuadro del mismo tamaño centrado en (u',v') en la imagen I_2. Se asume que son valores enteros impares. La similitud se calcula entre niveles de intensidad de los cuadros. Estos son algunos de los criterios más usados:

- Suma de las diferencias absolutas (SAD, por sus siglas en inglés):

$$SAD = \sum_{k=-a}^{a} \sum_{l=-b}^{b} \left| I_1(u+k, v+l) - I_2(u'+k, v'+l) \right| \tag{2.2.5}$$

- Suma de las diferencias al cuadrado (SSD, por sus siglas en inglés):

$$SSD = \sum_{k=-a}^{a} \sum_{l=-b}^{b} \left[I_1(u+k, v+l) - I_2(u'+k, v'+l) \right]^2 \tag{2.2.6}$$

- Correlación cruzada normalizada (NCC, por sus siglas en inglés):

$$NCC = \frac{\sum_{k=-a}^{a} \sum_{l=-b}^{b} \left[I_1(u+k, v+l) - \mu_1) \right] \left[I_2(u'+k, v'+l) - \mu_2) \right]}{\sqrt[2]{\sum_{k=-a}^{a} \sum_{l=-b}^{b} \left[I_1(u+k, v+l) - \mu_1) \right]^2 \sum_{k=-a}^{a} \sum_{l=-b}^{b} \left[I_2(u'+k, v'+l) - \mu_2) \right]^2}} \tag{2.2.7}$$

$$\mu_1 = \frac{1}{mn} \sum_{k=-a}^{a} \sum_{l=-b}^{b} I_1(u+k, v+l) \tag{2.2.8}$$

$$\mu_2 = \frac{1}{mn} \sum_{k=-a}^{a} \sum_{l=-b}^{b} I_2(u'+k, v'+l) \tag{2.2.9}$$

La SAD es la medida más sencilla. Esta se calcula al restar los pixeles entre la imagen de referencia I_1 y la imagen objetivo I_2 seguido de la suma del absoluto de las diferencias dentro del cuadro. La SSD tiene una mayor complejidad computacional comparada con SAD, porque tiene más operaciones. Si las imágenes I_1 y I_2 se emparejan perfectamente, el resultado de aplicar SAD o SSD será cero.

La NCC es aún más complicada dado que incluye más operaciones; sin embargo, provee mayor distinción y es invariante a cambios afines de intensidad. Finalmente, los valores de la NCC se encuentran entre -1 y 1 en donde 1 corresponde a la máxima similitud entre dos cuadros de imágenes.

En la mayoría de los casos, la apariencia local del punto no es buen descriptor porque la apariencia cambiará con cambios en la orientación, en la escala y en los puntos de vista. Por tal motivo, estas técnicas se usan cuando las imágenes fueron tomadas a posiciones cercanas entre ellas.

Existe otro tipo de descriptores en los que una vez que se han identificado los puntos de interés, se calcula un vector que contiene información como la textura o la intensidad de la vecindad de cada punto. SIFT fue el primer método de caracterización vectorial para puntos de interés reportado en la literatura.

2.2.5.1. Descriptor SIFT

Después de obtener los puntos característicos usando DoG se calcula la orientación del punto para hacerlo invariante a la rotación. Para esto se calcula un histograma de orientación de gradientes usando la vecindad del punto. El histograma es llenado con las orientaciones ponderadas por la magnitud del gradiente del pixel correspondiente.

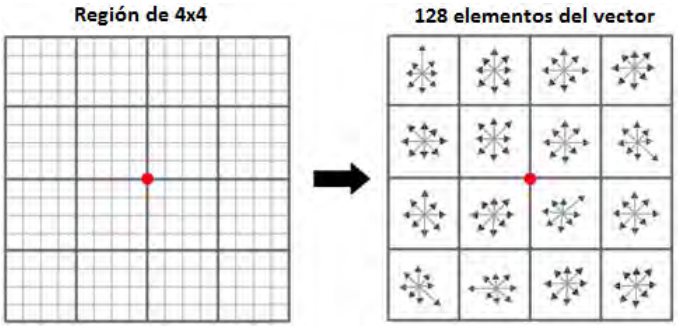


Figura 2.2.8: Forma en la que SIFT obtiene los valores de su vector de descripción. Se crean histogramas de orientaciones ponderadas sobre la vecindad del punto de interés. La orientación del punto y los valores de estos histogramas conforman al vector de descripción.

Una vez que el histograma fue llenado, se busca la orientación con el pico mayor, y esa es asignada al punto. En caso de que exista más de un pico, se crean nuevos puntos clave y se les asigna la orientación de cada pico.

El último paso consiste en calcular los valores que tendrá el vector de descripción. El descriptor se basa en histogramas de orientación de gradientes. Para obtener invariancia a la rotación los puntos se giran relativamente a partir de la orientación del punto de interés. La vecindad alrededor del punto se divide en regiones de 4 x 4, y un histograma de orientaciones con ocho divisiones se calcula por cada región. Finalmente, el descriptor se construye almacenando todos los valores de los histogramas uno tras otro en un vector. El tamaño final del vector es de 128 elementos (4 x 4 x 8). El vector, por último, es normalizado para lograr invariancia a la iluminación.

2.2.5.2. Descriptor SURF

De manera similar a SIFT, SURF primero asigna una orientación a cada punto de interés. Esto lo hace convolucionando dos wavelets Haar a una región circular alrededor del punto. El tamaño de los wavelets y de la región depende de la escala en la que se encontró el punto. Las respuestas al filtro, ponderadas con una Gaussiana alrededor del punto, son representadas como vectores en un espacio bidimensional y se suman con una ventana angular rotacional. El vector resultante con el mayor tamaño determina la orientación del punto. Una región cuadrada alrededor del punto se divide en 4 x 4 subregiones y el siguiente vector característico se calcula:

$$\left[\sum d_x,\ \sum d_y,\ \sum \left|d_x\right|,\ \sum \left|d_y\right|\right]$$

d_xy $d_y$son las respuestas al filtro a los wavelets Haar y todas las sumas son sobre puntos regularmente espaciados en su subregión. Se concatenan todos los

vectores característicos de cada subregión para formar el vector descriptor con 64 elementos (4 x 4 x 4).

2.2.5.3. Descriptor CenSurE

CenSurE no define un descriptor propio, más bien modifica el descriptor de SURF reduciendo el efecto de los bordes mediante el uso de traslapes entre cuadros de subregiones. Además, mejora el tiempo de ejecución del algortimo. A este descriptor modificado lo nombraron MU-SURF.

2.2.6. Descriptores binarios

En los últimos años ha habido un interés por hacer descriptores que se calculen más rápido, que sean más compactos, y que además sigan siendo robustos a cambios ocasionados por rotación, escala o ruido. Los descriptores binarios han sido una solución a este problema, estos vectores difieren de los que plantea SIFT y SURF en los siguientes puntos:

- Los vectores ya no almacenan valores de punto flotante, sólo almacenan un cero o un uno en cada elemento. Esto reduce la memoria necesaria para guardar al descriptor.
- El descriptor binario se genera al comparar la intensidad entre varios pares de pixeles escogidos alrededor del punto de interés, después de haber aplicado un filtro Gaussiano para reducir el ruido. Estas operaciones simples se ejecutan en menor tiempo que las operaciones realizadas por SIFT o SURF para calcular su descriptor.
- Encontrar la similitud entre vectores -proceso que se usa para el emparejamiento de puntos- es mucho más rápido. Los vectores de SIFT y SURF usan la norma L_2 para medir la distancia entre dos vectores y buscan el vector que arroje la distancia menor; en el caso de los descriptores binarios se usa la distancia de Hamming. Esta distancia se calcula usando una operación XOR, que una computadora realiza de manera eficiente.

El primer descriptor binario que usó los conceptos planteados fue BRIEF [8], este obtenia las diferencias de intensidades entre 512 pares de pixeles, las posiciones de los pixeles eran previamente seleccionadas al azar usando una distribución Gaussiana alrededor del punto. Este descriptor no era invariante a escala ni a rotación, el detector tenía que proveer esa información. Posteriormente, se propuso un algoritmo llamado ORB (Oriented Fast and Rotated BRIEF [37]), su descriptor se basaba en BRIEF y le agregaba invariancia en la escala y en la rotación, además era más robusto al ruido. De forma similar, Leutenegger et al. [24] propuso BRISK (Binary Robust Invariant Scalable Keypoints), su descriptor era igual invariante a la rotación y a la escala. BRISK construye el descriptor usando un patrón circular de muestreo con un número limitado de puntos igualmente espaciados sobre cículos concéntricos (Figura 2.2.9). Los pares de puntos son divididos a partir de la distancia en pares pequeños y largos.

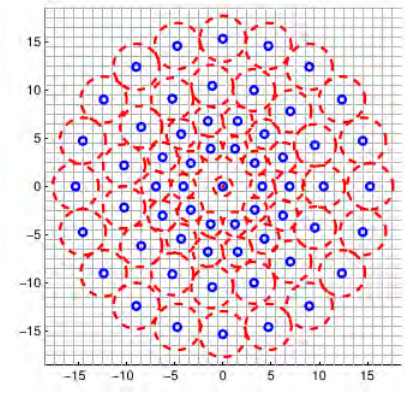


Figura 2.2.9: Patrón usado en el descriptor BRISK. Se escogen parejas de círculos a partir de un patrón definido. Estas combinaciones dan información sobre la escala y la orientación del punto. Adaptada de [24].

Los pares largos se usan para estimar la dirección del punto de interés mientras que los pequeños se usan para construir el descriptor después de haber rotado el patrón. Recientemente, se propuso FREAK [6], este descriptor supera a sus sucesores y tiene un desempeño similar al descriptor de SIFT.

2.2.6.1. Descriptor FREAK

FREAK (Fast Retina Keypoint) es un descriptor que usa las ideas propuestas en BRISK y además se inspira en el sistema visual humano, principalmente en la topografía que tienen los conos y los bastones en la retina del ojo. Este descriptor usa un patrón similar a BRISK, es una patrón circular basado en los campos receptivos de la retina humana que imita las densidades de los receptores al tener mayor representación en el área central (Figura 2.2.10). Esta concentración decrese exponencialmente cuando se aleja del centro, y además existe un traslape en los campos receptivos. Un filtro Gaussiano con diferentes desviaciones estándar por cada nivel es aplicado para reducir el ruido.

El descriptor se forma usando diferencias entre pares de puntos que comparten el mismo filtro, si la resta es positiva se asigna un valor de 1, en caso de ser igual o menor a cero se deja el valor de cero. Los autores diseñaron el descriptor considerando pares cuyas restas puedan representar mejor al punto, para ello consideran las parejas con varianza más alta. Experimentalmente se encontraron 512 pares que dan la mejor representación al punto, agregar más pares no mejora el desempeño del descriptor. Finalmente, se determina la orientación del punto usando el mismo concepto de BRISK al usar pares con distancias largas sobre una configuración simétrica.

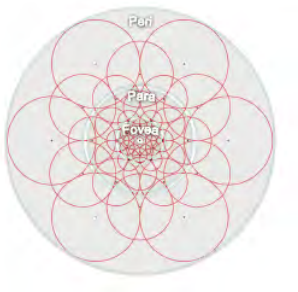


Figura 2.2.10: Patrón basado en de BRISK y en la retina humana usado en FREAK.

2.3. Emparejamiento de los puntos de interés

El emparejamiento consiste en encontrar las relaciones entre puntos que representan un mismo objeto en la escena y que se encuentran en distintas imágenes. Una vez que se tienen vectores de descripción asociados a los puntos de interés; se compara cada vector de la imagen contra los vectores de la otra imagen. Para esta tarea se pueden usar técnicas de aprendizaje de máquina como k-nearest neighbors. El método más simple de k-nearest neighbors es el de Fuerza Bruta, éste toma a los vectores del primer conjunto y los compara con vectores del segundo conjunto, y empareja a aquellos con la mayor similitud. La forma en que se comparan es obteniendo la norma entre los dos vectores; esta norma puede ser la distancia euclidiana para vectores de tipo flotante generados por SIFT o SURF, o la distancia de Hamming para vectores binarios generados por BRISK o FREAK. Este método para emparejar devuelve buenos resultados pero es muy costoso computacionalmente, por tal motivo en ocasiones se opta por usar métodos aproximados que sacrifican un poco de precisión para aumentar la rapidez. Estos métodos se conocen como approximate-nearest neighbors, a continuación se listan algunos de estos emparejadores:

- k-d Tree
- Best bin first
- Randomized k-d Tree
- Hierarchical k-mean tree
- Local sensitive hashing
- Hierarchical clustering tree

La mayoría de estos métodos se encuentran integrados en la biblioteca de software FLANN (Fast Library for Approximate Nearest Neighbors [32]).

Figura 2.3.1: Se muestra el emparejamiento de dos imágenes usando Fuerza Bruta, a pesar de que este método resulta ser muy bueno, en aplicaciones como la odometría visual su desempeño es muy bajo. Para mejorar el emparejamiento se deben de agregar más restricciones durante la búsqueda.

2.3.1. Emparejamiento robusto con RANSAC

El emparejamiento de puntos esta usualmente contaminado por asociaciones erróneas, como se muestra en la Figura 2.3.1. Posibles causas de estos errores son el ruido, oclusiones, y cambios de perspectiva y de iluminación; el modelo matemático del detector o del descriptor no suelen tomar en cuenta estas afectaciones. Para que la estimación de la posición de la cámara sea precisa, es importante que las asociaciones erróneas sean removidas. La estimación puede resultar muy mala debido a estos errores. Éste es un paso importante en un sistema de odometría visual.

Una posible solución consiste en tomar en cuenta el modelo y las restricciones geométricas que impone. Uno de los métodos más populares es RANSAC (Random Sample Consensus [11]), éste es un algoritmo para estimar robustamente los parámetros de un modelo a partir de un conjunto de datos que presentan cierta cantidad de información errónea. La idea es calcular una hipótesis del modelo usando información seleccionada al azar y posteriormente verificar la hipótesis en el resto de la información. La hipótesis que muestre el mayor consenso con el resto de la información es seleccionada como solución.

Para el problema del emparejamiento robusto se necesitan conocer las asociaciones hechas por alguno de las técnicas de aprendizaje de máquina mencionadas anteriormente, y tener conocimiento sobre las restricciones geométricas entre las vistas. Estas restricciones, que serán estudiadas en la sección siguiente, son la geometría epipolar y el error de reproyección de puntos del mundo en las imágenes.

RANSAC es un método iterativo y no determinístico ya que muestra diferentes soluciones en distintas corridas; sin embargo, la solución tiende a ser estable cuando el número de iteraciones crece. El método no necesita revisar todas

Algoritmo 2.1 RANSAC

1. Estado inicial: A es un conjunto de N correspondencias de puntos
2. Repetir hasta que el número máximo de iteraciones se alcance
   - 2.1 Al azar seleccionar una muestra de s puntos de A
   - 2.2 Ajustar un modelo con s
   - 2.3 Evaluar los puntos restantes con el modelo obtenido y medir el error.
   - 2.4 Construir el conjunto de puntos válidos (contar el número de puntos cuya distancia del modelo es $< d$)
   - 2.5 Guardar los puntos válidos
3. El conjunto con el mayor número de puntos válidos es escogido como solución
4. Se estima el modelo usando todos los valores válidos

las combinaciones posibles sino sólo un subconjunto de ellas si es que tenemos una estimación aproximada sobre el porcentaje de emparejamientos correctos en todo el conjunto de asociaciones. Se puede calcular el número de iteraciones con la siguiente fórmula:

$$N = \frac{log(1-p)}{log(1-(1-\varepsilon)^s)} \tag{2.3.1}$$

aquí s es el número de puntos con los cuales el modelo puede ser instanciado, ε es el porcentaje de datos erróneos en la información, y p es la probabilidad requerida para tener éxito. La Figura 2.3.2 muestra el resultado de aplicar RANSAC a los emparejamientos mostrados en la Figura 2.3.1, como se puede ver se elimanron en su mayoría las asociaciones erroneas.

2.4. Estimación del movimiento

El proceso de estimación de la pose es el paso crucial en la odometría visual, con ella podremos obtener los parámetros extrínsecos de la cámara. Esta estimación se realiza entre frames vecinos, se concatenan todas las estimaciones para obtener la trayectoria del agente. Para el caso de odometría monocular existen dos formas para calcular la transformación T_k entre dos imagánes I_{k-1} y I_k. Estas se diferencian a partir de las características emparejadas en instantes contiguos.

- Estimación con características 2D a 2D: En este caso, las características son las posiciones de los puntos en las imágenes entre I_{k-1}y I_k.

Figura 2.3.2: Emparejamiento con Fuerza Bruta + RANSAC. En esta imagen se muestra como solo se conservan los emparejamientos que cumplen con la restricción epipolar. Sin embargo, aún quedan emparejamientos falsos (emparejamiento entre un punto de la botella y un punto del cono en la parte inferior) pero estos errores ya son mínimos.

- Estimación con características 3D a 2D: En este caso, se conocen las posiciones 3D de puntos en el mundo en un instante $k-1$ y se conocen sus proyecciones en una imagen 2D en el instante k.

2.4.1. Estimación con características 2D a 2D

La relación que existe entre dos vistas que observan la misma escena en posiciones distintas puede ser descrita usando la geometría epipolar [19]. La geometría epipolar es independiente de la estructura de la escena, y sólo depende de los parámetros intrínsecos y de la posición relativa de las vistas. En esencia, esta geometría combina dos modelos pinhole (uno por cada cámara); y la intersección de los planos de la imagen con los puntos proyectados. La línea base, que es la distancia entre las vistas, funciona como eje. En la Figura 2.4.1 se ilustran los conceptos.

En la Figura 2.4.1 observamos un punto que es proyectado en los dos planos. La relación que existe entre el punto, sus proyecciones y los centros de las cámaras es el plano epipolar. Estos componentes son coplanares. La terminología adecuada para los componentes mostrados es la siguiente:

- El epipolo es el punto de intersección entre la línea que une los centros de las cámaras (la línea base) y el plano de la imagen.
- Un plano epipolar es un plano que contiene a la línea base y que se forma con el haz que se proyecta en las vistas.
- Una línea epipolar es la intersección entre un plano epipolar y el plano de la imagen. Todas las líneas epipolares se intersecan en el epipolo.

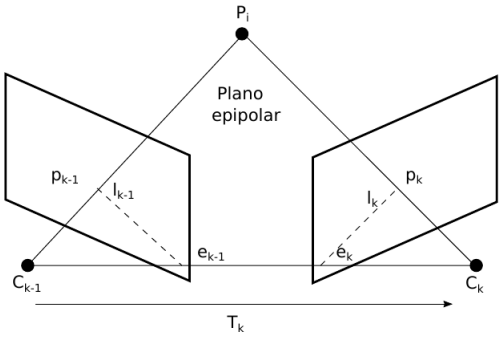


Figura 2.4.1: Geometría epipolar: se muestran las relaciones que existen entre un punto del mundo P_i, su proyección en los planos p_{k-1}, p_k, y la posición del centro de las cámaras C_{k-1}, C_k.

Matemáticamente, se puede expresar la relación entre un punto en la imagen y su línea epipolar correspondiente. La matriz fundamental es la representación algebraica de la geometría epipolar. Esta matriz de 3 x 3 mapea las coordenadas de la línea epipolar a las coordenadas del punto de la imagen.

$$\begin{bmatrix} l_x \\ l_y \\ l_z \end{bmatrix} = F \begin{bmatrix} P_x \\ P_y \\ 1 \end{bmatrix} \tag{2.4.1}$$

También, si se tiene un punto en la escena X y su proyección en la primera imagen es x y su proyección en la segunda imagen es x' se cumple la siguiente condición:

$$xFx' = 0 \tag{2.4.2}$$

Para calcular la matriz fundamental se necesitan encontrar correspondencias de puntos entre las dos vistas. A partir de estas correspondencias existen varios métodos que estiman la matriz F. El algoritmo de los 8 puntos es el más sencillo, como su nombre lo indica necesita de ocho correspondencias, y de la construcción y resolución de un sistema de ecuaciones lineales. Otras soluciones usan métodos iterativos para minimizar criterios algebraicos o ciertas distancias geométricas (método Gold Standard).

Hasta este punto conocemos que la matriz fundamental relaciona algebráicamente a los puntos entre dos vistas; sin embargo, para conocer la traslación unitaria y la rotación relativa entre las imágenes se debe obtener la matriz esencial. La matriz esencial es una especialización de la matriz fundamental, es el caso de la normalización en las coordenadas de la imagen. La matriz fundamental, por definición, tiene siete grados de libertad mientras que la matriz esencial sólo tiene cinco. Estos cinco grados de libertad son por los tres valores de la rotación y los tres valores de la traslación; pero existe una ambigüedad en la

escala, es decir, la magnitud de la traslación guardada en la matriz puede tener distintas escalas, esto elimina un grado de libertad.

La matriz esencial se puede calcular a través de la matriz fundamental usando la siguiente ecuación:

$$E = K^T F K \tag{2.4.3}$$

Donde K es la matriz de parámetros intrínsecos de la cámara y F la matriz fundamental. También se puede obtener el algoritmo usando el método de los ocho puntos, o aún mejor el algoritmo de los cinco puntos propuesto por Níster, que es más rápido y menos susceptible al ruido.

La forma de la matriz esencial en términos de rotación y traslación es la siguiente:

$$E \approx t_k^- R_k \tag{2.4.4}$$

donde $t_k = \begin{bmatrix} t_x & t_y & t_z \end{bmatrix}^T$ y $t_k^- = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}$.

Una vez calculada una estimación de la matriz esencial es necesario extraer al vector de traslación y a la matriz de rotación. En general, se pueden extraer cuatro soluciones de una sola matriz esencial usando descomposición en valores singulares, para escoger que solución cumple con el movimiento se usa un punto triangulado para eliminar la ambigüedad; la solución es aquella que tenga el punto enfrente de las dos vistas.

Un problema con el uso de la matriz esencial es que no se tiene el valor de la escala absoluta. Sin embargo, se puede tener un estimado a partir de las escalas relativas entre vistas contiguas.

2.4.2. Triangulación

La triangulación consiste en determinar la posición de un punto en tres dimensiones dadas las posiciones en dos imágenes con cámaras calibradas y con pose conocida. Este proceso requiere la intersección de dos rayos conocidos en el espacio. El problema es muy sencillo cuando no hay ruido presente; se puede resolver con cálculos geométricos muy simples. Pero cuando hay ruido, errores debido a la calibración de la cámara, o incertidumbre en la localización del punto de interés; los rayos por lo general nunca se intersecarán. Una solución sencilla o ingenua a este problema es buscar el punto medio de la recta con la distancia mínima entre los rayos, este punto se considera como el punto de intersección. Este método se ilustra en la Figura 2.4.2. Sin embargo, este método no encuentra resultados óptimos y es poco útil en términos de reconstrucción proyectiva.

Existen otros métodos más robustos para determinar el punto 3D en la escena. En el artículo de Hartley and Sturm [20] se hace una evaluación de algunos métodos de triangulación como:

- Polinomial

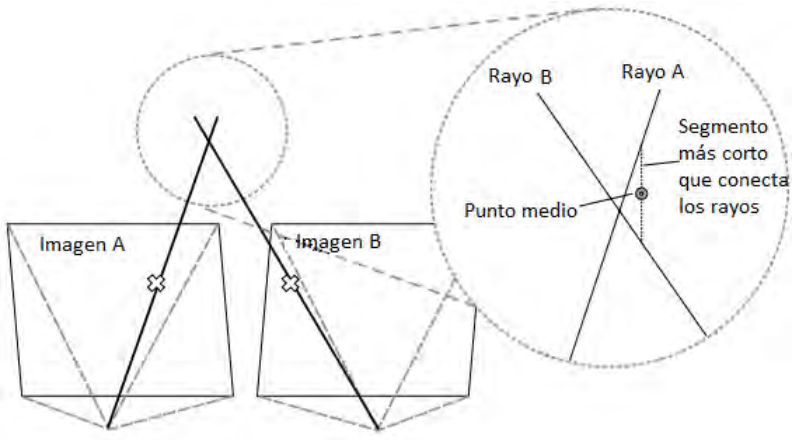


Figura 2.4.2: Triangulación: Método del punto medio para estimar un punto 3D del mundo.

- Linear-Eigen
- Linear-LS
- Iterative-Eigen
- Iterative-LS

Los autores recomiendan usar el método Polinomial, el método que ellos mismos desarrollaron. Esta técnica no iterativa busca corregir las correspondicas entre las dos imágenes x y x' que cumplan con la restricción epipolar (ecuación 2.4.2). Se calculan las nuevas correspondecias $\widehat{x}$ y $\widehat{x}'$ que minimizen la siguiente función de costo sujeta a la restricción epipolar:

$$C(x, x') = d(x, \widehat{x})^2 + d(x', \widehat{x}')^2 \tag{2.4.5}$$

donde $d(x, y)$ es la distancia euclidiana entre los puntos. Con las nuevas correspondencias se pueden calcular los puntos 3D con simples métodos de triangulación lineal. La triangulación Polinomial encuentra la solución óptima cuando el ruido sigue un modelo gaussiano y es el método idoneo para la reconstrucción projectiva; sin embargo, un inconveniente que tiene es su tiempo de ejecución, es el método más lento de todos. Una alternativa es el Iterative-LS, método que tiene mejor tiempo de ejecución y casi no tiene problemas con la reconstrucción projectiva (no se recomienda usarlo con ese propósito). Uno de sus principales inconvenientes es la falta de convergencia durante la triangulación de algunos puntos.

2.4.3. Estimación con características 3D a 2D

Es posible determinar la transformación de cuerpo rígido [R|T] entre imágenes a partir de geometría tridimensional conocida. A este problema se le conoce como *Perspectiva desde n Puntos (PnP)*, el cual consiste en determinar la posición y orientación de una cámara dados sus parámetros intrínsecos, y un conjunto de n correspondencias entre puntos 3D y sus proyecciones 2D. A este método también se le conoce como calibración extrínseca.

Existen tanto métodos iterativos como no iterativos. Los métodos iterativos consisten en minimizar un criterio que usualmente es una función de reproyección de los puntos sobre la imagen, estos pueden alcanzar una gran precisión si convergen adecuadamente; pero tienen una complejidad computacional más elevada y son muy susceptibles al ruido. Por otro lado, los no iterativos tienen una complejidad computacional menor, haciéndolos más rápidos pero pierden precisión en la estimación.

La formulación general de los métodos iterativos es encontrar la transformación T_k que minimize el siguiente criterio:

$$min_{T_k} \sum_i \left\| p_k^i - \widehat{p}_{k-1}^i \right\|^2 \tag{2.4.6}$$

En esta ecuación p_k^i son los puntos en la imagen I_k y $\widehat{p}_{k-1}^i$ son las reproyecciones de los puntos 3D X_{k-1}^ien la imagen I_k de acuerdo a la transformación T_k.

Existe una técnica no iterativa llamada EPnP (Efficient Perspective-n-Point Camera Pose Estimation [23]) que calcula una solución única y precisa para la posición de la cámara con cuatro o más puntos. Como en la mayoría de los problemas PnP primero se calcula la localización de los puntos tridimensionales en relación al marco de referencia de la cámara, la causa en la eficiencia de este algoritmo es la representación de estos puntos tridimensionales como sumas ponderadas de puntos de control; los puntos de control son usados para el resto de las operaciones que implican calcular la orientación y posición de la cámara. Esta solución suele tener un buen grado de precisión y además se ejecuta rápidamente.

2.4.4. Keyframes

El proceso de estimación de la pose recae principalmente en las asociaciones de puntos característicos, y en el caso de usar PnP, de la reconstrucción 3D de la escena (los puntos tridimensionales). En la mayoría de las ocasiones los frames que fueron tomados de forma contigua contienen casi la misma información ya que el intervalo que los separa es muy corto y el movimiento es mínimo. Por tal motivo, se necesita buscar aquellos frames que aportan la mayor información en relación a los puntos detectados y los cambios de movimiento. Estos son los frames característicos o Keyframes. Existen criterios heurísticos para la selección de Keyframes como los siguientes:

- Debe de haber trascurrido un tiempo mínimo.

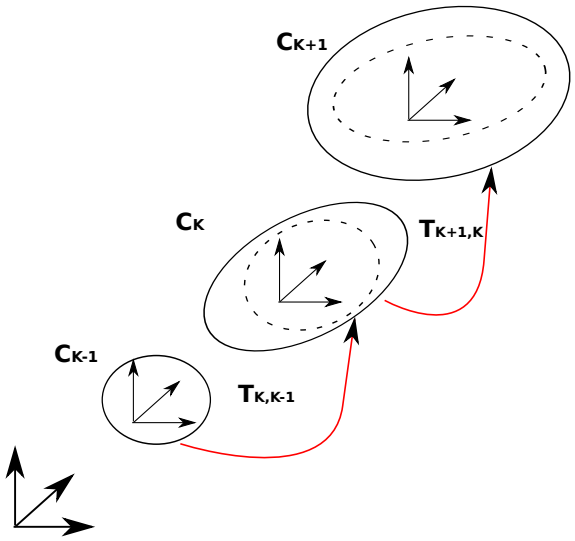


Figura 2.5.1: La incertidumbre de la posición de la cámara es una combinación entre la incertidumbre de la pose anterior y la incertidumbre en la transformación. Es un error incremental que se debe minimizar.

- Debe de haber una buena fracción de puntos emparejados entre frames.
- El error de reproyección de los puntos 3D sobre el nunevo frame debe ser menor a un umbral.
- Si la posición actual de la cámara se encuentra alejada del último Keyframe capturado.

La selección de Keyframes es fundamental para la odometría visual y se debe de realizar antes de actualizar la posición.

2.5. Proceso de optimización de la trayectoria estimada

La odometría visual funciona calculando la trayectoria de manera incremental; se calcula de manera discreta, pose tras pose. Los errores que suceden en las estimaciones se van acumulando uno tras otro generando desviaciones en la trayectoria. Es importante llevar al mínimo las desviaciones y errores generados; esto se puede lograr usando un método de optimización local sobre un número n de poses estimadas. Algunas de las técnicas más usadas son el bundle adjustment local [44] y la optimización de grafos.

2.5.1. Bundle adjustment local

Bundle Adjustment es el problema de refinar de manera óptima la estructura 3D (puntos de la escena) y los parámetros (extrínsecos y/o intrínsecos) de un conjunto de vistas. Este método minimiza una función de costo del error de reproyección para estimar los parámetros de la estructura y de las vistas.

La formulación del problema es la siguiente: Se considera que un conjunto de puntos de la escena X_j es visto por un conjunto de cámaras con matrices de proyección P_i. Cada cámara proyecta los puntos X_j usando $x_{ij} = P_i X_j$, de modo que x_{ij} son las coordenadas del punto j sobre la imagen i. Para obtener las matrices de proyección P_i y los puntos de la escena X_j óptimos se minimiza la siguiente función de costo:

$$min_{P_i, X_j} \sum_{ij} d(P_i X_j, x_{ij})^2 \tag{2.5.1}$$

Esta función de costo es la suma de los errores de reproyección al cuadrado, aquí $d(x, y)$ es la distancia euclidiana entre los puntos de la imagen x y y. Este es un problema de minimización no lineal que se resuelve usando métodos de mínimos cuadrados como Levenberg-Marquardt. Si se usa Levenberg-Marquardt se deben de resolver ecuaciones con la siguiente forma:

$$(J^T J + \mu I)(\frac{1}{\mu})g = -g, \; con \; g = J^T f \tag{2.5.2}$$

donde J es el Jacabiano de la función de reproyección f, esta usa como parámetros las matrices de reproyección P_i y los puntos 3D X_j para generar las coordenadas x_{ij} de esos puntos sobre la imagen. La matriz Jacobiana se construye con los valores de las deridavas parciales $\partial x_{ij}/\partial P_K$ y $\partial x_{ij}/\partial X_K$, con esto en cuenta podemos deducir que estas derivadas son cero a menos que $i = k$ cuando se deriva con respecto P, y que $j = k$ cuando se deriva con respecto a X. Esto sucede porque las coordenadas proyectadas del punto 3D j en la imagen i sólo dependen de la matriz de proyección i, y estas coordenadas sólo dependen del punto j y de ningún otro. Gracias a la estructura de esta matriz, con escasos (*sparse*) valores distintos a cero, se pueden usar técnicas como la descomposición de Cholesky que reducen el tiempo de ejecución. Esta forma de resolver el problema se le conoce como Sparse Bundle Adjustment (SBA).

El bundle adjustment es un problema con alta complejidad computacional y que incrementa con el número de imágenes y puntos de la escena usados. Es por tal motivo que aplicaciones que requieren ejecución en tiempo real (odomometría visual, slam visual, etc.) no deben de usar la optimización sobre toda la secuencia de imágenes, más bien se debe de escoger un subconjunto que contenga la información más reciente, y que sólo se refinen esos parámetros. A esto se le conoce como Bundle Adjustment Local, ya que la minimización no es sobre todo el conjunto de imágenes. En la Figura se ilustra la operación del Bundle Adjustment Local.

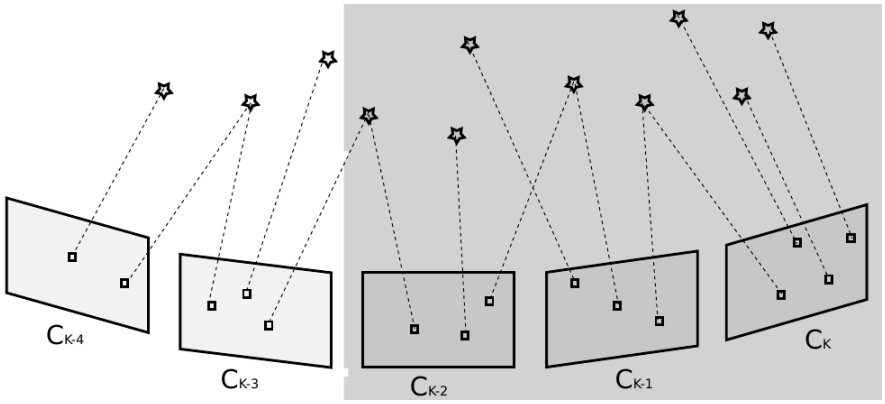


Figura 2.5.2: El Bundle Adjustment Local sólo optimiza los parámetros de las vistas más recientes, en esta figura se observa que sólo se usarán 3 vistas para refinar la estimación.

Capítulo 3

Arquitectura y desarrollo del sistema

La autolocalización de un robot durante el proceso de navegación es una tarea vital para lograr su autonomía. Es necesario que la información de su posición o su odometría se calcule en tiempo real, y que además sea preciso. Cuando se habla de tiempo real se espera que el sistema trabaje con una frecuencia mayor a 10 Hz, de esta manera se podrá conocer la pose del robot sin tener retrasos. Tomando en cuenta estas consideraciones se propuso el algoritmo que se muestra en la Figura 3.0.1. También se usó la biblioteca de software OpenCV, una plataforma conocida y muy popular en visión por computadora. OpenCV provee estructuras de datos para el manejo eficiente de imágenes y, además cuenta con una gran cantidad de algoritmos para procesamiento de imágenes, detección y descripción de puntos de interés, operaciones con matrices y reconstrucción 3D. Esta biblioteca es multiplataforma y se usa a través de C/C++ o Python, por cuestiones de eficiencia el sistema se programó en C++. Una implementación del Sparse Bundle Adjustment aún no está disponible para OpenCV, por tal motivo se usó un paquete externo llamado SSBA (Simple Sparse Bundle Adjustment), éste implementa eficientemente un Sparse Bundle Adjustment y tiene una facilidad para seleccionar qué parámetros (intrínsecos o extrínsecos) se desean refinar. A continuación se da una explicación de los pasos usados para estimar la posición del robot usando una secuencia de imágenes.

3.1. Estado inicial

En un principio se asume que se conoce la transformación $T_{1,0}$ entre el primer par de imágenes, y que además estas son tomadas en intervalos separados para poder calcular puntos tridimensionales con poca incertidumbre. Esta inicialización se espera que sea provista por el robot que implemente el algoritmo usando algún otro sensor que determine su posición como encoders, IMU o GPS. Esta primera transformación es fundamental ya que no es posible determinar

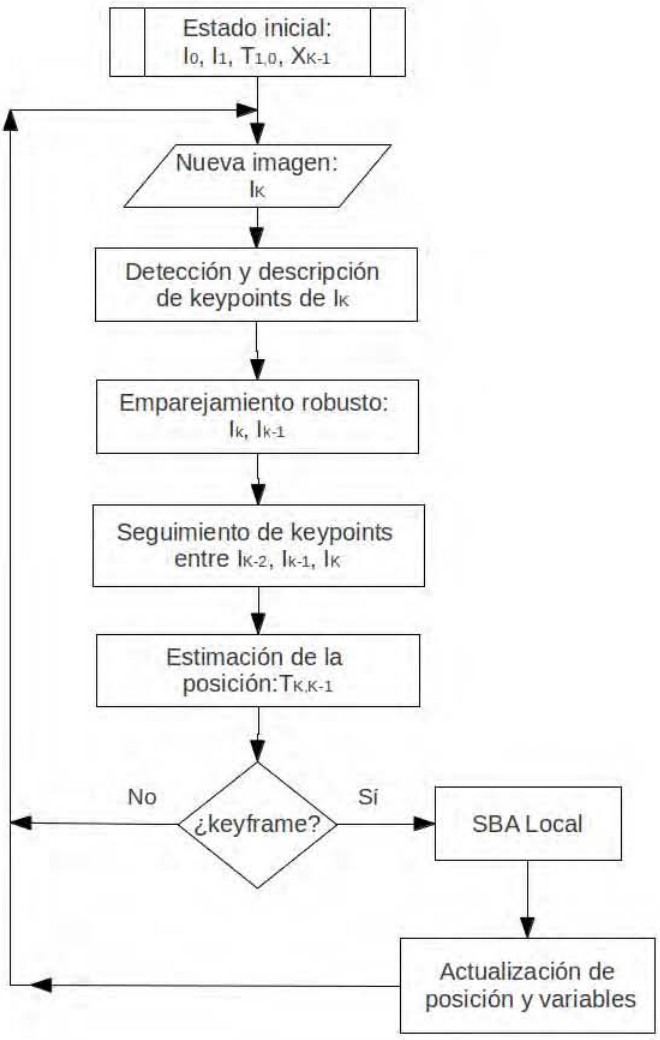


Figura 3.0.1: Diagrama de flujo del algoritmo propuesto.

la escala de la traslación a partir de sólo dos vistas. Conociendo la transformación $T_{1,0}$, los puntos de interés de las imágenes I_0 e I_1, y sus correspondencias podemos estimar mediante una triangulación los puntos 3D de la escena. Estos son los datos necesarios para que el sistema sea automático, desde este punto en adelante el algoritmo podrá estimar la posición de la cámara con sólo la secuencia de imágenes siguientes como entrada.

3.2. Detección y descripción de keypoints

En un principio, cuando se obtiene una nueva imagen se buscan sus puntos de interés; después se calculan sus descriptores para finalmente emparejar esos puntos con los puntos de los últimos dos frames. Este proceso requiere de un detector que sea eficiente, que proporcione una gran cantidad de puntos y que tenga buena precisión en la localización. También es necesario que el descriptor a utilizar sea bastante robusto y al mismo tiempo eficiente. Por tales motivos se decidió por un detector FAST multiescala y el descriptor FREAK.

Se escogió FAST porque es el detector con el mejor tiempo de ejecución y que además proporciona una gran cantidad de puntos. Uno de los principales problemas de los detectores basados en esquinas es que no son invariantes ante cambios de escala, ocasionando falta de repetibilidad; sin embargo, combinando el detector FAST con los conceptos de pirámide Gaussiana se obtiene un detector invariante a la escala.

Las pirámides Gaussianas se usan para realizar análisis multiresolución, crear una de estas pirámides implica aplicar un filtro Gaussiano a la imagen y submuestrearla en un factor de 2 hasta obtener el nivel deseado. La imagen en la base de la pirámide es la que tiene la mayor resolución, mientras que las imágenes superiores tendrán una resolución más burda (Figura 3.2.1).

El primer paso consiste en aplicar el siguiente filtro Gaussiano sobre la imagen original:

g=

$$g = \frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

Posteriormente, se eliminan las filas y columnas pares quedando una imágen con sólo un cuarto del tamaño original. Este proceso se repite hasta obtener el número de niveles deseados. Este procedimiento puede ser descrito por la siguiente convolución:

$$PG_n(i,j) = \sum_{u=-2}^{2}\sum_{v=-2}^{2} g(u,v)PG_{n-1}(2i+u, 2j+v) \tag{3.2.1}$$

donde PG_n es el nivel n de la pirámide que se obtiene a partir de la convoulción de las filas y columnas impares del nivel anterior $n-1$ con un filtro

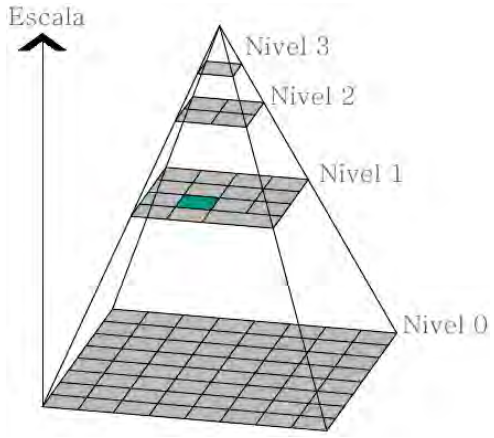


Figura 3.2.1: El detector FAST se aplica a cada celda de la cuadrícula en todos los nivéles de la pirámide Gaussiana y sólo se conservan los puntos con la mayor respuesta en su respectiva celda.

gaussiano de tamaño 5 x 5.

En este sistema de odometría visual monocular se aplica el detector FAST a la imagen original y a otros tres nivéles superiores de la pirámide, de esta manera se gana repetibilidad en la detección de puntos.

Otra característica que se desea es una buena distribución de los puntos en la imagen, la calidad en la estimación aumenta cuando se usa el mayor campo de visión posible. Por tal motivo, se crea una cuadrícula de 6 x 8 sobre el primer nivel de la pirámide; el algoritmo detector se aplica sobre cada celda y sólo un número máximo de puntos se conserva, de esta manera se conservan sólo los puntos más fuertes por región. Este proceso se realiza sobre el resto de los nivéles de la pirámide.

El descriptor que se escogió fue FREAK, este descriptor reciente es muy robusto y tiene un excelente tiempo de ejecución, además genera descriptores binarios los cuáles son asociados con otros descriptores más rápidamente usando la norma de Hamming.

3.3. Emparejamiento robusto

El emparejamiento de puntos entre la imagen I_{K-1}y la imagen I_K se lleva acabo usando un algoritmo de Fuerza Bruta. Se determinó que este algoritmo tiene un mejor desempeño que las técnicas de vecinos cercanos aproximados (approximate nearest neighbors). Fuerza Bruta encuentra un mayor número de asociaciones correctas y en un tiempo menor. Técnicas como Local Sensitive Hashing tienen mejores tiempos de ejecución cuando se tienen que emparejar una gran cantidad de puntos (cientos de miles o más), en este caso sólo se tienen que emparejar, en promedio, 1000 puntos.

	Matriz Fundamental (8 puntos)	Matriz Esencial (5 puntos)
Precisión	Menor, es susceptible a ruido	Mayor, la solución explota mejor las restricciones geométricas
Complejidad	Menor, resolución de un sistema de ecuaciones lineal	Mayor, se calculan los coeficientes de un polinomio de décimo grado y finalmente se obtienen sus raíces
Restricciones	Impreciso cuando el movimiento es paralelo al plano de la imagen	Sin restricciones
RANSAC	Se requiere un mayor número de hipótesis para encontrar una solución confiable pero el tiempo de ejecución de cada una de ellas es menor.	Se requiere un menor número de hipótesis pero el tiempo de ejecución de cada hipótesis es mayor

Figura 3.3.1: Comparación entre el filtrado de los puntos usando la matriz fundamental o la matriz esencial [25].

Para incrementar la calidad de la asociación se usa una verificación cruzada para eliminar posibles errores. Esta verificación funciona de la siguiente manera: cuando el emparejador de Fuerza Bruta encuentra las mejores asociaciones (aquellos vectores cuya distancia haya sido la menor) se verifica que el vector i de la primera imagen este emparejado con el vector j de la segunda, y viceversa; de no ser así se elimina ese emparejamiento. Usando esta técnica se asegura que se tengan pares consistentes.

Después de este proceso existe una alta probabilidad de encontrar aún asociaciones erróneas, por tal motivo se explotan las restricciones geométricas (geometría epipolar) entre las vistas para filtrar los datos incorrectos. En este punto se decidió usar cualquiera de los siguiente métodos para filtrar los emparejamientos: el primero consiste en calcular la matriz fundamental con el algoritmo de los 8 puntos, y el segundo consiste en calcular la matriz esencial con el algoritmo de los 5 puntos; en ambos casos se usa un esquema de RANSAC. Existen ventajas y desventajas de usar alguna de estas técnicas, esto se muestra en la Figura 3.3.1.

Estos métodos son usados con el algoritmo 2.1. Al final se obtienen los emparejamientos que mejor se ajustan al modelo que generó el menor error. La matriz fundamental que se encontró si se usa el algoritmo de 8 puntos será usada posteriormente para la triangulación polinomial, y si se calculó la matriz esencial con el algoritmo de 5 puntos se determina la matriz fundamental con esta relación $F = K^{-T}EK^{-1}$.

3.4. Seguimiento de puntos

En este paso se encuentra la relación que existe entre los puntos encontrados en las imágenes I_{K-2}, I_{K-1} e I_K, es decir, se obtiene la trayectoria de los puntos entre estos tres frames vecinos. La forma en que se determina es haciendo corresponder los emparejamientos anteriores, es decir, encontrar un punto en común entre los emparejamientos entre I_{K-2}- I_{K-1} e I_{K-1}- I_K. La implementación que se desarrolló es muy simple, sólo se almacenan los índices de los emparejamientos que son comunes en el primer y en el segundo par.

3.5. Estimación de la posición

La estimación de la posición se determina usando *Perspectiva desde n Puntos (Pnp)*, en específico se usa EPnP debido a que es un método con tiempo de ejecución bajo y no es tan susceptible a ruido. Para usar EPnP se necesitan conocer al menos 4 puntos tridimensionales y sus correspondencias en la imagen I_K. El estado inicial ya nos proporciona una transformación y las correspondencias entre las dos primeras imágenes, con esto realizamos una triangulación para obtener los puntos tridimensionales usando el método polinomial. Estos puntos 3D son filtrados a paritr de las relaciones encontradas en el paso anterior, se busca que haya congruencia entre los tres frames, es decir, se eliminan aquellos puntos 3D que no estén en la imagen I_K. Ya que se conocen estos valores se ejecuta el EPnP usando RANSAC, de esta manera se asegura la eliminación de alguna asociación incorrecta remanente y de puntos tridimensionales que no hayan sido calculados correctamente. Tanto la matriz fundamental como la esencial proporcionan información sobre la rotación relativa que existe entre los últimos dos frames, con el uso de esta información se acota el espacio de búsqueda del algoritmo, además de que ahora son necesarias menos iteraciones y se descartan posibles estimaciones erróneas. Finalmente, la estimación de la posición se refina a través de un método iterativo que minimiza la siguiente función de costo

$$\sum_{i=1}^{n} d(x_i, KTX_i)^2$$

En donde x_i representa las coordenadas de los puntos en la imagen, K los parámetros intrínsecos de la cámara, T los parámetros extrínsecos estimados con EPnP, y X_i representa a los puntos tridimensionales. Es decir, se minimiza el error de reprojección de los puntos tridimensionales en la imagen I_K. OpenCV cuenta con una función paralelizada que encuentra la posición y orientación de la cámara usando EPnP, esto disminuye el tiempo de ejecución ya que se hace uso de los procesadores disponibles en la computadora.

3.6. Selección de Keyframes

Como se había mencionado anteriormente la selección de Keyframes es un paso esencial en la odometría visual. Aquí se decide si la imagen I_K es considerada para actualizar la pose del robot o si es desechada para posteriormente tomar una nueva imagen. En el sistema existen dos condiciones que se deben de cumplir para que la imagen se considere como Keyframe: la primera condición es que no debe de ser una imagen cercana al último Keyframe capturado; y la segunda condición consiste en seleccionar a la imagen cuya transformación genere los puntos 3D con una incertidumbre menor a un umbral. En realidad, ambas condiciones son similares, sólo se busca que haya un cierta distancia o disparidad entre los Keyframes.

En un principio, una variable se asigna a cero cuando recien se ha encontrado un Keyframe, esta incrementará su valor en uno por cada imagen nueva que se captura; mientras esta variable no llegue a un valor específico el programa desechará automáticamente esas imágenes asegurando dos cosas: uno, que no se escojan imágenes cercanas como Keyframes y dos, que se disminuya el uso de recursos computacionales durante ese lapso. Una vez que el contador alcanza un el umbral se comienzan a estimar posiciones con EPnP como se describió en el paso anterior.

Conociendo la última posición se ejecutan dos operaciones para determinar la certeza en la estimación: la primera consiste en reprojectar los puntos tridimensionales en el último Keyframe y la imagen más reciente, se almacenan las distancias entre los puntos reales y los puntos reproyectados y se analiza que el valor del promedio de las distancias sea cercano a cero y que la desviación estándar sea menor a un umbral; la segunda operación es similar a la primera, sólo que en esta los puntos tridimensionales son generados por los frames mencionados y de igual manera se compara el promedio y la desviación estándar con otro umbral. Estas operaciones nos indican la calidad de la reconstrucción 3D, mientras la calidad de reconstrucción no sea lo suficientemente buena no se considerarán a las imágenes entrantes como Keyframes. La incertidumbre en la triangulación de puntos se puede observar en la Figura 3.6.1.

3.7. SBA local

Una vez que se selecciona un nuevo Keyframe se efectua el Sparse Bundle Adjustment sobre esa imagen y los últimos dos Keyframes, es decir, sobre una ventana de tres frames. El fin de este proceso, como ya se mencionó anteriormente, es refinar la estimación para que exista congruencia entre el nuevo Keyframe y los dos últimos asegurando que el error de reproyección sea mínimo. En el estado inicial se conocen dos frames y su transformación, estos se consideran como los primeros dos Keyframes. En un principio, cuando el sistema encuentra un nuevo Keyframe en la secuencia sólo se optimiaza la posición de esta nueva imagen y las posiciones de las primeras dos imágenes no se refinan, esto porque se tiene una gran certidumbre sobre la primera transformación.

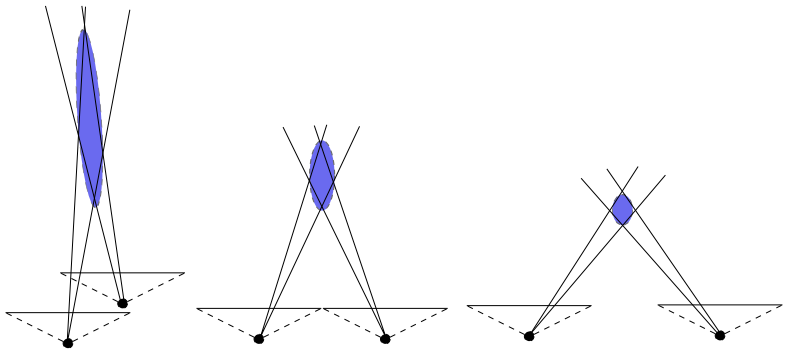


Figura 3.6.1: La región azul indica la incertidumbre en la localización del punto 3D. Esta incertidumbre depende del ángulo entre los rayos proyectados.

Nuevamente, cuando se haya otro Keyframe se toman solamente los últimos tres, pero ahora se refinarán las posiciones de los dos Keyframes más recientes, y la posición del primer Keyframe se refinará

En un principio las imágenes que se conocen en el estado inical se consideran como los dos primeros Keyframes, cuando se encuentra un nuevo Keyframe el SBA se ejecuta usando las posiciones de la cámaras, y los puntos tridimensionales con sus respectivas proyecciones en las tres vistas; se especifíca dentro del optimizador que las primeras dos posiciones son fijas y que sólo se desea refinar la última posición. Esto se hace así porque se considera que existe una alta certidumbre en las primeras dos posiciones. Cuando se encuentra un nuevo Keyframe el sistema desecha el Keyframe más antiguo para que sólo haya tres vistas dentro de la optimización. Para las ejecuciones siguientes del optimizador se mantiene fija la posción del primer Keyframe y sólo se mejoran las posiciones de los últimos dos Keyframes, así mantenemos fija la posición que ya ha sido refinada más de una vez, lo que ahorra tiempo de computo.

Para este paso se usa el paquete SSBA (Simple Sparse Bundle Adjustment [45]) que es un conjunto de funciones específicas para resolver el SBA. Este hace uso de operaciones incluidas en la biblioteca de Software SuiteSparse que se diseñaron para ejecutarse de forma eficiente sobre el procesador, minimizando el tiempo de ejecución. SSBA se configuró para que sólo refinaran los valores de los puntos 3D estimados y los parámetros extrínsecos de las vistas.

3.8. Actualización de posición y variables

Este paso se ejecuta cuando se detecta un nuevo Keyframe, y sólo hace un corrimiento general de la información, primero se elimina el Keyframe más antiguo y se reasignan los índices de las variables, es también aquí donde se generan los nuevos puntos tridimensionales usando el método polinomial con la matriz fundamental estimada en el paso de emparejamiento robusto.

Capítulo 4

Experimentación y resultados

4.1. Configuración de los experimentos

Una vez definido el algoritmo de odometría visual se procedió a probarlo con dos secuencias de imágenes. La primera de ellas es una trayectoria lineal con una longitud de 2.12m en un cuarto cerrado (Figura 4.1.2), las fotografías son perpendilculares al movimiento y se toman en intervalos de un centímetro, es decir, se tomaron 212 fotos. En este entorno se colocaron varios objetos que ocasionaran una alta detección de puntos de interés, y la distancia entre los objetos de la escena y la cámara es corta, lo que aumenta la precisión en la triangulación. La segunda secuencia de imágenes consiste de igual manera en una trayectoria lineal de 24m, esta vez en un espacio abierto (Figura 4.1.9) y sin un intervalo definido entre frames, las fotografías son igualmente perpendilculares al movimiento y en total se capturaron 1278 imágenes. En esta secuencia los puntos de interés se encuentran más alejados de la cámara y existen frames que contienen textura uniforme, lo que dificulta la extracción de puntos característicos.

Ambas secuencias fueron tomadas con la cámara PlayStation Eye (Figura 4.1.1). Este dispositivo se escogió porque puede tomar imágenes a 120 Hz con una resolución 320 x 240 pixeles, y a 60 Hz con una resolución de 640 x 480 pixeles, con esto se puede evitar hasta cierto punto el barrido de la imagen ocasionado por el movimiento.

El sistema de odometría visual cuenta con un gran número de parámetros los que pueden hacer que el desempeño del sistema cambie significativamente. La rapidez y precisión que exhibe el sistema son objetivos en conflicto, al modificar los parámetros se puede lograr un punto medio aceptable entre estas dos características. Los parámetros principales del sistema son:

- El umbral usado por el detector de puntos de interés, si el umbral es muy alto sólo los puntos con una respuesta muy alta se conservarán; mientras

Figura 4.1.1: Cámara PlayStation Eye usada para tomar las fotografías en los experimentos. Puede tomar fotos con una resolución de 640 x 480 pixeles a 60 Hz o fotos de 320 x 240 pixeles a 120 Hz. Además cuenta con dos zooms ajustables.

que si es muy baja bastantes puntos con poca repetibilidad permanecerán.

- El máximo número de puntos que se desea encontrar por celda de la cuadrícula, si este valor es muy pequeño pocos puntos con respuesta alta se conservarán y estos son los que usualmente tienen mayor índice de repetibilidad, esto ocasionaría dificultades en la etapa de seguimiento; en caso de ser muy grande el sistema sólo tardaría más tiempo en ejecutarse.
- El número de iteraciones usadas en el RANSAC para filtrar las asociaciones correctas tanto en la estimación de la matriz Fundamental y Esencial, y en la estimación de la pose con Perspectiva desde n Puntos. Mientras se ejecuten más iteraciones existe una mayor probabilidad de encontrar aquel conjunto de datos que genere al modelo que mejor describa a los datos, pero igualmente aumentaría el tiempo de ejecución.
- La distancia mínima usada por el RANSAC para determinar si un punto es parte del modelo estimado. Con este valor decidimos si consideramos a un emparejamiento como cierto o falso, si el valor es alto una mayor cantidad de emparejamientos se conservarán, pero también existe una mayor probabilidad de que se dejen pasar asociaciones erróneas.
- El umbral de reproyección máximo usado para determinar si se considera una nueva imagen como Keyframe. Este parámetro mide la calidad de la estimación, si es muy alto muchas imágenes se considerarán como Keyframes lo que puede afectar a la trayectoria si no son buenas estimaciones.
- Y el número de imágenes que no serán procesadas después de que se haya encontrado un Keyframe. Este parámetro se utiliza para disminuir el uso de recursos tan pronto se haya procesado un Keyframe.

4.1.1. Resultados de la primera secuencia

Los siguiente parámetros se usaron para realizar las pruebas sobre la primera secuencia:

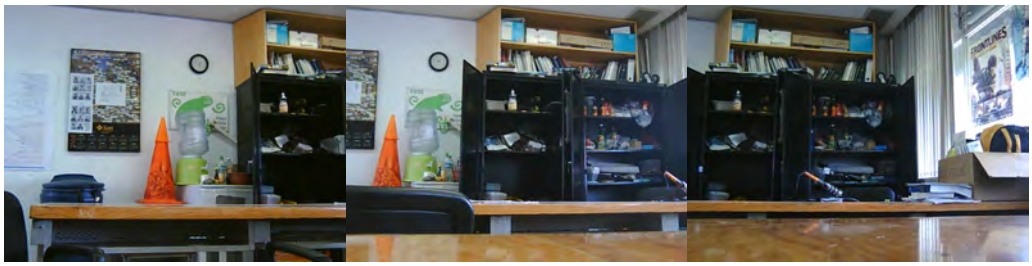


Figura 4.1.2: Cubículo de PROTECO. Estas imágenes son parte de la primera secuencia, de derecha a izquierda son la imagen con índice 0, 100 y 200.

Parámetros	Valores
Transformación inicial	30 cm
Umbral del detector	30
Número máximo de puntos por celda	15
Probabilidad de éxito en la estimación de la matriz Fundamental/Esencial	0.99
Número de iteraciones usadas en PnP+RANSAC	500
Distancia mínima usada en el RANSAC	3.0
Umbral del error de reproyección máximo permitido por Keyframe	0.9
Número de imágenes no procesadas después de un Keyframe	10

La secuencia de imágenes se analizó en dos ocasiones, en una se cálculo la matriz Fundamental y en otra se calculó la matriz Esencial, ambas para filtrar asociaciones erroneas. Por cada ocasión, el algoritmo se ejecutó diez veces.

Este análisis comienza mostrando el comportamiento del sistema ilustrando el tiempo de ejecución y el error en la estimación de una sola ejecución, de esta manera se pueden analizar sus características principales. Posteriormente se analizarán los resultados obtenidos por varias ejecuciones del programa, así se podrá generalizar el desempeño del mismo.

En la figura 4.1.3 se muestran dos gráficas, en *(a)* se grafica el tiempo que tarda en ejecutarse el programa por cada frame de la secuencia y en *(b)* se observa la precisión con la que calcula la trayectoria en cada Keyframe. De *(a)* podemos decir que el programa no tiene un intervalo de tiempo constante, y se observa claramente con los picos sobresalientes es en dónde se localizaron los Keyframes, también se puede ver que después de haber encontrado un Keyframe

hay un periodo en el que no se realiza procesamiento. En *(b)* se aprecia la fluctuación del error en cada estimación, en un principio pareciera que el error acumulado por la falta de precisión en la transformación podría afectar a la estimación, pero no sucede así, al contrario la última estimación es la que menor error tiene. Con esto se comprueba que el Sparse Bundle Adjustment reduce el error ocasionado por transformaciones imprecisas y la incertidumbre en la posición de cada cámara usando como referencia los puntos tridimensionales y sus respectivas proyecciones.

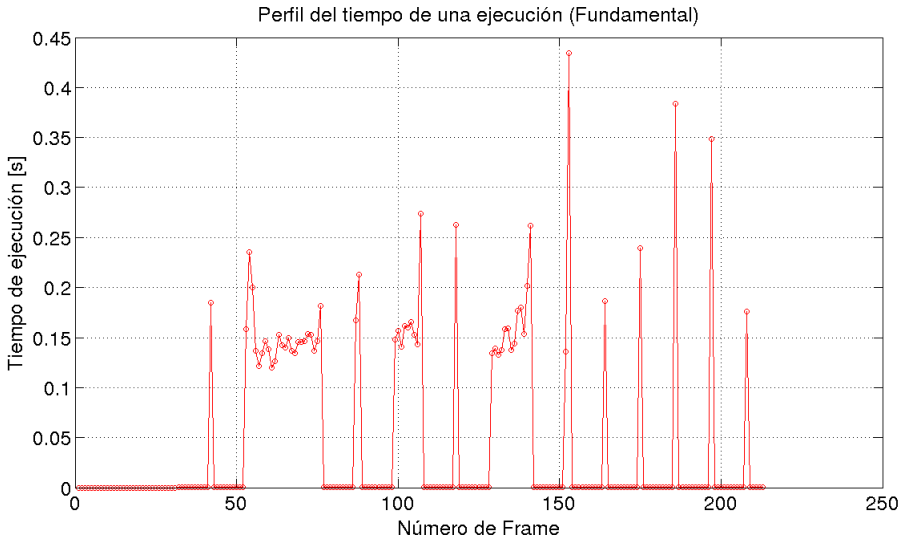
En la figura 4.1.4 se muestran los valores del error que ocurrieron con mayor frecuencia en la estimación de la trayectoria en distintas corridas. Se puede observar en la figura que tanto en *(a)* como en *(b)* existen picos en valores cercanos a 2, esto sucede porque todas las ejecuciones mostraron el mismo error de reproyección en su tercer Keyframe (el primer Keyframe detectado por el sistema). Observando la distribución del error tanto en *(a)* y en *(b)* podemos afirmar que en promedio el cálculo de la trayectoria es más preciso si se usa la matriz Esencial.

En la Figura 4.1.5 se muestra el error acumulado en cada una de las iteraciones, con estas gráficas podemos determinar cuáles fueron las mejores y las peores corridas en *(a)* y en *(b)*. Como se había comprobado anteriormente, las estimaciones que usan la matriz Esencial tienen menor error.

En la Figura 4.1.6 se dibujan las trayectorias (la mejor y la peor) con respecto a la trayectoria original. Se puede ver de nuevo el comportamiento del estimador, las poses son calculadas de forma discreta y en ocasiones pueden estar más cerca o más alejadas de la trayectoria original. En la Figura 4.1.7 se ponen como referencia los puntos tridimensionales calculados de una de las ejecuciones, esto se hace para visualizar la precisión de la trayectoria, que es relativamente buena.

El perfil de tiempos del sistema se muestra en la Figura 4.1.8, aquí se puede visualizar qué tareas consumen mayor tiempo. El emparejamiento robusto y el Sparse Bundle Adjustment resultaron ser los procesos más costosos mientras que el seguimiento, la detección y la descripción de los puntos de interés fueron los más rápidos. Con estas gráficas determinamos que tareas deben mejorarse. De la tabla *(c)* vemos que si bien el uso de la matriz esencial produce trayectorias más precisas, su tiempo de ejecución es más costoso por 30 ms.

Esta primera secuencia fue de gran ayuda para la depuración del código así como del refinamiento empírico de parámetros del sistema, el cual inicialmente se obtuvo de las sugerencias e interpretación de resultados encontrados en la literatura respectiva.



(a)

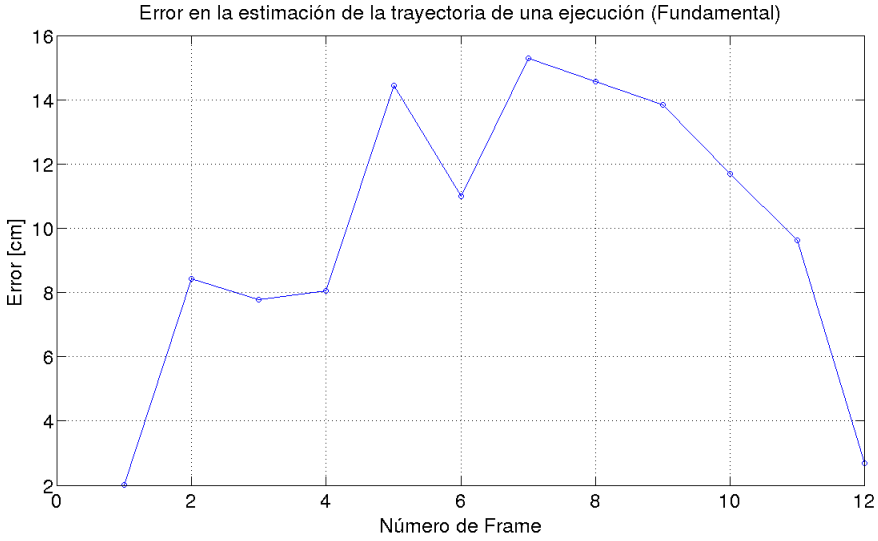
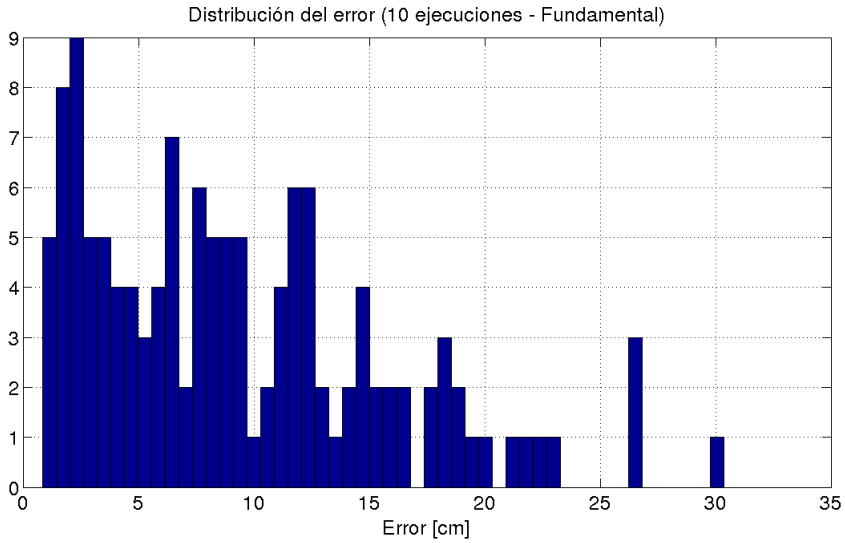
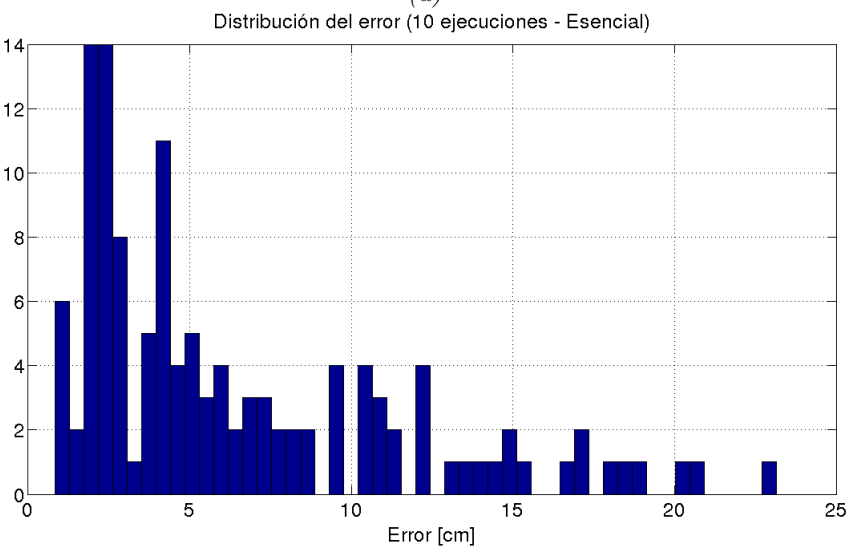


Figura 4.1.3: Perfil de tiempo y error de estimación en una sola ejecución. En *(a)* se puede observar el tiempo que tarda el sistema en procesar un frame, se muestra a los Keyframes como los picos más sobresalientes seguidos por tiempos cercanos a cero. Y en *(b)* se observa el error de estimación por Keyframe, se puede observar que el error no es del todo incremental como sucede en la odometría por ruedas, esto se sucede porque el SBA minimiza este error a partir de los puntos 3D y sus reproyecciones.

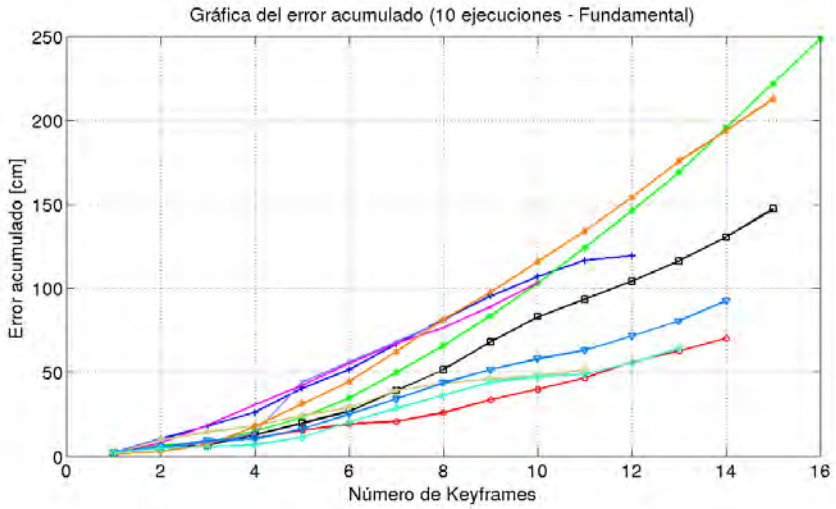


(a)

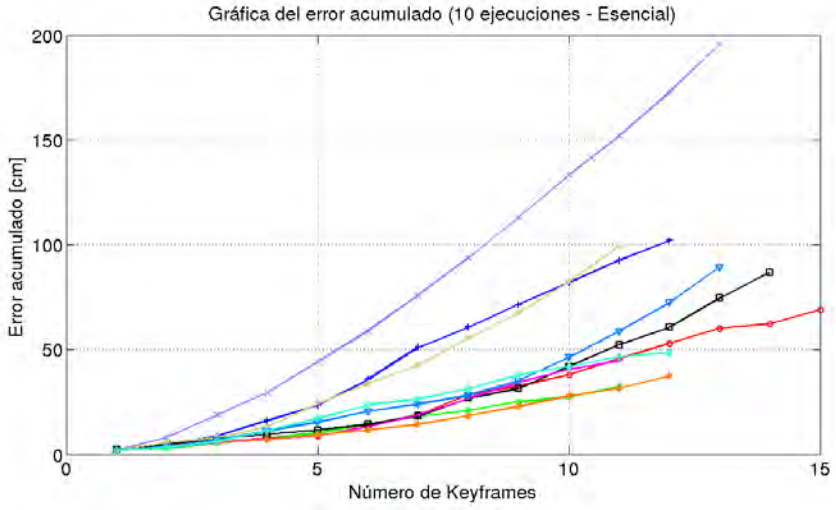


(b)

Figura 4.1.4: La distribución de los errores de las trayectorias estimadas*; (a)* matriz Fundamental, promedio = 9.3042 y desviación estándar = 6.4830; *(b)* matriz Esencial, promedio = 6.4956 y desviación estándar = 5.1094.

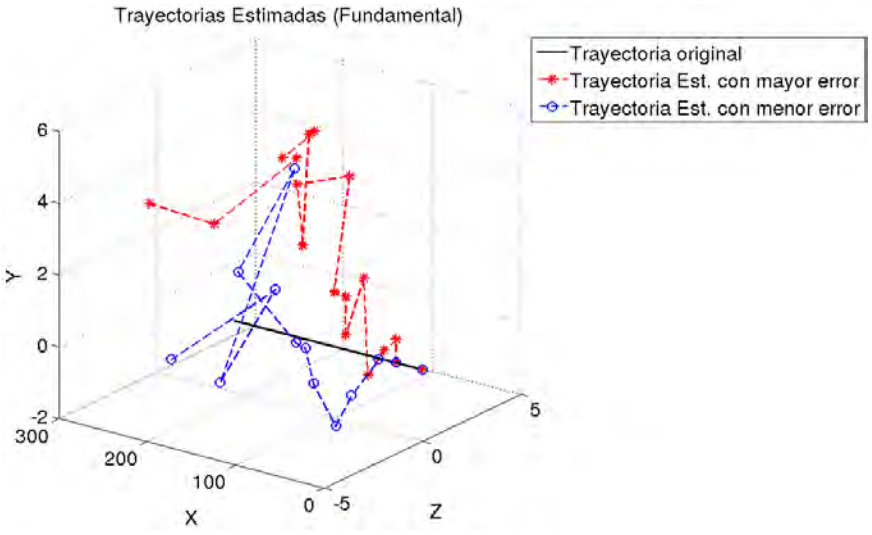


(a)

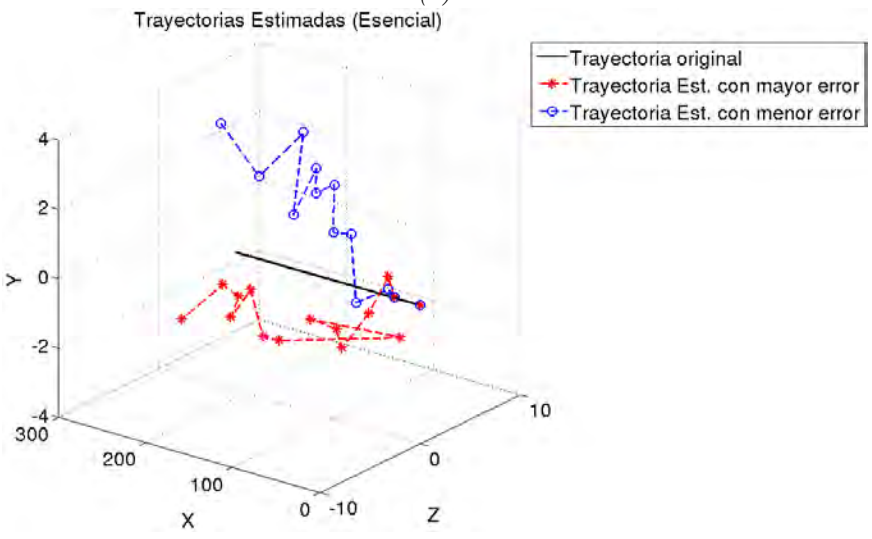


(b)

Figura 4.1.5: Con estas gráficas del error acumulado se puede determinar el desempeño individual de cada una de las ejecuciones. Las diferentes pendientes entre puntos indican las fluctuaciones en el error en cada estimación, es decir, en ocasiones el error puede ser mayor o menor a la estimación anterior.

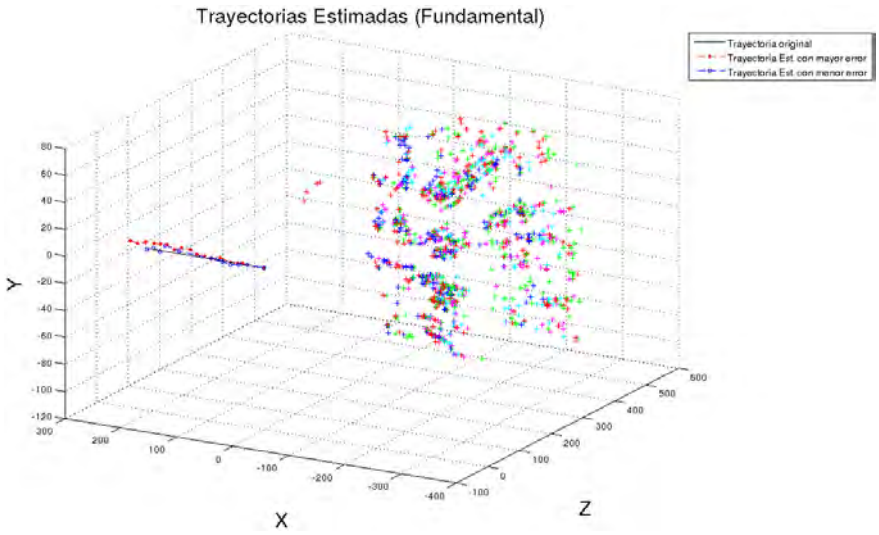


(a)

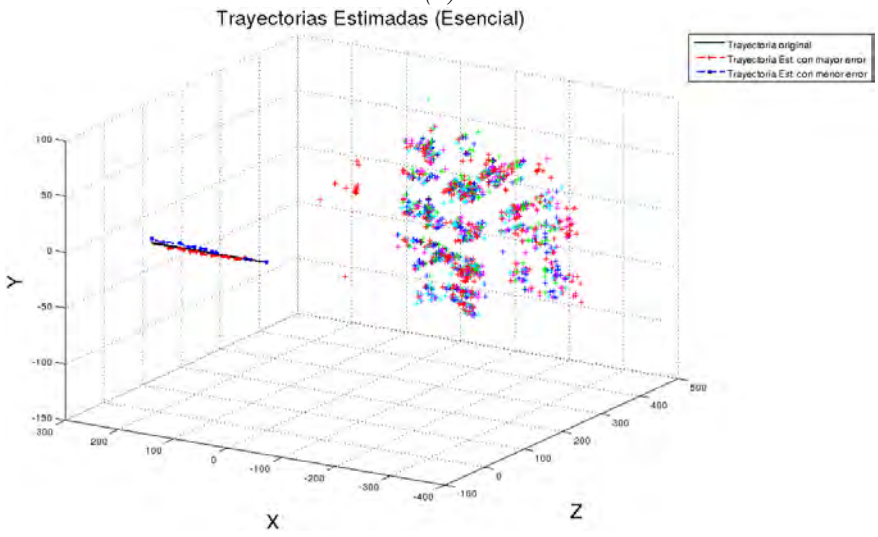


(b)

Figura 4.1.6: Se muestran las trayectorias con mejor y peor desempeño con respecto a la trayectoria original usando *(a)* la matriz Fundamental y *(b)* la matriz Esencial. Aunque la trayectoria roja es la que tiene mayor error acumulado se puede ver que no es una mala estimación y que le da seguimiento a la trayectoria original.

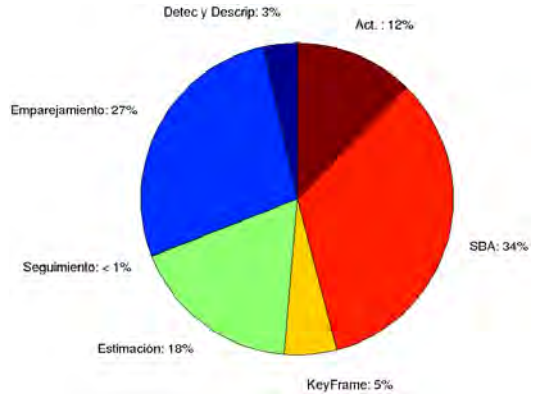


(a)

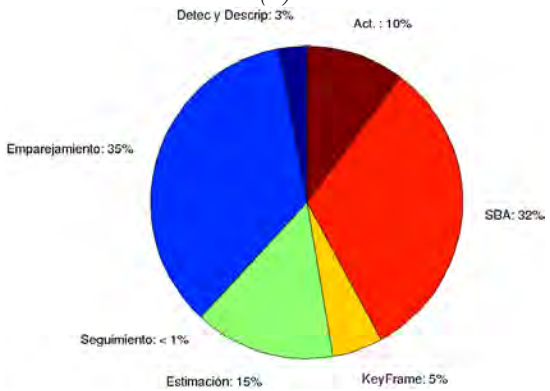


(b)

Figura 4.1.7: Vista de las trayectorias con respecto a la trayectoria original y con respecto a los puntos tridimensionales calculados para (a) Fundamental y (b) Esencial. A partir de los puntos tridimensionales podemos observar que existe una buena distribución de puntos de interés en la escena.



(a)



(b)

Tareas	Fundamental [s]	Esencial [s]
Detección y Descripción	0.0088	0.0083
Emparejamiento Robusto	0.0701	0.1028
Seguimiento	0.00010423	0.00013041
Estimación	0.0451	0.0425
Detección de Keyframe	0.0139	0.0152
SBA	0.0857	0.0929
Actualización	0.0314	0.0298
Total	0.2553	0.2916

(c)

Figura 4.1.8: Se muestra el porcentaje de tiempo de cada tarea usando la matriz *(a)* Fundamental y *(b)* Esencial. De la tabla *(c)* observamos la cantidad de tiempo promedio por tarea. Las etapas que son más costosas computacionalmente son el emparejamiento robusto y el Sparse Bundle Adjustment.

4.1.2. Resultados de la segunda secuencia

Los siguiente parámetros se usaron para realizar las pruebas sobre la segunda secuencia:

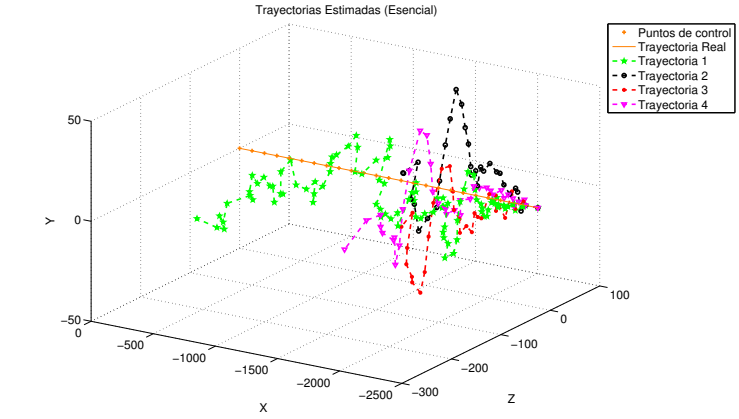
Parámetros	Valores
Transformación inicial	100 cm
Umbral del detector	10
Número máximo de puntos por celda	60
Probabilidad de éxito en la estimación de la matriz Fundamental/Esencial	0.99
Número de iteraciones usadas en PnP+RANSAC	2000
Distancia mínima usada en el RANSAC	1.0
Umbral del error de reproyección máximo permitido por Keyframe	1
Número de imágenes no procesadas después de un Keyframe	10

Estos parámetros se ajustaron de ta manera que la estimación fuera precisa, es decir, se usan muchos puntos por celda y muchas iteraciones para aumentar la probabilidad de una buena estimación; sin embargo como se muestra en la Figura 4.1.10 el sistema no pudo completar la trayectoria. Se puede apreciar que las estimaciones se detienen cuando el sistema comienza a calcular valores cercanos a los 10 m, en esta gráfica se muestra que sólo una ejecución pudo continuar más allá de ese valor pero aún así no pudo terminar la trayectoria.

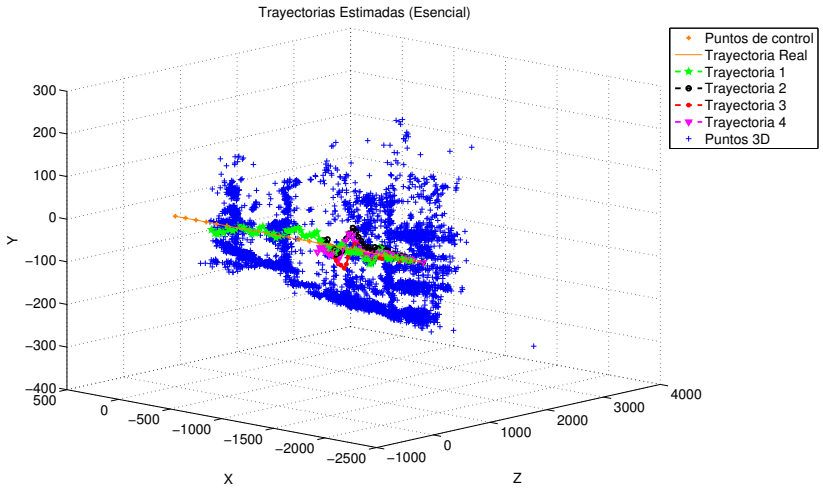
Se analizó detalladamente la infomación usada por el sistema en cada frame, y finalmente se diagnosticó que el problema se encontraba en el proceso de estimación de la pose (EPnP con RANSAC). Para demostrar esta idea se graficaron las hipótesis generadas por RANSAC en tres Keyframes; en la Figura 4.1.11 se muestran las hipótesis de los Keyframes anteriores al fallo, y en la Figura 4.1.12 se observa la distribución de posiciones cuando el sistema falla. Las gráficas con estimaciones previas al error muestran una distribución monomodal en cuyo pico se encuentra la estimación con el máximo número de puntos. Sin embargo en el Keyframe del error observamos que la distribución de estimaciones es bimodal, se tienen dos picos. El algoritmo de RANSAC está diseñado para escoger sólo a la estimación con la mayor cantidad de puntos, pero en este Keyframe existen al menos siete valores máximos y están distribuidos entre los dos picos. Esto nos dice que existe una alta probabilidad de que estimaciones con un alto número de puntos podrían ser estimaciones erróneas. Por tal motivo, escoger a la posición con el máximo número de puntos no es un criterio de evaluación válido para

Figura 4.1.9: Fotografía del espacio abierto en donde se tomó la segunda secuencia. El movimiento es paralelo a las casas.

estas situaciones, se necesita otra solución más robusta. En el siguiente capítulo se plantean posibles soluciones a este problema.

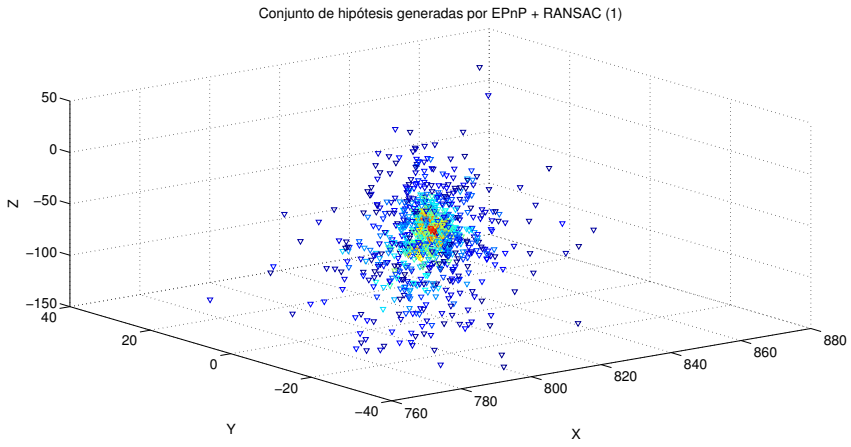


(a)

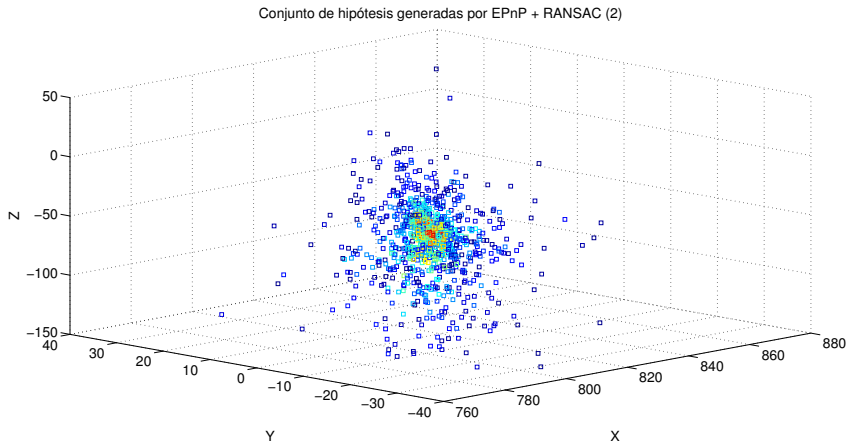


(b)

Figura 4.1.10: En la gráfica *(a)* se puede ver la estimación incompleta de las trayectorias. Y en *(b)* se ponen en perspectiva estas trayectorias con respecto a los puntos 3D calculados. La odometría visual además de estimar la posición también genera un mapa de puntos de la escena, en la imagen se pueden apreciar las paredes localizadas entre las casas, las banquetas y el adoquín.

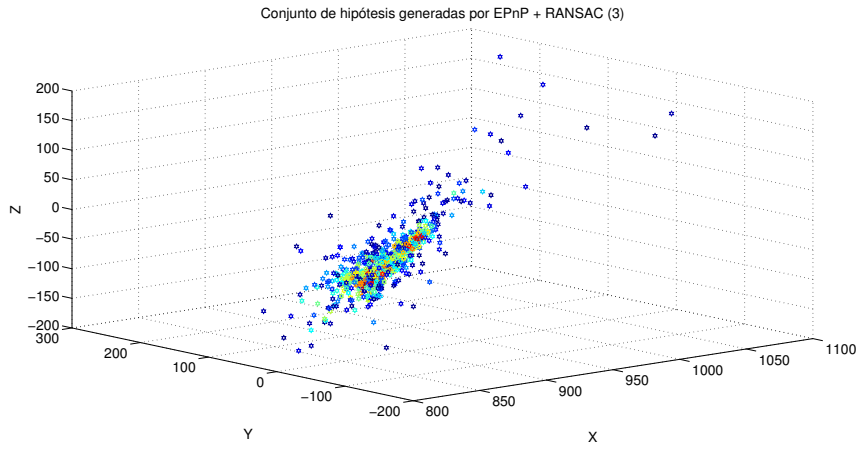


(a)

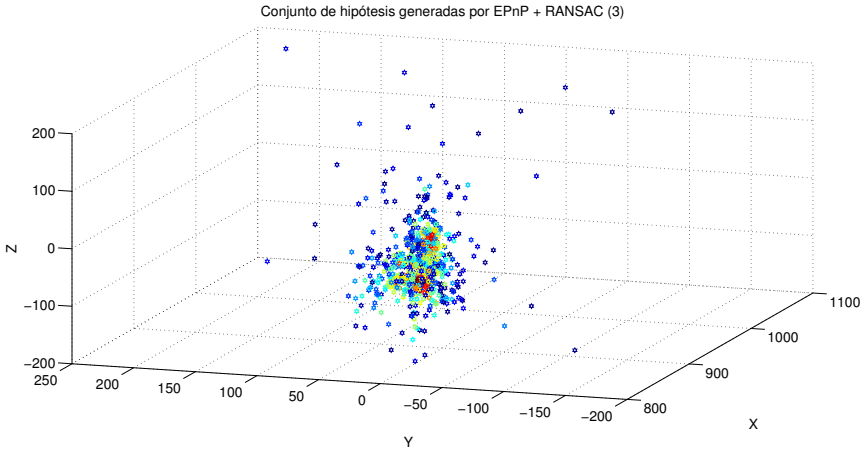


(b)

Figura 4.1.11: Distribución de las posiciones estimadas por EPnP con RANSAC antes de que fallara el sistema en el Keyframe$_k$. Las posiciones en *(a)* son las que se estimaron en el Keyframe$_{k-2}$ y las posiciones en *(b)* se estimaron en el Keyframe$_{k-1}$. Los colores indican el número de puntos que se encuentran dentro del modelo estimado, el color azul es para aquellos con pocos puntos, y el rojo cuanta con la mayoría. Se puede apreciar que las estimaciones siguen una distribución monomodal.



(a)



(b)

Figura 4.1.12: Las gráficas *(a)* y *(b)* muestran la disposición de las estimaciones en diferentes perspectivas de manera que se pueda apreciar con mayor detalle como varía la distribución en este Keyframe que falló. La notación usada es la misma que en la Figura 4.1.11. Aquí se puede observar que la distribución es multimodal, se tienen dos regiones con máximos. Tener más de un máximo en regiones distintas del espacio es la causa por la que el sistema falla, una de esas estimaciones considerada como máximo debe ser falsa.

Capítulo 5

Conclusiones y trabajo futuro

5.1. Conclusiones

Con los resultados obtenidos de las pruebas se pueden hacer las siguiente conclusiones:

- La transformación entre los dos primeros Keyframes que se provee en el estado inicial debe ser precisa, si existe mucha incertidumbre el cálculo de la trayectoria puede fallar. La falla sucede porque los primeros puntos tridimensionales son calculados usando la primera transformación, si estos puntos tridimensionales no tienen congruencia con sus respectivas proyecciones el error de reproyección será elevado, ocasionando que no haya Keyframes.

- La integración de FAST con una pirámide Gaussiana y con una detección distribuida eleva la repetibilidad de los puntos encontrados y aumenta la calidad de la estimación. Sin embargo, la localización del punto es pobre en los niveles superiores de la pirámide lo que genera errores en la trayectoria. Una alternativa a la implementación usada es BRISK, un detector de esquinas que localiza mejor al punto encontrado y además es multiescala, este detector es muy robusto pero aún es muy lento comparado con FAST.

- Encontrar las asociaciones correctas de puntos resulta ser la tarea más complicada hasta el momento. Dentro del sistema es la operación que más tiempo consume y la más propensa a fallar. El enfoque que se adoptó dentro del sistema fue emparejar los puntos usando sólamente la similitud entre los vectores de descripción, posteriormente se filtran aquellos puntos que no cumplen con la restricción epipolar. Esta aproximación sí devuelve los emparejamientos correctos, pero elimina una gran cantidad de puntos, sólo se conserva un 10 o 20 por ciento del total de puntos encontrados.

Es necesario un emparejamiento que no sólo dependa de la distancia entre vectores, se debe de hacer una asociación inteligente, definir áreas de búsqueda en donde exista una alta probabilidad de encontrar la asociación correcta, en vez de hacer una búsqueda sobre toda la imagen.

- La función que realiza el seguimiento de puntos se ejecuta de manera eficiente a pesar de ser una implementación sencilla y que tiene una complejidad de $O(n^2)$.

- La estimación de la posición es precisa y rápida con el método EPnP, aunque con tan sólo 4 puntos se puede hacer una estimación de la pose, se encontró que son necesarios más puntos para verificar dicha estimación. Si después de estimar la posición usando RANSAC sobreviven pocos emparejamientos (menos de 10) existe una alta probabilidad de que la estimación sea mala. De acuerdo a los resultados de la segunda secuencia se determinó que se debe usar un método distinto o una variación al algoritmo de RANSAC original debido a que no es lo suficientemente robusto. Se espera poder resolver el problema usando MAPSAC, esta variación usa una estimación maximum a posteriori que consiste en considerar una distribución calculada anteriormente en el sistema, es decir, una distribución ya conocida (prior) para estimar el Maximum Likelihood. Esta estimación puede tener una salidad multimodal y se empleó exitosamente en [15] para su sistema de SLAM monocular.

- Hasta el momento sólo existe un criterio de selección de Keyframes, que consiste en medir le error de reproyección de la estimación; sin embargo, es necesario un criterio que mida cuál es la calidad del seguimiento, y capturar un Keyframe mientras esta sea alta.

- El Sparse Bundle Adjustment funcionó como se esperaba, refinó las estimaciones del movimiento haciendo que el error disminuyera . Sin emabargo, considero que la función de costo que minimiza es muy propensa a fallar en caso de malos emparejamientos, en [12] los autores usan la siguiente función para medir el error residual que es resistente a malas asociaciones

$$e_r = ln(1 + \frac{e^2}{\sigma^2})$$

en donde e es el error de reproyección y σ es la desviación estándar del punto detectado.

Los mayores inconvenientes del sistema son el tiempo de ejecución lento y la falta de robustez. Es necesario que el sistema se ejecute en tiempo real y que pueda estimar de manera precisa la posición si se desae usar sobre un robot móvil. En la sección siguiente se plantean ideas que pueden aumentar el desempeño del algoritmo. Otro de los problemas que tienen los sistemas monoculares en general son las rotaciones puras. Para poder estimar la posición de la cámara es necesario conocer los puntos 3D que se obtienen por triangulación; sin embargo para realizar la triangulación es necesario que exista una disparidad entre vistas, la cual no existe cuando el movimiento no tiene traslación. Una posible

solución a este problema se plantea en [15] en donde a partir de una función de puntaje usando la matriz esencial y la matriz de homografía se determina si el movimiento tiene poca disparidad. Otra posible solución es fusionar la información de la odometría visual con otro sensor como una IMU como se desarrolló en [1].

A pesar de todas estas desventajas con este trabajo se lograron las bases para un sistema que usa una sola cámara para estimar la posición.

5.2. Trabajo futuro

Existe mucho trabajo por hacer para mejorar el desempeño del sistema, algunas de las ideas en las que se piensa trabajar más adelante se muestran a continuación.

- **Tareas a corto plazo:**
  - **Sustituir el paquete SSBA:** Aunque este paquete es eficiente y cuenta con muchas configuraciones para especificar el tipo de optimización ya no es soportado por su creador y su documentación es muy precaria. Por tal motivo se desea usar Ceres-Solver [16], un paquete que tiene como propósito la minimización de funciones de costo y provee de un gran número de técnicas para hacerlo. Es actualmente soportado por Google y en el se podrán evaluar nuevas funciones de costo para resolver el Bundle Adjustment.
  - **Aumentar el número de Keyframes en la optimización:** Si se incrementa el número de Keyframes usados en el Sparse Bundle Adjustment se aumentará la precisión en la estimación pero, de la misma manera el tiempo de ejecución. Se necesita una evaluación para conocer cuántos Keyframes se pueden usar en el Bundle Adjustment sin afectar en gran medida el tiempo de ejecución.
  - **Guardar Keyframes y puntos 3D:** En el sistema sólo se almacenan Keyframes, sin embargo considero necesario guardar puntos tridimensionales generados en tiempos anteriores. De esta manera se contarán con más restricciones cuando se efectue la optimización en caso de que el seguimiento de puntos sea muy bajo.
  - **Usar una variación del RANSAC original**: En la actualidad existen muchas variaciones al algoritmo de RANSAC original [9], las modificaciones hacen que el algoritmo sea más robusto, o más rápido o más preciso. Es necesaria una evaluación para determinar qué algoritmo incrementaría el desempeño del sistema.
  - **Integración con ROS:** El sistema está programado sobre la biblioteca de OpenCV, la cual ya tiene integración con ROS; sin embargo, ROS define variables y funciones que pueden ser usados de forma concurrente por un robot, es necesario usar esta metodología para tener mayor modularidad y portabilidad del sistema de odometría.

- **Emparejamiento epipolar:** Una solución al problema del emparejamiento es realizar búsquedas en áreas restringidas, éstas áreas puede ser las líneas epipolares. En un principio se procede a usar el algoritmo de Fuerza Bruta sobre los conjuntos de puntos que tengan la mayor respuesta entre dos imágenes, posteriormente se filtran usando el cálculo de la matriz Esencial, conociendo esta hipótesis de la matriz esencial asumimos que la pareja de un punto en la primera imagen se encuentra sobre la línea epipolar de la segunda imagen. De esta manera se emparejan los puntos restantes usando Fuerza Bruta y que además cumplan hasta cierto punto con la restricción epipolar. Así habrá una mayor cantidad de asociaciones correctas

- **Tareas a largo plazo:**

  - **Reconocimiento de lugares**: Aunque el reconocimiento de lugares va más enfocado a sistemas de SLAM, considero importante la capacidad de relocalización en caso de que el robot no pueda estimar nuevas posiciones debido a un pobre seguimiento de puntos u oclusiones momentáneas en su campo de visión. Soluciones exitosas a este problema se describen en [10, 33].
  - **Fusión con otros sensores:** Los robots cuentan con una gran cantidad de sensores además de la cámara. Se puede fusionar la información de todos aquellos sensores que calculen la posición para obtener una estimación más precisa. Se requiere un estudio de qué sensores y qué técnicas son usados para mejorar la estimación de la pose en la odometría visual.
  - **Detección de características usando modelos de atención visual:** La detección de puntos de interés se lleva acabo encontrando regiones que tienen grandes cambios en intensidad en sus vecindades, una alta repetibilidad es crucial para el algoritmo de odometría; sin embargo los humanos usamos más información de la escena para determinar objectos sobresalientes, algunas característias son el color, la orientación, el contraste, la curvatura y la profundidad, por mencionar algunos. La fusión de esta información usando un modelo de atención visual computacional puede hacer a los detectores y descriptores más robustos. Una introducción a los modelos de atención visual puede ser encontrada en [13].
  - **Uso del algebra y grupos de Lie:** Las posiciones de la cámara y su movimiento estan descritos por una transformación de cuerpo rígido que usa doce parámetros (matriz de rotación y vector de traslación), es una representación sobreparametrizada ya que al final sólo se necesitan seis parámetros: tres que describen la posición y los otros tres que describen la orientación. Con el uso del algebra de Lie [4] se puede reducir este número de parámetros, con esto se incrementaría la velocidad de convergencia en el Bundle Adjustment ya que el espacio de búsqueda tendría menos dimensiones.

- **Capacidad de adaptar los parámetros dinámicamente:** Como se mencionó con anterioridad hay muchos parámetros que se deben de especificar dentro del algoritmo, el ajuste de ellos puede hacer al algoritmo más robusto, o más preciso, o más rápido dependiendo del entorno en el que trabaje. Un algoritmo que sea capaz de evaluar las condiciones del entorno puede ajustar sus parámetros dinámicamente para tener un mejor desempeño. Existe muy poco trabajo documentado que trate este problema, sin embargo considero importante estudiar posibles soluciones e implementaciones.
- **Ejecución en tiempo constante:** Es deseabla que el tiempo de ejecución del algoritmo por frame sea constante, ésto con el objetivo de poder conocer ese intervalo de tiempo entre frames para más adelante poder estimar la velocidad y la aceleración del robot.
- **Usar las restricciones no holonómicas del robot:** Como se plantea en [38] se pueden reducir el número de parámetros si se explotan las restricciones no holonómicas del robot. Es decir, si conocemos cómo se puede mover el robot podemos restringir las variables usadas en la estimación del sistema. Esto incrementará tanto el tiempo de ejecución como la su precisión.

Bibliografía

[1] M Achtelik, S Lynen, S Weiss, L Kneip, M Chli, and R Siegwart. Visual-inertial slam for a small helicopter in large outdoor environments. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012. Video: http://margaritachli.com/videos/IROS2012video.mp4.

[2] Markus Achtelik, Michael Achtelik, Yorick Brunet, Margarita Chli, Savvas A. Chatzichristofis, Jean-Dominique Decotignie, Klaus-Michael Doth, Friedrich Fraundorfer, Laurent Kneip, Daniel Gurdan, Lionel Heng, Elias B. Kosmatopoulos, Lefteris Doitsidis, Gim Hee Lee, Simon Lynen, Agostino Martinelli, Lorenz Meier, Marc Pollefeys, Damien Piguet, Alessandro Renzaglia, Davide Scaramuzza, Roland Siegwart, Jan Stumpf, Petri Tanskanen, Chiara Troiani, and Stephan Weiss. sfly. `http://www.sfly.org`, 2009-2011. En línea; consultado 12-enero-2013.

[3] Markus Achtelik, Michael Achtelik, Yorick Brunet, Margarita Chli, Savvas A. Chatzichristofis, Jean-Dominique Decotignie, Klaus-Michael Doth, Friedrich Fraundorfer, Laurent Kneip, Daniel Gurdan, Lionel Heng, Elias B. Kosmatopoulos, Lefteris Doitsidis, Gim Hee Lee, Simon Lynen, Agostino Martinelli, Lorenz Meier, Marc Pollefeys, Damien Piguet, Alessandro Renzaglia, Davide Scaramuzza, Roland Siegwart, Jan Stumpf, Petri Tanskanen, Chiara Troiani, and Stephan Weiss. Sfly: Swarm of micro flying robots. In *IROS*, pages 2649–2650, 2012.

[4] Motilal Agrawal. A lie algebraic approach for consistent pose registration for general euclidean motion. In *IROS*, pages 1891–1897, 2006.

[5] Motilal Agrawal, Kurt Konolige, and Morten Rufus Blas. Censure: Center surround extremas for realtime feature detection and matching. In *ECCV (4)*, pages 102–115, 2008.

[6] Alexandre Alahi, Raphael Ortiz, and Pierre Vandergheynst. Freak: Fast retina keypoint. In *CVPR*, pages 510–517, 2012.

[7] Herbert Bay, Tinne Tuytelaars, and Luc J. Van Gool. Surf: Speeded up robust features. In *ECCV (1)*, pages 404–417, 2006.

[8] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *ECCV (4)*, pages 778–792, 2010.

[9] Sunglok Choi, Taemin Kim, and Wonpil Yu. Performance evaluation of ransac family. In *BMVC*, pages 1–12, 2009.

[10] Mark Joseph Cummins and Paul M. Newman. Fab-map: Appearance-based place recognition and mapping using a learned visual vocabulary model. In *ICML*, pages 3–10, 2010.

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

[12] Friedrich Fraundorfer, Davide Scaramuzza, and Marc Pollefeys. A constricted bundle adjustment parameterization for relative scale estimation in visual odometry. In *ICRA*, pages 1899–1904, 2010.

[13] Simone Frintrop, Erich Rome, and Henrik I. Christensen. Computational visual attention systems and their cognitive foundations: A survey. *TAP*, 7(1), 2010.

[14] Steffen Gauglitz, Tobias Höllerer, and Matthew Turk. Evaluation of interest point detectors and feature descriptors for visual tracking. *International Journal of Computer Vision*, 94(3):335–360, 2011.

[15] Steffen Gauglitz, Chris Sweeney, Jonathan Ventura, Matthew Turk, and Tobias Höllerer. Live tracking and mapping from both general and rotation-only camera motion. In *ISMAR*, pages 13–22, 2012.

[16] Google. Ceres-solver. `http://code.google.com/p/ceres-solver`, 2013. En línea; consultado 7-abril-2013.

[17] The Oxford Mobile Robotics Group. Robotcar. `http://mrg.robots.ox.ac.uk/robotcar`, 2012. En línea; consultado 20-mayo-2013.

[18] Chris Harris and Mike Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988.

[19] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[20] Richard I. Hartley and Peter F. Sturm. Triangulation. *Computer Vision and Image Understanding*, 68(2):146–157, 1997.

[21] Willow Garage Intel Corporation and Itseez. Opencv. `http://opencv.org/`, 2013. En línea; consultado 27-mayo-2013.

[22] Georg Klein and David W. Murray. Parallel tracking and mapping for small ar workspaces. In *ISMAR*, pages 225–234, 2007.

[23] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate $o(n)$ solution to the pnp problem. *International Journal of Computer Vision*, 81(2):155–166, 2009.

[24] Stefan Leutenegger, Margarita Chli, and Roland Siegwart. Brisk: Binary robust invariant scalable keypoints. In *ICCV*, pages 2548–2555, 2011.

[25] Hongdong Li and Richard I. Hartley. Five-point motion estimation made easy. In *ICPR (1)*, pages 630–633, 2006.

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

[27] O. Pizzaro S. Williams B. Upcroft M. Warren, P. Corke. https://wiki.qut.edu.au/display/cyphy/Ben+Upcroft En línea; consultado 25-mayo-2013.

[28] José Martínez-Carranza and Andrew Calway. Efficient visual odometry using a structure-driven temporal map. In *ICRA*, pages 5210–5215, 2012.

[29] Hans P. Moravec. Towards automatic visual obstacle avoidance. In *IJCAI*, page 584, 1977.

[30] Hans P. Moravec. Rover visual obstacle avoidance. In *IJCAI*, pages 785–790, 1981.

[31] E. Mouragnon, Maxime Lhuillier, Michel Dhome, Fabien Dekeyser, and Patrick Sayd. Real time localization and 3d reconstruction. In *CVPR (1)*, pages 363–370, 2006.

[32] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP'09)*, pages 331–340. INSTICC Press, 2009.

[33] David Nistér and Henrik Stewénius. Scalable recognition with a vocabulary tree. In *CVPR (2)*, pages 2161–2168, 2006.

[34] David Nistér, Oleg Naroditsky, and James R. Bergen. Visual odometry. In *CVPR (1)*, pages 652–659, 2004.

[35] Illah R. Nourbakhsh Roland Siegwart and Davide Scaramuzza. *Introduction to Autonomous Mobile Robots*. The MIT Press, second edition, 2011.

[36] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *ECCV (1)*, pages 430–443, 2006.

[37] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary R. Bradski. Orb: An efficient alternative to sift or surf. In *ICCV*, pages 2564–2571, 2011.

[38] Davide Scaramuzza. 1-point-ransac structure from motion for vehicle-mounted cameras by exploiting non-holonomic constraints. *International Journal of Computer Vision*, 95(1):74–85, 2011.

[39] Davide Scaramuzza and Friedrich Fraundorfer. Visual odometry [tutorial]. *IEEE Robot. Automat. Mag.*, 18(4):80–92, 2011.

[40] Davide Scaramuzza, Friedrich Fraundorfer, and Roland Siegwart. Real-time monocular visual odometry for on-road vehicles with 1-point ransac. In *ICRA*, pages 4293–4299, 2009.

[41] J. Shi and C. Tomasi. Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[42] Frank Steinbrücker, Jürgen Sturm, and Daniel Cremers. Real-time visual odometry from dense rgb-d images. In *ICCV Workshops*, pages 719–722, 2011.

[43] Niko Sünderhauf, Kurt Konolige, Simon Lacroix, and Peter Protzel. Visual odometry using sparse bundle adjustment on an autonomous outdoor vehicle. In *AMS*, pages 157–163, 2005.

[44] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle adjustment - a modern synthesis. In *Workshop on Vision Algorithms*, pages 298–372, 1999.

[45] Christopher Zach. Simple sparse bundle adjustment. `http://www.inf.ethz.ch/personal/chzach/opensource.html`, 2011. En línea; consultado 5-Diciembre-2012.