Universidad Miguel Hernández de Elche

MÁSTER UNIVERSITARIO EN ROBÓTICA

"Desarrollo de algoritmos de odometría visual mediante una cámara de 360 grados"

Trabajo de Fin de Máster

Curso académico 2017/2018

Autor: María Flores Tenza
Tutor/es: David Valiente García
Luis Payá Castelló

Índice general

Índice de figuras

Índice de tablas

Capítulo 1
Introducción

La utilización de los robots móviles en la sociedad se ha incrementado exponencialmente en los últimos años. En este sentido, el hecho de que no se encuentren fijos y puedan navegar por el entorno en el que se encuentran ha provocado el aumento los ámbitos en lo que se puede emplear la robótica, como por ejemplo en ambientes de difícil acceso para el ser humano: misiones espaciales (mantenimiento de estaciones orbitales, etc.), tareas submarinas (mediciones, misiones de búsqueda y rescate, etc.) o subterráneas (minería, construcción de túneles, etc.), en vigilancia e intervención de seguridad (desactivación de explosivos, operación en zonas radioactivas, etc.) y aplicaciones militares [47]. En la actualidad, este tipo de robot se está utilizando también en el hogar para la realización de tareas domésticas, entretenimiento, e igualmente como asistentes robóticos para personas de mayor edad [15].

Por tanto, se puede decir que los robots móviles son vehículos autónomos que se desplazan por un entorno y realizan al mismo tiempo una tarea. El hecho de que presenten navegación autónoma hace que tengan la capacidad de planificar y seguir una trayectoria de forma óptima evitando los obstáculos presentes en su entorno de trabajo [22]. El problema de la navegación se puede dividir en cuatro etapas: (a) percepción, el robot extrae información relevante mediante los sensores, (b) localización, calcula su posición en el mapa, (c) planificación de ruta, el robot define la trayectoria que debe seguir para llegar a su objetivo y (d) control de movimiento, regula su movimiento para seguir la trayectoria deseada [33]. El sistema de percepción tiene como objetivos permitir que el robot navegue de forma segura detectando y localizando los obstáculos que pueda encontrar, modelar el entorno y conocer su posición. Algunos de los sensores que se pueden utilizar para este fin son sonar, ultrasónico, laser o cámaras.

La utilización de éstos últimos se ha visto incrementada debido a las ventajas que presenta. De hecho, se aplica en distintos ámbitos de la robótica móvil, como la localización, navegación visual, odometría visual, SLAM, etc. Los sistemas de visión ofrecen abundante información sobre la escena, siendo similar a la que obtendría el ojo humano. Entre otras de sus ventajas se encuentran su bajo coste y que son fáciles de usar, además son más eficientes en términos de consumo que otros sensores, por lo que son una buena opción en tareas que requieran un elevado tiempo para su realización. Se pueden encontrar distintos tipos de cámaras, sin embargo, en los últimos años las cámaras omnidireccionales son las más utilizadas ya que la mayor ventaja que presenta, frente al resto, es su gran campo de visión pudiendo incluir toda la escena en una única imagen.

La información visual ofrecida por las imágenes se puede extraer mediante métodos basados en características locales o basados en características globales. El primer modo

consiste en obtener un conjunto de puntos pertenecientes a cada imagen con unas determinadas características.

Sin embargo, los métodos basados en la apariencia global utilizan las imágenes en su conjunto, no extrae ninguna información local. Se calculará el descriptor, que contendrá información sobre su apariencia global, para cada imagen. Este tipo de métodos presenta algunas ventajas en aquellos entornos dinámicos o mal estructurados [46].

1.1 Objetivos

El objetivo de este trabajo es estudiar el comportamiento de la cámara VIRB 360 como sensor para obtener el movimiento de un robot móvil.

En primer lugar, se obtendrán una serie de imágenes tomada a un patrón de calibración. El siguiente paso, por tanto, será obtener los parámetros que modelan la cámara mencionada. Se realizará utilizando distintos métodos de calibración, para finalmente escoger los parámetros del método con el que se haya conseguido un menor error de reproyección.

Tras haber calibrado la cámara, se realizará una base de datos con imágenes tomadas sobre una rejilla, moviendo la cámara sobre ella. El siguiente paso es calcular el movimiento de la cámara entre una vista y otra, y posteriormente, dado que se conoce cuanto se ha movido la cámara para obtener las imágenes, comparar los resultados obtenidos por la odometría con los reales. A la hora de obtener el movimiento se utilizarán distintos detectores de puntos característicos y se estudiará con cuál se obtiene menos error.

1.2 Apartados

El presente trabajo se encuentra estructurado de la siguiente manera:

- Capítulo 2: se realiza una breve introducción de las aplicaciones de la visión en el ámbito de la robótica móvil. Para ello se describen algunas de ellas como por ejemplo la odometría visual, el SLAM y el control visual.

- Capítulo 3: en este capítulo se habla de forma más detallada sobre los sistemas de visión, concretamente de los omnidireccionales ya que la cámara utilizada en este trabajo presenta esta configuración. También se encuentra un apartado en el que se explican los distintos tipos de proyección. De hecho, en este mismo capítulo se hablará sobre el sistema de visión utilizado, que es la cámara VIRB 360.

- Capítulo 4: está dedicado a la calibración, dado que es un paso importante que se debe realizar antes de la odometría visual o de otras técnicas en las que se utiliza imágenes como información con el fin de obtener medidas. Se detallan algunos de los métodos que existen y además se han destinado dos apartados de este capítulo para presentar los dos modelos para cámara con lente de ojo de pez que se van a utilizar: Scaramuzza y Mei.

- Capítulo 5: se introduce la geometría epipolar, ya que en ella se basa la odometría visual y se habla de forma más detalla de ésta. En el apartado dedicado a la odometría visual se explican los pasos que se deben llevar a cabo. Además, dado que uno de ellos consiste en extraer características de las imágenes y posteriormente obtener los descriptores, se explican los métodos utilizados en este trabajo: SURF, BRISK, KAZE y HARRIS.

- Capítulo 6: una vez comentados, en los capítulos anteriores, de forma teórica todos los conceptos, en este se explica cada uno de los pasos que se han implementado en este trabajo. Se encuentra dividido mediante dos apartados, el primero dedicado a la calibración en el que se describen los pasos que se han llevado a cabo a la hora de calibrar la cámara con las distintas aplicaciones. Por último, los pasos realizados para obtener las poses relativas de la cámara.

- Capítulo 7: se muestran todos los resultados que se han obtenido tras haber realizado lo expuesto en el capítulo anterior. Se encuentra también dividido en dos apartados, uno para la calibración y otro para la odometría visual.

- Capítulo 8: se exponen las conclusiones extraídas, así como las posibles mejoras que se podrían realizar en un futuro.

Capítulo 2
Robótica visual

Una gran parte de la investigación y desarrollo de la robótica actual se basan en técnicas que proporcionen a los robots un mayor grado de autonomía y versatilidad. Durante años, los robots han utilizado en mayor o menor medida de visión para la navegación. De hecho, se puede decir que el primer intento de utilizar la visión artificial fue con el robot llamado Shakey [44] seguido del Stanford Cart [41]. Este último disponía de un sistema de visión compuesto por dos cámaras una de las cuales podía moverse para proporcionar distintos puntos de vista para la visión estereoscópica. Debido a la poca potencia informática que existía en ese momento, fue lento y las ejecuciones de prueba tardaron mucho tiempo [34].

El hecho de incorporar visión en robots les otorga la capacidad de percibir visualmente el medio ambiente e interactuar con él. La robótica visual hace uso de los métodos de visión por computador para llevar a cabo las tareas que debe realizar el robot. Las tareas típicas son la navegación hacia una ubicación determinada evitando obstáculos y reconocer objetos, entre otras. El objetivo de la visión por computador es intentar entender la escena y los objetos que se encuentran, semejándose a la función que realiza el ser humano. La visión se ha utilizado en aplicaciones robóticas durante años, sobre todo en entornos industriales, servicios médicos y robótica acuática.

Así mismo, la visión se utiliza en diversas técnicas de la robótica, debido al gran contenido de información que proporcionan sobre la escena. A continuación se muestran algunas de ellas junto con algunos de los trabajos que se encuentran en la literatura: SLAM (Kim et al. [28] y Lukierski et al [32]), navegación de un robot (Zhang et al. [64] y Hart et al. [24]), control visual (Liu et al. [30] y Markovic et al.[36]) y structure from motion (Micusik y Pajdla [39] y Kawanishi et al. [27]).

2.1 Odometría Visual

Se denomina odometría visual al proceso de estimación del movimiento del robot – rotación y translación con respecto a un marco de referencia – mediante la observación de una secuencia de imágenes tomadas de su entorno [63].

Este término fue propuesto por Niestér [45] en 2004 debido a su gran similitud con la odometría, que utiliza datos proporcionados por los encoders de las ruedas para estimar el movimiento. Sin embargo, se puede decir que la odometría visual presenta mayores ventajas, ya que, por ejemplo, no se ve afectada el deslizamiento de la rueda en un terreno desigual u otras condiciones adversas. De hecho, se ha comprobado que la odometría

visual proporciona una estimación de trayectorias más precisa, con un rango de error en la posición relativa de 0,1% a 2%. Por esta razón, en algunas ocasiones resulta interesante utilizarla como complemento para la odometría mediante encoders, GPS, unidades de medidas inerciales (IMUs) o odometría mediante láser (estima el movimiento mediante escaneos de laser consecutivos) [52].

El problema de estimar el movimiento de un robot mediante la observación de una secuencia de imágenes comenzó en la década de 1980 por Moravec [40]. En su trabajo, utilizó una cámara que se deslizaba sobre un raíl en dirección perpendicular con respecto al movimiento del robot, tomando imágenes en intervalos equidistantes. Las características de las imágenes se obtuvieron mediante la detección de esquinas, y posteriormente se buscaba correspondencias a lo largo de las líneas epipolares, utilizando correlación cruzada. A partir de ese año hasta el 2000, las investigaciones sobre la odometría visual fueron impulsadas por la NASA para una misión espacial en Marte [12]. Scaramuzza y Siegwart [55] realizaron una investigación basada en el uso de odometría visual en vehículos terrestres que navegaban por entornos al aire libre, para ello disponían de una cámara omnidireccional. El proceso que llevaron a cabo en su trabajo se encontraba constituido por dos partes. Por un lado, extraían características de tipo SIFT y eliminaban las falsas correspondencias mediante RANSAC, mientras que por otro lado utilizaba un método de apariencia global. Finalmente fusionaba las estimaciones obtenidas por cada método utilizado.

En los últimos años la odometría visual ha ganado importancia en una amplia gama de aplicaciones. Se utiliza mayormente en vehículos terrestres, misiones espaciales, vehículos aéreos y submarinos.

La aplicación de la odometría en vehículos aéreos no tripulados (UAV) es poder realizar despegues y aterrizajes autónomos y navegación punto a punto. En los vehículos autónomos submarinos su aplicación juega un papel importante, ya que ni pueden depender del GPS para la estimación de su posición. Esto se debe a que en ocasiones la señal se degrada o deja de estar disponible en estos entornos. Siendo la odometría visual una solución rentable.

2.2 SLAM Visual

El SLAM (*simultaneous localization and mapping*) es una técnica que se emplea cuando un robot móvil autónomo se desplaza por un entorno desconocido y construye de forma incremental un mapa de este entorno al mismo tiempo que utiliza este mapa para obtener su ubicación absoluta, obteniéndose el inconveniente de que el ruido en la estimación de la pose del robot lleva al ruido en la estimación del mapa y viceversa [21].

Por tanto, se puede decir que el SLAM engloba la realización de dos tareas. Por un lado, construir un mapa del entorno en el que se encuentra a priori desconocido. Por otro, estimar su trayectoria al desplazarse dentro de este entorno. En suma, podemos señalar que el SLAM consiste en que el robot pueda crear un mapa y localizarse en el de forma simultánea.

El problema clásico de localización consiste en obtener, dado un mapa conocido, la posición del robot. Por el contrario, en el problema de creación de mapas lo que se pretende es, sabiendo en todo momento cual es la posición del robot, tomar datos con los sensores de medida para obtener información del entorno y construir el mapa. En cambio, se puede decir que el SLAM engloba ambos problemas por lo que la principal dificultad de esto es que no se tiene conocimiento ni de la posición ni del entorno, únicamente se conoce el control y las observaciones las cuales presentan ruido. Así, se puede señalar que SLAM engloba la odometría, la predicción y extracción de *landmarks*, la asociación de datos, la estimación de posición y actualización del mapa. Estos procesos componen la raíz del algoritmo y se llevan a cabo de forma cíclica.

En otro orden de cosas, se han utilizado diversos sensores para resolver el problema de SLAM. Sin embargo, recientemente ha aumentado el interés en SLAM basado es imágenes, conocido como SLAM visual. Esto se debe a la gran información visual que proporcionan este tipo de sensores. El único inconveniente es que requiere de un mayor coste computacional y de algoritmos sofisticados para poder procesar las imágenes y extraer de ellas las características. Sin embargo, debido a los avances tecnológicos es posible la implementación en tiempo real de los algoritmos requeridos [63].

Se pueden encontrar una gran cantidad de trabajos que utilizan esta técnica. Por ejemplo, Davison et al. [13] en su trabajo construyeron un mapa extrayendo características dispersas de imágenes tomadas con una cama monocular, y combinando las nuevas características obtenidas con las observadas utilizando una correlación de diferencia de suma de cuadrados normalizada. Otro trabajo es el de Berenguer et al. [9] que presentan un algoritmo de SLAM el cual utiliza la información visual, extraída mediante descriptores de apariencia global, proporcionada por un sistema catadióptrico.

2.2.1 Breve comparativa entre SLAM visual y odometría visual

En este apartado se va a tratar de trazar las diferencias existentes entre el SLAM visual y la odometría visual. Por un lado, como hemos visto, SLAM es un proceso en el que se requiere que un robot se localice en un entorno desconocido y construya un mapa de este entorno al mismo tiempo, sin ninguna información previa, con la ayuda de sensores externos (o un solo sensor), que en el caso de SLAM visual serían las cámaras. Por su parte, la odometría oisual trata de estimar la pose de un agente utilizando solo la entrada de una o varias cámaras conectadas a ella. Por lo tanto, podemos señalar que el objetivo de SLAM en general (y SLAM Visual en particular) es obtener una estimación global y consistente de la trayectoria del robot. Por otro lado, la finalidad de la odometría visual es recuperar el camino de manera gradual, y potencialmente optimizar sólo en las últimas poses de n trayectorias. Además, en esta última sólo es relevante la consistencia local de la trayectoria y el mapa local se utiliza para obtener una estimación precisa de la trayectoria local (mientras que SLAM se refiere a la consistencia del mapa global). El SLAM, por tanto, engloba la odometría visual, la detección de cierre de ciclos y la optimización gráfica. De forma gráfica, *vid.* la Figura 1.

Si el usuario sólo está interesado en la trayectoria de la cámara y no en el mapa del entorno, se puede usar un método completo de SLAM Visual en lugar de una de las técnicas descritas [52] por Scaramuzza. Un método de SLAM Visual es potencialmente mucho más preciso, porque impone muchas más restricciones en el camino. Sin embargo, no es necesariamente más robusto (por ejemplo, los valores atípicos en el cierre de bucle

pueden afectar gravemente a la consistencia del mapa). Además, es más compleja y computacionalmente más costoso. Al final, la opción entre SLAM visual y odometría visual depende del equilibrio entre el rendimiento y la consistencia y la simplicidad en la implementación. La odometría visual cambia la consistencia del rendimiento en tiempo real, sin necesidad de realizar un seguimiento de la historia anterior de la cámara. En este sentido, aunque esta última no resuelve el problema de la deriva, los investigadores han demostrado que los métodos de odometría visual tienen un rendimiento significativamente mejor que la odometría de las ruedas y las técnicas de cálculo de puntos muertos. Además, el coste de las cámaras es mucho menor en comparación con las IMU precisas y los escáneres laser.

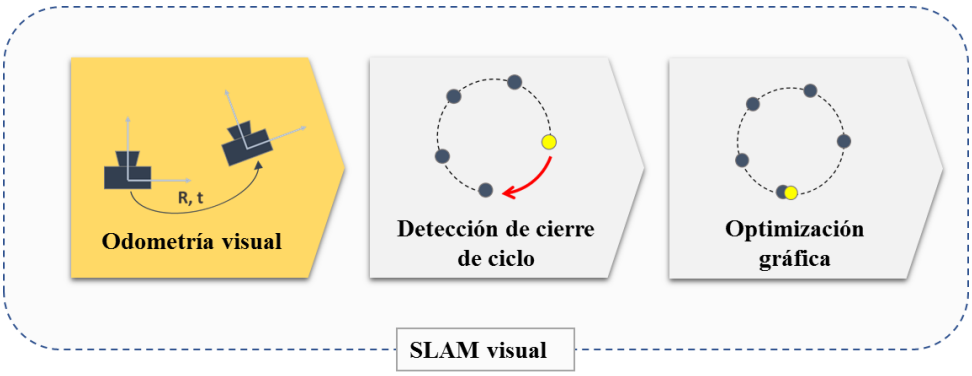


Figura 1: Gráfico de SLAM visual (odometría visual, detección de cierre de ciclo y optimización gráfica)

2.3 Control visual

El control visual es una técnica que utiliza los datos proporcionados por una cámara para controlar el movimiento de un robot. Se basa en técnicas de procesamiento de imágenes, visión por computador y la teoría de control. El hecho de introducir información visual en el esquema de control ha aumentado la flexibilidad y la precisión de una tarea determinada, proporcionando una mayor precisión de posición, robustez al ruido del sensor y la reactividad a los cambios ambientales [25].

La cámara puede estar montada directamente en un manipulador o en un robot móvil, en cuyo caso el movimiento del robot induce el movimiento de la cámara, o la cámara puede fijarse en el área de trabajo para que pueda observar el robot movimiento desde una configuración estacionaria [11]. Estas configuraciones son conocidas como "*eye-in-hand*" y "*fixed camera*" respectivamente. Los esquemas para este control se diferencian básicamente en la forma en la que se encuentra diseñado. En primer lugar, se tiene el control visual basado en imágenes. En este caso el error se calcula de forma directa entre las características extraída de la imagen actual y la deseada. Las mediciones que se utilizan son las coordenadas de píxel de un conjunto de características de imagen.

En segundo lugar, se encuentra el control visual basado en posición. Los datos de visión, en este caso, se utiliza para construir la representación 3D de la escena, es decir, tanto la señal de entrada como la de error se expresa en el espacio cartesiano. Las características que se extraen de la imagen o del modelo 3D del objeto se utiliza para hallar la posición y orientación del objetivo con respecto a la cámara. Con esto se obtiene un error

como resultado de la diferencia que existe entre la pose actual y la deseada en el espacio de trabajo. Por ejemplo, Yahya y Arshad [62] proponen en su trabajo utilizar este tipo de control, visual basado en posición, para el acoplamiento de un AUV.

Por último, existe una combinación de ambos, denominado control visual 2D ½. En este método la entrada se expresa en parte en el espacio cartesiano y en parte en el plano imagen. Se basa en estimar el desplazamiento realizado por la cámara, tanto rotación como translación, entre la vista actual y la deseada de un objeto [35]. Para controlar la rotación de la cámara se utiliza la estimada entre el sistema de referencia actual y el deseado, es decir, esta parte se realiza en el espacio cartesiano. Sin embargo, la translación se expresa en el espacio 2D imagen. Este método se basa en estimar el desplazamiento de forma parcial de la cámara desde la posición actual a la deseada en cada iteración de la ley de control.

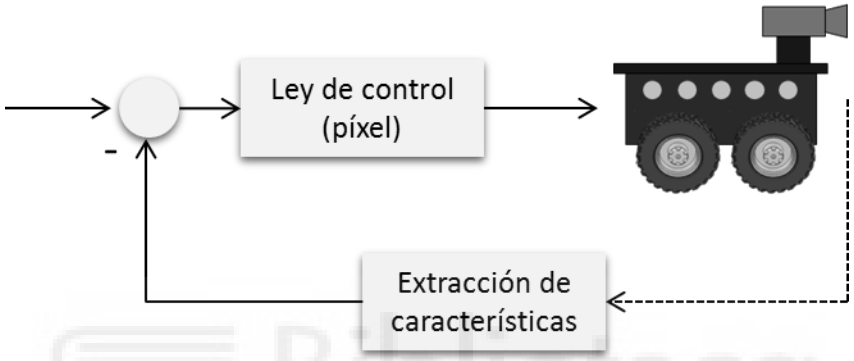


Figura 2: Control visual basado en imágenes.

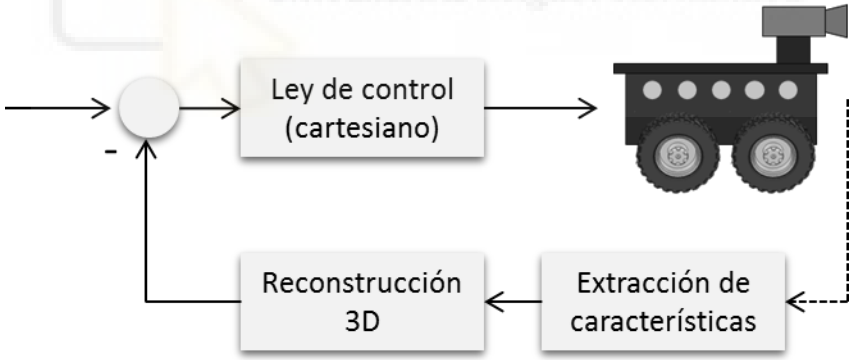


Figura 3: Control visual basado en posición.

Capítulo 3
Sistemas de visión

Los robots móviles requieren ser dotados de sistemas sensoriales para obtener información del entorno, normalmente desconocido a priori, en el que se encuentra y así poder llevar a cabo la tarea propuesta de forma autónoma. Los sistemas sensoriales permiten resolver el problema de localización y posicionamiento presente en la navegación de los robots móviles. En la Tabla 1 se puede ver las ventajas e inconvenientes que presentan los sensores más utilizados en localización [5]

Si bien es cierto que los sensores pueden ser clasificados de distintas formas, la más habitual es aquella que los divide en propioceptivos y exteroceptivos. Los primeros son sensores que miden el estado interno del robot, como por ejemplo el desplazamiento o el nivel de batería. Dentro de los sensores propioceptivos se encuentran los encoders, que son utilizados en la odometría para obtener de forma aproximada la posición del robot respecto de un sistema de referencia inercial. También pertenecen a este tipo de sensores los giróscopos y acelerómetros los cuales proporcionan información sobre la orientación y aceleración. Los sensores propioceptivos tienen como inconveniente la acumulación de errores y por tanto las estimaciones realizadas con estos sensores no son del todo viables, debido a esto en ocasiones se utilizan de forma conjunta con otros sensores. Por otro lado, se encuentran los exteroceptivos, que obtienen información del entorno [3]. Algunos de ellos son: el GPS, el sónar y el láser.

En primer lugar, el *Global Positioning System* (GPS) proporciona una medida de la posición absoluta del robot. Se trata de un sistema de navegación basado en satélites que permite a los usuarios determinar con precisión su ubicación en cualquier punto de la Tierra o ligeramente por encima de ella. En definitiva, consiste en consiste en una constelación nominal de 24 satélites operacionales que orbitan alrededor de la Tierra y transmiten señales codificadas de radiofrecuencia. Están dispuestos de modo que se colocan cuatro satélites en cada uno de los seis planos orbitales para garantizar una cobertura mundial continua. Solo se necesitan cuatro satélites para proporcionar información de ubicación o posicionamiento.

En este sentido, mediante la triangulación los receptores de tierra pueden calcular su posición utilizando el tiempo de viaje de las señales del satélite e información sobre su ubicación actual que se incluye en la señal transmitida. Cada satélite está equipado con un transmisor y receptor de radio y relojes atómicos. Los relojes del receptor no son tan precisos como los relojes atómicos y normalmente muestran un sesgo. Este sesgo genera errores en el tiempo de viaje de las señales y conduce a errores en el cálculo de las distancias a los satélites. Teóricamente, al usar el principio de triangulación, un receptor GPS requiere los rangos de tres satélites solo para calcular la posición 3D (latitud, longitud y

altitud), pero se requiere un cuarto satélite para estimar el desplazamiento del reloj del receptor desde el reloj del sistema y para corregir la polarización del reloj en el receptor.

Funciona correctamente en entornos externos, sin embargo, ocurre lo contrario al utilizarlo en un ambiente interior ya que no proporciona información de posicionamiento de forma precisa. Además, tiene diversos usos: desde la navegación al aire libre hasta la geología, pasando por la agricultura, la construcción o el transporte público. Sus principales ventajas son su inmunidad a la acumulación de errores a lo largo del tiempo y su estabilidad a largo plazo. El GPS es una tecnología revolucionaria para la navegación al aire libre y es efectivo en áreas con una clara vista del cielo, pero no se puede usar en espacios interiores, confinados, subterráneos y subacuáticos. Las limitaciones del GPS incluyen interrupciones causadas por el bloqueo de la señal del satélite, ocasionalmente alto contenido de ruido, efectos de propagación múltiple, bajo ancho de banda e interferencia o atasco. Los cortes de GPS se producen en cañones urbanos, túneles y otros entornos y lugares restringidos por GPS.

En otro orden de cosas, podemos diferenciar distintas clases de GPS con distintas funciones. Por un lado, el GPS autónomo común se utiliza para el posicionamiento y tiene una precisión de 10 m. Por otro lado, el GPS diferencial y el GPS cinemático en tiempo real se inventaron para mejorar la precisión del GPS y permitir la localización en entornos de campo abierto al aire libre dentro de un orden de metro o centímetro. Son técnicas de posicionamiento relativo que emplean dos o más receptores simultáneamente para rastrear los mismos satélites. El GPS diferencial consiste principalmente en tres elementos: un receptor GPS (estación base) ubicado en una ubicación conocida, un receptor GPS (receptor de usuario) y un medio de comunicación de radio entre estos dos receptores. El mismo puede corregir los errores de polarización del receptor del usuario mediante el uso de errores de polarización medidos en la estación base. Por su parte, el GPS cinemático en tiempo real proporciona mediciones en tiempo real con precisión de centímetro. La primera solución requiere un mínimo de cuatro satélites comunes y proporciona un rango de precisión de aproximadamente 20 cm a 1 m. La segunda solución requiere al menos cinco satélites comunes y proporciona una precisión de 2 cm.

Por otro lado, el sónar (*Sound Navigation And Ranging*) es una técnica que usa la propagación del sonido bajo el agua (principalmente) para navegar, comunicarse o detectar objetos sumergidos. Generalmente, se emplea como un medio de localización acústica emitiendo impulsos sonoros (a diferencia del radar que utiliza ondas electromagnéticas). Así, utilizan energía acústica para detectar objetos y medir distancias desde el sensor hasta los objetos objetivo. El sensor mide el tiempo de vuelo (TOF), que es el tiempo desde la transmisión de la señal hasta la recepción. Dado que se conoce la velocidad de transmisión de una señal ultrasónica, se puede calcular la distancia al objetivo que refleja la señal. Los sensores de sónar se pueden utilizar para localizar robots móviles a través del emparejamiento o triangulación del modelo calculando el cambio de postura entre cada dos entradas de sensores adquiridas en dos poses diferentes. Tienen dos partes principales, esto es, transmisor y receptor. El primero de ellos envía un pulso ultrasónico corto. Por su parte, el receptor recibe lo que regresa de la señal una vez que se ha reflejado en los objetos cercanos.

El mayor inconveniente de estos sensores es el reflejo de ondas de señal que dependen en gran medida del material y la orientación de la superficie del objeto. Además, son sensibles al ruido del entorno y a otros robots que usan ultrasonidos con la misma frecuencia. Se supone que muchos objetos en el entorno son reflectores especulares para

ondas ultrasónicas, lo que hace que un sensor de sonda reciba un eco reflejado en lugar del primero.

En tercer y último lugar, el láser usa la emisión inducida o estimulada para generar un haz de luz coherente tanto espacial como temporalmente. La coherencia espacial se corresponde con la capacidad de un haz para permanecer con un pequeño tamaño al transmitirse por el vacío en largas distancias y la coherencia temporal se relaciona con la capacidad para concentrar la emisión en un rango espectral muy estrecho.

Los sensores láser se pueden utilizar en varias aplicaciones relacionadas con el posicionamiento. Es una tecnología de teledetección para la medición de distancia que implica transmitir un láser hacia el objetivo y luego analizar la luz reflejada. Las mediciones de alcance basadas en láser dependen de las técnicas TOF o de desplazamiento de fase. Al igual que el sensor de sonda, en un sistema TOF, se envía un pulso corto de láser y se mide el tiempo hasta que vuelve. Este tipo de sensor a menudo se denomina radar láser o detector de luz y sensor de rango (LIDAR). Sin embargo, en los sistemas de desplazamiento de fase, se transmite una señal continua. La fase de la señal devuelta se compara con una señal de referencia generada por la misma fuente. La velocidad del objetivo y la distancia a ella se miden con el desplazamiento Doppler.

LIDAR se usa principalmente en detección y evitación de obstáculos, mapeo y captura de movimiento 3D. LIDAR puede integrarse con GPS e INS para mejorar la precisión de las aplicaciones de posicionamiento al aire libre. Aunque los sensores de sonda tienen un gran ancho de haz que permite una mayor cobertura, la resolución angular con un escáner láser es mucho mejor que la de uno ultrasónico. Una desventaja de LIDAR en comparación con los sensores de sonda es que implica una solución muy costosa. Además, el análisis de datos LIDAR tiene un alto costo computacional y puede afectar la respuesta de las aplicaciones en tiempo real. La manera iterativa de calcular la correspondencia óptima entre dos escaneos láser aumenta el costo computacional. Además, el escaneo puede fallar cuando el material parece transparente para el láser, como el vidrio, porque las reflexiones sobre estas superficies dan lugar a datos sospechosos [5].

3.1 Sensores de visión

Otro tipo de sensor exteroceptivo es el de visión, se puede decir que estos sensores presentan una gran ventaja y es que proporcionan una gran cantidad de información sobre el entorno. Algunas de las tareas que se pueden realizar con estos sensores son extraer características, estimar la posición o reconocer patrones entre otras. Una cámara es un mecanismo de formación de imágenes como resultado de que la luz se proyecte sobre una superficie sensible a la misma. Se pueden obtener diferentes cámaras variando tres elementos: (1) la geometría de la superficie, (2) la distribución geométrica y propiedades ópticas de los fotorreceptores, y (3) la forma en la que se recoge y proyecta la luz sobre la superficie (lentes simples o múltiples). Los sistemas de visión procesan estas imágenes para reconocer, navegar, y generalmente interactuar con el entorno [43]. Las cámaras presentan diversas ventajas en relación con los sensores ya descritos, como su bajo coste, peso y consumo de energía, siendo esta última una característica de gran utilidad para los robots autónomos, además de la alta información que ofrecen del entorno.

Tabla 1. Comparación de algunos de los sensores utilizados para la localización [5].

Sensor	**Ventajas**	**Desventajas**
Odometría	Simple determinar la posición orientación. Precisión a corto plazo, y permite altas tasas de muestreo. Solución a bajo coste.	Desviación de la posición debido al deslizamiento de la rueda. Acumulación de errores en el tiempo. La estimación de la velocidad requiere una diferenciación numérica que produce ruido adicional.
GPS	Posición absoluta con valor de error conocido. Sin acumulación de errores en el tiempo.	No disponible en áreas interiores, bajo el agua y cerradas.
Sónar	Proporciona una medida de distancia escalar del sensor al objeto. Solución económica	La reflexión de la onda de señal depende del material u orientación de la superficie del obstáculo. Sufre de interferencia si se utilizan múltiples sensores. Baja resolución angular y velocidad de exploración
Láser	Similar a los sensores de sonda, pero tiene una mayor precisión y velocidad de escaneo. Devuelva la distancia a un único punto (telémetro) o una serie de distancias (escáner).	La reflexión de la onda de señal depende del material u orientación de la superficie del obstáculo. Solución costosa
Cámara	Las imágenes almacenan una gran información significativa. Proporcionan una alta precisión de localización. Solución económica	Requiere técnicas de procesamiento de imágenes y extracción de datos. Alto costo computacional para procesar imágenes.

Los sistemas de visión pueden encontrarse en diversas configuraciones en función del número de cámaras utilizadas y del campo de visión que ofrecen. Entre estas configuraciones se encuentran las monoculares, binoculares (estéreo) e incluso trinocular. Estas dos últimas configuraciones permiten medir la profundidad de las imágenes. El inconveniente que presentan es que requieren de varias imágenes para obtener información

completa del entorno. Sin embargo, esto no ocurre con las omnidireccionales que pueden estar compuestas por varias cámaras apuntando hacia distintas direcciones o compuesta por una única cámara y una superficie reflexiva [46].

3.1.1 Omnidireccional

La mayor ventaja que presenta este tipo de cámaras es su gran campo de visión (FOV), permitiendo que en una misma imagen pueda estar incluida toda una escena. Esta es la principal razón por la que ha aumentado el uso de cámaras omnidireccionales en robótica. Sin embargo, no es la única, otra ventaja es que las características que aparecen en las imágenes son más estables, ya que, debido a su amplio campo de visión, a pesar de que el robot se mueva siguen permaneciendo. Los sistemas de visión omnidireccional permiten obtener una descripción más completa del entorno en una única imagen [46].

Los distintos tipos de cámaras omnidireccionales que existen pueden ser clasificadas como central o no central. Se dice que es no central cuando los rayos ópticos procedentes de la cámara y reflejados por la superficie del espejo no se cruzan en un punto único. Por el contrario, serán cámaras centrales cuando cada rayo al reflejarse en la superficie del espejo se cruza en un único punto denominado punto de vista único efectivo.

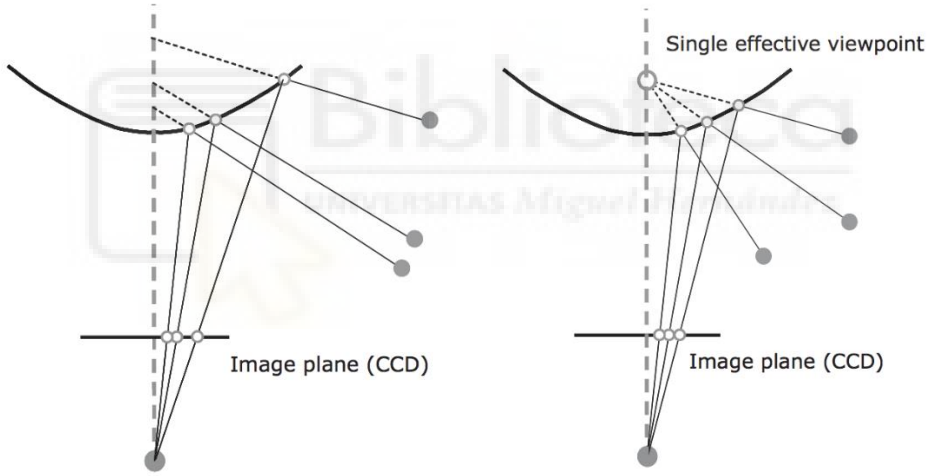


Figura 4: Cámaras omnidireccionales catadióptricas. (a) Sistema de visión no central. (b) Sistema de visión central.

El modelo de una cámara omnidireccional es más complicado que el de una cámara estándar de proyección (modelo Pinhole). Este modelo debe tener en cuenta tanto la reflexión por el espejo en el caso de una cámara catadióptrica como de la refracción causada por las lentes en el caso de las cámaras de ojo de pez. En la literatura se pueden encontrar distintos modelos creados para describir este tipo de cámaras, algunos se verán en el capítulo 4. Las cámaras omnidireccionales pueden obtenerse de distintas formas: sistemas catadióptricos, cámaras con lente de ojo de pez o como combinación de distintas cámaras.

3.1.1.1 Sistemas de visión catadióptricos

Los sistemas de visión catadióptricos son una combinación de cámaras estándar con espejos convexos. Estos sistemas se pueden obtener mediante un espejo hiperbólico y una cámara con lente de proyección perspectiva, o utilizando un espejo parabólico y una cámara con lente de proyección ortográfica. Para este último caso, es necesario incorporar dicha lente entre la cámara y el espejo ya que, al ser parabólico, los rayos reflejados sobre el espejo salen de forma paralela al eje y la lente permitirá que converjan hacia el centro de las cámaras proporcionando una imagen enfocada en el plano imagen. También se pueden encontrar utilizando espejos esféricos, cónicos o elípticos.

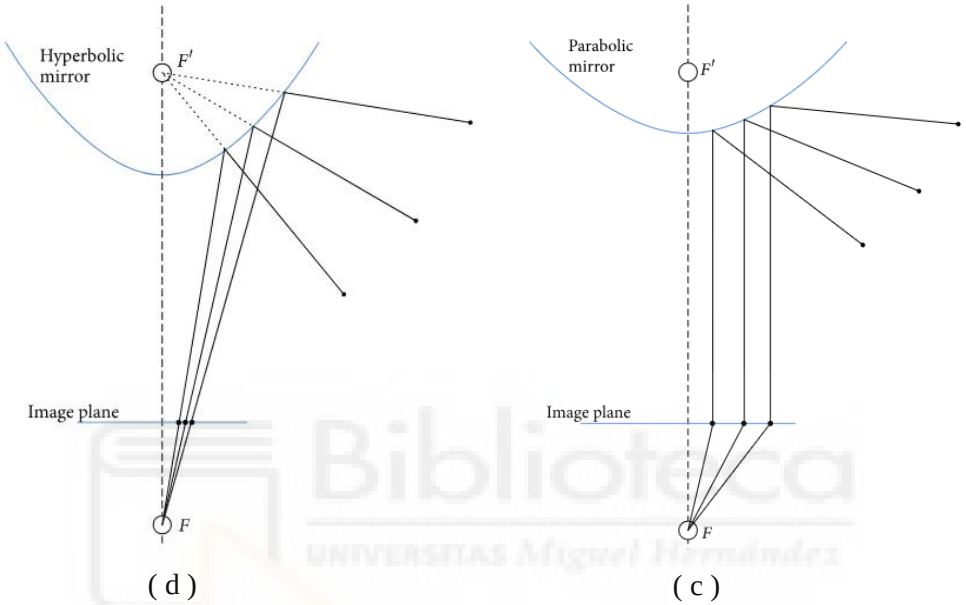


Figura 5: Sistemas de visión catadióptrico: (a) espejo hiperbólico y (b) espejo parabólico.

Payá et al. [47] utilizan en su trabajo únicamente información visual proporcionada por un sistema de este tipo para resolver el problema de SLAM. El método que proponen consta de dos partes, inicialmente el robot incluirá las trayectorias recorridas en el modelo, el cual se construye a partir de imágenes. Después, el robot estimará su pose con respecto a la trayectoria.

3.1.1.2 Cámara con lente de ojo de pez

La combinación de una cámara convencional con lentes de ojo de pez proporciona un campo de visión superior a los 180°. De hecho, usando un conjunto de dos de estas lentes, se obtendría una vista de 360° del entorno. Por ejemplo, un dispositivo que utiliza este sistema es la cámara Gear 360 de Samsung, o la utilizada en este trabajo la cual se describirá en el apartado 3.2. Konstantinos Delibasis et al.[14] utiliza una cámara omnidireccional de este tipo para la localización en tiempo real de robots que se encuentran en entornos interiores.

(a) (b)

Figura 6: (a) Lente ojo de pez. (b) Cámara Gear 360 de Samsung.

3.1.1.3 Múltiples cámaras

Se puede conseguir cámaras omnidireccionales combinando distintas cámaras orientadas hacia distintas direcciones de forma que los campos de visión queden superpuestos. Por ejemplo, la cámara PointGray Ladybug se encuentra compuesta por seis cámaras que recogen aproximadamente el 90% del campo visual del entorno. Por ejemplo, en el trabajo realizado por J. Tardif et al. [57] desarrollan un algoritmo para estimar la trayectoria con precisión sin la necesidad de un modelo de movimiento, el experimento lo realizan utilizando un sistema formado por la cámara anterior montada sobre un vehículo.

Figura 7: Múltiples cámaras. PointGray Ladybug

3.1.1.4 Tipos de proyección

A partir de la imagen omnidireccional se puede obtener distintas representaciones de la escena mediante la proyección de la información visual en distintos planos y geometrías [43].

Imagen esférica

El primer tipo de proyección que se va a explicar consiste en proyectar una imagen omnidireccional sobre una esfera unitaria con centro en el foco del espejo. Cada pixel de la esfera unitaria adquiere el valor del rayo del mundo real que tiene la misma dirección con respecto al foco de la hipérbola. Hay que tener en cuenta que es necesario que, con carácter previo, el sistema visión catadióptrica sea calibrado para obtener la proyección. En la Figura 8 se puede observar el esquema de proyección de los rayos recogidos en el mundo real por el espejo hiperbólico sobre la esfera. Tanto el espejo como el plano

imagen se muestran en color azul, además el punto F corresponde al foco de la cámara mientras que F' al del espejo. Si se ha realizado la calibración del sistema, los píxeles del plano imagen se pueden proyectar en el espejo, tras esto cada punto del espejo pueden ser proyectados sobre una esfera unidad [46].

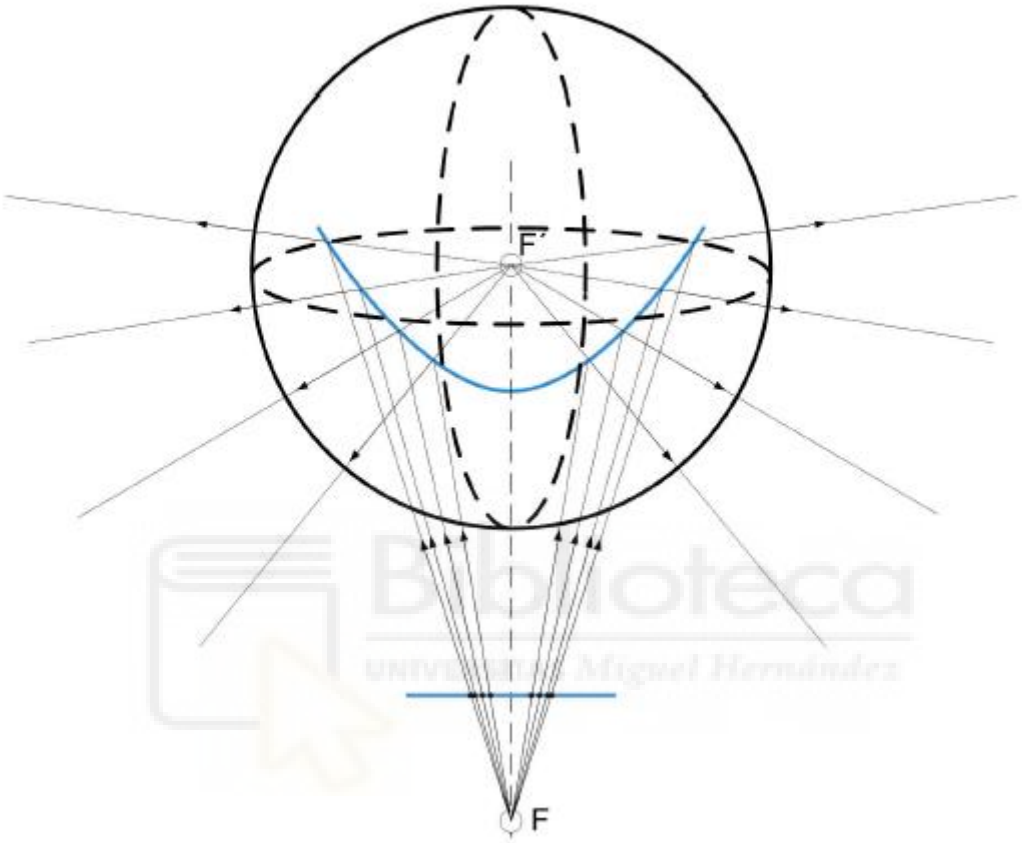


Figura 8: Modelo de proyección: imagen esférica.

Imagen Panorámica

Por otro lado, la representación panorámica es una de las más utilizadas en los trabajos que se centran en la navegación robótica. Ello se debe a que se trata de una representación más comprensible que la omnidireccional y permite aplicar la mayoría de los algoritmos de procesamiento de imágenes pensados para imágenes con proyección perspectiva.

En la Figura 9 se puede observar la proyección de la información omnidireccional sobre una superficie cilíndrica. Tal imagen se forma en función de la geometría del espejo, lo que se puede definir como mapeado polar no lineal, pues las coordenadas radiales dependen de la función de proyección $f(\rho)$.

Para obtener una imagen panorámica, el método más sencillo consiste en modificar el sistema de coordenadas de la imagen omnidireccional (mapeada en coordenadas polares) a un sistema de coordenadas rectangulares. Con esta transformación, cada circunferencia de la imagen se corresponde a una línea horizontal de la imagen panorámica y las

coordenadas radiales de la imagen omnidireccional pasan a ser líneas verticales de la imagen panorámica. No obstante, con este método, se crea una correspondencia entre el radio de la imagen omnidireccional y la altura en la imagen panorámica.

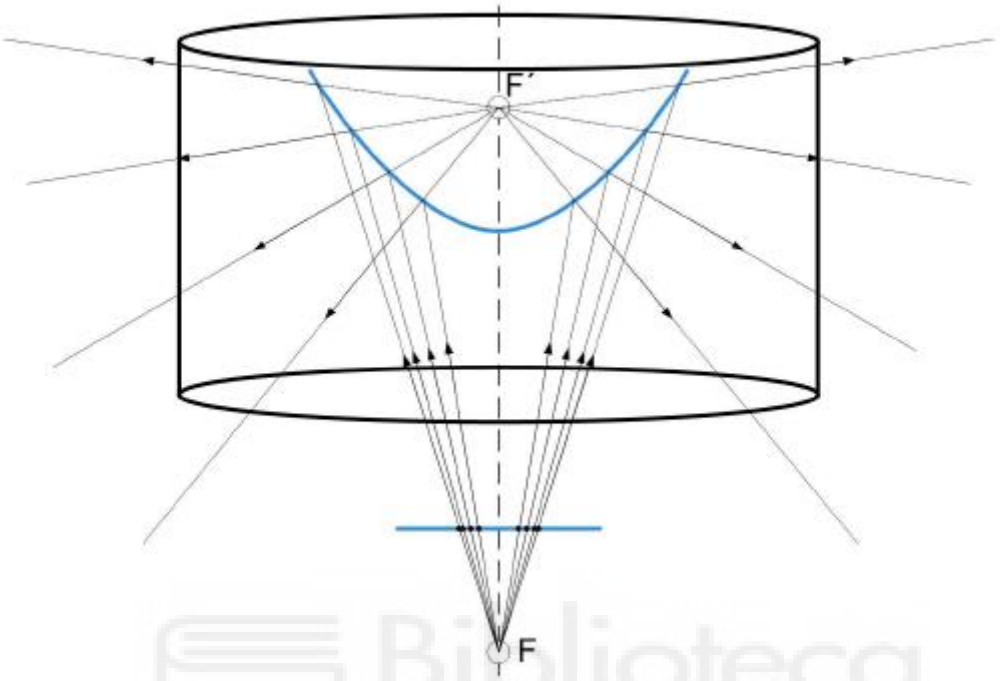


Figura 9: Modelo de proyección: imagen panorámica.

Imagen Perspectiva

En otro orden de cosas, a partir de la información recogida en la imagen omnidireccional y de la calibración del sistema de visión se pueden obtener varias imágenes proyectivas de la escena. La imagen que se obtiene se aproxima a la proporcionada por una cámara convencional situada en el foco del espejo. En la Figura 10 se puede ver el esquema de este modelo de proyección. Los píxeles del plano imagen son proyectados de nuevos al espejo, siendo conocidos los parámetros de calibración del sistema, y tras esto cada punto se proyecta sobre un plano [46].

Imagen Ortográfica

Por último, la imagen ortográfica (también denominada vista de pájaro) puede considerarse un caso particular de proyección perspectiva. El plano de proyección de la imagen está situado perpendicularmente al eje de la cámara para permitir obtener la imagen ortográfica. Esta proyección es equivalente a situar una cámara convencional en el foco del espejo apuntando hacia el plano del suelo, cuando el plano de proyección es paralelo al plano XY. El esquema de este modelo de proyección se muestra en Figura 11, en este caso los pixeles son proyectados en un plano horizontal.

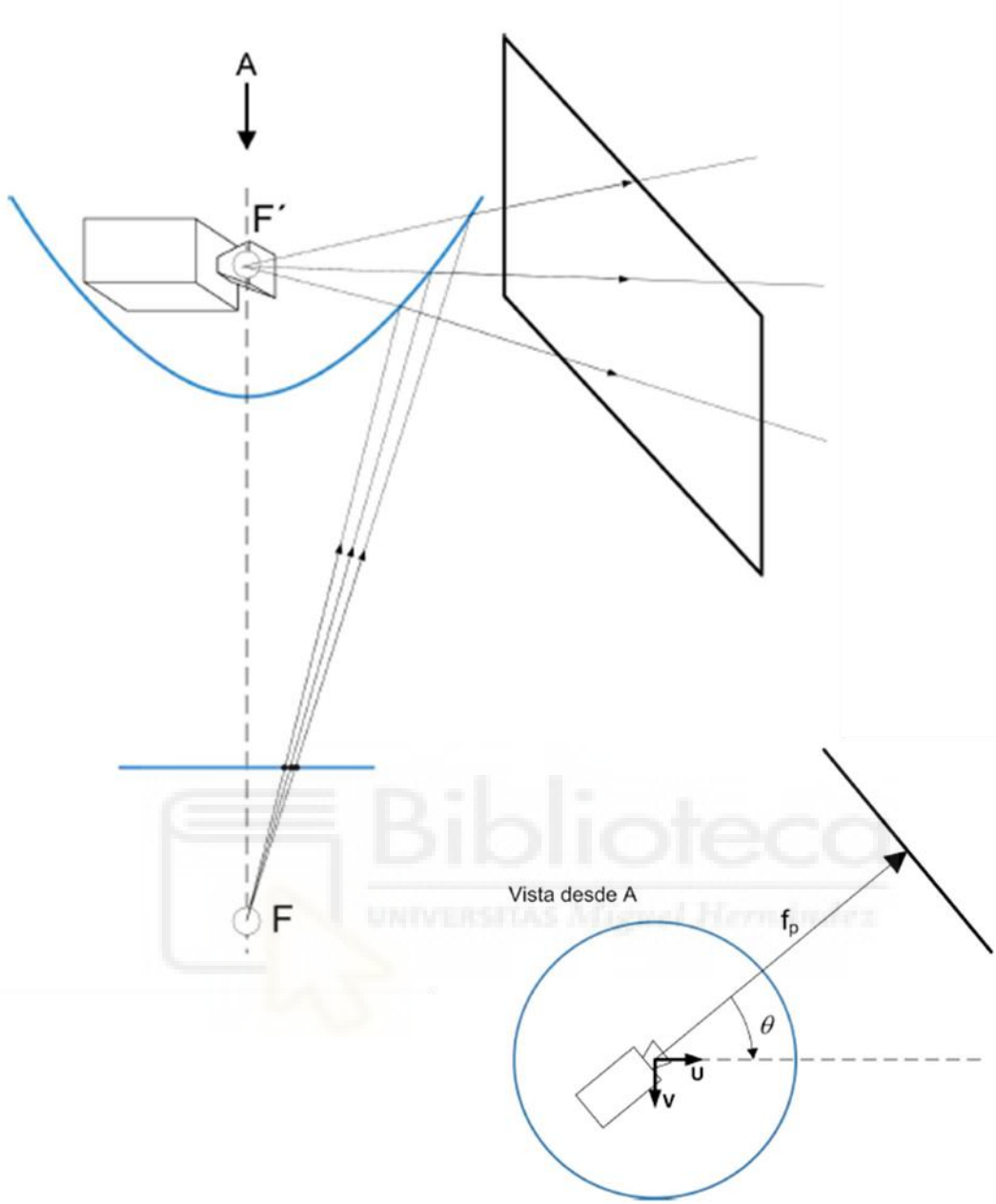


Figura 10: Modelo de proyección: imagen perspectiva.

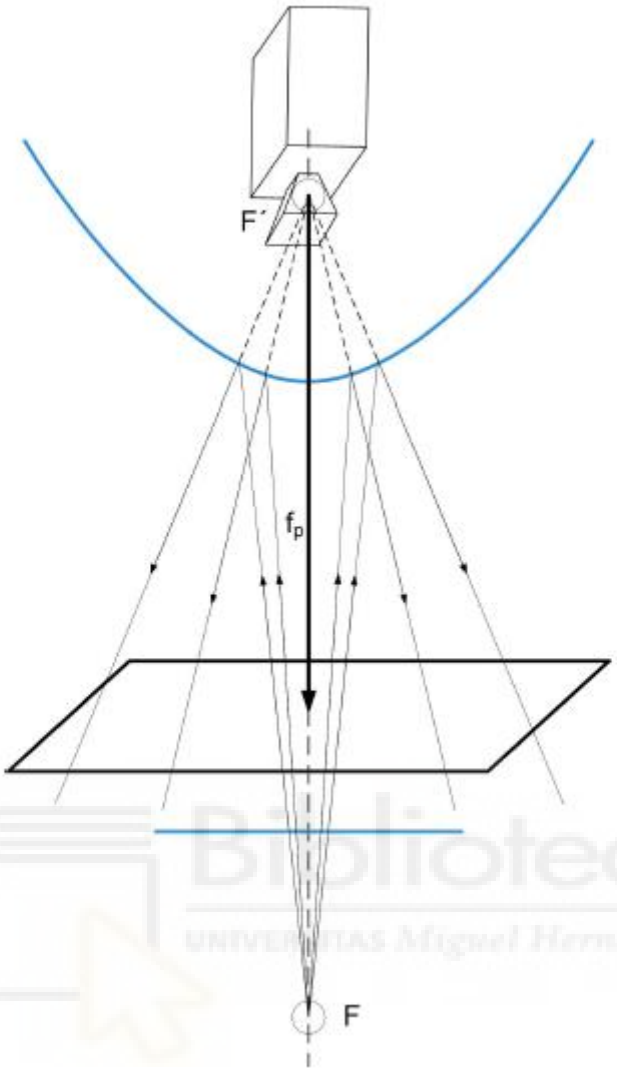


Figura 11: Modelo de proyección: vista ortográfica.

3.2 Sistema de visión utilizado

En este trabajo se pretende obtener la localización de un robot móvil utilizando como sistema de visión la cámara Garmin VIRB 360.

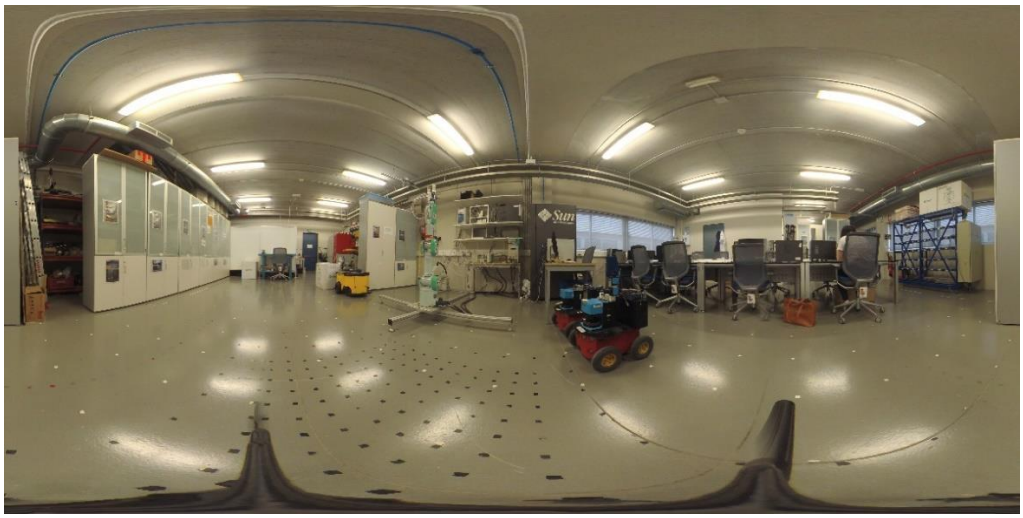
Figura 12: Cámara Garmin VIRB 360.

Se trata de una cámara de 360º compuesta por dos lentes con un campo de visón (FOV) de 201,8 grados[1]. Esta cámara como indica su nombre es capaz de capturar una esfera completa de vídeo, audio o fotos en alta resolución.

Tabla 2: Especificaciones ópticas de la cámara.

SENSOR	1/2.3" Backside-Illuminated CMOS Sensor Count: 2 Pixel Count: 12MP per Sensor Pixel Size: 1.55 um
LENS	8-Elements, Replaceable Strengthened Glass Cover Lens Lens Count: 2 FOV (Unstitched): 201.8^0 per Lens FOV (Traditional 16:9): 141^0 Horizontal Depth of Focus: 40cm to ∞ Effective Focal Length: 1.036mm 35mm Equivalent Focal Length: 6mm Fixed Aperture: f/2.0
ELECTRONIC SHUTTER SPEED	30 ─ 1/8000s

Se puede escoger entre distintos modos de lentes: 360, delantera, trasera y RAW. En la Figura 13 se pueden observar las imágenes que ofrece cada uno de estos modos, en el primero (a) realiza imágenes completamente esféricas en formato panorámico, realizando automáticamente el *stitching*. Con las dos siguientes opciones (b) (c) se obtienen fotos en formato tradicional usando la lente delantera o trasera. Por último, permite escoger el modo RAW (d) que captura imágenes esféricas sin realizar el *stitching* por lo que por cada disparo se obtienen dos imágenes de cada lente. En este trabajo se ha utilizado las imágenes utilizando el modo de lente RAW.

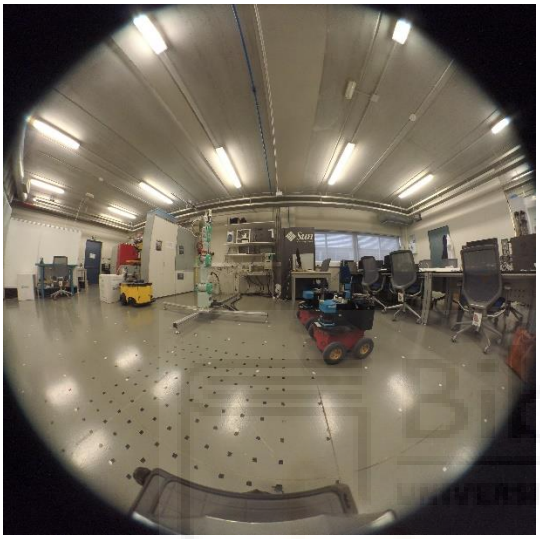


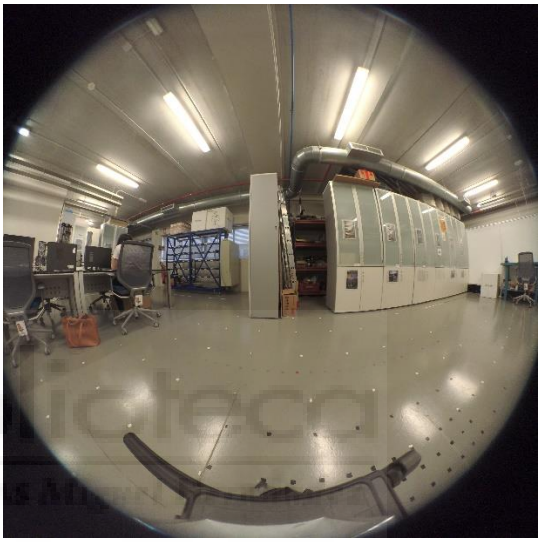
(a)

[1] Las especificaciones de la cámara se pueden ver en: www8.garmin.com/automotive/pdfs/VIRB360-specs.pdf

(b)

(c)





(d)

Figura 13: Modos de ver la imagen: (a) 360, (b) delantera, (c) trasera y (d) RAW.

Capítulo 4
Calibración

La calibración de una cámara es un paso imprescindible cuando se requiera obtener, a partir de una serie de imágenes, medidas de una determinada escena. Por lo que respecta a los métodos para realizarla, pueden clasificarse en tres categorías [58]. Por un lado, empleando las restricciones epipolares y las correspondencias de puntos entre múltiples vistas. Por ejemplo, Micušık y Pajdla [39] utilizan la autocalibración de los sistemas catadióptricos y con lentes de ojo de pez basados en la determinación de la geometría epipolar partiendo de las correspondencias entre dos imágenes tras resolver un problema de autovalores polinomiales. En este modelo se usa RANSAC para suprimir valores atípicos sin ser preciso conocer el escenario con anterioridad. Más tarde, con el ajuste del bloque de paquetes se estiman los parámetros de la cámara y se reconstruye la escena.

Por otro lado, encontramos la reproyección espacial de una imagen de un objeto no plano con coordenadas del mundo 3D conocidas. Schneider et. al. [56] se centran en el estudio de la validez de las proyecciones geométricas para lentes de ojo de pez. Así, la cámara es calibrada empleando una imagen de una estancia preparada con 3D-passpoints que cubre el campo de visión de la lente. Además, se añaden con posterioridad parámetros de distorsión tangencial y radial al modelo de proyección para mejorar la calidad del modelo y tratar de compensar las desviaciones del mundo real del modelo geométrico. Puig et. al. [50] proponen un enfoque de calibración de cámaras catadióptricas utilizando la resección espacial donde las coordenadas elevadas se emplean para proyectar linealmente puntos de la escena al plano de la imagen. Asimismo, tras calcular la matriz de proyección a partir de puntos de la escena distribuidos sobre un objeto de calibración 3D, se aplica un refinamiento final no lineal.

Por último, el tercer método de calibración consiste en la observación de un objeto plano (como por ejemplo el tablero de ajedrez) con coordenadas de un objeto tridimensional conocidas donde son necesarias al menos dos imágenes del objeto plano. Algunos autores que han empleado este método son Scaramuzza et al. [53], Mei y Rives [37] y Zhang [65]. En este capítulo se explicarán de forma detallada los modelos de Scaramuzza y Mei ya que son lo que se han utilizado en este trabajo a la hora de calibrar.

Se puede decir de forma general, que el proceso de calibración se divide en dos pasos. En el primero se debe escoger un modelo que describa correctamente el sistema de imágenes utilizado. Este modelo define la relación entre las coordenadas de los puntos de la escena y las coordenadas de la imagen. En este trabajo se ha utilizado una cámara con lente de ojo de pez ya que debido al gran ángulo de visión presenta grandes ventajas para su utilización en diversas aplicaciones como captura de objetos, el seguimiento de objetos móviles y especialmente la navegación con visión de robot móvil. Sin embargo, las

imágenes que se toman con este tipo de cámaras tienen distorsiones debido a que la superficie real de la imagen no es plana sino esférica. Por esta razón, no se puede utilizar el método convencional para calibrarlas. Sin embargo, en la literatura existe una gran cantidad de métodos propuestos para este tipo de cámaras (Scaramuzza et al. [53] [54], Mei y Rives [37], Micušık [38], , Kannala y Brandt [26], Geyer y Daniilidis [20]. El segundo paso es obtener los parámetros que definen el modelo escogido, tanto los internos como los externos, así como la distorsión presente en la imagen.

Los parámetros extrínsecos definen la orientación y la posición de la cámara respecto a un sistema conocido, denominado sistema de coordenadas del mundo. Por tanto, estos parámetros consisten en una rotación R y una traslación t. Mientras que, los parámetros intrínsecos describen la geometría y la óptica del conjunto cámara y tarjeta de adquisición de imágenes. De otra forma, los parámetros extrínsecos describen la transformación que mapea los puntos expresados en el sistema de coordenadas del mundo al sistema de coordenadas de la cámara. Los parámetros intrínsecos modelan el mapeo de puntos desde el plano de imagen ideal al sensor de imagen.

4.1 Modelo de Mei

El modelo propuesto por Mei [37] combina el modelo de proyección de Geyer [19] y Barreto [7] con una función de distorsión radial. Por tanto, este modelo de proyección permite calibrar, y por tanto obtener los parámetros de los tipos de cámara parabólica, catadióptrica y dióptrica. Esta última se puede utilizar para cámaras con lente de ojo de pez, ya que se refiere a la óptica y no a los espejos como en el caso de las cámaras catadióptricas. En este modelo los puntos en coordenadas del mundo se proyectan en una esfera unidad antes de ser proyectados sobre el plano imagen normalizado.

En primer lugar, un punto de la escena en coordenadas de la cámara X = (X, Y,Z)' se proyecta sobre la esfera unidad, y posteriormente se pasa a un nuevo sistema de referencia cuyo origen es el centro de la esfera Cp=(0,0, ξ).

$$(\chi_s)_{Fm} = \frac{\chi}{\|\chi\|} = (X_s, Y_s, Z_s) \tag{1}$$

$$(\chi_s)_{Fp} = (X_s, Y_s, Z_s + \xi) \tag{2}$$

Después se proyecta sobre el plano imagen normalizado y se suma la distorsión radial y tangencial.

$$m_u = \left(\frac{X_s}{Zs + \xi}, \frac{Y_s}{Z_s + \xi}, 1\right) = \hbar(\chi_s) \tag{3}$$

$$m_d = m_u + D(m_u, V) \tag{4}$$

donde V contiene los parámetros de distorsión, y D la función de distorsión que añade una mayor flexibilidad a la desalineación. Finalmente, se proyecta sobre el plano imagen mediante una matriz de proyección generalizada denominada K (f es la longitud focal generalizada, (xc,yc) el punto principal y s el skew)

$$p = K \cdot m_d = \begin{bmatrix} f & fs & x_c \\ 0 & fr & y_c \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

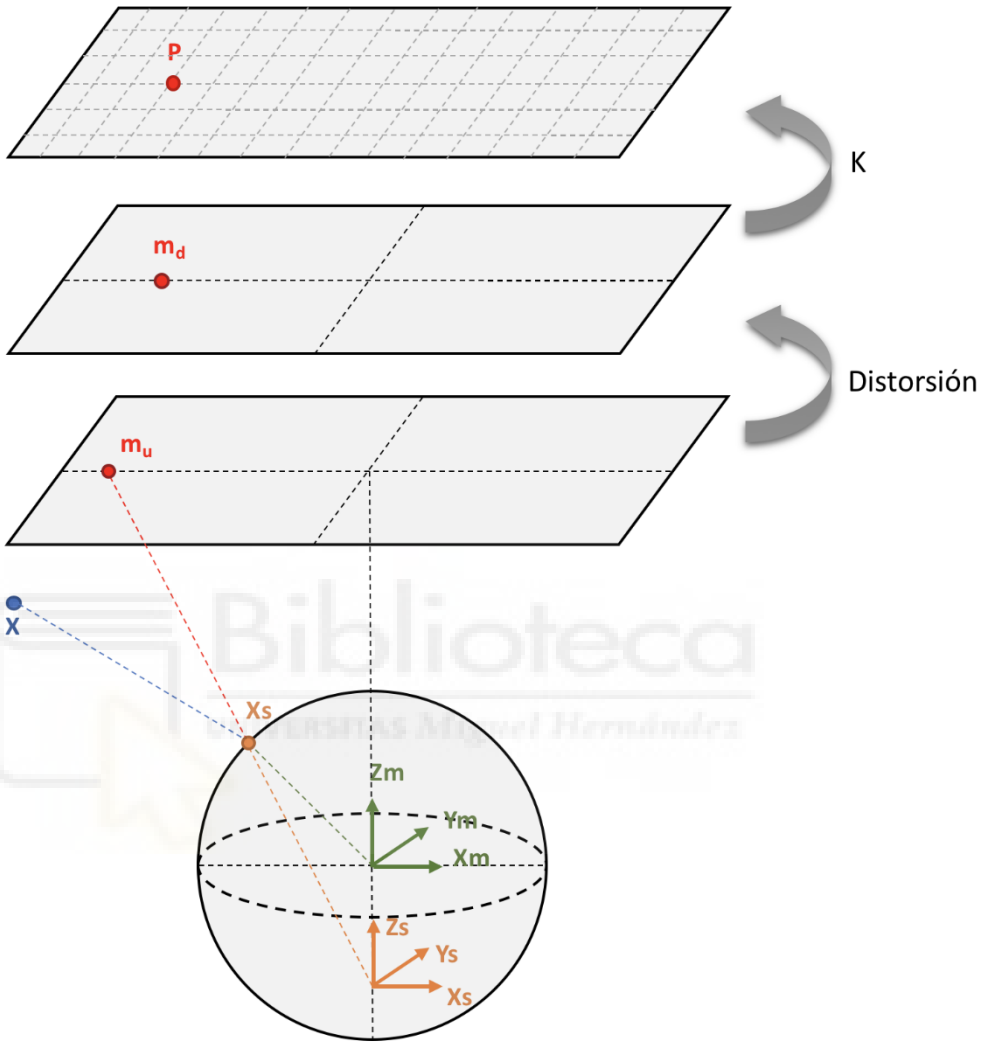


Figura 14: Proyección modelo Mei.

Toolbox en Matlab

Mei desarrolló una Toolbox de calibración que proporciona los parámetros intrínsecos, extrínsecos y de distorsión de la cámara en Matlab. Se puede utilizar para calibrar diversos tipos de cámaras como parabólica, catadióptrica y dióptrica Antes de iniciar la Toolbox se requiere tomar varias imágenes a un patrón de calibración desde distintas posiciones y ángulos.

Los parámetros del modelo son estimados mediante una minimización no lineal del error de reproyección con el algoritmo de Levenberg-Marquardt. Con el fin de que se obtenga unos resultados satisfactorios se debe obtener antes una estimación inicial de los parámetros. Para ello, asume que los errores del modelo teórico son pequeños e inicializa los parámetros de distorsión y *skew* a cero. Para la estimación inicial de la longitud focal

el usuario debe seleccionar al menos tres puntos alineados no radiales. Posteriormente, dado que la Toolbox dispone de un algoritmo de extracción de esquina semi automático, únicamente necesita que se seleccione cuatro esquinas del patrón, en sentido de las agujas del reloj, en cada imagen, con las que se estimará los parámetros de calibración.

4.2 Modelo de Scaramuzza

Scaramuzza propuso un nuevo modelo que se puede utilizar para cualquier cámara omnidireccional central, incluso las lentes de ojo de pez. Este modelo trata el sistema de forma compacta.

Dado un punto p en una imagen, cuyas coordenadas en pixeles con respecto al centro de la imagen omnidireccional son (u, v), está relacionado con P, su correspondiente vector 3D, por la siguiente ecuación.

$$P = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} u \\ v \\ f(u,v) \end{bmatrix} \tag{6}$$

La función $f(u,v)$ únicamente depende de la distancia de un punto al centro de la imagen, debido a que el espejo es rotacionalmente simétrico, por tanto, se podría simplificar a $f(\rho)$ siendo $\rho = \sqrt{x^2 + y^2}$. Esta función se define como un polinomio de Taylor cuyo término de primer orden ha sido eliminado, esto se debe a que, para los espejos hiperbólicos y parabólicos, así como para las cámaras con lente de ojo de pez, la primera derivada en el punto $\rho = 0$ del polinomio es igual a cero.

$$f(\rho) = a_0 + a_2\rho^2 + \cdots + a_N\rho^N \tag{7}$$

Scaramuzza también modela los errores debidos al proceso de digitalización y debidos a una posible desalineación entre los ejes de la cámara y del espejo, a través de una transformación afín.

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} c & d \\ e & 1 \end{bmatrix} \cdot \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} C_x \\ C_y \end{bmatrix} \tag{8}$$

donde (u′, v′) corresponden a las coordenadas reales distorsionadas y (u, v) a las coordenadas distorsionadas ideales.

Toolbox en Matlab

Scaramuzza también desarrolló una Toolbox para Matlab donde implementó su modelo de cámara, para la cual no se requiere tener conocimiento previo sobre esta. Se puede utilizar para calibrar cualquier cámara omnidireccional central incluyendo las cámaras con lente de ojo de pez. Al igual que la Toolbox de Mei, se requiere de una serie de imágenes, desde diferentes posiciones y orientaciones, de un patrón de calibración para obtener los parámetros intrínsecos y de distorsión.

Los parámetros se estiman mediante la resolución de un problema lineal de mínimos cuadrados compuesto por cuatro pasos, sin tener en cuenta las transformaciones

afines. Seguidamente se realiza un refinamiento no lineal con mínimos cuadrados formado por dos etapas, obteniéndose una convergencia más rápida sin afectar a los resultados finales. En la primera, se estiman los parámetros extrínsecos ignorando lo intrínsecos. Estos últimos se optimizan en la segunda etapa utilizando las posiciones del patrón de calibración obtenidas en la etapa anterior.

El proceso de calibración puede realizarse de forma manual o con detección automática, a diferencia de la Toolbox de Mei. Las posiciones de las esquinas se pueden cambiar manualmente si han sido calculadas incorrectamente por la detección automática de esquinas.

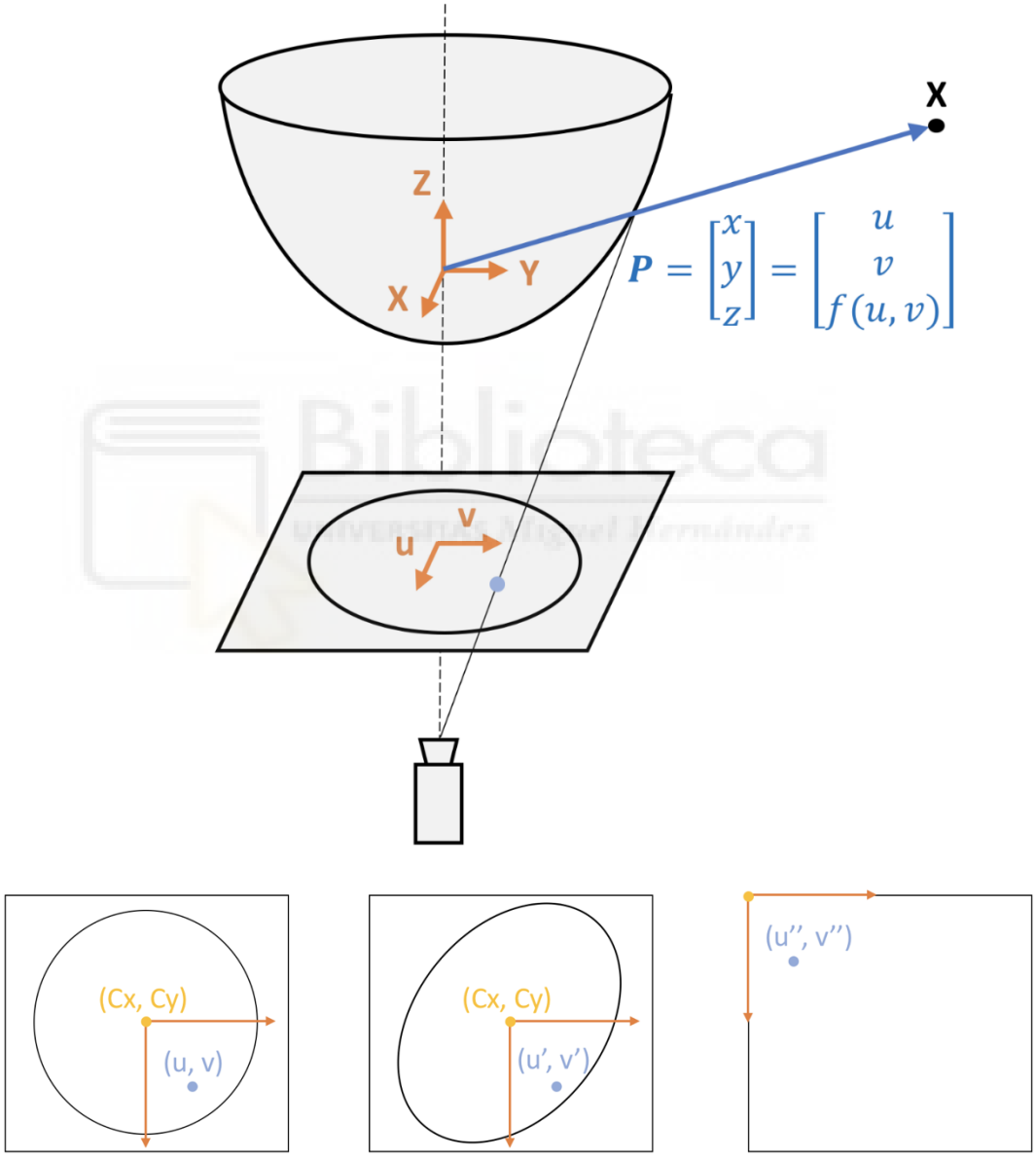


Figura 15: Modelo proyección Scaramuzza.

Capítulo 5
Localización visual

En este capítulo se va a hablar de forma detallada de la odometría visual definiendo el concepto y explicando los pasos que la componen. Pero antes de comenzar es importante describir algunos conceptos de la geometría epipolar ya que la odometría visual se basa en ella. Por tanto, este capítulo se divide en dos apartados: geometría epipolar y odometría visual.

5.1 Geometría epipolar

La geometría epipolar es una geometría proyectiva intrínseca entre dos imágenes, que solo depende de la calibración de la cámara y de su pose relativa, no de la estructura de la escena.

Dado un punto de la escena M, se proyecta sobre el plano de la primera imagen x y sobre el de la segunda x'. Los centros de cada imagen están unidos por una recta llamada línea de base. El plano que forma el punto M y los dos centros de proyección de ambas cámaras se denomina plano epipolar. Existe un haz de planos epipolares cuyo eje es la línea de base. Los epipolos son los puntos de intersección de la línea base con el plano imagen asociado a esa vista. Dicho de otra forma, el epipolo e_1 es la proyección en la primera cámara del centro de proyección de la otra, y al contrario.

La restricción epipolar establece que para la proyección del punto M sobre el plano de la primera imagen x se tiene que su proyección sobre el plano de la segunda imagen x' deberá estar sobre la línea epipolar l'x, y viceversa. Siendo la línea epipolar la recta resultante de la intersección del plano epipolar con el plano imagen. Por tanto, a la hora de buscar correspondencias de puntos entre imágenes esta condición restringe la búsqueda a la línea epipolar. Esta restricción puede expresarse de forma algebraica como:

$$x'^T \cdot F \cdot x = 0 \tag{9}$$

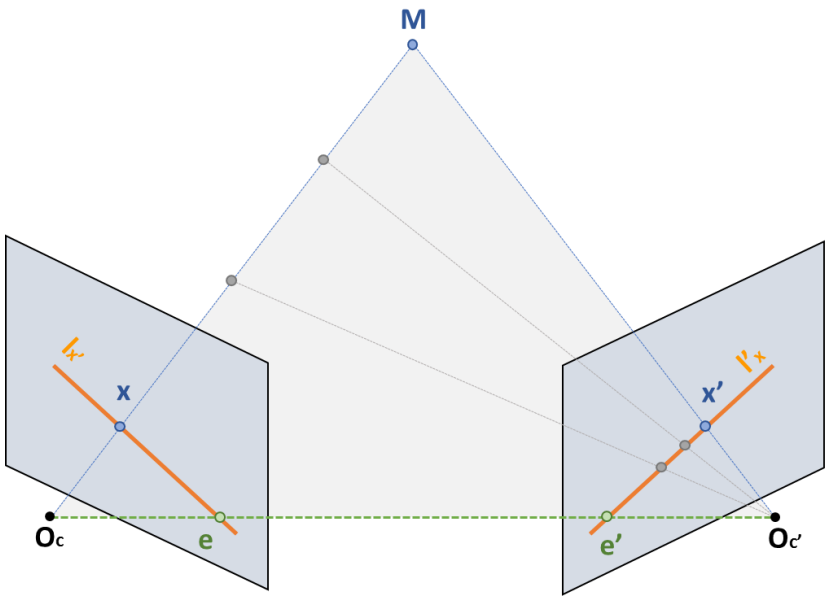


Figura 16: Geometría epipolar.

Donde la matriz F se conoce como matriz fundamental y recoge todas las restricciones asociadas a esta geometría. Esta matriz relaciona puntos expresados en pixeles por lo que no requiere que las cámaras se encuentren calibradas.

Por otro lado, se tiene la matriz esencial, que se trata de una especialización de la matriz anterior para el caso de que las coordenadas se encuentren normalizadas, es decir, que las cámaras se encuentren calibradas y por tanto se conozca la matriz de parámetros intrínsecos. Esta matriz dispone de menos grados de liberta en comparación con la matriz fundamental.

Teniendo en cuenta que la matriz de proyección de la cámara viene definida por $P = K[R|t]$ y que $x = PX$ es un punto en la imagen. Si como se ha comentado, la matriz de calibración K es conocida, entonces se puede expresar el punto de la imagen en coordenadas normalizadas $\hat{x} = K^{-1} \cdot x$, obteniendo por tanto que $x = [R|t]X$.

Si se considera que la ecuación anterior corresponde a la imagen del puno X con respecto a una cámara cuya matriz de calibración es una matriz identidad. Entonces, $K^{-1} \cdot P = [R|t]$ se denomina matriz de una cámara normalizada, ya que se ha eliminado el efecto de la matriz de calibración conocida.

La matriz fundamental correspondiente a un par de cámaras normalizadas, cuyas matrices son $P = [I|0]$ y $P' = [R|t]$, se denomina matriz esencial. La ecuación de condición de epipolaridad para la matriz esencial se muestra a continuación.

$$\hat{x}'^{T} \cdot E \cdot \hat{x} = 0 \tag{10}$$

Finalmente, si en la ecuación anterior se sustituye $\hat{x} = K^{-1} \cdot x$ se obtiene que $x'^{T} K'^{-1} E K^{-1} x = 0$. Al compararla con la ecuación de epipolaridad para la matriz fundamental se observa que la relación entre esta y la matriz esencial es que $E = K'^{T} F K$ [60]. En el apartado 5.2.3 se explica cómo descomponer la matriz esencial.

5.2 Odometría Visual

La odometría visual tiene como objetivo estimar la posición relativa del robot utilizando la información extraída de una secuencia de imágenes. Con el fin de que el proceso sea eficaz es importante que se cumplan una serie de requisitos. Por ejemplo, debe haber suficiente iluminación en el ambiente, así como la escena debe ser mayormente estática y con suficiente textura que permita extraer el movimiento aparente. También es importante que exista solapamiento entre las imágenes [52].

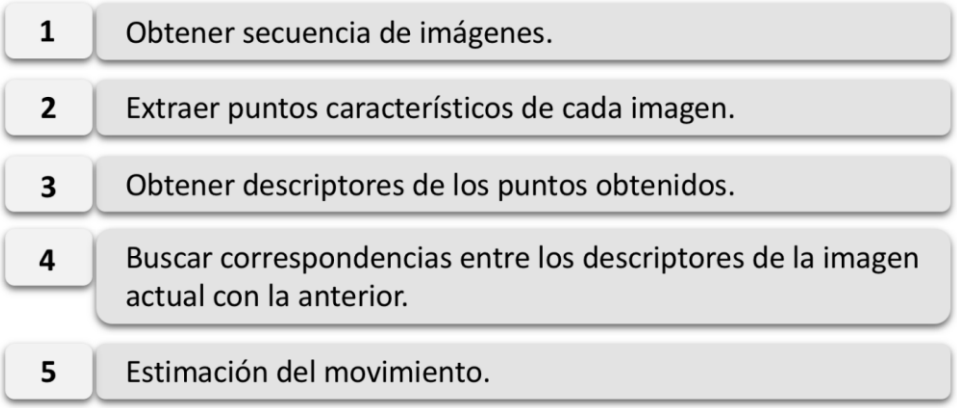


Figura 17: Algoritmo de la odometría visual.

En el caso de la primera imagen de la secuencia, se considera que la cámara se encuentra en el origen, por tanto su rotación es una matriz identidad y su vector translación nulo, así pues únicamente se extraerán las características. Después para cada nueva imagen que se tome, se deben (1) detectar características y obtener des (2) buscar correspondencias con las imágenes anteriores. Existen dos formas para realizar estos dos primeros pasos. Se pueden encontrar características en una imagen y, mediante técnicas de búsqueda locales, intentar localizarlas en las imágenes siguientes, o detectar características en todas las imágenes y buscar correspondencias basándose en alguna similaridad métrica que presenten los descriptores. El siguiente paso es (3) calcular el movimiento relativo entre el par de imágenes, obteniendo la orientación y translación. La pose de la cámara se obtiene teniendo en cuenta las de las cámaras anteriores. A la hora de calcular el movimiento de la cámara se asume que cualquier punto de la escena presenta la misma intensidad en todas las imágenes en las que aparezca.

5.2.1 Extracción y descripción de puntos característicos

En esta etapa se pretende encontrar en una imagen aquellos puntos que cumplan unas determinadas características, como por ejemplo esquinas, líneas, regiones o bordes, que permitirán identificar la imagen. En la literatura se pueden encontrar múltiples métodos para la extracción de características, en este apartado se han definido algunos de ellos de forma detallada, concretamente los que se utilizarán en este trabajo. Los detectores de puntos característicos tienen que ser capaces de localizar estos puntos en otras imágenes, aunque exista entre ellas variaciones de puntos de vista y/o ángulo, e incluso iluminación.

El siguiente paso, es obtener los descriptores para cada uno de los puntos detectados. El descriptor se caracteriza por proporcionar información relevante y característica

de la región que rodea al punto de interés, permitiendo que pueda ser identificado en otras imágenes. Esa fase del algoritmo, como ya se ha comentado, se puede hacer realizar utilizando uno de los dos métodos expuestos.

En primer lugar, para el método basado en apariencia, los descriptores utilizados son llamados globales ya que proporcionan información sobre toda la imagen, por tanto, se obtiene un descriptor para cada imagen. Este método presenta buenos resultados en entornos no estructurados y dinámicos donde es difícil extraer puntos característicos estables. Los algoritmos que se emplean son conceptualmente más simples, por lo que resulta una alternativa sistemática e intuitiva a la hora de resolver los problemas de creación de mapas y de localización, siendo generalmente mapas topológicos que no incluyen información métrica. En segundo lugar, el método basado en características, como su nombre indica, consiste en extraer un número limitado de características locales que sean relevantes y obtener sus descriptores con la propiedad de ser invariantes [48].

La mayoría de las implementaciones de odometría visual utilizan el método basado en características ya que es más preciso y rápido que el método basado en apariencia. Sin embargo, se necesita que realicen las correspondencias entre los puntos de cada par de imágenes de forma robusta.

Por ejemplo, Valiente et al. [59] en su trabajo resuelven el problema de odometría visual utilizando los dos métodos que se acaban de exponer, mediante el uso de un sistema de visión catadióptrico.

5.2.2.1 SURF

El detector local de características *Speeded Up Robust Features* (SURF) fue propuesto por Bay et al.[8], se caracteriza por ser invariante ante rotación y escala. Se basa en calcular el determinante de la matriz Hessiana para diferentes escales de imagen. Este detector de características está inspirado en el detector SIFT, pero con mayor eficacia y más rápido.

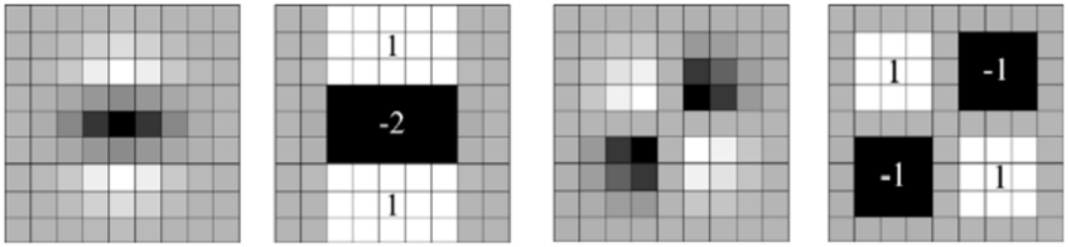
Por lo que se refiere a la detección de puntos de interés, en este algoritmo se emplea una aproximación básica de la matriz Hessiana, Debido a su gran precisión es usado para hallar puntos donde la determinante es máxima. Dado un punto **x** = (x, y) de una imagen I, la matriz Hessiana está determinada por:

$$H\,(x,\sigma) = \begin{bmatrix} L_{xx}\,(x,\sigma) & L_{xy}\,(x,\sigma) \\ L_{xy}\,(x,\sigma) & L_{yy}\,(x,\sigma) \end{bmatrix} \tag{11}$$

$L_{xx}\,(x,\sigma)$ *e*s la convolución de segundo orden de la Gaussiana con la imagen I en el punto **x**. Se maneja de manera similar para $L_{xy}\,(x,\sigma)$ y $L_{yy}\,(x,\sigma)$ debido al éxito obtenido por Lowe [31] se usará una aproximación de la matriz Hessiana con los filtros de caja (L_{xx}, L_{xy}, L_{yy}) con un valor σ = 1.2 los cuales también serán discretizados y serán representados por D_{xx}, D_{xy}, D_{yy} (derivadas parciales) evaluándolos con un costo computacional muy bajo gracias al uso de imágenes integrales.

La determinante de las aproximaciones de las derivadas parciales D_{xx}, D_{xy}, D_{yy} se calcula en la ecuación que se expondrá a continuación, donde el valor de 0.9 se debe a la aproximación del filtro Gaussiano.

$$\det\left(H_{aprox}\right) = D_{xx}\, D_{yy} - (0.9D_{xy})^2 \tag{12}$$



(a) (b)

Figura 18: (a) Derivada parcial de segundo orden L_{yy} y su aproximación D_{yy} y (b) derivada parcial de segundo orden L_{xy} y su aproximación D_{xy}.

Por otro lado, hay que tener en cuenta que en el detector SURF la representación espacio-escala es similar a SIFT, es decir, que está dividido en octavas. Sin embargo, hay un cambio de paradigma ya que en SIFT se realiza un re-muestreo de la imagen original y en el caso SURF el filtro crecerá a medida que la escala se incrementa, como se muestra a continuación.

En el algoritmo de detección de SURF las octavas están compuestas por un número fijo de imágenes que son resultado de la convolución de la imagen original con una serie de filtros que incrementan su tamaño. La imagen de salida se obtiene a través de la convulsión de la imagen original con un filtro de dimensiones 9x9 (que corresponde a la derivada parcial de segundo orden de una gaussiana con σ = 1.2). La misma es considerada como la escala inicial y las capas siguientes serán incrementos graduales de los filtros. Por último, para calcular la localización de todos los puntos de interés en todas las escalas, se procede a la eliminación de los puntos que no cumplan la condición de máximo en un vecindario de 3x3x3. De esta manera, el máximo determinante de la matriz Hessiana es interpolado en la escala y posición de la imagen.

Por otro lado, vamos a analizar brevemente la extracción de descriptores. En este sentido, hay que obtener la orientación de cada punto clave para que el descriptor sea invariante ante la rotación. Por ello, hay que hallar la respuesta wavelet Haar en las direcciones x e y en un radio de 6σ para cada punto de la imagen I. El tamaño del wavelet es de 4σ, los filtros usados se muestran en figura 3.9, donde se pueden emplear de nuevo las imágenes integrales para filtrar rápidamente.

Tras calcular las respuestas wavelet con un valor de gaussiano $\sigma = 2s$ (donde s es la escala en que el punto fue detectado) centrado en el punto de interés, las mismas son representadas como puntos en el espacio (la respuesta horizontal a lo largo del eje de abscisas y el vertical a lo largo de la ordenada). Más tarde se obtiene la orientación dominante mediante la suma de todas las respuestas de cada sector dentro de una ventana móvil de valor $\pi/3$. El mayor vector obtenido en todas las ventanas será la orientación del punto de interés.

Para la extracción del descriptor, primero hay que construir una región cuadrada de tamaño $20s$ que estará centrada en el punto de interés y orientada con el valor obtenido en el paso anterior. Dentro de esta ventana se formará 4x4 subregiones, para cada subregión se calculará la respuesta wavelet Haar de 25 puntos de muestra regularmente distribuidos, sumando las respuestas paralelas a los ejes x e y se obtiene lo siguiente:

$$v_{subregión} = \left[\sum dx \sum dy \sum dx \sum dy\right] \quad (13)$$

De esta forma por cada una de las 16 subregiones tenemos 4 valores, tenemos un vector descriptor de longitud 64.

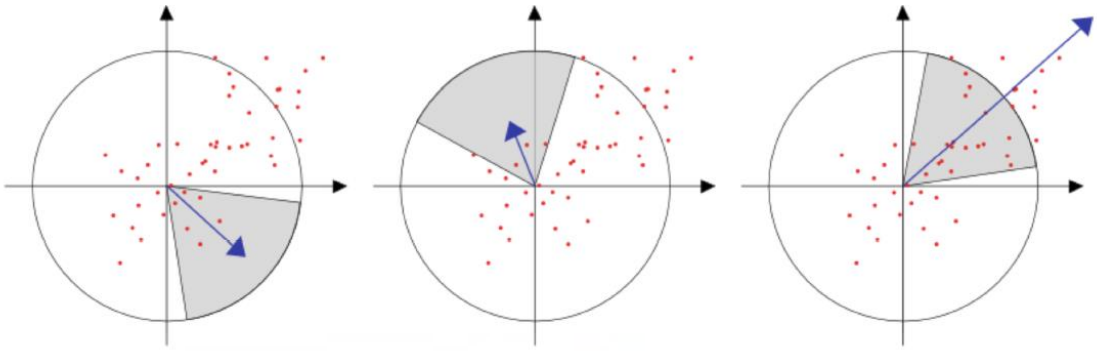


Figura 19: Cálculo de la orientación mediante una ventana de 60º de tamaño.

5.2.2.2 BRISK

En primer lugar, hay que destacar que BRISK (*Binary Robust Scalable Keypoints*) fue propuesto por S. Leutenegger et al.[29]. Se trata de un método que permite la detección de puntos característicos del tipo esquina. Una característica que presentan es que son invariantes a escala, para ello la detección de puntos se realiza a múltiples escalas. Además, hay que resaltar que este modelo posee gran modularidad, pudiendo emplearse el detector BRISK en combinación con cualquier descriptor de puntos clave y viceversa. En cuanto a la detección espacio-escala de puntos de interés, debemos tener en cuenta que está inspirado en el algoritmo AGAST (extensión del algoritmo FAST), eficiente para extracción de características. Con el fin de obtener invariancia a cambios de escala, se llevará a cabo una búsqueda de máximos no solo en la imagen sino también en espacio-escala usando el valor s como medida de prominencia, además de estimar cada punto de control en un espacio-escala continuo.

Primero, se crea una pirámide de capas espacio-escala que consiste de n octavas c_i y n intra-octavas d_i con n = 4 como valor típico, las octavas son formadas muestreando entre 2 la imagen original c_o. Cada intra-octavas d_i estará ubicada entre las capas c_i y c_{i+1} de tal forma que se cumple que $t\,(c_i) = 2^i\ t\ t\,(d_i) = 2^i * 1.5$.

Igualmente, hay que tener en cuenta que en BRISK se hace uso de una máscara 9-16 (detector FAST 9-16), lo cual significa que al menos 9 pixeles consecutivos de los formados por un radio de 16 pixeles son lo suficientemente más brillantes o más oscuros que el píxel central, tal que esa distancia debe superar un umbral T y así identificar potenciales zonas de interés. Más tarde, se realiza una eliminación de los puntos no

máximos. Para tal cometido se debe cumplir que el punto a evaluar debe tener el máximo valor s (s es el máximo umbral considerando el punto como una esquina) respecto a sus 8 vecinos, luego dicho valor también debe ser máximo respecto a la capa superior e inferior.

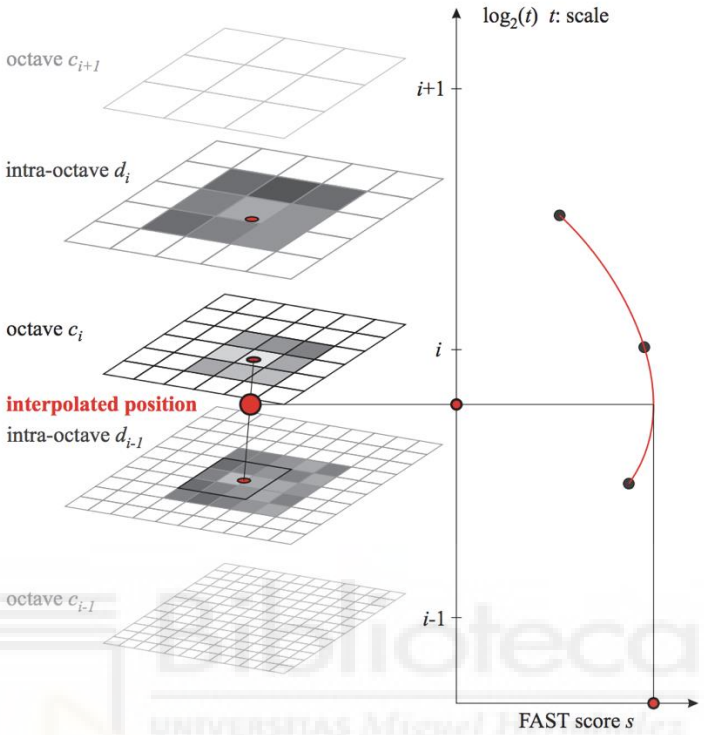


Figura 20: Detección del punto de interés en el espacio de escala.

Tras lo anterior, se obtiene luego de 3 en 3 el valor de prominencia s (en la capa del punto de interés, la capa superior e inferior). Con estos puntos obtenidos se ajusta una función 2D cuadrática, la proyección de la función obtenida la ajustamos a una parábola, donde se estime el máximo "score" s de la función, realizaremos una interpolación para obtener la escala en la cual se obtiene el valor máximo s. Este valor (que denominaremos t) será la escala verdadera del máximo, esto es, obtenemos el valor en una escala continua. Este valor t será usado para calcular el patrón de muestreo en la sección de descripción del punto de interés.

En otro orden de cosas, en la descripción de puntos de interés el descriptor BRISK hace uso de una cadena binaria de valores. De esta manera podremos hacer uso de un menor tiempo computacional, además de requerir de menos recursos para la evaluación del mismo. Para ello utilizaremos un patrón de muestreo que nos permitirá con simples comparaciones de brillo e identificando la dirección de cada punto clave obtener invariancia ante la rotación y en general robustez.

Asimismo, está inspirado en el descriptor DAISY y se aplica un kernel gaussiano con una desviación estándar σ_i (proporcional a la distancia entre puntos en el círculo) a cada punto $\mathbf{p}_i$, dado un número N de puntos en el patrón de muestreo tendremos N. $(N -$

$1)/2$ pares de puntos. Con los valores de intensidad de dos puntos $I\ (\mathbf{p}_i, \sigma_i)$ e $I\ (\mathbf{p}_j, \sigma_j)$ estimaremos el gradiente local de la siguiente manera:

$$\mathbf{g}\ (\mathbf{p}_i, \mathbf{p}_j) = \ (\mathbf{p}_i - \ \mathbf{p}_j).\frac{I(\mathbf{p}_j, \sigma_j) - (\mathbf{p}_i, \sigma_i)}{\|(\mathbf{p}_i - \ \mathbf{p}_j)\|^2} \tag{14}$$

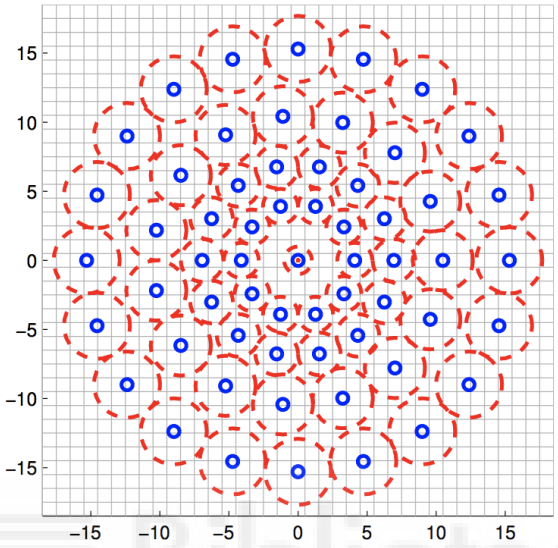


Figura 21: Patrón de muestreo. Los círculos azules son las ubicaciones de los muestreos y los rojos la desviación estándar del Kernel Gaussiano utilizado para suavizar la intensidad.

A continuación, hay que definir un par de subconjuntos, pares distancias cortas *S* y pares distancias largas *L*, donde definiremos estos subconjuntos mediante umbrales. Los puntos pertenecen a distancias cortas si las distancias entre los puntos no superan el umbral $\delta_{max} = 9.75t$ y las distancias largas si son mayores al umbral $\delta_{min} = 13.67t$ el valor de la gradiente (15) se usará para obtener un ángulo con el cual se rotará al patrón de muestreo.

$$\mathbf{g} = \begin{pmatrix} g_x \\ g_y \end{pmatrix} = \frac{1}{L}. \sum_{(\mathbf{p}_i, \mathbf{p}_j) \in L} \mathbf{g}(\mathbf{p}_i, \mathbf{p}_j). \tag{15}$$

Con el valor hallado anteriormente obtenemos el ángulo $\propto = \arctan 2\ (g_y, g_x)$. Con este último rotaremos el patrón de muestreo alrededor del punto de interés. Con el subconjunto de distancias cortas y mediante una comparación de las intensidades de los puntos en el patrón se forma el vector descriptor b.

$$b = \begin{cases} 1, & I\ (\mathbf{p}_j^{\propto}, \sigma_j) > I\ (\mathbf{p}_i^{\propto}, \sigma_j) \quad \forall (\mathbf{p}_i^{\propto}, \mathbf{p}_j^{\propto}) \in S \\ 0, & \text{en otros casos} \end{cases} \tag{16}$$

Finalmente, después de pasar por el patrón de muestreo, aplicar los umbrales y realizar las comparaciones de intensidad entre los puntos del patrón se obtiene una cadena de 512 bits. Al realizar una aplicación como la comparación entre dos imágenes estos

vectores formados serán fácilmente evaluados, ya que requerirá de una simple operación XOR bit a bit. De esta forma bajaremos los tiempos computacionales además de reducir espacio en memoria.

5.2.2.3 KAZE

El detector de puntos de interés KAZE fue propuesto Alcantarilla et al. [1]se basa en el determinante normalizado de estaca de la matriz Hessiana, calculándose en niveles de escala múltiples.

Por lo que respecta a la detección de puntos de interés, se calcula la respuesta del determinante escala-normalizado de Hessian en los niveles múltiples de la escala. Para la detección de características multiescala, el conjunto de los operadores diferenciales debe normalizarse con respecto a la escala, ya que en general la amplitud de los derivados espaciales disminuye con la misma. En este sentido:

$$L_{Hessian} = \sigma^2\left(L_{xx}L_{yy} - L_{xy}^2\right) \tag{17}$$

Donde (L_{xx}, L_{yy}) son los derivados horizontales y verticales de segundo orden respectivamente, y L_{xy} es el segundo orden derivado cruzado. Dado el conjunto de imágenes filtradas de la escala no lineal de espacio L^i, se analiza la respuesta del detector a diferentes niveles de escala σ_i. Se busca la máxima en escala y ubicación espacial. La búsqueda de la misma se realiza en todas las imágenes filtradas excepto i = 0 e i = N. Cada una se busca sobre una ventana rectangular del tamaño $\sigma_i \times \sigma_i$ en la corriente i, superior i + 1 y más bajo i − 1 en filtrado de imágenes.

Para acelerar la búsqueda de máxima, en primer lugar, hay que comprobar las respuestas en una ventana de tamaño 3 × 3 píxeles, con el fin de descartar rápidamente las respuestas no máximas. Finalmente, la posición del *keypoin*t se estima con exactitud del píxel secundario usando el método propuesto por Brown et al. [10].

El sistema de los derivados de la primera y segunda orden se aproxima por medio de 3 × 3 filtros de Scharr de diversos tamaños derivados del paso σ_i. Los derivados de segundo orden son aproximados mediante el uso de filtros Scharr consecutivos en las coordenadas deseadas de los derivados. Estos filtros aproximan la invariación de rotación significativamente mejor que otros filtros populares tales como filtros de Sobel o diferenciación central estándar de las diferencias [61]. A pesar de que sería necesario calcular los derivados multiescala para cada píxel, se reutiliza el mismo conjunto de derivados que se calculan en el paso de detección.

Por otro lado, para obtener descriptores de rotación invariantes, se necesario estimar la orientación dominante en un barrio local centrado en la ubicación del *keypoint*. De forma similar a SURF, la orientación dominante se encuentra en un área circular de radio $6\sigma_i$ con un paso de muestreo de tamaño σ_i. Para cada una de las muestras en la circular área, los derivados de primera orden L_x y L_y se ponderan con Gauss centrado en el punto de interés. Entonces, las respuestas derivadas se representan como puntos en el espacio vectorial y la orientación dominante se encuentra al sumar las respuestas dentro de un segmento de círculo corredizo que cubre un ángulo de $\pi/3$. Desde el vector más largo, se obtiene la orientación dominante.

Para construir el descriptor, hay que usar el descriptor M-surf adaptado a nuestra escala no lineal en el marco espacial. Para una función detectada a escala σ_i los derivados L_x de primer orden y L_y del tamaño σ_i se computan sobre una rejilla rectangular de $24\sigma_i \times 24\sigma_i$. Esta cuadrícula está dividida en 4 × 4 subregiones del tamaño $9\sigma_i \times 9\sigma_i$ con un traslapo de $2\sigma_i$. Las respuestas derivadas en cada subregión se ponderan con gauss (σ_1 = 2,5 σ_i) centrado en el centro de la subregión y resumido en un vector de descriptor $d_v = (\sum L_x, \sum L_y, \sum|L_x|, \sum|L_y|)$. Tras ello, cada vector de la subregión se pondera usando un gauss (σ_2 = 1,5 σ_i) definido sobre una máscara de 4 × 4 y centrada en el interés del *keypoint*. Al considerar la orientación dominante del *keypoint*, cada una de las muestras en la cuadrícula rectangular se gira de acuerdo a la orientación dominante. Además, los derivados también se calculan de acuerdo con la misma. Finalmente, el vector descriptor de la longitud 64 se normaliza en un vector unitario para lograr la invariación del contraste.

5.2.2.4 HARRIS

Se trata de uno de los detectores de puntos de interés que más se utiliza y fue propuesto por Harris y Stephens en 1988 [23], que se caracteriza por ser invariante a cambios de escala, rotación y variaciones de iluminación, así como al ruido presente en la imagen [6].

Las esquinas son puntos característicos muy poco susceptibles a cambios de rotación y escala. Una esquina o *corner* se caracteriza por ser una región de la imagen con cambios de intensidad en diferentes direcciones. Éste será el principio básico de búsqueda de puntos de Harris. Filtrando la imagen con una ventana móvil en diferentes direcciones, se obtienen tres tipos de región.

- *Flat* o región plana: No hay cambios en ninguna dirección.
- *Edge* o borde: No hay cambios en la dirección del propio *edge.*
- *Corner* o esquina: Hay cambios significativos en todas direcciones.

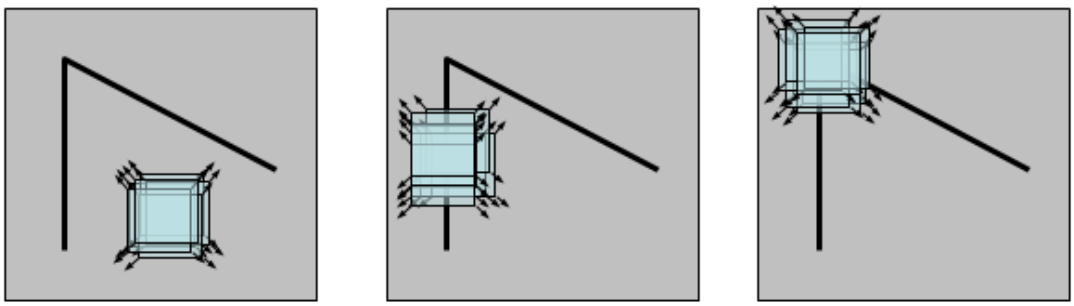


Figura 22: Tipos de regiones que pueden detectarse (de izquierda a derecha): región plana, borde y esquina.

Este método se basa en buscar puntos característicos que sean esquinas, para detectarlos matemáticamente utiliza una matriz cuyos valores propios definen el gradiente de intensidad con respecto a un eje de coordenadas de la imagen. En función de los valores que tengan λ_1 y λ_2 puede ser que se trata de un borde si uno de ellos es elevado y el otro no, zonas de intensidad constante si ambos valores son pequeños, o una esquina si ambos

valores son elevados. Según lo comentado, el método de detector de esquina Harris buscará aquellos puntos que cumplan con la última condición.

- *Flat*. Si ambos valores son pequeños, indica que la función de autocorrelación es plana, por tanto, la zona de la imagen tiene una intensidad aproximadamente constante
- *Edge*. Si uno de los valores es pequeño y otro es elevado, la función de autocorrelación tendrá un cierto rizado
- *Corner*. Si los dos valores son elevados, en la función se observarán picos bruscos.

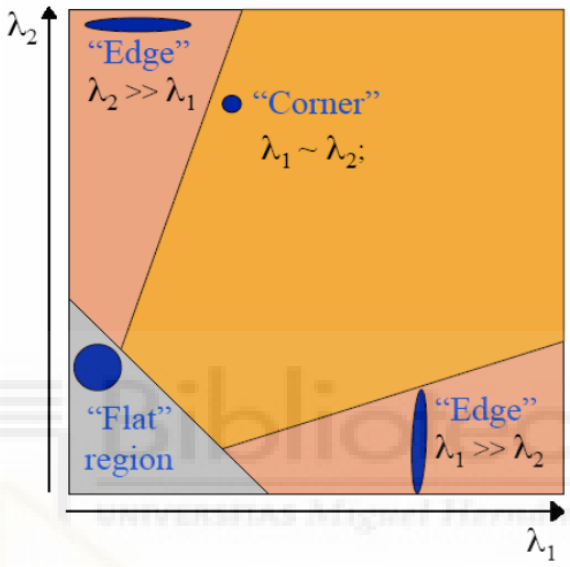


Figura 23: Clasificación de las regiones en función de los valores propios de la matriz de autocorrelación.

Se puede justificar intuitivamente el motivo de esta clasificación a partir de los valores propios. λ1 y λ2 reflejan los modos de variación de las direcciones principales de los gradientes. Por ese motivo, cuando en una región de la imagen los dos valores son elevados, deducimos que localmente existen dos direcciones importantes de los gradientes, y se concluye en que es una esquina.

5.2.2 Matching

Tras haber obtenido los puntos característicos de las imágenes y sus descriptores, se debe buscar correspondencias entre estos puntos. Esto quiere decir, que se pretende ver si algunos de los puntos detectados en una imagen coinciden con los de otra imagen.

En este proceso se pueden producir emparejamientos erróneos, que pueden deberse a distintos factores como la iluminación, el ruido presente en la imagen o incluso por perspectiva. Los resultados también varían en función del tipo de descriptor utilizado, cuanto más complejos sean mayor fiabilidad de correspondencias se obtendrá. Debido a este problema propuesto, es aconsejable utilizar algún método que elimine las falsas correspondencias, por ejemplo, RANSAC o mínimos cuadrados.

5.2.2.5 RANSAC (*RANdom SAmple Consensus*)

El algoritmo RANSAC fue propuesto por Fischler y Bolles [17] se trata de un método iterativo que se utiliza para la obtención de los parámetros de un modelo matemático de un conjunto de datos en presencia de valores atípicos. Este algoritmo se caracteriza por utilizar de forma aleatoria un conjunto de datos lo más reducido posible para estimar el modelo y después verificar este con el resto de los puntos para finalmente seleccionar como solución aquel modelo que presente un mayor número de inliers. A continuación, se muestra en la tabla de forma resumida los pasos de este algoritmo.

Tabla 3: Pasos del algoritmo RANSAC.

Algoritmo RANSAC
1: Seleccionar de forma aleatoria un subconjunto n con el mínimo número puntos necesarios para determinar el modelo.
2: Obtener un modelo a partir del subconjunto.
3: Determinar cuántos puntos satisfacen el modelo, es decir, la distancia es menor que t (inliers).
4: Repetir los pasos 2) y 3) un número determinado de iteraciones.
5: Seleccionar el modelo que presente un mayor número de inliers.

En el algoritmo existen tres parámetros que se debe determinar su valor. En primer lugar, la distancia t que determina si un punto es compatible con el modelo (*inlier*) o no (*outlier*). Por otro lado, el número de iteraciones N necesarias para garantizar que encuentra una solución correcta. Se calcula en función de la probabilidad de éxito p de que el algoritmo seleccione al menos un subconjunto que no incluya valores atípicos. Siendo u la probabilidad de que un punto del conjunto sea *inlier*, la probabilidad de que haya un *outlier* es e=1-u.

$$1 - p = (1 - u^m)^N \tag{18}$$

$$N = \frac{log(1-p)}{log(1-(1-e)^m)} \tag{19}$$

Por último, se encuentra el umbral T que indica el número mínimo de inliers requerido para considerar que la estimación del modelo es buena.

5.2.3 Estimación del movimiento

Se trata de la última fase del algoritmo, en la cual se pretende obtener la pose relativa entre dos imágenes, es decir, la rotación y translación realizada para la captura de la segunda imagen con respecto a la primera. Como ya se ha comentado, para la estimación del movimiento el algoritmo se basa en la geometría epipolar. De hecho, el primer paso consiste en calcular la matriz esencial, que posteriormente se descompondrá obteniendo la matriz de rotación y el vector de translación.

La matriz esencial únicamente tiene cinco grados de libertad, la matriz de rotación y la translación aportan tres grados de libertad. Sin embargo, al tratarse de una relación entre coordenadas homogéneas la matriz se encuentra definida salvo por un factor de escala[52]. Por tanto, sólo necesita un mínimo de cinco correspondencias para su obtención.

Dada una correspondencia formada por los puntos $p_1 = [x_1\ y_1\ z_1]'$ y $p_2 = [x_2\ y_2\ z_2]'$, cuyas coordenadas de la imagen se encuentran proyectadas sobre la esfera, para calcular la matriz esencial se debe resolver la ecuación mediante descomposición en valores singulares (SVD).

$$[x_1x_2 \quad y_1x_2 \quad z_1x_2 \quad x_1y_2 \quad y_1y_2 \quad z_1y_2 \quad x_1z_2 \quad y_1z_2 \quad z_1z_2] \cdot E = 0 \tag{20}$$

A partir de la matriz esencial se pueden obtener las matrices de proyección de las cámaras

$$E = [t]_x R = SR \tag{21}$$

donde S una matriz antisimétrica y R una matriz de rotación. Utilizando matrices auxiliares, siendo W ortogonal y Z antisimétrica:

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad Z = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{22}$$

En primer lugar, para obtener la rotación y translación se descompone en valores singulares SVD(E) = [U|S|V]:

$$S = UZU^T \tag{23}$$

$$R_1 = UWV^T \tag{24}$$

$$R_2 = UW^TV^T \tag{25}$$

Con esta factorización se pueden obtener la matriz de proyección de la segunda cámara mediante cuatro combinaciones posibles, y después obtener la imagen en el plano x' [60].

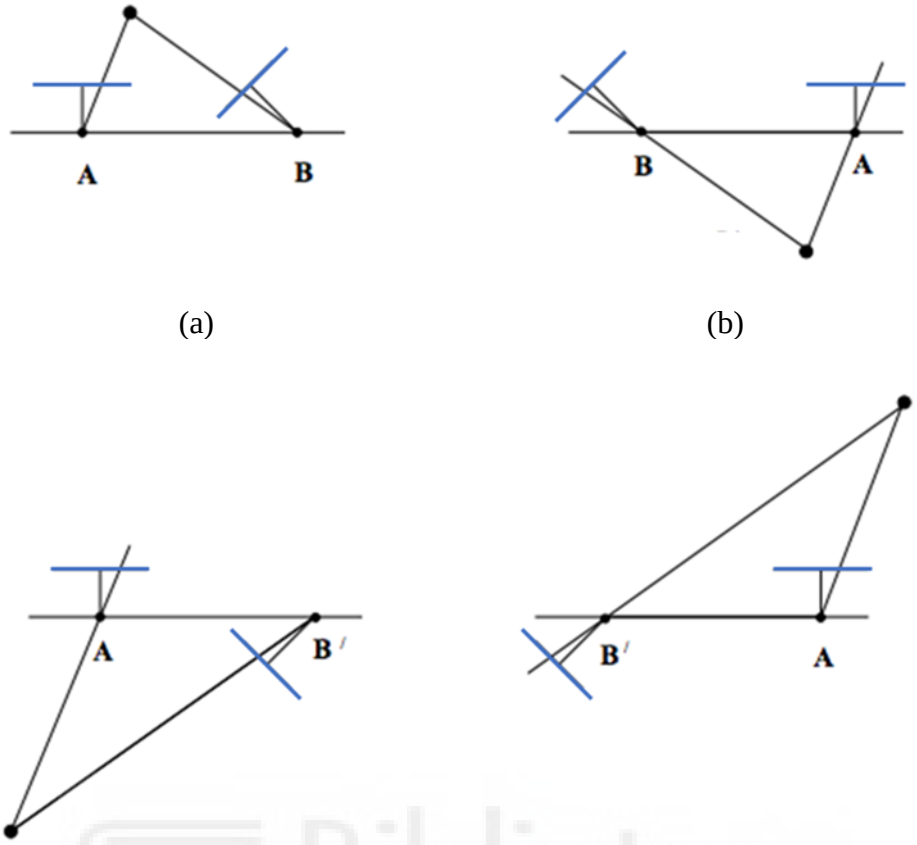
$$P1 = [R_1|t_x] \tag{26}$$

$$P2 = [R_1| - t_x] \tag{27}$$

$$P3 = [R_2|t_x] \tag{28}$$

$$P4 = [R_2| - t_x] \tag{29}$$

Para escoger la solución correcta se construyen los puntos 3D para las cuatro posibles configuraciones, y finalmente se escoge aquella en la que los puntos se encuentren por delante de la cámara. Al observar la Figura 24 se puede ver como únicamente en la (a) los puntos reconstruidos se encuentran delante de ambas cámaras.



(a) (b)

(c) (d)

Figura 24: Representación de las cuatro posibles soluciones.

Capítulo 6
Detalles de la implementación

En este capítulo se va a describir los pasos realizados a la hora de calibrar la cámara VIRB 360 para posteriormente crear una base de datos con imágenes y calcular las poses relativas de la cámara para la obtención de dichas imágenes. En la Figura 28 se muestra un esquema con las distintas etapas implementadas en este apartado.

6.1 Calibración

Se va a realizar la calibración de la cámara VIRB 360 mediante distintas aplicaciones, para finalmente escoger aquella que presente un menor error de reproyección.

El paso previo a obtener los parámetros de la cámara es realizar varias imágenes a un patrón de calibración como el que se muestra en la Figura 25 desde distintas posiciones. Como ya se ha mencionado en el capítulo 3, ente trabajo se tomarán las imágenes en modo RAW.

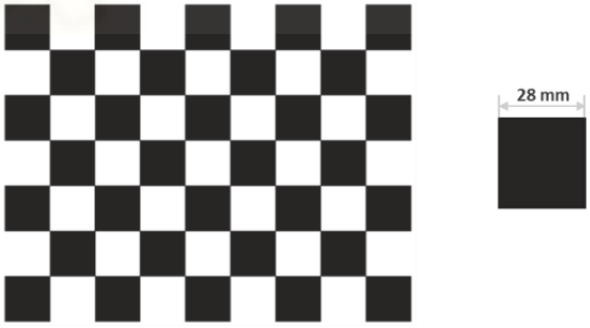


Figura 25: Patrón de calibración de 7 x 5 cuadrados (excluyendo los exteriores) de 28 mm.

A continuación, se muestran los métodos utilizados, los pasos que se han llevado a cabo y los resultados obtenidos.

6.1.1 SINGLE CAMERA CALIBRATOR APP

Se trata de una aplicación de MATLAB, a la cual se pude acceder mediante el comando *cameraCalibrator*. Dicha aplicación permite calibrar una cámara estimando los parámetros intrínsecos, extrínsecos y de distorsión de la lente. Además, se puede eliminar

los efectos de distorsión, realizar la medición de objetos planos o la reconstrucción de escenas[2].

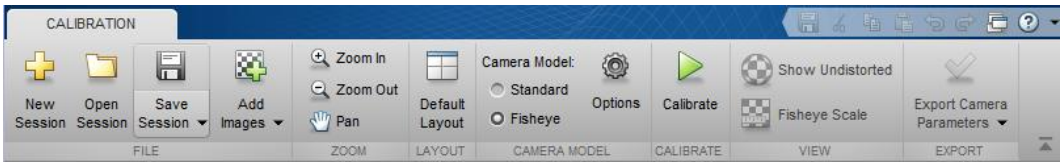


Figura 26: Barra de herramientas de *cameraCalibrator*[3].

El primer paso para realizar la calibración de la cámara haciendo uso de esta aplicación es agregar las imágenes, después aparecerá un cuadro de diálogo en el que se debe introducir el tamaño de los cuadrados del patrón de calibración, en este caso 28mm.

Una vez se hayan procesado las imágenes, como se puede ver en la imagen Figura 27, aparecerán los puntos detectados, de forma automática, del patrón mediante círculos verdes. Además, aparecerá indicado el origen (0,0) de los ejes con un cuadrado amarillo.

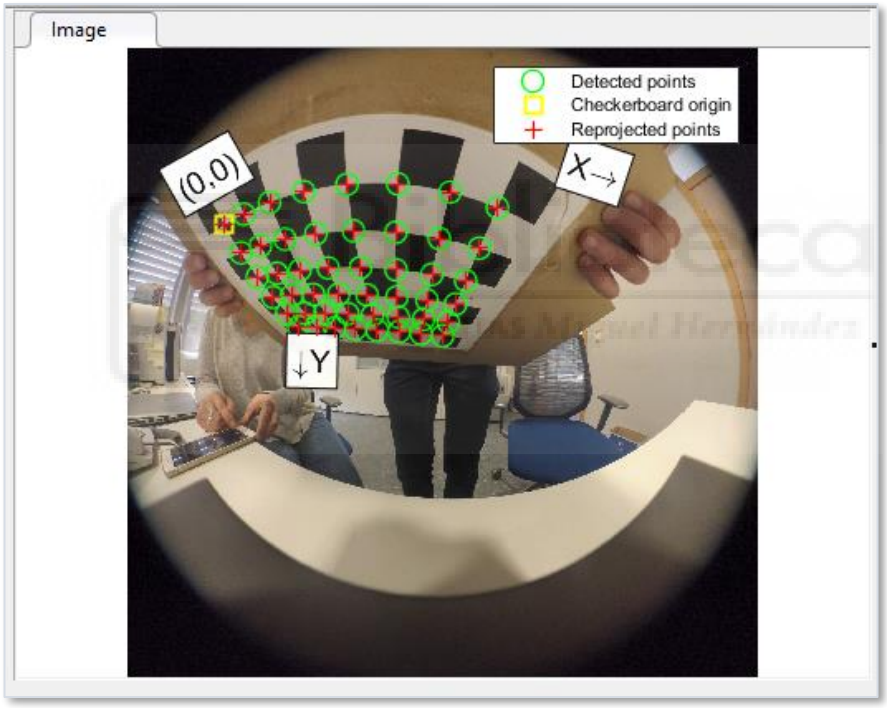


Figura 27: Detección de las esquinas. Camera Calibrator.

El último paso antes de realizar la calibración es configurar los parámetros de la cámara. En primer lugar, el modelo de la cámara. Para ello se elige la opción de *Fisheye* que se encuentra en la sección *Camera Model*. Después, se selecciona *Estimate Alignment* y finalmente se ejecuta la calibración.

Tras la calibración, la aplicación muestra en un gráfico de barras el error de reproyección promedio por imagen además del error promedio general. Además, resalta las

[2] Información extraída de: https://es.mathworks.com/help/vision/ug/single-camera-calibrator-app.html (Última consulta 18/05/2018).
[3] La versión utilizada de Matlab es la R2018a.

barras correspondientes a las imágenes que presentan un error mayor al general. Por otro lado, la aplicación muestra en un diagrama los parámetros extrínsecos. La aplicación también permite ver las imágenes sin distorsión.

6.1.2 OCAMCALB TOOLBOX

Se trata de una Toolbox para Matlab, creada por Davide Scaramuzza, la cual permite calibrar una cámara omnidireccional. Se puede emplear para cámaras catadióptricas y con lentes de ojo de pez de hasta 195º.[4]

La OcamCalb Toolbox se ejecuta descomprimiendo el archivo[5] y escribiendo en la ventana de comandos de Matlab ocam_calib, y aparecerá la siguiente barra de herramientas.

Primero, se han cargado las imágenes – las cuales deben estar en la misma carpeta que la Toolbox – proporcionándole a la aplicación el nombre base con el que han sido guardadas, así como el formato, en este caso jpg. Una vez realizado eso se ha seleccionado la opción *Extract Corners grid* de la barra de herramientas. La aplicación, antes de comenzar con la selección de esquinas, necesita que se introduzca el número de cuadrados existentes a lo largo del eje X y del eje Y, y el tamaño de cada uno para cada uno respecto de ambos ejes. Se ha establecido siete cuadrados a lo largo del eje X y cinco en el Y.

Como ya se ha comentado en el Capítulo 4, La Toolbox permite poder realizar la extracción de esquinas de forma automática o manual, pero antes se debe determinar si se quiere que la selección de las imágenes sea de forma automática – siempre que la extracción de las esquinas funcionará bien para el conjunto de datos – o si se desea que se procesen las imágenes de forma individual.

En este trabajo se ha optado por que sea individual y que la extracción sea manual. Dentro de esta opción, se puede seleccionar un detector de esquinas que permitirá refinar la posición de la esquina tratando de interpolar la ubicación alrededor de la posición que se escoja de forma manual. Para definir el tamaño de la ventana del detector de esquina se han utilizado los valores predeterminados.

Dado que se ha seleccionado extracción manual, se debe hacer clic en cada una de las esquinas. La primera esquina sobre la que se haga clic será el origen del marco de referencia, y los siguientes se deberán realizar siguiendo un orden a lo largo del eje Y. Se debe prestar atención en que se conserve la correspondencia de puntos con el resto de las imágenes. Por tanto, la primera esquina deberá ser la misma para todas.

Tras haber realizado el paso anterior con éxito, se ha seleccionado la opción *Calibration*. Dado que la aplicación pide que se determine el orden máximo del polinomio, el cual aproxima la función que proyecta cada punto de píxel al espacio 3D, se ha determinado que sea de orden cuatro.

Después, se ha selecciona el botón de *Calibration Refinement* para iniciar el algoritmo de Levenberg-Marquadt con el que se pretende minimizar la suma de los errores de

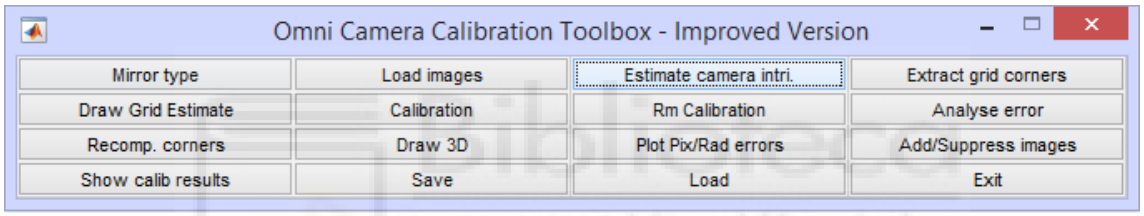
[4] Para más información https://sites.google.com/site/scarabotix/ocamcalib-toolbox .
[5] Se puede descargar en https://sites.google.com/site/scarabotix/ocamcalib-toolbox/ocamcalib-toolbox-download-page (Última consulta 19/05/2018). La versión utilizada es la v3.0.

reproyección al cuadrado. Se debe determinar el número de iteraciones, en este trabajo se ha optado por no limitarlo, es decir, se ha escogido la opción de sin límite. El siguiente paso que se ha realizado es la reproyección de las esquinas de acuerdo con los valores obtenidos en el paso anterior, mediante la opción *Reproject on images*. Al igual que la aplicación vista anteriormente, se puede ver la posición de cada uno de los patrones de calibración respecto del sistema de referencia de la cámara, presionando el botón *Show Extrinsic*

Se analiza el error de reproyección cometido de cada esquina, para ello la Toolbox dispone de la opción Analyse error. En la figura se pueden observar cada esquina mediante un punto, y con un determinado color que corresponde a la imagen que pertenece.

6.1.3 OMNIDIRECTIONAL CALIBRATION TOOLBOX

En este apartado se va a realizar la calibración de la cámara usando esta Toolbox, que fue desarrollada por Christopher Mei. Se trata de una aplicación para la cual no es necesario conocer la cámara previamente, únicamente se requiere de cuatro puntos para cada uno de los patrones de calibración. Se accede a ella escribiendo en la ventana de comandos de Matlab omni_calib_gui.[6]



El modelo de proyección que utiliza es una combinación del de Geyer y Barreto y una función de distorsión.

La primera opción que aparece en la ventana es la de seleccionar el tipo de cámara que se va a utilizar: (1) parábola, (2) catadióptrico o (3) dióptrico (lente ojo de pez), por lo que se ha escogido la opción tres.

Después de haber cargado las imágenes – que al igual que en la Toolbox de Scaramuzza deben encontrarse en el mismo directorio – y proporcionarle tanto el nombre base de las imágenes como el formato, se ha seleccionado la opción de Estimate camera intri. Para realizar una estimación de los parámetros intrínsecos, la aplicación pide que seleccione al menos tres puntos alineados. El siguiente paso que se ha realizado es el de extracción de las esquinas del patrón de calibración mediante la opción Extract grid corners. Después de haber seleccionado, en cada imagen las cuatro esquinas externas del patrón de calibración, comenzando por el origen y en sentido de las agujas del reloj, se ha pasado a obtener los parámetros de la cámara.

Esta aplicación, al igual que la de Scaramuzza, permite observar de forma gráfica el error de reproyección mediante la opción Analyse error. Cada color representa una imagen, por lo que al presionar sobre uno de ellos se podrá observar a que imagen pertenece ese punto y en que coordenadas se encuentra. Dado que hay puntos que presentaban

[6] Se puede descargar la Toolbox en la siguiente página:
http://www.robots.ox.ac.uk/~cmei/Toolbox.html

un error elevado, se ha recalculado la extracción de puntos, presionando sobre el botón de Recomp. Corners.

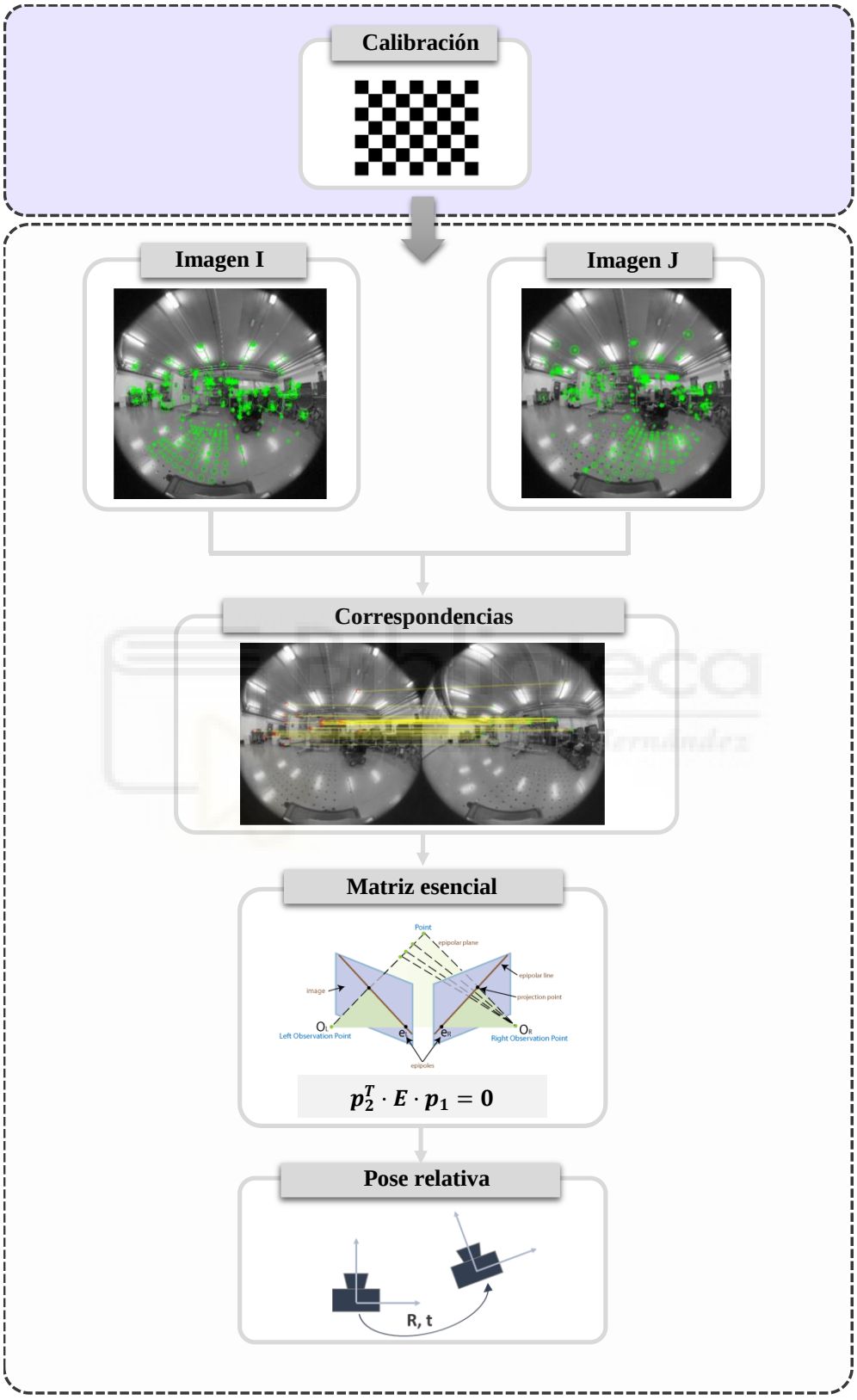


Figura 28: Esquema general con los pasos implementados en este trabajo.

6.2 Odometría visual

En esta sección se van a definir las distintas fases que se han implementado en este trabajo en el algoritmo que permite obtener las poses relativas entre dos imágenes tomadas con la cámara VIRB 360.

6.2.1 Adquisición de imágenes

La primera etapa de este algoritmo consiste en la creación de una base de datos con distintas imágenes que se van a emplear como fuente de información para las siguientes fases. En la Figura se pueden observar las imágenes adquiridas posicionando la cámara sobre una rejilla de 4x4 que presenta una distancia tanto vertical como horizontal entre dos posiciones consecutivas de 20 cm. Este proceso se ha repetido tres veces: las primeras dieciséis imágenes se han obtenido sin rotar la cámara, después se han tomado rotándola aproximadamente 30º sobre su eje Z y para finalizar haciéndolo sobre su eje Y.

Por tanto, como resultado se ha obtenido una base de datos compuesta por cuarenta y ocho imágenes. La posición en la rejilla y orientación de la cámara para cada una de las imágenes capturadas es conocida y se encuentra guardada en un fichero de Matlab. Dicha información se utilizará posteriormente para comparar los resultados de translación y orientación que se obtengan tras realizar las siguientes etapas con los reales y por tanto calcular el error cometido.

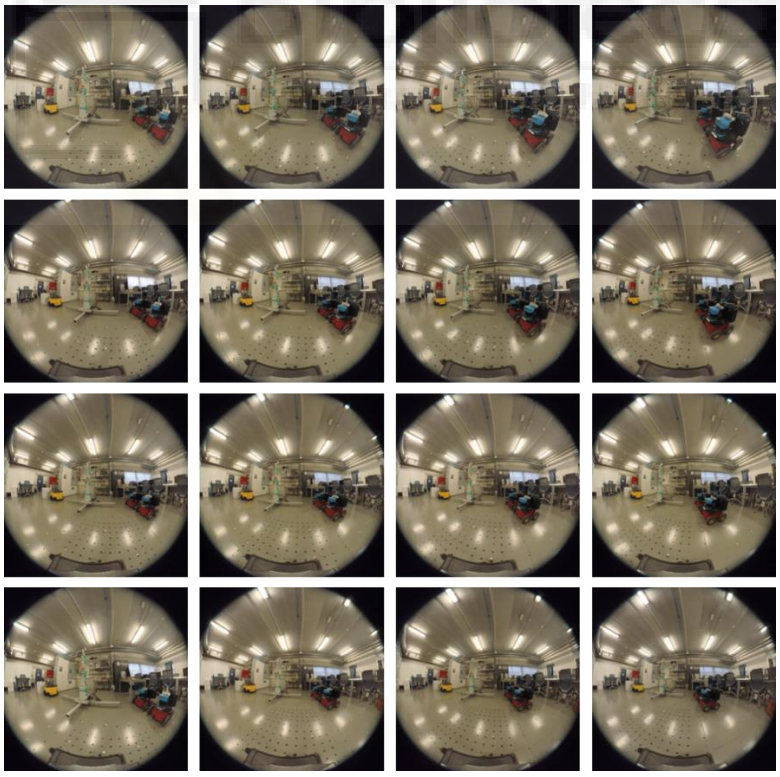


Figura 29: Imágenes tomadas desde distintas posiciones de una rejilla de 4x4 (sin rotar la cámara).

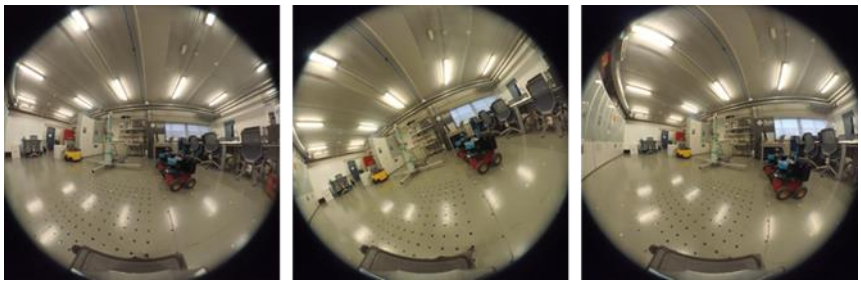


Figura 30: De izquierda a derecha: sin rotar la cámara, rotándola 30º sobre su eje Z y 30º sobre su eje Y.

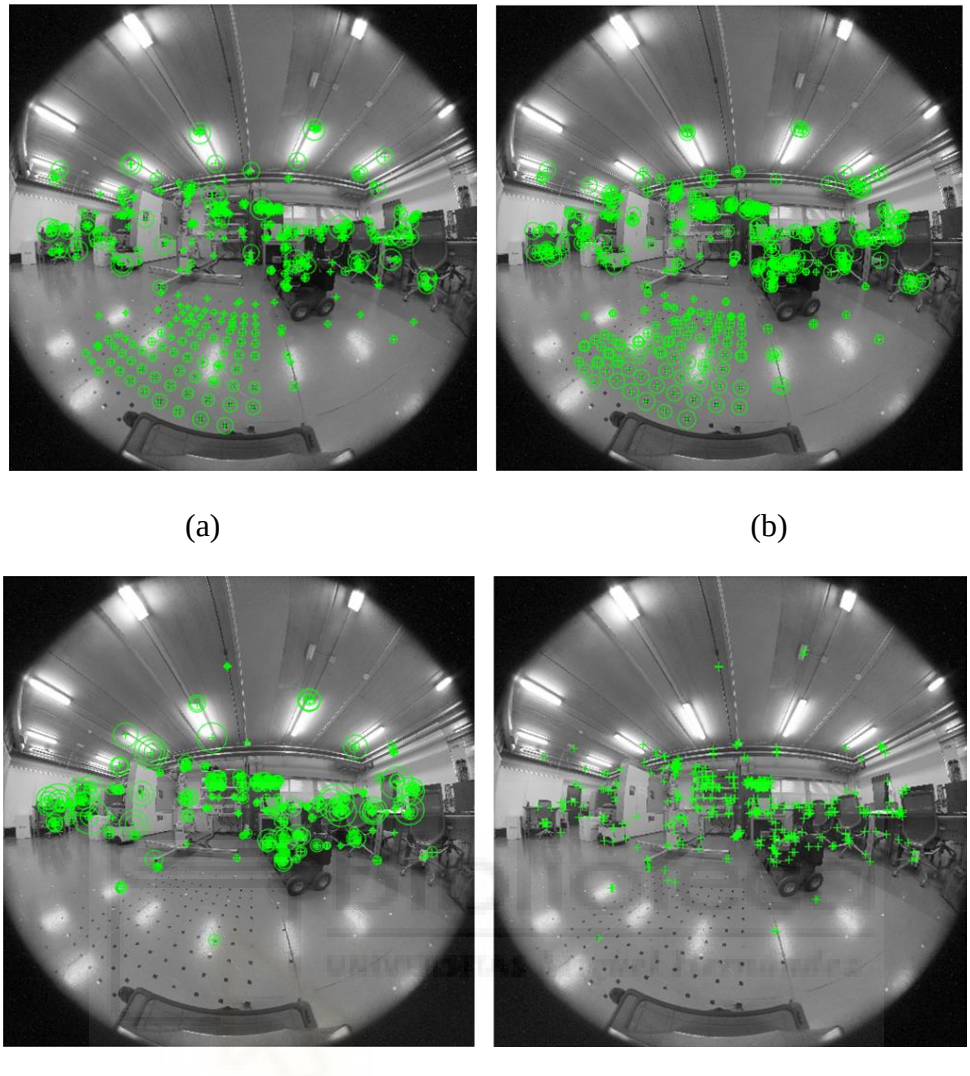
6.2.2 Detectar y describir puntos característicos

El primer paso en esta fase consiste en obtener los puntos característicos de las imágenes utilizando los métodos expuestos en la sección 5.2.1. Para esta parte del algoritmo se ha creado una función en Matlab, la cual los parámetros de entrada que recibe son una imagen en escala de grises y una variable que adquiere un valor entero el cual determinará el tipo de puntos característicos que se pretende detectar. Dependiendo del valor de este último parámetro se llamará a la función correspondiente que se encuentra incluida en la Toolbox de Matlab.

En la siguiente tabla se puede observar en la primera columna los posibles valores del segundo parámetro de entrada de la función, en la segunda columna el tipo de característica que corresponde a dicho valor y en la última la función de Matlab que corresponde a ese tipo.

Tabla 4: Tipos de detectores implementados y sus correspondientes funciones en Matlab.

	Tipo de característica	**Función de Matlab**
1	SURF	detectSURFFeatures
2	KAZE	detectKAZEFeatures
3	BRISK	detectBRISKFeatures
4	HARRIS	detectHarrisFeatures

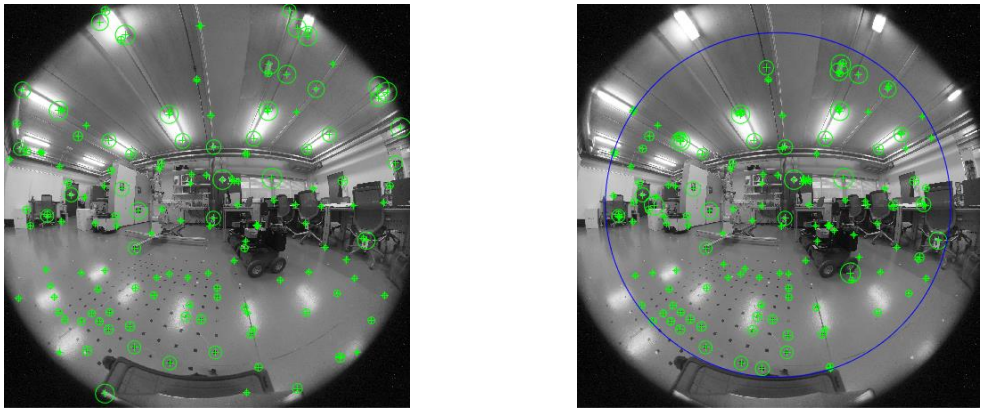


(a) (b)

(c) (d)

Figura 31: Tipos de puntos detectados: (a) SURF, (b) KAZE, (c) BRISK y (d) HARRIS.

Las funciones incluidas en Matlab devolverán un objeto con las coordenadas de los puntos detectados. Dado que existe una zona de solape entre la cámara delantera y la trasera, se requiere eliminar aquellos puntos de interés que se encuentren en ella. Con este objetivo se ha creado otra función, que recibirá como entrada las coordenadas de todos los puntos detectados y se pasarán a coordenadas polares. Se eliminarán aquellos cuyo valor de ρ sea mayor que el radio calculado según los grados de FOV que se requiera. En la siguiente figura se puede ver en la imagen de la izquierda todos los puntos característicos detectados en el paso anterior, y en la de la derecha se muestran los resultantes después de llamar a dicha función escogiendo 170º.



(a) (b)

Figura 32: Detección de puntos SURF: (a) todos, (b) eliminados los de la zona de solape.

El siguiente paso tras haber detectado los puntos característicos es extraer sus descriptores. Para este objetivo se ha utilizado la función implementada en la Toolbox de Matlab, denominada extractFeatures. Se le dará como argumento de entrada la imagen y el subconjunto de puntos escogidos, la función determinará el método de extracción de los descriptores en función del tipo de objeto en el que se encuentran almacenados los puntos que recibe. En la siguiente tabla se puede observar el método que emplea esta función para cada una de las clases de objeto.

Tabla 5: Descriptor utilizado en función del tipo de puntos de interés en Matlab.

Tipo de objeto	Método
SURFPoints	SURF
BRISKPoints	BRISK
KAZEPoints	KAZE
cornerPoints	FREAK

6.2.3 Correspondencia de puntos característicos

Después de haber realizado todos los pasos anteriores, el siguiente es encontrar correspondencias entre los puntos, es decir, buscar si un punto de interés correspondiente a una imagen se encuentra presente en la otra. Un aspecto importante a tener en cuenta en esta fase es que pueden aparecer falsas correspondencias, obteniendo un resultado erróneo. Debido a que existen ocasiones en las que no es fácil determinar a simple vista si las correspondencias son correctas, se debe utilizar algún método para eliminarlas. En este trabajo se podrá comparar los resultados al utilizar RANSAC con los obtenidos al no implementar este método en el algoritmo.

En primer lugar, para comparar los diversos descriptores obtenidos en la etapa anterior y obtener un emparejamiento de estos, se ha utilizado la función denominada matchFeatures, la cual se encuentra implementada en Matlab. Esta función recibe como parámetro de entrada los descriptores, de cada imagen además de una serie de opciones. Por ejemplo, al establecer que el parámetro 'Unique' tiene valor true, se especifica que las correspondencias que devuelva son únicas, es decir, un descriptor de la imagen primera sólo podrá coincidir con otro de la segunda imagen y no con más.

6.2.4 Estimación del movimiento

Se trata de la última etapa del algoritmo para la obtención del movimiento de la cámara, que se divide en dos partes. Como ya se ha comentado en el capítulo anterior, las proyecciones de una misma escena en dos imágenes se encuentran relacionadas entre sí mediante la geometría epipolar. Dado que se conocen los parámetros intrínsecos de la cámara, dicha relación viene determinada por la matriz esencial, por tanto, el primer paso es la obtención de la misma.

En primer lugar, se obtienen las proyecciones P sobre la esfera unidad de los puntos en coordenadas de la imagen. La Toolbox de Scaramuzza incluye una función denominada cam2world para ello.

El siguiente paso es obtener la matriz esencial, en un primer momento se intentó utilizar la que se encuentra implementada en Matlab, pero finalmente se descartó esta idea ya que esta no sirve para cámaras no lineales. Los pasos implementados para la estimación de la matriz esencial se detallan a continuación.

Una vez obtenido las coordenadas de proyección de cada par de correspondencia, $p_1 = [x_1\ y_1\ z_1]'$ y $p_2 = [x_2\ y_2\ z_2]'$, se pasa a resolver la condición de epipolaridad (10) obteniendo la matriz esencial.

$$[x_1x_2 \quad y_1x_2 \quad z_1x_2 \quad x_1y_2 \quad y_1y_2 \quad z_1y_2 \quad x_1z_2 \quad y_1z_2 \quad z_1z_2] \cdot E = 0$$

El método mediante el cual se resuelve es el de descomposición en valores singulares (SVD) siendo necesario tener al menos cinco correspondencias para ello. Sabiendo que la matriz se puede descomponer en una rotación y translación, en este trabajo se ha simplificado el sistema de ecuaciones dado que se conoce la forma que debe tener la matriz esencial al saber los movimientos posibles de la cámara.

$$E = [t]_x R = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{30}$$

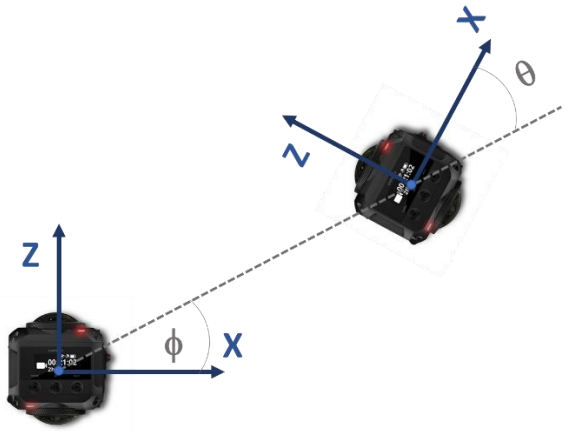


Figura 33: Movimiento planar de la cámara.

Se trata de un movimiento planar, compuesto por una rotación θ sobre el eje Y de la cámara y una translación definida por el ángulo ϕ.

$$R = \begin{bmatrix} cos\theta & 0 & sen\theta \\ 0 & 1 & 0 \\ -sen\theta & 0 & cos\theta \end{bmatrix} \qquad T = \begin{bmatrix} sen\phi \\ 0 \\ cos\phi \end{bmatrix} \tag{31}$$

Obteniendo, por tanto, como resultado una matriz de la siguiente forma

$$E = \begin{bmatrix} 0 & \cos(\theta - \phi) & 0 \\ -cos\phi & 0 & sen\phi \\ 0 & sen(\theta - \phi) & 0 \end{bmatrix} = \begin{bmatrix} 0 & e_2 & 0 \\ e_4 & 0 & e_1 \\ 0 & e_3 & 0 \end{bmatrix} \tag{32}$$

Tras haber adaptado la matriz esencial a un movimiento planar, el sistema de ecuaciones (20) queda de la siguiente forma:

$$[x_1 \cdot y_2 \quad y_1 \cdot x_2 \quad y_1 \cdot z_2 \quad z_1 \cdot y_2] \cdot E = 0 \tag{33}$$

Todo lo mencionado hasta ahora de como estimar la matriz esencial, se encuentra en una función la cual, en el caso de que se seleccione la opción de RANSAC será llamada cada vez que la función donde se ha implementado dicho algoritmo se encuentre en el paso de obtener el modelo para un subconjunto. En lo referente a determinar si dicho modelo es aceptable, se utilizará la restricción epipolar para ello y se calculará la distancia

El segundo paso que se realiza en esta etapa es descomponer la matriz esencial calculada obteniendo la rotación y translación correspondiente al movimiento relativo entre los sistemas ópticos de la cámara al obtener las imágenes. El método utilizado para ello es el que se encuentra implementado en [60]. Dado que se conoce la forma de la matriz esencial:

$$\phi = \arctan\left(\frac{e_1}{-e_4}\right) = \arctan\left(\frac{sen\phi}{-cos\ \phi}\right) \tag{34}$$

Las dos posibles translaciones resultantes son:

$$t_1 = [sen\ \phi, 0, \cos\phi] \tag{35}$$

$$t_2 = [sen\ \phi + \pi, 0, \cos\phi + \pi] \tag{36}$$

Una vez obtenido ϕ se pasa a calcular θ de igual forma:

$$\theta = \arctan\left(\frac{e_3}{e_2}\right) = \arctan\left(\frac{sen\ \theta - \phi}{cos\ \theta - \phi}\right) + \phi \tag{37}$$

Al igual que ocurre con la translación, en el caso de la rotación también existe dos posibles soluciones:

$$R_1 = \begin{bmatrix} cos\theta & 0 & sen\theta \\ 0 & 1 & 0 \\ -sen\theta & 0 & cos\theta \end{bmatrix} \tag{38}$$

$$R_2 = \begin{bmatrix} 2\cos^2\phi - 1 & 2\cos\phi sen\,\phi & 0 \\ 2\cos\phi sen\,\phi & 2sen^2\phi - 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{39}$$

Como ya se ha comentado en el apartado 5.2.3 existen cuatro posibles soluciones, de las cuales una es la correcta, aquella que, al reconstruir los puntos 3D, se encuentren delante de ambas cámaras.

El método utilizado para ello es el que se expone a continuación. Se debe obtener para cada correspondencia los rayos de proyección de un punto 3D de la escena X, con el fin de poder obtener su signo (r y r'). Por tanto, la solución correcta será aquella cuyos valores de r y r' sean positivos, lo que querrá decir que existe intersección frente de ambas cámaras. Siendo la transformación entre poses:

$$X = t_x + RX' \tag{40}$$

$$rx = b + r_1 Rx' \tag{41}$$

Puede ser formulado como un problema de mínimos cuadrados de la siguiente forma:

$$[x_1 \quad -b]\begin{bmatrix} r/r' \\ 1/r_1 \end{bmatrix} = [Rx'] \tag{42}$$

Finalmente, la solución se extrae utilizando la técnica d la seudo inversa:

$$P = (A^T A)^{-1} A^T c \tag{43}$$

El conjunto de rotación R y translación t correcto es aquel que, para cada par de correspondencia, tanto r como r' son positivos, denominándose solución positiva. En la práctica no siempre se produce que dichos valores sean positivos para todos los pares de correspondencias debido a la presencia de incertidumbre presente en ellas, a pesar de que la solución sea la correcta. Por ello, se escogerá aquella que disponga de un mayor número de soluciones positivas.

Capítulo 7
Resultados

7.1 Calibración

En primer lugar, se ha realizado la calibración de la cámara mediante la Toolbox que proporciona Scaramuzza, utilizando su modelo para estas cámaras explicado en el apartado 4.2. Los resultados obtenidos, tras llevar a cabo los pasos expuestos en el apartado 6.1.2, se muestran a continuación.

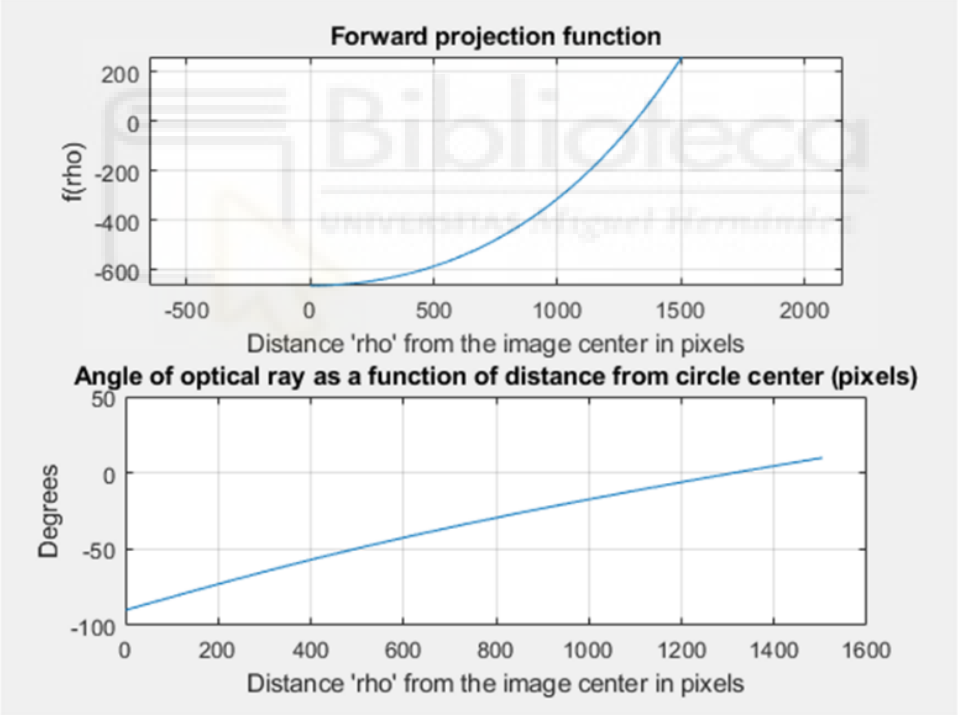


Figura 34: Representación de la función de proyección y del ángulo del rayo óptico con respecto a ρ.

Los parámetros que se obtienen en la calibración mediante el modelo de Scaramuzza son los coeficientes del polinomio de la función de proyección $f(\rho)$. Dado que según Scaramuzza el coeficiente de orden uno es cero, en la tabla se puede ver los valores obtenidos para el resto de los coeficientes. También se tiene el centro de distorsión [cx, cy] y los elementos de la matriz *skew* (*vid.* Ecuación (8)).

Tabla 6: Resultados de la calibración mediante la Toolbox de Scaramuzza.

Parámetros	Valor
[a0 a2 a3 a4]	[-666.2503 3.0602e-04 1.94006e-09 4.47702e-11]
[cx cy]	[1.499e+03 1.5024e+03]
c	1.000468
d	-0.000437
e	-0.002089
[fila col]	[3000 3008]

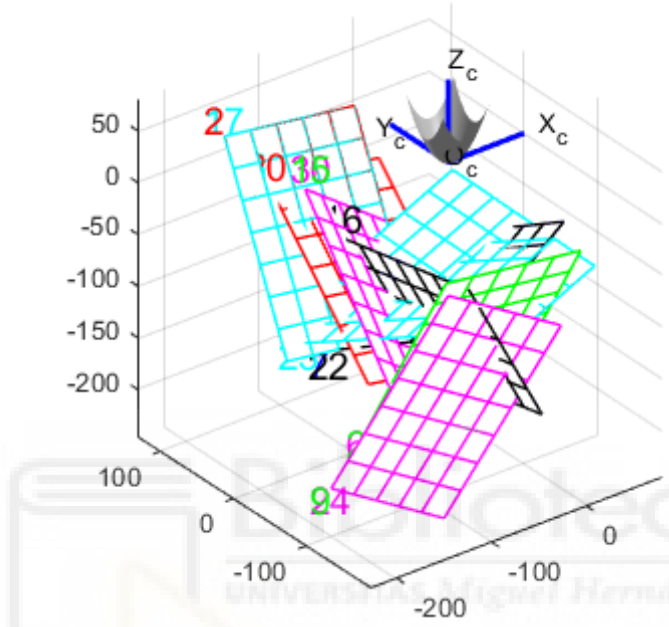


Figura 35: Representación parámetros extrínsecos (Scaramuzza).

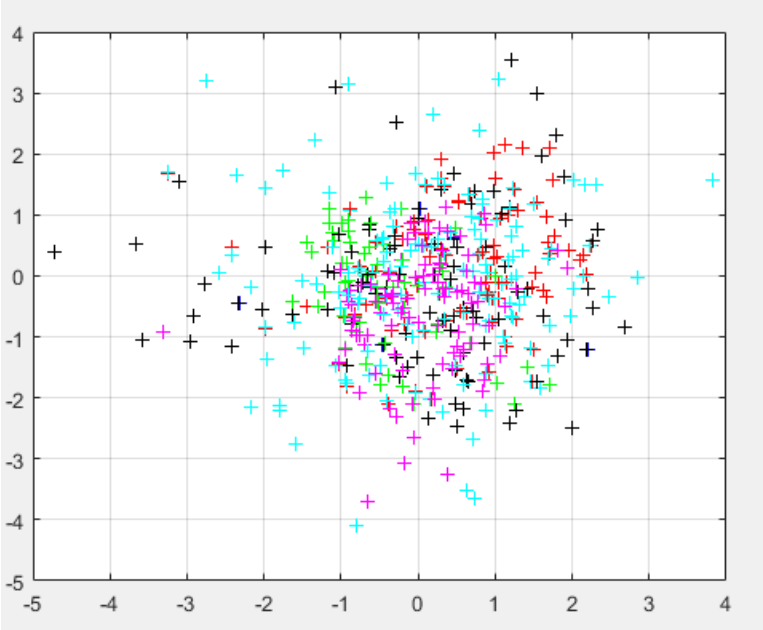


Figura 36: Error de reproyección (Scaramuzza).

Como ya se ha comentado Matlab dispone de una aplicación denominada Camera Calibrator, cuyo algoritmo de calibración, para la opción de cámara fisheye, utiliza el mismo modelo para la cámara que la Toolbox anterior, es decir, el de Scaramuzza. Finalmente, tras haber realizado los pasos detallados en 6.1.1 se han obtenido los siguientes valores que definen el modelo.

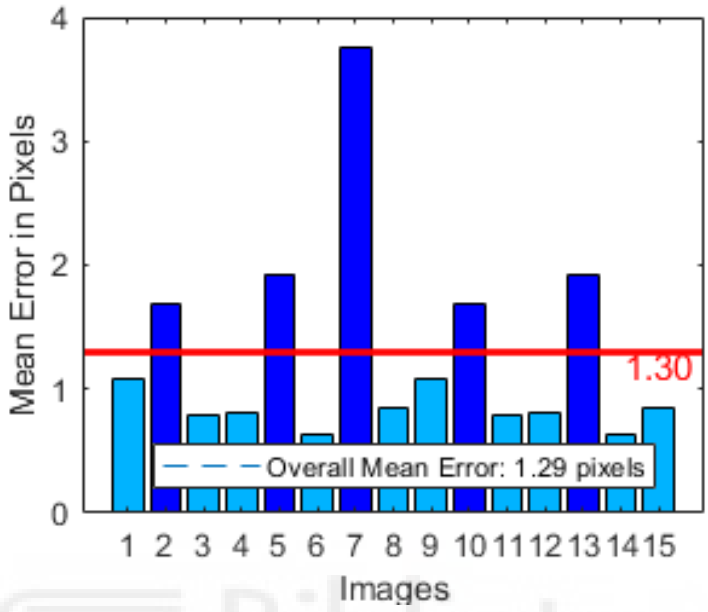
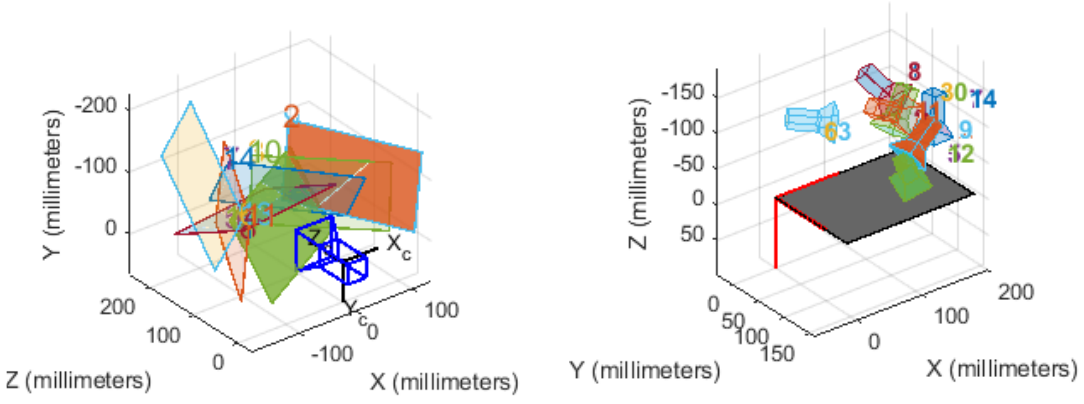


Figura 37: Diagrama de barras del error de reproyección (App Matlab).

En la Figura 37 se puede ver el valor medio de reproyección obtenido tras realizar la calibración. Se muestra en un diagrama de barras donde cada una representa el error debido a esa imagen, además se puede ver como las imágenes 2, 5, 7, 10 y 13 tienen un error de reproyección superior a la media.



(a) (b)

Figura 38: Representación parámetros extrínsecos (App Matlab): (a) vista centrada en la cámara, es decir, la cámara se encontraba estacionaria cuando se capturaron las imágenes, y (b) de la vista centrada en el patrón, este estaba estacionario mientras la cámara se disponía en distintas posiciones.

A continuación, se muestra en una tabla los valores de los parámetros de calibración obtenidos.

Tabla 7: Resultados de la calibración mediante CameraCalibrator.

Parámetros	Valor
[a0 a2 a3 a4]	[669.2938 -2.8548e-04 -2.7716e-08 -3.4346e-11]
[cx cy]	[1.4854e+03 1.5022e+03]
c	0,9976
d	-0,0179
e	0,0117
[fila col]	[3000 3008]

La tercera aplicación que se ha utilizado para la calibración de la cámara es la Toolbox creada por Christopher Mei, cuyo modelo utilizado es el expuesto en 4.1. En este modelo los parámetros que lo definen es la distancia focal γ, el punto principal cc= [u_0 v_0], skew y los coeficientes de distorsión Kc.

Tabla 8: Resultados de la calibración mediante la Toolbox de Mei.

Parámetros	Valor
[γ_1 γ_2]	[1.4205e+03 1.4200e+03]
[u_0 v_0]	[1.5100e+03 1.5010e+03]
Kc	[0.1151 -0.0084 -9.3319e-04 1.6944e-04 0]
ξ	1.1779

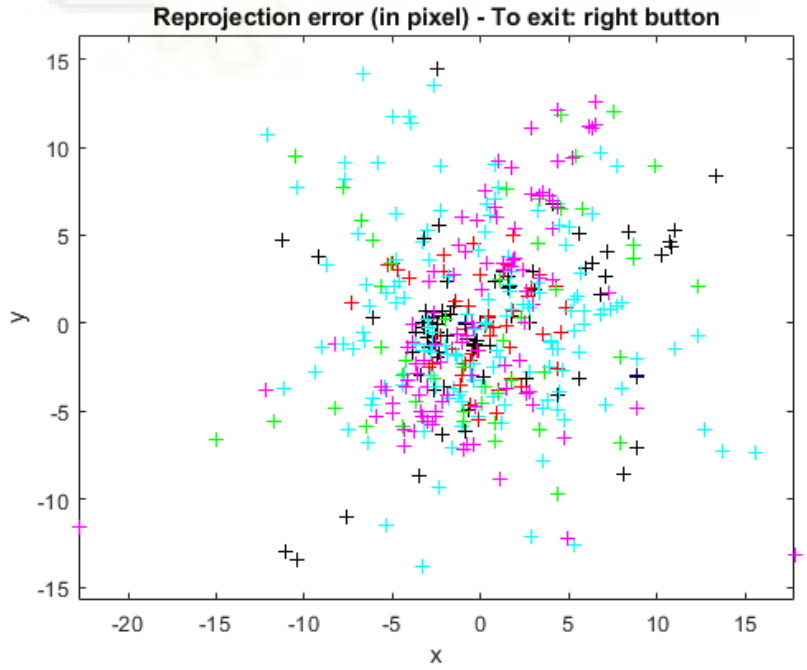


Figura 39: Errores de reproyección (Mei).

Por último, se van a mostrar en una tabla los valores de error obtenidos tras utilizar cada una de las aplicaciones ya comentadas, para poder compararlas.

Tabla 9: Errores de reproyección obtenidos.

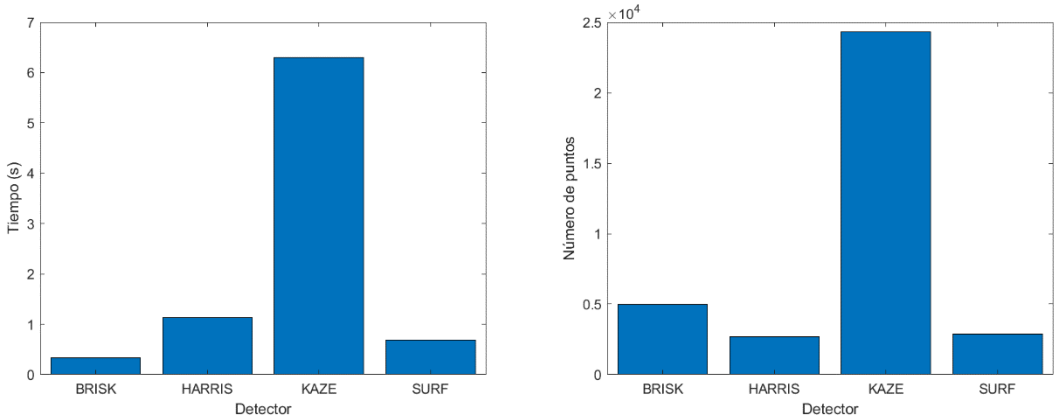
Aplicación	Valor error
ocam_calib	1.434446
Camera Calibrator	1.29
omni_calib_gui	[1.35803 1.50049]

Tras observar la Tabla 1, se obtiene la conclusión que de todos los métodos utilizado para obtener los parámetros de calibración el que obtiene un menor error de reproyección es la aplicación implementada en Matlab. Para la siguiente fase de este trabajo se va a utilizar los parámetros de la cámara obtenidos con esta aplicación, aunque sería conveniente probar también con los resultados obtenidos con el resto de las aplicaciones en un futuro. Un aspecto que destacar de esta fase es que se han obtenido ciertos inconvenientes con algunas de las imágenes en todas las aplicaciones utilizadas, en el caso en que la detección de las esquinas del patrón era automática la misma aplicación descartaba imágenes, y si era manual, a pesar de clicar sobre una esquina la aplicación ponía el punto en otro sitio y por tanto se eliminaba dicha imagen para la obtención de la calibración ya que provocaba un error más elevado.

7.2 Odometría Visual

Dado que en el apartado 6.2 se ha explicado todos los pasos realizados en este trabajo para obtener las poses relativas, en este se van a mostrar los resultados obtenidos.

En primer lugar, se va a realizar un estudio acerca de los detectores utilizados para observar el tiempo computacional medio que emplean, así como el número de puntos característicos medio que detectan. Los resultados que se muestran a continuación se han obtenido teniendo en cuenta únicamente la parte de detectar puntos característicos no de todo el algoritmo completo.



(a) (b)

Figura 40: Etapa de detección de características: (a) coste computacional requerido por cada tipo de descriptor implementado. (b) Número de puntos que detecta cada tipo.

El detector de características que requiere de un mayor tiempo computacional, al observar los resultados obtenidos Figura 40 (a), es el detector KAZE. De hecho, la diferencia de segundos con el resto de los métodos de detección de características es significativa. Por el contrario, el que ha obtenido los puntos característicos en un menor tiempo es el detector BRISK. Sin embargo, también es el detector KAZE el que más puntos detecta Figura 40 (b), siendo HARRIS el que menos.

A continuación, se van a mostrar algunas graficas con los resultados obtenidos, para estudiar si estos son satisfactorios. En primer lugar, se va a mostrar un diagrama de barras en la que cada una corresponde a un tipo de detector y se indica cuál ha sido el error medio obtenido, sin utilizar RANSAC.

Se puede ver que los detectores BRISK y HARRIS presentan un error mayor, con poca diferencia entre ellos. Lo mismo ocurre con KAZE y SURF, pero con un error menor en comparación con los anteriores. En el caso de tener que escoger un tipo, en función del error medio cometido, sería puntos SURF ya que, aunque KAZE presenta también un error menor, requiere de un mayor coste computacional, siendo por tanto una desventaja si se quiere implementar en tiempo real.

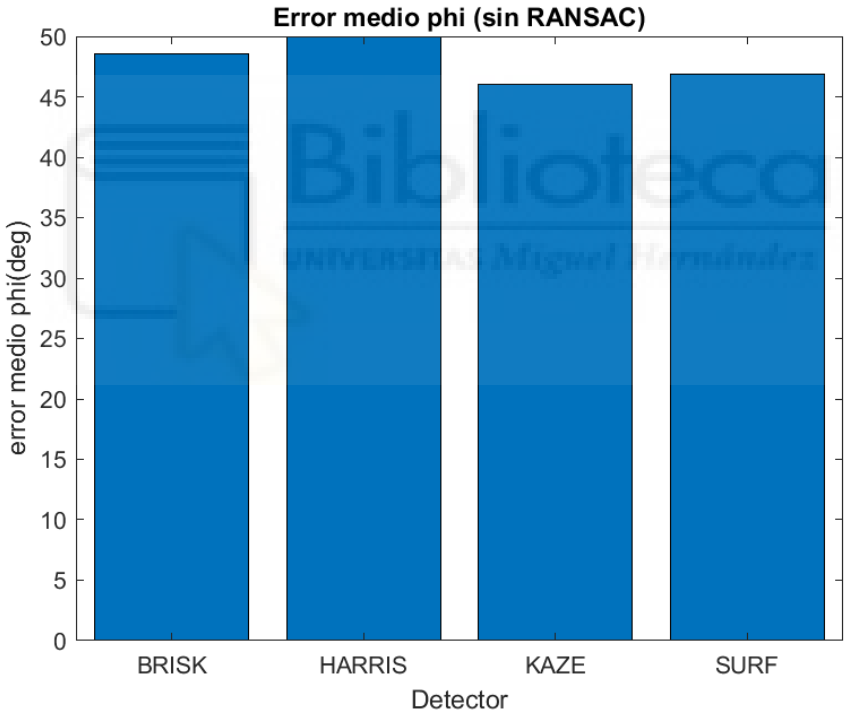


Figura 41: Error medio del ángulo ϕ en función del tipo de detector utilizado (sin RANSAC).

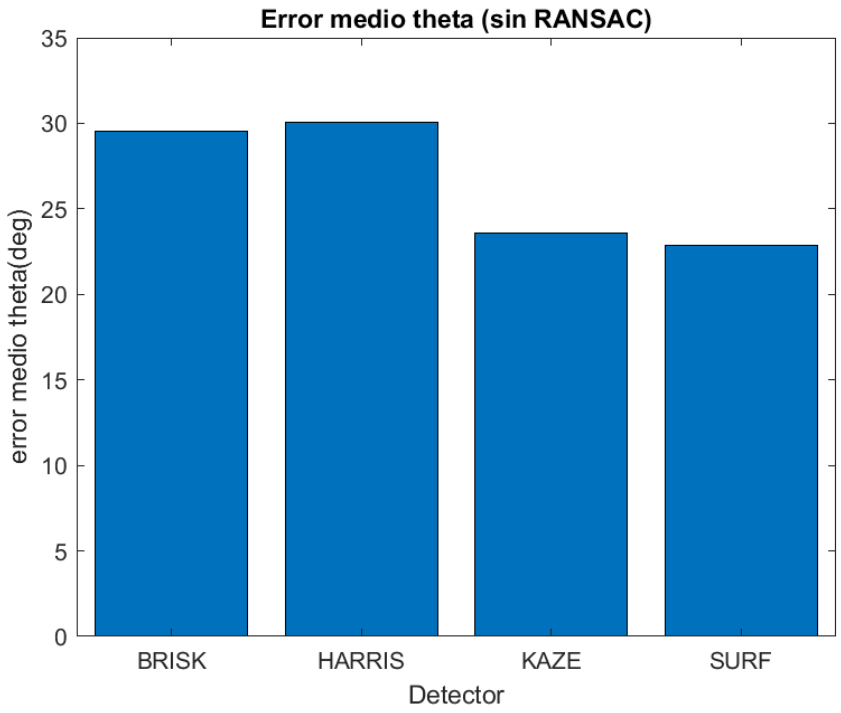


Figura 42: Error medio del ángulo θ en función del tipo de detector utilizado (sin RANSAC).

Con el fin de comparar e intentar obtener un error menor, se va a mostrar los resultados obtenidos al implementar RANSAC para eliminar falsas correspondencias.

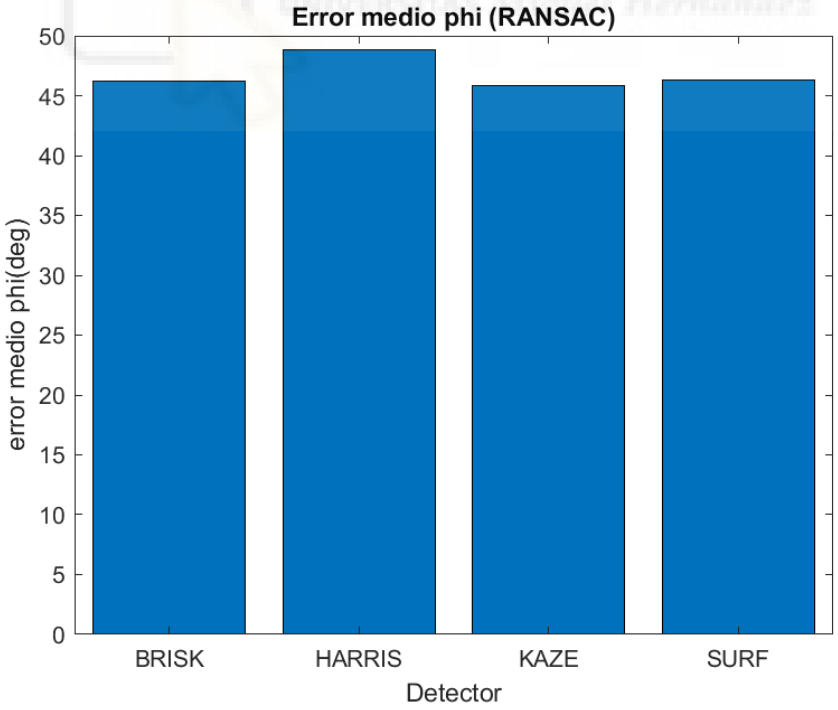


Figura 43: Error medio del ángulo ϕ en función del tipo de detector utilizado (RANSAC).

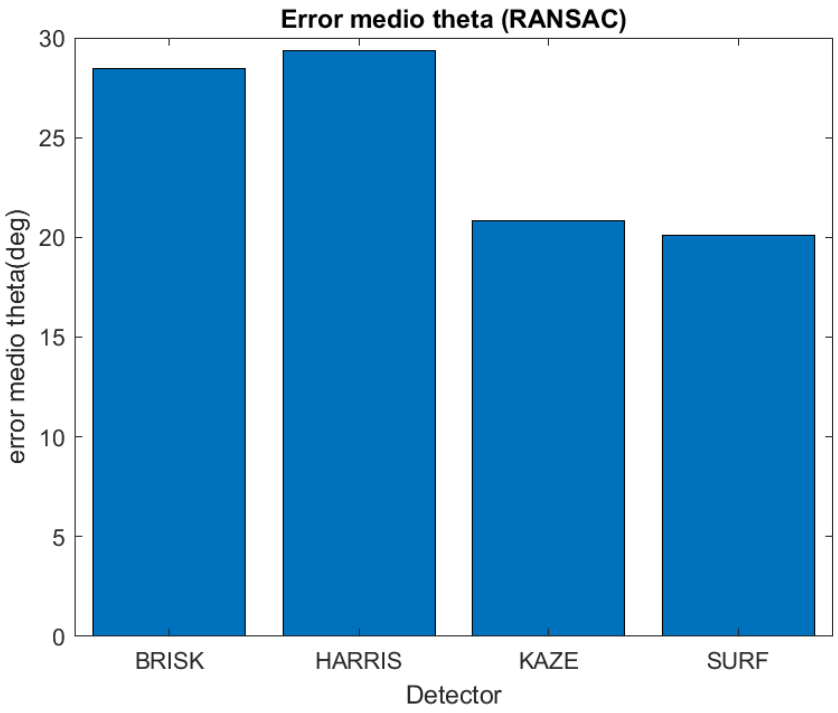


Figura 44: Error medio del ángulo θ en función del tipo de detector utilizado (RANSAC).

Al utilizar RANSAC el error medio al estimar los ángulos que definen el movimiento de la cámara no han variado mucho, únicamente se ha producido una disminución de apenas unos grados, de hecho, apenas es apreciable en los diagramas de barras. Hay que tener en cuenta que los errores son elevados debido a que se trata de una cámara con lente de ojo de pez, es decir, no lineal.

El siguiente paso es ver el comportamiento del error en función de la distancia recorrida para tomar la segunda imagen.

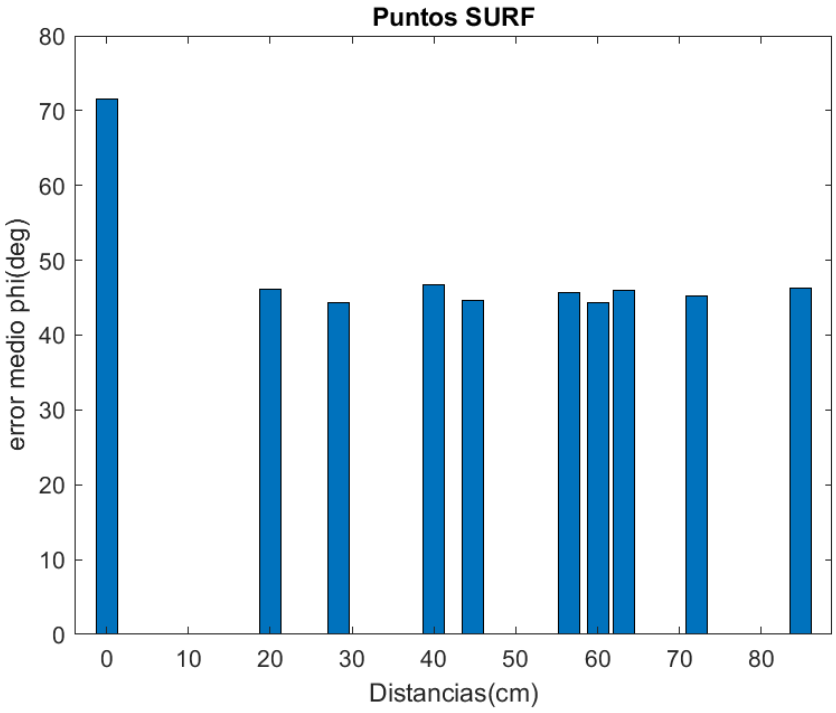


Figura 45: Error medio de ϕ obtenido mediante puntos SURF en función de la distancia recorrida.

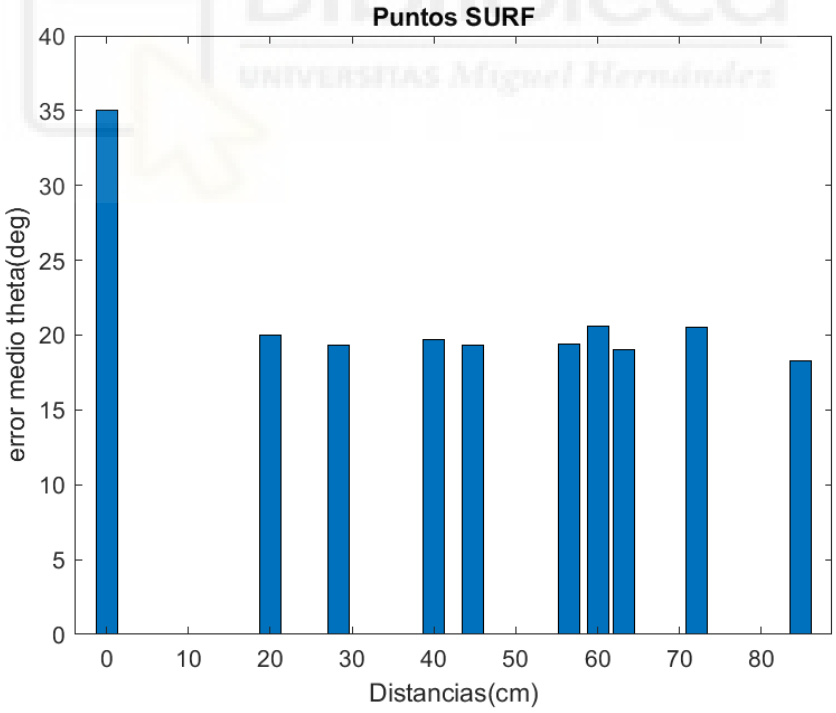


Figura 46: Error medio de θ obtenido mediante puntos SURF en función de la distancia recorrida.

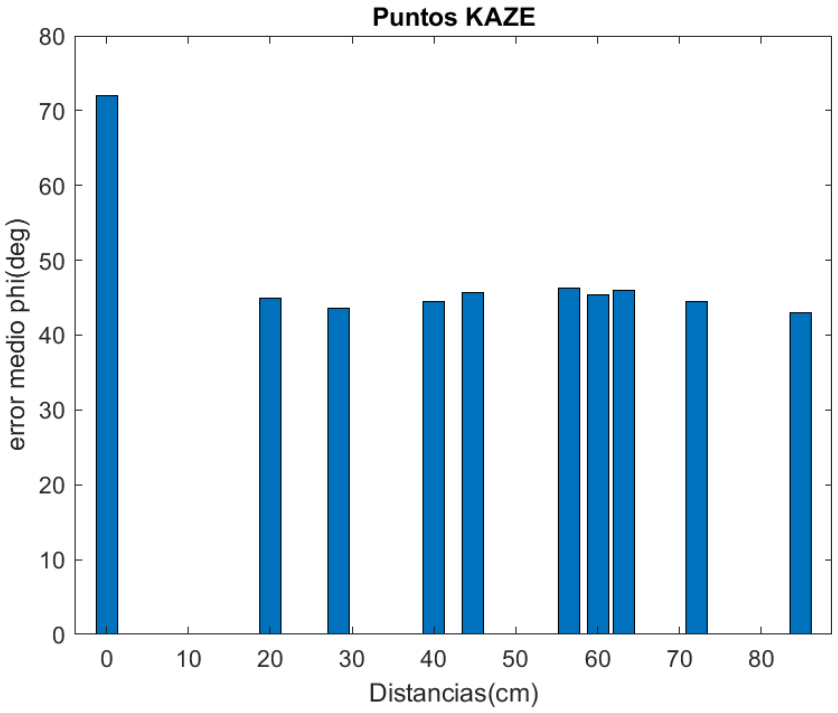


Figura 47: Error medio de ϕ obtenido mediante puntos KAZE en función de la distancia recorrida.

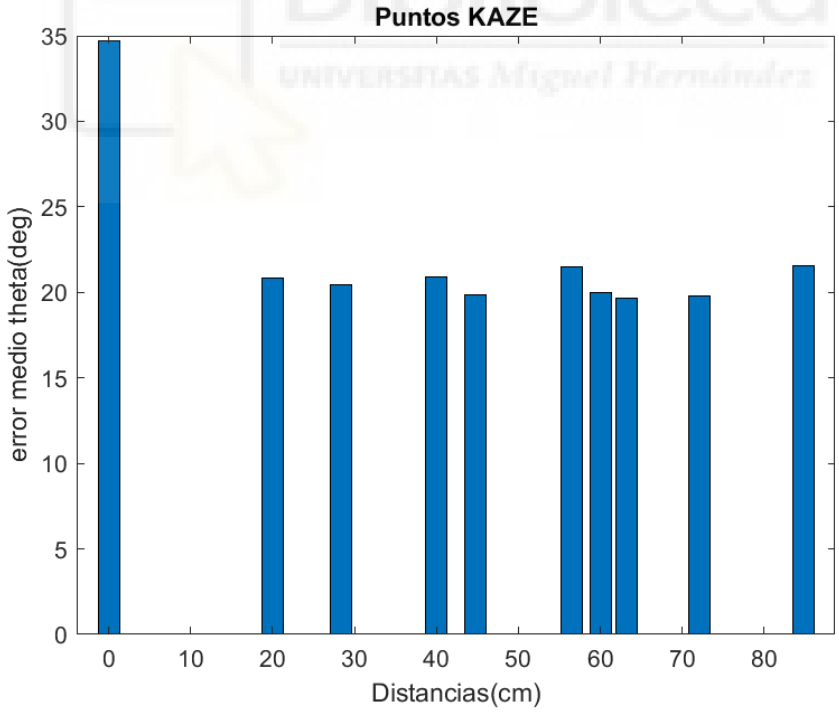


Figura 48: Error medio de θ obtenido mediante puntos KAZE en función de la distancia recorrida.

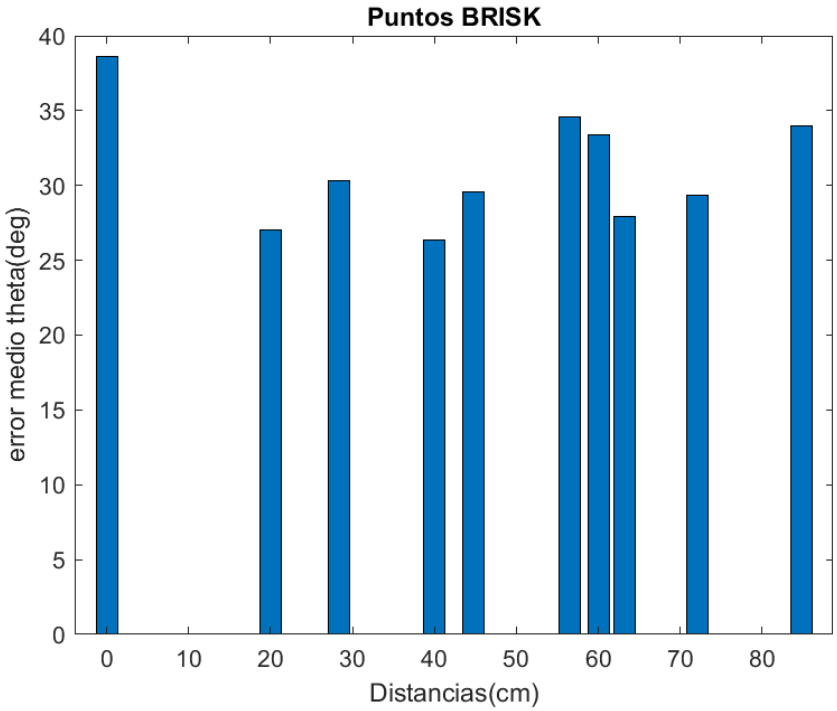


Figura 49: Error medio de θ obtenido mediante puntos BRISK en función de la distancia recorrida

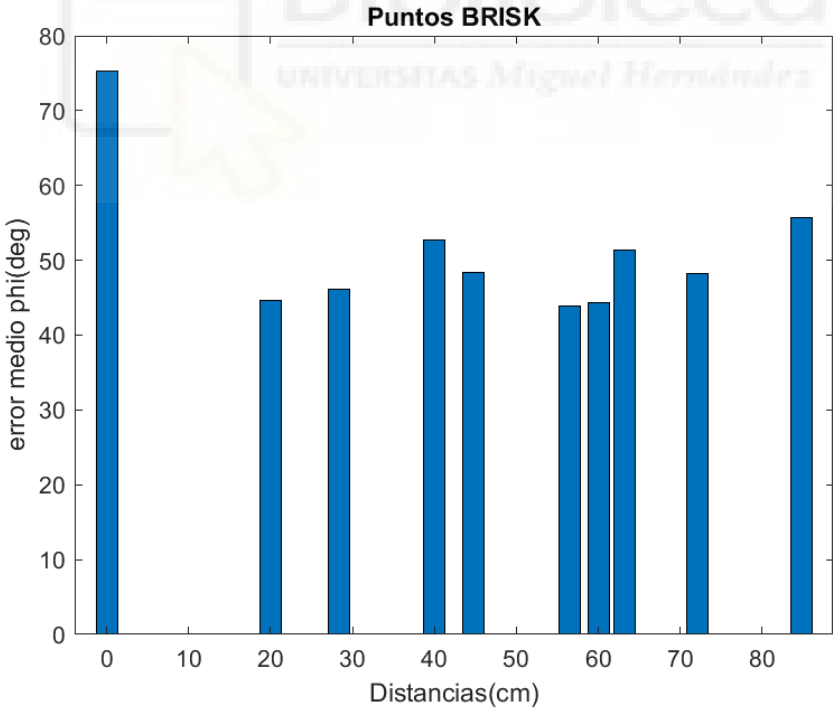


Figura 50: Error medio de ϕ obtenido mediante puntos BRISK en función de la distancia recorrida.

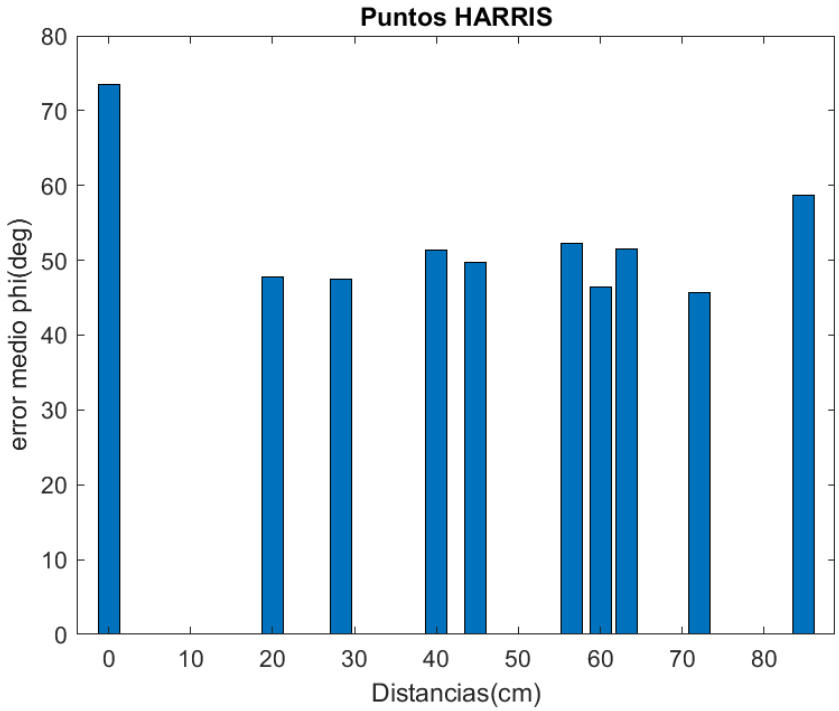


Figura 51: Error medio de ϕ obtenido mediante puntos HARRIS en función de la distancia recorrida.

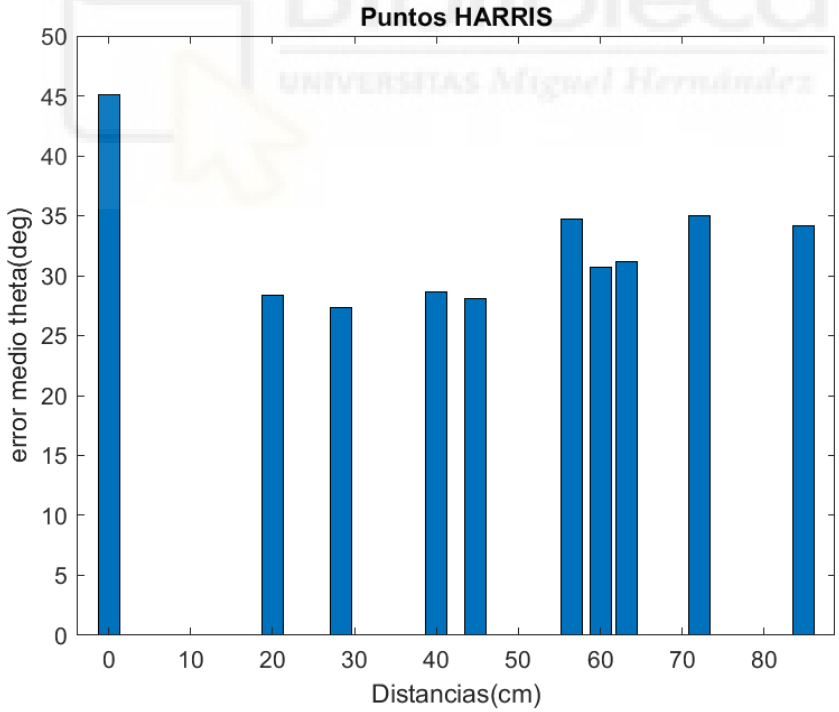


Figura 52: Error medio de θ obtenido mediante puntos HARRIS en función de la distancia recorrida

Al observar el error medio en función de la distancia recorrida por la cámara para la toma de la segunda imagen, se espera que cuanto mayor sea la distancia el error medio aumente. Sin embargo, esto no ocurre en las gráficas obtenidas. Se puede deber a que

para la toma de las imágenes no se han realizado distancias largas, de hecho, si se comparan las imágenes, la información capturada del entorno en cada uno no es muy distinta. Además, se puede ver como no hay ninguna distancia para la cual el error sea elevado con respecto al resto, excepto para el caso en el que no ha habido translación. En cuyo caso este error tan elevado se deba a que cuando la translación es nula existe rotación sobre el eje Y, y dicha rotación produce un mayor error a la hora de estimar los ángulos.

Por otro lado, se han obtenido las gráficas para cada descriptor en las que se muestra el error medio en función del número de correspondencias obtenidos.

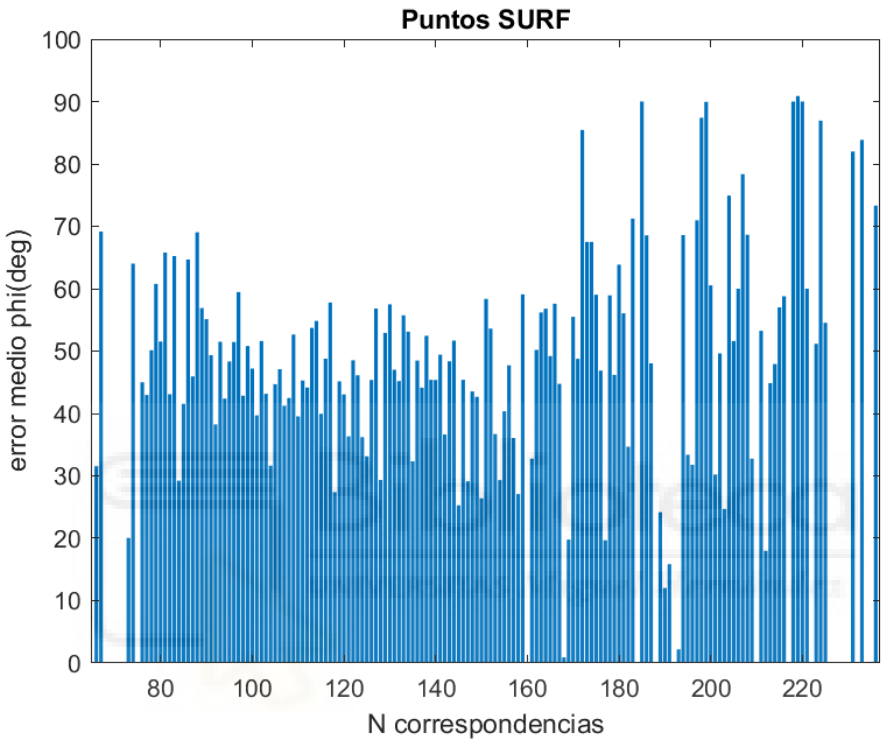


Figura 53: Error medio de ϕ obtenido mediante puntos SURF en función del número de correspondencias.

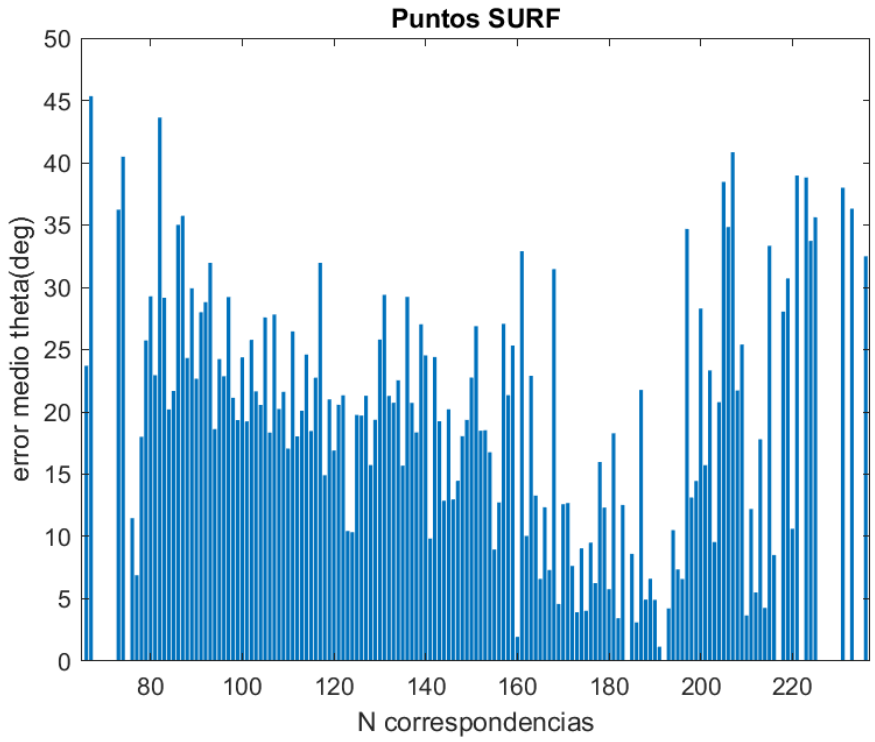


Figura 54: Error medio de θ obtenido mediante puntos SURF en función del número de correspondencias.

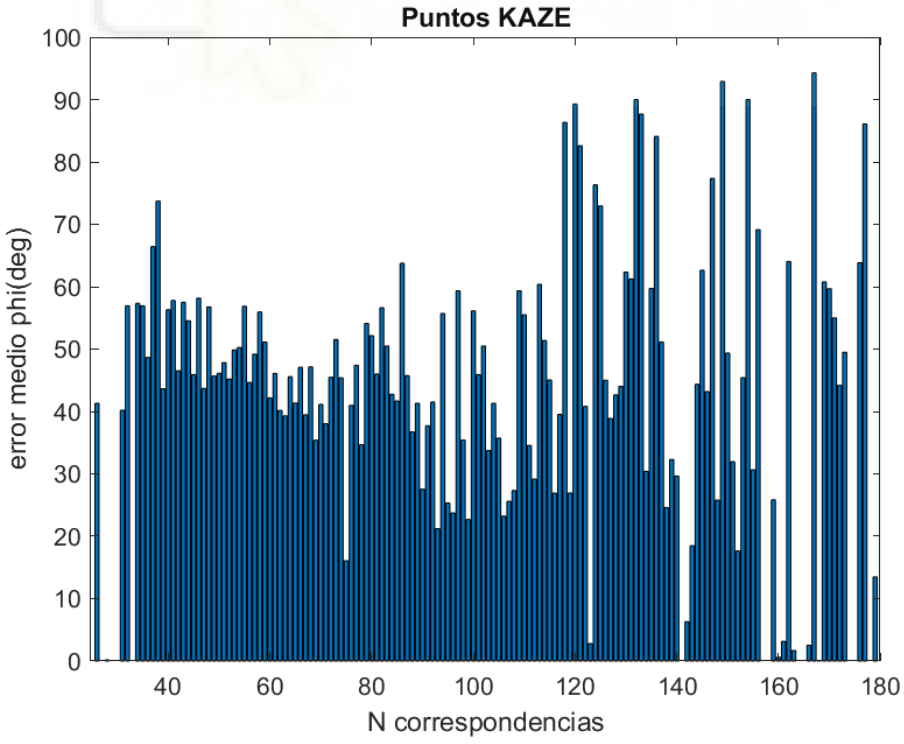


Figura 55: Error medio de ϕ obtenido mediante puntos KAZE en función del número de correspondencias.

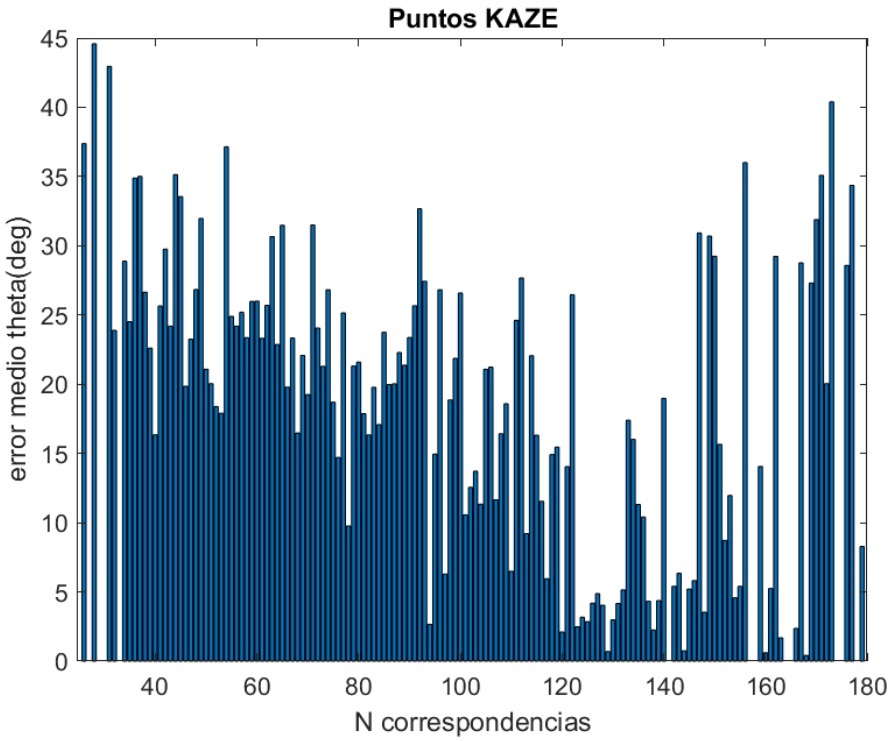


Figura 56: Error medio de θ obtenido mediante puntos KAZE en función del número de correspondencias.

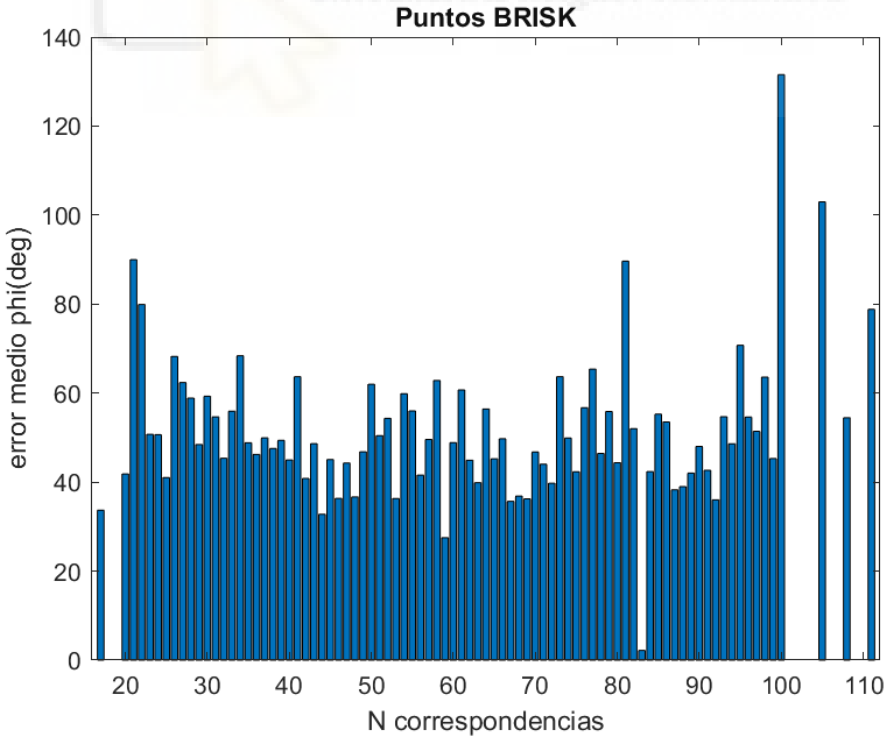


Figura 57: Error medio de ϕ obtenido mediante puntos BRISK en función del número de correspondencias.

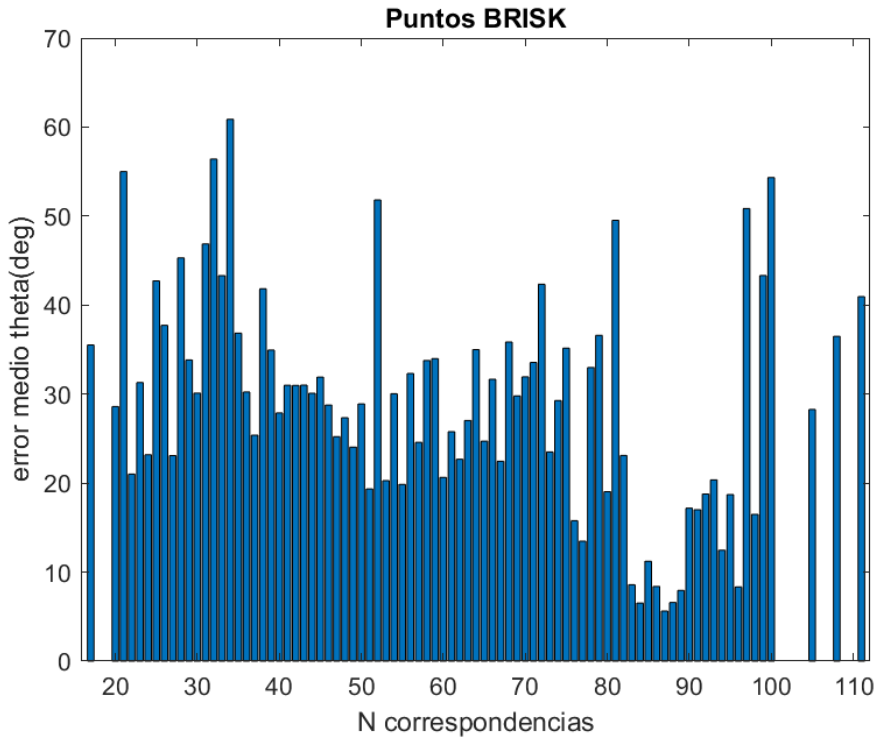


Figura 58: Error medio de θ obtenido mediante puntos BRISK en función del número de correspondencias.

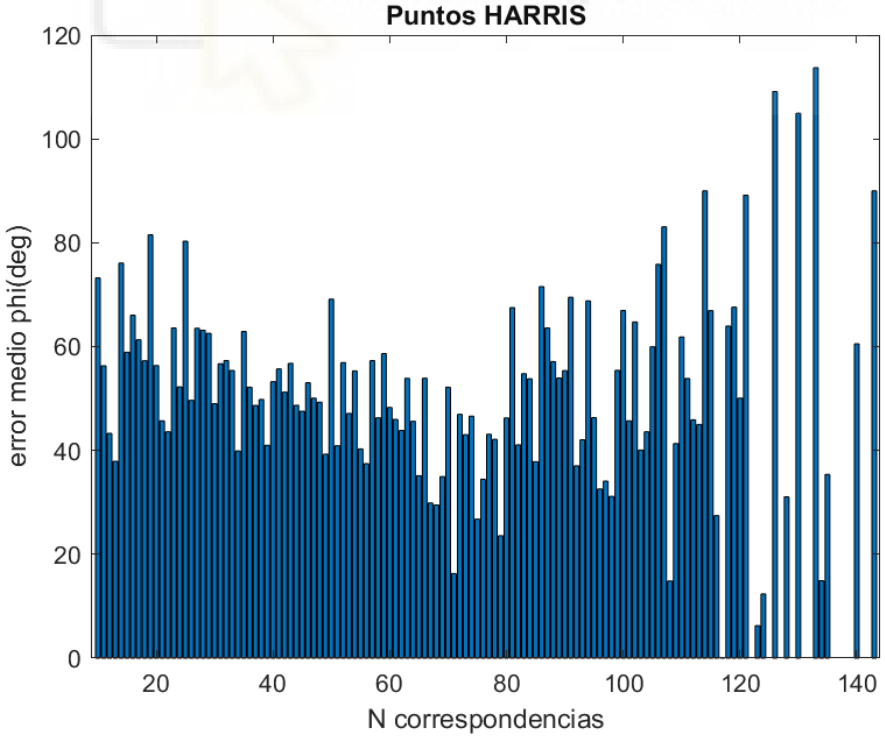


Figura 59: Error medio de ϕ obtenido mediante puntos HARRIS en función del número de correspondencias.

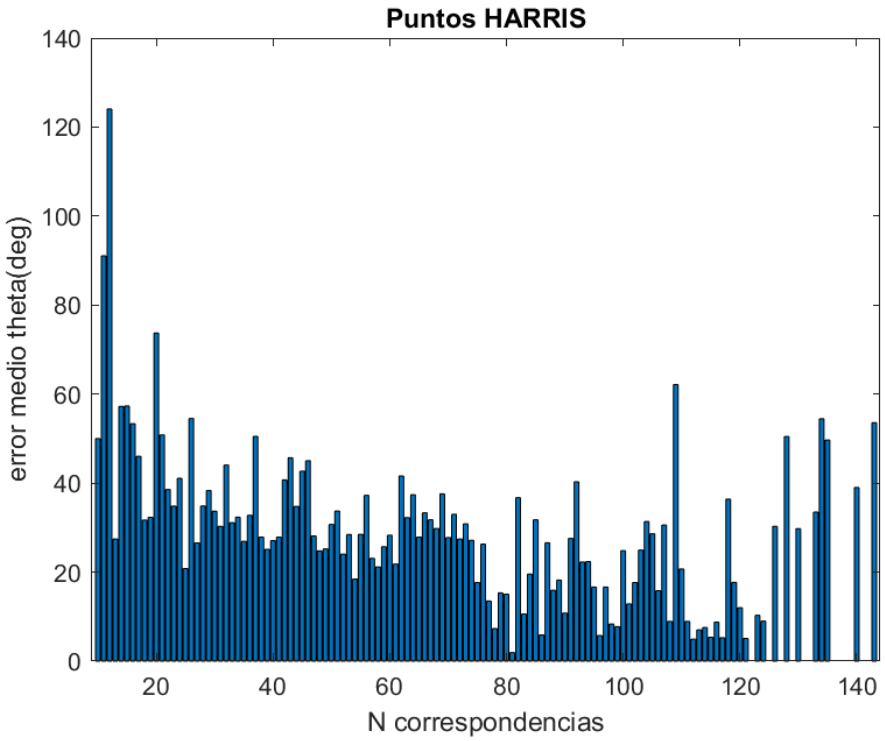


Figura 60: Error medio de θ obtenido mediante puntos HARRIS en función del número de correspondencias.

El comportamiento esperado es que para un mayor número de correspondencias el error sea menor. En las gráficas mostradas no vemos este comportamiento a simple vista, pero si se puede ver que, si se observa por zonas, en aquella donde el número de correspondencia es mayor presenta valores altos, pero también los valores de errores más bajos.

Capítulo 8
Conclusiones y líneas futuras

Este trabajo consiste en estimar la pose relativa entre dos imágenes tomadas con una cámara de 360 grados, concretamente con la Garmin VIRB 360. El primer objetivo ha sido obtener los parámetros de la cámara. Para este fin se han utilizado algunas aplicaciones que se encuentran disponibles para Matlab, como son la Toolbox de Davide Scaramuzza, la app cameraCalibrator y la Toolbox de Christopher Mei. De los resultados obtenidos, se ha podido ver que el menor error de reproyección, al calibrar, ha sido el obtenido mediante la aplicación implementada en Matlab.

El siguiente objetivo de este trabajo es obtener la pose relativa entre dos imágenes. En la etapa del algoritmo correspondiente a detectar los puntos característicos de las imágenes, se han utilizado SURF, KAZE, BRISK y HARRIS. Se ha podido ver que KAZE requiere de un elevado tiempo computacional, de hecho, la diferencia con el resto es considerada, por tanto, en el caso de llevar a cabo el trabajo en tiempo real este tipo de detector no sería beneficioso. Al observar las gráficas obtenidas se ha podido llevar a la conclusión de que los errores medios al estimar los ángulos son elevados, esto se debe a que la cámara es no lineal. Esto ocasiona algunos inconvenientes, por ejemplo, al calibrar con la aplicación cameraCalibrator, en el primer paso de introducir las imágenes algunas han sido descartadas de forma automática por no poder detectar correctamente las esquinas. Lo mismo ha ocurrido con las otras aplicaciones, cuando se ha tenido que clicar sobre las esquinas del patrón de calibración de forma manual, en algunas imágenes, no era capaz de seleccionar esa esquina y se cometía mucho error por lo que se decidieron eliminar dichas imágenes. Esto podría haber ocurrido igualmente en esta etapa y es por ello por lo que la estimación de la pose presente un error un poco elevado. Además, incluso eliminando falsas correspondencias mediante RANSAC no se ha conseguido disminuir mucho el error.

En cuanto a posibles líneas futuras, existen varias opciones que se podrían realizar para completar este trabajo, algunas de ellas no se han llevado a cabo por problemas de tiempo o incluso algunas se han intentado, pero finalmente no se han podido ejecutar debido a algunos problemas que han surgido.

En primer lugar, se podría obtener la imagen panorámica con la imagen delantera y la trasera que proporciona la cámara. Esto se intentó realizar, pero no se pudo debido a que el centro de ambas cámaras no coincide, existe un determinado desfase, y las imágenes resultantes se encuentran a distinta escala. Por ello, se deja como posible mejora solucionar este problema para poder realizar el *stitching* y obtener por tanto una imagen completa del entorno, que es el objetivo de esta cámara. Se tendría que realizar además la calibración de la cámara trasera siguiendo los pasos realizados en este trabajo.

Por otro lado, en este trabajo se ha estudiado la parte de odometría visual utilizando los parámetros obtenidos mediante la aplicación de Matlab, ya que como se ha comentado, ha sido con la que se ha obtenido un menor error de reproyección. Sin embargo, en algunas ocasiones, y debido a que la diferencia con respecto a la de Mei por ejemplo no es muy elevada, esto no implica que al utilizar este modelo se vaya a obtener una mejor estimación de la pose. Por tanto, se podría completar este trabajo realizando el apartado 6.2 con los parámetros del cámara obtenidos mediante las otras aplicaciones. De hecho, se ha intento realizar usando los parámetros que se han obtenido al calibrar con la Toolbox de Mei, pero debido a que la función que proporciona para la proyección de los puntos de la imagen en la esfera unidad se encuentra implementada en c++ finalmente no se ha podido realizar debido a que el ordenador no se encontraba preparado para ello y se producía error en el código. También se podría probar con otros tipos de detectores, por ejemplo, SIFT o ORB, para ver si con ellos se obtendrían un error menor.

Dado que este trabajo ha consistido en estudiar los resultados de localización y orientación entre las distintas poses de la cámara para la captura de imágenes, con el propósito de completar este trabajo sería interesante montar la cámara sobre un robot móvil e intentar estimar su trayectoria. A igual que ver su comportamiento en ambientes exteriores.

Bibliografía

[1] Alcantarilla, P.F., Bartoli, A., Davison, A.J., "KAZE Features." ECCV 2012, Part VI, LNCS 7577. 2012, p. 214

[2] Altun, A. A., Taghiyev, A., "Advanced image processing techniques and applications for biological objects," 2nd IEEE International Conference on Computational Intelligence and Applications (ICCIA), Beijing, 2017, pp. 340-344.

[3] Alves, S. F. R., Rosario, J.M, Ferasoli Filho, H., Rincon, L. K. A., Yamasaki, R. A. T., "Conceptual bases of robot navigation modeling, control and applications", Advances in Robot Navigation, , InTech, 2011, pp 6-8.

[4] Amorós, F. J., "Aplicación de la Apariencia Global de la Información Visual Omnidireccional en Color a Tareas de Navegación Robótica en Espacio 2 ½ D", Tesis doctoral.

[5] Aqel, M. O., Marhaban, M. H., Saripan, M. I., Ismail, N. B., "Review of visual odometry: types, approaches, challenges, and applications," SpringerPlus, vol. 5, no. 1, p. 1897, 2016.

[6] Ballesta, M., Gil, A., Reinoso, O., "Evaluation of interest point dtectors for Visual SLAM." Internal Journal of Factoy Automation, Robotics and Soft Computing, vol., 2007, pp. 86-95.

[7] Barreto, J. P., Araujo, H., "Issues on the geometry of central catadioptric image formation.", en CVPR, volume 2, 2001, pp. 422–427.

[8] Bay, H., Ess, A., Tuytelaars, T., Van Gool, L., "SURF:Speeded Up Robust Features." Computer Vision and Image Understanding (CVIU). Vol. 110, No. 3, 2008, pp. 346–359.

[9] Berenguer, Y., Payá, L., Ballesta, M., Jiménez, L. M., Cebollada, S., Reinoso, O., "Algoritmo de SLAM utilizando apariencia global de imágenes omnidireccionales." Actas de las XXXVIII Jornadas de Automática, 2017.

[10] Brown, M., Lowe, D., "Invariant features from interest point groups." In: British Machine Vision Conf., BMVC, Cardiff, UK, 2002.

[11] Chaumette, F., Hutchinson, S., "Visual servo control. I. Basic approaches," en IEEE Robotics & Automation Magazine, vol. 13, no. 4, 2006, pp. 82-90.

[12] Cheng, Y., Maimone, M., Matthies, L., "Visual odometry on the mars exploration rovers.", In: 2005 IEEE International Conference on Systems, Man and Cybernetics, vol. 1, 2005, pp. 903–910.

[13] Davison, A., "Real-time simultaneous localisation and mapping with a single camera.", In: Ninth IEEE International Conference on Computer Vision, 2003, Proceedings, 2003, pp. 1403–1410.

[14] Delibasis, K., Plagianakos, V., Maglogiannis, I., "Real Time Indoor Robot Localization Using a Stationary Fisheye Camera.", en AIAI, 2013, pp.245-254.

[15] Dudek, G., Jenkin, M., “Computational Principles of Mobile Robotics”, New York, Cambridge University Press, 2010.

[16] Ferrier, J. L., Bernard, A., Gusikhin, O., Madani, K., “Informatics in Control, Automation and Robotics”, Springer, 2009.

[17] Fischler, M.A., Bolles R.C., “Random sample consensus: A paradigm for modelfitting with applications to image analysis and automated cartography.“,Communications of the ACM, 24(6):381–395, 1981.

[18] Fraundorfer, F., Scaramuzza, D., “Visual Odometry Part II: Matching, Robustness, Optimization, and Applications”, IEEE ROBOTICS & AUTOMATION MAGAZINE, 2012.

[19] Geyer, C, Daniilidis, K. “A unifying theory for central panoramic systems and practical implications. En ECCV, 2000, pp. 445–461.

[20] Geyer, C., Daniilidis, K.,” A unifying theory for central panoramic systems and practical implications.”, In: Computer VisionECCV 2000. Springer, pp. 445–461.

[21] Gil, A., Mozos, O. M., Ballesta, M., Reinoso, O., “A comparative evaluation of interest point detectors and local descriptors for visual slam.” Machine Vision and Applications, 21(6), 2010, pp 905–920.

[22] Gil, A., Reinoso, O., Ballesta, M., Juliá, M., Payá, L., “Estimation of Visual Maps with a Robot Network Equipped with Vision Sensors.” Sensors, 2010.

[23] Harris, C., and M. Stephens, "A Combined Corner and Edge Detector," Proceedings of the 4th Alvey Vision Conference, August 1988, pp. 147-151.

[24] Hart, C., Kreinar, E., Chrzanowski, D., Daltorio, K. A., Quinn, R. D., “A low-cost robot using omni-directional vision enables insect-like behaviors,” en IEEE International Conference on Roboticsand Automation (ICRA) (Seattle, WA: IEEE), 2015, pp. 5871–5878.

[25] Jiang, N., Lv, J., Kobayashi, Y., Emaru, T., "Adaptive image-based visual servoing of nonholonomic mobile robot with on-board camera," IEEE/SICE International Symposium on System Integration (SII), Nagoya, 2015, pp. 983-988.

[26] Kannala, J., Brandt, S.S., “A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses.”, IEEE Trans. Pattern Anal. Machine Intell. 28 (8), 2006, pp 1335–1340.

[27] Kawanishi, R., Yamashita, A., Kaneko, T. “Construction of 3d envionment model from an omnidirectional image sequence,” en Proceedings ofthe 3rd Asia International Symposium on Mechatronics, TP1-3 (2) (Sapporo), 2008, pp. 1–6.

[28] Kim, S.-H., Park, J.-H., Jung, I.-K., “Global localization of mobile robot using an omni-directional camera” en IPCV, 2014.

[29] Leutenegger, S., Chli, M. and Siegwart, R. Y., "BRISK: Binary Robust invariant scalable keypoints," 2011 International Conference on Computer Vision, Barcelona, 2011, pp. 2548-2555.

[30] Liu, M., Pradalier, C., Siegwart, R., “Visual homing from scale with an uncalibrated omnidirectional camera.” IEEE Trans. Robot. 29, 2013, pp. 1353–1365.

[31] Lowe, D., “Distinctive image features from scale-invariant keypoints,” International Journal of Computer Vision, 2004.

[32] Lukierski, R., Leutenegger, S., Davison, A. J., “Rapid free-space mapping from a single omnidirectional camera,” en ECMR, 2005, pp. 1–8.

[33] Mac, T. T., Copot, C., Tran, D. T., “Heuristic approaches in robot path planning: A survey”, Robotics and Autonomous Systems,Volume 86, 2016, pp. 13-28.

[34] Macesau, G., Moldoveanu, F., “Computer Vision Based Mobile Robot Navigation in Unknown Environments,” Bulletin of the Transilvania University of Brasov, Series I: Engineering Sciences, vol. 3, no. 52, 2010

[35] Malis, E., Chaumette, F., Boudet, S., "2½D visual servoing," en IEEE Transactions on Robotics and Automation, vol. 15, no. 2, 1999, pp. 238-250.

[36] Marković, I., Chaumette, F., Petrović, I., “Moving object detection, tracking and following using an omnidirectional camera on a mobile robot,” en IEEE International Conference on Robotics and Automation (ICRA) (Hong Kong: IEEE), 2014, pp. 5630–5635.

[37] Mei, C., Rives, P., “Single View Point Omnidirectional Camera Calibration from Planar Grids.”, IEEE International Conference on Robotics and Automation (ICRA), Rome, Italia, 2007, pp. 3945-3950.

[38] Micušık, B., “Two-view geometry of omnidirectional cameras.” Ph.D. thesis, Czech Technical University, 2004.

[39] Micusik, B., Pajdla, T., “Structure from motion with wide circular field of view cameras.” IEEE Trans. Pattern Anal. Mach. Intell. 28, 2006, pp. 1135–1149.

[40] Moravec H., “Obstacle avoidance and navigation in the real world by a seeing robot rover.”, Stanford: Stanford Univ.; 1980

[41] Moravec, H.P., “Towards Automatic Visual Obstacle Avoidance.” In: Proceedings of the 5th International Joint Conference on Artificial Intelligence, Cambridge, UK, 1977, p. 584-587.

[42] Neira, J., Davison, A. J., Leonard, J. J., "Guest Editorial Special Issue on Visual SLAM." in IEEE Transactions on Robotics, vol. 24, no. 5, 2008, pp. 929-931.

[43] Neumann, J., Fermuller, C., Aloimonos, Y., "Eyes from eyes: new cameras for structure from motion," Omnidirectional, 2002. Vision Proceedings. Third Workshop on, 2002, pp. 19-26.

[44] Nilsson, N.J., “Shakey the Robot.” In: Technical Report Nr. 323, SRI International, Menlo Park, USA, 1984.

[45] Nistér, D., Naroditsky, O., Bergen, J., “Visual odometry.” In: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2004), vol. 1, 2004, pp. I–652.

[46] Payá, L., Gil, A., Reinoso, O. “A State of art Revie on Mapping and Localization of mobile Robots Using Omnidirectional Vision Sensors” Journal of Sensors Volume 2017.

[47] Payá, L., Reinoso, O., Jiménez, L.M., Juliá, M., "Estimating the position and orientation of a mobile robot with respect to a trajectory using omnidirectional imaging and global appearance" en PLOS ONE, 2017.

[48] Payá, L.; Amorós, F.; Fernández, L.; Reinoso, O., “Performance of Global-Appearance Descriptors in Map Building and Localization Using Omnidirectional Vision.”, Sensors 2014, 14, 3033–3064.

[49] Poddar, S., Kottath, R., Karar, V., “Evolution of Visual Odometry Techniques.”, 2018

[50] Puig, L., Bastanlar, Y., Sturm, P., Guerrero, J.J., Barreto, J., “Calibration of central catadioptric cameras using a dlt-like approach.” Int. J. Comput. Vision 93 (1), 101–114, 2011.

[51] Puig, L., Guerrero, J. J., “Omnidirectional Vision Systems: Calibration, Feature Extraction and 3D”, Springer, 2013.

[52] Scaramuzza, D., Fraundorfer, F., “Visual Odometry Part I: The First 30 Years and Fundamentals”, IEEE ROBOTICS & AUTOMATION MAGAZINE, 2011.

[53] Scaramuzza, D., Martinelli, A., Siegwart, R., "A flexible technique for accurate omnidirectional camera calibration and structure from motion.” In: IEEE International Conference on Computer Vision Systems, 2006 ICVS’06. IEEE, 45-45.

[54] Scaramuzza, D., Martinelli, A., Siegwart, R., “A toolbox for easily calibrating omnidirectional cameras.”, IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE,2006, pp. 5695–5701.

[55] Scaramuzza, D., Siegwart, R, “Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles.”, IEEE Trans. Robot. **24**(5), 2008, pp 1015–1026.

[56] Schneider, D., Schwalbe, E., Maas, H.-G., “Validation of geometric models for fisheye lenses.”, ISPRS J. Photogr. Remote Sensing 64 (3), 2009, pp 259–266.

[57] Tardif, J., Pavlidis, Y. and K. Daniilidis, "Monocular visual odometry in urban environments using an omnidirectional camera.", 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice, 2008, pp. 2531-2538.

[58] Urban, S., Leitloff, J., Hinz, S., “Improved wide-angle, fisheye and omnidirectional camera calibration.”, ISPRS J. Photogramm. Remote Sens.2015, 108, pp 72–79.

[59] Valiente, D., Fernández, L., Gil, A., Payá, L., Reinoso, O., “Visual Odometry through Appearance- and Feature-Based Method with Omnidirectional Images,” Journal of Robotics, vol. 2012, Article ID 797063, 2012.

[60] Valiente, D., “View-based slam in the 2-1/2d space with omnidirectional images and feature point information.” 2016. Tesis Doctoral. Universidad Miguel Hernández de Elche.

[61] Weickert, J., Scharr, H., “A scheme for coherence-enhancing diffusion filtering with optimized rotation invariance.”, Journal of Visual Communication and Image Representation 13, 2002, pp. 103–118.

[62] Yahya, M. F., Arshad, M. R., "Position-based visual servoing for underwater docking of an autonomous underwater vehicle," 2016 IEEE International Conference on Underwater System Technology: Theory and Applications (USYS), Penang, 2016, pp. 121-126.

[63] Yousif, K.; Bab-Hadiashar, A.; Hoseinnezhad, R. “An overview to visual odometry and visual SLAM”, Applications to mobile robotics. Intell. Ind. Syst.2015, 1, 289–311.

[64] Zhang, C., Xu, J., Xi, N., Jia, Y., Li, W., “Development of an omnidirectional 3d camera for robot navigation,” en IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), 2012, pp. 262–267.

[65] Zhang, Z., “Flexible camera calibration by viewing a plane from unknown orientations.”, The Proceedings of the Seventh IEEE International Conference on Computer Vision, 1999, vol. 1. IEEE, pp. 666–673.