UNIVERSIDAD
REY JUAN CARLOS

MÁSTER OFICIAL EN VISIÓN ARTIFICIAL

Curso Académico 2019/2020

Trabajo Fin de Máster

Algoritmo de autolocalización basado en visión y sensor inercial: VIO-SDSLAM

Autor:

Javier Martínez del Río

Tutores:

José María Cañas Plaza

Diego Martín Martín

Resumen

Una de las principales áreas de investigación del campo de la Visión Artificial y la Robótica es la localización y mapeado simultáneos (SLAM por sus siglas en inglés) que engloba al conjunto de técnicas y algoritmos de autolocalización. Esta técnica es capaz de estimar la pose de la cámara (posición y orientación) al mismo tiempo que construye un mapa del entorno que la rodea. Los sistemas monoculares (una sola cámara) de Visual SLAM son de especial interés ya que permiten implementar el algoritmo en dispositivos económicos y ligeros, como por ejemplo teléfonos inteligentes y vehículos aéreos no tripulados.

Lamentablemente, esta configuración presenta una serie de problemas inherentes a los propios sensores de adquisición monoculares, entre los cuales se encuentran la ambigüedad en la escala de representación del mundo (debido a la pérdida de información 3D al momento de ser proyectada sobre el plano imagen) y la incapacidad del algoritmo de recuperarse ante una pérdida de la calidad visual a no ser que revisite una zona conocida.

En este Trabajo de Fin de Máster se explora cómo complementar a SD-SLAM (un algoritmo de Visual SLAM) con una unidad de medición inercial (IMU) capaz de suministrar información del mundo real, con el objetivo de solventar los problemas anteriormente mencionados. Para ello se ha diseñado y programado un nuevo algoritmo denominado VIO-SDSLAM. Para validar el correcto funcionamiento del diseño del algoritmo propuesto se han realizado numerosos experimentos empleando conjuntos de datos internacionales. Las distintas pruebas muestran cómo VIO-SDSLAM consigue dar una buena solución a estos problemas y al mismo tiempo aumenta la robustez, dotándolo de nuevas características ausentes en su algoritmo predecesor.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En este capítulo se introducirá de forma general el marco en el que se encuadra el presente Trabajo de Fin de Máster (TFM). Concretamente, se explicará brevemente qué es la Visión Artificial y, en particular, el campo de la autolocalización visual en entornos desconocidos mediante el uso de algoritmos de SLAM. Por otra parte, se expondrá qué es la robótica por ser el principal campo de aplicación del algoritmo de Visual SLAM de este trabajo.

1.1. Visión Artificial

La Visión Artificial es la rama de la Inteligencia Artificial orientada a la captura y procesamiento de imágenes. En primer lugar, la captura es posible mediante el desarrollo de sensores capaces de recibir, procesar y almacenar parte del espectro electromagnético, como la luz visible o el infrarrojo, obteniendo imágenes con distintos tipos de información. En segundo lugar, el procesamiento engloba toda la parte del desarrollo de algoritmos capaces de interpretar el contenido de dichas imágenes.

Las imágenes contienen una gran cantidad de información útil para numerosos problemas, sin embargo, extraer y procesar dicha información no es un proceso sencillo. De hecho, la propia representación de las imágenes, que suele darse en forma de matriz numérica, ya es confusa. Como se puede apreciar en la Figura 1.1, nuestros sentidos no interpretan de igual modo una imagen (izquierda) que su representación (derecha).

Los orígenes de este campo se remontan a la década de los 60, cuando se conectó por primera vez una cámara a un computador con el fin de obtener y analizar la información. Larry Roberts fue una de las primeras personas en desarrollar un experimento en el cual no solo se capturaban imágenes, sino que se analizaba su contenido (una estructura de bloques) para posteriormente reproducirlo desde otra perspectiva.

En sus inicios, las técnicas de visión artificial estuvieron limitadas por la capacidad de cómputo de los ordenadores de la época. Sin embargo, en los últimos años, con el avance de

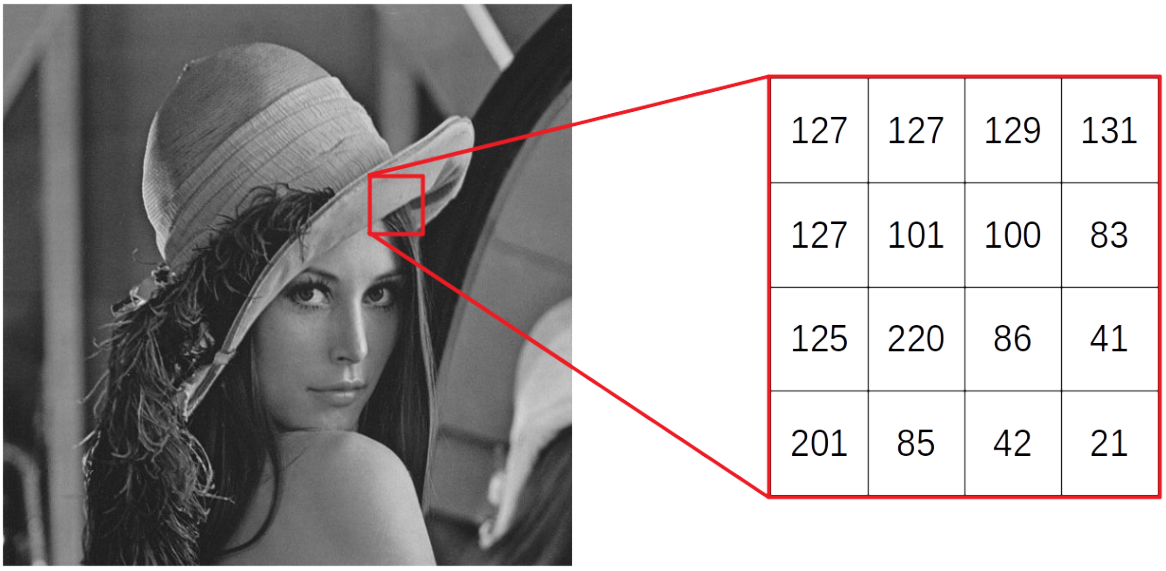


Figura 1.1: Comparación entre la información visual y su representación numérica.

la tecnología y la reducción del coste económico del *hardware*, se ha producido un gran avance en este campo. Esto ha permitido la aparición de algoritmos capaces de trabajar en tiempo real (entendiéndolo como el procesamiento de al menos 30 fotogramas por segundo) y la visión artificial está presente en campos como la vigilancia, la robótica, la medicina e incluso los videojuegos. Algunas de las aplicaciones clásicas de la visión artificial son la siguientes:

- **Detección de objetos.** Es posible identificar si un objeto se encuentra en una imagen realizando búsquedas por rasgos característicos del mismo, como la forma, el color, textura o patrones presentes en él.
- **Seguimiento de objetos.** Como extensión de la tarea anterior, es interesante poder realizar el seguimiento a un mismo objeto, de forma inequívoca, a lo largo del tiempo en una secuencia de imágenes.
- **Reconocimiento de caracteres.** Esta técnica, conocida como OCR por sus siglas en inglés (*Optical Character Recognition*), permite la detección e identificación de los caracteres en un documento, ya sean digitales o manuscritos, con la finalidad de digitalizar el contenido de los mismos.

El rendimiento y precisión de estas tareas ha aumentado en los últimos años debido al uso de las redes neuronales, más concretamente al uso de técnicas de *Deep Learning*. La mejora ha sido tal que algunos sistemas de reconocimiento de patrones visuales obtienen tasas de error inferiores a las humanas. El primer sistema «sobrehumano» fue obtenido en 2011 en la competición de reconocimiento de señales de tráfico IJCNN, donde el sistema vencedor obtuvo una tasa de error dos veces mejor que la obtenida por sujetos de prueba humanos [33].

Sin embargo, hay campos donde las técnicas de *Deep Learning* aún no han destacado especialmente respecto a técnicas más clásicas de Visión Artificial, como por ejemplo la autolocalización. Este problema consiste en la estimación de la ubicación de la cámara en el entorno tridimensional (conocido o desconocido) que la rodea. Esta técnica es esencial para campos como la robótica, donde es imprescindible para un robot o un dron localizarse en el entorno para evitar choques y/o comportamientos anómalos; o para el reciente campo de la realidad aumentada.

1.2. Visual SLAM

Se conoce como localización y mapeado simultáneo o SLAM por sus siglas en inglés (*Simultaneous Localization And Mapping*) al conjunto de técnicas y algoritmos de autolocalización. SLAM es el término empleado para describir el proceso por el cual, a partir de la información obtenida de sensores, es posible generar un mapa del entorno que lo rodea, al mismo tiempo que se localiza en él. Tiene sus orígenes en el campo de la robótica donde cobra especial importancia en robots autónomos que operan en entornos desconocidos.

Cuando el sensor principal de SLAM son una o más cámaras nos referimos al modelo como Visual SLAM. Pese a ser una cuestión activa que aún está en desarrollo, dado que no hay una solución exacta para el problema que trata de abordar, se emplean técnicas de Visual SLAM en numerosas aplicaciones. Algunas de ellas son las siguientes:

- **Robot aspirador:** estos dispositivos autónomos son equipados con cámaras (u otros sensores similares como los LIDAR) siendo capaces de generar un mapa de los hogares navegando por ellos. La ventaja en este caso de conocer el entorno y su ubicación es la capacidad de generar y optimizar las rutas de limpieza, al mismo tiempo que evitan obstáculos en su trayectoria. En la Figura 1.2 puede observarse el mapa construído por el ronot aspirado Roomba 980.

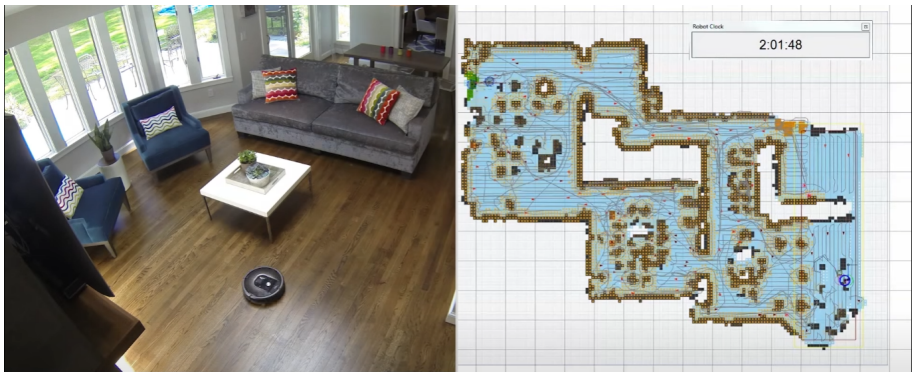


Figura 1.2: Ejemplo de la Roomba 980 recorriendo el entorno real (imagen izquierda) y construyendo el mapa del entorno (imagen derecha).

- **UAV:** los vehículos aéreos no tripulados (UAV por sus siglas en inglés) como los drones (Figura 1.3) pueden emplear SLAM para visualizar el entorno que los rodea y tomar decisiones en tiempo real.

Figura 1.3: Dron con cámara incorporada.

- **Realidad aumentada:** SLAM no solo tiene cabida en el mundo de la robótica, en el campo de la realidad aumentada también es posible hacer uso de estos algoritmos para relacionar de un mejor modo el mundo real con el virtual. Estas aplicaciones emplean el mapa obtenido para introducir los elementos visuales de forma más realista. Esto solo es posible si se conoce la posición del sensor y el entorno. Un ejemplo de realidad aumentada puede observarse en la Figura 1.4, donde se incluye mobiliario en una habitación de forma más inmersiva.

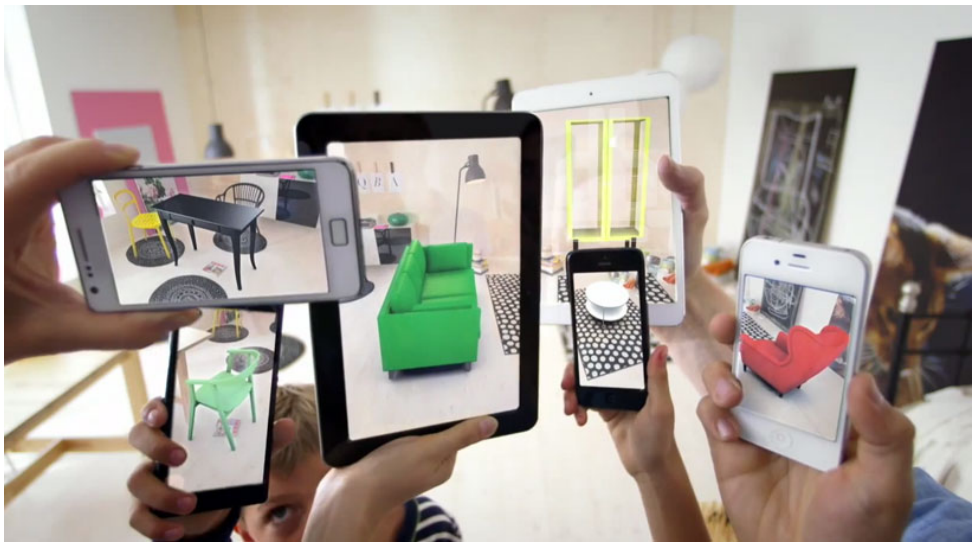


Figura 1.4: Ejemplo de Realidad Aumentada.

Visual SLAM tiene sus raíces en la línea de investigación conocida como *Structure from motion*. Dicha línea está centrada en la reconstrucción automática de estructuras 3D a partir de un conjunto de imágenes (similar al proceso de mapeo realizado en Visual SLAM). Este proceso se basa en la premisa de que, dadas múltiples vistas de un mismo punto tridimensional a lo largo de distintas imágenes, es posible estimar su posición en el espacio mediante el uso de triangulación. En la Figura 1.5 se puede observar un ejemplo de reconstrucción 3D a partir de tres imágenes.

Structure from motion supuso un gran avance en las técnicas de detección de puntos característicos, también llamados puntos de interés, que serían de gran utilidad posteriormente en Visual SLAM. Estos puntos son llamados característicos porque son lo suficientemente únicos (por sus propiedades o entorno que los rodean) para ser detectados de nuevo ante cambios de iluminación o distintos ángulos de vista. Los puntos característicos son empleados por la gran mayoría de algoritmos de Visual SLAM, como se verá en el próximo capítulo. Los extractores de puntos característicos más conocidos son *SIFT* [22], *SURF* [1], *FAST* [29] y *ORB* [32].

Esta técnica serviría como idea básica del funcionamiento de Visual SLAM. De forma general, la estructura a reconstruir es el entorno por donde navega la cámara. Por otra parte, el conjunto de las imágenes es generado a lo largo del tiempo, y de cada una de ellas se extraen puntos de interés para estimar su posición tridimensional, dando lugar a un mapa de puntos. Por último, se hace uso de ese mapa para estimar las diferentes posiciones de la cámara.

Esta no es la única aproximación al problema, ya que debido al avance de la tecnología, es posible emplear no sólo sistemas formados por una única cámara (modelos que denominaremos *Monoculares*), sino utilizar sistemas más complejos. Por ejemplo se puede emplear un par estéreo, formado por dos cámaras separadas entre sí una distancia conocida, aprovechando esta configuración para estimar la información de profundidad de toda la escena; o directamente emplear una cámara *RGBD* que ya aporta la imagen de profundidad, sin necesidad de calcularla. Todos estos modelos y sistemas tienen sus ventajas y desventajas como veremos más adelante.

Figura 1.5: Reconstrucción 3D mediante *Structure from Motion*.

1.2.1. Conceptos

La mayoría de algoritmos de Visual SLAM pueden clasificarse en tres categorías en función de cómo estimen la posición de la cámara:

- Métodos basados en características. Estas técnicas extraen puntos de interés de las imágenes y determinan el desplazamiento realizado por la cámara minimizando el error de retro-proyección de los puntos. Uno de los algoritmos de Visual SLAM más conocidos perteneciente a esta categoría es ORB-SLAM.
- Métodos directos. Emplean los valores de intensidad de los píxeles de la totalidad de la imagen para estimar el desplazamiento minimizando el error fotométrico. Un algoritmo que emplea métodos directos es LSD-SLAM [8].
- Métodos híbridos. Hacen uso de una combinación de los dos métodos anteriores. Un ejemplo de este tipo de algoritmos es SD-SLAM.

Indiferentemente de la clasificación a la cual pertenezca un algoritmo de Visual SLAM, todos ellos tienen asociados una serie de conceptos [13] que merece la pena detallar, dado que se hará uso de ellos a lo largo de este trabajo.

Calidad: la calidad del algoritmo dependerá de la eficiencia temporal, la precisión espacial de la pose (posición y orientación tridimensional) y la robustez.

Eficiencia temporal: medida como el tiempo de ejecución de cada iteración del algoritmo de Visual SLAM. Para considerar que el algoritmo es apto para trabajar en tiempo real deberá ser capaz de procesar al menos 30 fotogramas (*frames*) por segundo.

Precisión de la posición: diferencia entre la pose estimada y la pose real, expresada como el error lineal y angular.

Robustez: entendida como la capacidad de recuperarse o seguir funcionando ante situaciones inesperadas, como oclusiones, imágenes borrosas, objetos dinámicos en la escena, etc.

Oclusiones: situación donde la cámara del sistema esté tapada total o parcialmente, de modo que no sea posible utilizar la totalidad de la imagen para obtener información.

Relocalización: capacidad para recuperarse de una pérdida por falta de información, siendo capaz de volver a estimar la posición de forma correcta dentro del mapa.

Cierre de bucle: capacidad de detectar cuándo la cámara vuelve, tras pasar un periodo de tiempo, a una zona del mundo que ya haya visitado con anterioridad. La dificultad de esta tarea

radica en cambios en la escala, iluminación, cálculo de la pose de la cámara u otros motivos similares. Por este motivo, un algoritmo de Visual SLAM debe ser capaz de reconocer este entorno conocido y modificar su mapa para hacerlo coincidir cuando se reobservan lugares ya visitados.

1.3. Robótica

Uno de los principales campos de investigación donde la visión artificial tiene una fuerte presencia es la robótica. Los robots son sistemas electromecánicos diseñados y programados con el fin de realizar unas determinadas funciones de manera autónoma.

Para lograr este fin hacen uso de tres tipos de componentes *hardware*: actuadores, sensores y unidades de procesamiento. Los actuadores y sensores permiten interactuar y obtener información del mundo real, mientras la unidad de procesamiento actúa como el cerebro que analiza los datos proporcionados por los sensores y, en consecuencia, toma las decisiones que deben realizar en cada momento los actuadores.

Existe una gran variedad de robots desarrollados para entornos industriales, ya que tienen la capacidad de realizar el trabajo de una forma muy precisa, siendo un claro ejemplo los brazos robóticos (Figura 1.6a) utilizados en numerosas cadenas de montaje. También son empleados en otros campos como la educación, defensa, medicina, la ayuda en el hogar e incluso la exploración espacial.

Algunos ejemplos de estos tipos de robots pueden ser los robots aspiradora y UAVs mencionados anteriormente, o más recientemente el desarrollo de los coches autónomos. En la conducción autónoma, los vehículos son dotados de numerosos sensores entre los cuales se incluyen cámaras. A partir de las imágenes obtenidas es posible detectar los carriles de circulación, otros vehículos (Figura 1.6b) e incluso peatones.

1.3.1. Sensores empleados en los algoritmos de autolocalización

Uno de los sensores más utilizados en robótica son las cámaras, especialmente cuando se desea emplear técnicas de autolocalización, debido a la gran cantidad de información del entorno que pueden proporcionar. Principalmente son utilizadas por robots móviles autónomos, es decir, aquellos especializados en la navegación sobre un terreno, conocido o desconocido, sin control humano.

A parte de las cámaras, existen diferentes tipos de sensores que suelen emplearse con algoritmos de autolocalización. Algunos de ellos proporcionan información del entorno que rodea al dispositivo, como por ejemplo los sensores LIDAR 3D. Estos dispositivos permiten estimar la distancia a la que se encuentran los objetos que rodean al sensor empleando un haz láser.

(a) (b)

Figura 1.6: Conjunto de brazos robóticos operando en una cadena de montaje de la industria del automóvil (a). *Software* de conducción autónoma de Nvidia Drive PX (b).

Estas medidas de distancia pueden interpretarse como una nube de puntos tridimensionales. La nube de puntos es similar a la obtenida en el proceso de mapeo de Visual SLAM, con la ventaja de no necesitar tiempo de computación. Además, no requiere de buenas condiciones lumínicas para funcionar correctamente, a diferencia de las cámaras. Las principales desventajas son la ausencia de información de color así como el coste económico del sensor.

Otros sensores proporcionan información de su posición espacial o de las aceleraciones y rotaciones a las que se ven sometidos. Estos son especialmente útiles para obtener información de posición y orientación. Algunos de los más utilizados son las brújulas, inclinómetros, sistemas GPS y las unidades de medición inercial (IMU por sus siglas en inglés), destacando especialmente estos dos últimos. El GPS es capaz de proporcionar directamente información de posición empleando trilateración sobre una red de satélites orbitales. La principal desventaja es la imposibilidad de funcionar en interiores, debido a que el sensor actúa como un receptor de los satélites y la señal no es capaz de atravesar la estructura de los edificios. Esta carencia puede ser solventada con sistemas de balizamiento instalados en interiores, pero requiere de su despliegue cada vez que se quiere hacer uso de ellos.

Por otra parte, una IMU es un dispositivo electrónico formado por distintos sensores inerciales capaz de funcionar en interiores y exteriores. Típicamente consta de un acelerómetro y un giroscopio, aunque también pueden incluir un magnetómetro para ser más precisos en la orientación, aunque este último sólo funciona adecuadamente en exteriores. La IMU es capaz de proporcionar en tiempo real información de aceleración lineal y velocidad angular del sistema.

Las IMU pueden clasificarse en distintas categorías o grados [16] en función de la calidad y la precisión que proporcionan:

- Grado *Marine* y grado de navegación. Estos son los sensores de mayor grado disponibles comercialmente, siendo lo suficientemente precisas como para ser utilizadas para nave-

gación a largas distancias. Suele ser uno de los componentes principales de los sistemas de navegación inercial empleados en aparatos como barcos, aviones, submarinos o naves espaciales, entre otros. Estos dispositivos se caracterizan por ser grandes, muy precisos y costosos.

- Grado táctico y grado industrial. Los dispositivos clasificados en estas categorías son los más diversos y abordan una amplia gama de sensores con distintos rendimientos, tamaños y costos. Estos se utilizan en aplicaciones que requieren datos de alto rendimiento a un coste menor, como robots industriales o drones. Debido a la reducción en la calidad de los datos que proporcionan, no son suficientemente robustas como para hacer uso únicamente de ellas como sistema de navegación sin apoyo de otros sensores.

- Grado automotriz. El grado más bajo de IMU disponible en el mercado comercial se conoce como grado automotriz. Estos tipos de dispositivos no son lo suficientemente precisos para ser utilizados en navegación inercial, incluso cuando están integradas con otros sistemas de navegación. Como resultado, estos sensores son empleados para detectar eventos, siendo instalados en sistemas de suspensión, airbags, sistemas de entretenimiento y otras aplicaciones similares.

El primer elemento que forma parte de la IMU, el acelerómetro, es un sensor electromecánico capaz de medir las fuerzas de aceleración que actúan sobre él. Normalmente están formados por una pieza capaz de desplazarse cuando se aplican fuerzas causadas por vibraciones o un cambio de movimiento (aceleración), provocando que esta masa se desplace generando una carga eléctrica que es proporcional a la fuerza que se ejerce sobre él (Figura 1.7). La aceleración proporcionada no es lineal, ya que un acelerómetro en reposo, al detectar las fuerzas que actúan sobre él, siente el tirón de la gravedad a menos que se encuentre en caída libre. Existen diferentes tipos de acelerómetros en función de la tecnología empleada para medir la aceleración, los más utilizados son los capacitivos, piezoeléctricos y piezorresistivos. Un hecho relevante en la fabricación de estos sensores es su miniaturización a través de la tecnología MEMS (sistemas microelectromecánicos).

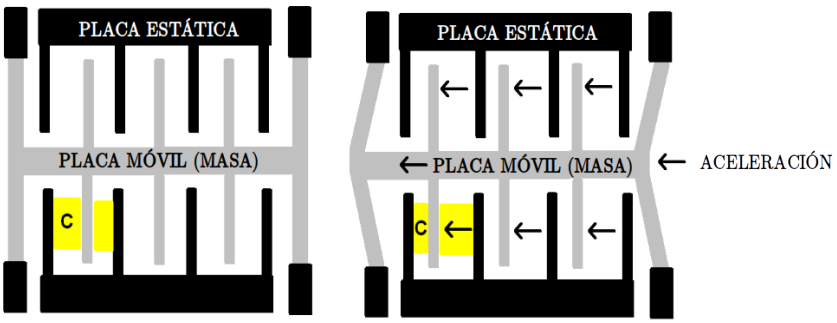


Figura 1.7: Funcionamiento de un acelerómetro.

El segundo elemento que forma parte de la IMU es el giroscopio, un sensor electromecánico capaz de detectar las velocidades angulares. Consta de uno o varios brazos en constante vibración en un mismo sentido. Al mover este sensor, la velocidad angular que recibe el giroscopio incide sobre estos brazos alterando el sentido de la vibración. Estas variaciones son detectadas por los sensores alojados en el giroscopio y convertidas a impulsos eléctricos. Esta velocidad puede ser integrada para obtener la orientación del dispositivo. Existen diferentes tipos de giroscopio en función de la tecnología empleada, los más utilizados son los giroscopios de fibra óptica (FOG), giroscopios láser de anillo (RLG) y, por otra parte, MEMS basados en silicio (Figura 1.8) o cuarzo.

Figura 1.8: Ejemplo de una placa con un giroscopio.

Una de las operaciones principales de Visual SLAM es el cálculo de la pose de la cámara, por tanto, una IMU tiene un gran potencial para complementar a la información visual en el cálculo de esta pose.

1.4. Estructura del documento

Una vez introducidas las disciplinas en las cuales se enmarca el desarrollo del proyecto, en los siguientes capítulos se describe el trabajo realizado.

En primer lugar, en el próximo capítulo se plantea el objetivo del proyecto, y posteriormente se realiza un repaso al estado del arte actual de las técnicas de autolocalización en entornos desconocidos. Los capítulos 4 y 5 describen el diseño, desarrollo y experimentos realizados, incluyendo la metodología empleada para la obtención de los resultados. Por último, se discute y exponen las conclusiones finales del trabajo tras haber analizado los resultados.

Capítulo 2

Objetivos

Una vez realizada una introducción a la temática del proyecto, en este capítulo se exponen los problemas asociados a Monocular SLAM así como los objetivos que se pretenden alcanzar.

2.1. Descripción del problema

Como se ha visto en el capítulo anterior, los algoritmos de Visual SLAM tienen como objetivo la creación de un mapa del entorno que rodea a la cámara así como la estimación de la posición de esta en cada instante de tiempo. Entre los distintos algoritmos de Visual SLAM se encuentra SD-SLAM, desarrollado por RoboticsLabURJC[1], el grupo de robótica de la Universidad Rey Juan Carlos.

Este algoritmo de Visual SLAM puede ser empleado haciendo uso de diferentes configuraciones de cámaras: sistemas monoculares (una única cámara), sistemas estéreo (dos cámaras) o empleando cámaras de profundidad. Para el desarrollo de este proyecto nos centraremos en los sistemas monoculares, debido a que son el sistema más económico, ligero y de menor tamaño. Debido a estas características clave es el sistema típicamente montado en robots aéreos.

Sin embargo, aún tiene asociados una serie de problemas aún sin resolver. Debido al coste computacional, falta de información y ciertas ambigüedades inherentes a los propios sensores de adquisición monoculares, es habitual encontrar una serie de problemas difíciles de abordar. Los principales problemas en los que se centrará el trabajo son los siguientes:

Ambigüedad en la escala. Está asociado al proceso de construcción del mapa inicial, donde no es posible conocer la distancia real a la cual se encuentran los objetos. Este es un problema inherente de los sistema monoculares debido a la pérdida de información 3D al momento de ser proyectada sobre el plano imagen (2 dimensiones). Es decir, un objeto en el mundo real de un determinado tamaño a una cierta distancia proyecta sobre el plano imagen la

[1] `http://www.robotica.gsyc.es/`

misma información que un objeto del doble de tamaño al doble de distancia. Un ejemplo visual de este problema puede observarse en la Figura 2.1. Debido a esto, los sistemas monoculares de SLAM fijan una escala arbitraria en cada proceso de inicialización de la cámara.

Modelo de movimiento. El coste computacional de determinar la nueva posición de la cámara en cada instante de tiempo es elevado. En los algoritmos basados en características este proceso suele ser iterativo, basándose en la retro-proyección y emparejamiento de los puntos visibles del mapa sobre cada nueva imagen. Con el objetivo de acelerar este proceso se emplean técnicas como la estimación de la nueva posición mediante modelos de movimiento o alineaciones de imágenes.

Recuperación ante pérdidas. Cuando la calidad visual disminuye, debido a oclusiones, bajas texturas, movimientos bruscos o secuestros entre otros motivos, provoca que el algoritmo entre en un estado de pérdida del cual no es capaz de recuperarse directamente.

Recientemente se están explorando distintos modos de complementar los algoritmos de Visual SLAM con sensores inerciales para hacer frente a algunos de estos problemas, aunque la mayoría de sensores añaden nuevas dificultades que hay que contemplar. Por ejemplo, la incorporación de una IMU puede complementar el proceso de la estimación de posición y orientación de la cámara. Sin embargo, este sensor sufre de derivas espaciales debido al error acumulativo proveniente de sus medidas. Estos errores pueden proceder de la calidad del sensor, proporcionando datos no ser muy fiables o muy ruidosos. Además el acelerómetro está influenciado

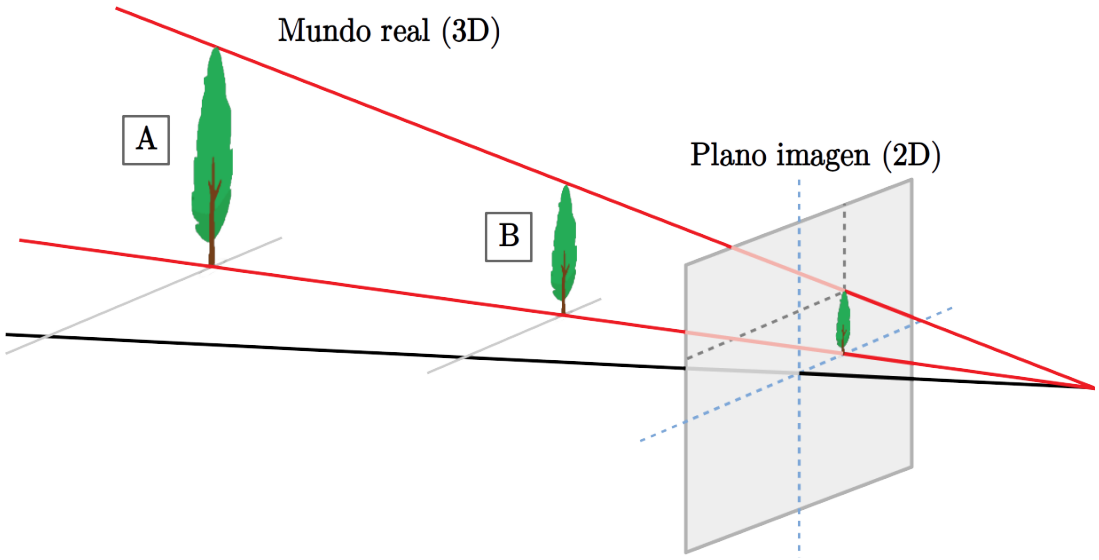


Figura 2.1: Pérdida de información de escala al proyectar objetos del mundo real (3D) al plano imagen (2D). Los árboles *A* y *B* tienen distinto tamaño y se encuentran a diferente distancia del sensor pero su proyección sobre el plano imagen es la misma.

por la fuerza de la gravedad y se necesita una estimación de la orientación para eliminar este efecto. Cualquier pequeño error producido bien por las medidas del sensor o por el modelo de odometría implementado será acumulado de una estimación a la siguiente, es decir, el error aumenta a medida que el tiempo avanza.

2.2. Objetivos

Este proyecto trata de explorar y aportar soluciones a los problemas de los algoritmos de Visual SLAM expuestos en el apartado anterior mediante la incorporación de sensores inerciales. En los últimos años ha crecido el interés por la integración de estos sensores con los principales algoritmos de Visual SLAM [10, 28, 35]. Algunos de los algoritmos de Visual SLAM pertenecientes al estado del arte son analizados en el capítulo 3.

Los objetivos concretos del proyectos, realizados sobre el algoritmo de SD-SLAM, son los siguientes:

1. **Procesar y extraer información fiable de movimiento desde los sensores inerciales.**
2. **Integrar y combinar la información visual junto con la proporcionada por los sensores inerciales.**
3. **Validar experimentalmente las soluciones aportadas con conjuntos de datos públicos internacionales.**

Junto a estos objetivos, se fijan dos únicos requisitos: trabajar en tiempo real y no ver comprometida la robustez inicial del algoritmo.

2.3. Metodología y plan de trabajo

La metodología empleada durante la realización de este Trabajo de Fin de Máster ha sido un modelo de ciclo de vida en espiral, más conocido como metodología ágil. Haciendo uso de esta metodología, los objetivos se realizan siguiendo una espiral en la que cada iteración representa un conjunto de actividades a desempeñar. Este modelo permite cambiar la forma de trabajo ante los nuevos descubrimientos del proyecto, consiguiendo adaptarse fácilmente a ellos, siendo especialmente interesante para proyectos donde se conocen los objetivos que se pretenden realizar pero se desconoce el camino exacto a seguir para lograrlos.

Cada una de las iteraciones de la espiral (conocidas como *sprints*) representa una fase del proyecto llevada a cabo en un intervalo corto de tiempo. Cada uno de los distintos *sprints* comienzan determinando una serie de objetivos que se pretenden realizar. Con esos objetivos

en mente se estudian distintas alternativas para alcanzarlos, abordando diferentes puntos de vista. Una vez establecida la alternativa a realizar se procede a desarrollar la solución propuesta. Finalmente, se estudia el resultado obtenido y, en base a ellos, se planifica la siguiente iteración. En el caso concreto del desarrollo de este proyecto, se han realizado reuniones semanales con los tutores para establecer los objetivos a realizar en cada iteración, evaluar las alternativas de desarrollo y estudiar los resultados obtenidos.

Una vez establecida la metodología a emplear, el proyecto se ha dividido en distintas etapas temporales con el fin de alcanzar los distintos objetivos propuestos. La etapas son las siguientes:

- **Introducción a los algoritmos de Visual SLAM.** En primer lugar se ha realizado un estudio y familiarización con las principales técnicas y algoritmos de Visual SLAM. Para el desarrollo de esta fase los tutores propusieron una serie de lecturas de trabajos relacionados con la temática.

- **Instalación y estudio de SD-SLAM.** En esta etapa se ha procedido a la instalación y estudio del algoritmo de Visual SLAM seleccionado para la realización del TFM: SD-SLAM.

- **Estudio de odometría inercial.** Se ha estudiado el funcionamiento de sensores inerciales y cómo extraer información útil para la autolocalización de ellos. Además, se estudió el estado del arte en las técnicas de Visual SLAM empleando odometría inercial.

- **Mejora del algoritmo SD-SLAM.** Se ha realizado la ampliación de SD-SLAM complementándolo con sensores inerciales para obtener un algoritmo de SLAM visual-inercial.

- **Validación experimental.** Todas las mejoras implementadas han sido validadas de forma experimental.

Capítulo 3

Estado del arte

En este capítulo se explican varios de los algoritmos más significativos que abordan el problema de Visual SLAM empleando una única cámara RGB así como el algoritmo SD-SLAM, que ha sido seleccionado para ampliarlo combinando información inercial y visual para una autolocalización más robusta y precisa.

3.1. MonoSLAM

El primer sistema monocular de Visual SLAM fue desarrollado entre los años 2002 y 2007 por Andrew Davison et al. [5, 6, 7] y finalmente fue presentada su versión final bajo el nombre de MonoSLAM. Esta aproximación está basada en un Filtro Extendido de Kalman (en adelante EKF por sus siglas en inglés) cuyo vector estado está formado tanto por la pose de la cámara así como por la posición de cada uno de los puntos 3D que conforman el mapa.

MonoSLAM necesita como mínimo 4 puntos cuya posición en el espacio o distancias entre sí sean conocidas para realizar el proceso de inicialización. Para este fin se suelen emplear plantillas en forma de tablero de ajedrez como en la Figura 3.1. Con esta información tiene una estimación de profundidad y de escala inicial para poder añadir nuevas características.

Por otra parte, el modelo de predicción se basa en un modelo de movimiento de la cámara de velocidad constante para la estimación de su posición y orientación. En función del movimiento de la cámara nuevos puntos son detectados mediante el operador de detección de Shi y Tomasi [20] y añadidos al mapa y al vector estado.

El modelo de modelo de observación se basa en la proyección de cada uno de los puntos 3D del mapa sobre el plano imagen teniendo en cuenta la pose de la cámara.

El principal problema de este método es que el coste computacional crece proporcionalmente con el tamaño del mapa. Debido a este motivo, MonoSLAM únicamente puede trabajar en tiempo real cuando el mapa consta aproximadamente de 100 puntos o menos (dependiendo del *hardware*). En entornos de mayor tamaño, el vector estado del EKF se vuelve muy grande

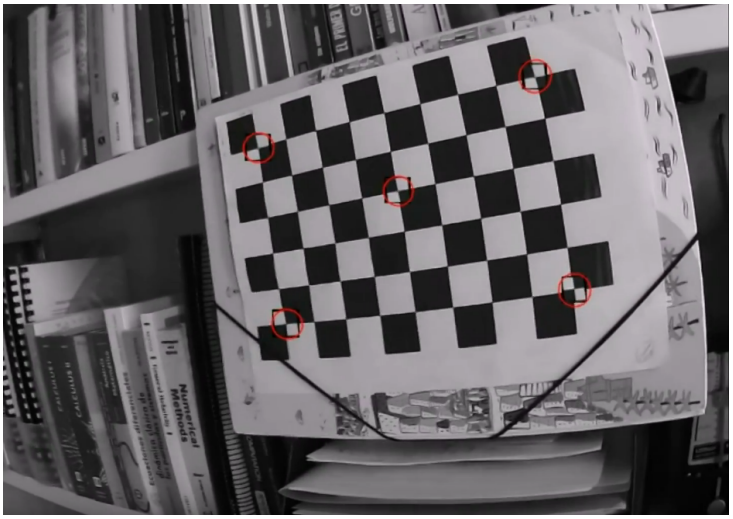


Figura 3.1: Ejemplo de inicialización del algoritmo de MonoSLAM a partir de una plantilla conocida.

debido al número de puntos contenidos en él.

3.2. PTAM

Con el objetivo de solventar el problema computacional de MonoSLAM, Georg Klein y David Murray presentan en el año 2007 el algoritmo *Parallel Tracking and Mapping* [21] más conocido como PTAM por sus siglas.

La idea principal tras este proyecto es la separación del proceso de estimación de la pose de la cámara (*Tracking*) del proceso de generación del mapa (*Mapping*) en 2 hilos asíncronos de ejecución. Esto es posible debido a que solamente es necesario que funcione en tiempo real el proceso de *Tracking*, mientras que el proceso de *Mapping*, que es computacionalmente más costoso, no es necesario que se ejecute en cada iteración. Esta idea hace que sea posible trabajar con mapas de miles de puntos en lugar de cientos como hacía MonoSLAM.

Por otra parte, PTAM añade el concepto de fotograma clave (*KeyFrame* en adelante) para, en lugar de almacenar las características de todos los fotogramas en el mapa, procesar únicamente este subconjunto de fotogramas. Las condiciones para crear un nuevo *KeyFrame* pueden ser temporales (transcurrir un determinado tiempo desde que se creó el último) o espaciales (la distancia respecto al anterior es significativa o la nueva localización es de buena calidad por la cantidad de nueva información que observa).

Por último, el proceso de construcción del mapa inicial del entorno ya no precisa de información a priori, como ocurría en MonoSLAM, sino que hace uso de dos fotogramas que tengan un desplazamiento suficiente entre ellos aplicando el algoritmo de cinco puntos [34]. Este desplazamiento tiene que ser espacial, no basta solo con una rotación. Como se comentó

en la sección 2.1 el mapa generado representará la realidad en una escala diferente a la real.

El proceso de *Tracking* consta de los siguientes pasos:

1. Detección de características: la imagen se subdivide en distintas resoluciones (aplicando una pirámide de imagen) y se aplica a cada una de ellas el detector de esquinas FAST [31] para calcular los puntos de interés. Un ejemplo de los puntos detectados puede observarse en la Figura 3.2a.

2. Estimación de la posición: la pose de la cámara se actualiza con un modelo de velocidad constante, al igual que se hacía en MonoSLAM.

3. Actualización de grano grueso: se selecciona un subconjunto de puntos 3D no mayor de 60 y se retro-proyectan sobre la imagen. Seguidamente, se realiza la búsqueda de su homólogo comparando parches de tamaño 8x8 en un amplio rango alrededor de la posición de la proyección. Una vez obtenidos los emparejamientos, se corrige la estimación actual de la cámara minimizando el error de retro-proyección mediante la técnica iterativa de optimización de Gauss-Newton [30].

4. Actualización de grano fino: se realiza el mismo proceso que en el caso anterior, pero en este caso se retro-proyectan hasta un total de 1000 puntos al mismo tiempo que se reduce el rango de búsqueda del homólogo. Finalmente, la pose de la cámara vuelve a ser corregida minimizando el error de retro-proyección dando lugar a la pose final.

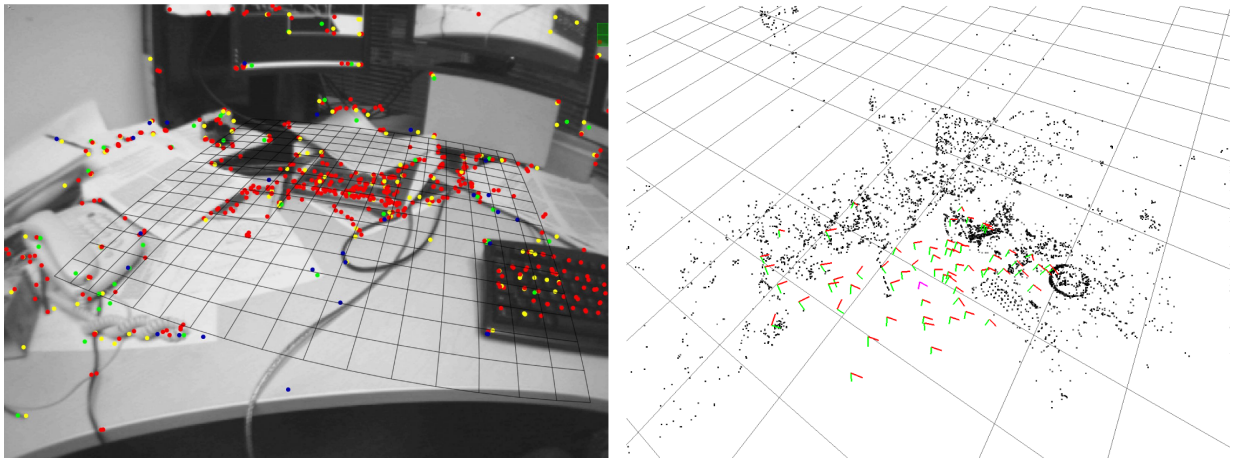
Por otra parte, el proceso de *Mapping*, tras ser inicializado, realiza las siguientes operaciones en cada iteración:

1. Creación de *KeyFrames*: se comprueba en cada instante de tiempo si es necesario crear un nuevo *Keyframe* haciendo uso de las condiciones explicadas anteriormente.

2. Añadir puntos al mapa: en el caso de que se haya añadido un nuevo *KeyFrame*, se detectan puntos característicos en la imagen y se buscan sus homólogos en los anteriores *KeyFrames*. Una vez realizados los emparejamientos, se calcula la posición 3D de cada punto mediante triangulación. Estos puntos son los que empleará el proceso de *Tracking* para corregir la estimación de la pose. Un ejemplo de los puntos detectados puede observarse en la Figura 3.2a.

3. Optimizar el mapa: con el objetivo de refinar la ubicación de los puntos 3D que conforman al mapa se emplea el algoritmo *Boundle Adjustment* [37]. Este es un proceso demandante computacionalmente, por lo que solo se realiza cuando el hilo se encuentra desocupado.

Tras la publicación de PTAM, la mayor parte de los nuevos algoritmos de Visual SLAM emplearán esta división del proceso de *Tracking* y *Mapping* en distintos hilos en sus aproximaciones.



(a) (b)

Figura 3.2: Puntos característicos detectados (a) y mapa generado (b) por PTAM.

3.3. SVO

SVO es la abreviatura de *Fast Semi-Direct Monocular Visual Odometry* [11], algoritmo presentado por Foster et al. en el año 2014. Los algoritmos de Visual SLAM pueden emplear técnicas basadas en características (como MonoSLAM y PTAM) o técnicas directas; SVO hace uso de ambas, convirtiéndolo en un método híbrido. Este algoritmo es similar a PTAM dado que separa los procesos de *Tracking* y *Mapping* en dos hilos de ejecución y la inicialización del mapa se realiza mediante homografía.

En el proceso de *Tracking* es donde podemos encontrar el uso del método híbrido para la estimación de la pose de la cámara obtenida como la minimización del error fotométrico entre cada nuevo fotograma y su anterior. Sin embargo, en lugar de calcularlo sobre la totalidad de la imagen se realiza sobre pequeños parches de tamaño 4x4. Estos parches se localizan alrededor de la proyección en el fotograma actual de los puntos del mapa visibles desde el fotograma anterior (Figura 3.3a).

Al combinar de este modo el uso de puntos de interés junto con los valores de intensidad de la imagen se reduce el coste computacional del proceso obteniendo buenos resultados. Tras obtener esta primera estimación de posición se realiza un ajuste más fino empleando parches de mayor tamaño (8x8 píxeles) para terminar de refinar la pose.

Por otra parte, el proceso de *Mapping*, al igual que PTAM, hace uso *KeyFrames* para la construcción del mapa 3D. Sin embargo, cada nuevo punto detectado no es añadido de instantáneamente al mapa. Esto se debe a que la profundidad a la que se encuentra el punto es modela como una distribución de probabilidad con una incertidumbre inicial muy alta.

La incertidumbre disminuye a medida que se obtienen nuevas observaciones del dicho punto en diferentes fotogramas (Figura 3.3b). Una vez que la varianza es lo suficientemente baja el

punto es añadido al mapa. Al realizar este proceso se puede tener la certeza que la posición de cada punto que es añadido es precisa, pero al mismo tiempo implica que el mapa constará de menos puntos.

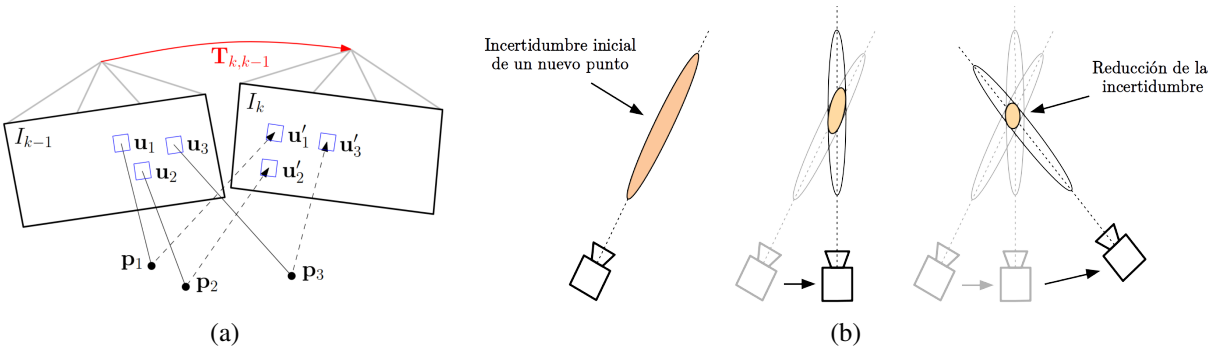


Figura 3.3: Proyección de los puntos del mapa visibles desde el fotograma anterior (a) e incertidumbre en la profundidad de un punto 3D (b) en SVO.

3.4. ORB-SLAM

ORB-SLAM es un algoritmo desarrollado en la Universidad de Zaragoza por Raúl Mur et al. presentado en el año 2015 [26] y mejorado en el año 2017 [27]. Es un método basado en características capaz de ser empleado mediante sistemas monoculares, estéreo o cámaras de profundidad RGB-D. El algoritmo toma este nombre debido a que hace uso de descriptores ORB [32] para la detección y emparejamiento de píxeles de interés.

Una de sus características principales es que, junto a los hilos de ejecución de *Tracking* y *Mapping*, añade un tercer proceso dedicado a la detección de cierres de bucle o *Looping* (explicado en la sección 1.2.1). Haciendo uso de estos tres hilos, el algoritmo puede funcionar en tiempo real siempre y cuando el procesador tenga cierta capacidad de cómputo, dado que, pese a que los descriptores ORB son más robustos que los empleados por los anteriores algoritmos, requieren de un mayor tiempo de cómputo.

Por otra parte, el proceso de inicialización se realiza empleando el método de homografía y, por otra parte, mediante la matriz fundamental haciendo uso del algoritmo de los ocho puntos [18]. Cada uno de los métodos generará un mapa diferente y se determinará de cuál de ellos hacer uso. Si la escena consta de un plano principal se escogerá el mapa obtenido por homografía, y en caso contrario el obtenido mediante la matriz fundamental. Este proceso permite obtener una mayor robustez en la generación del mapa inicial que será determinante en los futuros procesos.

Tanto el proceso de *Mapping* como el de *Tracking* son muy similares a los implementados

en PTAM. El hilo de *Mapping* se encarga de crear *KeyFrames* cuando sea necesario, añadir nuevos puntos 3D al mapa y optimizar el mismo haciendo uso del algoritmo de *Boundle Adjustment*. Sin embargo, en este caso, el emparejamiento de los puntos 3D se realiza mediante los descriptores ORB. Además, ORB-SLAM construye un grafo de co-visibilidad que relaciona unos *KeyFrames* con otros en función de cuantos puntos 3D son observados de forma común. Esto ayuda, entre otras cosas, a eliminar *KeyFrames* redundantes.

Al igual en los casos anteriores, el proceso de *Tracking* realiza en cada iteración una primera estimación de la posición de la cámara empleando un modelo de velocidad constante y empareja los puntos 3D visibles en el fotograma actual y anterior, calculando la posición final a partir de los emparejamientos obtenidos por triangulación. Además, incorpora un mecanismo para recuperar la posición de la cámara ante pérdidas (oclusiones, baja calidad visual, secuestros, etc.). Esto se consigue buscando *KeyFrames* cuya información visual concuerden con la del fotograma actual. Por tanto, es necesario que la cámara vuelva a pasar por un lugar previo almacenado en el mapa para poder relocalizarse. Para reducir el coste computacional de la obtención de los posibles *KeyFrames* candidatos se utiliza un modelo de bolsa de palabras [12]. En la Figura 3.4 se pueden observar puntos característicos detectados mediante ORB y el mapa reconstruído por ORB-SLAM.

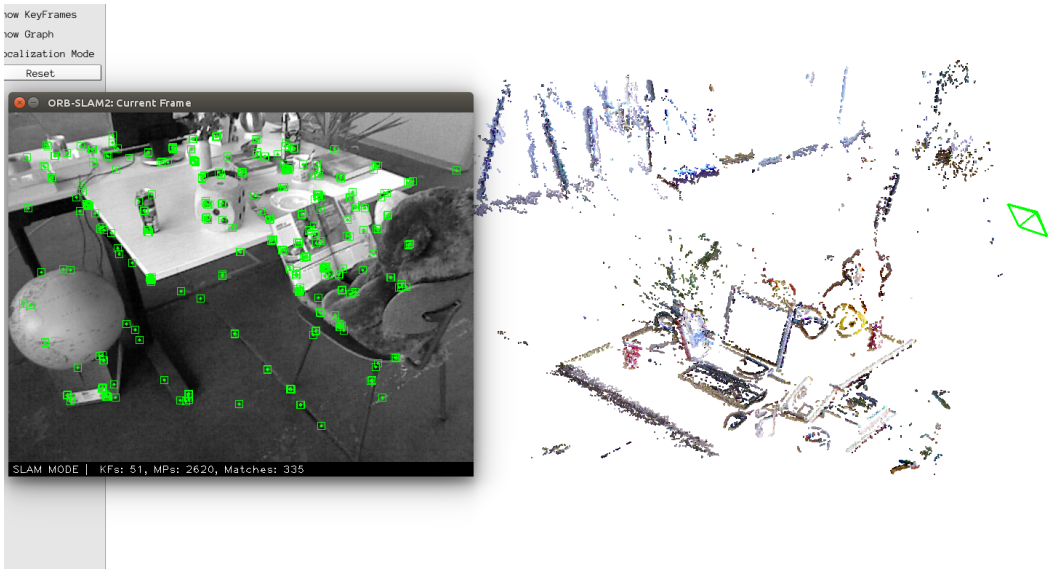


Figura 3.4: A la izquierda se muestra la localización de los puntos característicos detectados con ORB en color verde y el mapa generado a la derecha.

Por último, la función del hilo de *Looping* es la detección de posibles cierres de bucle. Funciona, igual que proceso de relocalización, haciendo uso del modelo de bolsa de palabras junto con el grafo de co-visibilidad. En el caso de detectar un cierre de bucle, se corrige la posición del fotograma actual para hacerla coincidir con el *KeyFrame* en cuestión, se eliminan posibles puntos 3D duplicados, y finalmente se corrige la posición de los *KeyFrames* restantes

para hacer coincidir la trayectoria con la nueva posición.

La estructura propuesta por ORB-SLAM otorga una gran robustez tanto en entornos pequeños como extensos, sumado al hecho de estar liberado como un proyecto de código abierto[1], ha hecho que sea uno de los algoritmos de Visual SLAM más conocidos y utilizados hoy en día.

3.5. SD-SLAM

El algoritmo de Visual SLAM seleccionado para el desarrollo del proyecto es SD-SLAM, que es el acrónimo de *Semi-Direct SLAM*, desarrollado en la Universidad Rey Juan Carlos por Eduardo Perdices como parte de su tesis doctoral [13] en el año 2017. SD-SLAM es el resultado de mejorar ORB-SLAM transformándolo, de un método basado en características, a uno híbrido debido a la adición de métodos directos.

Al ser una extensión de ORB-SLAM consta de todas las características que este disponía, como el uso de 3 hilos de procesamiento (*Tracking*, *Mapping* y *Looping*), soporte para sistemas monoculares, estéreo y cámaras de profundidad RGB-D, uso de descriptores ORB y módulo de relocalización. Pero además, añade una serie de nuevas características como son el uso de métodos directos en el proceso de estimación de la pose de la cámara y una nueva forma de representar los puntos 3D, como se detallará más adelante.

Donde encontramos la principal diferencia es en el hilo de *Tracking*. En ORB-SLAM este proceso consistía, en primer lugar, en una estimación de la posición basada en un modelo de velocidad constante para posteriormente realizar una corrección basada en el emparejamiento de puntos ORB minimizando el error de retro-proyección. Este último proceso es iterativo y, por tanto, si se proporciona una pose inicial cercana a la real el algoritmo convergerá en un menor tiempo.

Con el objetivo de proporcionar una mejor estimación inicial antes de comenzar el proceso de emparejamiento, SD-SLAM añade un método directo basado en la minimización del error fotométrico, similar al realizado por SVO, que denomina alineamiento de imágenes. Este proceso consiste en la proyección de los puntos 3D del fotograma anterior visibles desde el fotograma actual y construye, para cada uno de ellos, parches de tamaño 4x4 píxeles para ser comparados. Además, mientras que en SVO estos parches se transforman aplicando una matriz afín asociada al movimiento antes de ser comparados, SD-SLAM evita este paso dado que presupone que el desplazamiento entre fotogramas será pequeño y se podrá asumir que no existe una gran variación en las poses y, por consiguiente, en los parches. Finalmente, se minimiza el error fotométrico (problema no lineal de mínimos cuadrados) con el algoritmo de Gauss-Newton [30].

[1] `https://github.com/raulmur/ORB_SLAM2`

Este proceso de alineación de imágenes se realiza haciendo uso de una pirámide de imágenes para detectar rápidamente el desplazamiento en las imágenes de menor tamaño, para posteriormente ser refinado a medida que se avanza por la pirámide. El resultado es una segunda estimación de pose muy rápida y cercana a la final, agilizando el proceso final de corrección (emparejamiento y minimización del error de retro-proyección). De este modo se logra una reducción en el tiempo de cómputo respecto a ORB-SLAM, resultando especialmente útil para incorporar este algoritmo en sistemas con capacidad de cómputo limitada, como por ejemplo los drones.

Los módulos de relocalización y detección de cierres de bucle (*Looping*) también han sido mejorados para hacer uso del alineamiento de imágenes. Estas dos operaciones se basan en la comparación del fotograma actual con un subconjunto de *KeyFrames* potencialmente similares, y por tanto, el tiempo de ejecucuón es dependiente del número de *KeyFrames* presentes en el mapa, dificultando su ejecución en tiempo real cuando este número crece. Al ser modificadas del mismo modo que se hizo en el proceso de *Tracking* permite reducir el tiempo de cómputo y tolerar así un mayor número de *KeyFrames*.

Por otra parte, el hilo de *Tracking* trabaja tal y como lo hacía en el algoritmo de ORB-SLAM exceptuando el modo en el que se representan los puntos 3D del mapa. La mayoría de algoritmos de Visual SLAM representan los puntos 3D mediante las coordenadas de su posición espacial en el mundo (X, Y, Z) desde un origen arbitrario (normalmente el primer *KeyFrame* del mapa). La posición 3D es calculada mediante triangulación. Este proceso es dependiente de que el ángulo que forman los rayos de retro-proyección sea suficientemente amplio; es decir, el desplazamiento (paralaje) entre las observaciones debe ser suficiente. A medida que este ángulo disminuye, el punto se aleja y su indeterminación crece, provocando potenciales errores. De hecho, existen objetos para los cuales nunca se logrará un desplazamiento suficiente para poder estimar su posición. Por ejemplo, las estrellas se podrían considerar en el "infinito" y, por tanto, no será posible obtener un paralaje suficiente para estimar su profundidad real.

SD-SLAM hace uso de la inversa de la profundidad [4] para representar los puntos 3D. Además, la posición de estos puntos se representa como una matriz de transformación (rotación y traslación) respecto de la cámara donde fueron detectados por primera vez. De este modo, los puntos 3D están referenciados a los distintos *KeyFrames* donde se detectaron por primera vez en lugar de a un origen arbitrario. La ventaja de esta representación en comparación a la parametrización clásica es que es posible representar puntos en el infinito, dado que la inversa de la profundidad en el infinito es 0.

Por todas las características descritas que dan forma al algoritmo de SD-SLAM se ha decido hacer uso de este para el desarrollo del proyecto. Los principales motivos para la elección de este algoritmo son los siguientes:

- Código abierto[2].
- Soporta sistemas monoculares.
- Capacidad de trabajar en tiempo real.
- Inicialización robusta (homografía y matriz fundamental).
- Incorpora un módulo para detectar cierres de bucle.
- Emplea hilos asíncronos para los módulos de *Tracking*, *Mapping* y *Looping*.
- Es capaz de representar puntos en el infinito.
- Es un algoritmo desarrollado por el grupo de robótica de la URJC, y se busca hacerlo más potente y versátil.

[2] `https://github.com/JdeRobot/SDslam`

Capítulo 4

Diseño e implementación

En este capítulo se describe la implementación realizada para estimar una trayectoria basada en odometría inercial y su integración en SD-SLAM, dotándolo de nuevas características que no contenía previamente.

4.1. Diseño

Como ya se ha comentado, la unidad de medición inercial (IMU) incorpora un acelerómetro y un giróscopo que permite medir cada componente de la aceleración lineal y la velocidad angular en las tres dimensiones. La idea principal es emplear estos valores para obtener una estimación de posición y orientación que complementen a las estimaciones de pose realizadas por el módulo de *Tracking* visual. El funcionamiento original en SD-SLAM de este módulo puede ser representado por una máquina de estados finitos, como puede observarse en la Figura 4.1.

En un primer momento, el algoritmo emplea un módulo de **Inicialización** encargado de construir el mapa inicial de puntos 3D a partir de dos fotogramas. La posición de estos puntos y las poses de la cámara para ambos fotogramas es obtenida por triangulación, haciendo uso de al menos 100 características comunes detectadas por ORB. Este proceso puede realizarse mediante Homografía o Matriz Fundamental, seleccionando el modelo que menor error de retro-proyección obtenga. Debido al desconocimiento de la información de profundidad, la escala del mapa inicial es arbitraria y varía en cada ejecución. La incertidumbre en la escala es uno de los principales problemas de los sistemas monoculares.

Una vez el sistema ha sido inicializado transita al estado ***Tracking* visual** encargado de realizar la estimación de la pose de la cámara cada vez que se recibe un nuevo fotograma. Este proceso se divide en varias fases, realizando en primer lugar una estimación basada en un modelo de velocidad constante, dado que es razonable asumir que entre dos fotogramas consecutivos no hay excesiva variación de movimiento. Seguidamente, esta estimación es refinada con la

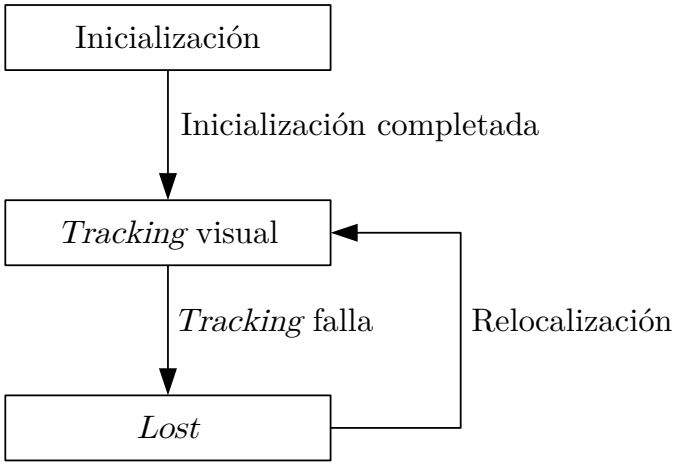


Figura 4.1: Funcionamiento del algoritmo SD-SLAM como una máquina de estados finitos.

información visual, aplicando el sistema híbrido de SD-SLAM (alineación de imágenes seguido de la reducción del error de retro-proyección con las características detectadas por ORB). Este proceso es satisfactorio siempre y cuando se detecten al menos 20 emparejamientos en el proceso de retro-proyección. En el caso de que este número descienda, SD-SLAM considera que la calidad visual se ha deteriorado provocando que la estimación de pose sea errónea y el algoritmo entrará en el estado de pérdida. Normalmente esta situación ocurre cuando la calidad visual disminuye total o parcialmente, debido a giros bruscos o secuestros del dispositivo, entre otros.

Durante el estado de pérdida o ***Lost***, el algoritmo tratará de relocalizarse comparando cada nuevo fotograma con aquellos *KeyFrames* visualmente similares. Cuando se detecten al menos 20 puntos comunes se considera que la cámara se encuentra en un lugar previamente visitado. Si esto llega a ocurrir, el sistema volvería a transitar al estado anterior. Esta es la mayor debilidad de SD-SLAM (y de los sistemas monoculares de Visual SLAM en general), ya que durante este estado el algoritmo no puede realizar las funciones para las que ha sido programado. De hecho, SD-SLAM puede permanecer en el estado de pérdida hasta la finalización de la ejecución en el caso de que la cámara nunca vuelva a transitar por una zona conocida.

Hay muchas maneras posibles de complementar el algoritmo puramente visual de SD-SLAM con información inercial. Algunas de las opciones de fusión que se han barajado a lo largo del desarrollo de este TFM son las siguientes:

1. La primera aproximación consistía en mantener una odometría inercial y una odometría visual totalmente paralelas a lo largo del tiempo. De esta forma se podría recurrir a las estimaciones realizadas por cada modelo en cualquier momento. Sin embargo, debido a la deriva espacial que sufren ambos modelos a medida que el tiempo avanza, la concor-

dancia entre ambos modelos es nula pasado un cierto periodo de tiempo y vuelve esta idea irrealizable.

2. La segunda idea consistía en complementar el proceso de estimación de la pose del módulo de *Tracking* visual con información inercial. En la actualidad ya existen diversos trabajos [25] que abordan esta idea, donde principalmente el modelo de velocidad constante original es sustituido por un modelo de odometría inercial, con la idea de aportar mayor robustez ante movimientos dinámicos. En un primer momento se llevó a cabo esta idea de dos maneras diferentes, sin embargo, no se pudo determinar si la mejora en robustez era real o no, dado que se obtenían resultados similares que al emplear el modelo de movimiento original (velocidad constante). Además, hay que tener en cuenta que los algoritmos de visual SLAM no son deterministas (en cada ejecución se obtiene un resultado diferente) y esto dificulta el proceso de evaluación.

3. La última idea consistía en realizar todas las estimaciones de pose con la odometría puramente visual, y solamente en los casos de pérdida recurrir a la odometría inercial. De este modo, mientras visión permanezca activa se hará uso de sus estimaciones para corregir y relacionar el modelo de odometría inercial con el modelo visual y, de esta forma, mantener la concordancia entre ambos modelos. En el momento en el que visión falle, el modelo de odometría inercial tomaría el relevo y comenzaría a suministrar estimaciones de pose a SD-SLAM hasta que visión sea capaz de recuperarse.

Después de pensar y probar las anteriores alternativas, la opción seleccionada para ahondar en ella es la tercera. El principal objetivo que se busca es dotar a SD-SLAM de la capacidad de mantener la estimación de trayectoria cuando la información visual proporcionada por la cámara sea insuficiente para realizar las operaciones habituales. De esta forma, se vuelve plausible la idea de recuperarse de una perdida sin necesidad de revisitar una zona conocida. Además, de forma necesaria se obtiene un factor de escala capaz de relacionar la escala de la escena (obtenida de forma arbitraria por la parte puramente visual) con la escala del mundo real. Un caso típico del comportamiento deseado puede observarse de manera esquemática en la Figura 4.2.

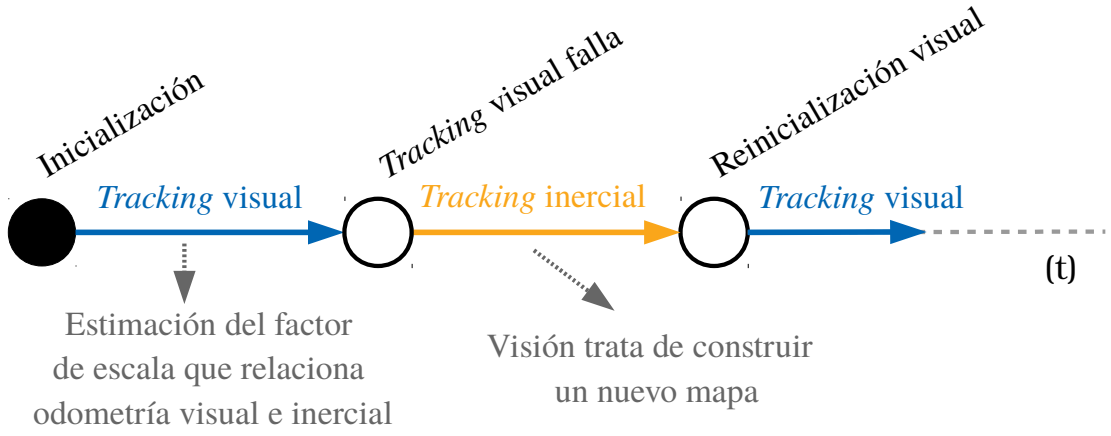


Figura 4.2: Caso típico del comportamiento esperado tras la ampliación de SD-SLAM.

Con este objetivo en mente, las principales características introducidas al funcionamiento original de SD-SLAM son las siguientes:

- Se estimará una odometría inercial a partir de la información proporcionada por la IMU que complementará la odometría visual. Las poses finales obtenidas en el proceso de *Tracking* visual al final de cada iteración servirán de retroalimentación y autocalibración del modelo inercial para que las estimaciones basadas únicamente en IMU sean más precisas.

- Debido a que la escala obtenida en el proceso de inicialización por los sistemas monoculares es arbitraria, se ampliará este módulo para que, además de construir el mapa inicial, estime un factor de escala λ que relacione tamaño del mundo de real con el mundo de SD-SLAM. Este es un parámetro necesario para poder combinar las estimaciones visuales con las inerciales.

- Mientras el algoritmo se encuentre en el estado de pérdida, se seguirá estimando la trayectoria de la cámara mediante la odometría inercial. Esto dará lugar a un nuevo estado llamado ***Tracking* Inercial**.

- Para transitar desde el nuevo estado a la odometría original de SD-SLAM, es necesario construir un nuevo mapa. Este proceso no era posible en el algoritmo original debido a la incapacidad de relacionar la posición de un nuevo mapa respecto al anterior. Con la incorporación de la odometría inercial desarrollada en este trabajo, esa relación se vuelve plausible y se realiza con éxito.

Al aplicar todas las nuevas características sobre SD-SLAM, el diagrama estados original y condiciones para transitar de uno a otro han sido modificados, como se puede observar en la

Figura 4.3. Primero, se ha renombrado el estado de pérdida a *Tracking* inercial, ya que mientras el algoritmo se encuentre en el estado de pérdida se continuará estimando la trayectoria con el modelo inercial. Y segundo, la transición al estado de *Tracking* visual desde el nuevo estado se realiza mediante una reinicialización en lugar de una relocalización. Estos cambios dan lugar a un nuevo algoritmo combinado que hemos bautizado como VIO-SDSLAM (Visual Inertial Odometry SD-SLAM).

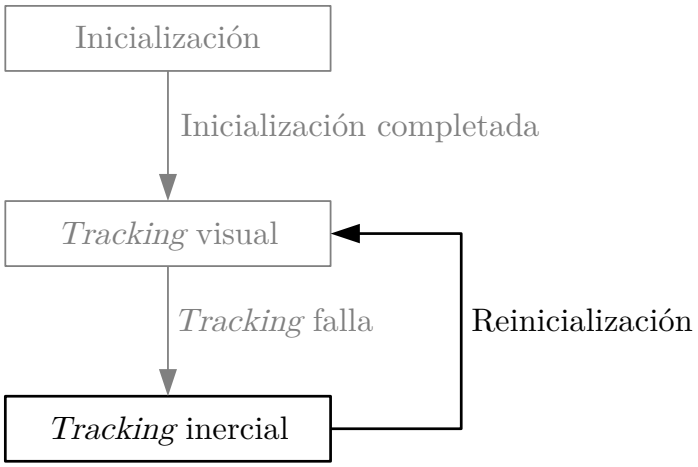


Figura 4.3: Funcionamiento del algoritmo VIO-SDSLAM como una máquina de estados finita con el nuevo estado de *Tracking* inercial basado en IMU.

4.2. Sistemas de referencia

Antes de comenzar con las ampliaciones realizas sobre SD-SLAM, es importante definir los tres sistemas de referencia con los que trabajaremos. Los dos primeros hacen referencia al modo en el que SD-SLAM expresa las poses de la cámara, que llamaremos Cámara y Mundo, siguiendo la notación habitualmente empleada en Visual SLAM. Por otra parte, las medidas y poses inerciales son expresadas en su propio sistema de referencia, que llamaremos IMU. A partir de ahora nos referiremos a los tres sistemas de referencia con los superíndices C, W e I, respectivamente, y la relación entre cada uno de ellos, que es explicada a continuación, puede observarse en la Figura 4.4.

En primer lugar, el sistema de referencia de Cámara solidario al dispositivo representa la posición del origen del mundo respecto de la pose de la cámara, es decir, el origen de coordenadas es la propia cámara. Todas las poses son expresadas mediante una matriz de coordenadas homogéneas de tamaño 4x4, formada por la orientación (matriz de rotación R de tamaño 3x3) y la posición (vector de traslación t 3x1). A diferencia de otros algoritmos de Visual SLAM

que representan cada nueva pose como el desplazamiento relativo llevado a cabo respecto de la pose anterior, en SD-SLAM cada pose es independiente, es decir, se da con respecto al mismo sistema de referencia, no con respecto al fotograma anterior.

Por otra parte, la pose de la cámara expresada por SD-SLAM en el sistema de referencia Mundo traslada el origen de coordenadas desde la cámara al origen del mundo. De este modo, la pose de la cámara en el sistema de Mundo (P^W) en cada momento es expresada como la traslación y rotación realizada desde el origen de coordenadas del mundo, y no viceversa como se hacía en el caso anterior. Es por este motivo que las poses en Cámara (P^C) y Mundo se relacionan entre sí mediante su inversa:

$$P^W = (P^C)^{-1} \tag{4.1}$$

donde la inversa es definida como:

$$\begin{pmatrix} R & t \\ 0^T & 1 \end{pmatrix}^{-1} = \begin{pmatrix} R^T & -R^T t \\ 0^T & 1 \end{pmatrix} \tag{4.2}$$

La dirección y sentido de los ejes de la cámara (como puede observarse en la Figura 4.4 con los colores rojo, verde y azul) expresada en Mundo emplea la convención EDN habitual en la representación de las cámaras, donde las coordenadas X, Y, Z representan las direcciones Este (East), Abajo (Down) y Norte (North).

Por último, las poses inerciales (P^I) se enmarcan en el sistema de referencia IMU. Este sistema expresa las poses del mismo modo que el sistema de referencia Mundo, es decir, las poses son expresadas como la rotación y traslación realizadas desde el origen del mundo. Sin embargo, hay que tener en cuenta dos diferencias clave:

- La escala de las poses es diferente, ya que las medidas inerciales están asociadas a la escala del mundo real y las poses de la cámara están asociadas a una escala arbitraria (obtenida en el proceso de inicialización de SD-SLAM Monocular). Es por este motivo que se necesita calcular un factor de escala λ que relacione la escala de ambos sistemas.

- La dirección y sentido de los ejes de este sistema es distinta y varía en función del fabricante de la IMU, aunque se suele emplear la convención NWU [2], donde la coordenadas X, Y, Z son asignadas las direcciones Norte (North), Oeste (West), y Arriba (Up). Indiferentemente de la convención empleada, los sistemas de referencia IMU y Mundo pueden ser relacionadas mediante una matriz de rotación que denominaremos R^{IW}. Esta matriz es variable, pero una vez determinada es invariante en el tiempo.

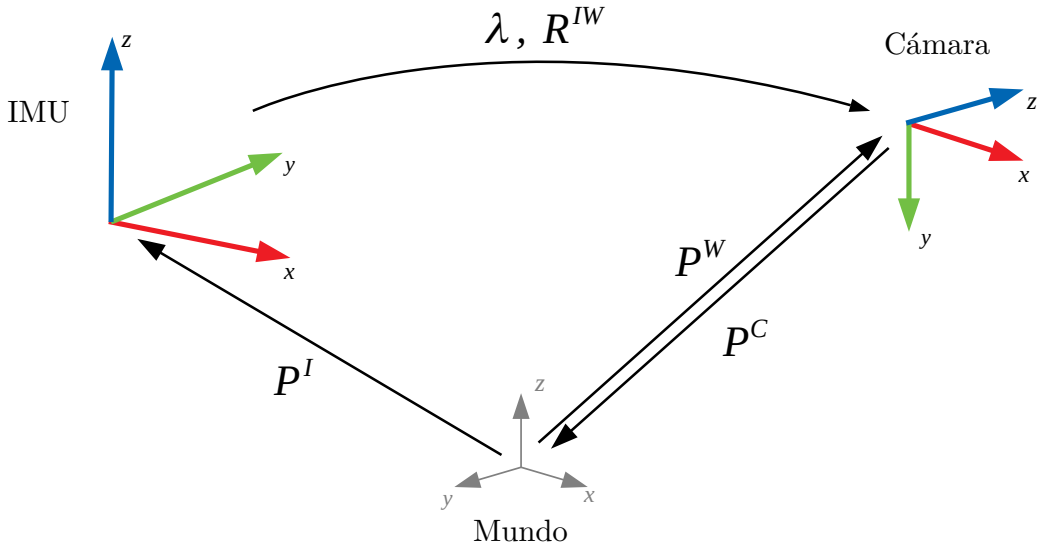


Figura 4.4: Representación de los sistemas de referencia empleados y la relación entre ellos.

4.3. Salto a Odometría inercial

La odometría inercial es el proceso de estimación de la posición y orientación de la cámara a partir de la información proporcionada por la IMU. Estas poses servirán para complementar las estimaciones realizadas por el algoritmo de SD-SLAM empleando únicamente visión, y así mejorar las estimaciones finales del algoritmo enriquecido, VIO-SDLSAM.

En primer lugar es necesario entender la naturaleza de estos valores obtenidos de los sensores inerciales. Las medidas del sensor inercial suelen ser bastante ruidosas y se suelen modelar [39] descomponiendo este error en dos tipos: uno que fluctúa rápidamente (ruido aleatorio), y otro que varía lentamente (*bias*) pues suele ser dependiente de factores como la temperatura del sensor. De este modo, las medidas pueden ser expresadas del siguiente modo:

$$\tilde{a}(t) = a(t) + b_a(t) + \eta_a(t) \tag{4.3}$$

$$\tilde{\omega}(t) = \omega(t) + b_g(t) + \eta_g(t) \tag{4.4}$$

donde:

- $\tilde{\boldsymbol{a}}(t)$ es la medida de aceleración proporcionada por el acelerómetro y $\boldsymbol{a}(t)$ la aceleración real a la cual es sometida, respectivamente, en el instante de tiempo t. Los valores son expresados en m/s^2.

- $\tilde{\boldsymbol{\omega}}(t)$ es la medida de velocidad angular proporcionada por el giroscopio y $\boldsymbol{\omega}(t)$ la velocidad angular real a la cual es sometida el sensor, respectivamente, en el instante de tiempo t. Los valores son expresados en rad/s.

- $\boldsymbol{b_a}(t)$ y $\boldsymbol{b_g}(t)$ es el sesgo o *bias* que afecta al acelerómetro y giroscopio. Puede ser considerado constante pues varía lentamente en el tiempo.

- $\boldsymbol{\eta_a}(t)$ y $\boldsymbol{\eta_g}(t)$ es el ruido aleatorio que afecta al acelerómetro y al giroscopio. Normalmente es modelado como un ruido blanco gaussiano de media cero.

Además, el vector de aceleración $\tilde{\boldsymbol{a}}(\boldsymbol{t})$ medido es suma de la aceleración del sensor y la aceleración de la gravedad, siendo constante en el sistema de referencia del sensor, pero su descomposición en componentes no constante en el sistema de referencia de la cámara. Esto implica que un acelerómetro en reposo, a menos que se encuentre en el espacio, siempre va a estar sometido a una aceleración de aproximadamente 9.8 m/s^2. Esta fuerza se reparte entre las tres componentes de la aceleración en función de la orientación del sensor, por tanto, no es recomendable estimar la posición del sensor a partir de la medición del acelerómetro sin eliminar primero esta componente.

4.3.1. Estimación inercial

Como se explicó en la sección 1.3.1, en el mercado existe una amplia variedad de IMUs, abordando distintas calidades y precios, desde las baratas y ruidosas hasta las más caras pero altamente precisas. La viabilidad de que la pose estimada sea fiable depende de la calidad del propio sensor. Sin embargo, independientemente de la calidad de la IMU y una vez introducida la naturaleza de los datos que obtenemos de este dispositivo, el proceso que seguiremos para obtener la pose (posición y orientación) del sensor puede expresarse mediante el diagrama mostrado en la Figura 4.5.

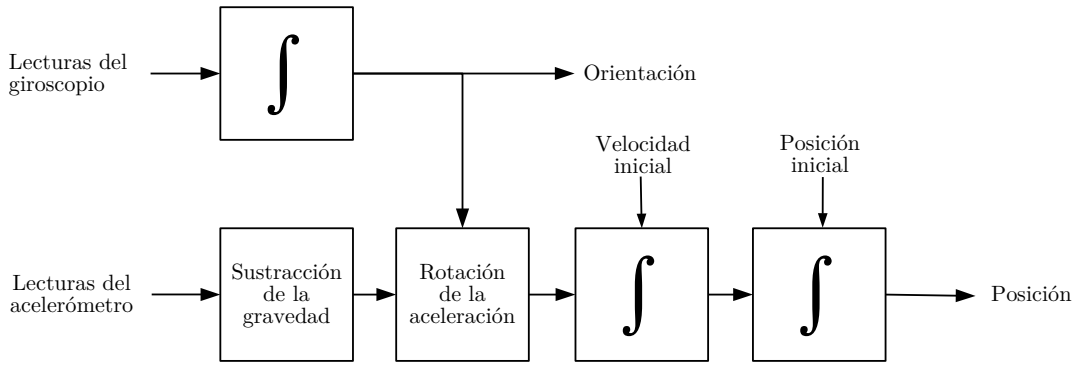


Figura 4.5: Proceso para la estimación de la posición y orientación del sensor IMU.

En primer lugar, se procesa la información del giroscopio para obtener una estimación de la orientación del sensor. Por otra parte, las lecturas de aceleración son solidarias al dispositivo, por tanto, se hace uso uso de la orientación anteriormente estimada para rotar el vector de aceleración y el vector de gravedad (para calcular su efecto sobre cada una de las componentes). Posteriormente se calcula la diferencia entre ambos vectores obteniendo así una aceleración lineal. Esta aceleración será procesada para obtener finalmente una estimación de la posición.

En la actualidad existen diversos algoritmos para obtener la orientación a partir de IMUs, como el uso de Filtros extendidos de Kalman (EKF), el algoritmo de Filtro Complementario [9] o el algoritmo de Madgwick [24]. Este último ha sido el seccionado para este proyecto. El algoritmo de Madgwick fusiona los datos del giroscopio, acelerómetro y, opcionalmente, magnetómetro, para estimar la orientación del sensor. Fue desarrollado en el año 2011 por Sebastian Madgwick.

En cada iteración, el algoritmo de Madgwick realiza una nueva estimación de orientación teniendo en cuenta la orientación anterior, el intervalo de tiempo transcurrido, las medidas de la IMU y un parámetro β. Este último se define como la ganancia del filtro y sirve para controlar la variación de la orientación de un instante de tiempo a otro. Así, unos valores bajos de β filtrarán gran parte del ruido, generando una respuesta más lenta pero realizando transiciones suaves, mientras que valores más altos provocarán una respuesta más rápida pero ruidosa.

Por otra parte, para realizar la estimación de la posición se integra la aceleración aplicando las ecuaciones del movimiento, asumiendo que la aceleración es constanwte entre cada medición del acelerómetro. En primer lugar, se elimina la componente de la gravedad de la señal del acelerómetro para aislar la aceleración lineal:

$$a = R^I(\tilde{a} - g) \tag{4.5}$$

donde a es la aceleración lineal, $\tilde{a}$ es la aceleración medida por el acelerómetro, R^I es la estimación de la orientación del sensor obtenida en el paso anterior, y g es el vector de gravedad que afecta al acelerómetro en reposo (por simplicidad se excluyen las dependencias explícitas con el tiempo).

Una vez se dispone la aceleración lineal, se estima la velocidad y seguidamente la posición:

$$v_t = v_{t-\Delta t} + a_t \Delta t \tag{4.6}$$

$$s_t = s_{t-\Delta t} + v_{t-\Delta t}\Delta t + \frac{1}{2} a_t \Delta t^2 \tag{4.7}$$

donde Δt es el intervalo de tiempo transcurrido entre el instante actual t y el anterior, v_t y $v_{t-\Delta t}$ son la velocidad lineal en el instante actual y el anterior respectivamente, s_t y $s_{t-\Delta t}$ es la posición en el instante actual y el anterior respectivamente.

El resultado de este proceso es la obtención de una estimación de posición ($\hat{s}_t^I$) y orientación ($\hat{R}_t^I$) en el sistema de referencia IMU. Para hacer uso de ellas en SD-SLAM es necesario trans-

formarlas a las coordenadas Mundo. Primero, la orientación únicamente requiere de aplicar la matriz de transformación que relaciona ambos sistemas:

$$\hat{R_t^W} = R^{IW}\hat{R}_t^I(R^{IW})^{-1} \tag{4.8}$$

Y segundo, para que la posición estimada por SD-SLAM y el incremento posición calculado desde la IMU se puedan combinar es necesario que ambas estén en la misma escala, ya que SD-SLAM inicializa en una escala arbitraria. Debido a este motivo, la transformación al sistema de referencia Mundo se realizará incrementalmente. De este modo, se define Δs_t^I como el desplazamiento realizado en el sistema inercial entre los instantes de tiempo t y $t-1$ como:

$$\Delta s_t^I = \hat{s}_t^I - s_{t-1}^I \tag{4.9}$$

siendo $\hat{s}_t^I$ la estimación de posición realizada y s_{t-1}^I la posición en el instante de tiempo anterior. Aplicando este desplazamiento es posible calcular la posición en el sistema de referencia Mundo de la siguiente manera:

$$\hat{s}_t^W = s_{t-1}^W + R^{IW}(\lambda\, \Delta s_t^I) \tag{4.10}$$

donde $\hat{s}_t^W$ es la estimación de posición inercial en coordenadas Mundo para el fotograma actual, s_{t-1}^W es la posición del fotograma anterior en coordenadas Mundo y λ el factor de escala que relaciona ambos sistemas.

Esta aproximación no es perfecta principalmente por dos motivos. En primer lugar, es dependiente de una buena estimación de orientación. Un error en esta estimación provoca una incorrecta proyección del vector de gravedad, causando que la aceleración sea integrada en una dirección errónea [19]. En segundo lugar, la estimación de la posición es obtenida a partir de una aceleración a la cual no se ha eliminado el ruido. Además, todos estos pequeños errores en la aceleración tenderán a acumularse en la velocidad estimada. Teniendo en cuenta estas consideraciones se puede deducir que las estimaciones de pose inerciales tenderán a degradarse en el tiempo. Por este motivo, el modelo inercial se corrige con las poses calculadas por el módulo de *Tracking* visual mientras este permanezca activo. Cuando visión falle (estado de pérdida), el proceso de corrección no podrá ser llevado a cabo debido a la ausencia de estimaciones de pose visuales.

El módulo de *Tracking* visual estima la pose de la cámara tras procesar cada nuevo fotograma realizando las operaciones mencionadas anteriormente: modelo de movimiento de velocidad constante, seguido de un alineamiento de imágenes y por último la reducción del error de retroproyección. La idea propuesta es emplear el desplazamiento realizado por la cámara (estimado por el modelo puramente visual) entre los dos últimos fotogramas procesados para corregir la estimación de posición y velocidad del modelo inercial. El desplazamiento de la cámara Δs^W

se puede expresar en el sistema de referencia de IMU del siguiente modo:

$$\Delta s^I = (R^{IW})^{-1} \frac{1}{\lambda} \Delta s^W \tag{4.11}$$

Con este valor se corrige la posición estimada por odometría inercial y posteriormente la velocidad:

$$s_t^I = s_{t-1}^I + \Delta s^I \tag{4.12}$$

$$v_t^I = \frac{s_t^I - s_{t-1}^I}{\Delta t} \tag{4.13}$$

Por otra parte, la orientación también es corregida, sin embargo, debido a que la orientación no es afectada por la diferencia en la escala de los sistemas, únicamente se necesita actualizar el valor de la orientación almacenado en el filtro Madgwick. De este modo, se empleará esta orientación como punto de partida para la siguiente estimación de orientación inercial. Para realizar la corrección se expresa la orientación obtenida por visión en el sistema de referencia IMU y se incluye en el estado del algoritmo Madgwick.

4.3.2. Estimación y actualización de la escala

Con el objetivo de hacer uso de las estimaciones inerciales en SD-SLAM Monocular es necesario estimar un factor de escala λ que relacione la escala del mundo real con la escala de la escena visual. De ese modo sí se podrá combinar con la información visual, que está en la escala real. El proceso de inicialización de los parámetros que relacionan ambos sistemas en un modelo Visual-Inercial determinará en gran medida su funcionamiento. En la literatura se ha abordado este problema con diferentes enfoques: haciendo uso de información mutua empleando Filtros de Kalman [25], planteándolo como un problema de optimización de grafos [28] o tratando de forma independiente el módulo de IMU y Visual SLAM [23]. Los sistemas Visuales-Inerciales de SLAM no han alcanzado aún los niveles de robustez requeridos en muchos casos debido, en gran parte, a los problemas y limitaciones que presenta esta etapa.

En este Trabajo de Fin de Máster, el proceso de inicialización tratará de forma independiente ambos sistemas, el visual y el inercial, hasta que el algoritmo construya el mapa inicial, momento en el cual se estimará el factor escala inicial. Durante este periodo, el módulo de odometría inercial trabajará de forma paralela al módulo visual, estimando la pose de cada nuevo fotograma que reciba SD-SLAM. La escala se calcula en el momento de creación del mapa inicial, ya que es cuando se calculan las dos primeras estimaciones de pose visuales, almacenadas en los dos primeros *KeyFrames* del mapa. Una vez se dispone de dos poses estimadas únicamente por visión, el factor de escala es calculado como la relación de entre el desplazamiento realizado por ambos modelos para los dos primeros *KeyFrames*.

Haciendo uso estos *KeyFrames*, que denominaremos 0 y 1, la ecuación 4.14 define el factor de escala λ como la relación entre las magnitudes del desplazamiento espacial realizado

por el modelo inercial respecto al estimado por visión. Los *KeyFrames* son los encargados de almacenar los factores de escala y proporcionárselos al modelo de movimiento inercial.

$$\lambda = \frac{\parallel s^W_{V,1} - s^W_{V,0} \parallel}{\parallel s^I_{I,1} - s^I_{I,0} \parallel} \tag{4.14}$$

donde $s^W_{V,0}$ y $s^W_{V,1}$ son la posición estimada por Visión para los dos primeros *KeyFrames* en coordenadas Mundo, y $s^I_{I,0}$ y $s^I_{I,1}$ son la posición estimada por la odometría inercial para los mismos fotogramas en el sistema de referencia de la IMU. Para la realización de este proceso no es necesario aplicar una transformación entre los sistemas de coordenadas, dado que únicamente se hace uso de la magnitud del vector, indiferentemente de su posición espacial u orientación.

Este factor de escala variará lentamente en el tiempo debido a la lenta deriva que sufren los sistemas monoculares, por lo que es conveniente actualizarlo de vez en cuando. Cada vez que el algoritmo procesa un fotograma, se calcula un nuevo factor de escala λ_k que es almacenado en un *buffer* hasta la creación de un nuevo *KeyFrame*. Al momento de actualizar el factor antiguo se ha optado por añadir un parámetro α que controle la velocidad de actualización entre el antiguo y el nuevo. La nueva escala se calcula como una ponderación entre la antigua y la escala media de los factores asociados a cada fotograma procesado desde el anterior *KeyFrame* almacenados en el *buffer*:

$$\lambda_{i+1} = (1-\alpha)\lambda_i + \alpha\frac{1}{n}\sum_{k=0}^{n}\lambda_k \tag{4.15}$$

donde λ_i representa el factor de escala actual, λ_{i+1} es el nuevo factor de escala estimado, n es el número fotogramas procesados entre dos *KeyFrames*, λ_k son los factores de escala estimados para cada fotograma y α es un parámetro que toma valores comprendidos en el intervalo cerrado $[0, 1]$, donde el valor 0 representa un factor de escala constante (el primer factor estimado) y el valor 1 ignora el factor previo para emplear únicamente la nueva estimación del factor de escala. En la Figura 4.6 puede observarse una representación visual del proceso de actualización de la escala, calculando el factor λ_{i+1} entre los *KeyFrames* i e $i+1$, procesando tres fotogramas entre la creación de ambos.

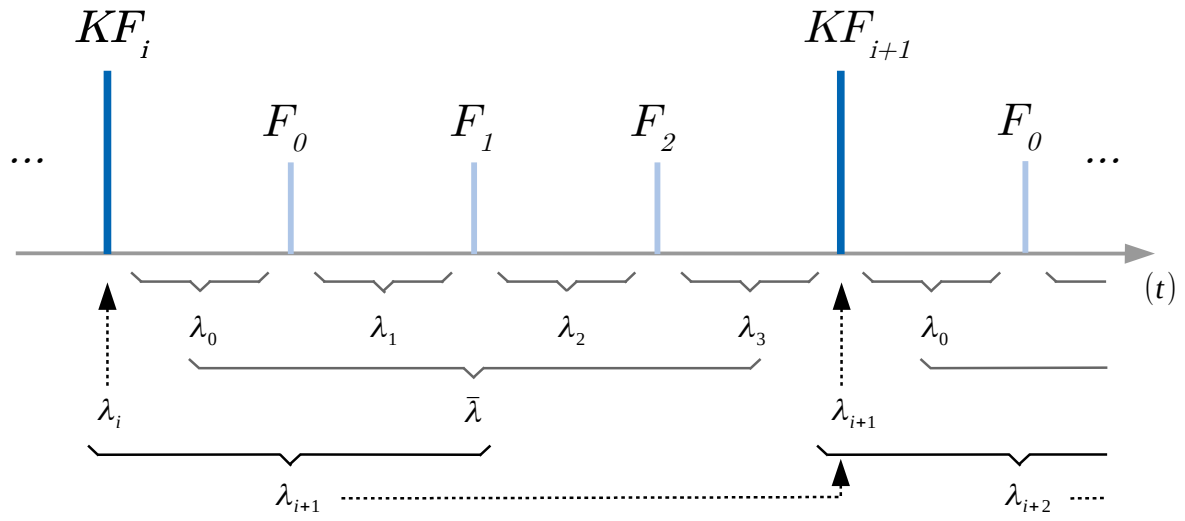


Figura 4.6: Representación visual del proceso de actualización de escala entre los *KeyFrames* i e $i+1$.

4.4. Odometría durante el estado de pérdida

En un sistema monocular, mientras este se encuentra en el estado de *pérdida*, no puede realizar ninguna de las tareas habituales del proceso de *Tracking* visual ni *Mapping*, debido a que ambas dependen enteramente de la información visual. Con la incorporación del sensor inercial, podemos hacer uso de la odometría generada por la IMU mientras el algoritmo se encuentre en el estado de pérdida para seguir estimando la trayectoria de la cámara. La idea es permanecer en este estado hasta que la calidad visual vuelva a ser suficiente como para transitar al estado de *Tracking* visual.

Durante este estado, se estima la pose de cada nuevo fotograma que recibe el algoritmo de SD-SLAM con el modelo de movimiento inercial. Debido a que esta estimación no será refinada posteriormente por la información visual junto con el mapa almacenado, los procesos de corrección de la estimación inercial no se realizarán. Del mismo modo, la escala entre el sistema de referencia IMU y Mundo se mantendrá constante, haciendo uso de la última calculada.

Sin embargo, es necesario solventar el problema de que SD-SLAM únicamente almacena la pose de los fotogramas que considera relevantes (*KeyFrames*) y, durante el estado de pérdida no se crea ninguno. Esto es lógico ya que, a priori, no tiene sentido crearlos en un estado al cual se ha llegado por falta de información visual. Sin embargo, al realizar el seguimiento de la trayectoria con la ayuda de la odometría inercial es interesante que esta trayectoria quede almacenada en forma de *Keyframes*. En caso contrario, al momento de recuperar la calidad visual provocaría un salto espacial entre la última posición registrada y la futura posición creada al recuperarse. Además, si esta fuese almacenada, supondría una relación entre las poses de

entrada y salida del estado de pérdida, permitiendo que al detectar un cierre de bucle, toda la trayectoria inercial se viese beneficiada por la corrección realizada en este proceso. Por este motivo se ha añadido el concepto de *Fake-KeyFrame*.

Los *Fake-KeyFrames* sirven únicamente de contenedores para registrar la pose estimada por el modelo de movimiento inercial mientras el algoritmo se encuentre en el estado de pérdida. La figura 4.7 muestra una representación visual de este concepto, donde a la izquierda se puede observar un *KeyFrame* ordinario con la imagen asociada, y a la derecha un *Fake-KeyFrame* vacío, representado la ausencia de información más allá de la pose de la cámara. Los *Fake-KeyFrames* son ignorados por el módulo de *Tracking*, ya que no tiene sentido que sean procesados si no constan de información visual relacionada con el mapa. Para evitar saturar la lista de *KeyFrames*, se construye un *Fake-KeyFrame* por cada cinco fotogramas procesados.

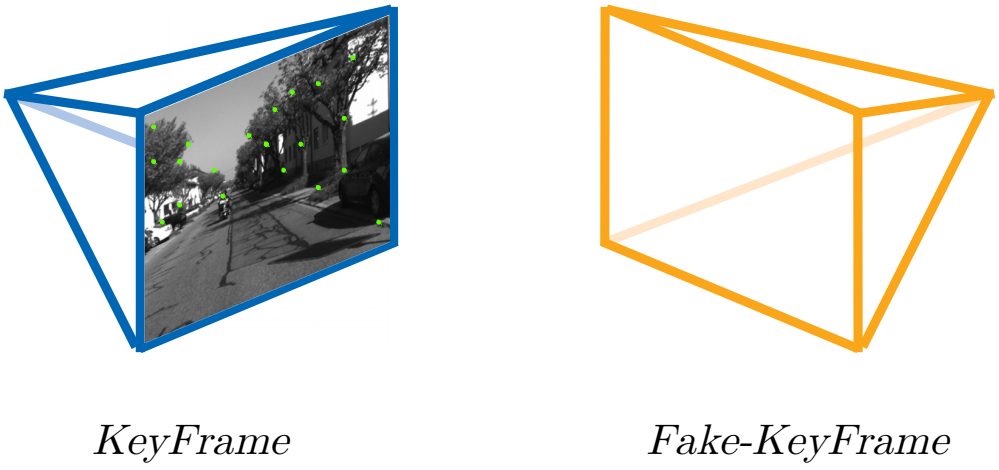


Figura 4.7: Representación visual del concepto de *KeyFrame* (izquierda) y *Fake-KeyFrame* (derecha).

Normalmente, la relación entre los *KeyFrames* se realiza en base a la información visual que comparten. Por este motivo, la relación entre los *Fake-KeyFrames* debe ser forzada, relacionado de forma consecutiva los unos con los otros. También es importante relacionar el último *KeyFrame* creado por visión antes de entrar en el estado de pérdida con el primer *Fake-KeyFrame*, así como del último *Fake-KeyFrame* con el primer *KeyFrame* del nuevo mapa, dado que son los puntos de enganche entre las estimaciones puramente visuales con estimaciones inerciales.

4.5. Restauración del funcionamiento puramente visual

Mientras que en los sistemas estéreo o RGB-D esta recuperación se puede hacer de forma instantánea debido a que la información de profundidad de las características detectadas es conocida, no es posible hacer lo mismo en los sistemas monoculares. Para realizar la recuperación

es necesario triangular la posición de las características haciendo uso de dos fotogramas con suficiente paralaje.

La calidad del mapa inicial y de la pose de los dos primeros *KeyFrames* en los algoritmos de monocular Visual SLAM es un factor crítico para las futuras características que se deseen añadir. Una mala estimación inicial conducirá a estimaciones erróneas tanto de poses como de características durante cierto periodo de tiempo hasta que el algoritmo pueda recuperarse (si es que llega a hacerlo). En consecuencia, el problema de la reinicialización se ha formulado para hacer uso de procedimiento similar al empleado para construir el mapa inicial en SD-SLAM. El proceso implementado consta de los siguientes pasos:

1. Estimar un nuevo mapa inicial.
2. Alinear las poses de los dos primeros *KeyFrames* y puntos del nuevo mapa con la trayectoria inercial.
3. Ajustar la escala del nuevo mapa para hacerla coincidir con el antiguo.
4. Comprobar la consistencia del nuevo mapa.
5. Integrar el nuevo mapa en el antiguo.

A continuación se detalla el proceso realizado en cada uno de los pasos. El primer paso es similar al realizado en la etapa de inicialización del algoritmo. Durante este proceso, como se explicó anteriormente, se trata de estimar la pose de los fotogramas iniciales como una transformación obtenida a partir de homografía o matriz fundamental, haciendo uso de los emparejamientos obtenidos por ORB. La pose del primer *KeyFrame* en el nuevo mapa se localiza centrada en el origen del sistema de referencia Mundo y los puntos del nuevo mapa son referenciados a este *KeyFrame*.

Debido a que durante el estado de pérdida se ha estimado la trayectoria realizada por odometría inercial, el segundo paso es el alineamiento de los dos nuevos *KeyFrames* y del nuevo mapa con esta trayectoria. La Figura 4.8 representa visualmente este proceso (omitiendo los puntos del mapa y el sistema de referencia de cada pose por comodidad). La transformación a aplicar en cada elemento es la siguiente:

- **Primer *KeyFrame***. Para combinar las trayectorias es necesario que este fotograma tenga asociada la misma pose que la última obtenida por odometría inercial. Teniendo en cuenta que el nuevo primer *KeyFrame* se encuentra localizado en el origen, simplemente se asigna la pose estimada inercialmente para dicho fotograma:

$$P_0^C = P_{I,0}^C \tag{4.16}$$

donde P_0^C es la nueva pose para el primer *KeyFrame* y $P_{I,0}^C$ es la pose obtenida por odometría inercial para el primer *KeyFrame*, ambas en el sistema de referencia Cámara.

- **Segundo *KeyFrame***. A la pose anteriormente calculada hay que añadirle el desplazamiento entre realizado entre los dos *KeyFrames*. Como el primer *keyFrame* se encontraba localizado en el origen, la pose del segundo *KeyFrame* representa el proprio desplazamiento realizado entre los dos fotogramas. La aplicación de la transformación sobre una pose es calculada como:

$$P_1^C = P_{V,1}^C \, P_0^C \tag{4.17}$$

donde P_1^C es la nueva pose para el primer *KeyFrame* en el sistema de referencia Cámara y $P_{V,1}^C$ es la pose obtenida en el proceso de estimación del mapa inicial para el segundo *KeyFrame*. Tras esta operación ambas trayectorias están alineadas, sin embargo, no hay que olvidar que en este punto la escala de la trayectoria inercial y la escala asociada a los dos nuevos *KeyFrames* obtenidos por visión aún es distinta.

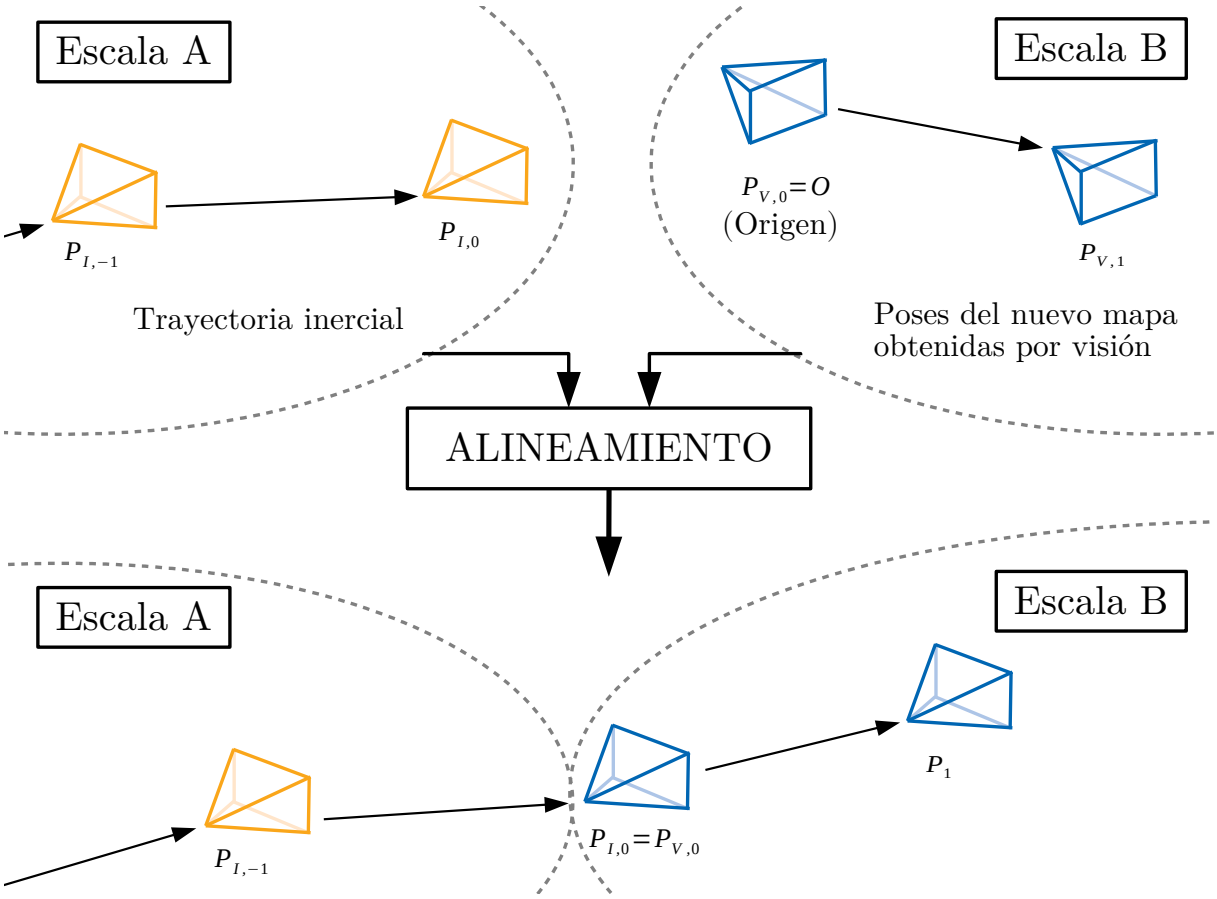


Figura 4.8: Representación visual del proceso de alineamiento entre la trayectoria inercial (formada por *Fake-FeyFrames*) y la pose de los nuevos *KeyFrames* estimados por visión.

- **Puntos del mapa**. Los puntos 3D son expresados en el sistema de referencia Mundo y son referenciados a la pose de la cámara que los detectó por primera vez, en este caso el

primer *KeyFrame*. Por este motivo, cada uno de los puntos p del mapa recibe la misma transformación que el primer *KeyFrame*, es decir, $P_{I,0}$. Para este proceso se hace uso de las componentes de la transformación (matriz de rotación $R^W_{I,0}$ y traslación $t^W_{I,0}$) en el sistema de referencia Mundo, dado que los puntos se encuentran en este sistema. La nueva posición de cada punto se calcula del siguiente modo:

$$p_i'^W = R^W_{I,0}\, p_i^W + t^W_{I,0} \tag{4.18}$$

donde p_i^W representan la posición original del punto y $p_i'^W$ la nueva para cada punto i.

Tras alinear las trayectorias y puntos del nuevo mapa, el tercer paso a realizar es un ajuste entre la escala del nuevo mapa y el antiguo. Al tratarse de un proceso de inicialización ordinario de un sistema monocular, la escala del nuevo mapa es arbitraria y no tiene porqué coincidir con la escala del mapa antiguo. No obstante, ambos mapas están relacionados con el mundo inercial, y por tanto, se puede corregir la escala del nuevo mapa. El objetivo es modificar la posición del segundo *KeyFrame* para que el desplazamiento realizado entre ambos *KeyFrames* coincida con la escala del mapa antiguo (que denominaremos A). La Figura 4.9 representa visualmente este proceso.

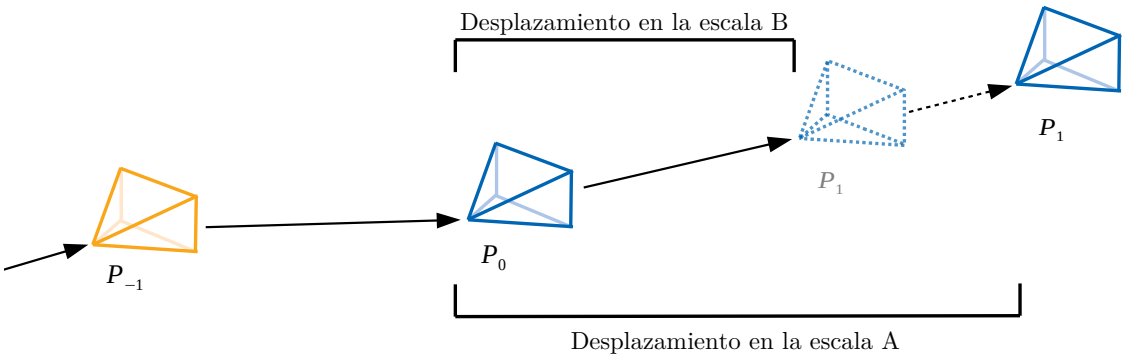


Figura 4.9: Representación visual del proceso de escalado del desplazamiento entre los nuevos *KeyFrames* (escala B) para hacerlos coincidir con la escala del mapa antiguo (escala A).

En primer lugar, se estima el factor de escala que relaciona el nuevo mapa (que llamaremos B) con el sistema de referencia de la IMU empleando la ecuación 4.14 obteniendo el factor λ^B. El desplazamiento espacial llevado a cabo entre los fotogramas se transforma a la escala del sistema de referencia de la IMU aplicando $1/\lambda^B$. En este punto se puede hacer uso del último factor de escala estimado en el mapa antiguo (λ^A) para finalmente transformar el desplazamiento a la escala antigua. Este proceso se puede expresar del siguiente modo:

$$s_1^{W(A)} = s_0^W + \frac{\lambda_A}{\lambda_B}\Delta s^{W(B)} \tag{4.19}$$

donde $s_1^{W_{(A)}}$ es la nueva posición para el segundo *KeyFrame* en coordenadas de Mundo A, s_0^W es la posición del primer *KeyFrame* del nuevo mapa, $\Delta s^{W_{(B)}}$ es el desplazamiento espacial entre los dos *KeyFrames* del nuevo mapa en coordenadas Mundo B.

El mismo proceso puede llevarse a cabo para escalar cada uno de los puntos del mapa. En este caso la ecuación se reformula del siguiente modo:

$$p_i^{W_{(A)}} = s_0^W + \frac{\lambda_A}{\lambda_B}(p_i^{W_{(B)}} - s_0^W) \tag{4.20}$$

donde $p_i^{W_{(B)}} - s_0^W$ es el vector formado por el punto i y la pose de la cámara en el primer *KeyFrame*.

Realmente, en este punto ya se podría realizar la fusión del nuevo mapa con el antiguo, sin embargo, como se verá en el capítulo de Validación experimental, el proceso de la estimación del mapa inicial de SD-SLAM no está exento de estimaciones erróneas. A causa de este motivo, la cuarta fase de este proceso consiste en comprobar la consistencia del nuevo mapa. Este proceso se basa en la idea de que el desplazamiento realizado entre los dos primeros *KeyFrames* estimado por odometría inercial debería ser similar (por lo menos en dirección) a las estimaciones de pose calculadas por visión. Siguiendo esta idea, en este paso comprueba el ángulo (θ) formado por los vectores de desplazamiento de la odometría inercial y odometría visual (calculadas en el apartado anterior) entre los dos primeros *Keyframes*. Esto puede verse de forma visual en la figura 4.10. Si el ángulo es mayor a cierto umbral, se asume que las estimaciones de pose realizadas por visión son erróneas y, en consecuencia, el mapa se descarta, quedando a la espera de recibir nuevos fotogramas para comenzar de nuevo el proceso de reinicialización.

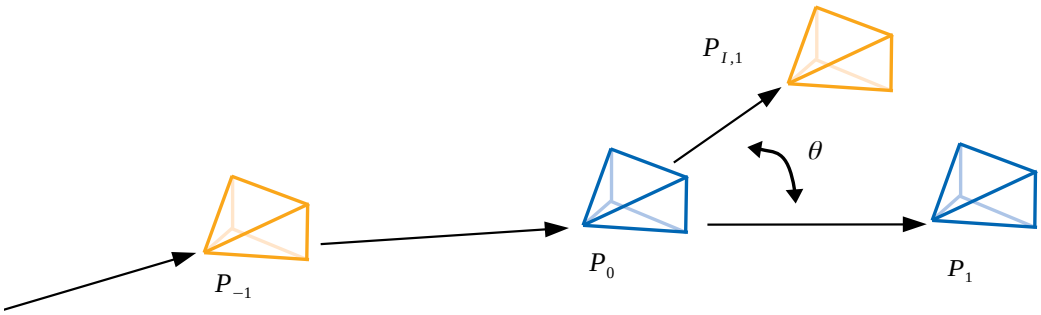


Figura 4.10: Representación visual del angulo formado entre la pose estimada para los dos primeros *KeyFrames* del nuevo mapa y las estimaciones inerciales para esos mismos fotogramas con el objetivo de comprobar la consistencia del nuevo mapa.

Finalmente, en este punto ya se encuentran los nuevos *KeyFrames* y puntos del mapa transformados para coincidir con la trayectoria inercial y escalados al tamaño del mapa en el mo-

mento de entrar en pérdida. Sin embargo, solo se desea trabajar con un mapa en memoria por lo que toda la información del nuevo mapa se añade al antiguo. Finalmente, se relaciona el último *Fake-KeyFrame* creado por la trayectoria inercial con el primer *KeyFrame* de este nuevo mapa. Por último, el algoritmo VIO-SDSLAM transita al estado de *Tracking* visual.

Capítulo 5

Validación experimental

En este capítulo se describen los experimentos realizados para evaluar el rendimiento y la calidad de las distintas ampliaciones desarrolladas en este TFM presentes VIO SD-SLAM comparándolo con SD-SLAM. Para ello también se introduce el conjunto de datos seleccionado para la evaluación así como las métricas empleadas.

El objetivo de este capítulo es validar experimentalmente que las ampliaciones realizadas en VIO SD-SLAM se han diseñado e implementado correctamente, así como evaluar los resultados obtenidos por SD-SLAM y VIO SD-SLAM en distintas situaciones.

Debido a que el modelo de odometría inercial planteado en VIO SD-SLAM no tiene en consideración el ruido asociado al acelerómetro, esto puede afectar a los resultados obtenidos. Por este motivo, algunos de los experimentos se realizarán empleando tres acelerómetros diferentes:

- ***Pseudo*-Ideal**. Acelerómetro obtenido a partir de la segunda derivada de la posición estimada por el GPS del *dataset*. Es considerado ideal puesto que no contendrá ningún tipo de ruido, pero solamente es capaz de recuperar la posición original mientras se conozca la velocidad inicial ideal antes de cada estimación; por este motivo se ha añadido el prefijo *Pseudo*.

- **Ruidoso**. Acelerómetro obtenido al añadir ruido aleatorio sobre las medidas del acelerómetro Pseudo-Ideal. Este sensor pretende representar uno más real que el anterior, dado que en la realidad siempre se medirá cierta componente de ruido en las medidas del sensor por muy buena estimación de este que se realice. El ruido aleatorio es modelado como una *Gaussiana* de media 0 y desviación σ_a. El valor de σ_a ha sido fijado en $0{,}05$.

- **Real**. Acelerómetro proporcionado por el *dataset KITTI* que consta del ruido blanco, *bias* no nulo y es afectado por la fuerza de la gravedad.

En los siguientes apartados se describen los experimentos realizados. Cada uno de ellos enfocado de forma individual en una de las ampliaciones implementadas sobre sobre el algoritmo

original de SD-SLAM.

5.1. Conjunto de datos para evaluación de algoritmos de Visual SLAM

El conjunto de datos (*dataset*) para evaluar las soluciones implementadas debe de constar de una serie de elementos imprescindibles, tales como un sistema de cámaras para proporcionar información visual, un sensor inercial y la secuencia verdadera de posiciones seguida por la cámara (*Groundtruth*) para comparar con la estimación realizada por SD-SLAM.

Un conjunto de datos interesante es el "benchmark suite" KITTI [15, 14] desarrollado de forma conjunta por el Instituto Tecnológico de Karlsruhe y el Instituto Tecnológico de Toyota. En comparación con otros conjuntos de datos, KITTI ofrece datos más realistas puesto que las secuencias están grabadas en un entorno real en lugar de ser realizados en el interior de un laboratorio.

Las secuencias están divididas en distintas categorías, siendo la de odometría la que proporciona todas las características que necesitamos. En esta categoría los datos son tomados desde un vehículo dotado de un par estéreo de cámaras, un sensor LIDAR (del que no se hace uso en el desarrollo de este proyecto) y un sistema de navegación GPS (OXTS RT 3003) que contiene un sensor inercial. En la Figura 5.1 se puede observar la disposición de los distintos sensores. Las imágenes son tomadas a una resolución de 1392x512 píxeles a una frecuencia de 10Hz. Lamentablemente no se dispone de las especificaciones del sensor inercial, ya que este sirve de apoyo al sistema principal GPS y, por tanto, los datos proporcionados por el fabricante hacen referencia a la precisión de este último. Además de los anteriores datos, KITTI proporciona los parámetros intrínsecos y extrínsecos de cada cámara, así como la matriz de transformación rígida entre los distintos sistemas de referencia de cada sensor. Por último, mencionar que las secuencias de las que se hace uso se encuentran sincronizadas y, por tanto, únicamente dispondremos de una medida del sensor inercial por cada fotograma, pese a que este funcione a más Hz que la cámara.

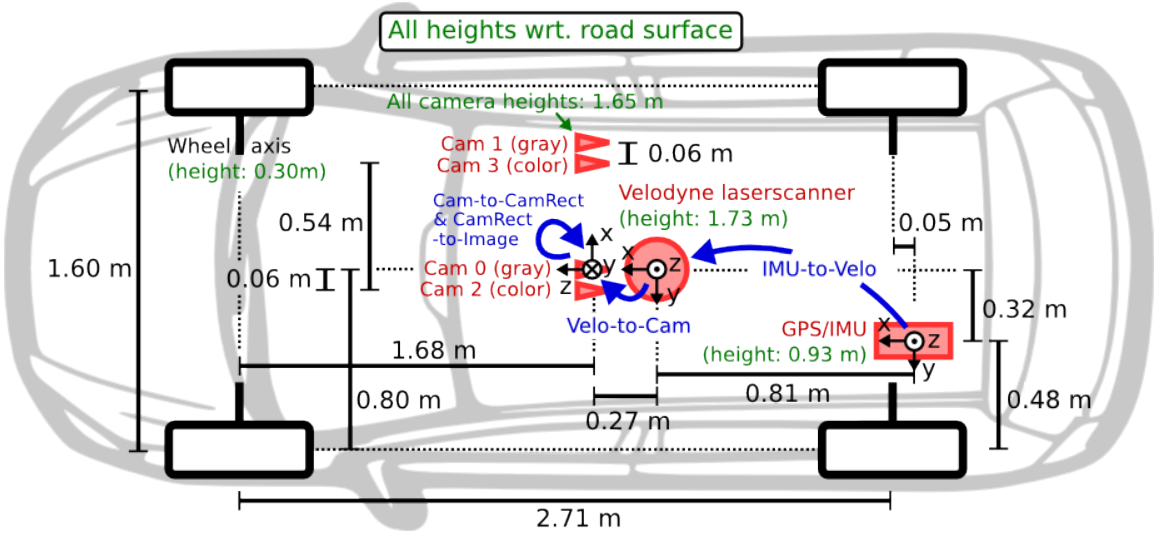
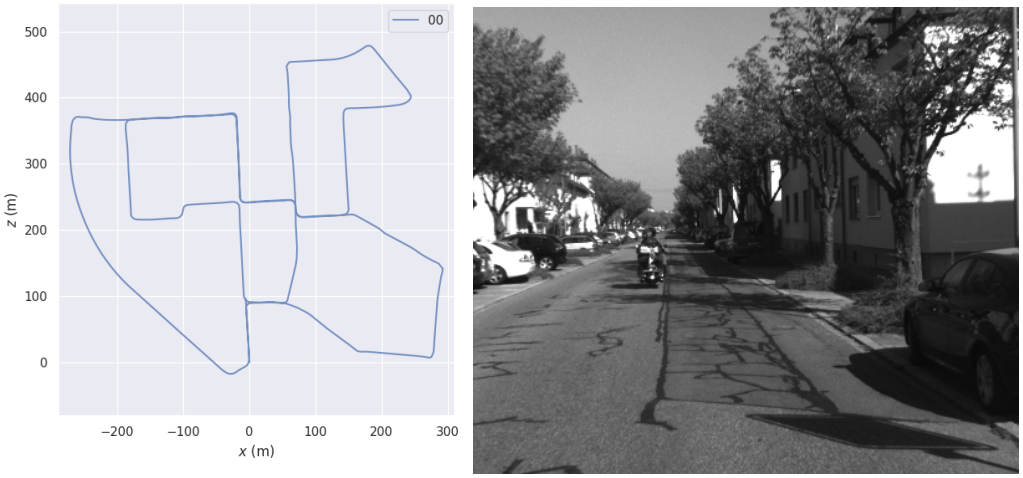


Figura 5.1: Configuración sensorial del vehículo de KITTI.

El conjunto de datos consta de 9 secuencias donde la mayoría de ellas se desarrollan en un entorno residencial. Esto quiere decir que consta elementos estáticos como edificios, vegetación o vehículos estacionados a cada lado de la calzada; y elementos dinámicos como vehículos en circulación o peatones. Debido a la larga duración de la mayoría de las secuencias se han seleccionado un subconjunto que permita cubrir una variedad de escenarios. A continuación se describen muy brevemente las secuencias seleccionadas para la validación experimental del algoritmo de Visual SLAM que se propone en este TFM:

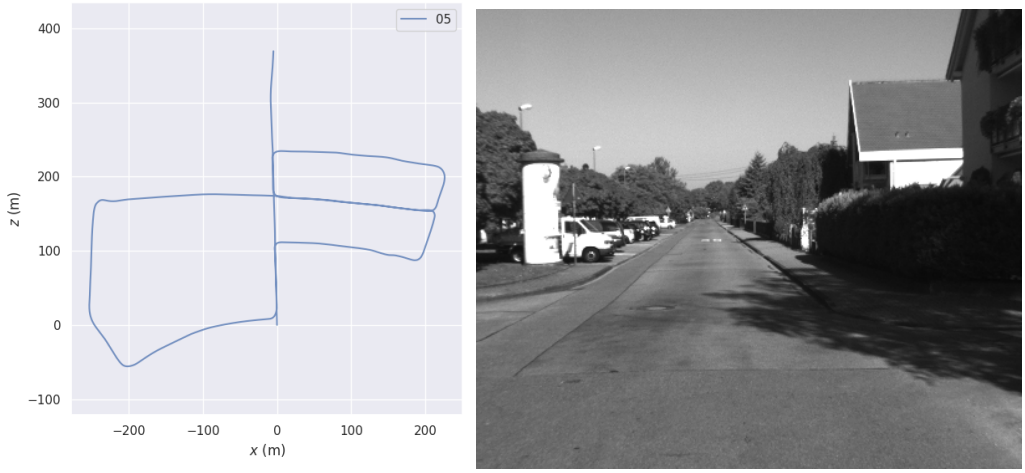
- **Secuencia 00.** Secuencia con una duración de de 7:50 minutos y un total de 4 cierres de bucle. Permitirá evaluar la robustez del algoritmo en un largo periodo de tiempo.



(a) (b)

Figura 5.2: Trayectoria realizada (a) en la secuencia 00 perteneciente al conjunto de datos Kitti Odometría y un fragmento de la imagen del entorno capturada por la cámara (b).

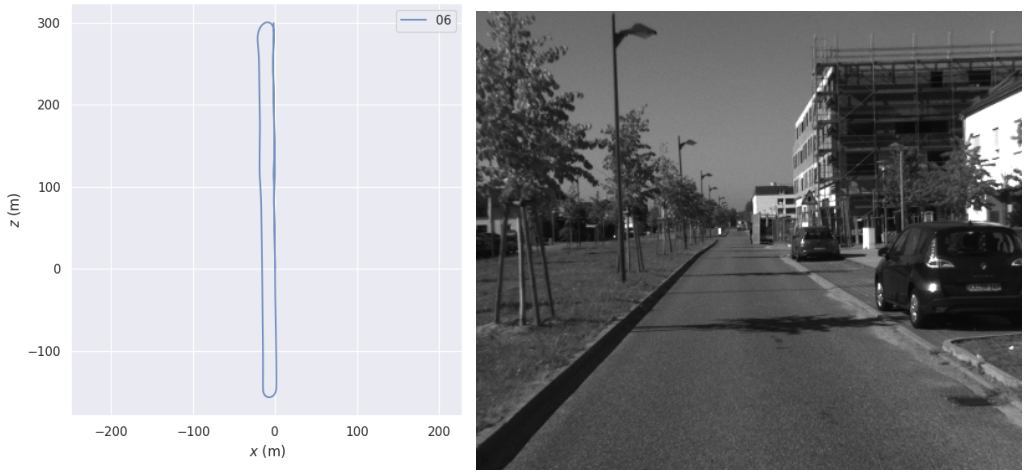
- **Secuencia 05.** Similar al caso anterior, esta secuencia consta de una duración de 5 minutos con un total de 2 cierres de bucle.



(a) (b)

Figura 5.3: Trayectoria realizada (a) en la secuencia 05 perteneciente al conjunto de datos Kitti Odometría y un fragmento de la imagen del entorno capturada por la cámara (b).

- **Secuencia 06.** Trayectoria sencilla que consta de 2 rectas y dos curvas cerradas de 90º que se toman a gran velocidad. Tiene una duración de 1:30 minutos y un único cierre de bucle.



(a) (b)

Figura 5.4: Trayectoria realizada (a) en la secuencia 06 perteneciente al conjunto de datos Kitti Odometría y un fragmento de la imagen del entorno capturada por la cámara (b).

- **Secuencia 07.** Secuencia con una duración de 1:30 minutos. Consta de un cierre de bucle que une el principio y el final de toda la trayectoria y que, en la mayoría de ocasiones, no es detectado por SD-SLAM.

(a) (b)

Figura 5.5: Trayectoria realizada (a) en la secuencia 07 perteneciente al conjunto de datos Kitti Odometría y un fragmento de la imagen del entorno capturada por la cámara (b).

5.2. Métricas de evaluación de algoritmos SLAM

Existen diferentes métricas para evaluar la calidad de los algoritmos de SLAM, algunas centradas en la calidad del mapa generado y otras en la trayectoria seguida por la cámara. En nuestro caso nos centraremos en estas últimas, concretamente en el error absoluto (ATE) y relativo (RPE) de las trayectorias [36]. Estas métricas requieren de la secuencia verdadera de referencia junto a la estimada por el algoritmo de SLAM que se pretende evaluar.

El error absoluto representa la consistencia global de una trayectoria. Es calculado como la distancia geométrica entre cada una de la posiciones de ambas trayectorias. En el cálculo del error absoluto no se emplea la orientación, dado que se asume que el error producido en esta se verá reflejado en la información de la posición espacial en los instantes siguientes.

Normalmente no es posible comparar directamente las secuencias, dado que cada secuencia consta de sus propios sellos temporales y sistemas de referencia. Por tanto, en primero lugar, es necesario alinearlas temporalmente para obtener los pares de posiciones a evaluar. Posteriormente, en caso de ser necesario, se realiza una transformación rígida para corregir la diferencia en el sistema de referencia aplicando una rotación y traslación; esta transformación suele representarse como SE(3). Además, en los sistemas monoculares, dado que su escala es arbitraria, es necesario realizar una corrección en esta mediante una matriz de similitud; representada como Sim(3).

En lugar de realizar una comparación directa entre las posiciones espaciales, el error relativo compara el movimiento (incremento) realizado de una posición a la siguiente. Esta métrica otorga información sobre la precisión local, siendo especialmente útil para calcular la deriva espacial o una estimación del error en orientación. En Visual SLAM este error suele ser calculado como el movimiento entre imágenes o *KeyFrames* consecutivos.

Para la obtención de estos valores en los experimentos se empleará la herramienta EVO [17], que proporciona una serie de ejecutables para manipular, evaluar y comparar la trayectoria obtenida por odometría y algoritmos de SLAM. Es capaz de estimar las transformación SE(3) y Sim(3) para alinear las trayectorias empleando el algoritmo de Umeyama [38].

5.3. Experimentos de Odometría inercial

Los experimentos que se presentan en esta sección pretenden evaluar las estimaciones de pose (posición y orientación) obtenidas empleando la odometría inercial, sin el uso de visión. Es importante conocer el comportamiento de la odometría inercial trabajando de forma individual puesto que se hace uso de ella tanto para estimar la escala inicial como para mantener la trayectoria cuando visión entre en el estado de pérdida.

Para realizar la evaluación se ha estimado la pose durante el primer minuto de cada una

de las secuencias del *dataset* haciendo uso únicamente de la información proporcionada por la IMU. Se realiza una estimación por cada medida leída del sensor. Como en el *dataset* se recibe una medida por cada fotograma sería equivalente a hacer una estimación por cada fotograma recibido por VIO SD-SLAM.

5.3.1. Error en orientación

En primer lugar, se analiza la estimación de orientación, dado que los errores en estas estimaciones repercutirán directamente en la estimación de posición. El valor del factor de ganancia β del filtro Madgwick ha sido configurado a $0{,}0085$ de manera experimental, y es constante a lo largo de todos los experimentos y secuencias. Este valor se ha obtenido tras probar con distintos valores en las secuencias del *dataset* KITTI y observar que otorgaba una buena respuesta ante giros dinámicos sin añadir excesivo ruido entre estimaciones. La Tabla 5.1 representa el error incremental (RPE) en la orientación expresado en grados entre cada par de estimaciones realizadas por la odometría inercial respecto a la secuencia de referencia. Para obtener los resultados se ha empleado únicamente el acelerómetro real, dado que apenas existen diferencias al estimar la orientación con los tres acelerómetros disponibles (real, ruidoso y pseudo-ideal).

	RMSE	**MIN**	**MAX**
Secuencia 00	0.17	0.02	0.32
Secuencia 05	0.12	0.01	0.32
Secuencia 06	0.12	0.02	0.33
Secuencia 07	0.13	0.03	0.31
Media	0.13	0.02	0.32

Tabla 5.1: RPE incremental en orientación expresado en grados de las estimaciones de odometría inercial durante 1 minuto.

Los resultados en todas las secuencias muestran un error en la estimación de la orientación bastante bajo, siendo la media $0{,}13°$. Debido a que el parámetro de ganancia del filtro Madgwick se ha establecido con un valor cercano a 0 provoca que los valores máximos de error se produzcan principalmente en las curvas, ya que el tiempo de respuesta es ligeramente inferior en beneficio de filtrar la mayor parte del ruido del giroscopio. Aún así, los valores máximos se mantienen bajos, siendo la media $0{,}32°$. Con estos resultados se puede afirmar que la estimación de orientación en el tiempo es robusta incluso bajo largos periodos de tiempo, degradándose lentamente. Los buenos resultados obtenidos pueden deberse a la calidad en sí del giroscopio que incorpora el sensor. Además, el proceso de estimación orientación acumula menos error que en

la estimación de posición, como se verá más adelante. Esto es debido a que solo hay que realizar dos pasos: estimación del ruido de la medida del sensor y realizar una única integración de la velocidad angular.

5.3.2. Error en posición

Por otra parte, para analizar el comportamiento de las predicciones de la posición se emplea el error absoluto en la trayectoria (ATE). En esta ocasión se emplea esta métrica ya que a la hora de emplear la odometría inercial en VIO SD-SLAM el error en esta estimación condicionará enormemente los resultados. La posición es dependiente del acelerómetro y, por este motivo, se realiza la prueba empleando cada uno de los acelerómetros disponibles. La trayectoria estimada por cada acelerómetro puede observarse en la Figura 5.6, y a partir de esta trayectoria se ha construido la Tabla 5.2 con el RMSE del error absoluto en la trayectoria en metros entre la secuencia de referencia *GT*, la verdadera, y la trayectoria estimada.

	Real	**Ruidoso**	***Pseudo*-Ideal**
Secuencia 00	72.6961	5.0766	0.3465
Secuencia 05	21.2380	4.9796	0.3637
Secuencia 06	43.2423	4.7222	0.5703
Secuencia 07	14.3084	4.8484	0.3711
Media	37.8712	4.9067	0.4129

Tabla 5.2: ATE (RMSE) en posición expresado en metros de las estimaciones de odometría inercial durante 1 minuto.

Como era de esperar, los resultados mejoran a medida que aumenta la precisión de los datos proporcionados por el acelerómetro. Por una parte, cuando se emplea el acelerómetro pseudo-ideal se obtiene una media de error inferior a 50 centímetros tras un minuto de ejecución, es decir, apenas hay deriva en el tiempo. Observando la trayectoria realizada (Figura 5.6) es posible apreciar cómo visualmente coincide con la trayectoria de referencia.

En el caso contrario se encuentra el acelerómetro real, donde la trayectoria se deteriora significativamente, provocando un error medio de casi 38 metros. En este caso se puede observar cómo la trayectoria descrita coincide con este valor de error, realizando estimaciones similares a la secuencia de referencia en los tramos iniciales pero desviándose a medida que pasa el tiempo. En ocasiones, como en la secuencia 00 (Figura 5.6a), se puede apreciar cómo es menos preciso en las curvas que en las rectas. Por ejemplo, la tercera curva se realiza muy abierta cuando en realidad es una curva cercana a los $90°$.

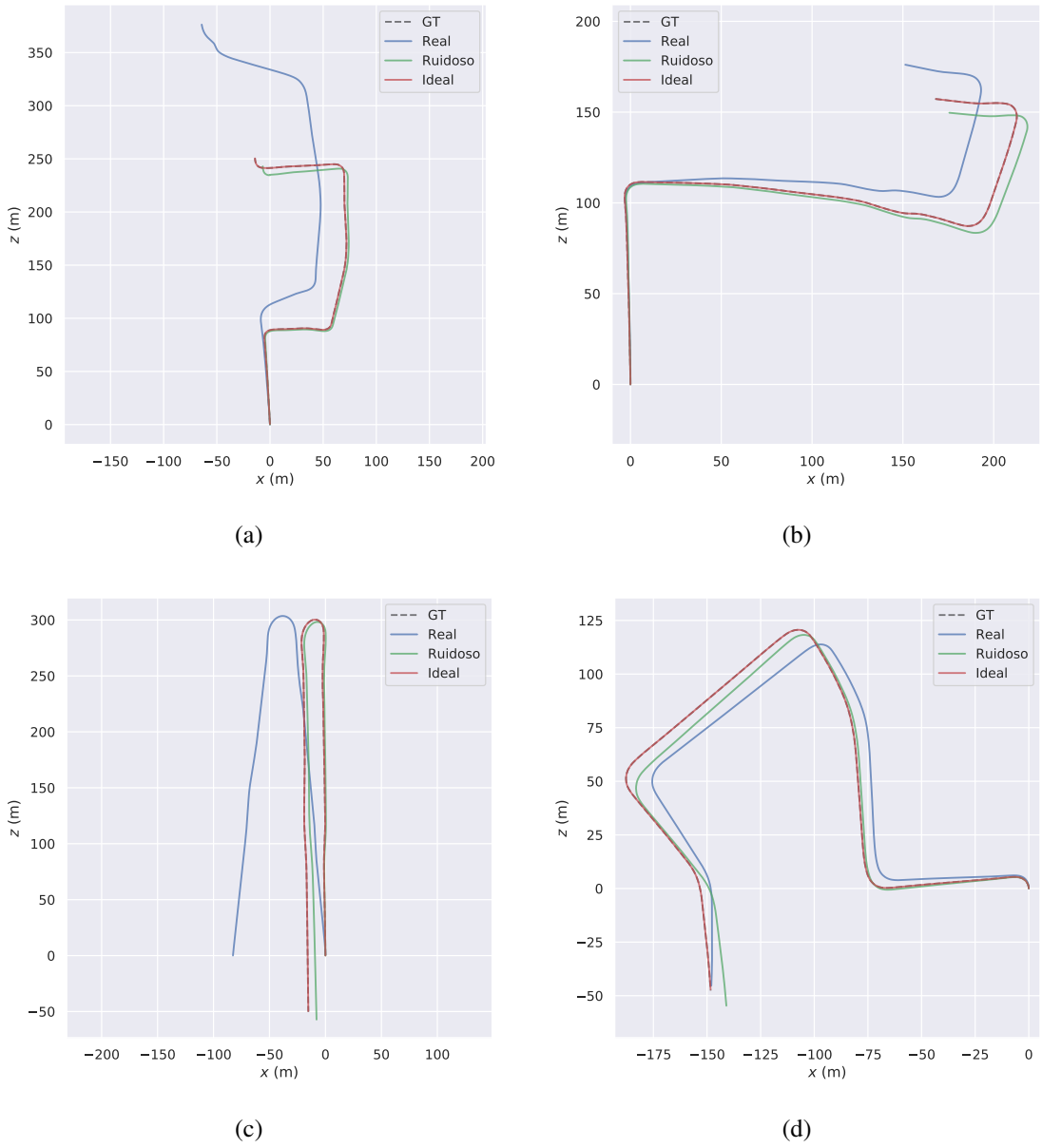
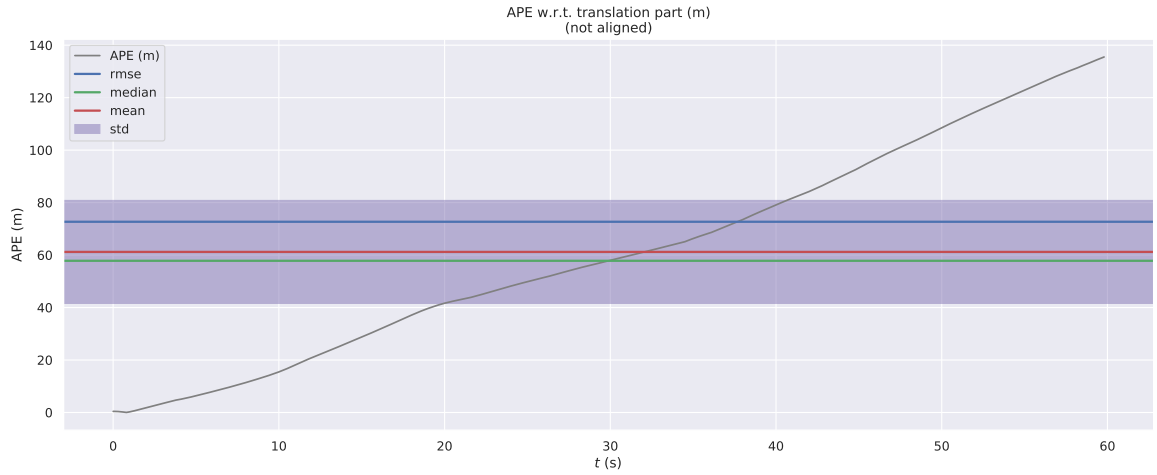


Figura 5.6: Trayectorias 2D (plano cenital) estimadas por odometría inercial. Por orden de enumeración: secuencia 00, 05, 06 y 07. La trayectoria real en cada secuencia está etiquetada como *GT*.

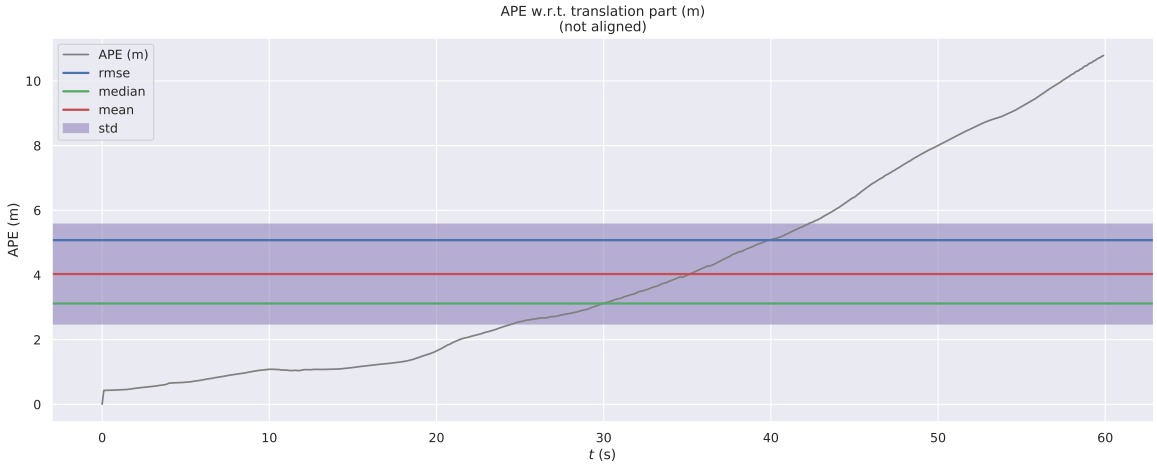
En un punto intermedio se encuentra el acelerómetro ruidoso, obteniendo unos resultados más cercanos al ideal que al real. La diferencia entre los resultados obtenidos por el acelerómetro ruidoso y real puede explicarse mediante el ruido *bias* presente en este último. El *bias* (sesgo), al fin y al cabo, es una pequeña diferencia (*offset*) entre la medición obtenida y la real, presente incluso cuando no hay movimiento. Dado que el modelo de movimiento inercial planteado no contempla este *offset*, provoca que en cada estimación realizada se añada un pequeño error fruto de la no-sustracción de este valor que se va acumulando a medida que pasa el tiempo,

provocando finalmente una deriva espacial.

Con el objetivo de analizar la evolución del error en función del tiempo se ha añadido la Figura 5.7 que representa la evolución del ATE haciendo uso del acelerómetro real (a) y ruidoso (b). Como puede observarse, el error tiende a aumentar, casi de forma lineal, a medida que avanza el tiempo. Sin embargo, haciendo uso del acelerómetro real la velocidad crecimiento del error es significativamente mayor por lo anteriormente expuesto. Este patrón se repite de igual modo en el resto de secuencias.



(a)



(b)

Figura 5.7: Evolución del ATE en función del tiempo en la Secuencia 00 haciendo uso del acelerómetro real (a) y ruidoso (b)

5.3.3. Conclusiones

A partir de todos los resultados expuestos en esta sección es posible llegar a una serie de conclusiones. En primer lugar, es factible hacer uso de la odometría puramente inercial para realizar estimaciones de pose. En segundo lugar, el tiempo del cual se puede hacer uso de la odometría inercial manteniendo el error por debajo de un umbral aceptable es dependiente de la calidad de los datos proporcionados por el sensor así como del modelo de movimiento implementado.

Por otra parte, de cara al funcionamiento de VIO SD-SLAM, es posible lanzar la odometría inercial de forma paralela al proceso de inicialización visual durante los primeros instantes de tiempo como se había planteado en el capítulo de diseño. Con los resultados obtenidos y el conocimiento de que el error en la estimación inercial crece a medida que avanza el tiempo, se puede deducir que cuanto antes se realice la primera estimación de escala entre la escena de visión y el mundo real más precisa será. Por último, cuando la odometría visual falle y se proceda a usar la odometría inercial, conviene permanecer el menor tiempo posible haciendo uso de esta para no degradar excesivamente la estimación de la trayectoria.

5.4. Experimentos de escala Visual-Inercial

Los experimentos que se presentan en esta sección pretenden estudiar la variación del factor de escala a lo largo del tiempo y evaluar la calidad de la estimación del factor de escala realizado por VIO SD-SLAM. Como se introdujo en el capítulo de diseño, este factor de escala es el encargado de relacionar la escala arbitraria generada por el sistema de visión monocular con la escala del mundo real y, por tanto, la calidad de VIO SD-SLAM dependerá en gran medida de una buena estimación de este factor.

Para calcular el error cometido en la estimación del factor de escala se necesita disponer de un factor de escala de referencia que llamaremos λ^{gt}. Este factor se calcula empleando las estimaciones de posición obtenidas por odometría visual y las posiciones de referencia verdaderas (*GroundTruth*) del mismo modo que se calcula la escala estimada, es decir:

$$\lambda_i^{gt} = \frac{\parallel s_{V,i} - s_{V,i-1} \parallel}{\parallel s_{gt,i} - s_{gt,i-1} \parallel} \tag{5.1}$$

donde $s_{V,i}$ y $s_{V,i-1}$ son la posición estimada por odometría visual para dos fotogramas consecutivos $i-1$ e i, y $s_{gt,i}$ y $s_{gt,i-1}$ son la posición de referencia para los mismos fotogramas. Realizando este proceso para cada par de estimaciones se obtiene su correspondiente factor de escala de referencia que representa la relación perfecta del movimiento realizado por visión para ese instante de tiempo con el movimiento esperado en el mundo real.

5.4.1. Análisis del factor de escala de referencia

Una vez se dispone de todos los factores de escala de referencia, en primer lugar, es necesario conocer cuánto varía a lo largo del tiempo. La Figura 5.8 representa el factor de escala de referencia para la secuencia 06. Se ha seleccionado esta secuencia ya que consta únicamente de dos tramos rectos y dos curvas cerradas, facilitando la identificación de cada una de estas situaciones en la figura para su posterior análisis.

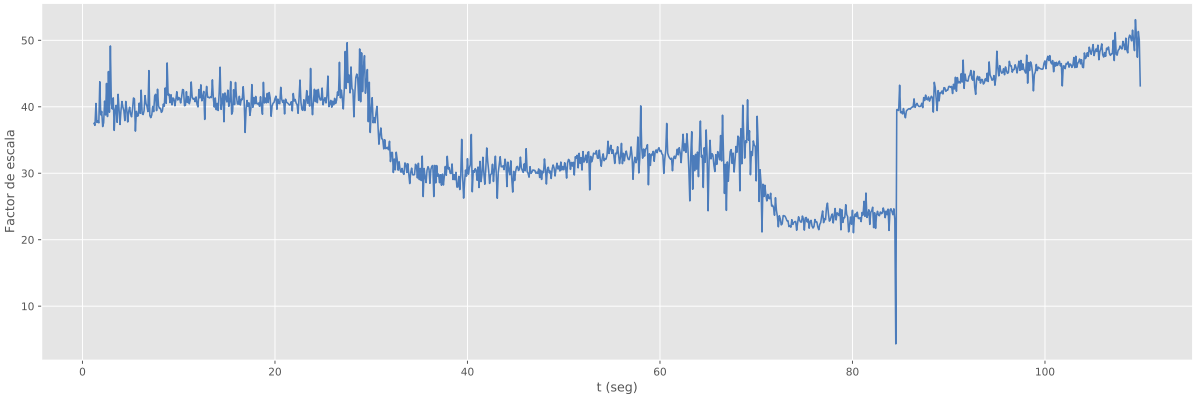


Figura 5.8: Evolución del factor de escala de referencia λ^{gt} a lo largo del tiempo en la secuencia 06.

Lo primero que se puede observar es que el factor de escala no es constante a lo largo del tiempo, variando lentamente en ocasiones y rápidamente en otras. Además, el factor presenta unas pequeñas variaciones de alta frecuencia entre valores consecutivos, dando como resultado una gráfica de aspecto ruidoso. Este efecto puede ser debido a que las estimaciones de visión no son perfectas y el desplazamiento estimado entre fotogramas consecutivos siempre es ligeramente mayor o menor que el realizado en la realidad, provocando que el factor de referencia fluctúe habitualmente.

Por otra parte, se pueden observar tres situaciones donde el factor de escala varía significativamente. Las dos primeras variaciones se producen alrededor del segundo 30 y 50, respectivamente, y corresponden a los tramos de curva de la trayectoria. Es decir, mientras que en los tramos de recta la escala varía lentamente, en los tramos de curva las variaciones suelen ser mayores. Esto puede ser causado por la menor presencia de puntos característicos debido a que la calidad visual de las imágenes desciende en estos tramos. Además, esta situación puede provocar que la estimación de posición tridimensional de los nuevos puntos característicos detectados y añadidos al mapa sea incorrecta. Todos estos factores pueden derivar en un cambio de la escala en la cual son representados los puntos del mapa. El patrón presentado en este ejemplo se repite de igual modo en el resto de secuencias, con mayor o menor énfasis.

La tercera situación de variación se produce en el segundo 85 aproximadamente y se ca-

racteriza por ser una variación brusca e instantánea de un instante del tiempo al siguiente. Este efecto es producido cuando se detecta un cierre de bucle, dado que en ese momento se recupera la escala estimada por primera vez.

5.4.2. Estimación del factor de escala realizado por VIO SD-SLAM

Una vez realizado un análisis sobre la variación del factor de escala de referencia, a continuación se procede a estudiar la calidad de la estimación de la escala realizada por VIO SD-SLAM. En el diseño propuesto en la sección 4.3.2 para la estimación y actualización de la escala por VIO SD-SLAM se introdujo el parámetro α que controlaba la velocidad de variación de la escala estimada. Este parámetro puede tomar cualquier valor real en el intervalo cerrado $[0, 1]$, donde el valor 0 representa un factor de escala constante (el primer factor estimado), el valor 1 la variación total ignorando el factor previo, y cualquier valor intermedio del intervalo representa una combinación entre el antiguo factor de escala y nuevo estimado.

Para realizar las pruebas se seleccionan una serie de valores del parámetro α para observar el efecto que producen diferentes configuraciones. Los extremos del intervalo se han descartado tras haber analizado la variación del factor de escala de referencia en la Figura 5.8. El valor 0 no tiene sentido seleccionarlo puesto que la escala no es constante a lo largo del tiempo; el valor 1 ha sido descartado debido a la rápida y continua variación del factor de escala, lo que podría traducirse en una estimación ruidosa. Por estos motivos todos los valores seleccionados representar una combinación entre la escala antigua y la nueva. Tratando de cubrir de forma aproximada el espectro de posibles valores de α se han seleccionado para los experimentos los siguientes: $0{,}25$, $0{,}50$ y $0{,}75$.

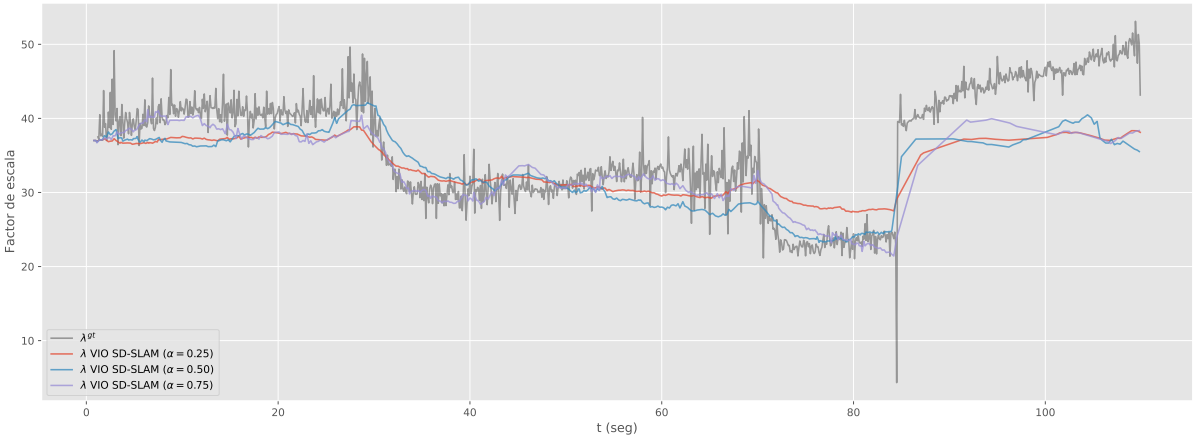


Figura 5.9: Comparativa entre el factor de escala de referencia λ^{gt} respecto al estimado por VIO SD-SLAM empleando distintos valores de α.

La figura 5.9 representa el factor de escala estimado por VIO SD-SLAM a lo largo de la

secuencia 06 empleando las tres configuraciones junto con el de referencia. Como se puede observar, el factor de escala inicial en un primer momento toma un valor cercano al inicial de referencia. Sin embargo, a medida que avanza el tiempo algunas estimaciones se aproximan mejor que otras al valor de referencia. Las configuraciones con los valores de α igual a $0{,}50$ y $0{,}75$ se aproximan bastante bien hasta el cierre de bucle, donde recupera los valores estimados la primera vez que recorrió el tramo, mientras que la secuencia de referencia continúa estimando nuevos valores. La configuración que emplea $\alpha = 0{,}25$ presenta una respuesta más lenta y por tanto no termina de ajustar correctamente cuando se produce un cambio brusco.

Los resultados obtenidos en el proceso de estimación del factor de escala por VIO SD-SLAM, si bien son aceptables, no son perfectos. Esto puede explicarse debido a que las estimaciones de odometría inercial están condicionadas a una serie de factores que añaden de forma acumulativa error al sistema inercial:

1. La calidad de las medidas realizadas por la IMU determina la estimación de aceleración lineal que se empleará tanto para la estimación de velocidad como de posición.

2. Las estimaciones de posición realizadas mediante odometría inercial tienen una fuerte dependencia de la velocidad inicial previa a la estimación. Debido a la deriva que sufre el sistema inercial a lo largo del tiempo, el diseño implementado en este proyecto presenta una solución basada en la información de las poses de visión para corregir, entre otros, el error de la velocidad. Sin embargo, como también se ha podido comprobar anteriormente, las estimaciones de visión no siempre son perfectas (entendiendo por perfecta que se ha estimado un movimiento acorde al verdadero realizado por la cámara en el mundo real) y esto puede derivar en algunas ocasiones en una estimación de velocidad incorrecta.

3. El valor de velocidad obtenido se encuentra en el sistema de referencia de Visión, y es necesario escalarla al mundo inercial mediante el factor de escala estimado para ese instante de tiempo. Es decir, el propio factor de escala condiciona la estimación del siguiente factor de escala.

La acumulación de los tres factores es lo que provoca que la estimación del factor de escala no sea perfecta. Además, hay que contar con un error más que ocurre cuando la inicialización del mapa por parte de visión es incorrecta, ya que pese a ser un proceso robusto no siempre es óptimo. La Figura 5.10 es una captura de la interfaz de VIO SD-SLAM en los primeros segundos de la secuencia 07, y representa en verde y negro los puntos del mapa, en azul los *KeyFrames* generados, y en rojo la posición actual de la cámara. La secuencia 07 no tiene prácticamente variación en altura pero, como se puede observar, la posición de los primeros *KeyFrames* no se corresponde con la realidad. El error en la estimación de la posición de los primeros puntos 3D del mapa y de la pose de los dos primeros *KeyFrames* provoca este efecto,

que deriva en una variación rápida en la posición y orientación hasta que finalmente es capaz de recuperarse del error inicial.

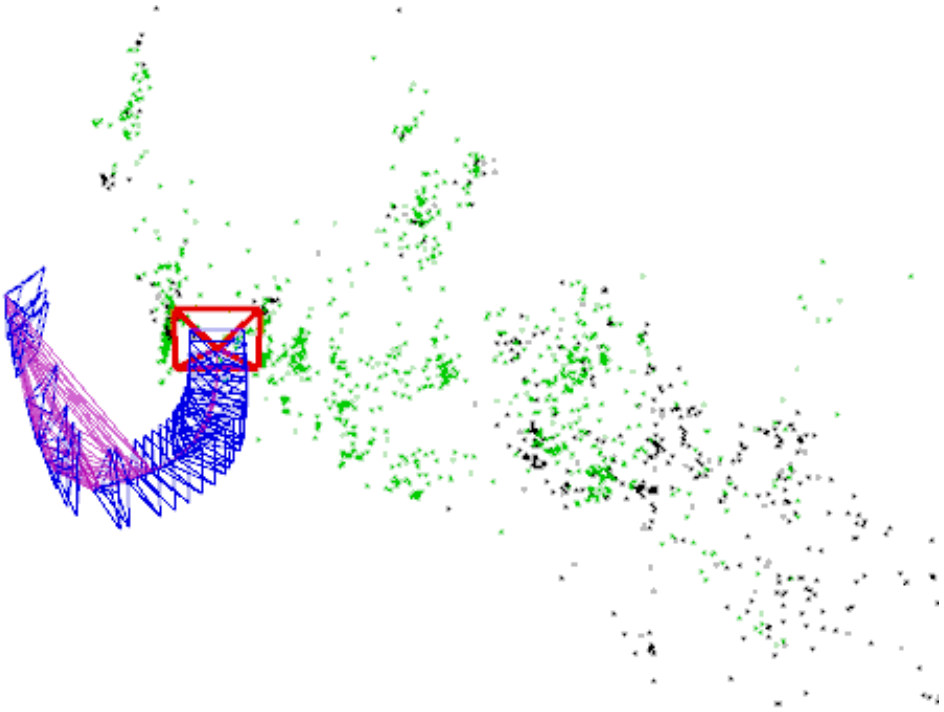


Figura 5.10: Inicialización incorrecta de SD-SLAM en la secuencia 07.

Este error inicial tiene una fuerte repercusión en el proceso de estimación del factor de escala, dado que el movimiento realizado entre los primeros *KeyFrames* no representa el movimiento real realizado por la cámara sino más bien representa ajustes necesarios para hacer coincidir el mapa 3D con la observaciones realizadas con la llegada de cada nuevo fotograma. Todo ello se traduce en correcciones erróneas sobre el modelo de movimiento inercial, provocando que la escala estimada no coincida con la esperada. La Figura 5.11 representa la estimación de escala para los primeros segundos de la secuencia 07. Como se puede observar, la primera estimación es similar al factor de referencia, sin embargo, la escala de referencia varía durante los 4 primeros segundos rápidamente, sin embargo, las estimaciones del factor de escala, condicionadas por la primera estimación, no son capaces de adecuarse a este suceso.

5.4.3. Reducción del error al introducir el ajuste de escala

Una vez introducida de forma visual el comportamiento de la estimación de la escala por VIO SD-SLAM, a continuación se estima de forma cuantitativa el error cometido. Como el factor de escala afecta al movimiento realizado entre dos instantes de tiempo, la métrica más adecuada para su evaluación es el error incremental entre posiciones, es decir, el RPE. Con esta metodología se escala el movimiento realizado entre *KeyFrames* consecutivos con el factor de

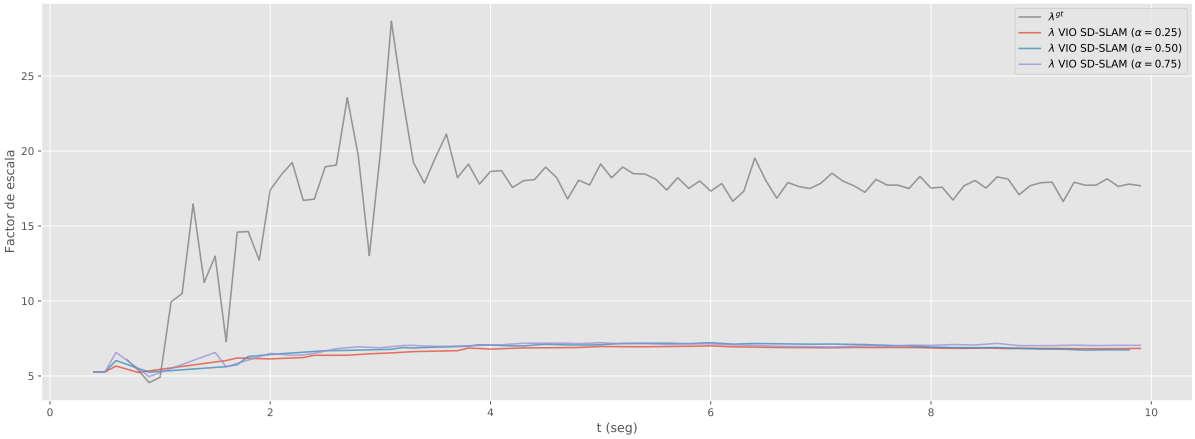


Figura 5.11: Comparativa entre el factor de escala de referencia (λ^{gt}) respecto al estimado por VIO SD-SLAM (λ) durante los primeros segundos de la secuencia 07.

escala estimado λ_i y con el factor de escala de referencia λ_i^{gt} para ese instante de tiempo y se calcular el error cometido entre ambos movimientos.

La Tabla 5.3 representa el error incremental RPE en metros generado por SD-SLAM respecto al producido por VIO SD-SLAM con las tres configuraciones de α seleccionadas. Como se puede observar, SD-SLAM obtiene un error mayor que su versión mejorada VIO SD-SLAM en todas las secuencias. Esto es debido a que al fin y al cabo SD-SLAM trabaja en una escala arbitraria que no tiene porqué corresponderse con la real. El error obtenido por VIO SD-SLAM se encuentra en todas las secuencias por debajo del metro entre *KeyFrames*, mientras que SD-SLAM llega en ocasiones a un error de casi 6 metros.

Con las mejoras introducidas en VIO SD-SLAM se obtiene un error cinco veces menor que el obtenido por su predecesor, llegando a obtener en el mejor de los casos un error ocho veces menor en la secuencia 00. Por otra parte, las tres configuraciones de VIO SD-SLAM obtienen

	SD-SLAM	**VIO SD-SLAM**		
		$\boldsymbol{\alpha = 0{,}25}$	$\boldsymbol{\alpha = 0{,}50}$	$\boldsymbol{\alpha = 0{,}75}$
Secuencia 00	5.8417	0.8966	0.7179	0.8735
Secuencia 05	2.5296	0.5979	0.6665	0.4616
Secuencia 06	3.9215	0.5271	0.4484	0.5002
Secuencia 07	1.6135	0.9274	0.8313	0.8853
Media	3.4765	0.7372	0.6660	0.6801

Tabla 5.3: RPE incremental (RMSE) en metros de la estimación de trayectoria tras aplicar el factor de escala.

unos resultados similares, siendo la configuración de α igual a $0{,}50$ la que logra el menor error medio. El parámetro α es configurable por el usuario, pero tras analizar los resultados obtenidos no es una mala idea fijarlo en $0{,}50$, obteniendo así una media entre la nueva estimación del factor de escala y la antigua. De forma individual el error mínimo en cada secuencia se encuentra repartido entre las diferentes configuraciones. Esto se debe a las estimaciones realizadas y el error acumulado en la velocidad, como se explicó anteriormente.

Por último, la secuencia donde la mejora en el error cometido es menor se produce en la secuencia 07. Esto era de esperar por el error inicial es la construcción del mapa inicial, que condiciona el resto de las estimaciones del factor de escala.

5.5. Experimentos de Reinicialización

Una de las principales características de VIO SD-SLAM es la capacidad de continuar funcionando cuando visión falla gracias a la incorporación de la odometría inercial. Con el objetivo de comprobar el funcionamiento y la calidad de VIO SD-SLAM se simulan pérdidas en la calidad visual en distintos tramos de las secuencias, ya que estas pérdidas no se observan de forma natural en las secuencias seleccionadas.

Para simular el fallo en visión se "apagará" la cámara, proporcionando a VIO SD-SLAM imágenes completamente negras donde el valor de intensidad de todos los píxeles es cero. Esta imagen imposibilita la realización del proceso habitual de *Tracking* de visión, dado que no se pueden obtener puntos de interés y provoca que VIO SD-SLAM transite al estado de *Tracking* inercial.

En total se han realizado tres pruebas diferentes:

- **Prueba 1:** La primera prueba tiene como objetivo verificar que VIO SD-SLAM se ha diseñado correctamente y es capaz de estimar la pose de la cámara mediante odometría inercial durante el estado pérdida y almacenarla en *Fake-KeyFrames*. Además se comprobará que la escala antes de entrar en el estado de pérdida y al salir es la misma.

- **Prueba 2:** La segunda prueba busca evaluar el comportamiento VIO SD-SLAM durante una pérdida en una situación en la cual la reinicialización se produce rápidamente.

- **Prueba 3:** La última prueba es similar a la anterior pero en una situación en la cual la reinicialización se demora en el tiempo.

A continuación se muestran los resultado obtenidos en cada una de las pruebas.

5.5.1. Prueba 1

El primer experimento que se presentan a continuación está enfocado a la verificación del diseño de VIO SD-SLAM, comprobando que durante el estado de pérdida nuestro algoritmo es capaz de estimar la pose mediante el uso de odometría inercial a la vez que se construyen de forma correcta los *Fake-KeyFrames* para almacenar la pose. Los *Fake-KeyFrames* son una de las principales ventajas esenciales de VIO SD-SLAM, volviendo al algoritmo más robusto en aplicaciones reales.

La metodología empleada consistirá en lanzar el algoritmo de VIO SD-SLAM con la secuencia 06 y esperar a que visión inicialice y, pasados unos segundos, se apagará la cámara. Durante este periodo se espera que nuestro algoritmo siga siendo capaz de estimar la pose de la cámara haciendo uso de la odometría inercial al mismo tiempo que crea *Fake-KeyFrames*. Tras cinco segundos en este estado, se volverá a encender la cámara y cuando la odometría de visión se restablezca (proceso de reinicialización correcto) se deberá empezar a usar esta.

Con el fin de observar estas situaciones, la interfaz gráfica de SD-SLAM ha sido modificada para representar los *KeyFrames* creados por visión en color azul y los *Fake-KeyFrames*, creados cuando se hace uso de la odometría inercial, en color naranja. Además, al finalizar la secuencia se almacenan en archivos de texto la pose contenida en cada uno de los *KeyFrames* de ambos tipos, un indicador para distinguir qué tipo de *KeyFrame* está asociado dicha pose y la escala estimada para cada uno de ellos.

La figura 5.12 representa una captura de la interfaz de VIO SD-SLAM para el experimento propuesto. Como se puede observar, el resultado obtenido es el comportamiento previsto en el diseño del algoritmo.

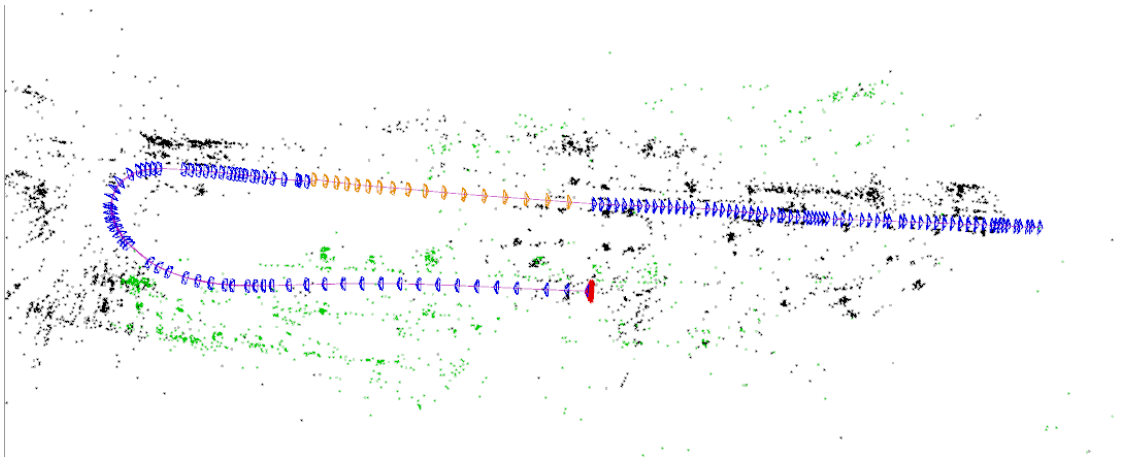


Figura 5.12: Trayectoria estimada por VIO SD-SLAM durante los 40 primeros segundos de la trayectoria 06. La imagen pertenece a la interfaz de VIO SD-SLAM y muestra en azul los *KeyFrames* estimados por visión y en naranja los *Fake-KeyFrames* estimados por odometría inercial durante el estado de pérdida.

Durante el estado de pérdida se puede observar cómo se continúa estimando la pose mediante odometría inercial y esta se almacenada en *Fake-KeyFrames* representados en color naranja. Por otra parte, la odometría visual tras reactivar la cámara tarda otros cinco segundos en reinicializarse. Como se puede observar en la imagen, el proceso de alinear la nueva trayectoria de visión con la llevada a cabo en el estado de pérdida se realiza correctamente. Por último, el factor de escala estimado por VIO SD-SLAM antes de entrar en el estado de pérdida era de $0{,}0421$, exactamente el mismo que el obtenido tras evaluar los dos primeros *KeyFrames* de la reinicialización de la odometría visual, validando la implementación del proceso de corrección de la escala del nuevo mapa.

El resultado satisfactorio de este pequeño experimento sirve como validación global de que el nuevo algoritmo VIO SD-SLAM ha sido bien programado y cada módulo realiza su función correctamente.

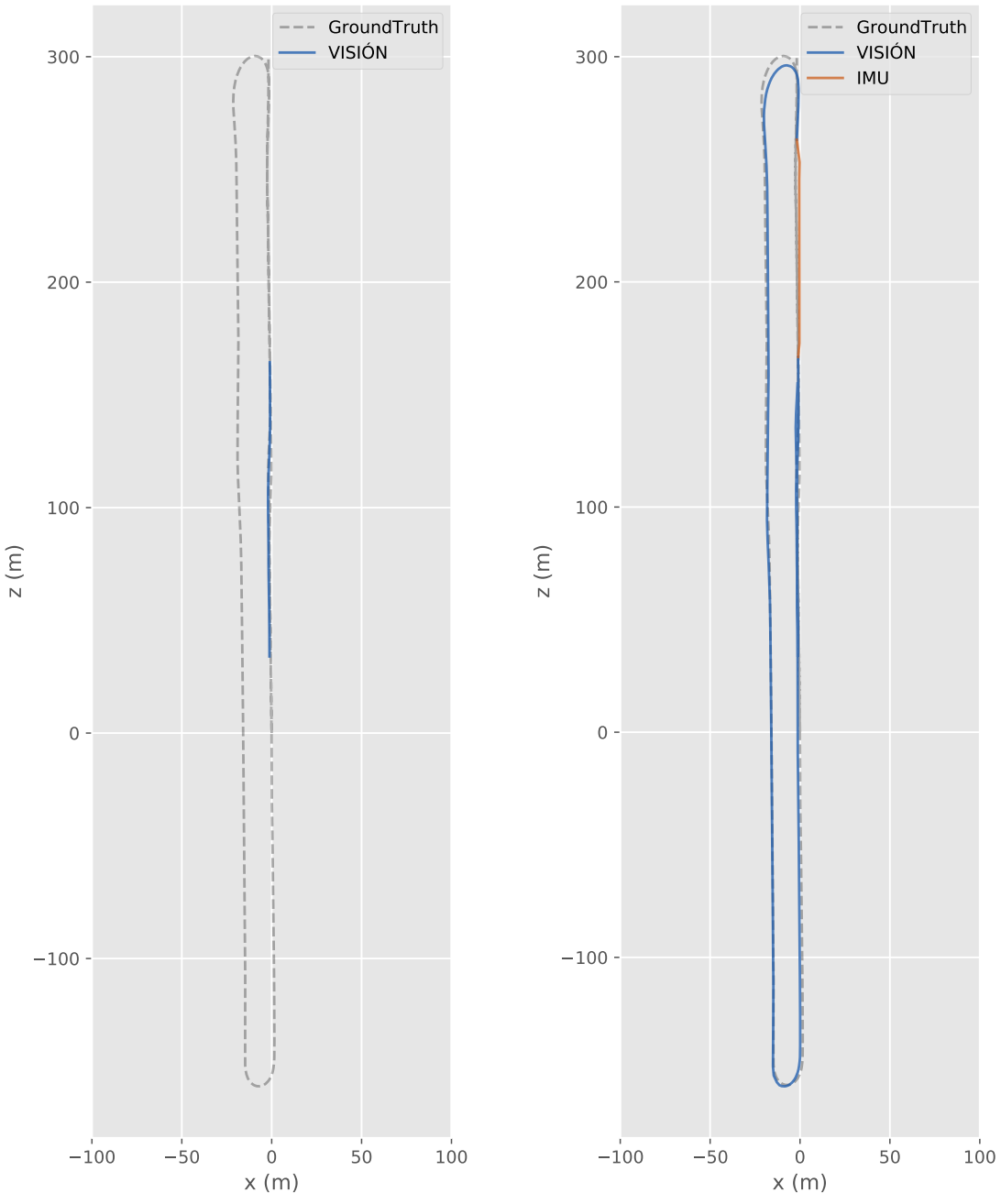
5.5.2. Prueba 2

Tras validar el correcto funcionamiento de VIO SD-SLAM en la prueba anterior, se procede a evaluar la calidad de los datos obtenidos en el proceso de reinicialización respecto a la secuencia de referencia. El tiempo de reinicialización depende enteramente de la parte visual, y puede llevarse a cabo a los pocos fotogramas o demorarse un largo periodo de tiempo. Durante esta prueba se busca evaluar el comportamiento de VIO SD-SLAM tras recuperarse en un corto periodo de tiempo de una pérdida. Con el fin de evaluar únicamente este comportamiento, la trayectoria estimada será escalada y alineada haciendo uso de la herramienta EVO.

Para la realización de esta prueba se empleará la secuencia 06 simulando una pérdida de textura apagando la cámara en un tramo recto durante el intervalo comprendido entre el segundo 12 y 17 de la secuencia. La cámara permanece apagada cinco segundos para forzar a estimar la trayectoria con odometría inercial como mínimo ese periodo de tiempo. La prueba será llevada a cabo tanto por SD-SLAM como por VIO SD-SLAM haciendo uso del acelerómetro real proporcionado por el *dataset*.

La Figura 5.13 representa una vista cenital de la trayectoria realizada por SD-SLAM a la izquierda y VIO SD-SLAM a la derecha. Igual que en el caso de la interfaz, los tramos realizados por odometría visual se representan en color azul y los tramos realizados con la IMU mediante odometría inercial se representan en color naranja.

Como se puede observar, la trayectoria realizada por SD-SLAM únicamente estima la localización de la cámara hasta que se produce la pérdida. A partir de ese momento no puede continuar realizando las estimaciones y queda a la espera de detectar una relocalización. Es decir, debido a una pérdida en el segundo 12 de los casi dos minutos que dura en total la prueba no vuelve a recuperarse hasta los segundos finales en los cuales se detecta la relocalización,



(a) (b)

Figura 5.13: Trayectoria estimada por SD-SLAM (a) y VIO SD-SLAM (b) en la secuencia 06 forzando una pérdida en el segundo 12.

producida al transitar por la zona inicial.

Por otra parte, VIO SD-SLAM realiza la estimación de la localización de la cámara durante toda la secuencia, gracias a la incorporación de la IMU que permite emplear la odometría inercial durante el estado de pérdida. En total se emplea la odometría inercial durante un periodo de

10,3 segundos, ya que a parte de los cinco segundos en los cuales la cámara permanece inactiva, requiere de otros cinco segundos para recuperar la nueva odometría visual. Debido a que en este caso la estimación de la escala antes de entrar en el estado de pérdida era aproximada a la de referencia, se trataba de un tramo recto y a la detección del cierre de bucle que se produce, la trayectoria estimada que se obtiene es cercana a la trayectoria de referencia.

En esta prueba no se han añadido tablas de error comparativas entre ambos algoritmos ya que carece de sentido comparar el error cometido por una trayectoria de 12 segundos respecto a una de dos minutos.

5.5.3. Prueba 3

La prueba realizada a continuación sigue la misma línea que la descrita en el apartado anterior pero en un entorno más exigente. Para ello se empleará la trayectoria 00 hasta detectar el primer cierre de bucle. En esta secuencia los datos proporcionados por la IMU real son más ruidosos (ver Figura 5.6 **a**) que en el resto de secuencias y la estimación de escala realizada es peor que en el caso anterior, como se observó en los experimentos realizados de la estimación del factor de escala. Además, en esta secuencia la odometría visual, como se verá más adelante, se demora en el tiempo hasta que es capaz de reinicializarse, provocando un mayor uso en el tiempo de la odometría inercial.

En esta ocasión se simula una pérdida de textura apagando la cámara justo antes de realizar la tercera curva de la secuencia, en el intervalo comprendido entre el segundo 39 y 44. Debido a la calidad de los datos proporcionados por la IMU real en esta secuencia, la prueba será llevada a cabo por VIO SD-SLAM haciendo uso de los tres acelerómetros disponibles (real, ruidoso y pseudo-ideal) y, como en el caso anterior, por SD-SLAM.

Los resultados obtenidos por SD-SLAM, como puede observarse en la Figura 5.14 (a), siguen siendo los mismos que en el caso anterior, ya que desde el momento en el que el algoritmo deja de recibir información visual deja de realizar estimaciones de pose, y no es capaz de relocalizarse hasta la parte final de la secuencia donde se realiza la relocalización.

Por otra parte, VIO SD-SLAM es capaz de realizar la estimación de la trayectoria durante toda la secuencia, indiferentemente del acelerómetro empleado. La reinicialización de la odometría visual se produce aproximadamente 30 segundos después de entrar en el estado de pérdida, más adelante se tratará de dar una explicación a esta demora. Por tanto, lo primero que se puede observar es que en las tres trayectorias el proceso de reinicialización de la odometría visual se demora significativamente en el tiempo. Esto se traduce en un mayor tiempo de uso de la odometría inercial provocando mayores derivas en la estimación de la pose de la cámara en el tiempo.

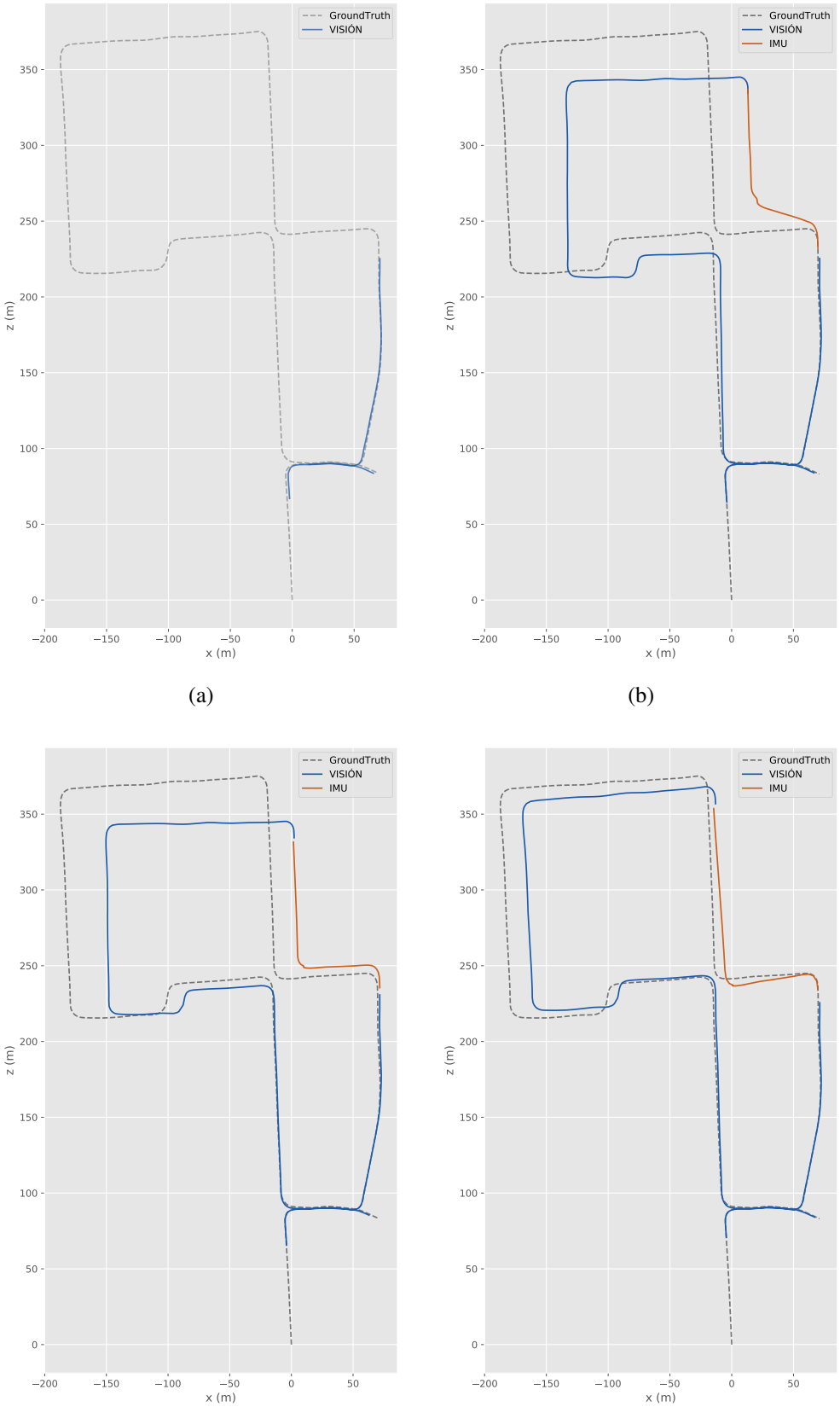


Figura 5.14: Trayectoria estimada por SD-SLAM (a) y VIO SD-SLAM con el acelerómetro real (b), ruidoso (c) y *pseudo*-ideal (d) en la primera parte de la secuencia 00 forzando una pérdida en la tercera curva.

Los resultados en la estimación de la trayectoria durante el estado de pérdida obtenidos con VIO SD-SLAM aumentan a medida que lo hace la calidad de las medidas proporcionadas por el acelerómetro. Como puede observarse en la Figura 5.14 (b), haciendo uso del acelerómetro real las curvas de 90° no llegan a realizarse completamente, provocando una deriva en la trayectoria. Esto es debido a la combinación del error en los datos del sensor así como del error en la velocidad inicial estimada antes de entrar en el estado de pérdida. El error en la estimación de la escala puede observarse en el primer tramo recto, donde es el recorrido realizado por odometría inercial es inferior al esperado. Este diferencia entre el factor de escala estimado por el algoritmo y el esperado también afecta al segundo tramo de odometría visual. Como se ha comentado anteriormente, al reinicializar la odometría visual se fuerza a mantener el mismo factor de escala estimado antes de entrar en el estado de pérdida, pero, si esta estimación no era lo suficientemente buena puede dar lugar a una diferencia entre ambos tramos de odometría visual. Sin embargo, al realizar el cierre de bucle este problema desaparece puesto que toda la secuencia se adapta para hacer coincidir ambos tramos visuales, incluyendo los tramos exclusivamente inerciales. Pese a no ser un resultado perfecto, en comparación con la trayectoria realizada por SD-SLAM es una notable mejora.

El resultado obtenido empleando el acelerómetro ruidoso, como puede observarse en la Figura 5.14 (c), mejora respecto al acelerómetro real. La estimación del factor de escala es ligeramente mejor, y sobretodo las medidas del sensor son mejores, estimando mucho mejor la trayectoria realizada durante el tramo de odometría inercial.

El resultado empleando el acelerómetro *pseudo*-ideal es el mejor de los cuatro, como puede observarse en la Figura 5.14 (d). La trayectoria es reconstruida prácticamente a la perfección. Hay que dejar claro que las medidas de este acelerómetro dejan de ser ideales en cuanto se realiza la primera corrección de la velocidad inicial haciendo uso de las poses estimadas por visión. Debido a este motivo, como en el caso anterior, la escala no se estima de forma perfecta. De hecho, puede observarse cómo al contener un error en la estimación de la velocidad inicial la primera curva no se realiza de forma correcta. A pesar de todo, los resultados obtenidos son significativamente mejores que en los casos anteriores.

Finalmente, se trata de dar una explicación a la demora en la reinicialización de la odometría visual. Este retraso puede ser debido a que gran parte de los puntos característicos que se emplean en el proceso de triangulación tienen poco paralaje, imposibilitando obtener una buena estimación de profundidad. La Figura 5.15 muestra una captura de la interfaz de VIO SD-SLAM durante el estado de pérdida de este experimento y en ella se representan los emparejamientos de cada par de puntos característicos detectados por ORB mediante una línea de color verde. Como puede observarse, los emparejamientos contenidos en los círculos rojos apenas constan de paralaje, dificultando obtener una buena transformación entre ambos fotogramas a partir de ellos.

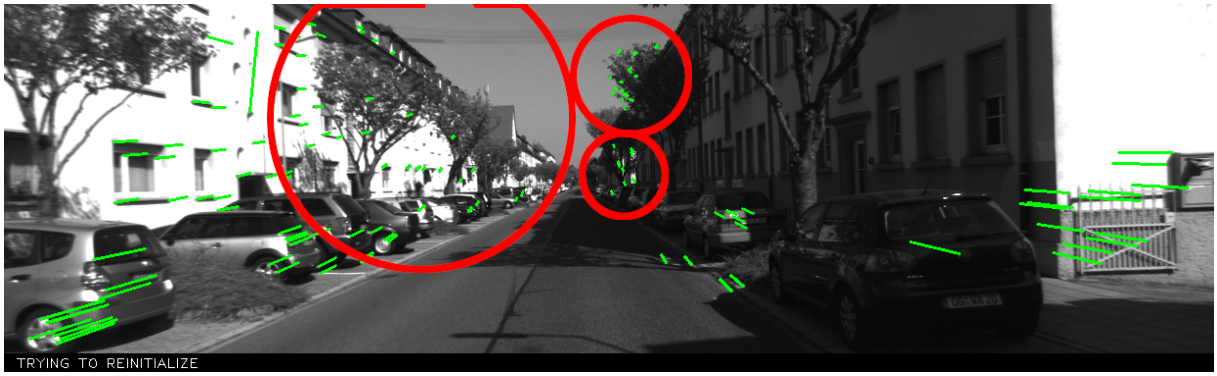


Figura 5.15: Emparejamientos de puntos de interés obtenidos por ORB durante el estado de pérdida en VIO SD-SLAM.

Sin embargo, durante este periodo, la parte puramente visual del algoritmo construye varios mapas iniciales, pero debido a que han sido generados con puntos sin el suficiente paralaje, el mapa así como la posición inicial de de los dos fotogramas son erróneos, provocando inicializaciones fallidas como la observada anteriormente en la Figura 5.10. Es decir, el nuevo mapa no tiene la suficiente consistencia para ser considerado válido. Por este motivo, únicamente se consideran reinicializaciones válidas aquellas en las cuales la estimación realizada por el proceso de inicialización de visión es similar al movimiento realizado en dichos fotogramas por la odometría inercial.

5.5.4. Conclusiones

A partir de los resultados obtenidos en esta sección se puede llegar a la conclusión de que incluir un sensor inercial en algoritmos de Visual SLAM empleando sistemas monoculares supone una gran mejora en robustez respecto al estado original del algoritmo. SD-SLAM era incapaz de seguir estimando la trayectoria de la cámara cuando la calidad visual se veía comprometida, obligando a detener el hilo de *Tracking* y *Mapping* hasta el momento de revisitar una zona conocida y, en el caso de que esto no ocurriese, el algoritmo quedaría en este estado de espera hasta la finalización de su ejecución. Originalmente no tenía sentido plantear una reinicialización de la parte visual, dado que si bien es posible construir un nuevo mapa, no se conoce la posición espacial para situarlo. Esto obligaría a descartar el mapa antiguo en favor del nuevo, situándolo en el origen. Es por este motivo que los algoritmos de Visual SLAM incluyen el módulo de relocalización (detectar cuando se revisita una zona conocida para recuperar la pose que se tenía en ese momento).

Sin embargo, VIO SD-SLAM permite mantener una estimación de la trayectoria realizada por la cámara durante el estado de pérdida de visión. Esto permite reinicializar la parte visual construyendo un nuevo mapa. En esta versión mejorada de SD-SLAM, al disponer de la tra-

yectoria realizada en el estado de pérdida por la odometría inercial, es posible dar una pose al nuevo mapa directamente relacionada con el antiguo, evitando tener que depender de revisitar una zona conocida. Como se ha podido observar, la calidad del sensor empleado, la estimación del factor de escala y el error acumulado en el modelo de movimiento inercial serán factores clave de una buena estimación de trayectoria.

Todos aportaciones realizadas han quedado validadas experimentalmente, como se discutirá en el siguiente capítulo junto a los resultados obtenidos.

Capítulo 6

Conclusiones

En los capítulos anteriores se ha descrito el diseño propuesto para mejorar SD-SLAM, un algoritmo de auto-localización 3D puramente visual, en un algoritmo que combina las odometrías visual e inercial denominado VIO SD-SLAM, así como una serie de experimentos para validar su funcionamiento. El objetivo principal del TFM era comprobar y demostrar de qué maneras se podía mejorar los resultados y la robustez de un algoritmo de Visual SLAM puro (SD-SLAM en este caso) al ser complementado con una unidad de medición inercial (IMU). Los objetivos específicos que se han explorado han sido principalmente la reconstrucción de la escala real a partir de la relación entre odometría visual e inercial, y hacer uso de esta última cuando la odometría puramente visual fallase. En este último capítulo se reflexiona sobre los resultados obtenidos, los objetivos cumplidos y posibles trabajos futuros.

6.1. Discusión

La integración de ambos sensores (cámara e IMU), como se ha podido observar a lo largo del desarrollo del trabajo, no es un proceso sencillo. Esto se debe principalmente a que tanto el modelo de odometría visual de un sistema monocular así como el modelo de odometría inercial empleando IMUs sufren derivas espaciales a lo largo del tiempo. Además, hay que añadir otros dos problemas: los modelos de odometría visual e inercial trabajan en escalas diferentes y la escala de odometría visual no es constante a lo largo del tiempo.

Con estos objetivos en mente, se barajó la opción de trabajar con la odometría inercial de forma totalmente paralela al sistema de Visual SLAM. Esta opción, como se ha podido observar y demostrar en la sección 5.3, es inviable a menos que se tenga acceso a un sensor IMU con una calidad de medidas enormemente alta. Lamentablemente, esta última opción supone un alto coste económico, y la línea de investigación en la que se enmarca este proyecto trata de combinar tanto cámaras como sensores IMU de bajo coste económico que suelen ir asociados a bajo peso y tamaño, estando ambos dispositivos presentes en muchos equipos tecnológicos

(como teléfonos móviles o drones).

Debido a este motivo, en el diseño implementado se ha tenido que seleccionar un modelo de odometría para corregir las estimaciones realizadas por el otro modelo. Por una parte, se ha observado que el error en el modelo de odometría inercial es acumulativo, es decir, el error producido en la primera estimación nunca desaparece y se extiende a la segunda estimación, y este a su vez a la tercera, y así sucesivamente. Esto implica que el error crecerá a lo largo del tiempo. Por otra parte, la odometría visual sufre, en primer lugar, una deriva en la escala en la cual representa el mundo (problema asociado a los sistemas monoculares) y, en segundo lugar, la propia calidad de las estimaciones de la localización de la cámara entre fotogramas fluctúa levemente. Teniendo todo lo anterior en cuenta, en el diseño de VIO SD-SLAM finalmente se ha optado por dar más importancia y fiabilidad a la información obtenida visualmente para realizar las correcciones del modelo inercial.

La propuesta implementada permite hacer uso de las estimaciones del modelo visual para corregir las derivas en posición y orientación del sistema inercial, así como la velocidad lineal para eliminar el error acumulado en esta componente. Sin embargo, en este punto nos encontramos ante una paradoja, puesto que se necesita aplicar el factor de escala para realizar el proceso de corrección (debido a que ambos sistemas trabajan en escalas diferentes) con el objetivo de estimar este mismo factor de escala en la siguiente iteración.

Como se ha observado en los experimentos realizados en la sección 5.4, la calidad de la estimación del factor de escala repercute directamente en la precisión del proceso de corrección y por tanto en la propia trayectoria realizada por el modelo de odometría inercial. El error en la estimación de posición inercial se debe principalmente al valor que toma la velocidad lineal, fruto de la estimación de movimiento realizada por la odometría visual y su posterior escalado al mundo real. En su conjunto es un problema complejo, puesto que cada paso realizado es propenso a cometer un pequeño error (ya sea en las medidas proporcionadas por los sensores, en la estimación del ruido, orientación, velocidad, pose o factor de escala) y todos ellos se acumulan de un paso al siguiente.

Teniendo todas estas cuestiones en cuenta, VIO SD-SLAM hace uso de la odometría visual mientras sea capaz de realizar sus funciones correctamente, trasladando la odometría inercial a un segundo plano encargándose de estimar el factor de escala que relaciona ambos modelos para poder transformar las estimaciones visuales a la escala del mundo real siempre que se desee. Solamente cuando el modelo visual entra en un estado de pérdida, la odometría inercial toma el relevo con el fin de continuar proporcionando una estimación de la trayectoria hasta que la odometría visual se recupere. La calidad de la trayectoria estimada inercialmente depende de todos los factores explicados anteriormente.

Asociado a este último escenario también se ha encontrado un problema que no se esperaba, y es la demora en la reinicialización de la odometría visual. Este problema, como se especuló

en la sección 5.5, puede estar asociado al *dataset* seleccionado y a la dificultad de realizar correctamente el proceso de triangulación en un tramo del recorrido donde una parte de las características emparejadas detectadas presentan un bajo paralaje debido a que se encuentran en el punto de fuga de las vías de circulación.

6.2. Objetivos alcanzados

El objetivo principal, que consistía en demostrar que la inclusión de unidades de medición inerciales pueden mejorar los algoritmos de Visual SLAM, se ha cumplido satisfactoriamente. Todo ello ha dado lugar a la mejora del algoritmo conocido como SD-SLAM en una nueva versión nombrada como VIO SD-SLAM. De forma especifica, pese a los problemas encontrados inherentes a los tipos de sistema y sensores empleados y discutidos en la sección anterior, se han logrado abordar de forma satisfactoria todos ellos y demostrando que VIO SD-SLAM funciona significativamente mejor que SD-SLAM.

Primero, con la adición de la IMU se puede realizar una estimación entre la escala arbitraria de visión y la escala real del mundo, eliminando el problema de la ambigüedad en la escala cuando se realiza correctamente. Y segundo, en robustez, SD-SLAM no es capaz de localizarse o estimar la trayectoria de la cámara cuando la calidad visual se ve comprometida. Este problema se solventa en VIO SD-SLAM mediante el uso del modelo de odometría inercial mientras se permanece en este estado. Todo ello se ha validado experimentalmente para verificar cada una de las ampliaciones, comparándolo cuando era necesario con su predecesor, dando lugar un nuevo algoritmo de Visual SLAM más robusto.

Todas las ampliaciones realizadas se han logrado cumpliendo con los requisitos propuestos, que eran trabajar en tiempo real y no ver comprometida la robustez inicial del algoritmo. El primer requisito se satisface ya que mientras la odometría visual está activa mantiene su funcionamiento original. El segundo requisito también se cumple ya que las mejoras introducidas suponen una complejidad constante en tiempo y espacio, a excepción del proceso de actualización de la escala. Este proceso requiere de realizar una media de los factores de escala estimados cuando se crea un nuevo *KeyFrame*. Debido a que de forma predeterminada se crea mínimo un *KeyFrame* por segundo, el número de factores de escala siempre será un conjunto pequeño y por tanto no afecta al rendimiento original del algoritmo.

6.3. Trabajos futuros

Algunos trabajos futuros derivados de los problemas encontrados a lo largo del desarrollo del TFM podrían suponer mejoras en el algoritmo desarrollado, así como en la odometría visual-inercial en general:

- Explorar nuevas soluciones para mejorar el modelo de movimiento inercial planteado. El principal punto a abordar debería ser la estimación y filtrado del ruido procedente del acelerómetro, tarea que no se ha realizado en este proyecto. Con ello se mejorarían los resultados al trabajar con la IMU real, aproximándolos a los obtenidos por la IMU *pseudo*-ideal.

- Realizar pruebas con vehículos aéreos no-tripulados (UAV) así como con diferentes tipos de IMUs para validar el correcto funcionamiento del algoritmo ante diferentes entornos, sistemas y calidad del sensor.

- Debido a la demora en el proceso de recuperación de la odometría visual desde el estado de pérdida, una posible solución sería plantear el proceso de reinicialización como un problema de reconstrucción 3D conocidas la pose de cada fotograma. Actualmente, el problema de reinicialización es puramente visual y parte de la detección y emparejamiento de puntos característicos. A partir de los emparejamientos se calcula la matriz de homografía o fundamental que minimiza el error de retro-proyección. Posteriormente, se estima el movimiento relativo (traslación y orientación) entre los dos fotogramas y, por último, conocidas las posiciones de los fotogramas y los puntos característicos coincidentes en ambas imágenes se calcula la posición tridimensional de cada punto mediante triangulación. Además, en el caso de VIO SD-SLAM se necesita transformar la escala de la nueva escena a escala de la trayectoria visual anterior antes de entrar en pérdida.

  La propuesta sería estimar directamente el mapa inicial haciendo uso de la estimación de pose realizada por la odometría inercial. Conocidas las poses de cada fotograma y un conjunto de puntos característicos emparejados, se podría triangular directamente la posición tridimensional de cada punto característico mediante por triangulación dado que la pose de cada fotograma ya es conocida. Además de agilizar el proceso de reinicialización, evitaría tener que transformar la escala del nuevo mapa puesto que se recuperaría directamente al emplear las poses de los fotogramas obtenidas por odometría inercial.

- Realizar distintas aproximaciones para estimar la escala como las planteadas por algunos trabajos como [28], o la más reciente publicada este mismo año [3]. Estas aproximaciones se basan en el uso de un conjunto de puntos 3D iniciales del mapa para estimar la escala, vector de gravedad y aproximaciones a los valores de ruido de los sensores inerciales.

- Plasmar los resultados obtenidos en un artículo científico.

Bibliografía

[1] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. volume 3951, pages 404–417, 07 2006.

[2] G. Cai, B. M. Chen, and T. H. Lee. *Unmanned Rotorcraft Systems*. Springer Publishing Company, Incorporated, 2011.

[3] C. Campos, J. M. M. Montiel, and J. D. Tardós. Inertial-only optimization for visual-inertial initialization. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 51–57, 2020.

[4] J. Civera, A. J. Davison, and J. M. M. Montiel. Inverse depth parametrization for monocular slam. *IEEE Transactions on Robotics*, 24(5):932–945, 2008.

[5] A. J. Davison. Slam with a single camera. *Concurrent Mapping and Localization for Autonomous Mobile Robots, ICRA*, 2002.

[6] A. J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *null*, page 1403. IEEE, 2003.

[7] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007.

[8] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *European Conference on Computer Vision (ECCV)*, September 2014.

[9] M. Euston, P. Coote, R. Mahony, J. Kim, and T. Hamel. A complementary filter for attitude estimation of a fixed-wing uav. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 340–345, 2008.

[10] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza. Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. 07 2015.

[11] C. Forster, M. Pizzoli, and D. Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22, 2014.

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

[13] E. P. García. Técnicas para la localización visual robusta de robots en tiempo real con y sin mapas.

[14] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[15] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[16] P. Groves. *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*. 03 2008.

[17] M. Grupp. evo: Python package for the evaluation of odometry and slam. `https://github.com/MichaelGrupp/evo`, 2017.

[18] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 edition, 2004.

[19] W. S. F. Iv, J. A. Wall, and D. Bevly. Characterization of various imu error sources and the effect on navigation performance. 2005.

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

[21] G. Klein and D. W. Murray. Parallel tracking and mapping for small ar workspaces. *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 225–234, 2007.

[22] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–, 11 2004.

[23] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart. A robust and modular multi-sensor fusion approach applied to mav navigation. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3923–3929, 2013.

[24] S. O. H. Madgwick, A. J. L. Harrison, and R. Vaidyanathan. Estimation of imu and marg orientation using a gradient descent algorithm. In *2011 IEEE International Conference on Rehabilitation Robotics*, pages 1–7, 2011.

[25] A. I. Mourikis and S. I. Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3565–3572, 2007.

[26] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[27] R. Mur-Artal and J. D. Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[28] R. Mur-Artal and J. D. Tardós. Visual-inertial monocular slam with map reuse. *IEEE Robotics and Automation Letters*, 2(2):796–803, 2017.

[29] N. Nain, V. Laxmi, B. Bhadviya, D. B M, and M. Ahmed. Fast feature point detector. In *2008 IEEE International Conference on Signal Image Technology and Internet Based Systems*, pages 301–306, 2008.

[30] J. Nazareth. Iterative methods for optimization by c. t. kelley. *SIAM Review*, 42:535–539, 01 2000.

[31] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In A. Leonardis, H. Bischof, and A. Pinz, editors, *Computer Vision – ECCV 2006*, pages 430–443, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[32] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: an efficient alternative to sift or surf. pages 2564–2571, 11 2011.

[33] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[34] H. Stewénius, C. Engels, and D. Nistér. Recent developments on direct relative orientation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 60(4):284 – 294, 2006.

[35] L. Stumberg, V. Usenko, and D. Cremers. Direct sparse visual-inertial odometry using dynamic marginalization. pages 2510–2517, 05 2018.

[36] J. Sturm, W. Burgard, and D. Cremers. Evaluating egomotion and structure-from-motion approaches using the TUM RGB-D benchmark. In *Proc. of the Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RJS International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[37] B. Triggs, P. Mclauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment - a modern synthesis. *ICCV '99 Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, pages 198–372, 01 2000.

[38] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):376–380, 1991.

[39] O. J. Woodman. An introduction to inertial navigation. *University of Cambridge. Technical Report*, (UCAM-CL-TR-696), 2007.