Localización y mapeo simultáneos mediante el uso de un sistema de visión estéreo

Pire, Taihú Aguará Nahuel

2017-03-02

Universidad de Buenos Aires

Facultad de Ciencias Exactas y Naturales

Departamento de Computación

Localización y mapeo simultáneos mediante el uso de un sistema de visión estéreo

Tesis presentada para optar al título de
Doctor de la Universidad de Buenos Aires en el área de Ciencias de la Computación

Taihú Aguará Nahuel Pire

Director: Dr. Julio Jacobo Berlles

Consejera de Estudios: Dra. Marta E. Mejail

Lugar de Trabajo: Laboratorio de Robótica y Sistemas Embebidos (LRSE), Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires

Buenos Aires, 2017

Localización y mapeo simultáneos mediante el uso de un sistema de visión estéreo

Para que un robot móvil pueda navegar o realizar tareas de manera autónoma, este debe conocer su pose (posición y orientación) y contar con una representación del entorno (mapa) en el que se encuentra. En entornos donde no se cuenta con un mapa previo y el robot no cuenta con información externa que le permita conocer su pose, debe realizar dichas tareas de manera simultánea. El problema de localizar a un robot y construir un mapa del entorno simultaneamente se denomina SLAM por las siglas en inglés de Simultaneous Localization and Mapping.

En esta tesis se presenta un método basado en visión estéreo para abordar el problema de SLAM. El método, denominado S-PTAM por el acrónimo en inglés de Stereo Parallel Tracking and Mapping, fue desarrollado de manera tal que sea capaz de correr en tiempo real en ambientes de grandes dimensiones permitiendo estimar de forma precisa la pose del robot a medida que construye un mapa del ambiente en un sistema de coordenadas global.

Para tener un desempeño óptimo, S-PTAM desacopla las tareas de localización y mapeo presentes en el problema de SLAM en dos hilos de ejecución independientes. Esto permite aprovechar el poder computacional de los procesadores de múltiples núcleos. Además de los módulos de localización y mapeo, se propone un módulo de detección y cierre de ciclos que permite reconocer lugares previamente visitados por el robot. Los ciclos detectados son utilizados para realizar una corrección tanto del mapa como de la trayectoria estimada, reduciendo efectivamente el error acumulado por el método hasta el momento.

S-PTAM trabaja sobre las características visuales extraídas de las imágenes provistas por la cámara estéreo. Para determinar qué extractor de características es el más adecuado en términos de precisión, robustez y costo computacional se presenta una comparación de los detectores y descriptores binarios más relevantes de la literatura.

Finalmente, se presentan experimentos con datasets públicos que permiten validar la precisión y la performance del método propuesto. Como resultado se obtuvo que S-PTAM es uno de los métodos de SLAM más precisos del estado del arte. S-PTAM fue publicado como software libre para facilitar su uso y comparación con otros métodos de SLAM.

Palabras clave: robótica móvil, SLAM, visión estéreo, cierre de ciclos

Stereo Parallel Tracking and Mapping

In order to allow a mobile robot navigate and perform tasks autonomously, it must know its pose (position and orientation) and have a representation of the environment (map). In environments where the robot does not have a previous map and no external information is provided to know its pose, it is necessary to perform both tasks simultaneously. The problem of localizing a robot and building a map of the environment simultaneously is called SLAM; this stands for Simultaneous Localization and Mapping.

In this thesis, a system based on stereo vision to address the problem of SLAM is presented. The method, called S-PTAM as an acronym for Stereo Parallel Tracking and Mapping, was developed. This method is intended to run in real-time for long trajectories, allowing to estimate the pose accurately as it builds a sparse map of the environment on a global coordinate system.

For optimal performance, S-PTAM decouples localization and mapping tasks of the SLAM problem into two independent threads, allowing us to take advantage of multicore processors. Besides the localization and the mapping modules, a loop closure module that can recognize places previously visited by the robot is proposed. The detected loops are used to refine the map and the estimated trajectory, effectively reducing the accumulated error of the method so far.

S-PTAM works on the visual features extracted from the images provided by the stereo camera. To determine which feature extractor is the most suitable in terms of accuracy, a comparison in terms of robustness and computational cost of the most relevant detectors and binary descriptors in the literature is performed.

Finally, experiments with public datasets for validating the accuracy and performance of the proposed method are presented. As a result S-PTAM is one of the most accurate SLAM methods of the state of the art. S-PTAM was released as free software to ease its use and to allow comparison with other SLAM methods.

Keywords: mobile robotics, SLAM, stereo vision, loop closure

A Aymi

Agradecimientos

Quiero agrader...

a Thomas porque es un compañero de batalla con el que fue un placer hacer este camino, juntos enfrentamos grandes desafíos y me alegra haberlo hecho con él.

a Facu por siempre contar con él dentro y fuera del laboratorio.

a Gastón por ser un tesista ejemplar, con el que aprendí más de lo que enseñé.

a Matías por la gran ayuda que me prestó en innumerables oportunidades.

a Pablo por los grandes debates que me hicieron crecer tanto en lo profesional como en los personal.

a Javier Civera por la motivación, dedicación y confianza que me tuvo para que mi trabajo sea mucho mejor.

a Marta y Julio por hacerme parte de la familia Porteña y siempre estar dispuestos a ayudarme en lo que necesite.

a las grandes personas que descubrí en el laboratorio a lo largo todo el trayecto: Javier, Sol, Kevin, Carlos y Pato.

a los chicos de Imágenes con los que compartí divertidos cafés y encuentros: Pachi, Seba, Norbert, Mailu, María Elena y Daniel.

a los chicos de de la Universidad de Zaragoza Alejo, Raúl, Leo y Dani con los cuales compartí grandes momentos en cada una de mis estancias allí.

a mi psicóloga Adriana por hacerme la cuesta de este trabajo menos empinada.

a Sergio y Emiliano por hacerme sentir como en casa.

a mis amigos de toda la vida: Matías, Ignacio, Lionel, Milton, Mirko y Sergio por celebrar conmigo cada despedida y cada llegada.

a mi Mamá y Papá; y mis hermanos Tesi y David por bancarme siempre y querer que sea buena persona (aunque a mi no siempre me guste serlo).

a mi novia Aymi, porque lo que aguantó esta mujer tiene nombre, y es el mio.

A todos ustedes mil gracias![1]

[1]Estos agradecimientos se escribieron en un par de días, y cada vez que los escribí y re-escribí, felizmente lloré.

Contents

Chapter 1

Introduction

A robust and accurate self-localization and mapping of the surrounding areas is an essential competence to perform most of the robotic tasks in a wide variety of applications. Due to the sensor noise, constructing and updating the map of an unknown environment has to be done simultaneously with the estimation of the robot pose within it. Such problem is referred to with the acronym SLAM, standing for Simultaneous Localization and Mapping, and has been the object of active research during the last two decades.

Most of the early works on SLAM made use of a laser rangefinder as the main sensor in combination with wheel odometry [1]. More recently, visual sensors -either passive [2] or active [3]- have become the dominant choice. The odometric information has become less relevant, making visual SLAM suitable for other applications like Augmented and Virtual Reality. The affordable, small and light nowadays cameras can provide high resolution data at high frame rates. Their range is unlimited -at the price of a large depth uncertainty for small parallax pixels-, in contrast to the range limits of laser rangefinders. Moreover, cameras are passive sensors and therefore do not interfere with each other, and unlike *Structured light range sensors* (SLRS), they can be used in both indoor and outdoor environments. These characteristics make cameras the best choice for a general multi-purpose mobile robotic platforms.

For the above reasons, visual SLAM has become one of the most studied topics in the latest decade. And nowadays it is possible to achieve robust and accurate visual SLAM results in real time. However, some significant challenges remain, particularly for monocular configurations -namely highly dynamic environments or fast camera motions. In these scenarios the stereo cameras offer a higher degree of robustness. Triangulating the depth from a single view -and hence initializing points without delay and with small uncertainty- allows to initialize the system robustly and to augment the map with undelayed depth information. In addition a stereo setting allows to recover the real scale and to alleviate the scale drift. While a monocular-inertial combination (e.g., [4]) can also be used to extract the real scale of the scene, the reader should notice that the two sensor settings are complementary. Inertial sensors are not reliable in periods of constant velocity motion. Stereo cameras, on the other hand, are equivalent to monocular ones for large -compared to their baseline- scene depths. A stereo-inertial combination (as in [5]) can be used to avoid their individual limitations. RGB-D sensors also provide the real scale of the scene for SLAM and have the added value of dense depth measurements (for example, [6]). However, the depth measurements are range-limited and they cannot work under direct sunlight. They are hence limited to indoor scenes, lacking the generality of stereo cameras.

In this Thesis we present a real-time SLAM system using a stereo camera, henceforth referred to as S-PTAM (from *Stereo Parallel Tracking and Mapping*). Stereo cameras allow to match the same visual point-landmarks on a pair of synchronized views, recovering their real depth accurately if the parallax is high enough. As the robot moves through the environment, it is possible to track the visual landmarks frame after frame by jointly estimating the landmark depths and tracking the robot pose. In the experiments of this Thesis, the stereo setting plays a key role in some challenging cases of dynamic objects and pure rotation motions.

Feature-based visual SLAM approaches rely on the quality and quantity of local image features. On the one hand, the accuracy of the localization heavily depends on the homogeneous deployment of features in images and the ability to track them for long periods, even from different points of view and lighting conditions. On the other hand, if the number of points in the map grows too quickly, this may slow down the whole system. To be able to keep the response of the system under real-time constraints, images have to be dropped or other parts of the system, like optimization routines, must use less computational resources.

Currently, there exist several local image feature extractors. A feature extractor is a combination of a salient point (called *keypoint*) detection procedure and a computation of a unique signature (called *descriptor*) for each of such detected points. The most commonly used detectors are SIFT [7], SURF [8], STAR [9], GFTT [10], FAST [11], AGAST [12], and the relatively recently proposed ORB [13], while among the most used descriptors we can mention SIFT, SURF, ORB, BRIEF [14], BRISK [15], and LATCH [16]. In this work, we also evaluate the impact of different state-of-the-art feature extractors on the performance of the visual SLAM localization method to find the best option.

Following the approach of Parallel Tracking and Mapping (PTAM) [17], S-PTAM divides the problem into two main parallel tasks: camera tracking and map optimization. These tasks are executed in two different threads, sharing only the map between them. The tracking thread matches features, creates new points and estimates the camera pose for every new frame, and the mapping thread iteratively refines the nearby point-landmarks that compose the map.

In addition, to providing a consistent global location on large scale trajectories, a Loop Closing module was incorporated to S-PTAM. This module can detect when the robot is going through a previously visited place and can perform a correction of the localization error generated by the accumulated drift of the system.

1.1 Contributions

The contribution of this Thesis is the design and implementation of S-PTAM, a visual SLAM system based on stereo vision, able to work in real time even in large scale trajectories. S-PTAM was developed from scratch to achieve a flexible, robust and accurate stereo SLAM system. Its main characteristics can be summarized as follows:

- The SLAM problem is heavily parallelized achieving real-time performance, whilst minimizing inter-thread dependency.
- The stereo constraints are used for point initialization, mapping and tracking phases, improving the accuracy and robustness of the system.
- Real-time loop detection and correction are included in the system. The loop detection is performed using appearance-based image mapping and the loop correction by optimizing the pose graph of the system.
- A maintenance process that runs in an independent thread iteratively refines the map (Bundle Adjustment) in a local co-visible area, improving global consistency.
- Although the method works with the only input of a stereo sequence, wheel odometry can also be used for further accuracy and robustness.
- Binary features are used to describe visual point-landmarks, thus reducing the storage requirements and the matching cost.

The Loop Closing module of S-PTAM presented in this thesis results from the undergraduate thesis "Detección y cierre de ciclos en sistemas SLAM basados en visión estéreo" of Gastón Ignacio Castro supervised by Taihú Pire and Pablo De Cristóforis [18].

The implementation of S-PTAM is open source and is publicly available in `http://github.com/lrse/sptam`. It is built upon the ROS (Robot Operating System) framework to ease distribution and integration.

1.2 Publications

Publications relevant for the thesis:

- T. Pire, T. Fischer, G. Castro, J. Civera, P. De Cristóforis and J. Jacobo Berlles: **S-PTAM: Stereo Parallel Tracking and Mapping**. In Robotics and Autonomous Systems (RAS). September 2016. (Accepted with minor revisions).
- T. Fischer, T. Pire, P. Čížek, P. De Cristóforis and J. Faigl. **Stereo Vision-based Localization for Hexapod Walking Robots Operating in Rough Terrains**. In 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Daejeon, Korea, October 9-14, 2016.
- G. Castro, P. De Cristóforis and T. Pire. **Detección y cierre de ciclos en sistemas SLAM basados en visión estéreo**. In 4to Concurso de Tesis de Licenciatura DC-FCEN-UBA. Ciudad Autónoma de Buenos Aires, Argentina, del 18 al 23 de Julio 2016.
- T. Pire, T. Fischer, J. Civera, P. De Cristóforis and J. Jacobo Berlles: **Stereo Parallel Tracking and Mapping for Robot localization**. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Hamburg, Germany, September 28 - October 02, 2015.
- T. Pire, T. Fischer and J. Faigl: **Impact assessment of image feature extractors on the performance of SLAM systems**. In Proceedings of the Student Conference on Planning in Artificial Intelligence and Robotics. Organized by Czech Technical University, Faculty of Electrical Engineering. Písek, Czech Republic. September 7, 2015.
- T. Pire and J. Jacobo Berlles: **Towards a Stereo Approach to PTAM**. In Joint Conference on Robotics and Intelligent Systems 2014, II Workshop on MSc Dissertation and PhD Thesis in Robotics. São Carlos, Brasil. October 2014.

Other publications during Doctoral career:

- P. De Cristóforis, M. Nitsche, T. Krajník, T. Pire and M. Mejail: **Hybrid vision-based navigation for mobile robots in mixed indoor/outdoor environments**. In Pattern Recognition Letters, vol. 53, pp. 118-128, 2015.
- M. Nitsche, T. Pire, T. Krajník, M. Kulich, and M. Mejail: **Monte Carlo Localization for Teach-and-Repeat Feature-Based Navigation**. In Advances in Autonomous Robotics Systems, Lecture Notes in Computer Science, M. Mistry, A. Leonardis, M. Witkowski, and C. Melhuish, Eds., Springer International Publishing, 2014, vol. 8717, pp. 13-24.
- T. Pire, P. De Cristóforis, M. Nitsche and J. Jacobo Berlles: **Stereo vision obstacle avoidance using disparity and elevation maps**. In IEEE RAS Summer School on "Robot Vision and Applications". VI Latin American Summer School on Robotics, Santiago, Chile. December 3-7, 2012
- T. Pire: **Evasión de obstáculos en tiempo real usando visión estéreo**. In VII Jornadas Argentinas de Robótica 2012. 21, 22 y 23 de Noviembre de 2012. Facultad de Ingeniería, Olavarría, UNICEN. Noviembre 2012.

1.3 Layout of this thesis

In chapter 2 the related work of visual SLAM is described. There, the paradigms to solve various Visual SLAM problems are discussed, and the most important SLAM methods of the literature are analyzed. In chapter 3 the basic concepts used along the whole Thesis are presented. In chapter 4, the proposed SLAM method is detailed. In chapter 5 the proposed method is evaluated using public datasets, and a comparison with other SLAM systems of the state of the art is performed. In addition, an assessment of the impact produced by the most relevant feature extractors in the literature on the proposed SLAM system is presented. In chapter 6 conclusions and future work derived from this Thesis are presented.

Capítulo 1

Introducción

Un sistema de auto-localización y mapeo robusto y preciso es esencial para que un robot pueda realizar la mayoría de las tareas en una amplia variedad de aplicaciones de manera autónoma. Debido al ruido presente en los sensores, la construcción y la actualización de un mapa del entorno tiene que hacerse de forma simultánea con la estimación de la pose del robot. Tal problema se conoce con el acrónimo SLAM, por las siglas en inglés de *Simultaneous Localization and Mapping*, y es objeto de investigación desde las últimas dos décadas.

La mayoría de los primeros trabajos sobre SLAM hicieron uso de un telémetro láser como sensor principal en combinación con la odometría proveniente de los encoders de las ruedas del robot [1]. Más recientemente, los sensores visuales -ya sean pasivos [2] o activos [3]- se han convertido en la opción dominante en el campo de la robótica móvil. Como resultado colateral, la información de odometría se ha vuelto menos relevante, haciendo que los sistemas de SLAM visual puedan utilizarse para otras aplicaciones como Realidad Aumentada y Realidad Virtual. Hoy en día, las cámaras son económicas, pequeñas y ligeras, y proporcionan decenas imágenes de alta resolución por unidad de segundo. Su alcance es ilimitado -bajo el costo de una gran incertidumbre en profundidad para aquellos píxeles con un pequeño paralaje-, en contraste con los límites de rango de los sensores láser. Por otra parte, las cámaras son sensores pasivos, y por lo tanto no interfieren unas con otras, y a diferencia de los sensores basados en visión estructurada (*Structured light range sensors*, SLRS), pueden ser utilizadas tanto en ambientes interiores como exteriores. Estas características hacen de las cámaras la mejor elección para una plataforma robótica móvil de propósito general.

Por las razones anteriores, el SLAM visual (*Visual SLAM*) se ha convertido en uno de los temas más estudiados en la última década. Y hoy en día es posible conseguir resultados sólidos y precisos de SLAM visual en tiempo real. Sin embargo, algunos problemas importantes siguen existiendo, sobre todo para las configuraciones monoculares que son altamente susceptibles a movimientos puros de rotación y no permiten obtener la escala del entorno reconstruido. Entre los escenarios más desafiantes en SLAM visual se pueden mencionar los entornos altamente dinámicos o los movimientos rápidos de cámara. En estos escenarios las cámaras estéreo ofrecen un mayor grado de robustez. La posibilidad de triangulación de la profundidad de una sola vista -y por ende la inicialización de puntos sin demora y con pequeña incertidumbre de puntos cercanos- permite inicializar al sistema de manera robusta y aumentar el mapa con información de profundidad sin retardo. Además, el uso de la información estéreo permite recuperar la escala real del mapa reconstruido, y evitar la deriva de la escala del mismo. Si bien una combinación monocular-inercial (por ejemplo, [4]) también se puede utilizar para extraer la escala real de la escena, el lector debe notar que los parámetros de ambos sensores son complementarios. Los sensores inerciales no son confiables en períodos de movimiento con velocidad constante. Las cámaras estéreo, por otro lado, presentan los mismos problemas que las cámaras monoculares para escenas de gran profundidad -en comparación con el *baseline* de la cámara-. Una combinación estéreo-inercial (como en [5]) se puede utilizar para evitar sus limitaciones individuales. En los últimos años, han surgido los sensores RGB-D que también proporcionan la escala real de la escena observada, y tienen el valor añadido de proveer una medición densa de profundidad de toda la escena observada [6]. Sin embargo, las mediciones de profundidad son de rango limitado y no pueden trabajar bajo la luz solar directa por lo que se limitan a las escenas de interior y carecen de la generalidad de las cámaras estéreo.

En esta Tesis se presenta un sistema de SLAM estéreo capaz de funcionar en tiempo real, denominado S-PTAM (por las siglas en inglés de *Stereo Parallel Tracking and Mapping*). Las cámaras estéreo permiten detectar un mismo punto de referencia (*landmark*) de la escena en ambas imágenes, permitiendo recuperar su profundidad con gran precisión, si el paralaje entre ambas cámaras es suficientemente grande. A medida que el robot se mueve a través del entorno, es posible realizar un seguimiento (*tracking*) de los landmarks reconstruidos frame a frame, y mejorar así la estimación de profundidad de los mismo, al mismo tiempo que es posible estimar la localización del robot. En los experimentos de este trabajo, el uso de una configuración estéreo juega un papel clave dado que en los mismos se presentan algunos casos difíciles como la presencia de objetos dinámicos dinámicos y movimientos de rotación puros.

Los enfoques de SLAM visual basados en características se basan en la calidad y cantidad de características visuales locales (*features*) de la imagen. Por un lado, la precisión de la localización depende en gran medida del despliegue homogéneo de las características en las imágenes, y la capacidad de realizar un seguimiento de ellos durante largos períodos de tiempo, incluso desde diferentes puntos de vista y condiciones de iluminación. Por otra parte, si el número de puntos del mapa crece demasiado, puede disminuir la velocidad de todo el sistema. Para mantener la respuesta del sistema bajo restricciones de tiempo real, algunas imágenes no serán procesadas u otras partes del sistema, como rutinas de optimización, deberán usar menos recursos computacionales. Actualmente, existen varios extractores de características locales de la imagen. Un extractor de características es una combinación de un procedimiento de detección de puntos salientes (llamados *keypoint*s) y el cálculo de una firma única (denominado *descriptor*) para cada punto detectado. Los detectores más utilizados son SIFT [7], SURF [8], STAR [9], GFTT [10], FAST [11], AGAST [12], y el hace relativamente poco tiempo propuesto ORB [13]; mientras que entre los descriptores más utilizados podemos mencionar SIFT, SURF, ORB, BRIEF [14], BRISK [15], y LATCH [16]. En esta Tesis también evaluamos el impacto producido, en términos de precisión y costo computacional, por los diferentes extractores de características del estado del arte en el rendimiento del sistema de SLAM visual propuesto.

Siguiendo el enfoque *Parallel Tracking and Mapping* (PTAM) propuesto en [17], S-PTAM divide el problema de SLAM en dos tareas principales que son llevadas acabo en paralelo: la estimación de la pose de la cámara y la optimización del mapa reconstruido. Estas tareas se ejecutan en dos hilos de ejecución diferentes, y solamente comparten el mapa reconstruido entre ellas. El hilo de *tracking* es el encargado de estima la pose de la cámara para cada nuevo frame estéreo mediante el seguimiento de las características visuales detectadas en las imágenes, y expandir el mapa mediante la triangulación de nuevos puntos 3D. El hilo de *mapping* es el encargdo de mantener y refinar iterativamente el mapa local del robot.

Además, para que S-PTAM provea una localización global consistente en trayectorias de gran escala se presenta un módulo de detección y cierre de ciclos (*Loop Closure*). Dicho modulo permite detectar cuándo el robot se encuentra transitando por un lugar que visitó previamente y realizar una corrección del error de localización acumulado por S-PTAM hasta el momento.

1.1 Contribuciones

La contribución de la Tesis es el diseño e implementación de S-PTAM, un sistema de SLAM visual basado en visión estéreo capaz de trabajar en tiempo real incluso en trayectorias largas. S-PTAM fue desarrollado de tal manera que sea un sistema SLAM estéreo flexible, robusto y preciso. Sus principales características pueden ser resumidas como sigue:

- Se explota el paralelismo natural del problema de SLAM permitiendo correr en tiempo real mientras que se minimiza la dependencia entre los hilos de ejecución

- Las restricciones estéreo se utilizan para la inicialización de un punto, el mapeo y de trackeo mejorando la presición y robustez del sistema.
- Un módulo de detección y corrección de ciclos en tiempo real se incluye en el sistema. La detección de ciclos se lleva a cabo por medio de un análisis basado en apariencia de las imágenes de la trayectoria y la corrección de ciclo mediante la optimización del gráfo de poses del sistema.
- Un proceso de mantenimiento ejecutado en un hilo de ejecución independiente iterativamente ejecuta operaciones de refinamiento de mapa (Bundle Adjustment) en una zona covisible local, lo que mejora la consistencia global.
- Aunque el método funciona con una secuencia de imágenes estéreo como única entrada, se puede proveer también como entrada la odometría de las ruedas de un robot para aumentar la precisión y robustez.
- Festures binarios son utilizados para describir los puntos visuales, reduciendo así el costo búqueda de correspondencias y el espacio de memoria requerido.

El módulo de cierre de ciclos de S-PTAM presentado en esta tesis resulta de la tesis de grado "Detección y cierre de ciclos en sistemas SLAM basados en visión estéreo" de Gastón Ignacio Castro supervisado por Taihú Pire y Pablo De Cristóforis [18].

La implementación de S-PTAM es de código abierto y se encuentra disponible públicamente en `http://github.com/lrse/sptam`. La implementación hace uso del framework ROS (Robot Operating System) para facilitar su distribución e integración.

1.2 Publicaciones

Publicaciones relevantes de la tesis:

- T. Pire, T. Fischer, G. Castro, J. Civera, P. De Cristóforis and J. Jacobo Berlles: **S-PTAM: Stereo Parallel Tracking and Mapping**. In Robotics and Autonomous Systems (RAS). September 2016. (Accepted with minor revisions).
- T. Fischer, T. Pire, P. Čížek, P. De Cristóforis and J. Faigl. **Stereo Vision-based Localization for Hexapod Walking Robots Operating in Rough Terrains**. In 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Daejeon, Korea, October 9-14, 2016.
- G. Castro, P. De Cristóforis and T. Pire. **Detección y cierre de ciclos en sistemas SLAM basados en visión estéreo**. In 4to Concurso de Tesis de Licenciatura DC-FCEN-UBA. Ciudad Autónoma de Buenos Aires, Argentina, del 18 al 23 de Julio 2016.
- T. Pire, T. Fischer, J. Civera, P. De Cristóforis and J. Jacobo Berlles: **Stereo Parallel Tracking and Mapping for Robot localization**. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Hamburg, Germany, September 28 - October 02, 2015.
- T. Pire, T. Fischer and J. Faigl: **Impact assessment of image feature extractors on the performance of SLAM systems**. In Proceedings of the Student Conference on Planning in Artificial Intelligence and Robotics. Organized by Czech Technical University, Faculty of Electrical Engineering. Písek, Czech Republic. September 7, 2015.
- T. Pire and J. Jacobo Berlles: **Towards a Stereo Approach to PTAM**. In Joint Conference on Robotics and Intelligent Systems 2014, II Workshop on MSc Dissertation and PhD Thesis in Robotics. São Carlos, Brasil. October 2014.

Otras publicaciones durante la carrera de doctorado:

- P. De Cristóforis, M. Nitsche, T. Krajník, T. Pire and M. Mejail: **Hybrid vision-based navigation for mobile robots in mixed indoor/outdoor environments**. In Pattern Recognition Letters, vol. 53, pp. 118-128, 2015.
- M. Nitsche, T. Pire, T. Krajník, M. Kulich, and M. Mejail: **Monte Carlo Localization for Teach-and-Repeat Feature-Based Navigation**. In Advances in Autonomous Robotics Systems, Lecture Notes in Computer Science, M. Mistry, A. Leonardis, M. Witkowski, and C. Melhuish, Eds., Springer International Publishing, 2014, vol. 8717, pp. 13-24.
- T. Pire, P. De Cristóforis, M. Nitsche and J. Jacobo Berlles: **Stereo vision obstacle avoidance using disparity and elevation maps**. In IEEE RAS Summer School on "Robot Vision and Applications". VI Latin American Summer School on Robotics, Santiago, Chile. December 3-7, 2012
- T. Pire: **Evasión de obstáculos en tiempo real usando visión estéreo**. In VII Jornadas Argentinas de Robótica 2012. 21, 22 y 23 de Noviembre de 2012. Facultad de Ingeniería, Olavarría, UNICEN. Noviembre 2012.

1.3 Esquema de la Tesis

El capítulo 2 presenta el estado del arte en Visual SLAM. En el mismo se comentan los distintos paradigmas para resolver Visual SLAM, y se analizan los métodos más relevantes de la literatura. En el capítulo 3 se presentan los fundamentos básicos utilizados a lo largo de toda la tesis. En el capítulo 4 se detalla el método SLAM desarrollado. En el capítulo 5 se evalúa el método en datasets de dominio público y se realiza una comparación con otros sistemas de SLAM del estado del arte. Además, se presenta una evaluación del impacto que producen los extractores de características visuales más relevantes de la literatura sobre el sistema de SLAM propuesto. En el capítulo 6 se presentan las conclusiones y trabajo futuro que se derivan de l trabajo de Tesis.

Chapter 2

Visual SLAM: state of the art

In this chapter the most relevant related work in the Visual SLAM field is presented. The problem of SLAM was proposed in the 80's and was addressed using methods based on filters. Filter based approaches were the first to be used to find an answer to SLAM, we can mention the Extended Kalman Filter (EKF) [19] and the Particle Filter [20]. The EKF based approaches (along with its dual Information Filter [21]) gained a strong reputation given their capacity to provide an accurate estimation of the robot pose and of the map of the surroundings, while at the same time providing a joint covariance of the error.

Among the non-parametric filters, we can mention the Particle Filter (a.k.a. Sequential Monte Carlo) and the family of Histogram Filter [22] based approaches. Both methodologies represent multi-modal belief, and are very useful in situations where the distribution of the global uncertainty is a special concern.

Since SLAM is a non-linear problem, filtering based approaches require the linearization of the problem. On one hand, EKF and its variants have the problem that the pose distribution becomes highly non-Gaussian after some time due to the non-linearity of sensor measurements and camera motion. On the other hand, non-parametric filters will eventually fail too, since the required number of particles will exceed all bounds.

In [23] the concept of Visual Odometry for stereo and monocular cameras was introduced for the first time. There, the authors propose a system which extracts features from each incoming image and tracks them along frames. For the monocular case, to estimate the camera pose, first the relative transformation between three of the frames is computed through the 5-point algorithm [24]. Then, the tracked features extracted from the frames are triangulated. As the system keeps tracking features associated to the triangulated points among images, camera pose is estimated using P3P methods [25, 26]. The process continues after new points are created. For the stereo case, the extracted features of the first image pair are triangulated. For the next frames, the features are tracked at image level, allowing us to obtain 3D-2D correspondences with the triangulated points. These correspondences are used to compute the camera pose using a P3P method. As in the monocular case, new points are triangulated after certain number of frames.

In the last decade, methods based on iterative nonlinear optimization were used to address the problem of SLAM. In Computer Vision, one kind of these methods is called *Bundle Adjustment* (BA) [27, 28]. BA is an iterative optimization technique which aims to minimize the distance between the reprojection of the three dimensional model and the associated points in the image. BA methods were computationally expensive, so formerly it was not possible to use them in real time due to hardware limitations. Later, new SLAM methods based on BA took advantage of the independence between localization and map estimation, parallelizing both tasks and obtaining better results reducing computational costs compared to the filter-based approaches.

2.1 Filter based SLAM methods

Early works in visual SLAM were based on probabilistic approaches to estimate the robot pose and map. Among the probabilistic methods, filtering approaches like the Extended Kalman Filter and the Particle Filter were the most used ones.

The Kalman Filter (KF) was invented in the 1950s as a technique for filtering and prediction in linear systems. The KF implements belief computation for continuous states. The EKF is a linearization of the KF for non-linear systems. It calculates an approximation to the true belief and represents this approximation using a Gaussian distribution. The EKF linearizes the observation and dynamic models of the system, and models uncertainties and errors using Gaussian distributions. The complexity of the EKF methods grow quadratically with the number of landmarks used for map representation, making its application prohibitive for large environments.

One of the first works that used the EKF to address the SLAM problem using a monocular camera was [29]. At the same time, the same problem was solved by [30] from the perspective of *Structure from Motion* (SfM) [31]. This approach infers the three-dimensional shape of a moving scene from its two-dimensional images. A mature version of the EKF-SLAM method presented in [29] was published later under the name of MonoSLAM in [2]. This system was one of the first vision systems to run in real-time (30 Hz) on restricted size environments. The authors focused their work in solving issues related to highly dynamic 3D motion, commodity vision-only sensing, processing efficiency and relaxed platform assumptions. To get real-time performance they used an *active* approach to guide feature mapping and measuring. For the same purpose, they also used a general motion model, to smooth camera movements , and a feature initialization approach based on a known target located on the scene. A strong limitation of MonoSLAM (and similar approaches) was that they could only make use of features that were close to the camera relative to its distance of translation, and therefore exhibited significant parallax during motion. The problem comes from the use of the Euclidean XYZ parametrization for feature initialization since position uncertainties for low parallax features are not well represented by the Gaussian distributions implicit in the EKF. To overcome this limitation, [32] proposes to use inverse-depth parametrization that uses the inverse depth of features relative to the camera from which they were first viewed, producing measurement equations with a high degree of linearity. The inverse-depth parametrization allows to work with features at all depths (even up to infinity). Among stereo EKF systems, we can mention [33] which uses inverse-depth parametrization for features with low disparity (farthest points) and Euclidean XYZ parametrization for features with high disparity (closest points). While there has been significant progress addressing the SLAM problem with EKF based approaches, EKF based systems are demonstrated to be inconsistent for SLAM in [34].

As alternative to EKF-based approaches, non-parametric filters were also studied at the same time. One of the first and most popular non-parametric approach is FastSLAM [35]. FastSLAM uses a Rao-Blackwellized (RBPF) representation of the posterior belief, using Particle Filter for the estimation of robot pose and an EKF for each landmark estimation. This approach is capable to work with thousands of landmarks in real-time. In [36], the FastSLAM algorithm is improved in terms of accuracy, efficiency and robustness. Following FastSLAM, which was conceived originally to be used with a laser range-finder as sensor, visual SLAM systems were proposed in the subsequent years. For example, *vSLAM* presented in [37] is a Monocular Visual SLAM based on FastSLAM. It uses SIFT (Scale-Invariant Feature Transform) features [7] to create a topological map. However, as the authors point out, the system has a strong memory consumption per landmark (40 KB to 400 KB depending of the number of visual features associated with the landmark), adding this to the problem of the number of landmarks the system can handle, making it prohibitive for large environments. The work in [38] presents another real-time Monocular Visual SLAM system based on FastSLAM. This system overcomes the cost of the algorithm, which grows rapidly with the number of landmarks. As a result, the system can work with hundreds of landmarks in real-time. Among stereo SLAM systems based on non-parametric filters, we can mention σSLAM [39, 40, 41]. σSLAM is a stereo RBPF-based system, capable of working on trajectories in the order of hundreds of meters and capable of reconstructing a map in the order of thousands of landmarks. However, the system is not capable of running in real-time given that the required computational time increases as the number of landmarks in the map increases.

2.2 Bundle Adjustment based SLAM methods

Note that EKF and non parametric-based approaches have important drawbacks. The former have the problem that the pose distribution will become highly non Gaussian after some time due to the non-linear sensor measurements and camera motion. The latter, as they use particle filters, will fail eventually too since the required number of particles will exceed all bounds.

As an alternative to filter-based approaches, there are techniques that use iterative non-linear optimization methods. These approaches come from the SfM side, where the problem is solved offline and the time is not a constraint. In [42] the Bundle Adjustment (referred there as a *smoothing* technique) used to solve the Visual SLAM problem is introduced in the robotics community. In that paper, the authors report that smoothing-based SLAM methods outperform the EKF-based ones. Expressly, they conclude that smoothing-based SLAM systems are much faster on large-scale problems than the filter-based ones, since smoothing techniques can be used in either batch or incremental mode, and are much better prepared to deal with non-linear processes and measurement models than the EKF. In the same year, a monocular SLAM system using BA techniques to improve the map and the camera pose estimation was presented in [43]. The system selects certain frames (called *keyframes*) for the triangulation of new map points and for the execution of BA. For each frame, the system localizes the camera using the 3D-2D correspondences established between the features extracted in the current frame and the map points observed by the last two keyframes. When a keyframe is selected, a Local BA is carried out considering the last N keyframes and the points viewed by them. However, the systems presented in [42] and [43] were not capable of running in real time due to the high cost of BA.

The first system that allows to work in real-time using BA techniques is PTAM (Parallel Tracking and Mapping) [17]. PTAM is a system designed, originally, to track a hand-held camera in a small *Augmented Reality* workspace. The authors propose to separate tracking and mapping into two separate tasks, allowing them to run in different threads, exploiting the processing power provided by multi-core computers. The first thread deals with the task of robustly tracking hand-held motion, while the other threads produces a 3D map of point features from previously observed video frames (keyframes). Observe that the separation of tracking and mapping was possible thanks to the use of BA techniques which can be applied in incremental batch mode. As a result, PTAM can work with thousands of landmarks in real-time at desktop size scenarios. Clipp et al. [44] proposes a stereo SLAM system strongly related to PTAM, but it can work in real-time at office size scenarios. The authors propose to divide the SLAM task into three different modules: Scene Flow, Visual Odometry and Global SLAM. The Scene Flow module calculates the sparse optical flow and selects keyframes based on the magnitude of the average flow. It then passes the keyframes and the 3D-2D constraints to the Visual Odometry module, which calculates the inter-keyframe motion using a P3P algorithm and passes this motion as well as the 3D points to the Global-SLAM module. The Global-SLAM module then performs loop detection, and global error correction of the map based on the detected loops. The main limitation of the proposed system is that only keyframe poses are computed. Another stereo version of PTAM is presented in [45]. However, just as in the original version of PTAM, the mapping thread creates new map points after performing a Global BA on each incoming keyframe. Therefore, if the map grows large enough, then the mapping will take some time to converge, delaying the creation of new points. In this context, the tracking process will eventually lose any reference to the map, making the system unsuitable for large scale environments. In this Thesis, to guarantee that new points will be available for each tracking iteration, the points are created within the tracking thread.

Large-scale environments are tackled in more recent approaches [46, 47, 48, 49, 50]. In [46] a system called FrameSLAM is proposed. In this work, the local and the global localization problems, for large trajectories, are addressed. The authors use Visual Odometry to determine the incremental motion of a stereo camera while a frame-feature graph is constructed. They propose to remove the frame-feature constraints and to keep only the frame-frame constraints; and after this, they propose to reduce further the number of frames, while still maintaining fidelity to the original system. The resulting map is a pose-graph called *skeleton*. The skeleton allows fast global optimizations and at the same time a good local localization using the local map. The reported results show that FrameSLAM achieves global localization in trajectories longer than 10 km in real-time. In [47], a hybrid representation consisting of a fully metric Euclidean environment map and a topological map, is proposed. This representation allows to work with large-scale environments conserving the metric of it. To perform Global BA, the map is divided into disjoint segments. Then, each segment is optimized individually. Finally, a global segment-wise optimization is performed treating the segments as rigid bodies, obtaining global consistency. Mei et al. present Relative SLAM system RSLAM [48]. In RSLAM, the map is represented as a sequence of relative poses (frames). The landmarks are in the coordinates of the frame that generates them (called *base frame*), and thus, recovering landmark estimates requires to traverse part of the graph. To provide an accurate local map, RSLAM performs a Local BA over the *active region* of frames (the closest frames in terms of distance to the current frame). The active region defines the landmarks that are visible from the current frame, representing the local environment around the robot. Landmarks with base frames belonging to poses that are placed within the active region, are projected into the current frame by composing the transforms along the edges of the graph. In this framework, loop closure consists on creating a new edge in the graph that can then be used to transfer 3D landmark estimates into the current frame and then evaluate their projection in the image. As non-global optimization is performed, the system does not provide a global map consistency. A double window optimization approach for map refinement is proposed to deal with loopy camera motion in [49]. In a loopy camera motion, the number of keyframes at the boundary is relatively large with respect to the total number of keyframes within the active window, and fixing them hampers convergence. The double window optimization approach deals with this kind of movement, defining an inner window and an outer window. The inner window uses point-pose constraints and is supported by the outer window which uses pose-pose constraints. In this way, while the inner window serves to model the local area as accurately as possible, the pose-graph constraints in the outer window stabilize the periphery.

Strasdat et al. in [51] then demonstrate that PTAM-based approaches offer the best accuracy for a given computational budget in comparison with EKF approaches. Moreover, the authors point out that, for both kinds of approaches, increasing the number of features is the most significant way to increase the accuracy. Meanwhile, increasing the number of frames has the main effect of increasing robustness. Once robustness is achieved, a further increase in the number of frames has only a minor effect on accuracy.

Finally, one of the most accurate and robust feature-based SLAM systems that have been proposed recently is [52]. This system, called ORB-SLAM2, builds on the PTAM framework adding state of the art techniques to obtain a robust monocular system, capable of working on large trajectories in real-time. In particular, ORB-SLAM2 uses [53] for loop detection, [54] to overcome scale drift on loop closing, and the covisibility map technique proposed in [49] for large trajectories, among other techniques.

2.3 Direct SLAM methods

New Visual SLAM methods (called *Direct methods*) capable of producing more dense –or even fully dense– maps than the feature-based ones were proposed recently. Direct methods, unlike feature-based methods, do not perform a feature extraction procedure, instead they use gray-level pixel information directly. During the tracking phase, direct methods use the entire image, minimizing the photometric error of the pixels between different frames.

In [55] a dense structure model, that is used to perform stereo SLAM, is presented. The authors propose to model dense environment structure incrementally by robustly integrating disparity maps from current and previous time instants. To refine the disparity model, a Local BA is performed over time to favor consistent 3D structure over noise. To avoid the construction of a 3D map, the SLAM system uses trifocal tensors.

The method DTAM (from Dense Parallel Tracking and Mapping) is presented in [56]. DTAM generates a texture-mapped scene model with millions of points. The model is composed of depth maps built from multiple keyframes. Each keyframe contains an inverse-depth map and the reference RGB image. In this way, a projective photometric cost volume as a function of the inverse depth, is defined for each keyframe. The average photometric error is computed by projecting a point in the volume into each of the overlapping images and summing the L1 norm of the individual photometric errors obtained. Then, to perform the dense mapping, a global energy minimization framework is computed to get the smallest photometric error. The smallest photometric error will occur at the inverse depth corresponding to the true surface. Once the scene model is computed, the dense alignment of the whole image is performed against that model to track the camera motion. To run DTAM in real-time requires a GPU.

A direct monocular SLAM algorithm (called LSD-SLAM) capable of working in large-scale environments is presented in [57]. LSD-SLAM, instead of using all the pixels of the image, operates with those pixels that have a non-negligible image gradient. These pixels are used to continuously estimate a semi-dense inverse depth map for the current frame, this in turn is used to track the motion of the camera using dense image alignment. Unlike DTAM, LSD-SLAM is capable of working in real-time without requiring the use of a GPU. Later, the authors extend LSD-SLAM to support a stereo configuration [58].

Concha and Civera propose a monocular system called DPPTAM (from Dense Piecewise Planar Tracking and Mapping) in [59]. DPPTAM complements the semi-dense mapping presented in [57] with planar surface reconstruction algorithm for low homogeneous-color regions. As a result, a full dense map is generated since both low and high textures regions can be reconstructed. The assumption that homogeneous-color regions belong to approximately planar areas is mostly true in human-environments.

To perform a proper three-dimensional reconstruction of the scene, DTAM and LSD-SLAM use regularization priors to deal with the slight variations of pixel intensities. In this way, to obtain a consistent inverse-depth map, the photometric error cost function comprises a data term and a regularization term that penalizes deviation from a spatially smooth inverse-depth map solution. The regularization term makes the inverse-depth of each pixel does not differ too much from the surrounding inverse-depths. To preserve sharp edges, if two adjacent inverse depth values are statistically different, they do not contribute to one another.

Recently, a Direct Sparse Odometry system was proposed in [60]. This system combines a fully direct probabilistic model (minimizing a photometric error) with consistent joint optimization of all model parameters, including geometry and camera motion. This is achieved in real time by omitting the smoothness prior (regularization term) used by DTAM and LSD-SLAM, and by sampling pixels evenly throughout the images, instead.

Although, using all the data of an image should intuitively result in methods that are more accurate than the feature-based ones, the opposite result has been reported in [50].

Capítulo 2

SLAM visual: Estado del Arte (resumen)

En este capítulo se de analizan los trabajos más relevantes del estado del arte en lo que se refiere a SLAM visual.

El problema de SLAM fue propuesto en la década de los 80 dando lugar a los métodos basados en filtros. Los enfoques basados en filtros bayesianos fueron los primeros en ser utilizados para encontrar una solución al problema de SLAM, entre ellos se pueden mencionar el *Filtro de Kalman Extendido* (EKF) y el *Filtro de Partículas* (PF). Los métodos basados en EKF -o su dual, el *Filtro de Información* (IF)- ganaron una fuerte reputación en la comunidad de róbotica móvil, dada la capacidad de proporcionar una estimación precisa de la pose del robot y el mapa circundante, mientras que al mismo tiempo se provee una región de incertidumbre alrededor de la pose estimada.

Entre los filtros bayesianos no paramétricos, podemos mencionar al Filtro de Partículas (también conocido en la literatura como *Sequential Monte Carlo*) y la familia de filtros basados en histograma. Ambas metodologías representan una creencia multimodal y son útiles en situaciones donde la distribución de la incertidumbre global es un problema a abordar.

Dado que SLAM es un problema no lineal, los enfoques basados en filtros requieren la linearización del problema. Por un lado, EKF y sus variantes tienen el problema de que la distribución de probabilidad correspondiente a la pose se vuelve altamente no gaussiana después de cierto tiempo debido a la no linealidad de las mediciones del sensor y al movimiento de la cámara. Los filtros no paramétricos eventualmente también fallarán, ya que el número requerido de partículas superará todo límite estipulado.

En los últimos años, el uso de cámaras como sensores para resolver el problema de SLAM se ha visto incrementado notoriamente, debido a la gran portabilidad, el bajo costo comercial y la gran cantidad de información que estas capturan del entorno. En particular, este crecimiento dio lugar al término de SLAM visual (*Visual SLAM*). Uno de los enfoques más relevantes dentro de SLAM visual es el basado en el uso de características visuales extraídas de las imágenes. Los métodos basados en características visuales minimizan el error de reproyección definido como la distancia euclídea entre la proyección de un punto del mapa y su correspondiente característica en la imagen.

Junto con el creciente uso de los métodos basados en características visuales ha surgido un nuevo paradigma basado en el uso de técnicas de optimización no lineal (*Bundle Adjustment*) para resolver el problema de SLAM. Dicho paradigma permite desacoplar la estimación de la pose de la cámara, de la refinación del mapa. Los métodos basados en este enfoque escalan mejor que los sistemas basados en filtros permitiendo ser aplicados en entornos de grandes dimensiones. Uno de los primeros sistemas de SLAM visual que siguieron este enfoque es PTAM (*Parallel Tracking and Mapping*). PTAM separa la tarea de localización y mapeo en dos hilos de ejecución diferentes, tomando ventaja de los procesadores con dos núcleos o más. Siguiendo a PTAM, nuevos y mejores sistemas de SLAM visual basados en características (tanto monoculares como estéreos) se han desarrollado dentro del campo de la robótica móvil para abordar el problema de la localización de un robot para recorridos de largas distancias.

Recientemente, se han propuesto nuevos métodos de SLAM visual (denominados *Métodos Directos*) capaces de producir mapas más densos -o incluso completamente densos- que los basados en características. Los métodos directos, a diferencia de los métodos basados en características, no realizan un procedimiento de extracción de características, sino que usan información de los píxeles de la imagen directamente. De esta manera, durante la etapa de *tracking*, en vez de minimizar el error de reproyección, estos minimizan el error fotométrico de los píxeles entre sucesivos frames.

Chapter 3

Preliminaries

In this chapter the mathematical and geometrical framework that is used along the whole thesis is presented. The material presented in this chapter is a compilation of different sources which complement each other, and it is already well known in the field of robotics and computer vision. But it is presented here for the sake of completeness of this thesis.

3.1 Single View

In this section we present the basic concepts of camera geometry and single view geometry. In particular, we describe how a 3D point in the world is projected to the image plane of a camera. Moreover, we discuss the limitations of single view to perform a 3D reconstruction of the observed scene.

3.1.1 Camera Model

In Computer Vision and related fields, the camera is defined as a mapping between the 3D world and a 2D image. A camera can be represented with different models. The most simple and the most widely used is the *pinhole camera model*.

In the pinhole camera model, the image point $\mathbf{u} = \begin{bmatrix} u & v \end{bmatrix}^\top$ is the determined as the intersection between the image plane and the ray that joins the world point $\mathbf{x} = \begin{bmatrix} x & y & z \end{bmatrix}^\top$ and the projection center $\mathbf{o}$. Figure 3.1 shows the geometry relations involved in the pinhole camera model.

We denote with $\dot{\mathbf{x}} = \begin{bmatrix} x & y & z & 1 \end{bmatrix}^\top$ the homogeneous representation of a point $\mathbf{x} = \begin{bmatrix} x & y & z \end{bmatrix}^\top$. Now, we define the *camera model*:

$$\mathbf{u} = f \operatorname{proj}(\mathbf{x}^{\mathrm{c}}) + \mathbf{c}, \tag{3.1}$$

where f is the focal length, $\mathbf{c}$ is the principal point, $\mathbf{x}^{\mathrm{c}}$ is the point in the camera coordinate system and the function $\operatorname{proj} : \mathbb{R}^n \to \mathbb{R}^{n-1}$ can be used to transform points from homogeneous to inhomogeneous coordinates, and also can be used to project points from 3D space to 2D image plane,

$$\operatorname{proj}(\mathbf{a}) = \frac{1}{a_n} \begin{bmatrix} a_1 \\ \vdots \\ a_{n-1} \end{bmatrix}.$$

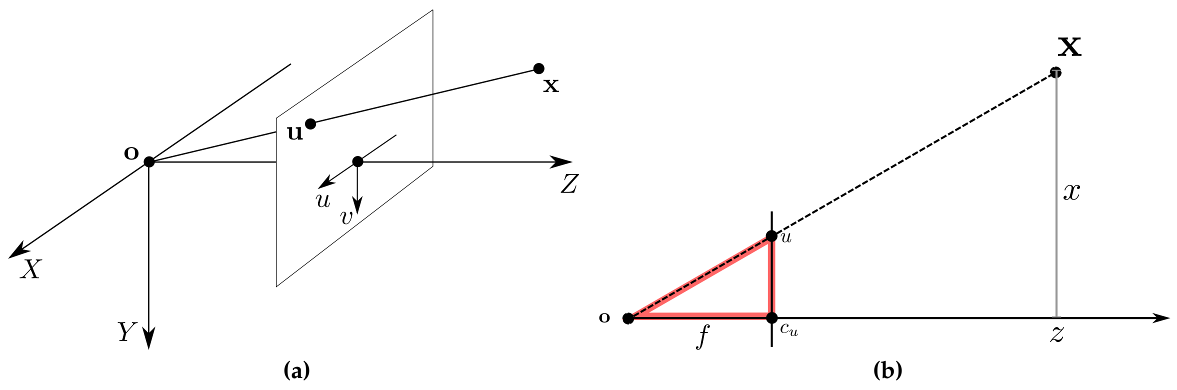


Figure 3.1: Pinhole camera model.

For example, if $\mathbf{x} = \begin{bmatrix} x & y & z \end{bmatrix}^\top$, then $\mathrm{proj}(\mathbf{x}) = \begin{bmatrix} x/z & y/z \end{bmatrix}$. Figure 3.1b shows the geometry involved in the projection of a 3D point onto the 2D image plane. It can be seen that $x/z = u/f$. Analogously we can say that $y/z = v/f$. From these equations we can see that $u = f(x/z)$ and that $v = f(y/z)$, so ecuation (3.1) holds.

In the general case, the camera is not located at the center of the world coordinate system. Therefore we have to add this transformation between the world and camera coordinate systems to the previous equation, yielding

$$\mathbf{u} = f\,\mathrm{proj}(\mathrm{proj}(\mathbf{E}^{\mathrm{cw}}\dot{\mathbf{x}}^{\mathrm{w}})) + \mathbf{c},$$

where $\mathbf{E}^{\mathrm{cw}}$ is a transformation matrix that transforms geometric elements from the world coordinate frame W to the camera coordinate frame C. This is,

$$\dot{\mathbf{x}}^{\mathrm{c}} = \mathbf{E}^{\mathrm{cw}}\dot{\mathbf{x}}^{\mathrm{w}}.$$

In particular, $\mathbf{E}^{\mathrm{cw}}$ is a transformation belonging to the Lie Group **SE**(3), the group of rigid-body motions in 3D space. Therefore, $\mathbf{E}^{\mathrm{cw}}$ is defined as $\mathbf{E}^{\mathrm{cw}} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}$ where $\mathbf{R}$ is a rotation matrix and $\mathbf{t}$ is a translation vector. Observe that one of the advantages of working with homogeneous coordinates is that this allows to use the **SE**(3) Lie Groups algebra.

There exists a matrix notation for the pinhole camera model which is also commonly used in computer vision

$$\mathbf{P} = \mathbf{K}\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}, \tag{3.2}$$

where $\mathbf{P}$ and is the *projection matrix* and $\mathbf{K}$ is the *calibration matrix*. The calibration matrix is defined as,

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

being $\begin{bmatrix} c_u & c_v \end{bmatrix}^\top$ the position of the principal point. So, using the projection matrix $\mathbf{P}$, the world point $\mathbf{x}^{\mathrm{w}}$ is mapped to the image point $\mathbf{u}$ by

$$\mathbf{u} = \mathrm{proj}\,(\mathbf{P}\dot{\mathbf{x}}^{\mathrm{w}}).$$

3.1.2 Camera Undistortion

In Section 3.1.1 we considered that the pinhole camera model is perfectly linear, this is, the world point, the image point and the optical center are collinear. This assumption is not true in practice because there is a radial distortion in the image produced by the real camera lens. Figure 3.2 presents some common distortions generated by the lens.

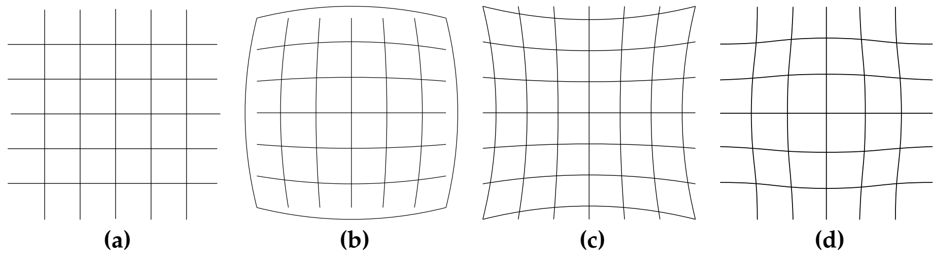


Figure 3.2: Distortion types: **(a)** no distortion, **(b)** barrel distortion, **(c)** pincushion distortion and **(d)** mustache distortion.

To correct this distortion a *radial distortion model* is computed which related image points in the real image with the ideal points, those that would have been obtained under a perfect linear camera. In this way, the camera can be represented with a linear pinhole model. In Figure 3.3 is possible to observe raw (distorted) image taken by the camera and the image after the undistortion.

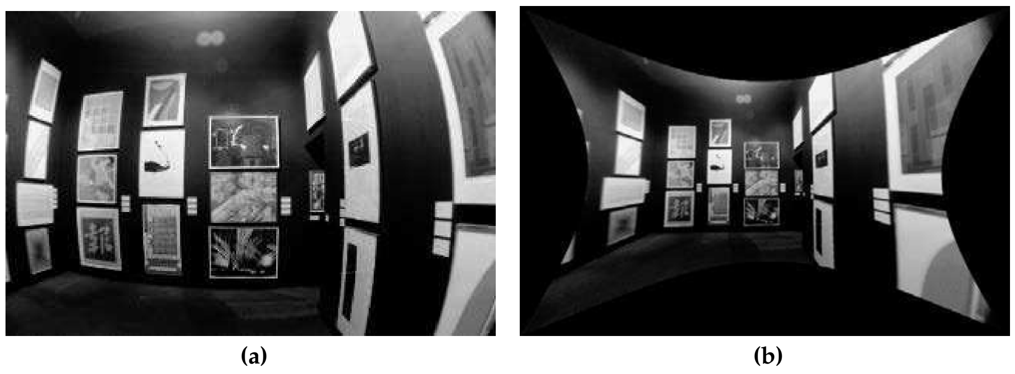


Figure 3.3: Image undistortion.

Given the actual projected point and the ideal point, the radial distortion effect can be modeled by

$$\begin{bmatrix} u_d \\ v_d \end{bmatrix} = L(\tilde{r}) \begin{bmatrix} \tilde{u} \\ \tilde{v} \end{bmatrix}$$

where, $\begin{bmatrix} \tilde{u} \\ \tilde{v} \end{bmatrix}$ is the ideal image position referred to the centre of the image (which obeys linear projection); $\begin{bmatrix} u_d \\ v_d \end{bmatrix}$ is the actual image position after radial distortion; $\tilde{r}$ is the radial distance $\sqrt{\tilde{u}^2 + \tilde{v}^2}$ from the center for radial distortion, and $L(\tilde{r})$ is a distortion factor, which is a function of the radius $\tilde{r}$.

Finally, the correction can be represented in pixel coordinates as

$$\hat{u} = u_c + L(r)(u - u_c)$$
$$\hat{v} = v_c + L(r)(v - v_c)$$

where $\begin{bmatrix} u \\ v \end{bmatrix}$ are the measured coordinates, $\begin{bmatrix} \tilde{u} \\ \tilde{v} \end{bmatrix}$ are the corrected coordinates, and $\begin{bmatrix} u_c \\ v_c \end{bmatrix}$ is the center of radial distortion, with $r^2 = (u - u_c)^2 + (v - v_c)^2$.

The distortion factor $L(r)$ is defined only for positive values of r and $L(0) = 1$. In practice, $L(r)$ is approximated by Taylor expansion

$$L(r) = 1 + k_1 r + k_2 r^2 + k_3 r^3 + \ldots.$$

The coefficients $\{k_1, k_1, k_1, \ldots, u_c, v_c\}$ are considered part of the internal calibration of the camera. Frequently, the principal point is used as the center for radial distortion, though these need not coincide exactly. The correction coefficients, together with the camera calibration matrix, specifies the mapping from an image point to a ray in the camera coordinate system, and are called *intrinsic parameters*.

3.1.3 Back projection

Given a point $\mathbf{u}$ in the image, it is possible to determine the set of points in space that map to this point. This set of points constitutes the ray in space passing through the camera center and the image point $\mathbf{u}$. So, the ray $\mathbf{x}(\lambda)$ can be defined as

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

where $\mathbf{P}^{+}$ is the *pseudo-inverse*[1] of $\mathbf{P}$ and $\mathbf{o}$ is the camera center. Figure 3.4 shows the back projection of an image point. Observe that the ray $\mathbf{x}(\lambda)$ is the line that joins the points $\mathbf{P}^{+}\mathbf{u}$ and $\mathbf{o}$ in the 3D space given that $\mathbf{P}\mathbf{P}^{+}\mathbf{u} = \mathbf{I}\mathbf{u} = \mathbf{u}$ and $\mathbf{P}\mathbf{o} = \mathbf{0}$.

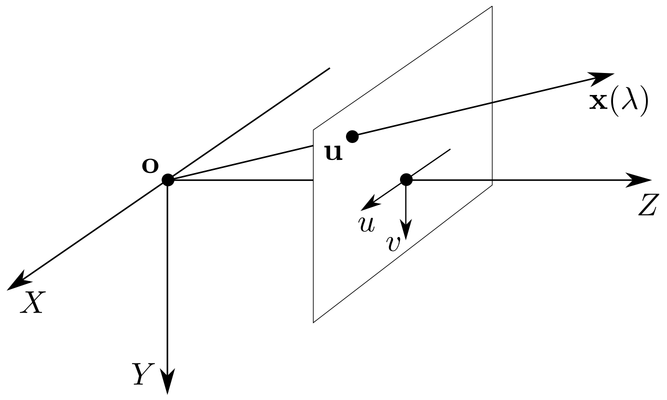


Figure 3.4: Back projection of an image point.

From above the reader must notice that it is not possible to perform a 3D reconstruction from a single view. In section 3.3, we will see how it is possible to perform a 3D reconstruction of the captured scene from two different views, when the displacement between both cameras is known. Note that we will need to

[1]The pseudo-inverse matrix $\mathbf{P}^{+}$ is defined as $\mathbf{P}^{+} = \mathbf{P}^{\top}\left(\mathbf{P}\mathbf{P}^{\top}\right)^{-1}$.

match the image points -from both views- that are associated with the same 3D point. Find this correspondences is not direct and it will explained in the next section.

3.2 Appearance based matching

In this section, we introduce two kinds of appearance based matching methods that are used in this thesis: *Image Features* and *Bag-of-Words*. The former summarizes the image information keeping only those image points that have high intensity gradient values. These image points can be compared easily between two different images to determine if they belong to the same 3D object. The Bag-of-Words matching, on the other hand, represents each image as a set of "visual words" and uses them to compute a global descriptor of the image. This global descriptor can be used to compare images allowing us to recognize, for example, if the images were taken at the same location.

3.2.1 Image Features

The matching problem motivates us to perform some image processing in order to detect those points that can be easily distinguished. This salient points on an image are called *features*, and the process of obtaining them is called *feature extraction*. In this way, feature extraction involves reducing the amount of data provided by images to only focus on those pixels that can be easily matched. Feature extraction algorithms are the combination of two stages: *detection* and *descrip*tion. The detection stage consists of the detection of point of interest, also called *salient points* or *keypoints*, of an image. This type of points correspond to image areas in which gradient is high. The description stage is in charge of describing univocally the detected features in order to establish correspondences of features between images. In particular, in this stage each region around the detected keypoint locations is converted into a more compact and stable (invariant) descriptor that can be matched against other descriptors. The matching between two descriptors succeeds if the distance between them is less than a given threshold. To compute the distance, the Hamming norm is customarily used for binary descriptors and the norm L_2 for floating point vector descriptors.

In the literature exists a large variety of feature extractors. Among the most used feature extractors we can mention SIFT [7], SURF [8], ORB [13], KAZE [61] and its accelerated version A-KAZE [62]. Moreover, there are algorithms developed exclusively for the detection stage, as Shi–Tomasi [10], STAR [9], FAST [11] and AGAST [12]; and for the description stage, as BRIEF [14], BRISK [15], FREAK [63], LATCH [16], LDB [64], LUCID [65] and BinBoost [66]. In Figure 3.5 the points detected by a Shi–Tomasi feature detector on a sample image are shown.

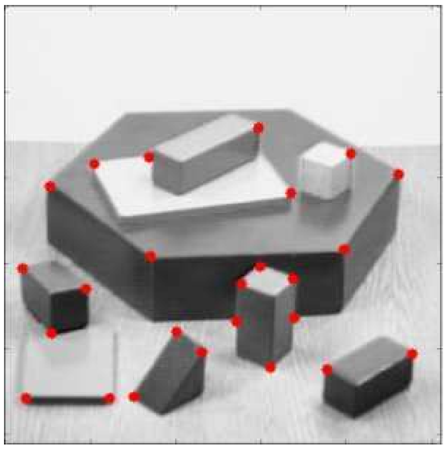


Figure 3.5: Shi–Tomasi features (corners) detected on a sample image.

3.2.2 Bag of words

In robotics and computer vision, usually it is required to efficiently compare an image against a set of images. This situation occurs, for example, when a robot (with a camera as sensor) must recognize a previously visited place. This problem has the name of *Loop Closure* in the literature. *Bag-of-Words* is an efficient matching method to compare an image against a bunch of images in terms of appearance [67].

During the training stage, the image feature descriptors of a set of training images are extracted. Figure 3.6 is a pictorial representation of the positions of the extracted feature points within the descriptor's feature space. Then, the set of feature descriptors is partitioned into *clusters*. The clusters are computed incrementally using *k-means*, this is, firstly the set of descriptors is partitioned into k_w clusters. For each resulting cluster, the most representative descriptor is selected as the center of the cluster. Then, each cluster is sub-partitioned into k sub-clusters. This process continues until it reaches a given level (L). Figure 3.7 shows how the clustering process works. The result of the clustering is a tree where the nodes are the representative descriptors selected in each subdivision. This tree is called *vocabulary tree* and its leafs are called *words*. Figure 3.8 illustrates the vocabulary tree resulting from the clustering. Observe that the tree will have a total number of k^L words. During the construction of the vocabulary tree, a weight is assigned to each computed word. This weight represents how much discriminative or relevant the word is in the training sequence. The more occurrences a word has, the less discriminative it will be and therefore the less weight it will have. In contrast, words with few occurrences will be more heavily weighted.

Let w_i be a word, its associated weight will be computed as

$$idf(i) = \log\left(\frac{N}{n_i}\right), \tag{3.3}$$

where N is the total number of images in the training sequence and n_i is number of occurrences of the word w_i in the whole training sequence.

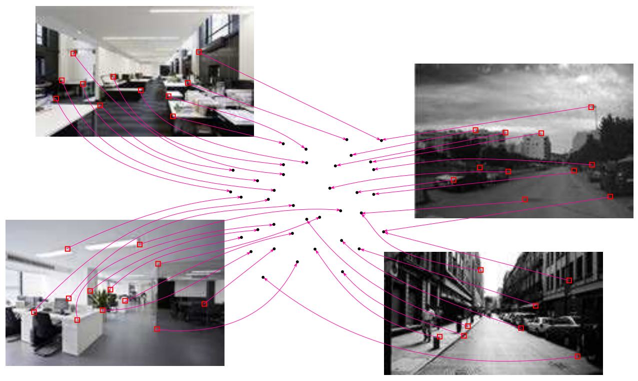


Figure 3.6: Extraction of descriptors from four sample images. The pink arrows indicate the association of each image feature with its corresponding descriptor. The set of points that are located in the middle of this figure reside in the descriptors space.

Once the vocabulary tree is computed from the training set, it can be used to process new query images. This is, given a query image I_t in time t, first, the feature descriptors are extracted. Then, each descriptor

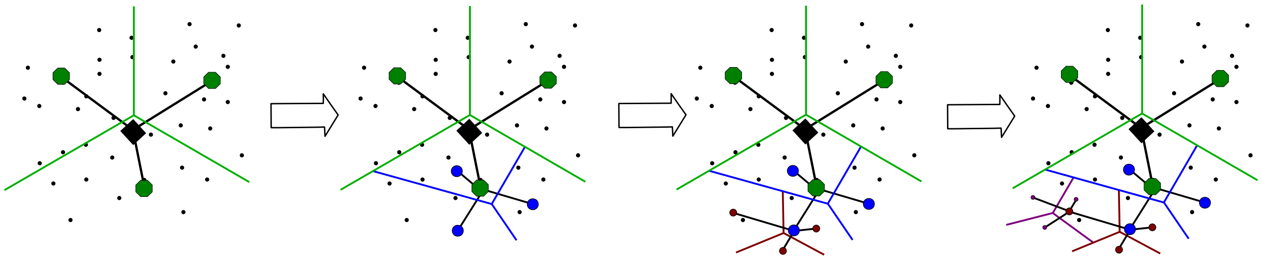


Figure 3.7: Bag of words clustering. The nodes represent cluster centers.

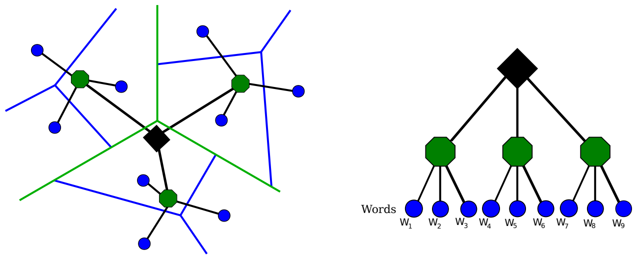


Figure 3.8: Vocabulary tree.

of the image is passed trough the vocabulary tree, following at each level the branch associated to the node which has less distance to the descriptor. In this way, it is possible to obtain the words presents in the image. Having the words present in the image with their associated weights, we can determine their relevance in the image I_t. To do this, the *term frecuency* (*tf*) for each word w_i present in image I_t is computed as

$$tf(i, I_t) = \frac{n_{iI_t}}{n_{I_t}}, \tag{3.4}$$

where n_{iI_t} is the number of occurrences of the word w_i in the image I_t and n_{I_t} is the total number of words found in I_t.

Finally, it is possible to calculate the *bag-of-words vector* v_t [67] associated to each query image. Given an image I_t, the bag-of-word vector, $v_t \in \mathbb{R}^W$, is calculated as follows

$$(v_t)_i = tf(i, I_t)idf(i) \qquad i \in 1, \ldots, W. \tag{3.5}$$

Figure 3.9 illustrates the process of calculating the bag-of-words vector for a given image. This bag-of-words vector allows to compare two images in terms of similarity.

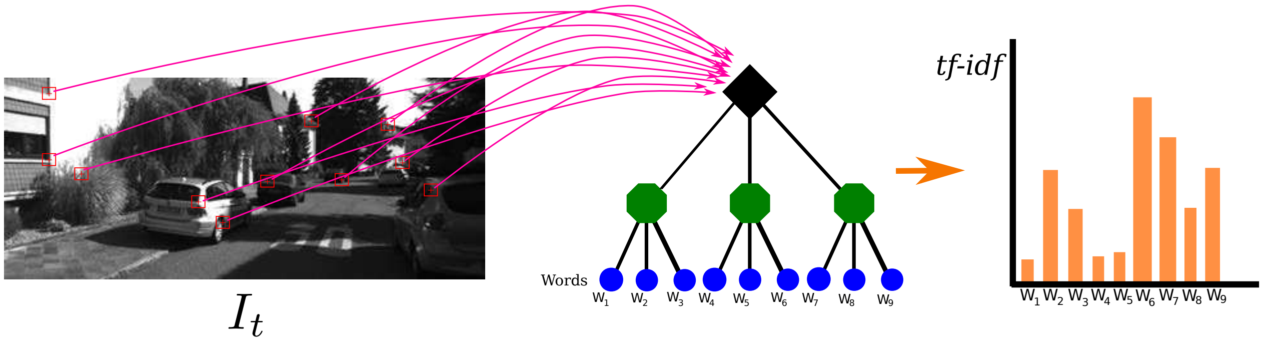


Figure 3.9: Processing a query image in the vocabulary tree.

3.3 Two views

As we saw in section 3.1.3, it is not possible to perform a 3D reconstruction of an image point from a single view. To estimate a 3D reconstruction of an scene at least two views are required. To make the things more clear, we need to find points that correspond to the same 3D point in both images (*stereo matching*). In this section we introduce the necessary concepts to understand 3D reconstruction from two views.

3.3.1 Epipolar geometry and Fundamental matrix

The *epipolar geometry* between two views is essentially the geometry of the intersection of the image planes with the pencil of planes that have the baseline as axis (the baseline is the line joining the camera centers). Figure 3.10 shows this intersection.

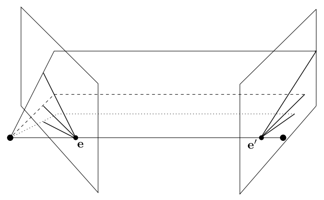


Figure 3.10: Pencil of planes intersecting the image planes. Image adapted from [68].

A given point $\mathbf{x}$ in 3D, it is imaged in two views, at $\mathbf{u}$ on the first image plane, and $\mathbf{u}'$ on the second image plane. The image points $\mathbf{u}$ and $\mathbf{u}'$, space point $\mathbf{x}$, and camera centers are coplanar, and lie in the plane denoted by $\boldsymbol{\pi}$. In the same way, the rays back-projected from $\mathbf{u}$ and $\mathbf{u}'$ intersect at $\mathbf{x}$, the rays are coplanar, and lie on plane $\boldsymbol{\pi}$. Figure 3.11 shows the epipolar plane and geometry entities just are described.

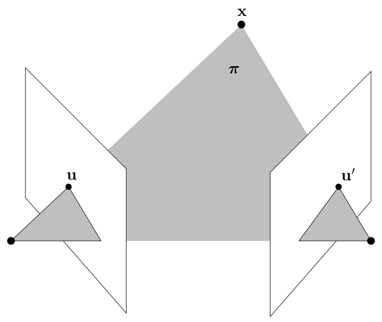


Figure 3.11: Epipolar plane. Image adapted from [68].

In the situation where $\mathbf{u}$ is only given, the search of the corresponding point $\mathbf{u}'$ in the second image can be constrained. This is, the plane $\boldsymbol{\pi}$ is determined by the baseline and the ray defined by $\mathbf{u}$. From the preceding explanation, we know that the ray corresponding to the (unknown) point $\mathbf{u}'$ lies on $\boldsymbol{\pi}$, hence the point $\mathbf{u}'$ lies on the line of intersection $\mathbf{l}'$ of $\boldsymbol{\pi}$ with the second image plane. This line $\mathbf{l}'$ is the image of the ray back-projected from $\mathbf{u}$ onto the second image plane. $\mathbf{l}'$ is called the *epipolar line* corresponding to $\mathbf{u}$. For any point $\mathbf{u}$, the search for its corresponding point $\mathbf{u}'$ needs not to cover the entire image plane and can be restricted to the epipolar line $\mathbf{l}'$. Figure 3.12 shows the epipolar line resulting from the projection of the ray back projected from $\mathbf{u}$ onto the second image plane.

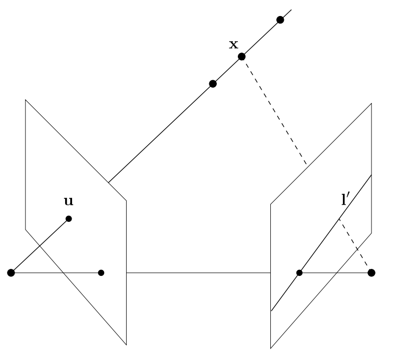


Figure 3.12: The epipolar line resulting from the projection of the ray back project by $\mathbf{u}$. Image adapted from [68].

The fundamental matrix $\mathbf{F}$ is the algebraic representation of epipolar geometry. It is a 3×3 matrix of rank 2, that maps a point on one image plane to its corresponding epipolar line on the other image plane,

$$\mathbf{l}' = \mathbf{F}\mathbf{u}.$$

If a point $\mathbf{x}$ in 3D is imaged as $\mathbf{u}$ in the first image plane, and $\mathbf{u}'$ in the second, then these image points satisfy the relation

$$\mathbf{u}'^{\top}\mathbf{F}\mathbf{u} = 0.$$

The Fundamental matrix allows us to restrict the search for stereo correspondence between two images by means of the epipolar lines. However, it must be noticed that the restricted search over epipolar lines does not ensure a correct matching between image points. For example, in areas of images with low texture (for example a white wall), it is not possible to identify which point from the first image corresponds to other point in the second image using straightforward raw image information.

3.3.2 Stereo rectification

Stereo image rectification consists of projecting images onto a common image plane in such a way that the corresponding points (points that correspond to the same 3D point) have the same row coordinates. This projection makes the image appear as though the two cameras are parallel. This is, the transformation between both cameras is a pure translation in the horizontal axis, called *baseline*. In Figure 3.13, a rectified stereo camera is shown. Observe that, to carry out the stereo rectification, both images must be previously undistorted.

In this way, after stereo rectification, corresponding points will have the same row component. Figure 3.14b shows the result of the rectification process on a sample of real stereo images. This result is particularly useful for the computation of *disparity*. The disparity is defined as the distance between two corresponding points on the left and on the right image of a stereo pair. Using disparity, it is possible to estimate the 3D position of the corresponding point.

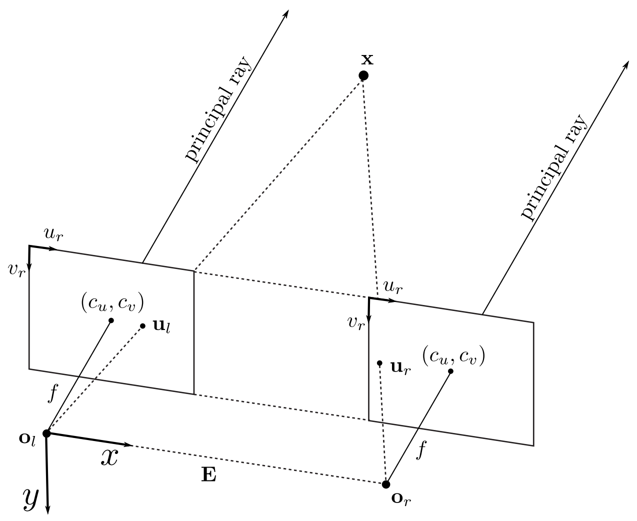


Figure 3.13: Stereo rectification. Image adapted from [68].

3.3.3 Stereo triangulation

After stereo rectification is carried out, it is possible to perform a 3D reconstruction from the corresponding image points found during the stereo matching process. Figure 3.15 shows the geometry involved during the triangulation of a point. The 3D position of a point $\mathbf{x}$ can be derived by means of the properties of similar triangles.

Using the similar triangles property between the black dashed line triangle and the red line triangle of Figure 3.15a, the following equation can be derived:

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

so

$$d = u_l - u_r = \frac{bf}{z}, \tag{3.6}$$

where d is the disparity. In the same way, using again the properties of similar triangles from Figure 3.15b, we obtain the following equations

$$\frac{x}{z} = \frac{u_l - c_u}{f} \qquad \frac{y}{z} = \frac{v_l - c_v}{f}. \tag{3.7}$$

From (3.6) and (3.7) we can get expressions for x, y and z, given by

$$\begin{aligned} x =& \frac{(u_l - c_u)z}{f}, \\ y =& \frac{(v_l - c_v)z}{f} \text{ and} \\ z =& \frac{bf}{u_l - u_r}, \end{aligned} \tag{3.8}$$

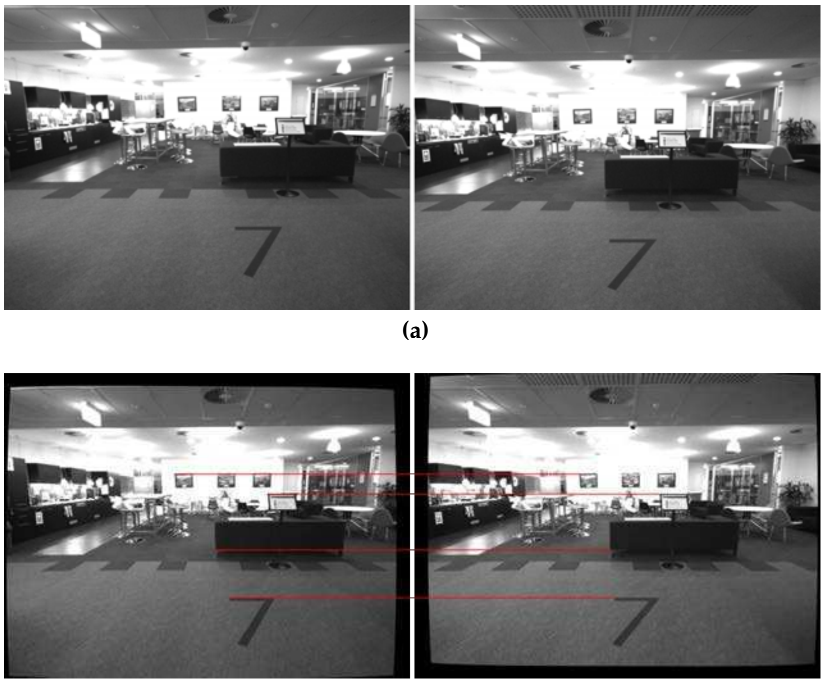


Figure 3.14: Stereo rectification on a sample of real stereo images. **(a)** The raw stereo images provided by the stereo camera. **(b)** The stereo images after rectification. Red lines represent the feature correspondences between both images.

respectively. From (3.8) it is possible to define the re-projection matrix **Q** which transforms vectors $\begin{bmatrix} u_l & v_l & d & 1 \end{bmatrix}^\top$ into 3D points in homogeneous coordinates:

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

where

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

3.4 Multiple views

In sections 3.1 and 3.3 we covered the basic concepts of working with one and two views. In this section, we introduce the different approaches to estimate the 3D geometry of the observed scene and the pose of the cameras involved in a multiple view case (two or more views).

3.4.1 Tensors (three and four views)

As in the case of two-view geometry, *Tensors* play the same role as the Fundamental matrix but for three and four views. Tensors are independent of the scene structure, depending only on the (projective) relations

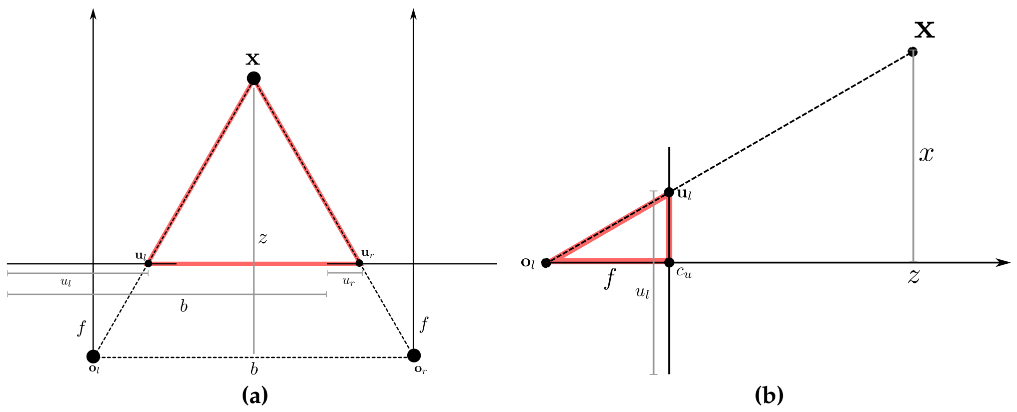


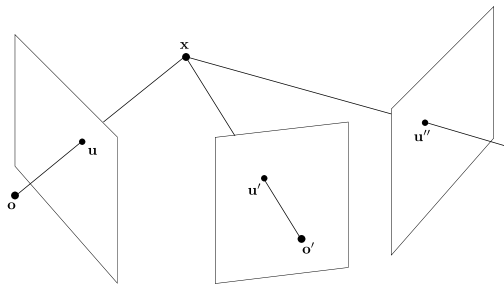
Figure 3.15: Stereo Triangulation.

between the cameras. For the case of three-view geometry, the tensor is called *Trifocal tensor*. The main difference between Trifocal tensor geometry compared with the two-view case is that the Trifocal tensor can determine the position of an entity presented in other two views in a third view. For example, given a point correspondence found in two views, the trifocal tensor determines the position in a third view. The same applies to lines. This transfer property, which arises from the incidence relations in 3D space, is useful to establish correspondences over multiple views. Figure 3.16 shows examples of incidence relations.

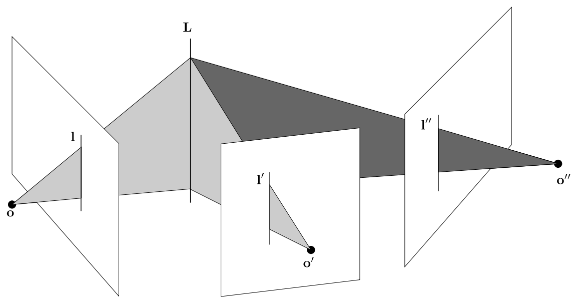
The trifocal tensor consists of three 3×3 matrices, $\mathbf{T}_1$, $\mathbf{T}_2$ and $\mathbf{T}_3$. It has 27 elements but only 18 degrees of freedom.

In what follows we list the incidence relations supported by trifocal tensors.

Line-line-line:	$\mathbf{l}' \left[\mathbf{T}_1 \quad \mathbf{T}_2 \quad \mathbf{T}_3\right] \mathbf{l}'' = \mathbf{l}^\top$	
Point-line-line:	$\mathbf{l}'^\top \left(\sum_i x_i \mathbf{T}_i\right) \mathbf{l}'' = \mathbf{0}^\top$	for a correspondence $\mathbf{x} \longleftrightarrow \mathbf{l}' \longleftrightarrow \mathbf{l}''$.
Point-line-point:	$\mathbf{l}'^\top \left(\sum_i x_i \mathbf{T}_i\right) \left[\mathbf{x}''\right]_\times = \mathbf{0}^\top$	for a correspondence $\mathbf{x} \longleftrightarrow \mathbf{l}' \longleftrightarrow \mathbf{x}''$.
Point-point-line:	$\left[\mathbf{x}'\right]_\times \left(\sum_i x_i \mathbf{T}_i\right) \mathbf{l}'' = \mathbf{0}$	for a correspondence $\mathbf{x} \longleftrightarrow \mathbf{x}' \longleftrightarrow \mathbf{l}''$.
Point-point-point:	$\left[\mathbf{x}'\right]_\times \left(\sum_i x_i \mathbf{T}_i\right) \left[\mathbf{x}''\right]_\times = \mathbf{0}_{3\times3}$	



(a) Point-point-point incidence relation.



(b) Line-line-line incidence relation.

Figure 3.16: Examples of incidence relations of tensors. Images adapted from [68].

For the case of four-view geometry, the tensor is called *Quadrifocal tensor*. The quadrifocal tensor encapsulates the relationships between imaged points and lines seen in four views. It is important to mark that tensors are independent of the scene structure observed, they only depend on the motion between views and the internal parameters of the cameras.

In the following section we are going to generalize the problem of the 3D reconstruction to a non fixed number of cameras.

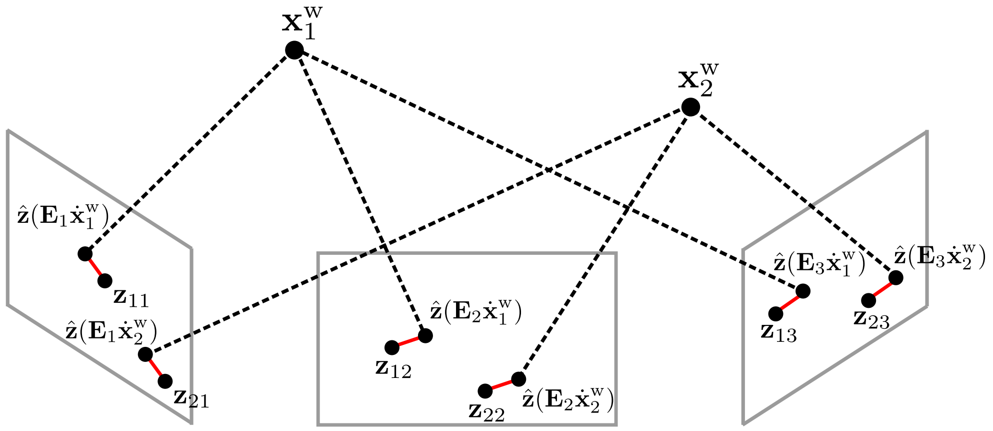
3.4.2 Bundle Adjustment

Bundle Adjustment[2] (BA) refers to the problem of refining a visual reconstruction to produce jointly optimal 3D structure and viewing parameter (camera pose and/or calibration) estimates. Although it is possible to estimate the intrinsic parameters of the involved cameras during BA, henceforth we are going to consider them fix and equal for all cameras. The BA problem can be then instantiated as the problem of, given a set of 3D points $\{\mathbf{x}_i^{\mathrm{w}}\}$, a set of camera poses $\{\mathbf{E}^{\mathrm{c}_j\mathrm{w}}\}$ and a set of measurements $\{\mathbf{z}_{ij}\}$ of the i-th point in the j-th camera, obtaining the best configuration of the parameters that minimizes the error between the observations (measurements) and the fitted values (projections). To compute this optimal solution, a cost function is minimized, typically using Gauss-Newton or Levenberg-Marquardt iteration algorithms described in (3.5).

In BA, the $u(.)$ vector-valued function in (3.13) is the reprojection error and is defined as

$$u(\mathbf{E}^{\mathrm{c}_j\mathrm{w}}, \mathbf{x}_i^{\mathrm{w}}) = \mathbf{z}_{ij} - \hat{\mathbf{z}}(\mathbf{E}^{\mathrm{c}_j\mathrm{w}}\dot{\mathbf{x}}_i^{\mathrm{w}}). \tag{3.9}$$

Observe that $\mathbf{E}^{\mathrm{c}_j\mathrm{w}}, \mathbf{x}_i^{\mathrm{w}}$ are the set of parameters that we want to estimate and $\hat{\mathbf{z}}(.)$ is a camera projection model. Figure 3.17 illustrates an example of BA problem where two world points are observed from three different views. In this figure, we have removed the coordinate systems associated to the camera poses for easy reading, so $\mathbf{E}_j$ represents the camera pose $\mathbf{E}^{\mathrm{c}_j\mathrm{w}}$. Since BA uses iterative non-linear optimization algorithms to minimize a given cost function, intrinsically, BA based methods require to have a good initial seed in order not to converge to a local minimum.



$\mathbf{x}_1^{\mathrm{w}}$
$\mathbf{x}_2^{\mathrm{w}}$
$\hat{\mathbf{z}}(\mathbf{E}_1\dot{\mathbf{x}}_1^{\mathrm{w}})$
$\mathbf{z}_{11}$
$\hat{\mathbf{z}}(\mathbf{E}_1\dot{\mathbf{x}}_2^{\mathrm{w}})$
$\mathbf{z}_{21}$
$\hat{\mathbf{z}}(\mathbf{E}_2\dot{\mathbf{x}}_1^{\mathrm{w}})$
$\mathbf{z}_{12}$
$\mathbf{z}_{22}$
$\hat{\mathbf{z}}(\mathbf{E}_2\dot{\mathbf{x}}_2^{\mathrm{w}})$
$\hat{\mathbf{z}}(\mathbf{E}_3\dot{\mathbf{x}}_1^{\mathrm{w}})$
$\hat{\mathbf{z}}(\mathbf{E}_3\dot{\mathbf{x}}_2^{\mathrm{w}})$
$\mathbf{z}_{13}$
$\mathbf{z}_{23}$

Figure 3.17: Example of Bundle Adjustment, two world points are observed from three different views. Bundle Adjustment minimizes the reprojection error (red segment) defined by the euclidean distance between the points' projections and the corresponding observations in the images. We have removed the coordinate systems associated to the camera poses for easy reading.

On the other hand, the Bundle Adjustment problem can be represented as a graph optimization problem where the nodes are the camera poses and the map point positions; and the edges are the measurements. Figure 3.18 illustrates the structure of a sample graph. In this figure, in addition to the simplification in the notation introduced in Figure 3.17, we also have removed the indication of the coordinate system for the 3D points, so the $\mathbf{x}_i^{\mathrm{w}}$ are rewritten as $\mathbf{x}_i$.

[2]The name refers to the 'bundles' of light rays leaving each 3D feature and converging on each camera center, which are 'adjusted' optimally with respect to both feature and camera positions.

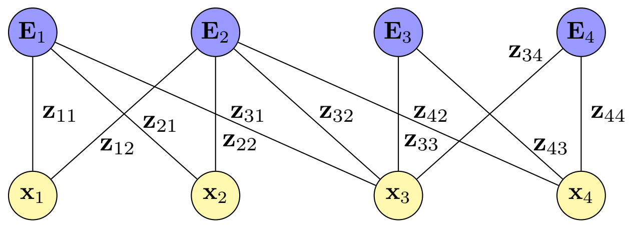


Figure 3.18: Point-Pose Graph representation of Bundle Adjustment. We have removed the coordinate systems associated to the camera poses and points for easy reading.

In Visual SLAM methods, generally not all points are viewed by all cameras. BA implementations can take advantage of this lack of interaction between parameters to speed up the computation. Moreover, Visual SLAM methods apply BA incrementally to run in real-time, i.e., instead of adjusting all points and cameras together, only the subset of cameras and points that are close to the current camera are considered to be adjusted. However, in Loop Closure situations, where the number of cameras and points to be adjusted is too high, the application of a full BA is intractable. In the next section, we present a common approach to deal with long loop closures optimization.

3.4.3 Pose-graph

To deal with long loop closures in graph based Visual SLAM systems, it is convenient to convert the pose-point graph in a pose-pose graph (frequently called *pose-graph*). To get the pose-pose graph from a point-pose graph the points are marginalized in a way that measurements that relate two global poses $\mathbf{E}^{\mathrm{c}_i\mathrm{w}}$ and $\mathbf{E}^{\mathrm{c}_j\mathrm{w}}$ are replaced by a single relative transformation constraint $\mathbf{E}^{\mathrm{c}_j\mathrm{c}_i}$. The relative transformation is given by

$$\mathbf{E}^{\mathrm{c}_j\mathrm{c}_i} = \mathbf{E}^{\mathrm{c}_j\mathrm{w}}(\mathbf{E}^{\mathrm{c}_i\mathrm{w}})^{-1}.$$

Figure 3.19 depicts the result of the marginalization of the measured points $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ shared by the two poses $\mathbf{E}^{\mathrm{c}_i\mathrm{w}}$ and $\mathbf{E}^{\mathrm{c}_j\mathrm{w}}$ by means of the solid line between the nodes $\mathbf{E}_i$ and $\mathbf{E}_j$ that represents the relative transformation $\mathbf{E}^{\mathrm{c}_j\mathrm{c}_i}$ computed between them. Figure 3.20 illustrates an example of a pose-point graph on a loop trajectory and the generated pose-graph after the marginalization of all the measured points. The edges on Figure 3.20b are added when the corresponding poses have a measured point in common in Figure 3.20a.

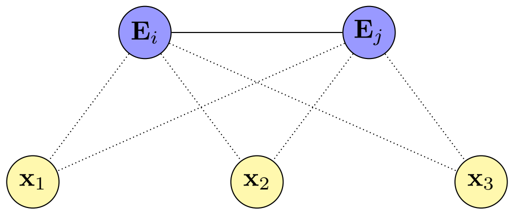


Figure 3.19: Result of the marginalization of the points $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ shared by the two poses $\mathbf{E}^{\mathrm{c}_i\mathrm{w}}$ and $\mathbf{E}^{\mathrm{c}_j\mathrm{w}}$ by means of the solid line between the nodes $\mathbf{E}_i$ and $\mathbf{E}_j$.

Observe that in Loop Closure, the relative transformation between the two poses that close the loop, represents the error accumulated along the loop. During a pose-graph optimization, this relative transformation is kept fixed, and remaining relative transformations are adjusted in such way that $\mathbf{E}^{\mathrm{c}_j\mathrm{c}_i}\mathbf{E}^{\mathrm{c}_i\mathrm{w}}(\mathbf{E}^{\mathrm{c}_j\mathrm{w}})^{-1}$

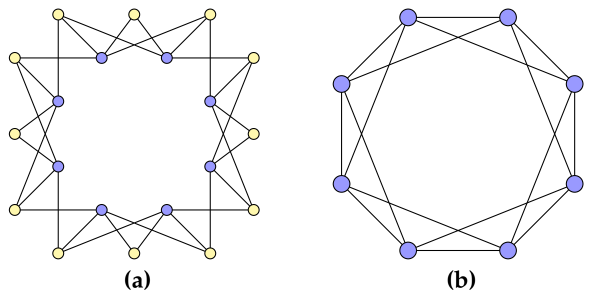


Figure 3.20: Generation of a pose-graph from a pose-point graph. **(a)** Pose-point graph. **(b)** Pose-graph. Blue nodes represent poses and yellow nodes represent measured points.

is close to the identity. In the next section, we present the Perspective-*n*-Point (P*n*P) family methods that can be used to compute the relative transformation between the two poses that close a loop.

3.4.4 Perspective-*n*-Point

In section 3.4.3, we anticipated the requirement of computing the relative transformation that closes a loop in the Loop Closure problem. Generally, in the Loop Closure scenario, the loop detection is carried out using appearance based methods (see section 3.2.2) between the current image and the previous processing images, and therefore there is no geometry information to estimate such transformation between cameras. However, in Visual SLAM methods there are 3D points that are shared by both cameras and can be used to estimate the relative transformation between them. One option is to use the *Iterative Closest Point* (ICP) algorithm introduced in [69] using the 3D point clouds observed by both cameras. However, the ICP algorithm requires a good initialization in order not to converge to a local minimum.

Other approaches that can be used as alternatives to ICP are those used to solve the Perspective-*n*-Point (P*n*P) problem. Formally, P*n*P is the problem of estimating the pose of a calibrated camera given a set of *n* 3D points (called *control points*) and their corresponding 2D image points. In a Loop Closure scenario it is possible to establish the 3D-2D correspondences that relate the 3D points measured by one camera with the corresponding image points measured by the other camera. Figure 3.21 depicts the P*n*P problem. This problem consists in estimation of the camera pose $\mathbf{E}^{\mathrm{cw}}$ given a number of 3D-2D correspondences between points $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_n$ in the world frame and bearing vectors $\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3 \ldots, \mathbf{f}_n$ in the camera frame. In the literature, there is a variety of methods to solve this problem [70, 71, 72, 73]. Generally, to achieve a better camera pose estimation, the solution returned by the PnP method is usually followed by a non-linear optimization.

In particular, when $n = 3$, the P*n*P problem is in the minimal form (called P3P) and can be solved with 3 points. However, the P3P problem will have four solutions. In practice, a fourth point is used to find the best solution among the four. Again, there are several methods to solve the P3P problem in the literature, here we can mention [25, 26].

3.5 Iterative optimization algorithms

In this section we present the iterative optimization algorithms used to solve non-linear least square problems. In Visual SLAM, these algorithms are used to perform *Bundle Adjustment*.

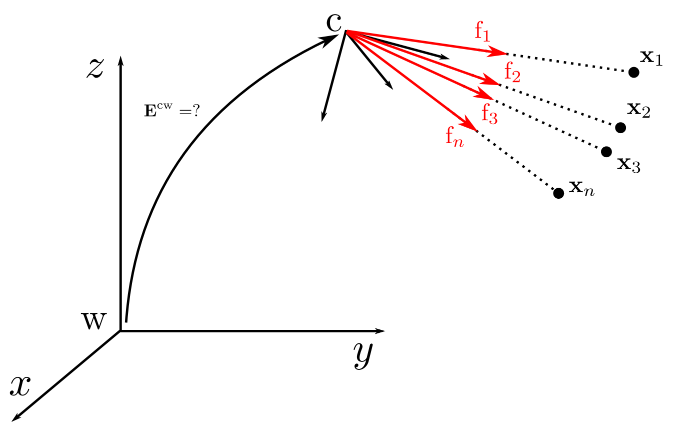


Figure 3.21: P*n*P problem instantiation. The camera pose $\mathbf{E}^{\mathrm{cw}}$ can be computed given a number of 3D-2D correspondences between points $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_n$ in the world frame and bearing vectors $\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3 \ldots, \mathbf{f}_n$ in the camera frame.

The most simple way to minimize a nonlinear function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is to start with an initial value $\mathbf{x}_0$ and successively update $\mathbf{x}$ to $\mathbf{x}_1, \mathbf{x}_2, \ldots$, such that $f(\mathbf{x})$ decreases at each iteration; that is, $f(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_k)$. One way to ensure this decrease is to follow the direction of descent, which is the opposite direction of the gradient vector. This method of searching for the minimum is called *steepest descent method*. At each iteration,

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k), \tag{3.10}$$

where α_k is usually a small constant called *step size* and the factor $-\nabla f(\mathbf{x}_k)$ is a vector that points to the steepest descent direction locally around $\mathbf{x}_k$. However, the steepest descent direction is not necessarily the best method to achieve the minimum of a function at large scale. A generalization of (3.10) is

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{D}_k \nabla f(\mathbf{x}_k), \tag{3.11}$$

where $\mathbf{D}_k \in \mathbb{R}^{n \times n}$ is a positive definite symmetric matrix and its value determines the particular search algorithm in use. For example, the steepest descent method is a particular case when $\mathbf{D}_k = \mathbf{I}$. In general, $\mathbf{D}_k$ can be viewed as a weight matrix that adjusts the descent direction according to more sophisticated local information about $f(.)$ than the gradient alone.

In practice, for each iteration the incremental update $\boldsymbol{\delta}$ is defined as

$$\boldsymbol{\delta} = \mathbf{x}_{k+1} - \mathbf{x}_k.$$

We can perform a minimization method by repetitively solving the linear system

$$\boldsymbol{\delta} = -\alpha_k \mathbf{D}_k \nabla f(\mathbf{x}_k),$$

followed by an additive update

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \boldsymbol{\delta}. \tag{3.12}$$

3.5.1 Newton method

The necessary and sufficient optimality conditions suggest that, around a (local) minimum, the function $f(.)$ resembles a quadratic function. That is, $f(\mathbf{x})$ can be approximated by a Taylor polynomial of grade 2,

$$f(\mathbf{x}) \approx f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^\top(\mathbf{x} - \mathbf{x}_k) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_k)^\top \nabla^2 f(\mathbf{x}_k)(\mathbf{x} - \mathbf{x}_k).$$

Therefore, by differentiating $f(.)$ and equating to zero we can obtain the next value $\mathbf{x}_{k+1}$. This is,

$$\nabla f(\mathbf{x}_k)^\top + \nabla^2 f(\mathbf{x}_k)(\mathbf{x}_{k+1} - \mathbf{x}_k) = 0.$$

Then, by solving for $\mathbf{x}_{k+1}$ we obtain the iterative step of the Newton method,

$$\mathbf{x}_{k+1} = \mathbf{x}_k - (\nabla^2 f(\mathbf{x}_k))^{-1} \nabla f(\mathbf{x}_k).$$

As in (3.11), it is possible to generalize the iteration step adding a step scale α_k

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k (\nabla^2 f(\mathbf{x}_k))^{-1} \nabla f(\mathbf{x}_k).$$

Note Newton's method is equal to (3.11) taking

$$\mathbf{D}_k = (\nabla^2 f(\mathbf{x}_k))^{-1}.$$

However, is not always feasible to apply Newton's method. For instance, computation of the *Hessian matrix* $\nabla^2 f(\mathbf{x}_k)$ is often expensive and sometimes not possible (e.g., the function $f(.)$ may not be twice differentiable).

3.5.2 Gauss-Newton

In cases where Newton's method is not practicable, alternatives to $\mathbf{D}_k$ are often adopted that, to some extent, approximate the matrix $\nabla^2 f(\mathbf{x}_k)^{-1}$ and use only the information encoded in the first derivative of $f(\mathbf{x})$. The *Gauss-Newton method* is one such method, which applies, however, only to the problem of minimizing *the sum of squares of real-valued functions*. This is, the objective function must have the form

$$f(\mathbf{x}) = \frac{1}{2}\|u(\mathbf{x})\|^2 = \frac{1}{2}\sum_{i=1}^{m} u_i(\mathbf{x})^2 \tag{3.13}$$

where $u(\mathbf{x}) = \begin{bmatrix} u_1(\mathbf{x}), & u_2(\mathbf{x}), & \dots, & u_m(\mathbf{x}) \end{bmatrix}^\top$ is a vector-valued function and $\|.\|$ is the 2-norm. Moreover, the first derivative is given by $\nabla f(\mathbf{x}) = \nabla u(\mathbf{x}) u(\mathbf{x})$ and the *Hessian matrix* is defined as

$$\nabla^2 f(\mathbf{x}_k) = \nabla u(\mathbf{x}_k) \nabla u(\mathbf{x}_k)^\top + \nabla^2 u(\mathbf{x}_k) u(\mathbf{x}_k)^\top \tag{3.14}$$

where $\nabla u(\mathbf{x})$ is called *Jacobian Matrix* and has the form of:

$$\nabla u(\mathbf{x}) = \begin{bmatrix} \frac{\partial u_1(\mathbf{x})}{\partial \mathbf{x}_1} & \frac{\partial u_2(\mathbf{x})}{\partial \mathbf{x}_1} & \dots & \frac{\partial u_m(\mathbf{x})}{\partial \mathbf{x}_1} \\ \frac{\partial u_1(\mathbf{x})}{\partial \mathbf{x}_2} & \frac{\partial u_2(\mathbf{x})}{\partial \mathbf{x}_2} & \dots & \frac{\partial u_m(\mathbf{x})}{\partial \mathbf{x}_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial u_1(\mathbf{x})}{\partial \mathbf{x}_n} & \frac{\partial u_2(\mathbf{x})}{\partial \mathbf{x}_n} & \dots & \frac{\partial u_m(\mathbf{x})}{\partial \mathbf{x}_n} \end{bmatrix} \in \mathbb{R}^{n \times m}.$$

For faster convergence to the minimum, the Gauss-Newton method makes the assumption that $f(\mathbf{x})$ is linear, so the second term on the right of (3.14) vanishes. Therefore, Gauss-Newton instead of $\mathbf{D}_k = (\nabla^2 f(\mathbf{x}_k))^{-1}$ uses

$$\mathbf{D}_k = (\nabla u(\mathbf{x}_k) \nabla u(\mathbf{x}_k)^\top)^{-1}.$$

Then, the iteration step on the Gauss-Newton method takes the form of

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k(\nabla u(\mathbf{x}_k)\nabla u(\mathbf{x}_k)^\top)^{-1}\nabla u(\mathbf{x}_k)u(\mathbf{x}_k).$$

The Gauss-Newton method is an approximation to Newton's method, especially when the value $\|u(\mathbf{x})\|^2$ is small. Here, we use the standard 2-norm to measure $u(\mathbf{x})$. The method only needs to be slightly changed when a different quadratic norm, say $\|u(\mathbf{x})\|_W = u(\mathbf{x})^\top \mathbf{W}u(\mathbf{x})$, for some positive definite symmetric matrix $\mathbf{W} \in \mathbb{R}^{m\times m}$, is used. We address this topic in section 3.5.4.

3.5.3 Levenberg-Marquardt

The *Levenberg-Marquardt* (LM) iteration method is a slight variation of the Gauss-Newton iteration method. Its only difference from the Gauss-Newton method is that it sets $\alpha_k = 1$ and uses

$$\mathbf{D}_i = (\lambda_k\mathbf{I} + \nabla u(\mathbf{x}_k)\nabla u(\mathbf{x}_k)^\top)^{-1},$$

for some value of $\lambda_k > 0$ that varies from iteration to iteration. A typical initial value of λ_k is 10^{-3} times the average of the diagonal elements of $\nabla u(\mathbf{x})\nabla u(\mathbf{x})^\top$. The value of λ_k changes given the following rules:

- If the current value of λ_k results in a decrease in the error, then the iteration is accepted and λ_k is divided by 10 and is used as the value for the next iteration.
- If λ_k results in an increase in the error, then it is multiplied by 10, and the iteration is tried again until a λ_k is found that results in a decrease in the error.

Intuitively, when λ_k gets close to 0 then the LM performs as the Gauss-Newton method, when λ_k is large, the LM performs as the steepest descent method.

3.5.4 M-estimators

In estimation problems of the Newton or Levenberg-Marquardt type, an important decision to make is to specify the form of the cost function. In (3.13), the cost function is assumed to be quadratic making the effect of outliers so strong in the minimization that the estimated parameters are distorted. A popular robust technique to deal with outliers is the so-called *M-estimators*. The M-estimators try to reduce the effect of outliers by replacing the squared residuals in (3.13), by another function of the residuals, yielding

$$f(\mathbf{x}) = \|u(\mathbf{x})\|_\rho = \sum_{i=1}^{m} \rho(u_i(\mathbf{x})), \tag{3.15}$$

where $\rho(.)$ is a symmetric, positive-definite function with a unique minimum at zero, and is chosen to grow slower than a quadratic function. Note that $\rho(x) = \frac{x^2}{2}$ in (3.13). Instead of solving directly this problem, we can implement it as an *iterated re-weighted least-squares* one. To this end, we need to rewrite the derivative of (3.15), as

$$\nabla f(\mathbf{x}) = \sum_{i=1}^{m} \frac{\partial\rho(u_i(\mathbf{x}))}{\partial\mathbf{x}} = \sum_{i=1}^{m} \psi(u_i(\mathbf{x}))\frac{\partial u_i(\mathbf{x})}{\partial\mathbf{x}}, \tag{3.16}$$

where $\psi(u) = \frac{\partial\rho}{\partial u}$ is called *influence function*. Then, we can rewrite (3.16), as

$$\nabla f(\mathbf{x}) = \sum_{i=1}^{m} w(u_i(\mathbf{x}))u_i(\mathbf{x})\frac{\partial u_i(\mathbf{x})}{\partial\mathbf{x}} = \sum_{i=1}^{m} w(u_i(\mathbf{x}))u_i(\mathbf{x})\nabla u_i(\mathbf{x}),$$

where $w(u) = \frac{\psi(u)}{u}$ is *called weight function*. This is exactly the system of equations that we obtain if we solve the following *iterated re-weighted least-squares* problem

$$f(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{m} w_i u_i^2(\mathbf{x}) = \frac{1}{2}u(x)\mathbf{W}u(\mathbf{x})^\top,$$

where w_i is the weight scalar value computed using the residual of the previous iteration. In the vector multiplication representation, matrix $\mathbf{W}$ is a diagonal matrix with w_i coefficients.

The influence function $\psi(u)$ measures the influence of a datum on the value of the parameter estimate. For example, for the least-squares estimation problems with $\rho(x) = \frac{x^2}{2}$, the influence function is $\psi(u) = x$, so the influence of a datum on the estimate increases linearly with the size of its error, which confirms the non-robustness of the least-squares estimate. When an estimator is robust, the influence function limits the effect of outliers on the estimation. The most used M-estimators with their respectively ρ, ψ and w functions are presented in Table 3.1. Furthermore, Table 3.2 shows graphically the functions ρ, ψ and w of each M-estimator.

Type	ρ-function	ψ (influence) function	w (weight) function
Least-squares (L_2)	$\frac{x^2}{2}$	x	1
Least-absolute (L_1)	$\lvert x\rvert$	$sgn(x)$	$\frac{1}{\lvert x\rvert}$
$L_1 - L_2$	$2\left(\sqrt{1+\frac{x^2}{2}}-1\right)$	$\frac{x}{\left(\sqrt{1+\frac{x^2}{2}}\right)}$	$\frac{1}{\left(\sqrt{1+\frac{x^2}{2}}\right)}$
Least-power (L_p)	$\frac{\lvert x\rvert^v}{v}$	$sgn(x)\lvert x\rvert^{v-1}$	$\lvert x\rvert^{v-2}$
Fair	$c^2\left(\frac{\lvert x\rvert}{c} - \log\left(1+\frac{\lvert x\rvert}{c}\right)\right)$	$\frac{x}{1+\frac{\lvert x\rvert}{c}}$	$\frac{1}{1+\frac{\lvert x\rvert}{c}}$
Huber $\begin{cases}\lvert x\rvert \le k \\ \lvert x\rvert \ge k\end{cases}$	$\begin{cases}\frac{x^2}{2} \\ k\left(\lvert x\rvert - \frac{k}{2}\right)\end{cases}$	$\begin{cases}x \\ k\,sgn(x)\end{cases}$	$\begin{cases}1 \\ \frac{k}{\lvert x\rvert}\end{cases}$
Cauchy	$\frac{c^2}{2}\log\left(1-\left(\frac{x}{c}\right)^2\right)$	$\frac{x}{1+\left(\frac{x}{c}\right)^2}$	$\frac{1}{1+\left(\frac{x}{c}\right)^2}$
Geman -McClure	$\frac{\frac{x^2}{2}}{1+x^2}$	$\frac{x}{(1+x^2)^2}$	$\frac{1}{(1+x^2)^2}$
Welsch	$\frac{c^2}{2}\left(1-\exp\left(-\left(\frac{x}{c}\right)^2\right)\right)$	$x\exp\left(-\left(\frac{x}{c}\right)^2\right)$	$\exp\left(-\left(\frac{x}{c}\right)^2\right)$
Tukey $\begin{cases}\lvert x\rvert \le c \\ \lvert x\rvert > c\end{cases}$	$\begin{cases}\frac{c^2}{6}\left(1-\left(1-\left(\frac{x}{c}\right)^2\right)^3\right) \\ \frac{c^2}{6}\end{cases}$	$\begin{cases}x\left(1-\left(\frac{x}{c}\right)^2\right)^2 \\ 0\end{cases}$	$\begin{cases}\left(1-\left(\frac{x}{c}\right)^2\right)^2 \\ 0\end{cases}$

Table 3.1: Most used M-estimators.

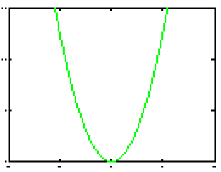
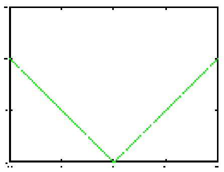
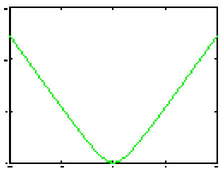
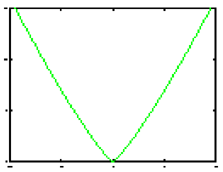
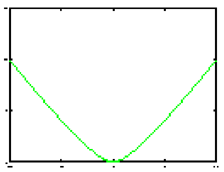
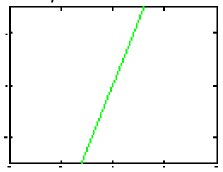
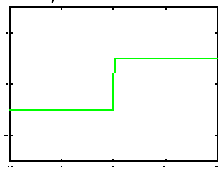
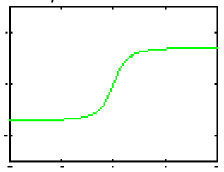
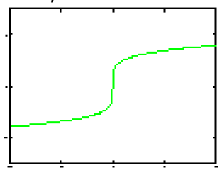
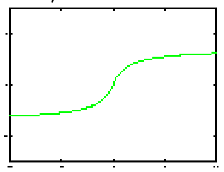
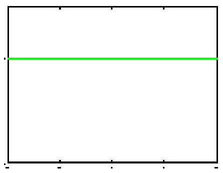
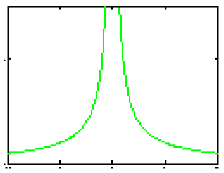
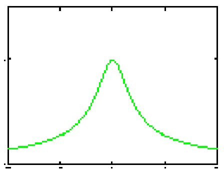
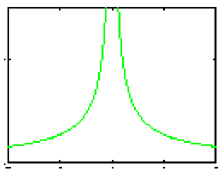
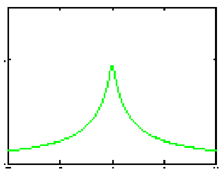
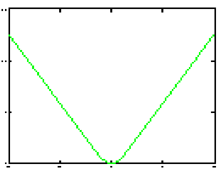
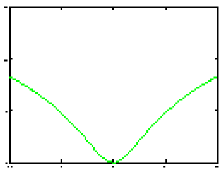
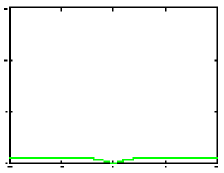
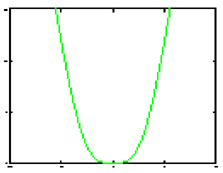
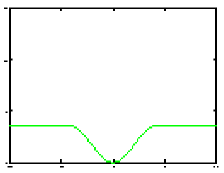
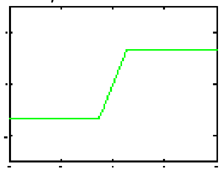
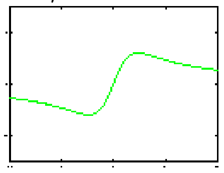
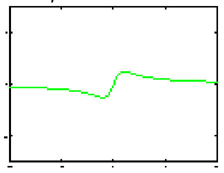
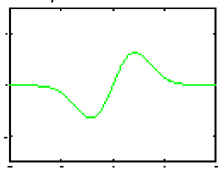
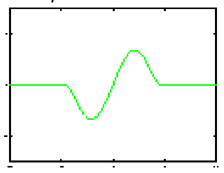
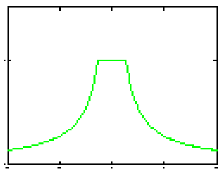
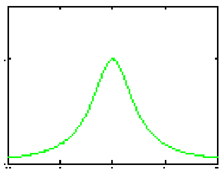
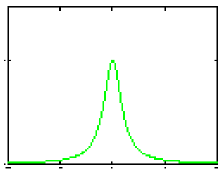
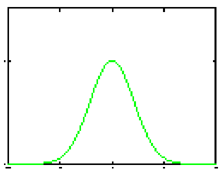
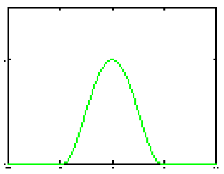
Least-squares	**Least-absolute**	$L_1 - L_2$	**Least-power**	**Fair**
				
ρ-function	ρ-function	ρ-function	ρ-function	ρ-function
				
ψ (influence) function	ψ (influence) function	ψ (influence) function	ψ (influence) function	ψ (influence) function
				
w (weight) function	w (weight) function	w (weight) function	w (weight) function	w (weight) function
Huber	**Cauchy**	**German-McClure**	**Welsch**	**Tukey**
				
ρ-function	ρ-function	ρ-function	ρ-function	ρ-function
				
ψ (influence) function	ψ (influence) function	ψ (influence) function	ψ (influence) function	ψ (influence) function
				
w (weight) function	w (weight) function	w (weight) function	w (weight) function	w (weight) function

Table 3.2: Most used M-estimators.

3.6 Lie Groups

In mobile robotics and computer vision is critical to have a mathematical framework capable to represent transformations of moving objects in the space. *Lie groups* can be used to represent transformations in 2D and 3D space. In particular, we are interested in working in the 3D space to represent camera pose and its motion, and thus only 3D transformations will be considered. In this context, the Lie groups address the operations of composition, inversion, differentiation and interpolation needed for the calculus of camera movement.

For instance, Lie groups allow us to represent rigid transformations in 3D space. The group of rigid transformations in 3D space, is denoted **SE**(3), and is well represented by affine transformations:

$$\mathbf{E} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix},$$

where $\mathbf{R}$ is a rotation matrix and $\mathbf{t}$ a translation vector. Note that $\mathbf{R} \in$ **SO**(3), where **SO**(3) is the Lie group of 3D rotation matrices, and $\mathbf{t} \in \mathbb{R}^3$.

This representation allows us to perform composition and inversion of rigid transformations using the standard matrix multiplication and inversion operations, respectively.

Given the 3D rigid transformations $\mathbf{E}_1$and $\mathbf{E}_2$, the composition operation is defined as

$$\begin{aligned} \mathbf{E}_1\mathbf{E}_2 &= \begin{bmatrix} \mathbf{R}_1 & \mathbf{t}_1 \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_2 & \mathbf{t}_2 \\ \mathbf{0} & 1 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{R}_1\mathbf{R}_2 & \mathbf{R}_1\mathbf{t}_2 + \mathbf{t}_1 \\ \mathbf{0} & 1 \end{bmatrix}. \end{aligned}$$

In the same way, the inversion of a 3D rigid transformation $\mathbf{E}$, is given by

$$\mathbf{E}^{-1} = \begin{bmatrix} \mathbf{R}^\top & -\mathbf{R}^\top\mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \quad .$$

Moreover, the **SE**(3) group can be used to act over 3D points (in homogeneous coordinates). Given the 3D point $\mathbf{x} = \begin{bmatrix} x & y & z & w \end{bmatrix}^\top$

$$\begin{aligned} \mathbf{E}\mathbf{x} &= \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{x} \\ &= \begin{bmatrix} \mathbf{R}\begin{bmatrix} x & y & z \end{bmatrix}^\top + w\mathbf{t} \\ w \end{bmatrix}. \end{aligned}$$

Every Lie group has an associated *Lie algebra*, which is the *tangent space* around the identity element of the group. In this way, Lie algebra is a vector space generated by differentiating the group of transformations along the chosen directions in the space, at the identity transformation. In other words, the Lie algebra of the **SE**(3) group, noted as $\mathfrak{se}(3)$, is the set of 4×4 matrices corresponding to differential translations and rotations. The tangent space is a vector space with the same dimension as the number of degrees of freedom of the group of transformations. In the case of **SE**(3), the number of degrees of freedom is six (three for rotation and three for translation). The basis elements of the tangent space are called *generators*. For instance, the generators of $\mathfrak{se}(3)$ are

$$\mathbf{G}_1 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \mathbf{G}_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \mathbf{G}_3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$
$$\mathbf{G}_4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \mathbf{G}_5 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \mathbf{G}_6 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

An element of $\mathfrak{se}(3)$ can be expressed as a linear combination of the generators $\mathbf{G}_1, \ldots, \mathbf{G}_6$, so

$$u_x\mathbf{G}_1 + u_y\mathbf{G}_2 + u_z\mathbf{G}_3 + \omega_x\mathbf{G}_4 + \omega_y\mathbf{G}_5 + \omega_z\mathbf{G}_6 \in \mathfrak{se}(3).$$

Observe that $\mathbf{u} = \begin{bmatrix} u_x & u_y & u_z \end{bmatrix}$ is the linear velocity of a point on the screw axis and $\boldsymbol{\omega} = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}$ is the angular velocity.

To relate elements in the tangent space with the associated elements in the underlying group, the *Exponential map* for every group is defined. Let us consider the Taylor expansion of the exponential function e^x,

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}.$$

It is possible to generalize this exponential function to square matrix arguments by defining the function $\exp : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ as $\exp(\mathbf{X}) = e^{\mathbf{X}}$. Then the Taylor expansion will be

$$\exp(\mathbf{X}) = \sum_{n=0}^{\infty} \frac{\mathbf{X}^n}{n!},$$

with $\mathbf{X}^0 = \mathbf{I}$. This function shares properties with the exponential function e^x. In particular, later we will use the property

$$\frac{\partial}{\partial t}\exp(t\mathbf{X}) = \mathbf{X}\exp(t\mathbf{X}) = \exp(t\mathbf{X})\mathbf{X}. \tag{3.17}$$

For the **SE**(3) group, the *exponential map* converts elements of the tangent space $\mathfrak{se}(3)$ to 3D transformations in the group **SE**(3). This is, given $\mathbf{\Phi} \in \mathfrak{se}(3)$, the exponential map is then defined as

$$\exp \mathbf{\Phi} = \begin{bmatrix} \mathbf{R} & \mathbf{V}\mathbf{u} \\ 0 & 1 \end{bmatrix},$$

where

$$\mathbf{R} = \mathbf{I} + a\boldsymbol{\omega}_\times + b\boldsymbol{\omega}_\times^2, \quad \mathbf{V} = \mathbf{I} + b\boldsymbol{\omega}_\times + c\boldsymbol{\omega}_\times^2,$$

and

$$a = \frac{\sin\theta}{\theta}, \quad b = \frac{1-\cos\theta}{\theta^2}, \quad c = \frac{1-a}{\theta^2},$$

with $\theta = \sqrt{\boldsymbol{\omega}^\top\boldsymbol{\omega}}$, and $\boldsymbol{\omega}_\times$ is the *skew-symmetric matrix* of $\boldsymbol{\omega}$, given by

$$\boldsymbol{\omega}_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}.$$

For implementation purposes, the Taylor expansion of a, b and c should be used when θ^2 is small.

3.6.1 Lie Groups on optimization methods

For the present work, it is important the use of Lie Groups in optimization methods to compute several camera poses capturing a scene. The reason for this comes from the need of computing partial derivatives of camera poses during the optimization. To clarify this, imagine that we want to compute the derivative of a rotation matrix $\mathbf{R}$ which belongs to **SO**(3) group. If we compute straight $\frac{\partial \mathbf{R}}{\partial \mathbf{R}_{(i,j)}}$, we will obtain a matrix which does not belong to **SO**(3) because an infinitesimal change of a single element of an orthogonal matrix will return a non-orthogonal matrix, that will be outside of the **SO**(3) group. Given that the **SO**(3) group has elements of three degree of freedom, there are three Cartesian directions in which it is possible to modify $\mathbf{R}$. These Cartesian directions are the basis vectors of the tangent space of $\mathbf{R}$.

The tangent space at $\mathbf{E}$ is spanned by smooth path of the form

$$\mathbf{E}_k(t) = \exp(\widehat{t\mathbf{e}_k})\mathbf{E},$$

where $\mathbf{e}_k \in \mathbb{R}^m$ is the Cartesian unit vector. We use the equation (3.17) obtaining

$$\left.\frac{\partial \exp(\boldsymbol{\epsilon})}{\partial \epsilon_k}\right|_{\boldsymbol{\epsilon}=\mathbf{0}} = \left.\frac{\partial \exp(\widehat{t\mathbf{e}_k})}{\partial t}\right|_{t=0} = \mathbf{G}_k.$$

The derivative of $\mathbf{E}$ then is computed as

$$\left.\frac{\partial \mathbf{E}_k(t)}{\partial t}\right|_{t=0} = \left.\frac{\partial \exp(\widehat{\boldsymbol{\epsilon}})\mathbf{E}}{\partial \epsilon_k}\right|_{\boldsymbol{\epsilon}=\mathbf{0}} = \mathbf{G}_k\mathbf{E}.$$

Finally, the iterative update rule presented in equation (3.12) must be replaced by

$$\mathbf{E}_{k+1} = \exp(\hat{\boldsymbol{\delta}})\mathbf{E}_k.$$

Capítulo 3

Preliminares (resumen)

En este capítulo se describen los conceptos básicos utilizados por el método de Visual SLAM desarrollado. Inicialmente, se analiza el modelo de cámara *pinhole* y se describe el proceso de desdistorción de una cámara. También, se comenta la limitación que presenta trabajar con una única vista, es decir, la imposibilidad de realizar una reconstrucción tridimensional de la escena observada.

Asimismo, se comentan los extractores de características visuales (*features*) de las imágenes más relevantes de la literatura. La extracción de características es una forma de representar de manera concisa la información presente en una imagen mediante la detección de puntos de alto gradiente (*keypoints*), y la generación de una firma única llamada descriptor (*descriptor*) para cada uno de ellos.

Por otro lado, se comenta el modelo de *Bag of Words* que permite representar a una imagen con un conjunto de palabras visuales. Las palabras visuales son computadas a partir de las características extraídas en la imagen. Esta forma de representar una imagen por un conjunto de palabras visuales permite realizar una eficiente comparación en términos de apariencia entre dos imágenes.

Se describe la geometría epipolar (*epipolar geometry*) que describe las relaciones existentes entre dos cámaras que observan una misma escena. Se presenta el concepto de rectificación de una cámara estéreo para facilitar la búsqueda de correspondencias entre un par de imágenes. Asimismo, se realiza la derivación matemática que permite obtener la triangulación de un punto 3D dada la correspondencia (*matching*) entre un punto de la imagen izquierda y un punto de la imagen derecha.

Se presentan los distintos enfoques que permiten representar la geometría entre múltiples vistas. Entre ellos, la geometría de Tensores (*Tensors*) que captura las relación geométrica de tres (*Trifocal Tensor*) y cuatro vistas *(Quadrifocal Tensor)*. El método de optimización *Bundle Adjustment* que permite estimar de manera simultánea las poses de múltiples cámaras y las posiciones de los puntos tridimensionales correspondientes a la escena observada, mediante el uso de las correspondencias entre los puntos del mapa reconstruido y las observaciones de los mismos en las imágenes. Se muestra además cómo el problema de Bundle Adjustment puede ser representado como un grafo donde los nodos son las cámaras y los puntos de la escena reconstruida; y las aristas son las correspondencias entre ambos tipos de nodos. También, se presenta cómo se puede realizar la marginalización de los nodos referentes a los puntos para la obtención de un grafo de poses (*pose-graph*). El grafo de poses resulta de gran interés en los problemas donde sea desea realizar una optimización de la trayectoria efectuada por un robot, y el mapa reconstruido es demasiado grande.

Por otra parte se comentan los métodos de Perspectiva por n Puntos (*Perspective-n-Point*) que permiten estimar la pose de una cámara dado un conjunto de puntos 3D y sus correspondientes puntos 2D en la imagen. Dicho método es ideal cuando no se cuenta con una pose inicial de la cámara suficientemente cercana a la solución óptima, haciendo imposible el uso de un método de optimización iterativo.

Por último, se derivan los algoritmos iterativos como el Descenso por el Gradiente, Método de Newton, Gauss-Newton y Levenberg-Marquardt utilizados en los métodos de optimización. También, de manera de lograr que los métodos de optimización sean robustos a mediciones espúreas (*outliers*), se propone la aplicación de *M-Estimators* en la función del coste a minimizar. Finalmente, se presentan los Grupos de Lie y el Álgebra de Lie que permiten trabajar de manera analítica con las transformaciones tridimensionales.

Chapter 4

S-PTAM

In this chapter we explain in detail the feature-based SLAM method proposed in this Thesis. The method is called S-PTAM from Stereo Parallel Tracking and Mapping, and uses a stereo camera as the main sensor. S-PTAM constructs and maintains a sparse map representation of the environment to perform an accurate camera localization. S-PTAM follows the Parallel Tracking and Mapping (PTAM) approach that separates the tracking task from the map maintenance task in two separate threads, taking advantage of multi-core CPU architectures. On the other hand, a Loop Closing module runs on a third thread to provide a consistent global localization. The proposed system is capable on working with long trajectories in real-time.

4.1 Overview

Figure 4.1 shows a scheme of the main components and the computation flow of S-PTAM. The system starts with the camera at the world reference frame and an empty map, which is immediately initialized by triangulating the features matched in the first stereo pair. The map initialization is not depicted in the figure for easy understanding.

From then on, the tracking thread estimates the current pose for each new incoming stereo frame by minimizing the re-projection error between the projected map points and their corresponding matches in the image. Once the current camera pose is estimated, the associated stereo frame could be selected to be added to the map (in such case is called *keyframe*). We will call *map point* the structure constituted by the 3D coordinates of a point plus its corresponding feature descriptor. If a keyframe is added to the map, then the unmatched features are triangulated from the stereo pair, and the pose and the new map points are added to the point-pose graph maintained by S-PTAM. Once the keyframe is processed by the tracking, the mapping thread actively searches for point correspondences between keyframes in order to strengthen the constraints of the point-pose graph. Immediately after, the mapping thread performs a Local Bundle Adjustment (LBA) over the point-pose subgraph defined by the last added keyframes. After the LBA, those measurements that were tagged as outliers by the robust cost function during the optimization, are removed. Finally, the map points that are not associated to any measurement are removed from the map.

To be able to correct the accumulated drift in long trajectories, S-PTAM runs a loop closure detection in a third thread. This thread searches for loop closure candidates using the visual appearance of salient points extracted on images. A potential loop candidate is validated if a high quality relative transformation between both closing keyframes is found. Once the loop is validated, a pose-graph is generated from the point-pose graph by marginalizing out map point nodes. Then, the relative transformation is added to the pose-graph as a fixed constraint. Finally, the loop correction is carried out in two stages. Firstly, an initial correction is computed by distributing the accumulated error over all the keyframes that form the loop. Secondly, an iterative pose-graph optimization is performed taking the initial loop correction as seed.

The following sections provide a more detailed explanation of each part of the method.

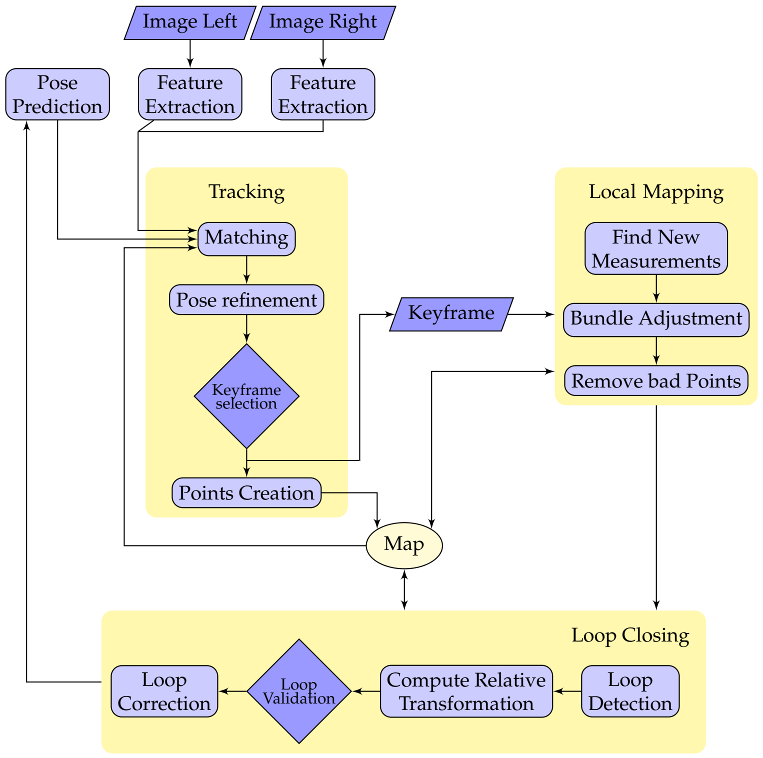


Figure 4.1: Main steps performed by the tracking, local mapping and loop closing threads. The map initialization is not depicted in the figure to easy understanding.

4.2 Feature Extraction

S-PTAM is a system that relies on local image features (the concept of local image features is presented in section 3.2.1). For each frame, the local image features corresponding to already mapped 3D points are used to estimate the camera pose. The position of the features that have not been already mapped to 3D points can be used to triangulate the 3D coordinates of new map points when the tracked frame is selected as keyframe. Map points must be tracked from a sequence of different camera poses (viewpoints) during robot navigation. This implies that feature descriptors must be invariant to viewpoint changes and this is a critical characteristic of the selected features.

The computational cost required by the keypoint detection and the descriptor extraction algorithms is also an important factor to achieve real-time performance. A good spatial distribution of the detected features on the image is essential to perform a good tracking of the camera and avoid ill-conditioned pose estimations. Moreover, the reader must notice that there is a trade off between the accuracy that can be obtained and the time performance of the system. The creation of too many or too few map points can affect notoriously the performance of the system. For instance, if the number of triangulated points in one keyframe is too low, it is possible that there will be not enough map points to be tracked by the next incoming frame and consequently the camera localization could be lost. In contrast, if the number of features that are triangulated is too high, the map will grow quickly and thus, the performance of the whole system will be compromised.

For the S-PTAM system implementation, the Shi–Tomasi detector [10] was selected to extract image keypoints, and the BRISK [15] descriptor was selected to obtain the keypoints' signature. The choice of this combination of detector and descriptor algorithms is based on a exhaustive assessment of existing methods which is presented in section 5.3.

4.3 Map Structure

The map constructed by S-PTAM is a non-directed bipartite point-pose graph $M = (V, L)$ where the nodes V are the map points P and the keyframes K stored along the camera trajectory; the edges L are the measurements (constraints) between them. This means that there are neither point-point constraints nor keyframe-keyframe constraints in the reconstructed map. An example of a sample graph can be seen in Figure 3.18. From now on, for ease of reading, we will refer to frames and keyframes using their corresponding transformation matrices $\mathbf{E}$ and $\mathbf{K}$, respectively. Also, we will omit the coordinate systems involved in the transformation matrices, for example, the $\mathbf{K}^{c_j w}$ could become $\mathbf{K}_j$.

One important feature of the map is that it allows us to efficiently obtain the set of keyframes that have at least one map point in common (*covisibility* keyframes). In S-PTAM, covisibility allows us to determine the local area of the map around the current camera pose, i.e., the set of map points that will be potentially found on the current frame. Formally, we note O_j, with $O_j \subseteq P$, the set of map points that are observed by the frame $\mathbf{E}_j$. Then in order to calculate the local map for the present frame $\mathbf{E}_i$ we do the following: we first determine the map points observed by the previous frame $\mathbf{E}_{i-1}$, this is O_{i-1}. Then, we get the set of keyframes that observe at least one map point of the set O_{i-1}, this is

$$CK_{i-1} = \{\mathbf{K}_j \in K : \exists p \in O_{i-1} \textit{ such that the edge } (p, \mathbf{K}_j) \in L\}.$$

Finally, we define the local map of the frame $\mathbf{E}_i$ as

$$local\_map_i = \{p \in P : \exists \mathbf{K}_j \in CK_{i-1} \textit{ such that the edge } (p, \mathbf{K}_j) \in L\}.$$

Figure 4.2 shows the local map defined to track the current frame $\mathbf{E}_i$. The red points are the map points observed by the frame $\mathbf{E}_{i-1}$.

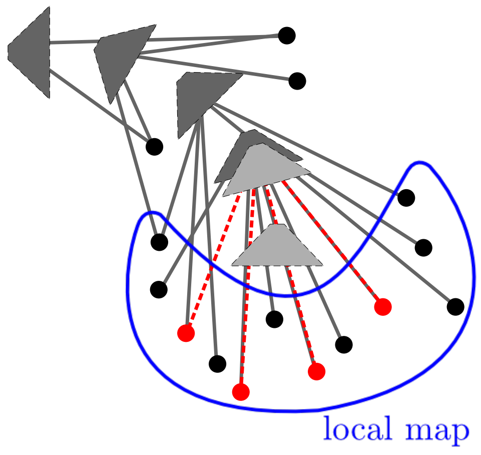


Figure 4.2: Local map corresponding to frame $\mathbf{E}_i$ in the context of a camera moving on a clockwise rotating trajectory. The dots represent map points. The red dots represent map points O_{i-1} observed by the frame $\mathbf{E}_{i-1}$. The dark gray trapezoids represent keyframes. The light gray trapezoids represent the frames $\mathbf{E}_{i-1}$ and $\mathbf{E}_i$. The two most recent keyframes are the ones that observe map points within the set O_{i-1}, therefore they constitute the set CK_{i-1}. Then, the local map of frame $\mathbf{E}_i$ is made up by the map points observed by keyframes of CK_{i-1}, surrounded by the blue curve.

4.4 Map initialization

As any feature-based SLAM method, it is important to create and maintain an accurate and populated map without outliers. For this reason, some restrictions about how and which points should be created during map initialization are performed in S-PTAM.

When the method starts, the initial camera pose is taken as the canonical pose, and therefore located on the origin of the world coordinate. An initial map of the scene viewed by the stereo camera is reconstructed from the first two incoming images. The map is initialized from the triangulation of the stereo correspondences found in the first stereo pair. As the stereo camera images are undistorted and rectified, the epipolar lines follow the image rows, then the search for correspondences between both images is restricted to one dimension, speeding up and robustifying the stereo matching process (section 3.3.2). To validate good matches, a minimum similarity threshold is required. Moreover, if more than one possible match is found, we keep the closest point (in terms of similarity) only if its distance to the second match is bigger than a given threshold.

Once stereo matching correspondences are found, the stereo triangulation of the matches is performed. During the triangulation, to avoid the reconstruction of spurious 3D points (caused by wrong matches), the points are filtered, checking if they fall inside of both camera fields of view (*frustum culling*). Figure 4.3 shows the stereo matching of S-PTAM, and the initial map reconstructed. Then, the map is considered to be valid, if the number of triangulated points is higher than a given threshold (10 points). In the cases in which the number of triangulated points is bellow this threshold, the map is discarded and the next incoming images are used for map initialization. The requirement of having a map with a minimum number of points is crucial for the posterior localization of the camera.

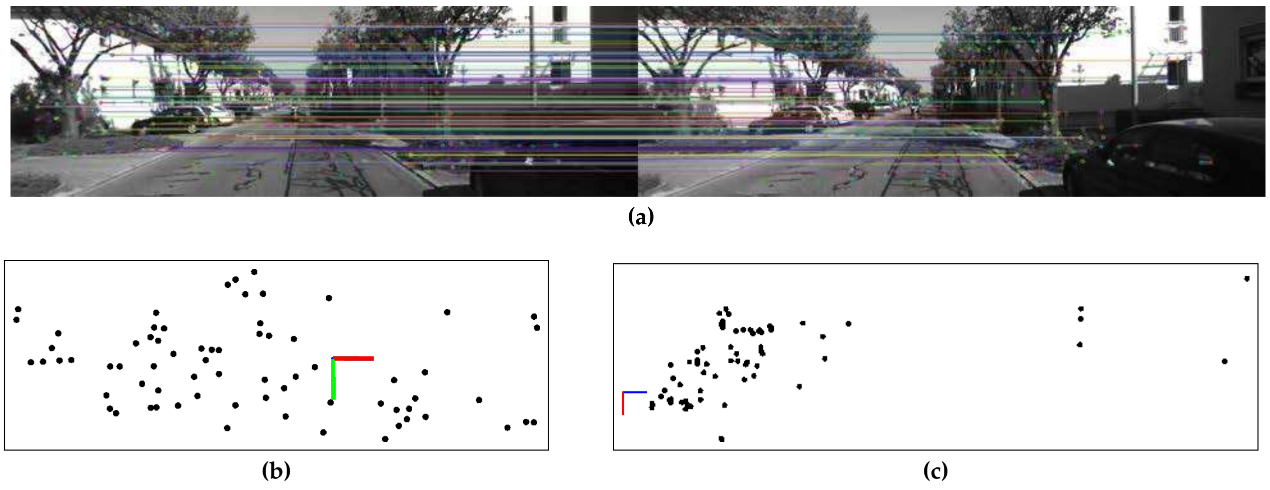


Figure 4.3: S-PTAM initialization. **(a)** Stereo matches between the left image and right image from the first stereo pair. **(b)** Initial map (point cloud) viewed from behind the stereo camera pair. The coordinate system presented in the figure corresponds to the left camera coordinate system of the stereo pair. The x-axis is drawn in red pointing to the right, the y-axis is drawn in green pointing downwards. The z-axis is not drawn because is perpendicular to the image. **(c)** Initial map viewed from the top. The coordinate system presented in the figure corresponds to the left camera coordinate system. The x-axis is drawn in red pointing to downwards, the z-axis is drawn in blue pointing to the right. The y-axis is not drawn because is perpendicular to the image.

It is important to mention that when a point is triangulated, the associated feature descriptor is stored with it in the map point structure (in practice we store the left descriptor). The descriptor stored in the map point allows us to track this point in future frames.

4.5 Tracking

The tracking phase is in charge of performing the localization of the camera pose, and can be decomposed into four sequential steps.

4.5.1 Pose prediction

A reasonable prediction of the current camera pose is desired in order to actively search the correspondences between the map and the extracted features. In our case, dead-reckoning based on wheel odometry is used, since it is available in most ground based robotic vehicles. If external odometry were not available, a *decaying velocity model* can be used instead. The *decaying velocity model* is described bellow.

To predict the camera pose for the frame i, $\mathbf{E}^{\mathrm{c}_i\mathrm{w}}$, we make use of the previously camera poses $\mathbf{E}^{\mathrm{c}_{i-1}\mathrm{w}}$ and $\mathbf{E}^{\mathrm{c}_{i-2}\mathrm{w}}$. In particular, we will work with the position 3-components position vector $\mathbf{p}$ and the quaternion orientation $\mathbf{q}$ representation for each pose. Then, the prediction is performed by

$$\hat{\mathbf{p}}_i \leftarrow \mathbf{p}_{i-1} + \mathbf{v}_{i-1},$$

$$\hat{\mathbf{q}}_i \leftarrow \mathbf{q}_{i-1}\boldsymbol{\omega}_{i-1}.$$

Observe that $\mathbf{v}$ is the displacement and $\mathbf{q}$ is the change in orientation between two consecutive frames, respectively. The predicted pose in matrix representation can be derived straightforwardly. Once the tracking is performed, the velocities are updated and smoothed in order to support acceleration changes:

$$\mathbf{v}_i \leftarrow ((\mathbf{p}_i - \mathbf{p}_{i-1}) + \mathbf{v}_{i-1})\, 0.5,$$

$$\boldsymbol{\omega}_i \leftarrow \operatorname{slerp}\left(\mathbf{q}_i\, \mathbf{q}_{i-1}^{-1}, \boldsymbol{\omega}_{i-1}, 0.5\right),$$

where $\operatorname{slerp}(.)$ is the spherical linear interpolation, and 0.5 is the smoothness parameter.

4.5.2 Matching

The matching step consists in finding correspondences between the current map and the image points extracted in the current frame. This 3D-2D correspondences are constraints that will be used to estimate the current camera pose. We project each map point inside the viewing frustum of the predicted stereo pose on the left and right images, and search for the match in a neighborhood of the point. The match is established by comparing the stored descriptor in the projected map point with the descriptors of the features that fall in the neighborhood. This comparison of descriptors follows the same validation of the stereo matching procedure used for the creation of the map points.

To speed up the matching between the map and the features extracted, each image is partitioned as a grid with a given cell size. For each map point, the search neighborhood is composed by the cells around the projection of that map point within a given radius. Figure 4.4 shows how the search neighborhood for a given projection is determined with a search radius of unitary magnitude.

Figure 4.5 shows the tracking of a stereo frame by S-PTAM in a real sequence. The result of the matching step is a set of 3D–2D correspondences that will be used to refine the current camera pose. The 2D image features are called *measurements*, given that they are the information measured by the camera. On the other hand, as we are working with a stereo camera, we will have 3D–2D monocular correspondences for both

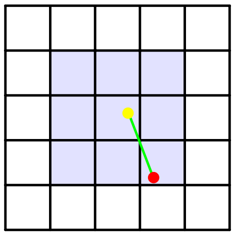


Figure 4.4: Grid neighborhood matching. The search neighborhood is indicated in light blue. The projection of the map point (yellow dot) matches with a feature (red dot) in the search neighborhood of radius one -centered at the map point projection cell- indicated in light blue.

left and right cameras. In particular, there are correspondences from both cameras that share the same 3D point. The pairs of 3D-2D monocular correspondences that share the same 3D point are what we will call *stereo measurement*s. Formally, given two monocular measurements of a same 3D point $\mathbf{z}^{\mathrm{l}} = \begin{bmatrix} u_l & v_l \end{bmatrix}^\top$ and $\mathbf{z}^{\mathrm{r}} = \begin{bmatrix} u_r & v_r \end{bmatrix}^\top$, the *stereo measurement* is defined as

$$\mathbf{z}^{\mathrm{s}} = \begin{bmatrix} u_l \\ v_l \\ u_r \end{bmatrix}.$$

Observe that as the stereo camera is undistorted and rectified, so $v_l = v_r$, and then no information is missed with this stereo representation.

4.5.3 Pose refinement

Once 3D–2D correspondences are obtained, the pose refinement step is carried out to improve the predicted the camera pose. This refinement step can be seen as a particular instance of the *Bundle Adjustment* problem presented in section 3.4.2, considering that only one camera should be adjusted and the map is kept fixed. Furthermore, given that the camera was calibrated previously, and therefore its intrinsic parameters are known, only extrinsic parameters (camera pose) should be estimated. The reprojection error function presented in 3.9 must be instantiated to take into account both monocular left and monocular right measurements, and stereo measurements. The cost function that should be minimized now has the form

$$f(\mathbf{E}^{\mathrm{cw}}) = \|u(\mathbf{E}^{\mathrm{cw}})\|_\rho$$

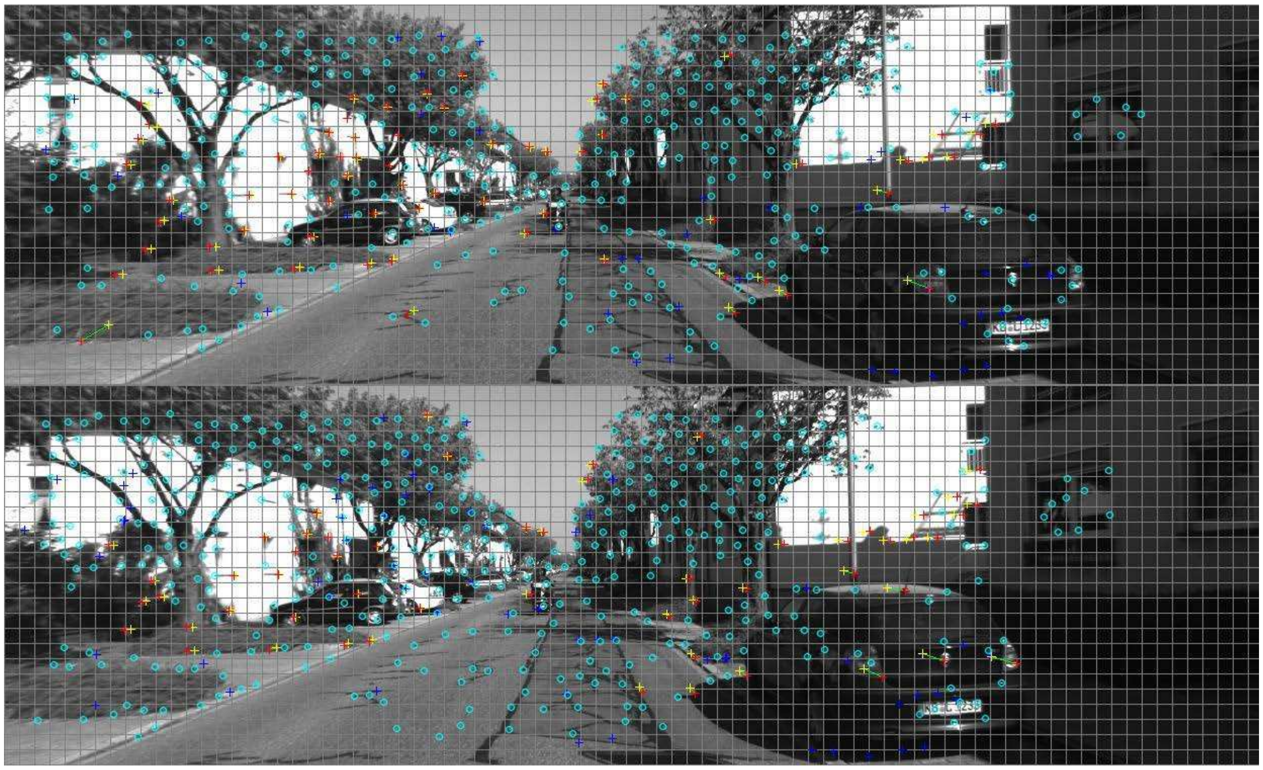
where $u(\mathbf{E}^{\mathrm{cw}}) = \begin{bmatrix} u_1(\mathbf{E}^{\mathrm{cw}}), & u_2(\mathbf{E}^{\mathrm{cw}}), & \ldots, & u_m(\mathbf{E}^{\mathrm{cw}}) \end{bmatrix}^\top$ with m the total number of points measure by the stereo pair, the $u_i(\mathbf{E}^{\mathrm{cw}})$ are given by

$$u_i(\mathbf{E}^{\mathrm{cw}}) = \begin{cases} \mathbf{z}_i^{\mathrm{l}} - \hat{\mathbf{z}}^{\mathrm{m}}(\mathbf{E}^{\mathrm{cw}}\dot{\mathbf{x}}_i^{\mathrm{w}}) & \text{for measurments made by the left camera only} \\ \mathbf{z}_i^{\mathrm{r}} - \hat{\mathbf{z}}^{\mathrm{m}}(\mathbf{E}^{\mathrm{rc}}\mathbf{E}^{\mathrm{cw}}\dot{\mathbf{x}}_i^{\mathrm{w}}) & \text{for measurments made by the right camera only,} \\ \mathbf{z}_i^{\mathrm{s}} - \hat{\mathbf{z}}^{\mathrm{s}}(\mathbf{E}^{\mathrm{cw}}\dot{\mathbf{x}}_i^{\mathrm{w}}) & \text{for measurments made by both cameras} \end{cases} \tag{4.1}$$

and $\|.\|_\rho$ is the Huber norm introduced in section (3.5.4). In equation (4.1), $\hat{\mathbf{z}}^{\mathrm{m}}$ and $\hat{\mathbf{z}}^{\mathrm{s}}$ are the monocular and the stereo projection models, respectively

$$\hat{\mathbf{z}}^{\mathrm{m}}(\mathbf{x}) = f\,\mathrm{proj}(\mathbf{x}) + \mathbf{c},$$

$$\hat{\mathbf{z}}^{\mathrm{s}}(\mathbf{x}) = \begin{bmatrix} f\,\mathrm{proj}(\mathbf{x}) + \mathbf{c} \\ f\frac{x-b}{z} + c_u \end{bmatrix}.$$


(a)

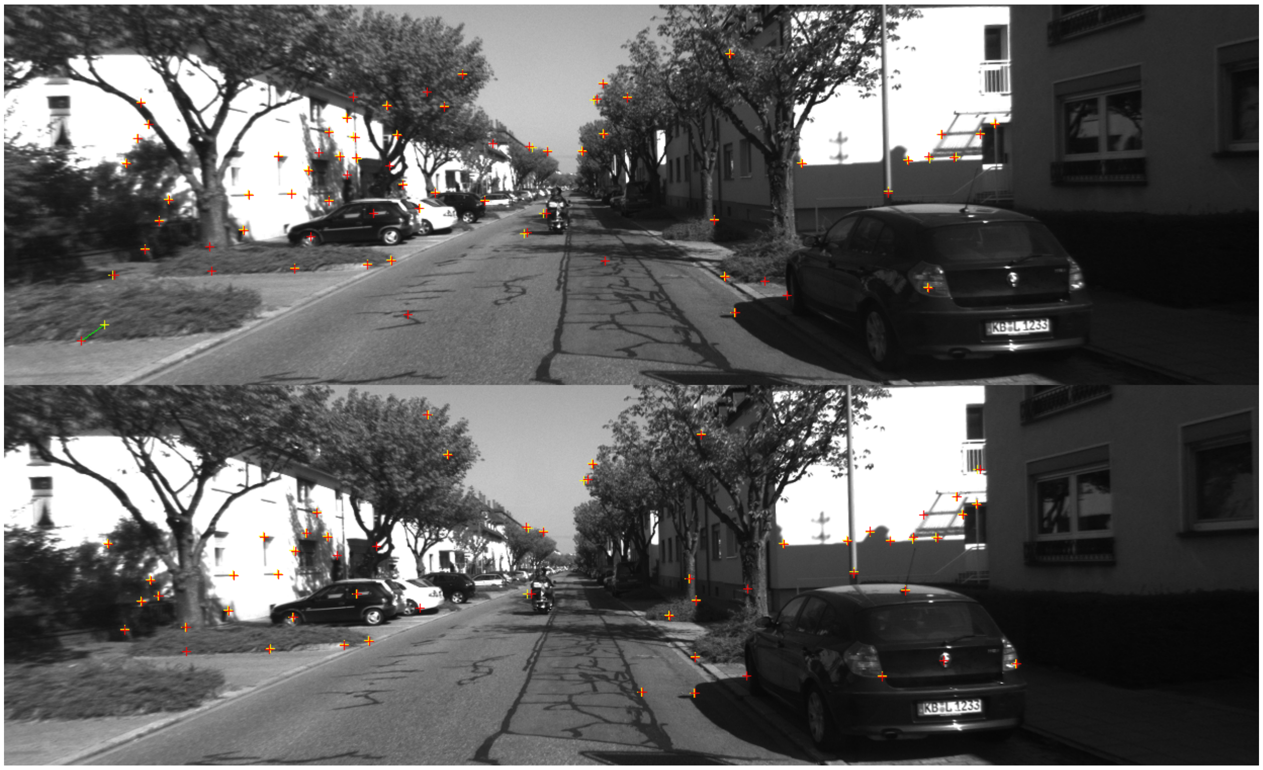

(b)

Figure 4.5: S-PTAM during the tracking of a stereo frame. **(a)** Matching between map point projections and features extracted from the images. Yellow marks are projections whose features (red marks) could be established. Cyan dots make reference to non matched features. Blue marks correspond to non matched projections. Green segments indicate the reprojection errors. **(b)** Matches after the camera pose refinement is carried out.

One additional consideration must be contemplated for this minimization. Instead of estimating directly the camera pose parameters on each iteration, the incremental motion is computed as it is pointed out in section 3.6.1. This is, the incremental update of the minimization method used is

$$\mathbf{E}^{\mathrm{cw}}_{k+1} = \exp(\hat{\boldsymbol{\delta}})\mathbf{E}^{\mathrm{cw}}_{k},$$

where $\hat{\boldsymbol{\delta}} = (t_x, t_y, t_z, \theta_{roll}, \theta_{pitch}, \theta_{yaw})$ is the motion vector which reduces the reprojection error on each iteration, and $\mathbf{E}^{\mathrm{cw}}_{k}$ is the estimated pose that resulted from the previous iteration. That is, for each iteration, the increment is computed and the camera pose is updated. In appendix A the Jacobians involved in the refinement step are detailed.

4.5.4 Keyframe selection and map point creation

Once the current pose is refined, the associated frame is selected to be a *keyframe* according to a selection heuristic. In the literature there exist several heuristics to determine if a frame must be a keyframe [17, 74, 75, 76, 52]. Some of the most used heuristics select the current frame to be a keyframe in terms of: the euclidean distance to the nearest keyframe in the map, the parallax achieved between the current frame and the nearest keyframe, the percentage of features shared with the nearest keyframe, the area coverage ratio on the image by the projected scene, or the information gain with its insertion, among others.

The use of keyframes has several reasons: to expand the map when new areas are visited, to be used during map refinement in the mapping phase and to limit the portion of the map to be projected in the image plane during the matching step. In S-PTAM the keyframes are selected in such a way that they are not too redundant, and at the same time the map points shared with other keyframes are enough to keep a precise localization. In this way, S-PTAM selects a frame as keyframe, if the number of tracked points is less than 90% of the points tracked by the last keyframe. If the current frame is selected as a new keyframe, the features that were not used during the tracking process are triangulated to create new map points. Other Visual SLAM systems (like PTAM [17] or ORB-SLAM2 [52]) create new map points once the keyframe is processed by the mapping thread. This may be not immediate, depending on the level of congestion of the mapping thread, potentially causing a tracking failure. In contrast, S-PTAM immediately creates and incorporates the new map points into the map after the tracking step to avoid the loss of potential map matches on the upcoming frames. Finally, the keyframe is queued into the map refinement thread, to be processed as soon as possible.

4.6 Local Mapping

The Local Mapping module is in charge of maintaining and refining the map of the local area around the robot.

4.6.1 Keyframes queue

When a keyframe is "sent" by the tracking module to the local mapping module, the keyframe is added to a queue. In this way, the tracker can create keyframes without having to wait for the mapping thread to be available to process the keyframe. Once the local mapping thread is free, it takes the keyframes stored in the queue to refine the local map. In order not to let non-processed keyframes to be accumulated in the queue, the local mapping thread can unqueue a fixed number of keyframes (e.g. up to 5 keyframes) at each unqueuing operation. This number of keyframes depends on the processing power available and on the frequency with which keyframes are created. It is important to clarify that it is not always possible to unqueue all the keyframes from the queue because if the queue is too large the refinement operations to be performed would be too computationally costly.

4.6.2 Procedure for finding new measurements

Once a keyframe (or a set of keyframes) is unqueued, the local mapping thread tries to add new constraints to the point-pose graph. To this end, the new triangulated map points are projected on and matched with the keyframes that are covisible with the keyframe that triangulated them. In this way, a more connected graph is obtained, and a better optimization can be performed. Figure 4.6 exemplifies the process of finding new measurements.

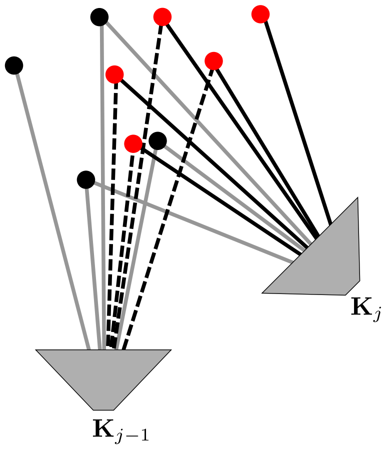


Figure 4.6: Procedure for finding new measurements. The new map points (red dots) triangulated by the $\mathbf{K}_j$ are projected on and matched with the covisible keyframe $\mathbf{K}_{j-1}$. Dotted lines represent the new measurements.

4.6.3 Map refinement (Local Bundle Adjustment)

This section details the local mapping algorithm that uses the stereo and the monocular measurements to refine a local area of the point-pose graph. This local area is composed by the keyframes extracted from the queue and their covisible keyframes; and also by the map points observed by them. The problem of jointly refining the camera poses of the keyframes and the 3D positition of the map points that belong to the local area is called the *Bundle Adjustment* problem, which was introduced in section 3.4.2. This is different from the minimization carried out in the Pose Refinement step in the tracking phase, where only one camera, the current one, is adjusted.

In this way, the BA of the local mapping phase can be formulated as follows: given an initial set of keyframe poses $\{\mathbf{K}^{\mathrm{c}_1\mathrm{w}}, \dots, \mathbf{K}^{\mathrm{c}_N\mathrm{w}}\}$, an initial set of 3D points $\mathbf{x}^{\mathrm{w}} = \{\mathbf{x}_1^{\mathrm{w}}, \dots, \mathbf{x}_M^{\mathrm{w}}\}$ and the measurement $\mathbf{z}_{ij}$ of the $\mathbf{x}_i^{\mathrm{w}}$ 3D point in the $\mathbf{K}^{\mathrm{c}_j\mathrm{w}}$ keyframe, the simultaneous estimation of the multiple cameras and the point cloud is achieved rewriting the reprojection error presented in equation (3.9) as

$$u_i(\mathbf{K}^{\mathrm{c}_j\mathrm{w}}, \mathbf{x}_i^{\mathrm{w}}) = \begin{cases} \mathbf{z}_{ij}^{\mathrm{l}} - \hat{\mathbf{z}}^{\mathrm{m}}(\mathbf{K}^{\mathrm{c}_j\mathrm{w}}\dot{\mathbf{x}}_i^{\mathrm{w}}) & \text{for measurments made by the left camera only} \\ \mathbf{z}_{ij}^{\mathrm{r}} - \hat{\mathbf{z}}^{\mathrm{m}}(\mathbf{E}^{\mathrm{rc}}\mathbf{K}^{\mathrm{c}_j\mathrm{w}}\dot{\mathbf{x}}_i^{\mathrm{w}}) & \text{for measurments made by the right camera only} \\ \mathbf{z}_{ij}^{\mathrm{s}} - \hat{\mathbf{z}}^{\mathrm{s}}(\mathbf{K}^{\mathrm{c}_j\mathrm{w}}\dot{\mathbf{x}}_i^{\mathrm{w}}) & \text{for measurments made by both cameras} \end{cases} . \tag{4.2}$$

Notice that here, in contrast to the reprojection error formulation in equation (4.1), equation 4.2 is the reprojection error where the dependence on the 3D points is included in the minimization. Appendix A details the Jacobians involved in this BA.

As we have done before, the Huber norm is used in the cost function to mitigate the effect of outliers during the minimization. Each time a measurement is tagged as outlier by the robust cost function it is removed from the pose-point graph. Then, if as a result of this process, a map point losses all of its associated measurements, then it is removed from the graph.

4.7 Loop Closing

Handling large environments requires a system capable of recognizing already-visited places and optimizing the map and the trajectory, in order to reduce the accumulated drift and maintain a globally consistent map model. To accomplish this task, the process is divided into three phases: the detection of revisited places, the estimation of their relative transformation and the loop correction.

In the detection phase, the keyframes provided by the local mapping are described by a bag-of-binary-words using a previously trained visual vocabulary. The computed bag-of-binary-words are used to create a keyframe database as proposed in [77]. For each new keyframe, the database is queried to obtain those previously added keyframes that are similar in terms of appearance to the new one.

If a loop closure candidate is found, the relative transformation between the queried keyframe and the loop candidate keyframe is estimated. This transformation will serve to measure the accumulated error and to validate the proposed loop determined by the matched keyframe.

Once a loop has been considered valid, a correction is propagated among all the keyframes of the loop. This propagation provides an initial seed for a later pose-graph optimization, yielding this way a more accurate solution that reduces the accumulated drift error.

4.7.1 Training

To construct the visual vocabulary to be used by the loop closing module during S-PTAM execution, an offline bag-of-words training must take place previously. In order not to bias the visual vocabulary, the training process is performed over several sequences of images coming from different environments (indoors and outdoors) and is done independently of the testing environment.

4.7.2 Image Database

To recognize places previously traversed by the robot, it is important to store the keyframes provided by the local mapping in an efficient structure in order to obtain all of the images that are similar to the query image. The database is formed by the visual vocabulary and the index that relates words with images, and vice versa. The index allows us to search images which contain a given word or, given an image, to retrieve the words it contains. Figure 4.7 shows an example of the database structure.

4.7.3 Loop detection

Loop detection is achieved by making use of the efficient appearance-based method proposed in [53]. Each new keyframe $\mathbf{K}_i$ is described as a bag-of-words vector $\mathbf{v}_i$ and the keyframe database is queried scoring any previously added $\mathbf{v}_j$ that shares words with $\mathbf{v}_i$ following the normalized similarity score

$$\eta\left(\mathbf{v}_i, \mathbf{v}_j\right) = \frac{s\left(\mathbf{v}_i, \mathbf{v}_j\right)}{s\left(\mathbf{v}_i, \mathbf{v}_{i-1}\right)},$$

where $\mathbf{v}_{i-1}$, with $i-1>0$, is the bag of words representation of the previously inserted keyframe, and the similarity score between two bags of words $s(\mathbf{v}_i, \mathbf{v}_j)$ is an $\mathbf{L}_1$-score which lies within the interval $[0,1]$, and is given by

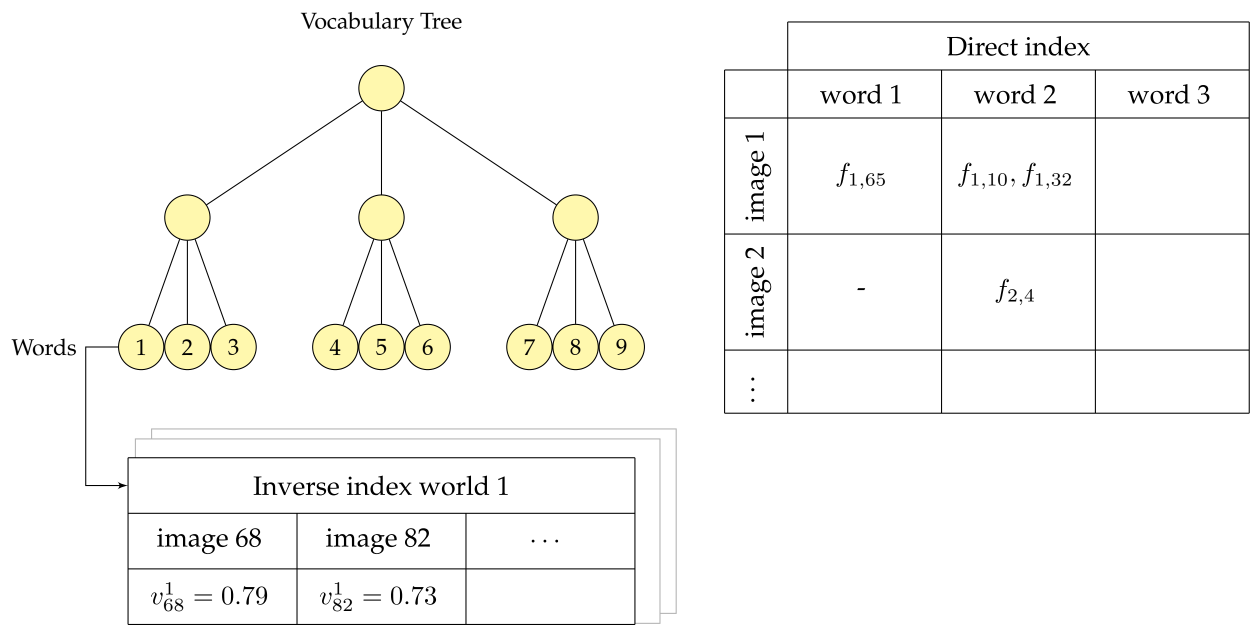


Figure 4.7: Database structure. The visual vocabulary has $L_w = 2$ levels and branching factor of $k_w = 3$, resulting in 9 different words. The inverse index relates each word with all the images, where it appears along with the *tf-idf* associated. The direct index stores the features of the images and their associated nodes at a certain level of the vocabulary. Image adapted from [53].

$$s(\mathbf{v}_i, \mathbf{v}_j) = 1 - \frac{1}{2}\left|\frac{\mathbf{v}_i}{|\mathbf{v}_i|} - \frac{\mathbf{v}_j}{|\mathbf{v}_j|}\right|.$$

In the cases in which the highest normalized similarity score exceeds a predefined threshold, its respective keyframe is considered a match and therefore a potential loop candidate. Observe that from this procedure a list of loop candidates is obtained.

4.7.4 Temporal consistency

To avoid false positive loop detections between subsequent frames, we follow the temporal consistency approach presented in [53]. This is, given the list of loop candidates resulting from a query keyframe $\mathbf{K}_t$, those candidates that are close in time are treated as a unique candidate called *island*. Formally, given a time window T_i that represents a set of timestamps $t_{n_i}, \ldots, t_{m_i}$, the island V_{T_i} is defined as the bag-of-words vectors $v_{t_{n_i}}, \ldots, v_{t_{m_i}}$ that are close in time. In this way, several consecutive loop candidates $< v_t, v_{t_{n_i}} >$ $, \ldots, < v_t, v_{t_{m_i}} >$ are represented solely by one candidate $< v_t, V_{T_i} >$ if the time gaps among the $t_{n_i}, \ldots, t_{m_i}$ are small.

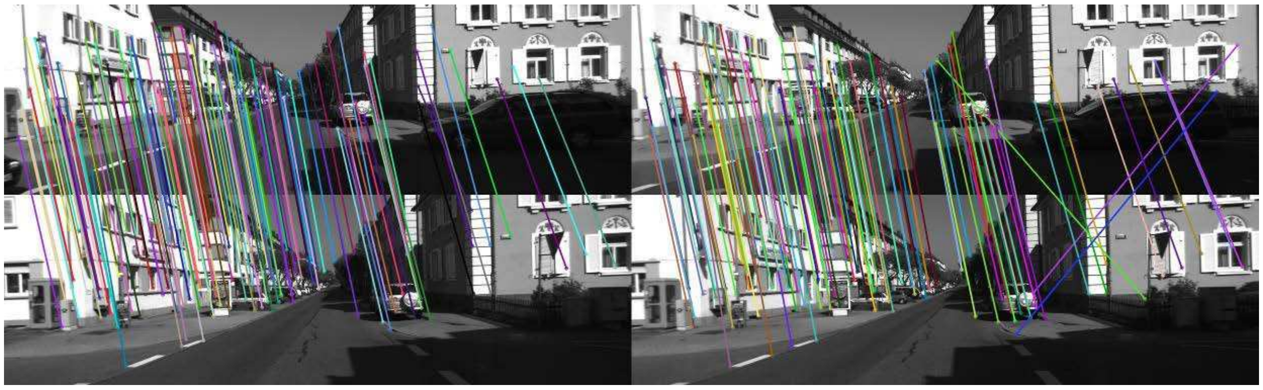
Consequently, the scoring between a bag-of-words vector v_t and an island V_{T_i} is computed as

$$H(v_t, V_{T_i}) = \sum_{j=n_i}^{m_i} \eta(v_t, v_{t_j}).$$

The island with the biggest value H then is selected as the best candidate group, and is notated with $V_{T'}$. Moreover, a temporal condition is applied: $< v_t, V_{T'} >$ must be consistent with k positive previous detections $< v_{t-\Delta t}, V_{T_1} >, \ldots, < v_{t-k\Delta t}, V_{T_k} >$ where T_i and T_{i+1} are close in time. If $V_{T'}$ fulfills the temporal condition, then the pair $< v_t, v_{t'} >$ with $t' \in T'$ and $v_{t'} \in V_{T'}$ will be the bag-of-words vector with highest value η in $V_{T'}$. This pair $< v_t, v_{t'} >$ is considered to be the best and unique candidate to the loop.

4.7.5 Loop transformation estimation

Once the current keyframe $\mathbf{K}_c$ has been successfully matched with a loop candidate keyframe $\mathbf{K}_\ell$, the relative transformation $\mathbf{E}^{c_c c_\ell}$ existing between $\mathbf{K}_c$ and $\mathbf{K}_\ell$ must be computed. This transformation will be used to perform the correction on the detected loop. In order to avoid false positive loops, an initial estimation of the relative transformation, along with a set of inlier associated map points, are computed in a first step. This initial transformation is computed using RANSAC [78] with a P3P (Perspective-3-Point) solver [26] over the 3D-2D correspondences between the 3D points observed by the $\mathbf{K}_c$ and the 2D image features extracted on $\mathbf{K}_\ell$. The aforementioned 3D-2D correspondences for the $\mathbf{K}_\ell$ are obtained by composing the 3D-2D correspondences for $\mathbf{K}_c$ and the 2D-2D correspondences between the keyframes $\mathbf{K}_c$ and $\mathbf{K}_\ell$. Figure 4.8 shows the matching established during the stage of validation of a candidate loop. If the percentage of inliers of the P3P solver exceeds a given threshold, then the detected loop is considered valid and a general PnP (Perspective-n-Point) solver [73] is used over the inlier points to estimate a more accurate relative transformation. The resulting $\mathbf{E}^{c_c c_\ell}$ is finally optimized with a non-linear optimization. Figure 4.9 depicts how the relative transformation $\mathbf{E}^{c_c c_\ell}$ is computed.


(a)

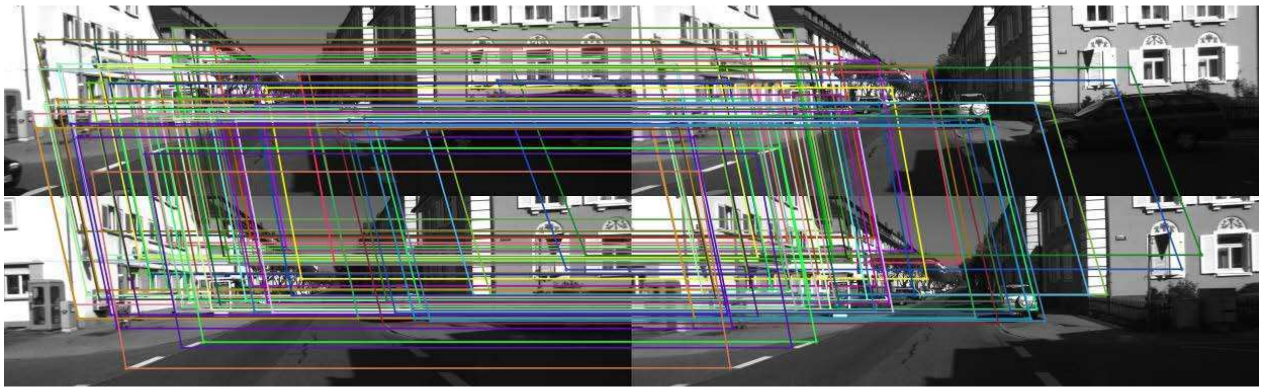

(b)

Figure 4.8: Correspondences established between two stereo keyframes $\mathbf{K}_c$ and $\mathbf{K}_\ell$ during the detection of a loop along the trajectory of the camera. The two upper pictures of the two subfigures correspond to the left and the right images of the keyframe $\mathbf{K}_c$. In an analogous way, the two lower pictures of the two subfigures correspond to the left and the right images of the keyframe $\mathbf{K}_\ell$. **(a)** Independent left and right matches between both pairs of images of a loop candidate. This matching between keyframes associates the 3D points triangulated from the current keyframe $\mathbf{K}_c$ with the 2D image points in the loop keyframe $\mathbf{K}_\ell$. **(b)** Filtering of wrong matches after the application of stereo matching. The horizontal lines depict the matches between the left and right features of the same keyframe. Note how the wrong matches that are easily detectable between right images in the subfigure (a) are discarded.

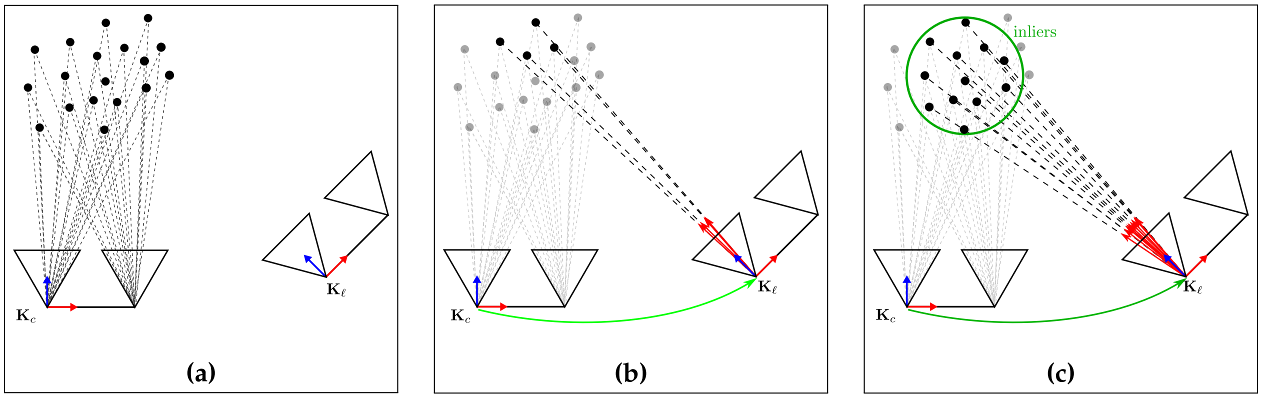


Figure 4.9: **(a)** 3D points observed from the stereo pair corresponding to the current keyframe $\mathbf{K}_c$. **(b)** After the 3D-2D correspondences between the 3D points and the image points in the loop closing keyframe $\mathbf{K}_\ell$ are found, a P3P solver is applied to obtain a first estimation of the relative transformation $\mathbf{E}^{c_c c_\ell}$ (green arrow) between both keyframes. **(c)** Finally, a PnP solver is applied on all of the inliers to refine the relative transformation.

4.7.6 Loop correction

Once a loop is validated the correction of the loop is carried out. The loop correction process estimates first an initial update using the computed relative transformation $\mathbf{E}^{c_c c_\ell}$. This correction is performed by propagating the $\mathbf{E}^{c_c c_\ell}$ transformation through the keyframes between $\mathbf{K}_\ell$ and $\mathbf{K}_c$.

Let $\{\mathbf{K}^{c_\ell w}, \ldots, \mathbf{K}^{c_j w}, \mathbf{K}^{c_{j+1} w}, \ldots, \mathbf{K}^{c_c w}\}$ be the keyframe poses belonging to the detected loop. The propagation proceeds backwards in time from the newer keyframes to the oldest ones, and is defined by:

$$\begin{aligned}\mathbf{K}^{c_c w}_{prop} =& \mathbf{E}^{c_c c_\ell}\mathbf{K}^{c_\ell w}\\ \mathbf{K}^{c_j w}_{prop} =& \text{interpolate}_j\left(\mathbf{K}^{c_j w}, \mathbf{E}^{c_j c_{j+1}}\mathbf{K}^{c_{j+1} w}_{prop}\right)\\ \mathbf{K}^{c_\ell w}_{prop} =& \mathbf{K}^{c_\ell w},\end{aligned}$$

where the low index *prop* refers to the camera poses after the propagation. $\text{interpolate}_j(.)$ performs a pose interpolation between non-propagated camera poses $\mathbf{K}^{c_j w}$ and propagated camera poses of $\mathbf{K}^{c_j w}_{prop}$ according to its distance to where the loop was detected. The interpolation was achieved using linear interpolation for the translations, and the spherical linear interpolation (Slerp) method proposed in [79] for the rotations. In this way, keyframes closer to the current keyframe (and therefore closer to the loop closure point) will be corrected more strongly, whereas the correction for the keyframes located farther from the loop closure point will be weaker.

Once the initial loop correction was performed, a pose graph optimization is carried out to get a more accurate solution. To this end, the residual error between two consecutive keyframes $\mathbf{K}^{c_i w}$ and $\mathbf{K}^{c_{i+1} w}$ is defined as

$$\boldsymbol{\Omega}_i = \mathbf{E}^{c_i c_{i+1}}(\mathbf{K}^{w c_{i+1}})^{-1}\mathbf{K}^{w c_i} = \mathbf{E}^{c_i c_{i+1}}\mathbf{K}^{c_{i+1} w}\mathbf{K}^{w c_i}.$$

At the same time, the residual error between the keyframes that close loops is defined as

$$\boldsymbol{\Omega}_{j,k} = \mathbf{E}^{c_j c_k}(\mathbf{K}^{w c_k})^{-1}\mathbf{K}^{w c_j} = \mathbf{E}^{c_j c_k}\mathbf{K}^{c_k w}\mathbf{K}^{w c_j},$$

where $\mathbf{E}^{c_j c_k}$ is the relative transformation of the detected loop between the keyframes $\mathbf{K}^{c_j w}$ and $\mathbf{K}^{c_k w}$. For the pose graph optimization, the closer to the identity matrix the $\boldsymbol{\Omega}_i$ and the $\boldsymbol{\Omega}_{j,k}$ matrices are, the lower the error will be. For details of this minimization see [80].

Finally, each point on the map is corrected by applying the same transformation that was applied to its original keyframe from where it was triangulated. Figure 4.10 summarizes the loop correction process.

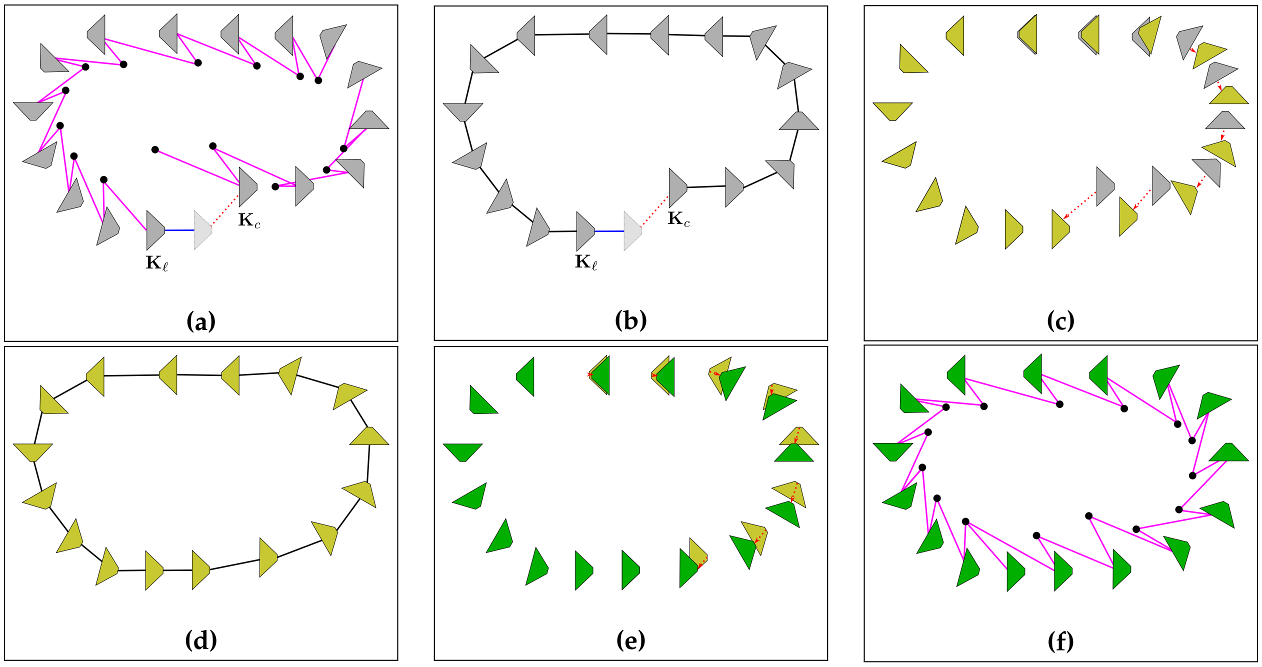


Figure 4.10: Loop correction stages. The trajectory depicted by the keyframe positions is clockwise. **(a)** Pose-point graph with a loop detected. The blue edge represents the relative transformation $\mathbf{E}^{c_c c_\ell}$ between the current keyframe $\mathbf{K}^{c_c w}$ and the loop closing keyframe $\mathbf{K}^{c_\ell w}$. The light gray keyframe indicates the pose where $\mathbf{K}^{c_c w}$ should be. Pink edges correspond to the 3D-2D measurements between keyframes and map points. The red dotted line corresponds to the global error obtained when the loop was detected. **(b)** Pose graph constraints. Black edges are the relative constraints between consecutive keyframe poses. **(c)** Error propagation along the trajectory. Yellow keyframes correspond to the corrected (propagated) keyframes. **(d)** Computed initial solution resulting from propagation. **(e)** Pose graph optimization starting from the keyframe propagation result. The final keyframe poses are shown in green. **(f)** Point correction. Each map point is corrected according to the keyframe that creates it.

4.7.7 Map update and components synchronization

To allow the system to operate in real time along with the loop correction extension, two properties must be satisfied:

- The tracking thread must remain operational in real time being able to work with any map point needed and to create new keyframes if required.
- The local mapping thread must be able to perform the Local Bundle Adjustment over keyframes and map points inside a defined sliding window.

To ensure that these two properties hold, after a loop has been validated, the initial loop correction propagation and pose graph optimization are performed over an internal copy of all the keyframes and map points present in the map. On the other hand, the subset of keyframes and map points observed by them selected by the local mapping thread is defined as the safe mapping window. Thereafter, the map update process, consisting of integrating the loop closure results carried out on the internal copy of the map into the map (resource shared by the tracking, the local mapping and the loop closure threads) is divided into three stages:

1. Update corrected keyframes and map points outside of the defined mapping window.
2. Update corrected keyframes and map points inside the defined mapping window applying any optimization that may have been performed by the Local Bundle Adjustment since the start of the loop closing process.
3. Correct any keyframe created and added to the map after the internal copy has been made. This

correction is achieved applying the same rigid transformation that has been applied for the correction of the current keyframe.

The tracking and mapping threads need to be paused only during the last two stages, where only a small fraction of the map is updated. After the map update, the only remaining part is to notify, to the pose predictor the transformation that must be applied to the ongoing trajectory.

Throughout the process, the tracking may encounter map points that are being actively corrected by the loop closing process, but it is expected that the tracking dismisses those map points while projecting and matching.

4.8 Implementation Details

In this section we explain in detail some relevant implementation decisions that allow the system to run in real time on a mobile platform, minimizing the impact on the pose estimation accuracy.

As keypoint detection and extraction is time consuming, the feature processing for each image of the stereo pair is split into two parallel threads.

Another bottleneck of the tracking phase is matching map points to recently extracted features. Since the map size scales linearly with the traveled distance, checking all points becomes infeasible on the long run. Because of this, only the map points that are in a covisibility area are considered. The covisibility area, for an incoming frame, is determined by all the points that are shared among near keyframes. These keyframes are those that observe the points tracked by the previous frame. Then, this map points are filtered by camera frustum culling. Points initialized from a very different point of view (more than 45 degrees) are also discarded.

Then, the remaining map points are projected onto the image plane to check for matches against the detected features. To speed up this process, detected features are grouped by spatial hashing into grid cells. The matching of a map point is then restricted to the features inside a neighborhood around its projection.

Global map optimization through Bundle Adjustment, as proposed in [17], becomes prohibitive for large scale environments. Consequently we perform only local optimization. The Local Bundle Adjustment (LBA) only refines a fixed number of queued keyframes, along a set of already refined nearby keyframes, and the corresponding subset of visible map points. Unlike PTAM, which runs LBA once for each single keyframe, S-PTAM grabs up to ten queued keyframes to avoid starvation. Our experiments show that the queue size never exceeds four keyframes.

Loop detection has been accomplished using the DBoW2 library [53] configured with $\alpha = 0.3$, temporal consistency $k = 0$ and no geometric check. A visual vocabulary of 6 levels with 10 clusters per level was trained with a combination of indoor and outdoor image sequences from the MIT Stata Center Dataset [81] and the Málaga Urban Dataset [82] with a total of 10 thousand images.

The OpenGV library [83] was used for solving the pose estimation needed for the loop correction with a central absolute variant of the methods aforementioned. A detected loop is validated if 80% of the 3D-2D correspondences between matched keyframes were found as inliers.

The g^2o (General Graph Optimization) library [84] was used to perform Levenberg-Marquardt minimization during tracking, Bundle Adjustment and pose graph optimization after loop closure. Other graph optimization libraries, Vertigo [85, 86] and GTSAM [42], were considered as alternative to g^2o. In [84] is shown a comparison between g^2o and GTSAM library. The comparison showed that g^2o outperforms GTSAM. On the other hand, Vertigo is an extension of g^2o and GTSAM which can solve pose graph optimization problems even with the presence of false positive loop closures. During the experimentation with S-PTAM no false positive loop closure was found (given the strong validation process carried out during the loop detection), and thus the use of Vertigo was dismissed.

The source code was built upon the ROS framework [87], in order to promote its usage by the robotics research community.

Capítulo 4

S-PTAM (resumen)

En este capítulo se presenta el método de SLAM desarrollado durante todo el trabajo de la Tesis. El método lleva el nombre de S-PTAM por las siglas en inglés de *Stereo Parallel Tracking and Mapping*.

S-PTAM comienza con la cámara en el origen del sistema de referencia del mundo y un mapa vacío, que es inicializado de inmediato mediante la triangulación de las correspondencias obtenidas entre las características visuales extraídas en el primer par estéreo. A partir de entonces, el hilo de *tracking* calcula la pose de cada nuevo frame estéreo, reduciendo el error de reproyección determinado por las proyecciones de los puntos del mapa y las características correspondientes en las imágenes. Una vez estimada la pose actual de la cámara, el frame estéreo podría ser seleccionado para ser añadido al mapa (en este caso es llamado *keyframe*). Si se añade un keyframe al mapa, entonces los features no emparejados con un punto del mapa son triangulados a partir del par estéreo. Acto seguido, la pose junto con los nuevos puntos del mapa son añadidos al grafo de puntos y poses (*point-pose graph*) mantenido por el S-PTAM. Una vez que el keyframe fue procesado por el tracking, el hilo de mapping realiza una búsqueda de correspondencias entre puntos del mapa y keyframes con el fin de reforzar las restricciones del grafo. Inmediatamente después, el hilo de mapping realiza un Bundle Adjustment sobre el subgrafo definido por los últimos keyframes añadidos. Después del refinamiento, las mediciones marcadas como outliers por la función de coste robusta durante la optimización son removidas. Finalmente, los puntos del mapa que no tienen ninguna medición son eliminados del mapa.

Para poder corregir la deriva acumulado en trayectorias largas, S-PTAM realiza una detección y cierre de ciclos en un tercer hilo de ejecución. Este hilo busca candidatos a cierre de ciclos comparando los keyframes procesados en términos de apariencia mediante el uso de un modelo *Bag of Words*. Un candidato a ciclo es validado si se encuentra una transformación relativa que este soportada por la mayoría de las correspondencias entre los dos keyframes de cierre. Una vez que es validado un ciclo, una grafo de poses (*pose-graph*) es calculado partir del *point-pose graph*, mediante la marginación de los nodos referentes a los puntos del mapa. A este grafo de poses, se añade la transformación relativa como una restricción fija para luego realizar la corrección del ciclo. Finalmente, la corrección del ciclo se lleva a cabo en dos etapas. En primer lugar, una corrección inicial se calcula distribuyendo el error acumulado a lo largo de todos los keyframes que componen el ciclo. En segundo lugar, una optimización es realizada sobre el grafo de poses tomando como semilla la corrección inicial.

Chapter 5

Experimental results

In this chapter we present several experiments to assess the performance, the accuracy and the robustness of S-PTAM. Moreover, we provide an exhaustive evaluation of the impact caused by the use of the most relevant feature detectors and descriptor extractors on S-PTAM. This result can be extrapolated to other feature-based SLAM systems. For all the experiments, a standard laptop with an Intel Core i7 @ $2.8\,\mathrm{GHz}$ processor and $16\,\mathrm{GB}$ RAM was used.

5.1 Datasets

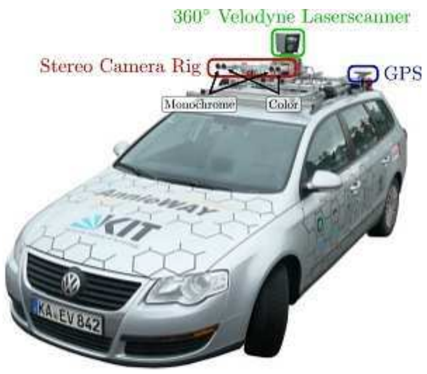
To assess the accuracy, robustness and computational cost of the S-PTAM system with the loop closure extension, the KITTI Vision Benchmark Suite dataset [88] and the Indoor Level 7 S-Block dataset [89] were used. They cover both outdoor large driving scenarios as well as indoor robotics, respectively.

5.1.1 KITTI Vision Benchmark Suite

The KITTI dataset provides a standard benchmarking framework which helps to compare the performance of our method to other state-of-the-art stereo vision-based SLAM systems. This dataset presents dynamic objects, changing light conditions and fast camera motions. The dataset is composed by twenty three sequences taken from a car going through an urban environment. The standard length of each sequence is in the order of various kilometers. There are eleven training sequences where the ground-truth is known and twelve test sequences. The stereo camera, mounted on the front of the vehicle, has a $\sim 60\,\mathrm{cm}$ baseline and a resolution of 1344×391 pixels and runs at a frame rate of $10\,\mathrm{Hz}$. Figure 5.1 depicts the sensors mounted on the vehicle used for the KITTI dataset and a set of sample frames captured along the sequences of the KITTI dataset.

5.1.2 Level 7 S-Block

The Level 7 S-Block dataset corresponds to a wheeled robot moving around an office environment under artificial illumination conditions. The stereo camera mounted on the robot has a $\sim 40\,\mathrm{cm}$ baseline and a resolution of 1280×1960pixels at a frame rate of $12\,\mathrm{Hz}$. During the trajectory of the robot, several loop closures are made over an extended period of time (more than 30 minutes). Figure 5.2 shows the robot platform used and sample frames captured along the robot trajectory.



(a)

(b)

Figure 5.1: **(a)** Sensors used in the KITTI dataset. **(b)** Sample frames selected from the KITTI dataset sequences.

(a)

(b)

Figure 5.2: **(a)** The Adept Guiabot used on the *Level 7* dataset. **(b)** Sample frames of the *Level 7* dataset.

5.2 Metric error

To assess the final impact of the different experiment configurations on the accuracy of S-PTAM, we extend a commonly used metric [90, 88] specifically designed for evaluating the performance of SLAM systems. Let $\mathbf{E}^{c_k w}$ be the estimated pose at frame k and $\mathbf{E}^{c_k w *}$ the corresponding ground truth pose. Then, we can express the difference (or motion) between two frames of a sequence as $\boldsymbol{\delta}^{c_i c_j} = \mathbf{E}^{c_i w}(\mathbf{E}^{c_j w})^{-1}$ [91], and analogously, for the difference between two ground truth frames we have $\boldsymbol{\delta}^{c_i c_j *} = \mathbf{E}^{c_i w *}(\mathbf{E}^{c_j w *})^{-1}$.

The *relative error* committed between frames i and j becomes $\boldsymbol{\delta}^{c_i c_j}(\boldsymbol{\delta}^{c_i c_j *})^{-1}$. The *root-mean-square error* (RMSE) metric that we will use to compare the results of S-PTAM with the ground-truth is based on the relative error. To obtain meaningful numerical results, we need to separate the error into a translational part ϵ_t and a rotational part ϵ_θ since they are different in nature. This separation is also suggested by the original authors [90]. Then, this RMSE metric is defined as

$$\epsilon_t = \sqrt{\frac{1}{N}\sum_{i,j} \operatorname{trans}\left(\boldsymbol{\delta}^{c_i c_j}(\boldsymbol{\delta}^{c_i c_j *})^{-1}\right)^2}$$

$$\epsilon_\theta = \sqrt{\frac{1}{N}\sum_{i,j} \operatorname{rot}\left(\boldsymbol{\delta}^{c_i c_j}(\boldsymbol{\delta}^{c_i c_j *})^{-1}\right)^2}.$$

where N is the number of relative displacements $\boldsymbol{\delta}^{c_i c_j}$, $\operatorname{trans}(.)$ is a function that returns the translation $\mathbf{t}$ part of a given displacement and $\operatorname{rot}(.)$ is a function that takes a displacement and calculates the angle θ

derived from the rotation part as follows

$$\operatorname{rot}(\mathbf{E}) = \arccos\left(\frac{\operatorname{Tr}(\mathbf{R}) - 1}{2}\right).$$

This proposed metric differs from the original metric [90] in that it takes the square root, which helps in interpreting numerical results, since the measurement units are the same as for the data.

This equations intentionally leave open the choice of which relative displacements $\boldsymbol{\delta}^{c_i c_j}$ are included in the metric. As discussed by the original authors [90], the choices will highlight different properties of the data. In our case, we strive for local consistency, which is better highlighted by taking displacements as small as possible. Therefore relative displacements are taken between consecutive frames, yielding:

$$\boldsymbol{\epsilon}_t = \sqrt{\frac{1}{N}\sum_k \operatorname{trans}\left(\boldsymbol{\delta}^{c_k c_{k+1}}(\boldsymbol{\delta}^{c_k c_{k+1}*})^{-1}\right)^2}$$

$$\boldsymbol{\epsilon}_\theta = \sqrt{\frac{1}{N}\sum_k \operatorname{rot}\left(\boldsymbol{\delta}^{c_k c_{k+1}}(\boldsymbol{\delta}^{c_k c_{k+1}*})^{-1}\right)^2}.$$

In the next section we will use this RMSE metric to compare the accuracy and time performance of S-PTAM for different feature extractors.

5.3 Features Evaluation

In this section, we assess the impact of image feature extractors on the performance of the S-PTAM system in terms of pose accuracy and computational requirements. Although the evaluation is performed using the S-PTAM system, the obtained results can be generalized for other stereo feature-based SLAM systems.

An image feature extractor can usually be split into two phases, detection and description. The feature detector is used to find salient areas in the image, while the feature descriptor captures and synthesizes the information in a local neighborhood of the selected area. A brief overview of the most commonly used feature extractor and descriptor algorithms, which were considered for the comparison, is presented below.

STAR A modified version of the CenSurE (Center Surrounded Extrema) detector [9], which is computationally less demanding at the expense of lower precision.

FAST Features from Accelerated Segment Test [11] is a feature detector focused on lowering the computational cost.

AGAST Adaptive and Generic Accelerated Segment Test, a corner detector based on FAST. Unlike FAST, AGAST does not have to be trained for a specific scene, but it dynamically adapts to the environment while processing an image [12].

Shi–Tomasi (GFTT) Good Features To Track is a detector focused on selecting features relevant to motion tracking by analyzing the amount of information they provide for that particular task [10].

BRIEF Binary Robust Independent Elementary Features [14] is a descriptor that describes an image area using a number of intensity comparisons of random pixel pairs. It is saved as a binary string, which reduces the computational complexity of the subsequent matching. BRIEF is not invariant to scale and orientation changes.

ORB Oriented FAST and Rotated BRIEF [13] is another attempt to achieve a scale and rotation invariant BRIEF, as a computationally efficient alternative to SIFT and SURF. It uses the FAST detector to achieve low computational requirements.

BRISK Binary Robust Invariant Scalable Keypoints [15] is a scale and rotation invariant version of BRIEF, but unlike BRIEF, it uses a deterministic comparison pattern.

LATCH Learned Arrangements of Three patCH codes is a binary descriptor extractor which computes each value descriptor vector through the comparison of patches instead of solely pixels as BRIEF or BRISK extractors. By comparing patches, visual information with more spatial information support is considered for each of the descriptor's bits, and their values are therefore less sensitive to noise [16].

Moreover, SIFT [7] and SURF [8] feature extractors were considered during the evaluation; however, the results are not plotted in the thesis given that they do not achieve good accuracy, and the high computational cost demanded by them is considerably higher than the binary features extractors.

The ORB descriptor relies on its own detector (ORB). For each of the BRIEF, BRISK and LATCH descriptors, the combinations with GFTT, FAST, AGAST and STAR detectors are considered. This amounts to thirteen detector-descriptor pairs that are evaluated in the present work.

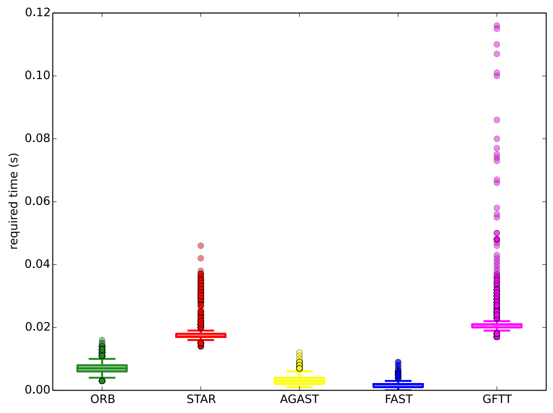
The experiments are performed over the KITTI Vision Benchmark Suite [88], which provides a reliable ground truth for several training sequences. Results are computed over all the training sequences, except for sequence 01 which records a car driving in a highway at high speed, together with a low feature scene, rendering the system ill-conditioned for visual odometry.

The quality of each feature extractor is measured directly as the error 5.2 committed when running the S-PTAM system over the sequences with that particular configuration, with respect to the provided ground truth. In Table 5.1 the parameters used for each feature extractor are detailed. To make the drift produced observable by each feature extractor , the loop closure module was disabled for this set of experiments.

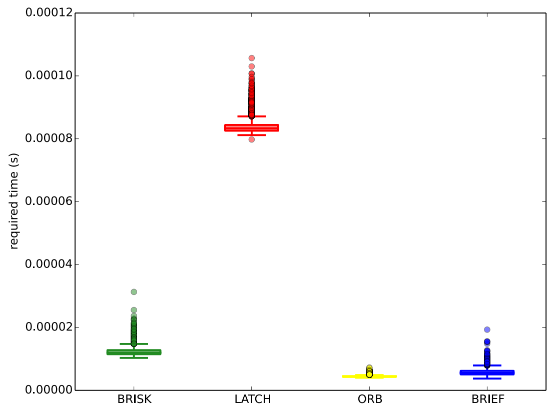
Detector / Descriptor	Parameters	Value
STAR	responseThreshold	20
FAST	threshold	60
AGAST	threshold	60
GFTT	minDistance	15.0
BRIEF	hammingThreshold	25
ORB	nLevels	1
	hammingThreshold	50
BRISK	hammingThreshold	100
LATCH	hammingThreshold	45
	rotationInvariance	false

Table 5.1: Parameters used for the feature detectors and descriptors in the evaluation. The parameters not appearing in the list use the default value in the OpenCV 3 implementation. We used the Hamming distance as the metric for the descriptor similarity, as all of them are binary.

An important requirement for every feature extractor is to track the camera pose in real time. First of all, the extraction cost should be low. In Figure 5.3 it can be seen how the different detectors and descriptor extractors perform in the context of the KITTI dataset. Note that the extraction time required by each method depends heavily on the processing power and resolution of the images. On the other hand, the number of extracted features also has a direct impact on the performance, since the map, and thus the amount of tracked points, scale with it. In Figure5.3a, GFTT is the most variable detector, presenting time demanding outliers. In real-time operation, each outlier may cause the loss of a stereo frame. However, losing a few scattered frames does not compromise S-PTAM's localization as we shall see in the on-line experiments presented in Section 5.4.



(a) Feature detection time for each image.



(b) Descriptor extraction time for each feature.

Figure 5.3: Feature extraction times. Data were measured over all KITTI training sequences (except 01). Boxes represent interquartile range (IQR), whiskers reach to $-1.5 \times IQR$ and $1.5 \times IQR$, and the points represent data beyond those ranges, considered outliers. The line inside the box represents the median.

Analyzing the accuracy obtained by S-PTAM running in off-line mode (having enough time to process each stereo frame), it is possible to see which feature extractor achieves the best accuracy. Table 5.2 and Table 5.3 show the RMSE translation and rotation errors achieved upper bounding the number of features to be extracted per frame. Upper bounds start at 500 features, given that almost all extractors fail to localize with a lower number of features. STAR / BRISK combination was the only one that was able to operate over all sequences with a 250 features upper bound. The ORB / ORB and GFTT / BRISK combinations outperform the others under the evaluated error metric. It is important to clarify that the number of features detected by GFTT remains around ~ 500 features despite the selected upper bounds. This is determined by the detector's characteristics and its implementation. The tables show that in general at greater number of features extracted, a greater number of features are tracked, and therefore a better accuracy is obtained. In particular, ORB / ORB clearly presents the aforementioned tendency. Nevertheless, the time performance of the system gets compromised if too many features are tracked. For real-time operation the number of features to be extracted should be carefully selected. Figure 5.4 shows the translation and rotation RMSE errors obtained by S-PTAM running in on-line mode with ORB / ORB features. Errors decrease up to ~ 1500 features, and increase rapidly thereafter due to the high number of frames lost.

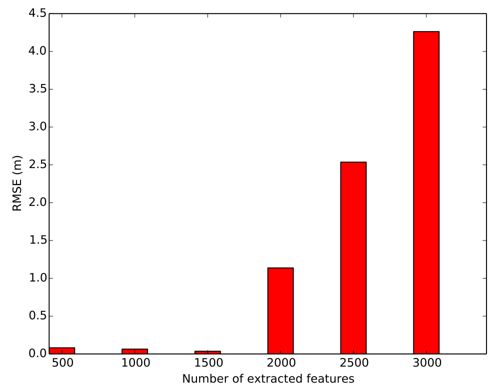
The experiments presented in the next section were carried out using the GFTT algorithm for the detection of features and BRISK was selected as the feature descriptor. At first this decision appears to be in conflict with previous works [92, 93] on binary feature evaluation. In [93] the FAST / BRIEF extractor is recommended in the same context as the experiments conducted in the present work. However, it does not consider the complexity of the further processing required by the huge amount of points extracted, and bases its detector choice solely on its speed. In [92], the BRIEF descriptor is preferred over BRISK, but BRISK is only paired with the AGAST detector. In [93] BRISK is not even considered.

	Features extracted per frame				
Extractor	500	1000	1500	2000	2500
AGAST / BRIEF	0.0601	0.0448	0.0441	0.0491	0.0452
AGAST / BRISK	0.0425	0.0361	0.0357	0.0357	0.0356
AGAST / LATCH	0.0887	0.0677	0.0796	0.1089	0.0793
FAST / BRIEF	0.0563	0.0511	0.0538	0.0473	0.0601
FAST / BRISK	0.0423	0.0342	0.034	0.0337	0.0345
FAST / LATCH	0.0906	0.0661	0.0891	0.1213	0.0727
GFTT / BRIEF	0.0333	0.0322	0.0322	0.0319	0.0317
GFTT / BRISK	**0.0301**	**0.0299**	**0.0299**	**0.0293**	0.0294
GFTT / LATCH	0.0443	0.0419	0.0419	0.0423	0.042
ORB / ORB	0.0462	0.0363	0.0321	0.0305	**0.0293**
STAR / BRIEF	0.0393	0.0368	0.037	0.0375	0.0371
STAR / BRISK	0.0449	0.0436	0.0415	0.0521	0.0529
STAR / LATCH	0.0589	0.0525	0.0496	0.0496	0.0495

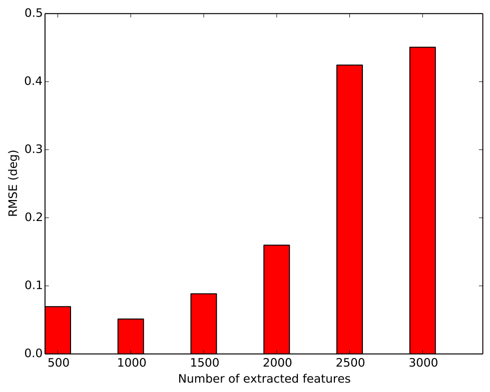
Table 5.2: Translation RMSE errors obtained for each feature extractor over all KITTI training sequences (except 01) limiting the number of features to be extracted. Good (small) relative error implies local consistency, sufficient for navigation.

	Features extracted per frame				
Extractor	500	1000	1500	2000	2500
AGAST / BRIEF	0.095	0.0833	0.0795	0.0835	0.0825
AGAST / BRISK	0.0881	0.0742	0.0768	0.0784	0.0767
AGAST / LATCH	0.119	0.0984	0.0976	0.0973	0.0955
FAST / BRIEF	0.0939	0.0851	0.0814	0.0813	0.0853
FAST / BRISK	0.0838	0.0764	0.0755	0.0737	0.0746
FAST / LATCH	0.1142	0.1001	0.0965	0.1002	0.0966
GFTT / BRIEF	0.0771	0.0745	0.0746	0.0744	0.0741
GFTT / BRISK	**0.0756**	**0.0741**	0.0741	0.0746	0.0738
GFTT / LATCH	0.0897	0.0863	0.0898	0.0866	0.0892
ORB / ORB	0.087	0.0762	**0.0724**	**0.0699**	**0.068**
STAR / BRIEF	0.0798	0.076	0.0747	0.0745	0.0759
STAR / BRISK	0.0848	0.08	0.0765	0.0768	0.0766
STAR / LATCH	0.102	0.0914	0.0895	0.0913	0.0915

Table 5.3: Rotation RMSE errors obtained for each feature extractor over all KITTI training sequences (except 01) limiting the number of features to be extracted. Good (small) relative error implies local consistency, sufficient for navigation.



(a) Translation RMSE errors.



(b) Rotation RMSE errors.

Figure 5.4: RMSE translation and rotation errors obtained by S-PTAM running in on-line fashion on sequence 04, with ORB / ORB as feature extractor. Six experiments were carried out changing the number of features to be extracted.

5.4 S-PTAM evaluation

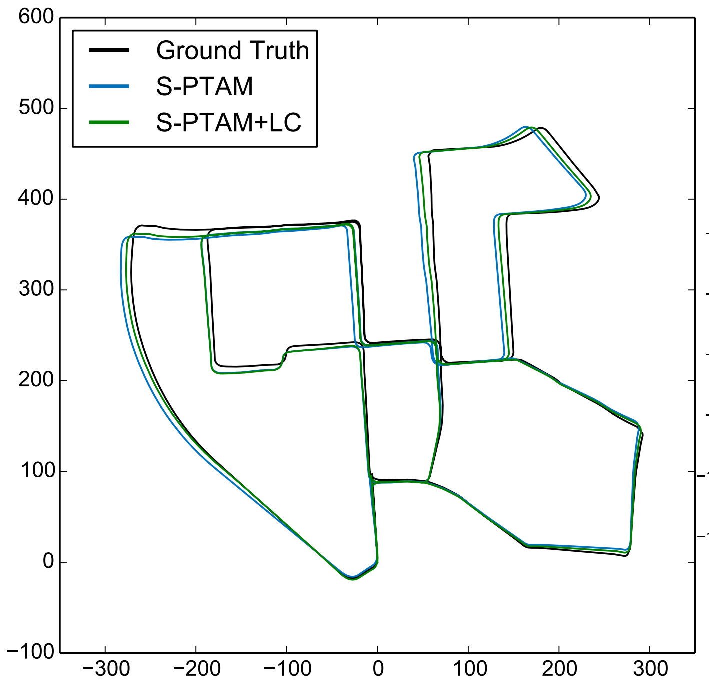
In this section S-PTAM is evaluated over the KITTI and the Level 7 datasets to assess its accuracy and time performance.

5.4.1 Evaluation on KITTI

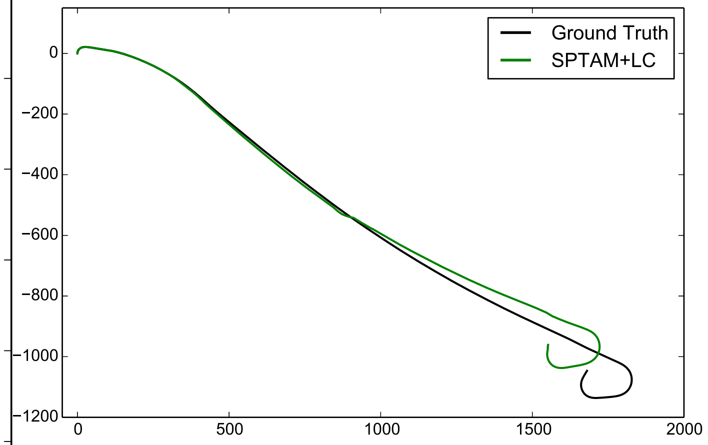
Figures 5.5, 5.6 and 5.7 show the performed trajectories estimated by S-PTAM with the loop closure extension compared with the ground truth. For those sequences where loops were detected (00, 02, 05, 06 and 07), a comparison between S-PTAM with loop closure extension and S-PTAM without loop closure is presented. Implemented methods for the loop detection and validation have shown to be robust as no false positives have occurred in any of the evaluated sequences. Figure 5.8 depicts the map reconstructed by S-PTAM for the KITTI sequence 00. Figure 5.9 shows the loops that were detected over the sequence 00, the Z-axis represents time and red lines link pairs of keyframes that were matched as positive loops. The loop correction proved to be able to operate without disrupting the tracking continuity. Figure 5.10 shows the absolute translation and rotation error respectively at each moment of the sequence 00. Absolute errors of the system with and without loop closure extension are presented. It can be seen that the first loop correction occurred after a significant period of time without any loops (where accumulated drift error increases substantially), significantly improving the global localization of the system. When a loop correction occurs, the translation error gets adjusted to the values registered at the time that the place was first visited. In Figure 5.10, between seconds 350 to 400, the car revisits a section previously mapped. It is interesting to note that the absolute error is not further reduced with higher numbers of detected loops. This is due to the accumulated error being already eliminated by the first loop closed in that segment. Peaks on measured error denote areas with low texture or high-speed turns. During the $\sim 4\,\mathrm{km}$ trajectory followed by the car, the maximum absolute localization error was of less that $15\,\mathrm{m}$.

Table 5.4 shows the performance of the loop detection and geometric validation methods on sequences that present loops in the trajectory (00, 02, 05, 06, 07 and 09). Loop associations proposed in [94] were used as ground truth. The appearance-based loop detection is permissive, generating a high number of detections with a large percentage of false positives. In contrast, the geometric validation implemented rejects false positives with 100% precision at expense of a lower recall. The number of loops finally validated is proportionally low compared to the amount of loops defined in the ground truth. As we have mentioned before, several loop corrections in close succession do not significantly improve the global localization. The method focuses on fewer loops with higher number of inlier correspondences. S-PTAM fails to detect a loop that occurs in the last 17 frames of sequence 09.

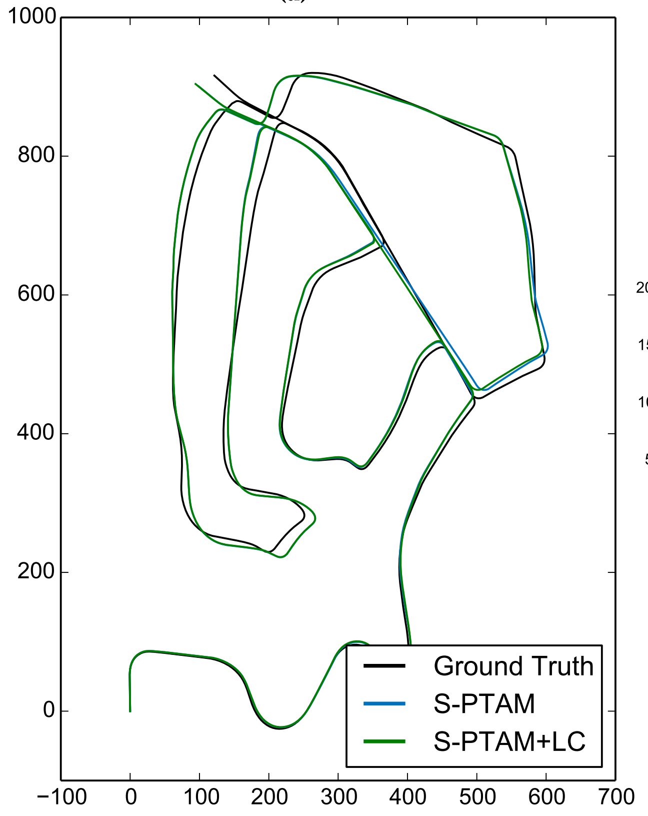
Table 5.5 shows the average temporal performance, measured for the costliest subroutines of the tracking process. Despite of the time consumed by the loop closure procedures, the tracking thread runs at $\sim 18\,\mathrm{Hz}$.



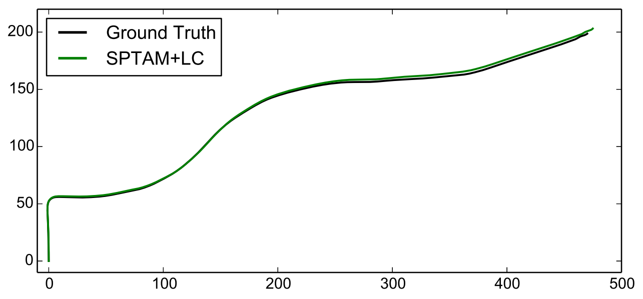
(a)



(b)

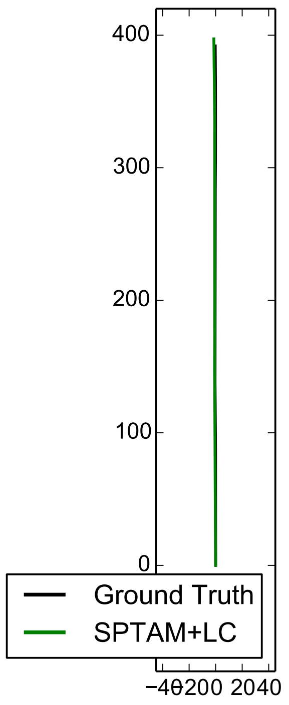


(c)

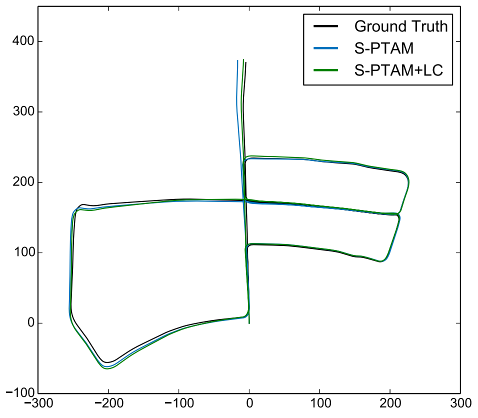


(d)

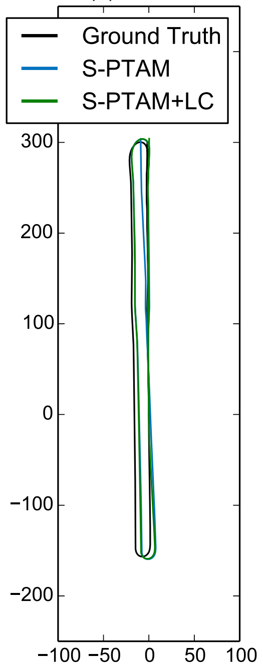
Figure 5.5: Trajectories of **(a)** 00, **(b)** 01, **(c)** 02 and **(d)** 03 sequences.



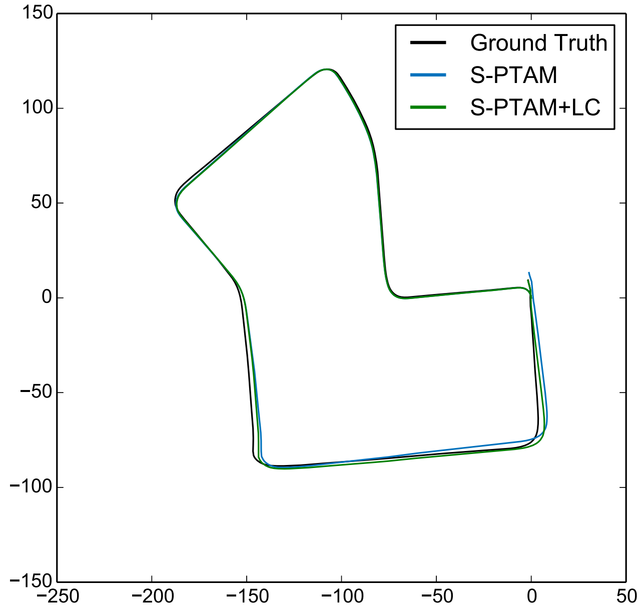
(a)



(b)

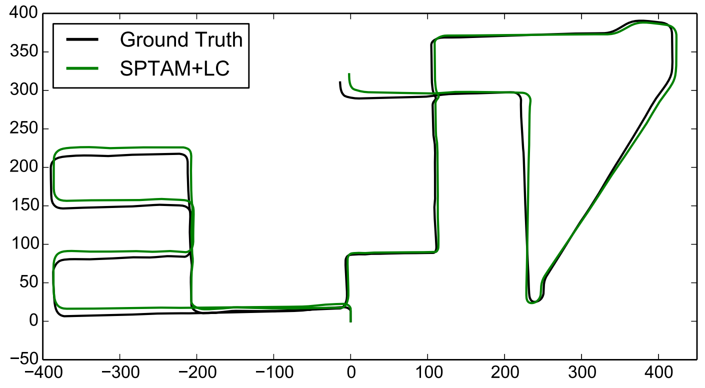


(c)

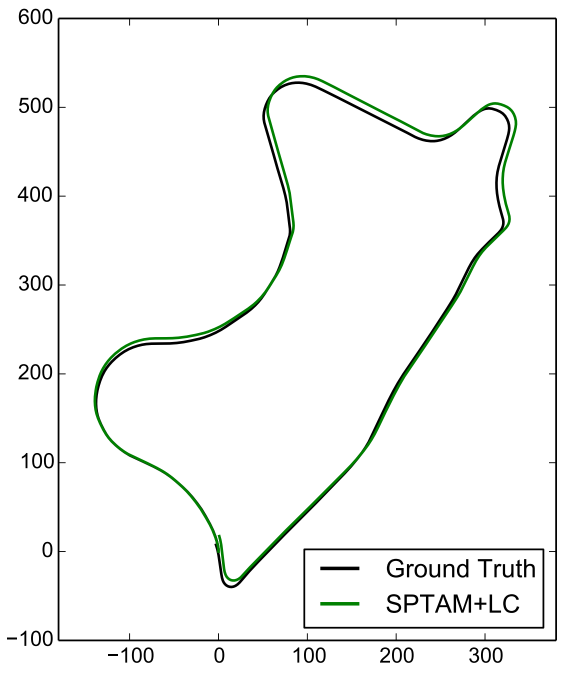


(d)

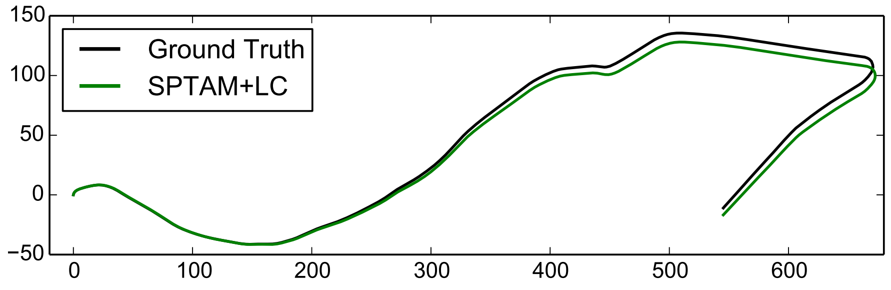
Figure 5.6: Trajectories of **(a)** 04, **(b)** 05, **(c)** 06 and **(d)** 07 sequences.



(a)



(b)



(c)

Figure 5.7: Trajectories of **(a)** 08, **(b)** 09 and **(c)** 10 sequences.

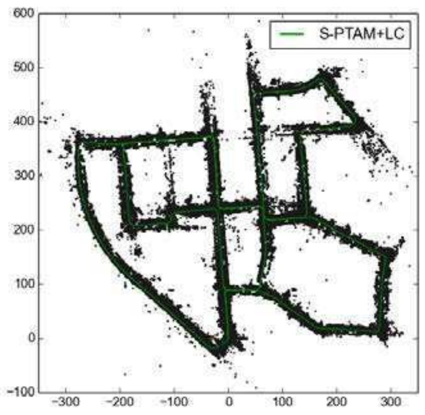


Figure 5.8: Map reconstructed by S-PTAM on sequence 00.

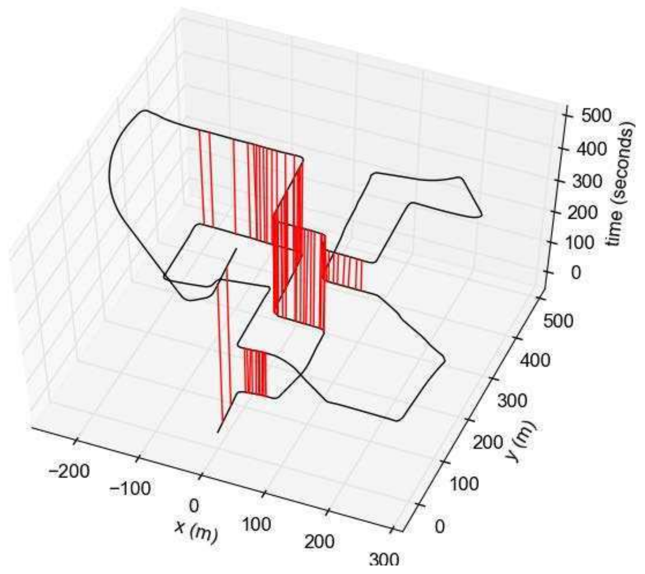
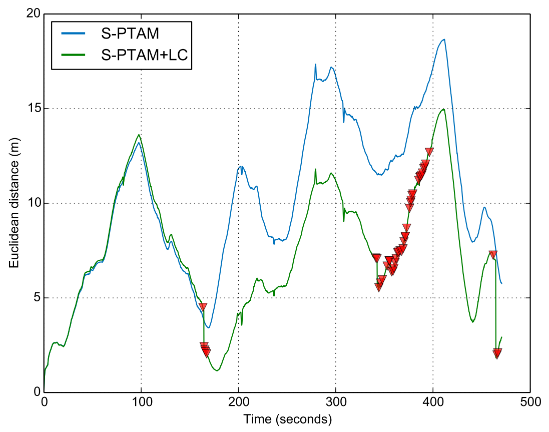


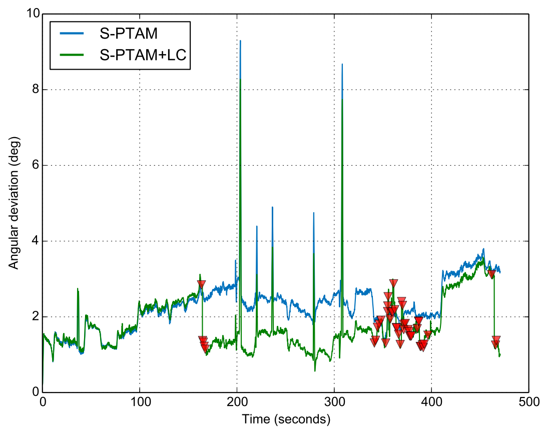
Figure 5.9: Loops detected over time on sequence 00. Red lines link pairs of matched keyframes.

Sequence	#Loops	Appearance			Appearance+ Geometric validation		
		#Detections	%Precision	%Recall	#Validations	%Precision	%Recall
KITTI00	732	2747	13,14	49,32	45	100	6,14
KITTI02	234	3010	3,12	40,17	5	100	2,14
KITTI05	320	1596	12,4	61,88	28	100	8,75
KITTI06	269	412	24,03	36,8	16	100	5,95
KITTI07	13	434	2,07	69,23	1	100	7,69
KITTI09	17	836	0,60	29,41	0	⁄	0

Table 5.4: Precision and recall results on KITTI sequences.



(a) Absolute translation error.



(b) Absolute rotation error.

Figure 5.10: Absolute translation and rotation errors on sequence 00. Red markers show when a loop was validated.

Tracking phase	time (ms)
Feature Extraction and description	31.53
Get Points (inside Frustum)	4.37
Matching	4.71
Pose Update	0.99
AddKeyFrame	13.62
Total	55.22

Table 5.5: Tracking phase average processing time per frame.

5.4.2 Evaluation on Level 7

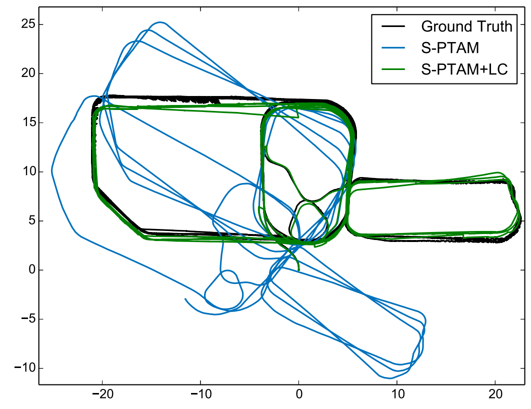
Unlike the KITTI dataset, which presents a low number of widely separated loops, the Level 7 dataset features a high number of them. Besides allowing to test S-PTAM in a different environment, this dataset also allows to make a proper evaluation of the loop closure extension in terms of time requirements. In Figure 5.11 the trajectory estimated by S-PTAM with and without the loop closure extension is presented, along with the loops that have been validated. The loop validation process implemented shows to be robust and accurate, given that, even when the scene is highly repetitive no false positive loops are detected.

In the Level 7 dataset 3706 appearance analysis, 2218 geometric analysis and 317 loop closures were performed. Table 5.6 details the processing times took by each loop closing phase. As we have mentioned in section 4.7.7 the Loop Closing module was designed to work completely in parallel with the tracking and local mapping threads. The tracking and the local mapping threads are stopped for only $1.6\,\mathrm{ms}$ on average. Note that the safe mapping window is independent of the map size. Conversely, the times required by the Loop Closing module to adjust the map and trajectory outside the safe mapping window grows with the size of the map. However, the latter is not an issue given that the optimization runs in a parallel thread, and it does not affect the tracking thread.

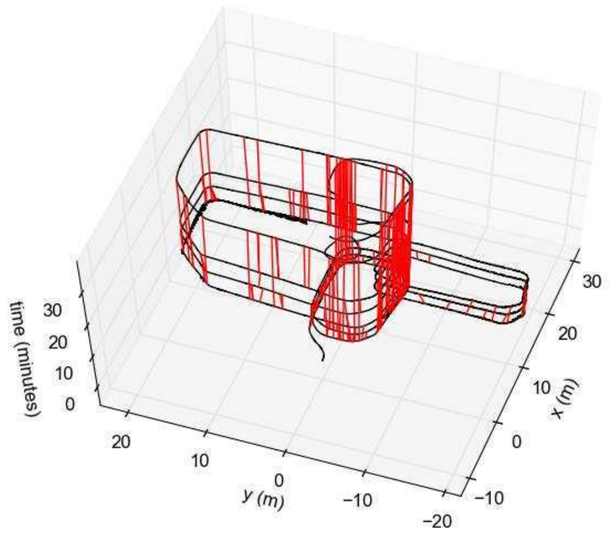
Loop Closure phase	#Measurements	Min (ms)	Avg (ms)	Max (ms)
Detections	3706	0	0,74	8
Validation and Relative transformation	2218	0	2,31	11
Loop correction	317	42	384,92	922
Map update	317	17	80,53	187
Map update (safe mapping window)	317	0	1,6	4

Table 5.6: Loop closing phase processing times.

Figure 5.12 shows that the loop correction processes and the map update (outside the safe mapping window) scale linearly with the number of keyframes. Note that the gaps between measurements indicate that there was no loop detected in that timespan.

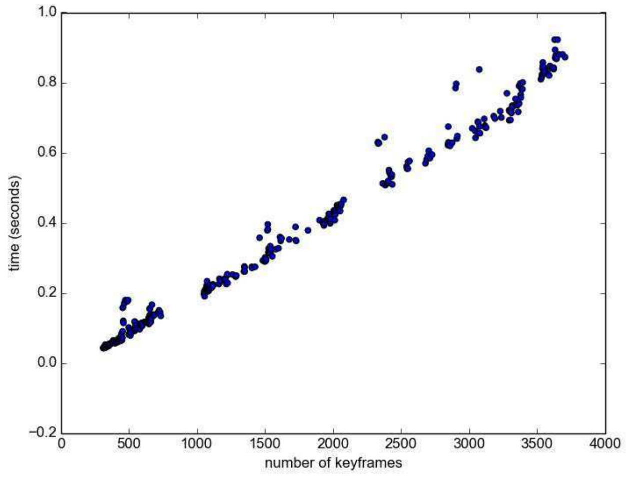


(a) Estimated trajectory.

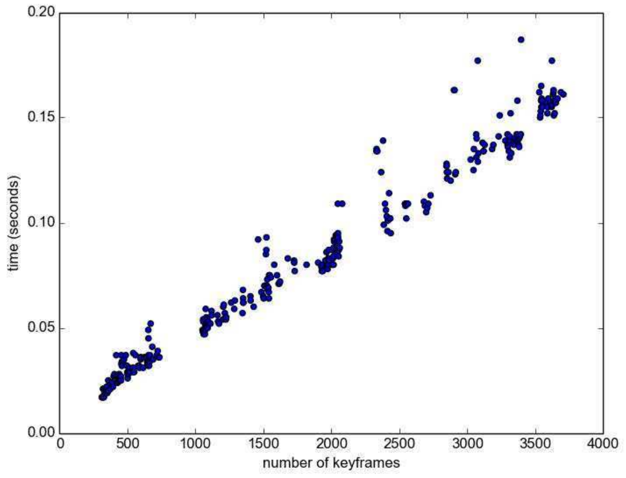


(b) Loops detected over time. Red lines link pairs of matched keyframes.

Figure 5.11: Estimated trajectory and loops detected over time on Level 7 dataset. Note that ground-truth information presents segments with some noise.



(a)



(b)

Figure 5.12: **(a)** Loop correction (initial loop correction and pose graph optimization) time over the number of keyframes in the map. **(b)** Map update times (for keyframes outside the safe mapping window).

5.5 Comparison with other SLAM systems

This section aims to compare S-PTAM with other state-of-the-art SLAM systems on the KITTI and the Level 7 datasets.

5.5.1 Evaluation on KITTI

The KITTI benchmark presents an exhaustive comparison of several state-of-the-art SLAM systems in the context of outdoor driving scenarios. In the benchmark, the errors measured are a form of relative mean square errors (MSE), normalized over distances and velocities. See [88] for further details on how this errors are computed.

In Table 5.7, which is an excerpt of the ranking on the benchmark website [95], S-PTAM is compared to the stereo version of ORB-SLAM2 [52] and the S-LSD-SLAM [58] system. Both are state-of-the-art reference systems in the visual SLAM community. ORB-SLAM2 presents the best translation error whilst S-PTAM gets the best rotation error. The direct stereo SLAM system S-LSD-SLAM performs worse than the feature-based ones in this context.

Method	Translation error (%)	Rotation error (deg/m)
ORB-SLAM2	**1.15**	0.0027
S-PTAM	1.19	**0.0025**
S-LSD-SLAM	1.20	0.0033

Table 5.7: Comparison of S-PTAM, ORB-SLAM2 and S-LSD-SLAM in KITTI Benchmark.

5.5.2 Evaluation on Level 7

In this section, we compare ORB-SLAM2 and S-PTAM systems over the Level 7 dataset. Figure 5.13 shows the trajectory estimated by both systems; and Figure 5.14 presents the absolute translation and rotation errors obtained. The figures show that S-PTAM and ORB-SLAM2 have comparable accuracy and both present similar error peaks around the same areas.

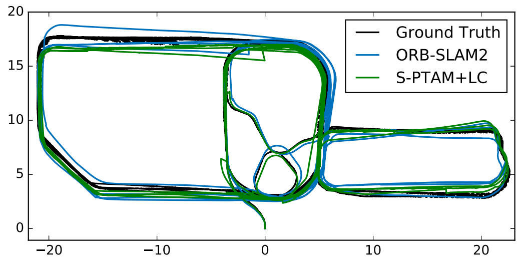
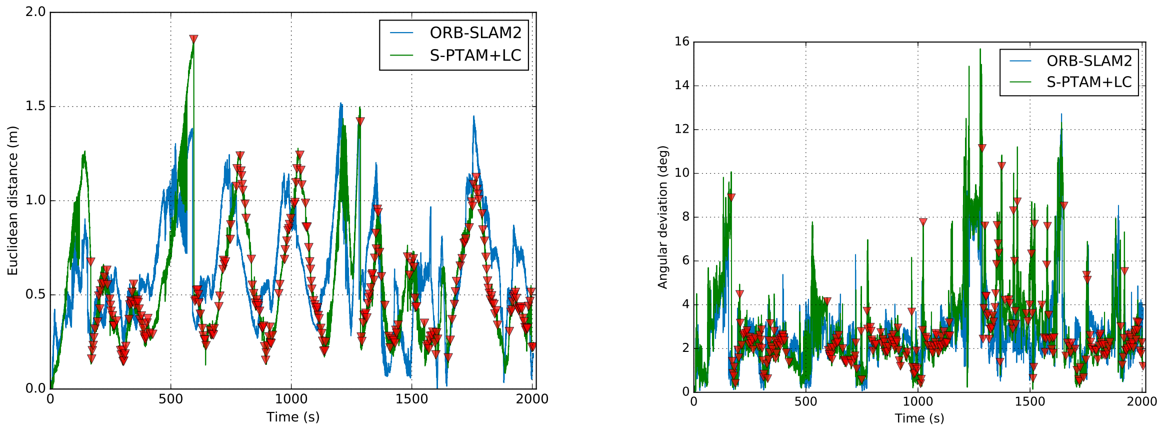


Figure 5.13: Comparison between trajectories estimated by ORB-SLAM2 and S-PTAM. Note that ground-truth information presents segments with some noise.



(a) Absolute translation error.

(b) Absolute rotation error.

Figure 5.14: Absolute translation and rotation errors obtained by ORB-SLAM2 and S-PTAM. Note that ground-truth information presents segments with some noise.

Capítulo 5

Resultados experimentales (resumen)

En este capítulo se presentan los resultados obtenidos en términos de rendimiento, precisión y robustez al evaluar S-PTAM en dos datasets de dominio público. Además, presentamos una evaluación exhaustiva del impacto que causa el uso de distintos extractores de características del estado del arte sobre S-PTAM. El resultado de dicha evaluación, si bien fue realizado sobre S-PTAM, puede ser extrapolado para otros sistemas SLAM basados en características visuales.

Un extractor de características de imagen por lo general se puede dividir en dos fases: la fase de detección y la fase de descripción. En la fase de detección se utilizá un algoritmo para encontrar puntos de alto gradiente (*keypoints*) de la imagen. En la fase de descripción se utiliza un algoritmo para describir de manera unívoca el punto detectado. Para esto el algoritmo de descripción captura y sintetiza la información de apariencia de la vecindad del punto detectado.

La evaluación de los extractores de características visuales consiste en determinar qué combinación detector/descriptor es la que proporciona mejores resultados en términos de precisión y costo computacional. Para la evaluación de las características visuales se utiliza el dataset público *KITTI Vision Benchmark Suite*, que está compuesto por secuencias de imágenes tomadas desde un automóvil que transita en un entorno urbano. El dataset proporciona un *ground-truth* para cada secuencia de entrenamiento. Para poder observar el error cometido por cada extractor de características, el módulo de Loop Closure es desactivado durante la evaluación. En la evaluación se utilizan los detectores GFTT, FAST, AGAST STAR y ORB; y los descriptores ORB, LATCH, BRIEF y BRISK. En particular, el descriptor de ORB se basa en su propio detector. Para los detectores GFTT, STAR, FAST y AGAST se consideraron todas las posibles combinaciones con los descriptores BRIEF, BRISK y LATCH. Esto equivale a un total de diez combinaciones de detector/descriptor que se evalúan en el presente trabajo. De la evaluación se concluye que GFTT/BRISK es la mejor combinación.

Para la evaluación del sistema S-PTAM en su totalidad -junto con el módulo de Loop Cloure- aparte de utilizar el dataset de KITTI, se utiliza el dataset *Level7 S-Block*. El dataset Level7 consta de una secuencia de imágenes tomadas por un robot móvil que se desplaza en un entorno de oficinas. De esta manera ambos datasets se complementan y proporcionan una evaluación completa del método propuesto. Como resultado de la evaluación, se observa que S-PTAM provee una localización precisa para cada secuencia. Asimismo, es posible apreciar en los experimentos cómo el módulo de Loop Closure reduce notoriamente el error de deriva acumulado por el sistema.

Finalmente se compara S-PTAM con los sistemas ORB-SLAM2 y S-LSD-SLAM, los cuales son dos de los sistemas de Visual SLAM estéreos más relevantes del estado del arte. En particular, S-LSD-SLAM es un método directo y ORB-SLAM2 es un método basado en características ORB. En los resultados obtenidos en el dataset de KITTI, ORB-SLAM2 presenta el mejor error de traslación, mientras que S-PTAM consigue el mejor error de rotación. El sistema directo S-LSD-SLAM, obtiene peores resultados que los basados en características. Además, se realiza una comparación con el sistema ORB-SLAM2 sobre el dataset Level7. En dicha comparación se puede observar que la precisión de ambos sistemas son comparables.

Chapter 6

Conclusions

In this Thesis, we present a stereo SLAM system for robot localization called S-PTAM (from Stereo Parallel Tracking and Mapping). S-PTAM incrementally builds a point-based sparse map representation of the workspace, using a stereo camera, and tracks the camera pose within it. To allow S-PTAM to run in large scale environments and to respond in real-time, the SLAM problem is heavily parallelized, separating the tracking and the map refinement routines into two threads, while minimizing inter-thread dependency. Moreover, to obtain global localization in large scale maps, a loop closing module capable to detect previously visited places and to perform a reduction of the accumulated error was developed. The Loop Closing module was designed in order not to disrupt the tracking and local mapping thread allowing the system to operate in real-time.

For each incoming stereo frame, S-PTAM extracts the image features from left and right images. Then, a matching process is carried out to establish correspondences between the features extracted on the images and the map reconstructed (point cloud) by the system until that moment. To find these correspondences, the map points are projected on each image plane and the associated feature point is searched inside a local neighborhood of the projection. A 3D-2D match is established if the distance between the computed feature descriptor and the descriptor stored in the projected map point is bellow a certain threshold. The set of 3D-2D matches obtained are used to refine the current camera pose through the minimization of the reprojection errors. Once the current camera pose is estimated, it is needed to determine if the camera is moving away from the reconstructed scene, and if so new map points should be created to make it possible the tracking of the next incoming frames. The frames that are used to extend the map are called *keyframes* and are sent to the mapping thread to perform the refinement of the map. The map refinement is performed by means of a *Bundle Adjustment*, which adjust the keyframes poses and the 3D position of the map points.

For S-PTAM to run in real-time, several optimizations were developed along the whole system. In particular, we focus on speeding up the tracking thread which is in charge of estimating the camera pose for each incoming stereo frame. For example, the feature extraction performed on the left and the right images is carried out in two parallel threads. On the other hand, in order to get all the possible map points that are highly probable to be observed from the current pose, a covisibility graph [49] was used to determine the local map around the current pose. In this way, it is possible to achieve constant time performance in maps that contain millions of points.

To increase the robustness and accuracy of the whole system, not only stereo measurements were used but also left and right monocular measurements were incorporated in the optimization procedures present in the tracking and the mapping thread. The use of both types of measurements was already demonstrated to improve the behavior of a stereo based EKF-SLAM [96].

Following the line of increasing the robustness of the tracking task, we take advantage of the stereo camera to triangulate new map points in the tracking thread, instead of doing it in the mapping thread, as most SLAM systems do. The creation of new map points in the tracking thread allows to expand the map immediately without having to wait for the local mapping thread to be able to handle it. For instance, in situations where a new scene is being visited and the mapping thread is performing a Bundle Adjustment optimization, the localization can get lost because the keyframes are not processed and the map is not extended with new points.

The Loop Closing module is divided into three stages: detection of the previously visited places (loops), computation of the accumulated drift error and correction of the detected loops. The loop detection stage consists of analyzing the similarity of the keyframes stored along the computed trajectory by S-PTAM, in terms of appearance. At a first stage, the similarity analysis is permissive, in a way that all possible loops are considered. As a second stage, to perform a proper detection, loop candidates are validated only if the concerned keyframes present a high level of similarity and are geometrically consistent. To do this, the validation process estimates the relative transformation between the two camera poses that close a loop. Once a loop is validated, a pose-graph is generated to perform the correction of the trajectory. To carry out the loop correction, firstly, an initial correction is performed by the distribution of the accumulated error along all the keyframes in the loop. Then, this initial correction is used as a seed to start the pose-graph optimization process. It is important to remark that the implementation of the Loop Closing module was designed in order to work in a full parallel way, in order not to harm the performance of the whole system.

On the other hand, the Thesis covers the assessment of the impact of image feature extractors on the performance of S-PTAM in terms of pose accuracy and computational requirements. From this evaluation, we conclude that the GFTT [10] keypoint detector and BRISK descriptor [15] combination gives a good trade-off between computational demands and accuracy for real-time applications. Although the evaluation was performed using the S-PTAM system, the obtained results can be extrapolated to other stereo feature-based SLAM systems.

The accuracy of the method was tested in public outdoor and indoor datasets, comparing results against the provided ground truth. The used datasets present different conditions, such as dynamic objects, changing light conditions and fast camera motions combined with low camera frame-rate. Furthermore, experiments were performed with simulated time to test the real-time performance of the system. Results indicate that the accuracy of S-PTAM is comparable to state-of the art approaches for mobile robot localization.

It is important to mention that at the beginning of this work there was a gap on stereo SLAM software. For this reason, the implementation of S-PTAM was released as open-source and is publicly available in `http://github.com/lrse/sptam`. In recent papers from different research groups, S-PTAM has been considered as part of the state-of-the-art in stereo SLAM methods and has been compared against other systems [97, 98]. Furthermore, in a recent paper from our group S-PTAM was extended to consider additional information from an inertial measurement unit (IMU) to perform self-localization on a hexapod crawling platform operating on rough terrains [99].

6.1 Future work

As future work we detect several improvements that would be interesting to implement or to perform research upon. For instance, the incorporation of uncertainty information on the estimated pose and on the map could be of great interest for tasks where measurement certainty is crucial. Furthermore, the uncertainty information could be used to fuse the pose estimated by S-PTAM with other localization systems or sensors, for example by means of an EKF filter. Moreover, the uncertainties of the pose and of the map points can be used to replace the proposed matching grid. This is, the matching neighborhood area could be determined by the uncertainty ellipse resulting from the projection of the map point uncertainties from poses that have also their own uncertainties.

To increase the performance of S-PTAM, instead of undistorting both images completely, it is possible to use a projection camera model that includes the distortion parameters, in this way it is enough to undistort the extracted features.

Since S-PTAM only estimates the localization of the camera from the tracking of the reconstructed 3D map, there are features in the images that do not have an associated map point and therefore they are not used during the pose refinement. This untracked features could be tracked frame to frame to add 2D-2D constraints to the pose optimization. Moreover, the tracking of image features along several views could be used also to perform a triangulation of map points with a longer parallax than the one obtained from the stereo.

Given that keyframes are added with significant overlap to keep a good tracking of the camera, it would be desirable to remove redundant keyframes after the camera localization has succeeded [100]. In this way, it is possible to save space and reduce the number of constraints in the graph without causing harm to the performance of the system.

To increase the robustness of S-PTAM, the inverse-depth parametrization [32] could be used for the representation of low disparity map points (those points that are far away from the camera) and the stereo system could be used for the triangulation of the closest points. The combination of stereo triangulation for the closest points and inverse-depth parametrization for far points was already reported for a stereo EKF-SLAM system in [33].

Another useful extension that could be implemented is a *relocalization* module, capable of recovering the camera localization in situations where the tracking module fails. For example, this could be used in low texture environments or during fast camera movements.

The new Direct SLAM methods seen to be the future of Visual SLAM, given that they use all the pixels of the images [56, 57, 58, 59], as opposed to feature-based SLAM methods that use only high gradient points in the images. However, feature-based SLAM systems have shown to be more accurate and robust in [50]. This unexpected result could be related to the error model used during camera pose optimization for each SLAM paradigm. Therefore, a comparison between the *photometric error* used by direct-based SLAM systems and the *reprojection error* used by feature-based SLAM systems can help to establish which of both paradigms is the most suitable to solve the SLAM problem. In any case feature-based SLAM systems will still remain relevant in the next years due to their mature and proven accuracy and robustness.

Capítulo 6

Conclusiones

En esta Tesis, se presenta un sistema de SLAM estéreo llamado S-PTAM (por las siglás en inglés de *Stereo Parallel Tracking and Mapping*) para la localización de un robot móvil. S-PTAM construye incrementalmente un mapa esparzo del entorno mediante el uso de una cámara estéreo, y realiza un seguimiento (*tracking*) de la cámara dentro del mismo. Para permitir que S-PTAM funcione en tiempo real en entornos de gran escala, el problema de SLAM es fuertemente paralelizado, separando las tareas de seguimiento (*tracking*) y mapeo (*mapping*) en dos hilos de ejecución, reduciendo al mínimo la dependencia entre dichos hilos. Por otra parte, para permitir una localización global en mapas de gran escala, se desarrolló un módulo de detección y cierre de ciclos (*Loop Closure*) capaz de detectar lugares visitados con anterioridad y llevar a cabo una reducción del error acumulado hasta el momento. El módulo de Loop Closure fue diseñado de maneral tal de no interrumpir los hilos de tracking y mapping, haciendo que el sistema opere en tiempo real en todo momento.

Para cada par de imágenes estéreo (*stereo frame*), S-PTAM extrae las características visuales de la imagen izquierda y derecha. A continuación, se lleva acabo un proceso de establecimiento de correspondencias entre las características visuales (*features*) extraídas en las imágenes y los puntos del mapa reconstruido por el sistema hasta ese momento. Para encontrar dichas correspondencias, los puntos del mapa son proyectados en los planos de las imágenes, para luego realizar la búsqueda de la característica correspondiente dentro de la vecindad de la proyección. Se establece una correspondencia 3D-2D, si la distancia entre el descriptor de la característica encontrada en la imagen y el descriptor del punto del mapa se encuentra por debajo de cierto umbral. El conjunto de correspondencias 3D-2D obtenidas se utilizan para refinar la pose de la cámara actual por medio de la minimización de los errores de reproyección. Una vez estimada la pose de la cámara actual, es necesario determinar si la cámara se encuentra alejándose de la escena reconstruida, y si es así, nuevos puntos del mapa deben ser generados para hacer posible el seguimiento de los subsiguientes frames. Los frames que son utilizados para extender el mapa se denominan *keyframes*, y luego de que el mapa es expandido estos junto con los puntos triangulados son enviados almacenados en el mapa. El mapa esta representado como un grafo donde los nodos son los puntos y poses de los keyframes; las aristas son las mediciones entre ambos. Una vez que el grafo es actualizado, el hilo de mapeo refina el subgrafo determinado por los últimos keyframes almacenados por S-PTAM. El refinamiento del mapa se realiza por medio de técnicas de *Bundle Adjustment*, que ajustan las poses de los keyframes almacenados y la posición de los puntos del mapa de manera de reducir el error de reproyección.

Para permitir que S-PTAM trabaje en tiempo real, varias optimizaciones se desarrollaron a lo largo de todo el sistema. En particular, nos centramos en aumentar la velocidad del hilo de seguimiento, encargado de estimar la pose de la cámara para cada par de imágenes entrante. Por ejemplo, la extracción de características visuales realizada en las imágenes izquierda y derecha se lleva a cabo en dos hilos ejecución paralelos. Además, durante la etapa de búsqueda de correspondencias, para obtener de forma eficiente los puntos del mapa que son altamente probables de ser observados por la cámara, un grafo de covisibility basado en el propuesto en [49] es utilizado para obtener el mapa local que rodea la pose actual. De esta manera es posible alcanzar un desempeño de tiempo constante independientemente del crecimiento del mapa.

Por otro lado, nos aprovechamos de la cámara estéreo para triangular nuevos puntos del mapa en el hilo de seguimiento, en vez de triangular los puntos en el hilo de mapeo como la mayoría de los sistemas SLAM. La creación de puntos del mapa en el hilo de seguimiento permite extender el mapa inmediatamente sin tener que esperar a que el hilo de mapeo pueda hacerlo. Por ejemplo, en situaciones en las que el hilo de mapeo está realizando un refinamiento del mapa, y el keyframe no es procesado inmediatamente, la localización de la cámara puede perderse debido a que el mapa no se ha extendido todavía.

Para aumentar la robustez y la precisión de todo el sistema en general, además de utilizar las mediciones estéreo en los procesos de optimización de los hilos de seguimiento y mapeo, también se utilizan las mediciones monoculares izquierdas y derechas de cada cámara. El uso de ambos tipos de mediciones ha demostrado mejorar las precisión de los sistemas SLAM que si se utilizan solo las mediciones estéreo [96].

El módulo de Loop Closure se divide en tres etapas: detección de los lugares previamente visitados (loops), cálculo del error de deriva acumulada y corrección del ciclo detectado. La etapa de detección de ciclo consiste en analizar la similitud en términos de apariencia entre los keyframes almacenados a lo largo de la trayectoria estimada por S-PTAM. En una primera etapa, el análisis de apariencia es permisivo de manera que se consideran todos los ciclos posibles. Como segundo paso, para llevar a cabo una detección adecuada, los candidatos a ciclo son validados sólo si se encuentra una transformación relativa que este soportada por la mayoría de las correspondencias entre los dos keyframes de cierre. Una vez que se valida un ciclo, se genera una grafo de poses (*pose-graph*) para llevar a cabo el proceso de corrección de la trayectoria. En primer lugar, se lleva a cabo una corrección inicial mediante la propagación del error acumulado a lo largo de todos los keyframes que componen el ciclo. Luego, dicha corrección inicial es utilizada como semilla para el método de optimización del grafo de poses. Es importante remarcar que la implementación del modulo de Loop Closure es capaz de funcionar paralelamente al resto de los hilos de ejecución de manera de no perjudicar el desempeño de todo el sistema.

Por otro lado, la Tesis cubre la evaluación del impacto de extractores de características visuales en el rendimiento de S-PTAM en términos de precisión y requerimiento computacional. A partir de esta evaluación, se concluye que la combinación del detector de *keypoints* GFTT [10] y el descriptor binario BRISK [15] presenta un buen compromiso entre costo computacional y precisión para aplicaciones de SLAM en tiempo real. A pesar de que la evaluación se ha realizado sobre el sistema S-PTAM, los resultados obtenidos pueden extrapolarse a otros sistemas SLAM basados en visión estéreo.

La precisión, robustez y requerimiento computacional del método fue probada en datasets públicos tanto de ambientes interiores como exteriores. Los datasets utilizados presentan diferentes condiciones, tales como objetos dinámicos, cambios bruscos de iluminación y movimientos rápido de cámara combinados con una velocidad de muestreo (*frame-rate*) baja de la cámara. Además, los experimentos se realizaron con tiempo simulado para probar el rendimiento en tiempo real del sistema. Los resultados indican que la precisión de S-PTAM es comparable con los enfoques SLAM presentes en el estado del arte.

Es importante mencionar que al principio de este trabajo no se encontraron disponibles softwares de SLAM estéreo. Por esta razón, la implementación de S-PTAM fue concebida como software open-source y está disponible públicamente en `http://github.com/lrse/sptam`. En trabajos recientes de diferentes grupos de investigación, S-PTAM ha sido considerado como parte del estado del arte de métodos de SLAM estéreo y comparado con otros sistemas [97, 98]. Además, en un artículo reciente de nuestro grupo, S-PTAM se ha extendido para considerar la información adicional provista por una unidad inercial (IMU, por el acrónimo de *Inertial Measurement Unit*) para realizar la auto-localización de un hexápodo que opere en terrenos irregulares [99].

6.1 Trabajo futuro

Como trabajo futuro detectamos varias mejoras que resultan interesantes de ser investigadas. Por ejemplo, la incorporación de información de incertidumbre (covarianza) en el resultado de la pose y el mapa provisto por S-PTAM podría ser de gran interés en tareas donde el margen de error es determinante. Además, la información de incertidumbre podría ser utilizada para obtener una mejor localización mediante fusión de la pose estima por S-PTAM con la pose estimada por otro sistema de localización o con mediciones provistas por otros sensores, por ejemplo, por medio de un filtro EKF. Por otra parte, las incertidumbres de la pose y de los puntos del mapa pueden ser utilizadas para reemplazar la grilla de búsqueda durante la etapa de *matching*. Esto es, la vecindad de búsqueda de las correspondencias en la imagen queda determinada por la covarianza (representada como una elipse) que resulta del computo entre la covarianza del punto del mapa y la covarianza de la pose de la cámara.

Para aumentar el rendimiento de S-PTAM, en lugar de desdistorsionar toda la imagen, es posible utilizar un modelo de cámara que incluya los parámetros de distorsión. De esta manera únicamente es necesario desdistorcionar las características extraídas.

Dado que S-PTAM únicamente estima la localización de la cámara mediante trackeo del mapa reconstruido, habrá características en las imágenes que no tienen un punto del mapa asociado, y por lo tanto no son utilizadas durante el refinamiento pose. Estas características podrían ser seguidas entre frame y frame para agregar restricciones 2D-2D a la optimización de la pose de la cámara. Además, el seguimiento de características a nivel imagen a lo largo de varias vistas podría ser utilizado también para llevar a cabo una triangulación de puntos del mapa con un mayor paralaje, que el obtenido a partir del sistema estéreo.

Dado que los keyframes se añaden con un solapamiento significativo para mantener un buen seguimiento de la cámara, sería deseable eliminar aquellos keyframes redundantes después de que se haya localizado la cámara con éxito [100]. De esta manera, es posible ahorrar espacio en memoria y reducir el número de restricciones en el grafo mantenido por S-PTAM sin dañar el rendimiento del sistema.

Para aumentar la robustez de S-PTAM en entornos de grandes dimensiones, se podría utilizar la parametrización de profundidad inversa (*inverse-depth parametrization*) [32] para la representación de los puntos del mapa de disparidad baja (aquellos puntos que están muy lejos de la cámara) y utilizar el sistema estéreo para la triangulación de los puntos más cercanos. La combinación de triangulación estéreo para puntos cercanos y la parametrización de profundidad inversa para puntos lejanos se ha reportado para un sistema EKF-SLAM en [33].

Otra extensión útil que podría implementarse fácilmente es un módulo de relocalización capaz de recuperar la localización de la cámara en situaciones en que el módulo de seguimiento falle, por ejemplo, en entornos con poca texturas o durante movimientos rápidos de la cámara.

Finalmente, los nuevos métodos de SLAM directos parecen ser el futuro del SLAM visual dado que pueden utilizar todos los píxeles de las imágenes [56, 57, 58, 59], en contraste con los métodos basados en features que están limitados a utilizar solo los puntos de alto gradiente de las imágenes. Sin embargo, los sistemas basados en características SLAM (incluido el método aquí propuesto) han mostrado ser más precisos y robustos. Este inesperado resultado podría estar relacionado con el modelo de error utilizado durante la optimización de la pose de la cámara por cada tipo de paradigma. Por lo tanto, una comparación entre el error fotométrico utilizado por los sistemas de SLAM directos y el error de reproyección utilizado por los sistemas SLAM basados en características puede ayudar a establecer cuál de los dos paradigmas es el más adecuado para resolver el problema SLAM. En cualquier caso, los sistemas SLAM basados en características seguirán siendo relevantes en los próximos años debido a su madura -y probada- precisión y robustez.

Appendix A

Jacobians

In this appendix we presents the jacobians involved in the Bundle Adjustment used during the tracking and mapping phases. The contents of this appendix are mostly based on H. Strasdat's PhD Thesis [80].

A.1 Projections and camera models

Let $\mathbf{a} \in \mathbb{R}^n$, the Jacobian of the projection function is given by

$$\frac{\partial \operatorname{proj}(\mathbf{a})}{\partial \mathbf{a}} = \frac{1}{a_n} \begin{bmatrix} \mathbf{I}_{n \times n} & -\frac{1}{a_n} \begin{bmatrix} a_1 \\ \vdots \\ a_{n-1} \end{bmatrix} \end{bmatrix} .$$

The Jacobian of the monocular model

$$\hat{\mathbf{z}}_{\mathrm{m}}(\mathbf{x}^{\mathrm{c}}) = f \operatorname{proj}(\mathbf{x}^{\mathrm{c}}) + \mathbf{c},$$

is

$$\frac{\partial \hat{\mathbf{z}}_{\mathrm{m}}(\mathbf{x}^{\mathrm{c}})}{\partial \mathbf{x}^{\mathrm{c}}} = f \frac{\operatorname{proj}(\mathbf{x}^{\mathrm{c}})}{\partial \mathbf{x}^{\mathrm{c}}} = \frac{f}{z} \begin{bmatrix} 1 & 0 & -\frac{x}{z} \\ 0 & 1 & -\frac{y}{z} \end{bmatrix} .$$

The Jacobian of the stereo model

$$\hat{\mathbf{z}}_{\mathrm{s}}(\mathbf{x}^{\mathrm{c}}) = \begin{bmatrix} f \operatorname{proj}(\mathbf{x}^{\mathrm{c}}) + \mathbf{c} \\ f \frac{x-b}{z} + c_u \end{bmatrix},$$

is

$$\frac{\partial \hat{\mathbf{z}}_{\mathrm{s}}(\mathbf{x}^{\mathrm{c}})}{\partial \mathbf{x}^{\mathrm{c}}} = \frac{f}{z} \begin{bmatrix} 1 & 0 & -\frac{x}{z} \\ 0 & 1 & -\frac{y}{z} \\ 1 & 0 & -\frac{x-b}{z} \end{bmatrix},$$

where b is the baseline.

A.2 Pose-Point transformations

The derivative of $\mathrm{proj}(\dot{\mathbf{x}}^{\mathrm{c}})$ with respect to $\dot{\mathbf{x}}^{\mathrm{c}}$ is

$$\left.\frac{\partial\,\mathrm{proj}(\mathbf{q})}{\partial\mathbf{q}}\right|_{\mathbf{q}=\dot{\mathbf{x}}^{\mathrm{c}}} = \begin{bmatrix}\mathbf{I}_{3\times3} & -\mathbf{x}^{\mathrm{c}}\end{bmatrix},$$

since $q_4 = 1$.

The point Jacobian is given by

$$\frac{\partial\,\mathrm{proj}(\mathbf{E}\dot{\mathbf{x}}^{\mathrm{w}})}{\partial\mathbf{x}^{\mathrm{w}}} = \frac{\partial(\mathbf{R}\mathbf{x}^{\mathrm{w}}+\mathbf{t})}{\partial\mathbf{x}^{\mathrm{w}}} = \mathbf{R}.$$

As we discussed in section 3.6.1, The Jacobian respect to a pose $\mathbf{E}$ in **SE**(3) is computed using the smooth paths $\mathbf{E}_k\,(t) = \exp\,(t\mathbf{e}_k)\,\mathbf{E}$ through $\mathbf{E}$. The partial derivative of the transformation $\mathrm{proj}(\mathbf{E}\dot{\mathbf{x}}^{\mathrm{w}})$ with respecto $\mathbf{E}$ is

$$\begin{aligned}
\left.\frac{\partial\,\mathrm{proj}(\exp(\hat{\boldsymbol{\epsilon}}_{\mathfrak{se}(3)})\mathbf{E}\dot{\mathbf{x}}^{\mathrm{w}})}{\partial\epsilon_k}\right|_{\boldsymbol{\epsilon}=\mathbf{0}} &= \left.\frac{\partial\,\mathrm{proj}(\exp(\hat{\boldsymbol{\epsilon}}_{\mathfrak{se}(3)})\dot{\mathbf{x}}^{\mathrm{c}})}{\partial\epsilon_k}\right|_{\substack{\boldsymbol{\epsilon}=\mathbf{0}\\ \mathbf{x}^{\mathrm{c}}=\mathbf{R}\mathbf{x}^{\mathrm{w}}+\mathbf{t}}} \\
&= \left.\frac{\partial\,\mathrm{proj}(\mathbf{q})}{\partial\mathbf{q}}\right|_{\mathbf{q}=\dot{\mathbf{x}}^{\mathrm{c}}} \left.\frac{\partial\exp(\hat{\boldsymbol{\epsilon}}_{\mathfrak{se}(3)})\dot{\mathbf{x}}^{\mathrm{c}}}{\partial\epsilon_k}\right|_{\boldsymbol{\epsilon}=\mathbf{0}} \\
&= \begin{bmatrix}\mathbf{I}_{3\times3} & -\mathbf{x}^{\mathrm{c}}\end{bmatrix}\mathbf{G}_k\dot{\mathbf{x}}^{\mathrm{c}},
\end{aligned}$$

with $\mathbf{G}_k$ being the k-th generator of **SE**(3). For **SE**(3), the last row of each generator $\mathbf{G}_k$ consists of zeros only (see section 3.6). Thus, the last entry of the vector is $\mathbf{G}_i\dot{\mathbf{x}}^{\mathrm{c}}$ zero too and therefore

$$\left.\frac{\partial\,\mathrm{proj}(\exp(\hat{\boldsymbol{\epsilon}}_{\mathfrak{se}(3)})\dot{\mathbf{x}}^{\mathrm{c}})}{\partial\boldsymbol{\epsilon}_k}\right|_{\boldsymbol{\epsilon}=\mathbf{0}} = \begin{bmatrix}\mathbf{I}_{3\times3} & -\mathbf{x}^{\mathrm{c}}\end{bmatrix}\mathbf{G}_k\dot{\mathbf{x}}^{\mathrm{c}} = \begin{bmatrix}\mathbf{I}_{3\times3} & \mathbf{0}\end{bmatrix}\mathbf{G}_k\dot{\mathbf{x}}^{\mathrm{c}}.$$

Thus, the pose Jacobian for **SE**(3) is

$$\begin{aligned}
\left.\frac{\partial\,\mathrm{proj}(\exp(\hat{\boldsymbol{\epsilon}}_{\mathfrak{se}(3)})\mathbf{E}\dot{\mathbf{x}}^{\mathrm{w}})}{\partial\epsilon_k}\right|_{\boldsymbol{\epsilon}=\mathbf{0}} &= \begin{bmatrix}\mathbf{I}_{3\times3} & -\mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{G}_1\dot{\mathbf{x}}^{\mathrm{c}} & \mathbf{G}_2\dot{\mathbf{x}}^{\mathrm{c}} & \mathbf{G}_3\dot{\mathbf{x}}^{\mathrm{c}} & \mathbf{G}_4\dot{\mathbf{x}}^{\mathrm{c}} & \mathbf{G}_5\dot{\mathbf{x}}^{\mathrm{c}} & \mathbf{G}_6\dot{\mathbf{x}}^{\mathrm{c}}\end{bmatrix} \\
&= \begin{bmatrix}\mathbf{I}_{3\times3} & -[\mathbf{x}^{\mathrm{c}}]_\times\end{bmatrix} \qquad \text{with} \quad \mathbf{x}^{\mathrm{c}} = \mathbf{R}\mathbf{x}^{\mathrm{w}}+\mathbf{t}.
\end{aligned}$$

A.3 Bundle Adjustment

The jacobians involved in Bundle Adjustment for a monocular camera are:

$$\begin{aligned}
\frac{\partial(\mathbf{z}-\hat{\mathbf{z}}_{\mathrm{m}}(\mathrm{proj}(\mathbf{E}\dot{\mathbf{x}}^{\mathrm{w}})))}{\partial\mathbf{x}^{\mathrm{w}}} &= \left.\frac{\partial\hat{\mathbf{z}}_{\mathrm{m}}(\mathbf{x}^{\mathrm{c}})}{\partial\mathbf{x}^{\mathrm{c}}}\right|_{\mathbf{x}^{\mathrm{c}}=\mathbf{R}\mathbf{x}^{\mathrm{w}}+\mathbf{t}} \frac{\partial(\mathbf{R}\mathbf{x}^{\mathrm{w}}+\mathbf{t})}{\partial\mathbf{x}^{\mathrm{w}}} \\
&= -\frac{f}{z}\begin{bmatrix}1 & 0 & -\frac{x}{z}\\ 0 & 1 & -\frac{y}{z}\end{bmatrix}\mathbf{R}
\end{aligned}$$

and

$$\left.\frac{\partial(\mathbf{z}-\hat{\mathbf{z}}_{\mathrm{m}}(\mathrm{proj}(\exp(\hat{\boldsymbol{\epsilon}}_{\mathfrak{se}(3)})\mathbf{E}\dot{\mathbf{x}}^{\mathrm{w}})))}{\partial\boldsymbol{\epsilon}}\right|_{\boldsymbol{\epsilon}=\mathbf{0}} = \left.\frac{\partial\hat{\mathbf{z}}_{\mathrm{m}}(\mathbf{x}^{\mathrm{c}})}{\partial\mathbf{x}^{\mathrm{c}}}\right|_{\mathbf{x}^{\mathrm{c}}=\mathbf{R}\mathbf{x}^{\mathrm{w}}+\mathbf{t}} \left.\frac{\partial\,\mathrm{proj}(\exp(\hat{\boldsymbol{\epsilon}})\dot{\mathbf{x}}^{\mathrm{c}})}{\partial\boldsymbol{\epsilon}}\right|_{\boldsymbol{\epsilon}=\mathbf{0}}$$
$$= -\frac{f}{z}\begin{bmatrix}1 & 0 & -\frac{x}{z}\\ 0 & 1 & -\frac{y}{z}\end{bmatrix}\begin{bmatrix}\mathbf{I}_{3\times3} & -[\mathbf{x}^{\mathrm{c}}]_{\times}\end{bmatrix}.$$

In a similar way, the Jacobians for a stereo camera have the following forms

$$\frac{\partial(\mathbf{z}-\hat{\mathbf{z}}_{\mathrm{s}}(\mathrm{proj}(\mathbf{E}\dot{\mathbf{x}}^{\mathrm{w}})))}{\partial\mathbf{x}^{\mathrm{w}}} = \left.\frac{\partial\hat{\mathbf{z}}_{\mathrm{s}}(\mathbf{x}^{\mathrm{c}})}{\partial\mathbf{x}^{\mathrm{c}}}\right|_{\mathbf{x}^{\mathrm{c}}=\mathbf{R}\mathbf{x}^{\mathrm{w}}+\mathbf{t}} \frac{\partial(\mathbf{R}\mathbf{x}^{\mathrm{w}}+\mathbf{t})}{\partial\mathbf{x}^{\mathrm{w}}}$$
$$= -\frac{f}{z}\begin{bmatrix}1 & 0 & -\frac{x}{z}\\ 0 & 1 & -\frac{y}{z}\\ 1 & 0 & -\frac{x-b}{z}\end{bmatrix}\mathbf{R},$$

and

$$\left.\frac{\partial(\mathbf{z}-\hat{\mathbf{z}}_{\mathrm{s}}(\mathrm{proj}(\exp(\hat{\boldsymbol{\epsilon}}_{\mathfrak{se}(3)})\mathbf{E}\dot{\mathbf{x}}^{\mathrm{w}})))}{\partial\boldsymbol{\epsilon}}\right|_{\boldsymbol{\epsilon}=\mathbf{0}} = \left.\frac{\partial\hat{\mathbf{z}}_{\mathrm{s}}(\mathbf{x}^{\mathrm{c}})}{\partial\mathbf{x}^{\mathrm{c}}}\right|_{\mathbf{x}^{\mathrm{c}}=\mathbf{R}\mathbf{x}^{\mathrm{w}}+\mathbf{t}} \left.\frac{\partial\,\mathrm{proj}(\exp(\hat{\boldsymbol{\epsilon}})\dot{\mathbf{x}}^{\mathrm{c}})}{\partial\boldsymbol{\epsilon}}\right|_{\boldsymbol{\epsilon}=\mathbf{0}}$$
$$= -\frac{f}{z}\begin{bmatrix}1 & 0 & -\frac{x}{z}\\ 0 & 1 & -\frac{y}{z}\\ 1 & 0 & -\frac{x-b}{z}\end{bmatrix}\begin{bmatrix}\mathbf{I}_{3\times3} & -[\mathbf{x}^{\mathrm{c}}]_{\times}\end{bmatrix}.$$

For a more detailed explanation of jacobians derivation the reader is referred to [80, 101].

Bibliography

[1] J. A. Castellanos, J. M. M. Montiel, J. Neira, and J. D. Tardos, "The SPmap: a probabilistic framework for simultaneous localizationand map building," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 5, pp. 948–952, October 1999. 1

[2] 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*, vol. 29, no. 6, pp. 1052–1067, June 2007. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2007.1049 1, 2.1

[3] C. Kerl, J. Sturm, and D. Cremers, "Dense visual SLAM for RGB-D cameras," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, November 2013, pp. 2100–2106. 1

[4] A. Concha, G. Loianno, V. Kumar, and J. Civera, "Visual-inertial direct SLAM," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 1331–1338. 1

[5] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, March 2015. [Online]. Available: http://dx.doi.org/10.1177/0278364914554813 1

[6] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison, "Elasticfusion: Dense slam without a pose graph," *Proceedings of Robotics: Science and Systems (RSS)*, July 2015. 1

[7] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004. 1, 2.1, 3.2.1, 5.3

[8] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Proceedings of the European Conference on Computer Vision (ECCV)*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, vol. 3951, pp. 404–417. [Online]. Available: http://dx.doi.org/10.1007/11744023_32 1, 3.2.1, 5.3

[9] M. Agrawal, K. Konolige, and M. Blas, "Censure: Center surround extremas for realtime feature detection and matching," in *Proceedings of the European Conference on Computer Vision (ECCV)*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008, vol. 5305, pp. 102–115. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-88693-8_8 1, 3.2.1, 5.3

[10] J. Shi and C. Tomasi, "Good features to track," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 1994, pp. 593–600. 1, 3.2.1, 4.2, 5.3, 6

[11] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Proceedings of the European Conference on Computer Vision (ECCV)*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, vol. 3951, pp. 430–443. [Online]. Available: http://dx.doi.org/10.1007/11744023_34 1, 3.2.1, 5.3

[12] E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger, "Adaptive and generic corner detection based on the accelerated segment test," in *Proceedings of the ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part II*, K. Daniilidis, P. Maragos, and N. Paragios, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, September 2010, pp. 183–196. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-15552-9_14 1, 3.2.1, 5.3

[13] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, November 2011, pp. 2564–2571. 1, 3.2.1, 5.3

[14] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *Proceedings of the European Conference on Computer Vision (ECCV)*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, vol. 6314, pp. 778–792. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-15561-1_56 1, 3.2.1, 5.3

[15] S. Leutenegger, M. Chli, and R. Y. Siegwart, "Brisk: Binary robust invariant scalable keypoints," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, ser. ICCV '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 2548–2555. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2011.6126542 1, 3.2.1, 4.2, 5.3, 6

[16] G. Levi and T. Hassner, "LATCH: Learned Arrangements of Three Patch Codes," *Computing Research Repository (CoRR)*, vol. abs/1501.03719, 2015. [Online]. Available: http://arxiv.org/abs/1501.03719 1, 3.2.1, 5.3

[17] G. Klein and D. Murray, "Parallel Tracking and Mapping for Small AR Workspaces," in *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 1–10. [Online]. Available: http://dx.doi.org/10.1109/ISMAR.2007.4538852 1, 2.2, 4.5.4, 4.8

[18] G. I. Castro, "Detección y cierre de ciclos en sistemas SLAM basados en visión estéreo," Bachelor Thesis, University of Buenos Aires, Ciudad Autónoma de Buenos Aires, Argentina, April 2016, supervised by: Taihú Pire and Pablo De Cristóforis. 1.1

[19] A. H. Jazwinski, *Stochastic processes and filtering theory*, ser. Mathematics in science and engineering. New York: Academic press, 1970, uKM. [Online]. Available: http://opac.inria.fr/record=b1078357 2

[20] N. Metropolis and S. Ulam, "The Monte Carlo Method," *Journal of the American Statistical Association*, vol. 44, no. 247, pp. 335–341, 1949, pMID: 18139350. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/01621459.1949.10483310 2

[21] J. Manyika and H. Durrant-Whyte, "Data Fusion and Sensor Management: An Information-Theoretic Approach," Prentice Hall, Upper Saddle River, 1994. 2

[22] D. Fox, W. Burgard, and S. Thrun, "Markov Localization for Mobile Robots in Dynamic Environments," *Journal of Artificial Intelligence Research*, vol. 11, pp. 391–427, July 1999. 2

[23] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1. IEEE, June 2004, pp. I–652–I–659. 2

[24] D. Nistér, "An efficient solution to the five-point relative pose problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 756–770, June 2004. 2

[25] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng, "Complete solution classification for the perspective-three-point problem," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 8, pp. 930–943, Aug. 2003. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2003.1217599 2, 3.4.4

[26] L. Kneip, D. Scaramuzza, and R. Siegwart, "A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2011, pp. 2969–2976. 2, 3.4.4, 4.7.5

[27] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, *Bundle Adjustment — A Modern Synthesis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, ch. 21, pp. 298–372. [Online]. Available: http://dx.doi.org/10.1007/3-540-44480-7_21 2

[28] D. Brown, "A solution to the general problem of multiple station analytical stereo triangulation," Patrick Airforce Base, Florida, Tech. Rep., 1958. 2

[29] A. J. Davison and D. W. Murray, "Simultaneous localization and map-building using active vision," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 7, pp. 865–880, 2002. 2.1

[30] A. Chiuso, P. Favaro, H. Jin, and S. Soatto, "Structure from motion causally integrated over time," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 4, pp. 523–535, April 2002. [Online]. Available: http://dx.doi.org/10.1109/34.993559 2.1

[31] S. Ullman, "The Interpretation of Structure from Motion," *Proceedings of the Royal Society of London B: Biological Sciences*, vol. 203, no. 1153, pp. 405–426, 1979. [Online]. Available: http://rspb.royalsocietypublishing.org/content/203/1153/405 2.1

[32] J. Civera, A. J. Davison, and J. M. M. Montiel, "Inverse Depth Parametrization for Monocular SLAM," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 932–945, October 2008. 2.1, 6.1

[33] L. M. Paz, P. Piniés, J. D. Tardós, and J. Neira, "Large-Scale 6-DOF SLAM With Stereo-in-Hand," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 946–957, October 2008. 2.1, 6.1

[34] S. J. Julier and J. K. Uhlmann, "A counter example to the theory of simultaneous localization and map building," in *Proceedings of the IEEE International Conference on Robotics and Automation (IROS)*, vol. 4, 2001, pp. 4238–4243. 2.1

[35] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem," in *Proceedings of the Eighteenth National Conference on Artificial Intelligence*. Menlo Park, CA, USA: American Association for Artificial Intelligence, 2002, pp. 593–598. [Online]. Available: http://dl.acm.org/citation.cfm?id=777092.777184 2.1

[36] M. Montemerlo, S. Thrun, D. Roller, and B. Wegbreit, "FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping That Provably Converges," in *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, ser. IJCAI'03. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003, pp. 1151–1156. [Online]. Available: http://dl.acm.org/citation.cfm?id=1630659.1630824 2.1

[37] N. Karlsson, E. Di Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian, and M. E. Munich, "The vSLAM Algorithm for Robust Localization and Mapping," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, April 2005, pp. 24–29. 2.1

[38] E. Eade and T. Drummond, "Scalable Monocular SLAM," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, June 2006, pp. 469–476. 2.1

[39] R. Sim, P. Elinas, and M. Griffin, "Vision-based slam using the rao-blackwellised particle filter," in *In IJCAI Workshop on Reasoning with Uncertainty in Robotics*, 2005, pp. 9–16. 2.1

[40] P. Elinas, R. Sim, and J. J. Little, "/spl sigma/slam: stereo vision slam using the rao-blackwellised particle filter and a novel mixture proposal distribution," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2006, pp. 1564–1570. 2.1

[41] R. Sim, P. Elinas, and J. J. Little, "A study of the rao-blackwellised particle filter for efficient and accurate vision-based slam," *International Journal of Computer Vision*, vol. 74, no. 3, pp. 303–318, 2007. 2.1

[42] F. Dellaert and M. Kaess, "Square Root SAM: Simultaneous Localization and Mapping via Square Root Information Smoothing," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, 2006. [Online]. Available: http://ijr.sagepub.com/content/25/12/1181.abstract 2.2, 4.8

[43] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd, "Real time localization and 3d reconstruction," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1. IEEE, 2006, pp. 363–370. 2.2

[44] B. Clipp, J. Lim, J. M. Frahm, and M. Pollefeys, "Parallel, real-time visual SLAM," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2010, pp. 3961–3968. 2.2

[45] S. A. Scherer, D. Dube, and A. Zell, "Using depth in visual simultaneous localisation and mapping," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2012, pp. 5216–5221. 2.2

[46] K. Konolige and M. Agrawal, "FrameSLAM: From Bundle Adjustment to Real-Time Visual Mapping," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1066–1077, October 2008. 2.2

[47] J. Lim, J. M. Frahm, and M. Pollefeys, "Online environment mapping," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2011, pp. 3489–3496. 2.2

[48] C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid, "Rslam: A system for large-scale mapping in constant-time using stereo," *International Journal of Computer Vision*, vol. 94, no. 2, pp. 198–214, 2011. [Online]. Available: http://dx.doi.org/10.1007/s11263-010-0361-7 2.2

[49] H. Strasdat, A. J. Davison, J. M. M. Montiel, and K. Konolige, "Double window optimisation for constant time visual SLAM," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, November 2011, pp. 2352–2359. 2.2, 6

[50] R. Mur-Artal and J. D. Tardós, "Probabilistic semi-dense mapping from highly accurate feature-based monocular SLAM," *Proceedings of Robotics: Science and Systems (RSS)*, July 2015. 2.2, 2.3, 6.1

[51] H. Strasdat, J. M. M. Montiel, and A. J. Davison, "Editors Choice Article: Visual SLAM: Why Filter?" *Image and Vision Computing*, vol. 30, no. 2, pp. 65–77, February 2012. [Online]. Available: http://dx.doi.org/10.1016/j.imavis.2012.02.009 2.2

[52] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras," *CoRR*, vol. abs/1610.06475, 2016. [Online]. Available: http://arxiv.org/abs/1610.06475 2.2, 4.5.4, 5.5.1

[53] D. Galvez-Lopez and J. D. Tardós, "Bags of Binary Words for Fast Place Recognition in Image Sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, October 2012. 2.2, 4.7.3, 4.7, 4.7.4, 4.8

[54] H. Strasdat, J. M. M. Montiel, and A. J. Davison, "Scale drift-aware large scale monocular slam," *Proceedings of Robotics: Science and Systems*, June 2010. 2.2

[55] T. Tykkälä and A. Comport, "A dense structure model for image based stereo slam," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2011, pp. 1758–1763. 2.3

[56] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense Tracking and Mapping in Real-time," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, ser. ICCV '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 2320–2327. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2011.6126513 2.3, 6.1

[57] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-Scale Direct Monocular SLAM," in *Proceedings of the European Conference on Computer Vision (ECCV)*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 834–849. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-10605-2_54 2.3, 6.1

[58] J. Engel, J. Stückler, and D. Cremers, "Large-scale direct slam with stereo cameras," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 1935–1942. [Online]. Available: http://dx.doi.org/10.1109/IROS.2015.7353631 2.3, 5.5.1, 6.1

[59] A. Concha and J. Civera, "DPPTAM: Dense piecewise planar tracking and mapping from a monocular sequence," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, September 2015, pp. 5686–5693. 2.3, 6.1

[60] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," in *arXiv:1607.02565*, July 2016. 2.3

[61] P. F. Alcantarilla, A. Bartoli, and A. J. Davison, "KAZE Features," in *Proceedings of the 12th European Conference on Computer Vision - Volume Part VI*, ser. Proceedings of the European Conference on Computer Vision (ECCV). Berlin, Heidelberg: Springer-Verlag, 2012, pp. 214–227. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-33783-3_16 3.2.1

[62] P. F. Alcantarilla, J. Nuevo, and A. Bartoli, "Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces," in *Proceedings of the British Machine Vision Conference (BMVC)*. BMVA Press, 2013. 3.2.1

[63] A. Alahi, R. Ortiz, and P. Vandergheynst, "FREAK: Fast Retina Keypoint," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2012, pp. 510–517. 3.2.1

[64] X. Yang and K.-T. Cheng, "LDB: An ultra-fast feature for scalable Augmented Reality on mobile devices," in *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, November 2012, pp. 49–57. 3.2.1

[65] A. Ziegler, E. Christiansen, D. Kriegman, and S. J. Belongie, "Locally uniform comparison image descriptor," in *Proceedings of the 25th Advances in Neural Information Processing Systems*, P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., 2012, pp. 1–9. [Online]. Available: http://books.nips.cc/papers/files/nips25/NIPS2012_0012.pdf 3.2.1

[66] T. Trzcinski, M. Christoudias, P. Fua, and V. Lepetit, "Boosting Binary Keypoint Descriptors," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013, pp. 2874–2881. 3.2.1

[67] J. Sivic and A. Zisserman, "Video google: a text retrieval approach to object matching in videos," in *Proceedings of the Ninth IEEE International Conference on Computer Vision*, vol. 2, October 2003, pp. 1470–1477. 3.2.2, 3.2.2

[68] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. New York, NY, USA: Cambridge University Press, 2003. 3.10, 3.11, 3.12, 3.13, 3.16

[69] P. J. Besl and H. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, February 1992. 3.4.4

[70] B. K. Horn, H. M. Hilden, and S. Negahdaripour, "Closed-form solution of absolute orientation using orthonormal matrices," *JOSA A*, vol. 5, no. 7, pp. 1127–1135, 1988. 3.4.4

[71] C. P. Lu, G. D. Hager, and E. Mjolsness, "Fast and globally convergent pose estimation from video images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 6, pp. 610–622, Jun 2000. 3.4.4

[72] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o(n) solution to the pnp problem," *International Journal of Computer Vision*, vol. 81, no. 2, pp. 155–166, 2008. [Online]. Available: http://dx.doi.org/10.1007/s11263-008-0152-6 3.4.4

[73] L. Kneip, H. Li, and Y. Seo, "Upnp: An optimal o(n) solution to the absolute pose problem with universal applicability," in *Proceedings of the European Conference on Computer Vision (ECCV)*, ser. Lecture Notes in Computer Science, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Springer International Publishing, 2014, vol. 8689, pp. 127–142. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-10590-1_9 3.4.4, 4.7.5

[74] J. Stalbaum and J. b. Song, "Keyframe and inlier selection for visual SLAM," in *10th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, October 2013, pp. 391–396. 4.5.4

[75] C. Pirchheim, D. Schmalstieg, and G. Reitmayr, "Handling pure camera rotation in keyframe-based SLAM," in *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, October 2013, pp. 229–238. 4.5.4

[76] K. Pirker, M. Rüther, and H. Bischof, "CD SLAM - continuous localization and mapping in a dynamic world," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Septeber 2011, pp. 3990–3997. 4.5.4

[77] R. Mur-Artal and J. D. Tardós, "Fast relocalisation and loop closing in keyframe-based slam," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 846–853. 4.7

[78] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981. [Online]. Available: http://doi.acm.org/10.1145/358669.358692 4.7.5

[79] K. Shoemake, "Animating Rotation with Quaternion Curves," *Special Interest Group on Computer GRAPHics and Interactive Techniques (SIGGRAPH)*, vol. 19, no. 3, pp. 245–254, July 1985. [Online]. Available: http://doi.acm.org/10.1145/325165.325242 4.7.6

[80] H. Strasdat, "Local Accuracy and Global Consistency for Efficient Visual SLAM," October 2012, phD Thesis. 4.7.6, A, A.3

[81] M. F. Fallon, H. Johannsson, M. Kaess, and J. J. Leonard, "The MIT Stata Center dataset," *The International Journal of Robotics Research, IJRR*, vol. 32, no. 14, pp. 1695–1699, December 2013. 4.8

[82] J.-L. Blanco, F.-A. Moreno, and J. González, "A collection of outdoor robotic datasets with centimeter-accuracy ground truth," *Autonomous Robots*, vol. 27, no. 4, pp. 327–351, November 2009. [Online]. Available: http://www.mrpt.org/Paper:Malaga_Dataset_2009 4.8

[83] L. Kneip and P. Furgale, "OpenGV: A unified and generalized approach to real-time calibrated geometric vision," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 1–8. 4.8

[84] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2011, pp. 3607–3613. 4.8

[85] N. Sünderhauf and P. Protzel, "Switchable constraints for robust pose graph SLAM," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2012, pp. 1879–1884. 4.8

[86] N. Sünderhauf, "Vertigo: Versatile Extensions for Robust Inference using Graph Optimization," http://www.openslam.org/vertigo. 4.8

[87] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009. 4.8

[88] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision Meets Robotics: The KITTI Dataset," *The International Journal of Robotics Research, IJRR*, vol. 32, no. 11, pp. 1231–1237, September 2013. [Online]. Available: http://dx.doi.org/10.1177/0278364913491297 5.1, 5.2, 5.3, 5.5.1

[89] L. Murphy, T. Morris, U. Fabrizi, M. Warren, M. Milford, B. Upcroft, M. Bosse, and P. Corke, "Experimental Comparison of Odometry Approaches," in *Experimental Robotics*, ser. Springer Tracts in Advanced Robotics, J. P. Desai, G. Dudek, O. Khatib, and V. Kumar, Eds. Springer International Publishing, 2013, vol. 88, pp. 877–890. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-00065-7_58 5.1

[90] R. Kümmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner, "On measuring the accuracy of SLAM algorithms," *Autonomous Robots*, vol. 27, no. 4, pp. 387–407, 2009. [Online]. Available: http://dx.doi.org/10.1007/s10514-009-9155-6 5.2

[91] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," in *Proceedings of the 1987 IEEE International Conference on Robotics and Automation*, vol. 4, March 1987, pp. 850–850. 5.2

[92] J. Heinly, E. Dunn, and J.-M. Frahm, "Comparative Evaluation of Binary Features," in *Proceedings of the ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part II*, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 759–773. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-33709-3_54 5.3

[93] A. Schmidt, M. Kraft, M. Fularz, and Z. Domagała, "Comparative assessment of point feature detectors in the context of robot navigation," *Journal of Automation Mobile Robotics and Intelligent Systems*, vol. 7, no. 1, pp. 11–20, 2013. 5.3

[94] R. Arroyo, P. F. Alcantarilla, L. M. Bergasa, J. J. Yebes, and S. Bronte, "Fast and effective visual place recognition using binary codes and disparity information," in *Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, September 2014, pp. 3089–3094. 5.4.1

[95] "The KITTI Vision Benchmark Suite." [Online]. Available: http://www.cvlibs.net/datasets/kitti/eval_odometry.php 5.5.1

[96] J. Sola, A. Monin, and M. Devy, "BiCamSLAM: Two times mono is more than stereo," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, April 2007, pp. 4795–4800. 6

[97] W. Ci and Y. Huang, "A Robust Method for Ego-Motion Estimation in Urban Environment Using Stereo Camera," *Sensors*, vol. 16, no. 10, p. 1704, 2016. [Online]. Available: http://www.mdpi.com/1424-8220/16/10/1704 6

[98] N. Krombach, D. Droeschel, and S. Behnke, "Combining Feature-based and Direct Methods for Semi-dense Real-time Stereo Visual Odometry," in *Proceedings of the 14th International Conference on Intelligent Autonomous Systems (IAS)*, July 2016. 6

[99] T. Fischer, T. Pire, P. Čížek, P. De Cristóforis, and J. Faigl, "Stereo Vision-based Localization for Hexapod Walking Robots Operating in Rough Terrains," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2016, to appear. 6

[100] Z. Dong, G. Zhang, J. Jia, and H. Bao, "Efficient Keyframe-based Real-time Camera Tracking," *Computer Vision and Image Understanding*, vol. 118, pp. 97–110, Jan. 2014. [Online]. Available: http://dx.doi.org/10.1016/j.cviu.2013.08.005 6.1

[101] J.-L. Blanco, "A tutorial on SE(3) transformation parameterizations and on-manifold optimization," University of Malaga, Tech. Rep., September 2010. A.3