TESIS DE LA UNIVERSIDAD DE ZARAGOZA

2017

27

Raúl Mur Artal

Real-Time Accurate Visual SLAM with Place Recognition

Departamento
Informática e Ingeniería de Sistemas

Director/es
Tardós Solano, Juan Domingo

ISSN 2254-7606

Prensas de la Universidad
Universidad Zaragoza

ISSN 2254-7606

Universidad Zaragoza
1542

Tesis Doctoral

REAL-TIME ACCURATE VISUAL SLAM WITH PLACE RECOGNITION

Autor

Raúl Mur Artal

Director/es

Tardós Solano, Juan Domingo

UNIVERSIDAD DE ZARAGOZA

Informática e Ingeniería de Sistemas

2017

Universidad Zaragoza

Ph.D Thesis

Real-Time Accurate Visual SLAM with Place Recognition

Raúl Mur Artal

Supervised by Juan D. Tardós Solano

Departamento de Informática e Ingeniería de Sistemas (DIIS)
Instituto de Investigación en Ingeniería de Aragón (I3A)
Escuela de Ingeniería y Arquitectura (EINA)

Universidad de Zaragoza

January 10, 2017

Real-Time Accurate Visual SLAM with Place Recognition

Raúl Mur Artal

Supervisor

Juan Domingo Tardós	Universidad de Zaragoza, Spain

Dissertation Committee

Giorgio Grisetti	Sapienza Università di Roma, Italy
Margarita Chli	ETH Zürich, Switzerland
Javier Civera	Universidad de Zaragoza, Spain
Javier González	Universidad de Málaga, Spain
José Neira	Universidad de Zaragoza, Spain

External Examiners

Michael Kaess	Carnegie Mellon University, USA
Stefan Leutenegger	Imperial College London, UK

Funding

Beca FPU (13/04175)	Ministerio de Educación, Spain
Beca DGA (B121/13)	Gobierno de Aragón, Spain
Proyecto DPI2012-32168	Dirección General de Investigación, Spain
Proyecto DPI2015-67275	Dirección General de Investigación, Spain
Fondo Social Europeo Grupo T04	Gobierno de Aragón, Spain

A Yovana e Inés

A mis padres y mi hermano

Acknowledgements

I want to firstly thank my supervisor Juan Domingo Tardós for his guidance and support through all these years, and for always providing his sincere opinion and comments. I can undoubtedly say that today I am a better researcher thanks to him. I want also to express my gratitude to José María Martínez Montiel who showed me for the first time a SLAM demo and motivated me to begin my PhD studies in this exciting field. Thanks also for being always available to discuss about research and see my latest results.

I want also to sincerely thank Andrew Davison for hosting me in his lab at Imperial College for a few months and for the kind words he always had to my work. I also thank all the guys at the Dyson and RobotVision labs for the warm welcome and making me feel as one of them during my visit. I wish all of you all the best and success in your careers.

My PhD life would have probably been more productive, but way less fun, without my lab colleagues. Thanks for making this journey more enjoyable and see you in the next tapas night.

Muchas gracias a mi familia y amigos. En especial, a mis padres y mi hermano por apoyarme siempre. A Yovana por estar a mi lado durante todo este tiempo, sobre todo en los momentos más estresantes, por su apoyo incondicional y su cariño. A Inés por su sonrisa cada mañana y cada vez que he vuelto a casa del laboratorio, nada me ha motivado más en estos años.

Resumen

El problema de localización y construcción simultánea de mapas (del inglés Simultaneous Localization and Mapping, abreviado SLAM) consiste en localizar un sensor en un mapa que se construye en línea. La tecnología de SLAM hace posible la localización de un robot en un entorno desconocido para él, procesando la información de sus sensores de a bordo y por tanto sin depender de infraestructuras externas. Un mapa permite localizarse en todo momento sin acumular deriva, a diferencia de una odometría donde se integran movimientos incrementales. Este tipo de tecnología es crítica para la navegación de robots de servicio y vehículos autónomos, o para la localización del usuario en aplicaciones de realidad aumentada o virtual.

La principal contribución de esta tesis es ORB-SLAM, un sistema de SLAM monocular basado en características que trabaja en tiempo real en ambientes pequeños y grandes, de interior y exterior. El sistema es robusto a elementos dinámicos en la escena, permite cerrar bucles y relocalizar la cámara incluso si el punto de vista ha cambiado significativamente, e incluye un método de inicialización completamente automático. ORB-SLAM es actualmente la solución más completa, precisa y fiable de SLAM monocular empleando una cámara como único sensor. El sistema, estando basado en características y ajuste de haces, ha demostrado una precisión y robustez sin precedentes en secuencias públicas estándar.

Adicionalmente se ha extendido ORB-SLAM para reconstruir el entorno de forma semi-densa. Nuestra solución desacopla la reconstrucción semi-densa de la estimación de la trayectoria de la cámara, lo que resulta en un sistema que combina la precisión y robustez del SLAM basado en características con las reconstrucciones más completas de los métodos directos. Además se ha extendido la solución monocular para aprovechar la información de cámaras estéreo, RGB-D y sensores inerciales, obteniendo precisiones superiores a otras soluciones del estado del arte. Con el fin de contribuir a la comunidad científica, hemos hecho libre el código de una implementación de nuestra solución de SLAM para cámaras monoculares, estéreo y RGB-D, siendo la primera solución de código libre capaz de funcionar con estos tres tipos de cámara.

Abstract

Simultaneous Localization and Mapping (SLAM) is the problem of localizing a sensor in a map that is built online. SLAM technology can enable robot localization in unknown environments by processing onboard sensors and therefore not relying on external infrastructure. A map allows to continually localize in the same environment without accumulating drift, in contrast to odometry approaches where incremental motion is integrated over time. Such technology is critical for the navigation of service robots and autonomous vehicles, or to localize a user in virtual or augmented reality applications.

The main contribution of this thesis is ORB-SLAM, a feature-based monocular SLAM system that operates in real time, in small and large, indoor and outdoor environments. The system is robust to severe motion clutter, allows wide baseline loop closing and relocalization, and includes full automatic initialization. ORB-SLAM is currently the most complete, accurate and reliable solution for SLAM using a monocular camera as single sensor. This system being based on features and bundle adjustment, has demonstrated unprecedented accuracy and robustness in standard public datasets.

A further contribution is an extension of ORB-SLAM to perform semi-dense reconstructions. We decouple the semi-dense reconstruction from the camera trajectory estimation, resulting in a system that combines the accuracy and robustness of feature-based SLAM with the more complete reconstruction of direct methods. We have also extended the original monocular solution to exploit the information from stereo, RGB-D and inertial sensors, achieving accuracy results beyond other state-of-the-art solutions. To the benefit of the scientific community, we have made open-source an implementation of our proposed SLAM solution for monocular, stereo and RGB-D cameras, which is the first-of-its-kind.

Contents

Chapter 1

Introduction

1.1 Visual Simultaneous Localization and Mapping

Imagine a robot that has to perform a task in an unknown environment, as a cleaning robot in a house or a surveillance drone in an installation. In order to succeed in the mission the robot needs to localize itself in this environment. There might exist an external infrastructure to localize the robot, as a motion capture or global positioning system (GPS). However these systems might be expensive, non-available or not as accurate as required. It is desirable that **localization** is performed, as animals and humans do, by processing information from onboard sensors.

The simplest method to localize a robot is to process the sensor information to compute incremental motion, which is called **odometry**. This allows to retrieve the trajectory of the robot, which will inevitably accumulate error making the estimated trajectory to **drift** from the real trajectory performed by the robot. Odometry techniques are suitable for short-term motion estimation, but for long-term operation in the same environment one would desire to have a **map** that allows **drift-free** localization. **Mapping** is the process of creating a map from onboard sensors given that localization is known. However in order to localize the robot we have introduced the necessity of having a map. This problem is known as **Simultaneous Localization and Mapping** (SLAM), where we aim to solve localization and mapping simultaneously. This problem involves several challenging subproblems:

- **Initialization**. As we aim to solve both mapping and localization at the same time, this is the problem of localizing the robot during the first moments when there is no map. Depending on the sensor this can be straightforward (e.g. stereo cameras, lasers) or very challenging (e.g. monocular cameras).

- **Loop closing**. SLAM and odometry techniques behave similarly in exploratory trajectories. However the map created by SLAM can be used to estimate the drift accumulated in exploration when returning to an already mapped environment. This loop closing capability requires an specific mechanism to detect loops and to correct them and its effectiveness is directly related to the richness of the sensor information to discriminate among visited locations.

- **Relocalization**. Localization is typically performed by predicting the current location of the robot from a previous state and finding correspondences between sensor measurements and the map. This process can fail for instance if the prediction made assumptions on the motion of the robot that are violated, or correspondences to the map cannot be found due to an occlusion of the sensor. In such events, a relocalization method is needed to find the location of the robot in the map without a prior prediction. This is also the problem of finding the initial location of the robot in the map when the robot is powered on in an already mapped environment. Relocalization also allows to perform mapping in several sessions. Again relocalization is more or less challenging depending on the sensor information being more or less rich to recognize the environment.

- **Map reuse**. A map can represent many aspects of the environment, like structure, semantics, topology, etc. However the main purpose of a map for SLAM is to localize the robot in it. Therefore when the robot is traversing an already mapped region, it should not duplicate the map but localize using the already existing map. The key problem here is how to determine which portion of the map has to be active to search for correspondences for localization in an scalable manner, which become critical for large scale operation. Relocalization and loop detection play an important role in determining this active portion of the map.

- **Real-time**. The goal of SLAM is to provide localization and map information to the robot so that it can be used to accomplish its mission. Therefore all algorithms have real-time constraints, and should scale well both in long-term and large-scale operation.

- **Robustness**. Finding correspondences between sensor measurements and the map can almost inevitably introduce wrong correspondences. SLAM techniques usually assume a static world and mapping dynamic elements can deteriorate accuracy or even make the system to fail. Therefore the SLAM system has to incorporate mechanisms to detect or reduce the impact of these outlier correspondences. Robustness also concerns the precision of the relocalization and loop detection method.

Among the sensors that provide information from the external world, known as **exteroceptive sensors**, vision is probably the most promising alternative. Cameras are passive sensors, at least traditional cameras (i.e. no RGB-D sensors), which means that they observe the world without altering it, in contrast to active sensors that require for example to send a signal and measure its reflection. Cameras are also relatively inexpensive and compact. We have the evidence that humans and many animals use vision as primary sensor to move through space and recognize places. In this sense, a single image contains a vast amount of information of the environment that can be used for **place recognition**, which is critical for loop detection and relocalization. Therefore **visual SLAM**, where a vision system is the main sensor, has been strongly developed in the last decade. In this thesis we aim to make our contribution to the advancement of the state-of-the-art of this field, which has great potential to enable life-changing technologies like autonomous vehicles, service robots or virtual and augmented reality applications.

In the following sections we introduce some basic mathematical formulations regarding sensors and optimization methods used in visual SLAM, we present the contributions of this work and the associated publications, open-source software and other dissemination.

1.2 Sensors for Visual SLAM

In this section we describe the main sensors for visual SLAM. A list of advantages and drawbacks of each sensor is shown in Table 1.1. The most common types of cameras for visual SLAM are shown in Fig. 1.1.

Table 1.1: Sensors for visual SLAM

Sensor	Advantages	Drawbacks
Monocular	• Smallest • Lowest power consumption • Cheapest • Minimal calibration	• Scale is unobservable • Scale drift • 3D only from multi-view • No mapping under pure rotations • Non-trivial SLAM initialization
Stereo	• 3D from one stereo frame • Trivial SLAM initialization	• More processing per frame • Extrinsic calibration
RGB-D	• Directly provide dense depth map • Trivial SLAM initialization • Dense maps	• Active sensor (interference) • Only indoors • Complex calibration • Power consumption
IMU	• Inter-frame motion estimation • Pitch and roll are observable • Scale for monocular SLAM	• Varying sensor biases • Gravity must be compensated • Observability issues • Visual-inertial calibration • Synchronization

(a) Monocular (b) Stereo (c) RGB-D

Figure 1.1: Different camera modalities for visual SLAM.

1.2.1 Monocular Camera

Monocular cameras are mainly composed of the image sensor and lens. We assume that the camera can be accurately modeled as a pinhole camera [28], once lens distortion has been removed, so that a 3D point $\mathbf{X}_{\mathtt{C}} \in \mathbb{R}^3$ in the camera coordinate reference system $\mathtt{C}$ is projected into 2D pixel coordinates $\mathbf{x}$ with the projection function $\pi_m : \mathbb{R}^3 \to \mathbb{R}^2$:

$$\mathbf{x} = \pi_m\left(\mathbf{X}_{\mathtt{C}}\right) = \begin{bmatrix} f_x \frac{X}{Z} + c_x \\ f_y \frac{Y}{Z} + c_y \end{bmatrix}, \quad \mathbf{X}_{\mathtt{C}} = [X, Y, Z]^T, \quad \mathbf{x} = [u, v]^T, \tag{1.1}$$

where f_x and f_y are the horizontal and vertical focal lengths, and c_x and c_y the horizontal and vertical coordinates of the principal point. These are intrinsic calibration parameters that can be computed from several images of a known calibration pattern. The camera coordinate system $\mathtt{C}$ has its origin at the optical center. With respect to the image, the Z axis is looking forward, the X axis is horizontal and points to the right, and the Y axis is vertical and points downwards. The projection function π assumes no distortion introduced by the lens. In practice distortion effects exist and have to be modeled so that one can transform from distorted to undistorted coordinates. Well known software and libraries like *Matlab* or *OpenCV* include toolboxes for camera calibration, including distortion. In this thesis we focus on cameras having a field of view (FOV) up to $\sim 100°$, for omnidirectional cameras and fisheye with very wide FOV, there exist more sophisticated mathematical models [70]. Global shutter cameras, which capture the full image at the same instant, are common in industry and when especially designed for computer vision, but consumer cameras are typically rolling shutter cameras, where pixel rows are captured at different time instants. Rolling shutter cameras produce artifacts when the camera or elements in the scene are moving, and reduce the accuracy of visual SLAM if not properly modeled. Modeling rolling shutter effect is out of the scope of this thesis, and we refer the reader to recent works such as [63] or [34].

Monocular cameras cannot observe the true scale of the world, and therefore monocular SLAM can only estimate the map and camera trajectory up to scale. In addition scale can drift and make distant portions of the map to be at different scales. Additional sources of information like IMU or known distances in the map are required to scale the solution.

1.2.2 Stereo Camera

Stereo cameras are composed of two rigidly attached cameras. Ideally both cameras are hardware synchronized so that image capturing is triggered at the same time. Depth can be estimated from just one stereo frame by finding correspondences between left and right pixels. To this end, in addition to intrinsic calibration of both cameras, the rotation and translation between both cameras have to be calibrated by processing several stereo frames of a calibration pattern. *OpenCV* also has a module for stereo calibration. The distance between both cameras, known as the baseline b, along with focal length and image resolution will determine the depth range at which depth estimation is accurate. As a rule of thumb depth can be accurately estimated if it is less than 40 times the stereo baseline [64]. In order to facilitate stereo matching, images are typically rectified, removing distortion and rotating them so that the epipolar lines are horizontal, i.e. the correspondence of a pixel in the left

image lies on the same row in the right image. The projection function for a rectified stereo camera $\pi_s : \mathbb{R}^3 \to \mathbb{R}^3$ is defined as follows:

$$\mathbf{x} = \pi_s\left(\mathbf{X}_{\mathtt{C}}\right) = \begin{bmatrix} f_x \frac{X}{Z} + c_x \\ f_y \frac{Y}{Z} + c_y \\ f_x \frac{X-b}{Z} + c_x \end{bmatrix}, \quad \mathbf{X}_{\mathtt{C}} = \left[X, Y, Z\right]^T, \quad \mathbf{x} = \left[u_L, v_L, u_R\right]^T, \tag{1.2}$$

where (u_L, v_L) are the coordinates in the left image and u_R is the horizontal coordinate in the right image. The vertical coordinates in both images are the same, $v_R = v_L$. We assume here that left and right cameras have the same intrinsic parameters after rectification.

1.2.3 RGB-D Camera

RGB-D cameras are the combination of a monocular RGB camera and a depth sensor, based on structured light or time of flight. By knowing the intrinsic calibration of the camera and extrinsic calibration between the camera and depth sensor, the measured depth can be registered into a depth map with 1:1 pixel correspondences to the RGB image. That implies that for every pixel in the image we know its depth without needing to perform a stereo matching as in the case of the stereo camera. However due to the nature of the depth sensor their use is restricted to indoors and the depth range is limited.

1.2.4 Inertial Measurement Unit

Inertial Measurement Units (IMU) are proprioceptive sensors composed of a gyroscope that measures the angular velocity, and an accelerometer that measures the linear acceleration of the sensor. While vision observes the external world, an IMU provides information of self-motion, which makes both sensors complementary. IMU can be used to estimate the motion between camera frames or to estimate the metric scale of monocular SLAM. Gravity can also be estimated which makes absolute pitch and roll observable.

The IMU, whose reference we denote with $\mathtt{B}$, measures the acceleration $\mathbf{a}_{\mathtt{B}}$ and angular velocity $\boldsymbol{\omega}_{\mathtt{B}}$ of the sensor at regular intervals Δt, typically at hundreds of Hertzs. In addition to sensor noise, both measurements are affected by slowly varying biases $\mathbf{b}_a$ and $\mathbf{b}_g$ of the accelerometer and gyroscope respectively. Moreover the accelerometer is subject to gravity $\mathbf{g}_{\mathtt{W}}$ and one needs to subtract its effect to compute the motion. The discrete evolution of the IMU orientation $\mathbf{R}_{\mathtt{WB}} \in \mathrm{SO}(3)$, position ${}_{\mathtt{W}}\mathbf{p}_{\mathtt{B}}$ and velocity ${}_{\mathtt{W}}\mathbf{v}_{\mathtt{B}}$, in the world reference $\mathtt{W}$, can be computed as follows [21]:

$$\begin{aligned} \mathbf{R}_{\mathtt{WB}}^{k+1} &= \mathbf{R}_{\mathtt{WB}}^{k} \operatorname{Exp}\left(\left(\boldsymbol{\omega}_{\mathtt{B}}^{k} - \boldsymbol{b}_g^k\right)\Delta t\right) \\ {}_{\mathtt{W}}\mathbf{v}_{\mathtt{B}}^{k+1} &= {}_{\mathtt{W}}\mathbf{v}_{\mathtt{B}}^{k} + \mathbf{g}_{\mathtt{W}}\Delta t + \mathbf{R}_{\mathtt{WB}}^{k}\left(\boldsymbol{a}_{\mathtt{B}}^{k} - \boldsymbol{b}_a^k\right)\Delta t \\ {}_{\mathtt{W}}\mathbf{p}_{\mathtt{B}}^{k+1} &= {}_{\mathtt{W}}\mathbf{p}_{\mathtt{B}}^{k} + {}_{\mathtt{W}}\mathbf{v}_{\mathtt{B}}^{k}\Delta t + \frac{1}{2}\mathbf{g}_{\mathtt{W}}\Delta t^2 + \frac{1}{2}\mathbf{R}_{\mathtt{WB}}^{k}\left(\boldsymbol{a}_{\mathtt{B}}^{k} - \boldsymbol{b}_a^k\right)\Delta t^2, \end{aligned} \tag{1.3}$$

where Exp is the exponential map for 3D rotation group SO(3) [21].

In order to fuse IMU and vision, both sensors should ideally be hardware synchronized so measurements from both sensors are timestamped with the same clock and without drift.

Moreover both sensors have to be extrinsically calibrated to know the transformation $\mathbf{T}_{\mathtt{CB}} = [\mathbf{R}_{\mathtt{CB}}|{}_{\mathtt{C}}\mathbf{p}_{\mathtt{B}}]$ between the reference of the camera and the IMU sensor [24].

1.3 Solving Visual SLAM

Given a stream of images by a vision sensor the question is how to exploit its information to perform visual SLAM. There are two main approaches: feature-based and direct methods.

1.3.1 Feature-based Methods and Bundle Adjustment

Feature-based methods process the images to extract distinctive interest points (keypoints) that can be reliably and repeatedly detected in images of the same scene ideally under different viewpoints and illumination conditions. A descriptor, typically a vector of binary or real values of a certain length, is computed for each keypoint by operating on a patch of pixels around the keypoint. This allows to match keypoints across images just by comparing their descriptors. The combination of a keypoint and its descriptor is called a feature. Once features have been extracted, the image can be discarded as feature-based methods only operate on these features. The advantage is that features are geometric entities that are easy to match and manipulate to compute initial solutions for geometry problems that are important in visual SLAM, like triangulation, epipolar geometry, the Perspective-n-Point (PnP) problem, and rigid body or similarity transformation between reference systems. Moreover feature-based optimizations are based on minimizing the reprojection error which is a geometric error with good convergence properties. Given a correspondence between a 3D point in world coordinates $\mathbf{X}_{\mathtt{W}}$ and a 2D keypoint $\mathbf{x}_{\mathtt{C}}$ in a monocular camera, the reprojection error $\mathbf{e}_{proj}$ is computed as follows:

$$\mathbf{e}_{proj} = \mathbf{x}_{\mathtt{C}} - \pi_m \left(\mathbf{R}_{\mathtt{CW}} \mathbf{X}_{\mathtt{W}} + {}_{\mathtt{C}}\mathbf{p}_{\mathtt{W}} \right), \tag{1.4}$$

where $\mathbf{R}_{\mathtt{CW}} \in \mathrm{SO}(3)$ and ${}_{\mathtt{C}}\mathbf{p}_{\mathtt{W}}$ are the rotation and translation of the inverse of the camera pose, which transforms points from world to camera coordinates.

The optimization of the positions of a set of points $\mathcal{P}$ and the poses of a set of cameras $\mathcal{C}$, minimizing the reprojection error, is called Bundle Adjustment (BA) [84] and it is the core optimization performed in modern feature-based visual SLAM:

$$\{\mathbf{X}^j_{\mathtt{W}}, \mathbf{R}^i_{\mathtt{CW}}, {}_{\mathtt{C}}\mathbf{p}^i_{\mathtt{W}} \mid \forall j \in \mathcal{P}, \forall i \in \mathcal{C}\} = \underset{\mathbf{X}^j_{\mathtt{W}}, \mathbf{R}^i_{\mathtt{CW}}, {}_{\mathtt{C}}\mathbf{p}^i_{\mathtt{W}}}{\operatorname{argmin}} \sum_{i,j} \rho \left(\left\| \mathbf{x}^j_i - \pi_m \left(\mathbf{R}^i_{\mathtt{CW}} \mathbf{X}^j_{\mathtt{W}} + {}_{\mathtt{C}}\mathbf{p}^i_{\mathtt{W}} \right) \right\|^2_{\mathbf{\Sigma}^j_i} \right), \tag{1.5}$$

where $\mathbf{x}^j_i$ is the keypoint associated to 3D point $\mathbf{X}^j_{\mathtt{W}}$ in camera i, $\mathbf{\Sigma}^j_i$ is the covariance of the location of keypoint $\mathbf{x}^j_i$ on the image of camera i, $\|\cdot\|_{\mathbf{\Sigma}}$ is the mahalanobis distance, and ρ is a robust cost function to downweight outlier correspondences. We use the Huber cost function in our implementations.

The main limitation of these approaches is precisely that they can only exploit visual information where features, typically corners, can be extracted. Lack of texture or motion blur can make a feature-based method to fail or perform very poorly. In addition the map generated by a feature-based approach is a sparse set of points with little use for other robotic tasks other than localization.

1.3.2 Direct Methods and the Photometric Error

Direct approaches use directly the sensor measurements, in this case the pixel intensity on the image. A direct method can be dense [60], if all pixels on the image are used, semi-dense [17] if only pixels with high gradient are considered, or sparse [16] if it only uses a small set of pixels on the image. These methods are able to exploit visual information without relying on keypoint detectors, and therefore are expected to be more accurate and robust when there is little texture in the scene or blur on the image. The reconstruction of direct SLAM consists in computing the depth associated to each pixel on selected cameras. The optimization is based on minimizing the photometric error. Given the 2D coordinates of a pixel $\mathbf{x}$ with estimated depth d on camera i, the photometric error $\mathbf{e}_{photo}$ when the pixel is observed in camera j is defined as follows:

$$\mathbf{e}_{photo} = I_i(\mathbf{x}) - I_j\left(\pi_m\left(\mathbf{R}^j_{\mathtt{CW}}\left(\mathbf{R}^i_{\mathtt{WC}}\pi_m^{-1}(\mathbf{x}, d) + {}_{\mathtt{W}}\mathbf{p}^i_{\mathtt{C}}\right) + {}_{\mathtt{C}}\mathbf{p}^j_{\mathtt{W}}\right)\right), \tag{1.6}$$

where $I : \mathbb{R}^2 \to \mathbb{R}^+$ is the function that returns the interpolated pixel intensity on the image for a given pixel position. $\pi_m^{-1} : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^3$ is the inverse projection function that computes the 3D location $\mathbf{X}_{\mathtt{C}}$ of a 2D point $\mathbf{x}$ in the image, given its depth d:

$$\mathbf{X}_{\mathtt{C}} = \pi_m^{-1}(\mathbf{x}, d) = \begin{bmatrix} d\,\frac{u-c_x}{f_x} \\ d\,\frac{v-c_y}{f_y} \\ d \end{bmatrix}, \quad \mathbf{X}_{\mathtt{C}} = [X, Y, Z]^T, \quad \mathbf{x} = [u, v]^T. \tag{1.7}$$

The photometric error (1.6) assumes lambertian surfaces, no gain or exposure changes between images, and no lens artifacts like vignetting. The recent sparse visual odometry work of Engel et al. [16] shows how to incorporate photometric calibration and exposure information in the photometric error. A photometric bundle adjustment, where several cameras and associated depth maps are jointly optimized to minimize the photometric error, is too computationally expensive for real-time SLAM in the case of dense and semi-dense approaches, which impose smoothness priors on the depth maps. Engel et al. [16] shows that by operating on a sparse set of pixels per image and without smoothness priors it is affordable to perform a sliding window photometric bundle adjustment in real-time and in standard CPUs.

The main limitation of dense and semi-dense approaches compared to a sparse direct approach is the computational complexity that forbids to jointly optimize in real-time structure and cameras, which reduce the achievable accuracy of these methods. In general direct optimizations only work if the initial seed for the optimization is close to the optimal, due to the nature of the photometric error. The reprojection of a pixel has to be close to the optimal projection so that the intensity gradients on the image can guide the optimization to its true location. To mitigate this problem, direct methods use image pyramids, but still the basin of convergence is narrower than for feature-based methods. This makes these methods more sensitive to rolling shutter or low frame-rate. Finally direct methods cannot provide initial solutions to geometry problems and rely on features to detect loops, compute the associated drift, or relocalize the camera.

1.3.3 Pose Graph Optimization

Bundle adjustment might be very expensive in large maps, moreover if the initial solution is far from the optimum. This is the case of loop closing, where the drift accumulated in the loop trajectory makes the state of the map to be far from the optimal and globally consistent map. An approximation of the bundle adjustment solution is to discard the structure and optimize only the camera poses, minimizing the relative transformation error. Given the pose of two cameras $\mathbf{T}_{\texttt{CW}}^{i} \in \mathrm{SE}(3)$ and $\mathbf{T}_{\texttt{CW}}^{j} \in \mathrm{SE}(3)$, and a measurement of the relative transformation $\hat{\mathbf{T}}_{\texttt{CC}}^{ij} \in \mathrm{SE}(3)$, the relative transformation error $\mathbf{e}_{rel}$ is:

$$\mathbf{e}_{rel}(i,j) = \mathrm{Log}_{\mathrm{SE}(3)} \left(\hat{\mathbf{T}}_{\texttt{CC}}^{ij} \mathbf{T}_{\texttt{CW}}^{j} \mathbf{T}_{\texttt{CW}}^{i^{-1}} \right), \tag{1.8}$$

where $\mathrm{Log}_{\mathrm{SE}(3)}$ is the logarithm map [73] that transforms from the manifold SE(3) to the tangent space which is locally Euclidean. In the monocular case, as there is scale drift, the poses and relative transformation errors are defined in terms of similarity transformations Sim(3) [75]. Given a set of edges in the pose graph $\mathcal{X}$, we define the cost to be minimized by the pose graph optimization:

$$\mathrm{C} = \sum_{(i,j)\in\mathcal{X}} \rho \left(\|\mathbf{e}_{rel}(i,j)\|^2_{\mathbf{\Sigma}_{ij}} \right). \tag{1.9}$$

After a pose graph optimization one can perform some iterations of bundle adjustment to get the optimal solution, which now will converge faster as the pose-graph optimization output is close to the optimum.

1.3.4 Non-Linear Optimization

Bundle adjustment and pose graph optimization are non-linear optimizations that can be solved using the standard Gauss-Newton method, and variants like Levenberg-Marquadt. Given a minimization problem of the form:

$$\mathbf{x} = \underset{\mathbf{x}}{\mathrm{argmin}}\, \mathrm{C}(\mathbf{x}) = \sum_{k} \rho \left(\|\mathbf{e}_k(x)\|^2_{\Sigma_k} \right), \tag{1.10}$$

the solution can be found by iteratively solving the normal equations:

$$\left(\mathbf{J}^T\mathbf{W}\mathbf{J}\right) \mathbf{\Delta x} = -\mathbf{J}^T\mathbf{W}\mathbf{e}, \tag{1.11}$$

where the hessian $\mathbf{J}^T\mathbf{W}\mathbf{J}$ and gradient $\mathbf{J}^T\mathbf{e}$ are computed as follows:

$$\begin{aligned} \mathbf{J}^T\mathbf{W}\mathbf{J} &= \sum_k \mathbf{J}_k^T \mathbf{W}_k \mathbf{J}_k \\ \mathbf{J}^T\mathbf{W}\mathbf{e} &= \sum_k \mathbf{J}_k^T \mathbf{W}_k \mathbf{e}_k \\ \mathbf{J}_k &= \frac{\partial \mathbf{e}_k(\mathbf{x} + \mathbf{\Delta x})}{\partial \mathbf{\Delta x}}, \end{aligned} \tag{1.12}$$

where $\mathbf{W}_k$ are the weights computed from the robust cost function ρ and the covariance of the measurements [84]. When dealing with camera poses or rotations the $\boldsymbol{\Delta}\mathbf{x}$ is computed on the tangent space of the associated manifold. For example, if we are optimizing a rotation $\mathbf{R} \in \mathrm{SO}(3)$ we would compute an increment as $\mathbf{R}\mathrm{Exp}(\boldsymbol{\delta\theta})$ and the Jacobians $\mathbf{J}_k$ would be computed with respect to an increment $\boldsymbol{\delta\theta}$ in the tangent space.

For more information about how to efficiently solve bundle adjustment we refer the reader to [84]. Non-linear optimization on manifolds SE(3), SO(3) and Sim(3) is greatly explained in [73] and a very clear description for SO(3) is given in [21]. In all our implementations we have used the C++ optimization package g2o [38], which is a flexible tool for optimization problems that can be represented as a graph.

1.4 Contributions

This thesis focus on investigating solutions for real-time, robust and accurate visual SLAM using monocular, stereo, RGB-D and inertial sensors, with the capability of loop closing, relocalization and map reuse. In particular we made the following contributions:

- **A place recognition method for real-time loop detection and relocalization** [54]. The method is based on DBoW2 [25] using ORB features [69] and is able to retrieve in less than 40ms the best image match in a keyframe database of 10K images. Our experiments show that a relocalization based on this method can handle scale changes between 0.4 and 2.9, any rotation around the optical axis, and an angular deviation of the optical axis up to 59 degrees to a given keyframe in the database. The method and evaluation is presented in **Chapter 3**. An improved version using covisibility information is presented in **Chapter 4**. We have use this place recognition module in all our visual SLAM approaches, with the same ORB vocabulary, achieving excellent results.

- **ORB-SLAM** [53]. A feature-based system which is currently the most reliable and complete solution for monocular SLAM, including a robust automatic initialization, loop closing and relocalization. The system operates in real-time in large environments and is able to process sequences from hand-held cameras, cars, ground robots or drones. We extensively evaluate the solution in 27 public sequences achieving unprecedented accuracy and robustness. Moreover we have made our implementation open-source. We describe ORB-SLAM and present the evaluation in **Chapter 4**.

- **A probabilistic semi-dense mapping method** [55], which, integrated in ORB-SLAM, results in a novel SLAM system that combines the high localization accuracy of feature-based approaches and the semi-dense reconstruction of direct methods. This system is described in **Chapter 5**.

- **ORB-SLAM2** [56]. This is the first open-source SLAM solution for monocular, stereo and RGB-D cameras. Our RGB-D results show that by using bundle adjustment we achieve more accuracy than state-of-the-art implementations based on ICP or direct

methods. By using close and far stereo points, and monocular observations in the bundle adjustment, our stereo results are more accurate than the state-of-the-art direct stereo SLAM. ORB-SLAM2 is described in **Chapter 6**.

- **Visual-inertial monocular ORB-SLAM** [57]. This is to the best of our knowledge the first tightly-coupled keyframe-based SLAM solution able to metrically close loops in real-time and reuse its map, to achieve drift-free localization in already mapped environments. We also propose a novel IMU initialization method, which computes the scale, the gravity direction, the velocity, and gyroscope and accelerometer biases, in a few seconds with high accuracy. We test our system in the 11 sequences of a recent micro-aerial vehicle public dataset achieving a typical scale factor error of 1% and centimeter precision. We compare to the state-of-the-art in visual-inertial odometry in sequences with revisiting, proving the better accuracy of our method due to map reuse and no drift accumulation. The system is described in **Chapter 7**.

1.5 Dissemination

1.5.1 Peer-Reviewed Publications

The research developed in this thesis has resulted in the following peer-reviewed publications:

- Raúl Mur-Artal and Juan D. Tardós. "Fast Relocalisation and Loop Closing in Keyframe-Based SLAM". *IEEE International Conference on Robotics and Automation (ICRA)*. Hong Kong, China, June 2014.

- Raúl Mur-Artal and Juan D. Tardós. "ORB-SLAM: Tracking and Mapping Recognizable Features". *RSS 14 Workshop on Multi VIew Geometry in RObotics (MVIGRO)*. Berkeley, USA, July 2014.

- Raúl Mur-Artal and Juan D. Tardós. "Probabilistic Semi-Dense Mapping from Highly Accurate Feature-Based Monocular SLAM". *Robotics: Science and Systems (RSS)*. Rome, Italy, July 2015.

- Raúl Mur-Artal, J. M. M. Montiel and Juan D. Tardós. "ORB-SLAM: A Versatile and Accurate Monocular SLAM System". *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147-1163, October 2015. **(2015 IEEE Transactions on Robotics Best Paper Award)**.

- Raúl Mur-Artal, and Juan D. Tardós. "Visual-Inertial Monocular SLAM with Map Reuse". *IEEE Robotics and Automation Letters*, 2017. (Accepted for publication).

The following publication is under review:

- Raúl Mur-Artal, and Juan D. Tardós. "ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras". *ArXiv preprint* arXiv:1610.06475, 2016.

1.5.2 Open-Source Software

We have released the following open-source software:

- **ORB-SLAM** (https://github.com/raulmur/ORB_SLAM),

  A Versatile and Accurate Monocular SLAM

- **ORB-SLAM2** (https://github.com/raulmur/ORB_SLAM2),

  Real-Time SLAM for Monocular, Stereo and RGB-D Cameras, with Loop Detection and Relocalization Capabilities.

The *Universidad de Zaragoza* has licensed the software to companies in Asia, Europe and America for its commercial exploitation in the fields of Robotics, Augmented and Virtual Reality.

1.5.3 Videos

Demonstrating videos of ORB-SLAM:

- KITTI 00: https://youtu.be/8DISRmsO2YQ
- KITTI 05: https://youtu.be/sr9H3ZsZCzc
- fr3_office: https://youtu.be/_9VcvGybsDA
- fr3_walking_halfsphere: https://youtu.be/ZdxgIbd7nhI

Demonstrating video of ORB-SLAM with semi-dense mapping:

- TUM RGB-D dataset: https://youtu.be/HlBmq7OLKrQ

Demonstrating videos of ORB-SLAM2:

- Overview: https://youtu.be/ufvPS5wJAx0
- Tsukuba: https://youtu.be/dF7_I2Lin54

Demonstrating videos of visual-inertial ORB-SLAM:

- EuRoC MH05: https://youtu.be/JXRCSovuxbA
- EuRoC V102: https://youtu.be/rdR5OR8egGI

1.5.4 Invited Talks

Some of the contents of this thesis have been presented in the following invited talks:

- Juan D. Tardós. "ORB-SLAM: a Real-Time Accurate Monocular SLAM System". *Qualcomm Augmented Reality Lecture Series*. Vienna, Austria, June 2015.
- Juan D. Tardós. "Visual SLAM: Feature-Based .vs. Direct Methods". *The Problem of Mobile Sensors Workshop at RSS*, Rome, Italy, July 2015.
- Raúl Mur-Artal. "Should we still do sparse feature based SLAM?". *The Future of Real-Time SLAM Workshop at ICCV*, Santiago, Chile, December 2015.
- Juan D. Tardós. "Feature-Based Visual SLAM". *SLAM Tutorial at ICRA*, Stockholm, Sweden, May 2016.

Chapter 2

Related Work

2.1 Place Recognition

The survey by Williams et al. [88] compared several approaches for place recognition and concluded that techniques based on appearance, that is image to image matching, scale better in large environments than map to map or image to map methods. Within appearance based methods, bags of words techniques [62], such as the probabilistic approach FAB-MAP by Cummins and Newman [12], are to the fore because of their high efficiency. DBoW2 by Gálvez-López and Tardós [25] used for the first time bags of binary words obtained from BRIEF descriptors [7] along with the very efficient FAST keypoint detector [68]. This reduced in more than one order of magnitude the time needed for feature extraction, compared to SURF [2] and SIFT [45] features that were used in bags of words approaches so far. Although the system demonstrated to be very efficient and robust, the use of BRIEF, neither rotation nor scale invariant, limited the system to in-plane trajectories and loop detection from similar viewpoints. In Chapter 3 we propose a bag of words place recognizer built on DBoW2 with ORB [69] features. ORB are binary while being invariant to rotation and scale (in a certain range), resulting in a very fast recognizer with good invariance to viewpoint. We demonstrate in Section 3.2 the high recall and robustness of a secuential loop detector using this recognizer in four different datasets, requiring less than 39ms (including feature extraction) to retrieve a loop candidate from a 10K image database. In Section 3.3 we analyze the invariance to viewpoint changes of a relocalization method based on this place recognizer, showing that it tolerates scale changes from 0.36 to 2.93, any rotation around the optical axis, and up to 59 degrees of deviation from the optical axis. In Chapter 4 we improve this place recognizer, using covisibility information and returning several hypotheses when querying the database instead of just the best match, and embeed it on our own Visual SLAM pipeline to perform relocalization and loop detection.

2.2 Monocular Initialization

Monocular SLAM requires a procedure to create an initial map because depth cannot be recovered from a single image. One way to solve the problem is to initially track a known

structure [13]. In the context of filtering approaches, points can be initialized with high uncertainty in depth using an inverse depth parametrization [9], which hopefully will later converge to their real positions. The recent semi-dense work of Engel et al. [17], follows a similar approach initializing the depth of the pixels to a random value with high variance.

Initialization methods based on multiple view geometry [28] compute the camera motion between two frames and triangulate an initial map. These methods either assume a dominant plane in the scene, like in [23, 36], and recover the relative camera pose from a homography using the method of Faugeras et al.[20], or like [42, 81], compute an essential matrix that models planar and general scenes, using the five-point algorithm of Nister [61], which requires to deal with multiple solutions. Both [20] and [61] methods are not well constrained under low parallax and suffer from a twofold ambiguity solution if all points of a planar scene are closer to one of the camera centers [43]. On the other hand if a non-planar scene is seen with parallax, a unique fundamental matrix can be computed with the eight-point algorithm [28] and the relative camera pose can be recovered without ambiguity. However planar or low-parallax configurations are degenerated cases for the eight-point algorithm an can yield catastrophic results if not detected.

We present in Section 4.2 a new automatic approach based on model selection between a homography for planar scenes and a fundamental matrix for non-planar scenes. A statistical approach to model selection was proposed by Torr et al. [82]. Under a similar rationale we have developed a heuristic initialization algorithm that takes into account the risk of selecting a fundamental matrix in close to degenerate cases (i.e. planar, nearly planar, and low parallax), favoring the selection of the homography. In the planar case, for the sake of safe operation, we refrain from initializing if the solution has a twofold ambiguity, as a corrupted solution could be selected. Our proposed method delays initialization until the method returns a unique solution with significant parallax.

2.3 Monocular SLAM

Monocular SLAM was initially solved by filtering [8, 9, 13, 14]. In that approach every frame is processed by the filter to jointly estimate the map landmark locations and the camera pose. It has the drawbacks of wasting computation in processing consecutive frames with little new information for map refinement and the accumulation of linearization errors. On the other hand keyframe-based approaches [36, 51] estimate the map using only selected frames (keyframes) allowing to perform more costly but accurate bundle adjustment [84] optimizations, as mapping is not tied to frame-rate. Strasdat et al.[76] demonstrated that keyframe-based techniques are more accurate than filtering for the same computational cost.

The most representative keyframe-based SLAM system is probably PTAM by Klein and Murray [36]. It was the first work to introduce the idea of splitting camera tracking and mapping in parallel threads, and demonstrated to be successful for real time augmented reality applications in small environments. The original version was later improved with edge features, a rotation estimation step during tracking and a better relocalization method [37]. The map points of PTAM correspond to FAST corners matched by patch correlation. This makes the points only useful for tracking but not for place recognition. In fact PTAM

does not detect large loops, and the relocalization is based on the correlation of low resolution thumbnails of the keyframes, yielding a low invariance to viewpoint.

Strasdat et al.[75] presented a large scale monocular SLAM system with a front-end based on optical flow implemented on a GPU, followed by FAST feature matching and *motion-only BA*, and a back-end based on *sliding-window BA*. Loop closures were solved with a pose graph optimization with similarity constraints (7DoF), that was able to correct the scale drift appearing in monocular SLAM. From this work we take the idea of loop closing with 7DoF pose graph optimization and apply it to the *Essential Graph* defined in Section 4.1.4

Strasdat et al.[74] adapted the front-end of PTAM to perform tracking in a local map retrieved from a covisibility graph, instead of the full map as done in the original approach. They proposed a double window optimization back-end that continuously performs BA in the inner window, and pose graph in a limited-size outer window. However, loop closing is only effective if the size of the outer window is large enough to include the whole loop. In our monocular SLAM system, described in Chapter 4 , we take advantage of the excellent ideas of using a local map based on covisibility, and building the pose graph from the covisibility graph, but apply them in a totally redesigned front-end and back-end. Another difference is that, instead of using specific features for loop detection (SURF), we perform the place recognition on the same tracked and mapped features, obtaining robust frame-rate relocalization and loop detection.

Pirker et al.[66] proposed CD-SLAM, a very complete system including loop closing, relocalization, large scale operation and efforts to work on dynamic environments. However map initialization is not mentioned. The lack of a public implementation does not allow us to perform a comparison of accuracy, robustness or large-scale capabilities.

The visual odometry of Song et al. [72] uses ORB features for tracking and a temporal sliding window BA back-end. In comparison our system is more general as they do not have global relocalization, loop closing and do not reuse the map. They are also using the known distance from the camera to the ground to limit monocular scale drift.

Lim et al.[42] use BRIEF features for tracking, mapping and loop detection. However the choice of BRIEF limits the system to in-plane trajectories. Their system only tracks points from the last keyframe so the map is not reused if revisited (similar to a visual odometry) and has the problem of growing unbounded. We compare qualitatively our results with this approach in Section 4.6.5.

The work of Engel et al.[17], known as LSD-SLAM, is a direct SLAM able to build large scale semi-dense maps. Their results are very impressive as the system is able to operate in real time, without GPU acceleration, building a semi-dense map, with more potential applications for robotics than the sparse output generated by feature-based SLAM. Nevertheless they still need features for loop detection and their camera localization accuracy is significantly lower than in our system and PTAM, as we show experimentally in Section 4.6.2. This surprising result is discussed in Section 4.7.2.

In a halfway between direct and feature-based methods is the semi-direct visual odometry SVO of Forster et al. [23]. Without requiring to extract features in every frame they are able to operate at high frame-rates obtaining impressive results in quadracopters. However as a visual odometry the method is neither able to reuse its map nor to close loops.

Regarding keyframe selection, it is clear that running BA with all the points and all the

frames is not feasible in real-time. The work of Strasdat et al. [76] showed that the most cost-effective approach is to keep as much points as possible, while keeping only non-redundant keyframes. The PTAM approach was to insert keyframes very cautiously to avoid an excessive growth of the computational complexity. This restrictive keyframe insertion policy makes the tracking fail in hard exploration conditions. Our *survival of the fittest* strategy achieves unprecedented robustness in difficult scenarios by inserting keyframes as quickly as possible, and removing later the redundant ones, to avoid the extra cost.

2.4 Stereo SLAM

A remarkable early stereo SLAM system was the work of Paz et al.[64]. Based on Conditionally Independent Divide and Conquer EKF-SLAM it was able to operate in larger environments than other approaches at that time. Most importantly, it was the first stereo SLAM exploiting both close and far points (i.e. points whose depth cannot be reliably estimated due to little disparity in the stereo camera), using an inverse depth parametrization [9] for the latter. They empirically showed that points can be reliably triangulated if their depth is less than $\sim$40 times the stereo baseline. In this work we follow this strategy of treating in a different way *close* and *far* points, as explained in Section 6.1.1.

Most modern stereo SLAM systems are keyframe-based [76] and perform BA optimization in a local area to achieve scalability. The work of Strasdat et al. [74] performs a joint optimization of BA (point-pose constraints) in an inner window of keyframes and pose-graph (pose-pose constraints) in an outer window. By limiting the size of these windows the method achieves constant time complexity, at the expense of not guaranteeing global consistency. The RSLAM of Mei et al. [49] uses a relative representation of landmarks and poses and performs relative BA in an active area which can be constrained for constant-time. RSLAM is able to close loops which allow to expand active areas at both sides of a loop, but global consistency is not enforced. The recent S-PTAM by Pire et al. [65] performs local BA, however it lacks large loop closing. Similar to these approaches our stereo visual SLAM, described in Chapter 6, performs BA in a local set of keyframes so that the complexity is independent of the map size and can operate in large environments. However our goal is to build a globally consistent map. Our system aligns first both sides of the loop, similar to RSLAM, so that the tracking is able to continue localizing using the old map and then performs in parallel a pose-graph optimization that minimizes the drift accumulated in the loop, followed by *full BA*.

The recent Stereo LSD-SLAM of Engel et al. [18] is a semi-dense direct approach that minimizes photometric error in image regions with high gradient. Not relying on features, the method is expected to be more robust to motion blur or poorly-textured environments. However as a direct method its performance can be severely degraded by unmodeled effects like rolling shutter or non-lambertian reflectance.

2.5 RGB-D SLAM

One of the earliest and most famed RGB-D SLAM systems was the KinectFusion by Newcombe et al. [59]. This method fused all depth data from the sensor into a volumetric dense

model that was used to track the camera pose using ICP. This system was limited to small workspaces due to its volumetric representation and the lack of loop closing. Kintinuous by Whelan et al. [86] was able to operate in large environments by using a rolling cyclical buffer and included loop closing using place recognition and pose graph optimization.

Probably the first popular open-source system was the RGB-D SLAM of Endres et al. [15]. This is a feature-based system, whose front-end computes frame-to-frame motion by feature matching and ICP. The backend performs pose-graph optimization with loop closure constraints from a heuristic search. Similarly the back-end of DVO-SLAM of Kerl et al. [35] optimizes a pose-graph where keyframe-to-keyframe constraints are computed from a visual odometry that minimizes both photometric and depth error. DVO-SLAM also searches for loop candidates in a heuristic fashion over all previous frames, instead of relying on place recognition.

The recent ElasticFusion by Whelan et al. [87] builds a surfel-based map of the environment. This is a map-centric approach that forget poses and performs loop closing applying a non-rigid deformation to the map, instead of a standard pose-graph optimization. The detailed reconstruction and localization accuracy of this system is impressive, but the current implementation is limited to room-size maps as the complexity scales with the number of surfels in the map.

As proposed by Strasdat et al. [74] we use depth information to synthesize a stereo coordinate for extracted features on the image. This way our visual SLAM system, described in Chapter 6 is agnostic about the input being stereo or RGB-D. Differently to all above methods our back-end is based on bundle adjustment and builds a globally consistent sparse reconstruction. Therefore our method is lightweight and works with standard CPUs. Our goal is long-term and globally consistent localization instead of building the most detailed dense reconstruction. However from the highly accurate keyframe poses one could fuse depth maps and get accurate reconstruction on-the-fly in a local area or post-process the depth maps from all keyframes after a *full BA* and get an accurate 3D model of the whole scene.

2.6 Visual-Inertial SLAM

Visual-inertial fusion has been a very active research topic in the last years. The recent research is focused on tightly-coupled (i.e. joint optimization of all sensor states) visual-inertial odometry, using keyframe-based non-linear optimization, such as the relevant works of Indelman et al. [30], Leutenegger et al. [41], Usenko et al. [85], Forster et al. [21] and Concha et al. [10], or filtering, such as the relevant works of Mourikis and Roumeliotis [52], Wu et al. [89] and Bloesch et al. [4]. Nevertheless these approaches are only able to compute incremental motion and lack the capability to close loops and reuse a map of an already mapped environment. This implies that estimated trajectory accumulates drift without bound, even if the sensor is always localizing in the same environment. This is due to the marginalization of past states to maintain a constant computational cost [4, 41, 52, 85, 89], or the use of full smoothing [21, 30], with an almost constant complexity in exploration but that can be as expensive as a batch method in the presence of loop closures [32]. The method of Jones and Soatto [31], based on filtering, is able to close loops topologically and reuse its

map, however global metric consistency is not enforced in real-time. Recently Lynen et al. [47] presented a very well engineered system to perform visual-inertial tracking in a given map which was built offline.

Building on the preintegration of Lupton and Sukkarieh [46], its application to the SO(3) manifold by Forster et al. [21] and its factor graph representation by Indelman et al. [30], we present in Chapter 7, Visual-Inertial ORB-SLAM, to the best of our knowledge the first keyframe-based Visual-Inertial SLAM that is able to metrically close loops in real-time and reuse the map that is being built online. Following the approach of ORB-SLAM, which we present in Chapter 4 and that is inspired by the work of Klein and Murray [36], our tracking optimizes the current frame assuming a fixed map, and our backend performs local Bundle Adjustment (BA), optimizing a local window of keyframes, including an outer window of fixed keyframes. In contrast to full smoothing, this approach allows for a constant time local BA, and by not marginalizing past states, we are able to reuse them. We detect large loops using place recognition and correct them using a lightweight pose-graph optimization, followed by full BA in a separate thread, not to interfere with real-time operation.

Chapter 3

Place Recognition with ORB Features

One key ability of any visual SLAM system is to recognize already mapped environments. This allows the system to relocalize the sensor after a tracking failure, which might happen due to an occlusion or an abrupt movement, and to detect trajectory loops which can be used to correct the error accumulated during exploration. In this Chapter we propose a place recognition method building on DBoW2 [25] in combination with ORB [69] features, for the tasks of sequential loop detection and relocalization. ORB are binary features which are fast to extract and match, while they are rotation invariant and scale invariant in a range. We build on this place recognition approach for the loop detection and relocalization of ORB-SLAM that we present in Chapter 4.

3.1 Review of DBoW2

In order to detect if an image corresponds to a revisited place, bag of words techniques summarize the content of an image by the visual words it contains. These visual words correspond to a discretization of the descriptor space, known as the visual vocabulary. DBoW2 creates a vocabulary structured as a tree [62], in an offline step over a big set of descriptors, extracted from a training image dataset.

Processing a new image consist in extracting keypoints and their descriptors, which are assigned to a visual word traversing the precomputed vocabulary tree. As descriptors are binary, distances are computed by the Hamming distance. The result is a bag of words vector, containing the *term frequency - inverse document frequency* (tf-idf) score for each word present in the image. This score is higher as more frequent is a word in the image and less it was in the training dataset. This bag of words vector is then compared against the bag of words vectors of the images in the database that is incrementally build. To speed up the search on the database, DBoW2 maintains an inverse index, which stores for each word, in which images it appeared.

The similarity between two bag of word vectors $\mathbf{v_1}$ and $\mathbf{v_2}$ is the L_1-score:

$$s(\mathbf{v_1}, \mathbf{v_2}) = 1 - \frac{1}{2}\left| \frac{\mathbf{v_1}}{|\mathbf{v_1}|} - \frac{\mathbf{v_2}}{|\mathbf{v_2}|} \right| \tag{3.1}$$

This score is normalized with the score one would expect to get for an image showing the same place. For loop detection in a video sequence a reference score can be computed from the previous image:

$$\eta(\mathbf{v_i}, \mathbf{v_j}) = \frac{s(\mathbf{v_i}, \mathbf{v_j})}{s(\mathbf{v_i}, \mathbf{v_{i-1}})} \tag{3.2}$$

Images in the database close in time may have similar scores. DBoW2 takes advantage of it, grouping images close in time and computing only one score for the group, which is the sum of the individual scores. Individual scores have to be higher than a threshold α to be considered. Once the database is searched, the group with the highest score is selected and the image with the maximum individual score is considered as a loop candidate for the query image.

A loop candidate in order no to be rejected has to be consistent with k previous queries. It means that the groups with the highest scores for the last k images must form an overlapping sequence. This temporal consistency test improves the robustness as a loop is only accepted if supported by enough evidence.

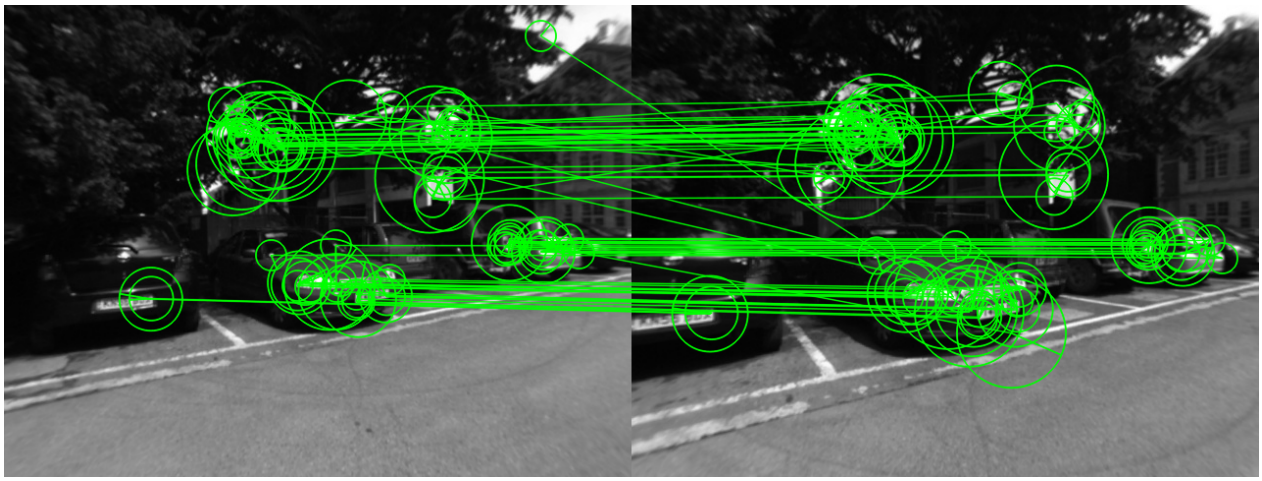
Finally a loop candidate is accepted if it passes a geometrical check, which consists in computing a fundamental matrix with RANSAC [28]. The search for initial correspondences is performed exhaustively but only between those features that belong to the same node at a level l of the vocabulary tree. The database uses a direct index that stores for each feature in an image the node at level l it belongs.

3.2 Place Recognition with ORB

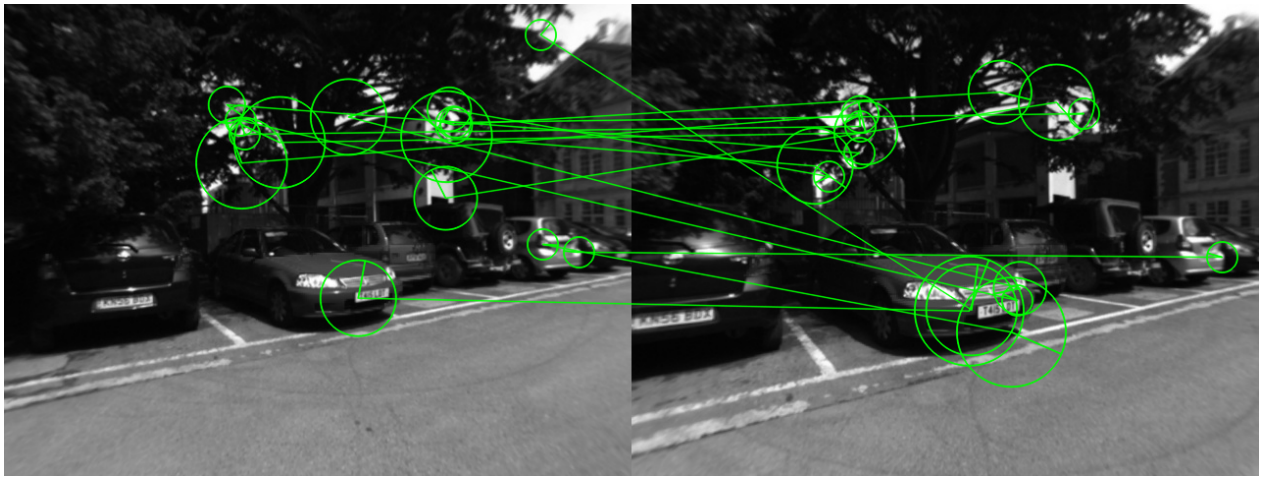
In this section we propose a place recognition in image sequences based on DBoW2 with ORB features. We rely on the ORB extractor of OpenCV library and extract 1000 keypoints at 8 scale levels with a scale factor of 1.2. We create the visual vocabulary in an offline step with the large dataset Bovisa 2008-09-01 [5]. This dataset is a sequence with outdoors and indoors areas, yielding a vocabulary that will provide good results in both scenarios. We build a vocabulary of 6 levels and 10 clusters per level, getting one million words. Such a big vocabulary is suggested in [25] to be efficient for recognition in large image databases.

The initial correspondences of ORB features between two images are computed by exhaustive descriptor comparison between features that belong to the same node at level 4 in the vocabulary tree (counting from leaves up), and applying a nearest neighbor ratio of 0.6. We propose the use of an orientation consistency test to filter out wrong correspondences, improving the robustness and speeding up the geometrical verification. For each initial match, we compute the rotation increment between the ORB keypoints in both images, voting for rotations discretized in 60 bins, which we found suitable in our experiments to discard most of the outliers. Only those correspondences of the three most voted rotations will be accepted. An example of this orientation consistency test is shown in Fig. 3.1.

In order to evaluate the performance of our place recognizer, we run it in four image sequences, NewCollege [71], Bicocca25b [5], Malaga6L [3] and CityCentre [11]. We measure the recall and precision of the loops detected, processing the sequences at the same frequency as in [25] and with the same geometrical test (fundamental matrix with RANSAC) in order



(a) Putative matches.



(b) Discarded matches by orientation consistency test.

Figure 3.1: Example of the orientation consistency test to reject outlier matches between oriented keypoints.

to make comparisons. In our geometrical test we include our orientation consistency test. Experiments were performed in an Intel Core i5 @ 2.30 GHz computer with the parameters shown in Table 3.1.

The results of our proposed loop detector and the results provided in [25] for DBoW2 with BRIEF and FAB-MAP 2.0 (only for two of the datasets) are shown in Table 3.2. Our proposed loop detector achieves high recall in all datasets with no false positives (100 % precision). Our results are better than the original results with BRIEF in all sequences, by the exception of Bicocca25b. This can be explained because Bicocca25b is an indoor sequence where loop events have a very similar point of view, in such case a descriptor neither invariant to rotation nor to scale is expected to get better results. On the other hand NewCollege, Malaga6L and CityCentre are outdoor sequences where there exist bigger viewpoint differences at loop events, and therefore ORB, rotation invariant and scale aware,

Table 3.1: Loop Detector Parameters

DBoW2 parameters
Temporal consistency (k): 3
Score threshold (α): 0.3
Direct index level (l): 4
Nearest. neighbor ratio: 0.6

ORB parameters
Number of features: 1000
Scale levels: 8
Scale factor: 1.2

Visual vocabulary
Vocabulary tree levels: 6
Vocabulary clusters/level: 10

Table 3.2: Comparison with the results provided in [25].

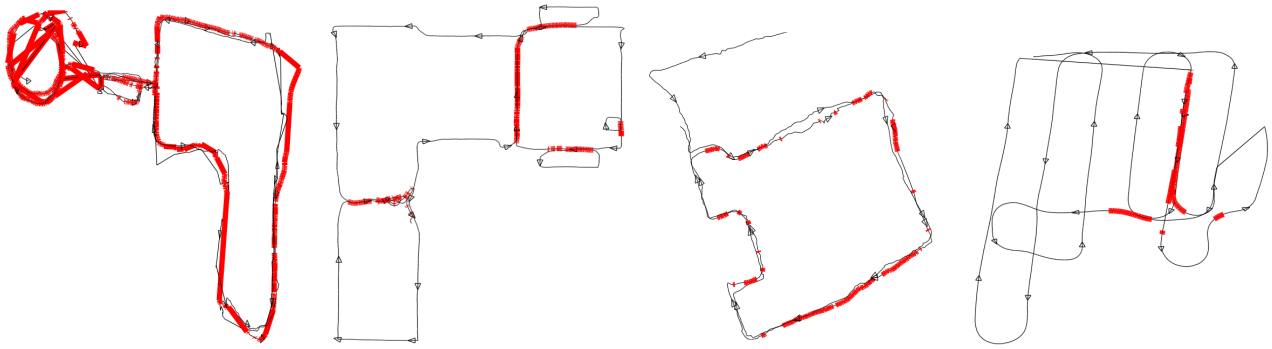
	Proposed		DBoW2		FAB-MAP 2	
Dataset	Precision	Recall	Precision	Recall	Precision	Recall
NewCollege	100%	**70%**	100%	56%	-	-
Bicocca25b	100%	77%	100%	**81%**	-	-
Malaga6L	100%	**82%**	100%	74%	100%	69%
CityCentre	100%	**43%**	100%	31%	100%	39%

performs better. Our approach gets also higher recall than FAB-MAP 2.0 in Malaga6L and CityCentre. Figure 3.2 shows the robot trajectories in each dataset and the loops detected by our proposed loop detector.

In order to complete the performance analysis of this loop detector, we ran it in a 10K image sequence from NewCollege and measured the extraction time of ORB features and the time spent to retrieve the best match when querying the database (not including geometrical test). Times are shown in Fig. 3.3. ORB extraction is performed in 13.3 ms on average, and image retrieval in less than 25 ms for a database containing 10K images. This makes the proposed method suitable for real-time place recognition in large maps.

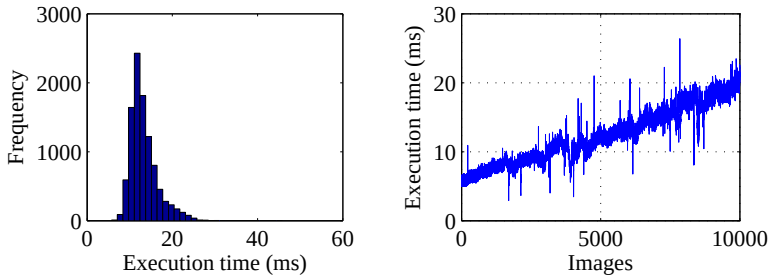
3.3 Viewpoint Invariance

The purpose of this experiment is to evaluate the robustness to viewpoint change of a relocation system built on the place recognition presented in the previous section. Specifically we measure the change in scale, the in-plane rotation and the change in viewpoint angle. We start from the reconstruction of a wall with several textured posters using PTAM [36], the mapping is then disabled, and the camera is occluded and moved to another place and we



(a) NewCollege (b) Bicocca25b (c) CityCentre (d) Malaga6L

Figure 3.2: Loop detection results in each dataset. The trajectory is drawn in red when a loop is detected.

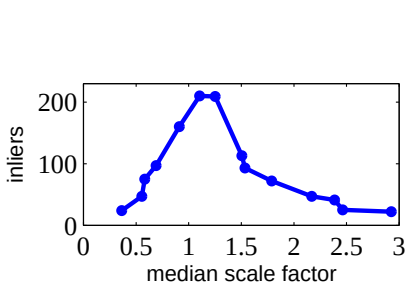


(a) ORB extraction (histogram) (b) Database Query

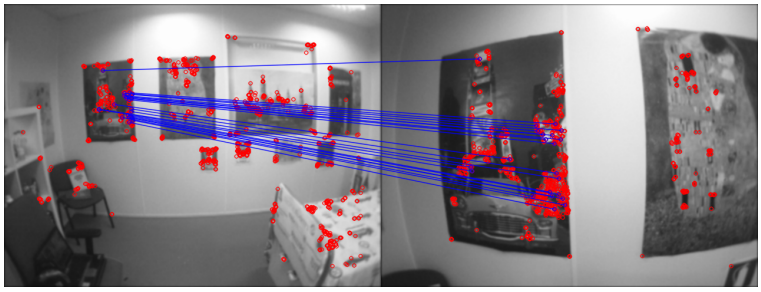
Figure 3.3: Execution times for 10K images from NewCollege.

try to relocalize it.

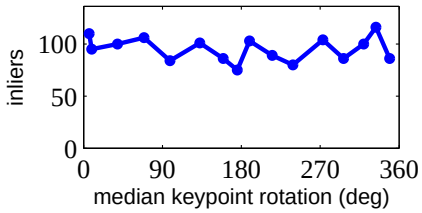
The relocalization method consists firstly in matching the current frame with a keyframe created by PTAM, which are stored in the database of DBoW2. We use the same ORB settings and vocabulary described in the previous section. Once there is a keyframe candidate, we search for correspondences between ORB in the current frame and the keyframe candidate. In order to compute the camera pose by using a Perspective-n-Point (PnP) algorithm, we need the 3D location for the ORB in the keyframe. As ORB features used for place recognition are different from the features used by PTAM for mapping we do not have 3D information for ORB in the keyframe, therefore we interpolate the depth of each ORB feature with its three nearest PTAM tracked features in the keyframe. All those ORB features that are further away than 10 pixels from a tracked feature are discarded, as interpolation might be poor. After this step we have a set of 2D (current frame) to 3D (keyframe) point correspondences. We then use a RANSAC scheme, selecting at each iteration four correspondences and solving a P4P problem [40], which returns a camera pose. If RANSAC is able to find a camera pose supported by more than 40% of the initial correspondences (but at less 20 inliers), in less than 178 iterations (99% of success), then the relocalization is considered successful. The camera pose is refined using all inliers, and recomputing it once again if more inliers are found. Fig. 3.4 shows the results of the experiments showing the PnP inliers for each



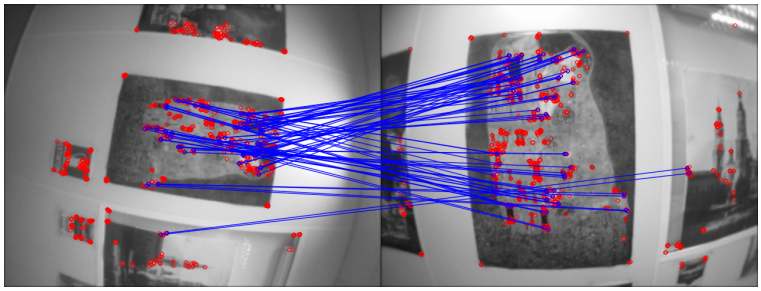
(a) scale change



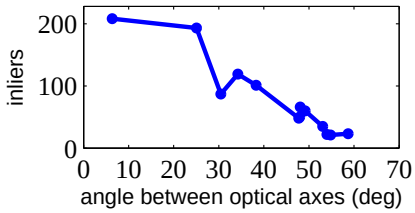
(b) scale change: 2.93



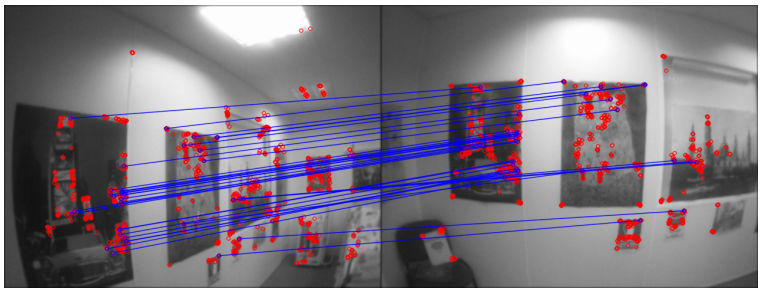
(c) in-plane rotation



(d) rotation: 98.7 deg.



(e) viewpoint angle



(f) angle difference: 53 deg.

Figure 3.4: On the left: PnP inliers at each relocalistation. On the right: an example for each experiment. Blue lines are the inlier correspondences.

case. Our system can handle scale changes between 0.36 and 2.93, any in-plane rotation, and the camera can relocate to a keyframe with an angle difference in the optical axis up to 59 degrees. This results are only indicative as they can vary with the scene properties (textures, object distribution, etc.).

3.4 Discussion

In this chapter we have proposed a place recognition method based on DBoW2 and ORB features with a novel orientation consistency test. The results of this place recognition for loop detection in image sequences, presented in Section 3.2 outperform other state-of-the-art approaches. We have also evaluated in Section 3.3 the invariance to viewpoint of a relocalization method based on this place recognition achieving very satisfactory results with

an invariance to scale between 0.36 and 2.93, any in-plane rotation, and an angle difference in the optical axis up to 59 degrees. We also integrated the loop detection and relocalization in PTAM and showed its good real-time performance in [54]. Based on this promising results we build on this place recognition in Chapter 4 and integrate it in all the visual SLAM systems described in this thesis. One of our main improvements that are explained in Section 4.1.5 is the use of covisibility information instead of temporal information. The place recognition described in this chapter groups images in the database by temporal order to boost the recall when querying the database. This is suitable for video sequences where the image database is built by sampling images at a regular interval and for trajectories with only one revisit. However when applied in a visual SLAM framework where the camera can continually revisit the same environment, the keyframe database might contain several keyframes of the same scene inserted at very different times. Therefore images should be grouped by covisibility of the same scene instead of by temporal ordering.

We have tested our visual SLAM approaches in the well-known public NewCollege [71], TUM RGB-D [78], KITTI [26] and EuRoC [6] datasets, and in hand-held demonstrations in many different environments, achieving excellent loop detection performance and exhibiting a bullet-proof relocalization. Moreover we have always used the same ORB vocabulary, which containing 1 million words, demonstrates that when the vocabulary is large it can be successfully applied to any environment. However, while the precision of the loop detector is very high, it can potentially detect false loop closures in very repetitive man-made environments that would result in map corruption. Although we have not addressed this problem in this thesis, we consider that loop closing strategies have to be robustified against false loop closures [39, 80].

Chapter 4

Monocular ORB-SLAM

Bundle Adjustment (BA) is known to provide accurate estimates of camera localizations as well as a sparse geometrical reconstruction [28, 84], given that a strong network of matches and good initial guesses are provided. For long time this approach was considered unaffordable for real time applications such as visual SLAM. Nowadays we know that to achieve accurate results at non-prohibitive computational cost, a real time SLAM algorithm has to provide BA with:

- Corresponding observations of scene features (map points) among a subset of selected frames (keyframes).
- As complexity grows with the number of keyframes, their selection should avoid unnecessary redundancy.
- A strong network configuration of keyframes and points to produce accurate results, that is, a well spread set of keyframes observing points with significant parallax and with plenty of loop closure matches.
- A good initial estimation of the keyframe poses and point locations for the non-linear optimization.
- A local map where to focus optimization in exploration, to achieve scalability.
- The ability to perform fast global optimizations (e.g. pose graph) to close loops in real-time.

The first real time application of BA was the visual odometry work of Mouragon et al.. [51], followed by the ground-breaking SLAM work of Klein and Murray [36], known as Parallel Tracking and Mapping (PTAM). This algorithm, while limited to small scale operation, provides simple but effective methods for keyframe selection, feature matching, point triangulation, camera localization for every frame, and relocalization after tracking failure. Unfortunately several factors severely limit its application: lack of loop closing and adequate handling of occlusions, low invariance to viewpoint of the relocalization and the need of human intervention for map bootstrapping.

In this chapter we build on the main ideas of PTAM, the place recognition work of Gálvez-López and Tardós [25], the scale-aware loop closing of Strasdat et al. [75] and the use of covisibility information for large scale operation [50, 74], to design from scratch ORB-SLAM, a novel monocular SLAM system whose main contributions are:

- Use of the same features for all tasks: tracking, mapping, relocalization and loop closing. This makes our system more efficient, simple and reliable. We use ORB features [69] which allow real-time performance without GPUs, providing good invariance to changes in viewpoint and illumination.

- Real time operation in large environments. Thanks to the use of a covisibility graph, tracking and mapping is focused in a local covisible area, independent of global map size.

- Real time loop closing based on the optimization of a pose graph that we call the *Essential Graph*. It is built from a spanning tree maintained by the system, loop closure links and strong edges from the covisibility graph.

- Real time camera relocalization with significant invariance to viewpoint and illumination. This allows recovery from tracking failure and also enhances map reuse.

- A new automatic and robust initialization procedure based on model selection that permits to create an initial map of planar and non-planar scenes.

- A *survival of the fittest* approach to map point and keyframe selection that is generous in the spawning but very restrictive in the culling. This policy improves tracking robustness, and enhances lifelong operation because redundant keyframes are discarded.

We present an extensive evaluation in popular public datasets from indoor and outdoor environments, including hand-held, car and robot sequences. To the best of our knowledge, this is the most complete and reliable solution to monocular SLAM, and for the benefit of the community we make the source code public, see Section 1.5.

4.1 System Overview

4.1.1 Feature Choice

One of the main design ideas in our system is that the same features used by the mapping and tracking are used for place recognition to perform frame-rate relocalization and loop detection. This makes our system efficient and avoids the need to interpolate the depth of the recognition features from near SLAM features as in previous works [74, 75]. We want features that need for extraction much less than 33ms per image without using GPU, which excludes the popular SIFT ($\sim$ 300ms) [45], SURF ($\sim$ 300ms) [2] or the recent A-KAZE ($\sim$ 100ms) [1]. To obtain general place recognition capabilities, we require rotation invariance, which excludes BRIEF [7] and LDB [90].

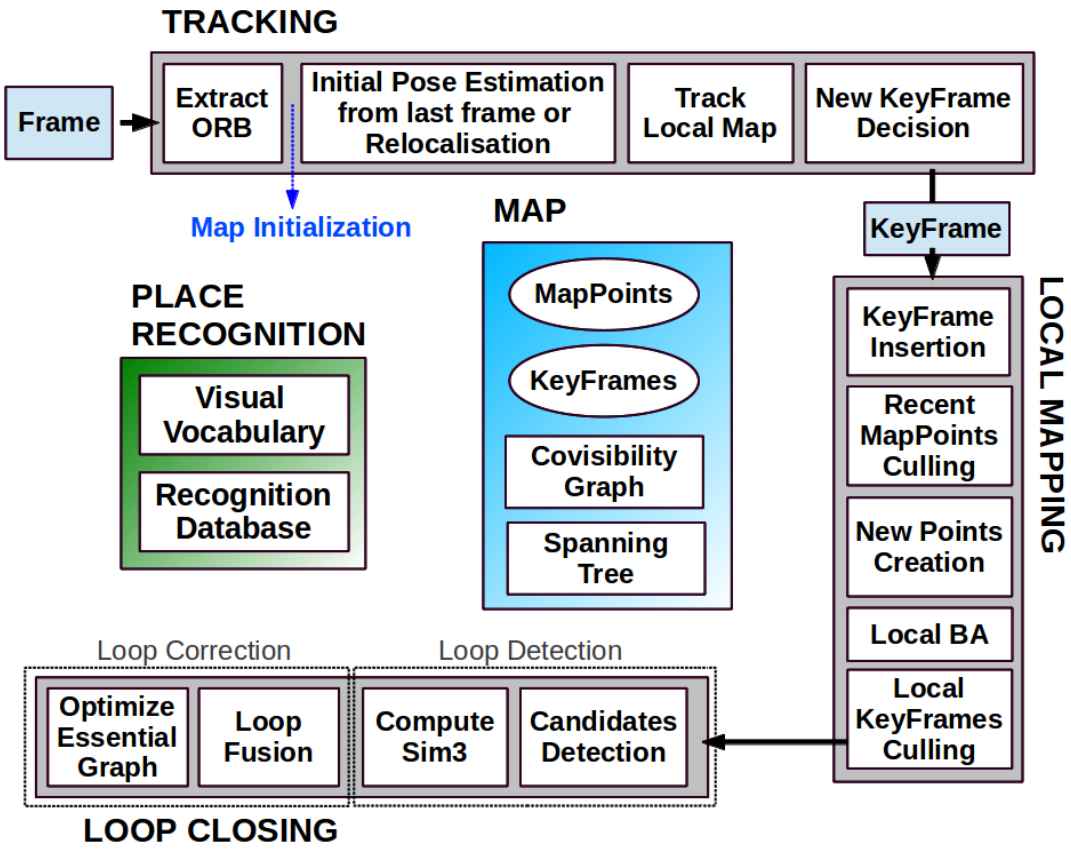
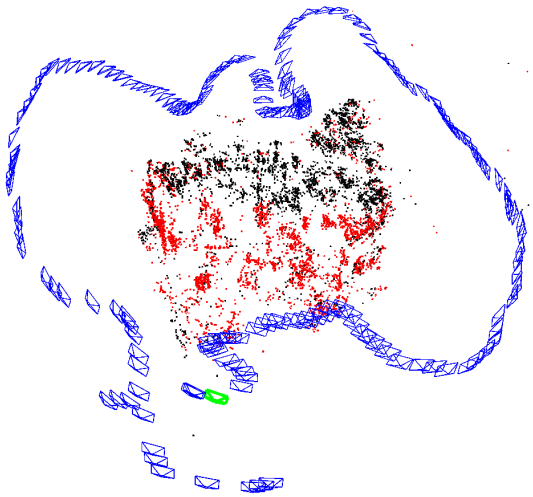


Figure 4.1: ORB-SLAM system overview, showing all the steps performed by the tracking, local mapping and loop closing threads. The main components of the place recognition module and the map are also shown.

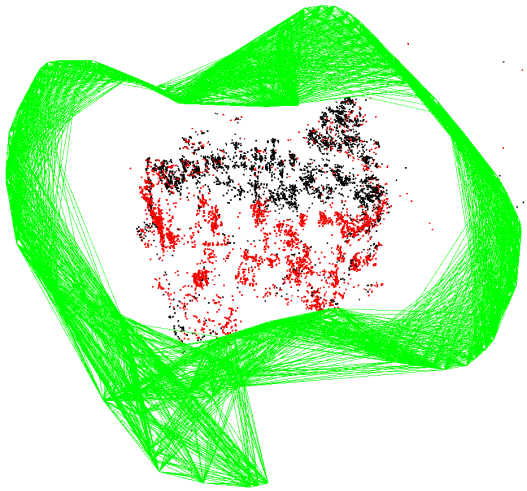
We chose ORB [69], which are oriented multi-scale FAST corners with a 256 bits descriptor associated. They are extremely fast to compute and match, while they have good invariance to viewpoint. This allows to match them from wide baselines, boosting the accuracy of BA. We already shown the good performance of ORB for place recognition in Chapter 3. While our current implementation make use of ORB, the techniques proposed are not restricted to these features.

4.1.2 System Threads

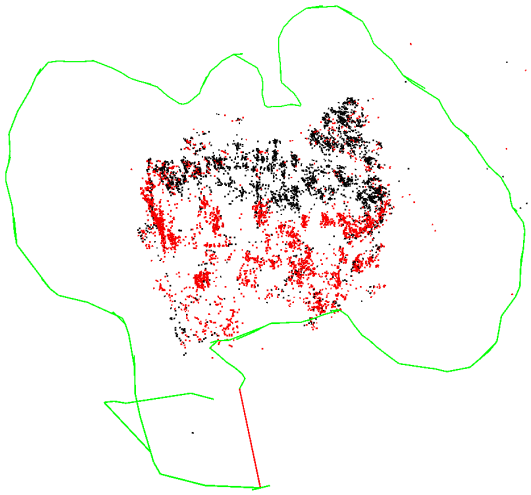
Our system, see an overview in Fig. 4.1, incorporates three threads that run in parallel: tracking, local mapping and loop closing. The tracking is in charge of localizing the camera with every frame and deciding when to insert a new keyframe. We perform first an initial feature matching with the previous frame and optimize the pose using motion-only BA. If the tracking is lost (e.g. due to occlusions or abrupt movements), the place recognition module is used to perform a global relocalization. Once there is an initial estimation of the camera pose and feature matchings, a local visible map is retrieved using the covisibility graph of keyframes that is maintained by the system, see Fig. 4.2a and Fig. 4.2b. Then matches



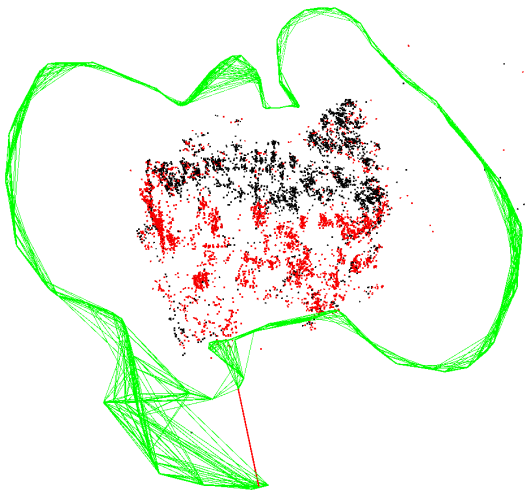
(a) KeyFrames (blue), Current Camera (green), MapPoints (black, red), Current Local MapPoints (red)



(b) Covisibility Graph



(c) Spanning Tree (green) and Loop Closure (red)



(d) Essential Graph

Figure 4.2: Reconstruction and graphs in the sequence *fr3_long_office_household* from the TUM RGB-D Benchmark [78].

with the local map points are searched by reprojection, and camera pose is optimized again with all matches. Finally the tracking thread decides if a new keyframe is inserted. All the tracking steps are explained in detail in Section 4.3. The novel procedure to create an initial map is presented in Section 4.2.

The local mapping processes new keyframes and performs local BA to achieve an optimal reconstruction in the surroundings of the camera pose. New correspondences for unmatched ORB in the new keyframe are searched in connected keyframes in the covisibility graph to triangulate new points. Some time after creation, based on the information gathered during the tracking, an exigent point culling policy is applied in order to retain only high quality

points. The local mapping is also in charge of culling redundant keyframes. We explain in detail all local mapping steps in Section 4.4.

The loop closing searches for loops with every new keyframe. If a loop is detected, we compute a similarity transformation that informs about the drift accumulated in the loop. Then both sides of the loop are aligned and duplicated points are fused. Finally a pose graph optimization over similarity constraints [75] is performed to achieve global consistency. The main novelty is that we perform the optimization over the *Essential Graph*, a sparser subgraph of the covisibility graph which is explained in Section 4.1.4. The loop detection and correction steps are explained in detail in Section 4.5.

We use the Levenberg-Marquardt algorithm implemented in g2o [38] to carry out all optimizations.

4.1.3 Map Management

Each map point p stores:

- Its 3D position $\mathbf{X}_{\mathtt{w}}$ in the world coordinate system $\mathtt{w}$.
- The viewing direction $\mathbf{n}$, which is the mean unit vector of all its viewing directions (the rays that join the point with the optical center of the keyframes that observe it).
- A representative ORB descriptor $\mathbf{D}$, which is the associated ORB descriptor whose hamming distance is minimum with respect to all other associated descriptors in the keyframes in which the point is observed.
- The maximum $d_{\max}$ and minimum $d_{\min}$ distances at which the point can be observed, according to the scale invariance limits of the ORB features.

Each keyframe K stores:

- The camera pose $\mathbf{T}_{\mathtt{cw}} \in SE(3)$, which is a rigid body transformation that transforms points from the world to the camera coordinate system.
- The camera intrinsics, including focal length and principal point.
- All the ORB features extracted in the frame, associated or not to a map point, whose coordinates are undistorted if a distortion model is provided.

Map points and keyframes are created with a generous policy, while a later very exigent culling mechanism is in charge of detecting redundant keyframes and wrongly matched or not trackable map points. This permits a flexible map expansion during exploration, which boosts tracking robustness under hard conditions (e.g. rotations, fast movements), while its size is bounded in continual revisits to the same environment, i.e. lifelong operation. Additionally our maps contain very few outliers compared to PTAM, at the expense of containing less points. Culling procedures of map points and keyframes are explained in Sections 4.4.2 and 4.4.5 respectively.

4.1.4 Covisibility Graph and Essential Graph

Covisibility information between keyframes is very useful in several tasks of our system, and is represented as an undirected weighted graph as in [74]. Each node is a keyframe and an edge between two keyframes exists if they share observations of the same map points (at least 15), being the weight θ of the edge the number of common map points.

In order to correct a loop we perform a pose graph optimization [75] that distributes the loop closing error along the graph. In order not to include all the edges provided by the covisibility graph, which can be very dense, we propose to build an *Essential Graph* that retains all the nodes (keyframes), but less edges, still preserving a strong network that yields accurate results. The system builds incrementally a spanning tree from the initial keyframe, which provides a connected subgraph of the covisibility graph with minimal number of edges. When a new keyframe is inserted, it is included in the tree linked to the keyframe which shares most point observations, and when a keyframe is erased by the culling policy, the system updates the links affected by that keyframe. The *Essential Graph* contains the spanning tree, the subset of edges from the covisibility graph with high covisibility ($\theta_{\min} = 100$), and the loop closure edges, resulting in a strong network of cameras. Fig. 4.2 shows an example of a covisibility graph, spanning tree and associated essential graph. As shown in the experiments of Section 4.6.5, when performing the pose graph optimization, the solution is so accurate that an additional full bundle adjustment optimization barely improves the solution. The efficiency of the essential graph and the influence of the $\theta_{\min}$ is explored at the end of Section 4.6.5.

4.1.5 Bags of Words Place Recognition

The system has embedded a bags of words place recognition module, based on DBoW2 [25], to perform loop detection and relocalization. Visual words are just a discretization of the descriptor space, which is known as the visual vocabulary. The vocabulary is created offline with the ORB descriptors extracted from a large set of images. If the images are general enough, the same vocabulary can be used for different environments getting a good performance, as shown in Chapter 3. The system builds incrementally a database that contains an invert index, which stores for each visual word in the vocabulary, in which keyframes it has been seen, so that querying the database can be done very efficiently. The database is also updated when a keyframe is deleted by the culling procedure.

Because there exists visual overlap between keyframes, when querying the database there will not exist a unique keyframe with a high score. The original DBoW2 took this overlapping into account, adding up the score of images that are close in time. This has the limitation of not including keyframes viewing the same place but inserted at a different time. Instead we group those keyframes that are connected in the covisibility graph. In addition our database returns all keyframe matches whose scores are higher than the 75% of the best score.

An additional benefit of the bags of words representation for feature matching was reported in [25]. When we want to compute the correspondences between two sets of ORB features, we can constraint the brute force matching only to those features that belong to the same node in the vocabulary tree at a certain level (we select the second out of six), speeding

up the search. We use this *trick* when searching matches for triangulating new points, and at loop detection and relocalization. We also refine the correspondences with the orientation consistency test, explained in Section 3.2, that discards outliers ensuring a coherent rotation for all correspondences.

4.2 Automatic Map Initialization

The goal of the map initialization is to compute the relative pose between two frames to triangulate an initial set of map points. This method should be independent of the scene (planar or general) and should not require human intervention to select a good two-view configuration, i.e. a configuration with significant parallax. We propose to compute in parallel two geometrical models, a homography assuming a planar scene and a fundamental matrix assuming a non-planar scene. We then use a heuristic to select a model and try to recover the relative pose with a specific method for the selected model. Our method only initializes when it is certain that the two-view configuration is safe, delaying the initialization if detecting low-parallax or the well-known twofold planar ambiguity [43]. The steps of our algorithm are:

1. Find initial correspondences:

   Extract ORB features (only at the finest scale) in the current frame F_c and search for matches $\mathbf{x}_c \leftrightarrow \mathbf{x}_r$ in the reference frame F_r. If not enough matches are found, reset the reference frame.

2. Parallel computation of the two models:

   Compute in parallel threads a homography $\mathbf{H}_{cr}$ and a fundamental matrix $\mathbf{F}_{cr}$:

   $$\mathbf{x}_c = \mathbf{H}_{cr}\,\mathbf{x}_r \qquad\qquad \mathbf{x}_c^T\,\mathbf{F}_{cr}\,\mathbf{x}_r = 0 \tag{4.1}$$

   with the normalized DLT and 8-point algorithms respectively as explained in [28] inside a RANSAC scheme. To make the procedure homogeneous for both models, the number of iterations is prefixed and the same for both models, along with the points to be used at each iteration, 8 for the fundamental matrix, and 4 of them for the homography. At each iteration we compute a score S_M for each model M (H for the homography, F for the fundamental matrix):

   $$\begin{aligned} S_M &= \sum_i \left(\rho_M\left(d^2_{cr}(\mathbf{x}^i_c, \mathbf{x}^i_r, M)\right) + \rho_M(d^2_{rc}\left(\mathbf{x}^i_c, \mathbf{x}^i_r, M\right))\right) \\ \rho_M(d^2) &= \begin{cases} \Gamma - d^2 & \text{if} \quad d^2 < T_M \\ 0 & \text{if} \quad d^2 \geq T_M \end{cases} \end{aligned} \tag{4.2}$$

   where d^2_{cr} and d^2_{rc} are the symmetric transfer errors [28] from one frame to the other. T_M is the outlier rejection threshold based on the χ^2 test at 95% ($T_H = 5.99$, $T_F = 3.84$, assuming a standard deviation of 1 pixel in the measurement error). Γ is defined equal

to T_H so that both models score equally for the same d in their inlier region, again to make the process homogeneous.

We keep the homography and fundamental matrix with highest score. If no model could be found (not enough inliers), we restart the process again from step 1.

3. Model selection:

   If the scene is planar, nearly planar or there is low parallax, it can be explained by a homography. However a fundamental matrix can also be found, but the problem is not well constrained [28] and any attempt to recover the motion from the fundamental matrix would yield wrong results. We should select the homography as the reconstruction method will correctly initialize from a plane or it will detect the low parallax case and refuse the initialization. On the other hand a non-planar scene with enough parallax can only be explained by the fundamental matrix, but a homography can also be found explaining a subset of the matches if they lie on a plane or they have low parallax (they are far away). In this case we should select the fundamental matrix. We have found that a robust heuristic is to compute:

$$R_H = \frac{S_H}{S_H + S_F} \tag{4.3}$$

   and select the homography if $R_H > 0.45$, which adequately captures the planar and low parallax cases. Otherwise, we select the fundamental matrix.

4. Motion and Structure from Motion recovery:

   Once a model is selected we retrieve the motion hypotheses associated. In the case of the homography we retrieve 8 motion hypotheses using the method of Faugeras et al. [20]. The method proposes cheriality tests to select the valid solution. However these tests fail if there is low parallax as points easily go in front or back of the cameras, which could yield the selection of a wrong solution. We propose to directly triangulate the eight solutions, and check if there is one solution with most points seen with parallax, in front of both cameras and with low reprojection error. If there is not a clear winner solution, we do not initialize and continue from step 1. This technique to disambiguate the solutions makes our initialization robust under low parallax and the twofold ambiguity configuration, and could be considered the key of the robustness of our method.

   In the case of the fundamental matrix, we convert it in an essential matrix using the calibration matrix $\mathbf{K}$:

$$\mathbf{E}_{rc} = \mathbf{K}^T \mathbf{F}_{rc} \mathbf{K} \tag{4.4}$$

   and then retrieve 4 motion hypotheses with the singular value decomposition method explained in [28]. We triangulate the four solutions and select the reconstruction as done for the homography.

5. Bundle adjustment:

   Finally we perform bundle adjustment to refine the initial reconstruction.

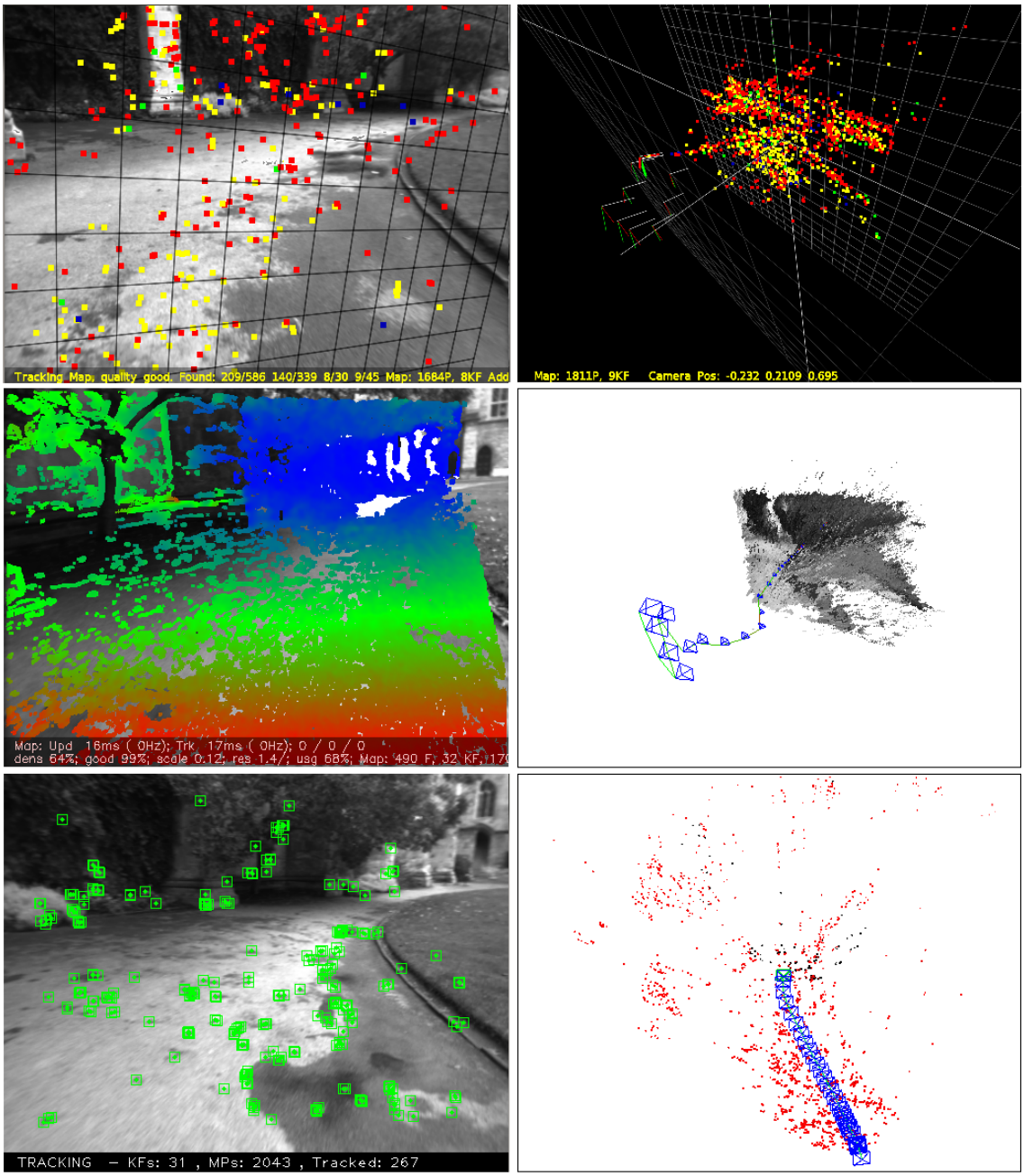


Figure 4.3: PTAM (top), LSD-SLAM (middle) and ORB-SLAM (bottom) after initialization in NewCollege[71]. PTAM and LSD-SLAM initialized a wrong planar solution while ORB-SLAM automatically initialized from the fundamental matrix when detected enough parallax. Depending on which keyframes are manually selected, PTAM is also able to initialize well.

An example of a challenging initialization in the outdoor NewCollege robot sequence [71] is shown in Fig. 4.3. It can be seen how PTAM and LSD-SLAM have wrongly initialized all points in a plane, while our method has automatically waited until there is enough parallax, initializing correctly from the fundamental matrix.

4.3 Tracking

In this section we describe the steps of the tracking thread that are performed with every frame. The camera pose optimizations, mentioned in several steps, consist in motion-only BA, a variant of the generic BA (1.5), where the camera pose $\mathbf{T}_{\mathtt{cw}} = [\mathbf{R}_{\mathtt{cw}}|_{\mathtt{c}}\mathbf{p}_{\mathtt{w}}]$ is optimized to minimize the reprojection error between 3D points $\mathbf{X}_{\mathtt{w}}$ matched to 2D keypoints $\mathbf{x}_c$:

$$\{\mathbf{R}_{\mathtt{cw}}, {}_{\mathtt{c}}\mathbf{p}_{\mathtt{w}}\} = \underset{\mathbf{R}_{\mathtt{cw}}, {}_{\mathtt{c}}\mathbf{p}_{\mathtt{w}}}{\operatorname{argmin}} \sum_j \rho\left(\left\|\mathbf{x}_{\mathtt{c}}^j - \pi_m\left(\mathbf{R}_{\mathtt{cw}}\mathbf{X}_{\mathtt{w}}^j + {}_{\mathtt{c}}\mathbf{p}_{\mathtt{w}}\right)\right\|^2_{\mathbf{\Sigma}_{\mathtt{j}}}\right) \tag{4.5}$$

4.3.1 ORB Extraction

We extract FAST corners at 8 scale levels with a scale factor of 1.2. For image resolutions from 512×384 to 752×480 pixels we found suitable to extract 1000 corners, for higher resolutions, as the 1241×376 in the KITTI dataset [26] we extract 2000 corners. In order to ensure an homogeneous distribution we divide each scale level in a grid, trying to extract at least 5 corners per cell. Then we detect corners in each cell, adapting the detector threshold if not enough corners are found. The amount of corners retained per cell is also adapted if some cells contains no corners (textureless or low contrast). The orientation and ORB descriptor are then computed on the retained FAST corners. The ORB descriptor is used in all feature matching, in contrast to the search by patch correlation in PTAM.

4.3.2 Initial Pose Estimation from Previous Frame

If tracking was successful for the last frame, we use a constant velocity motion model to predict the camera pose and perform a guided search of the map points observed in the last frame. If not enough matches were found (i.e. motion model is clearly violated), we use a wider search of the map points around their position in the last frame. The pose is then optimized with the found correspondences.

4.3.3 Initial Pose Estimation via Global Relocalization

If the tracking is lost, we convert the frame into bag of words and query the recognition database for keyframe candidates for global relocalization. We compute correspondences with ORB associated to map points in each keyframe, as explained in section 4.1.5. We then perform alternatively RANSAC iterations for each keyframe and try to find a camera pose using the PnP algorithm [40]. If we find a camera pose with enough inliers, we optimize the pose and perform a guided search of more matches with the map points of the candidate keyframe. Finally the camera pose is again optimized, and if supported with enough inliers, tracking procedure continues.

4.3.4 Track Local Map

Once we have an estimation of the camera pose and an initial set of feature matches, we can project the map into the frame and search more map point correspondences. To bound the

complexity in large maps, we only project a local map. This local map contains the set of keyframes $\mathcal{K}_1$, that share map points with the current frame, and a set $\mathcal{K}_2$ with neighbors to the keyframes $\mathcal{K}_1$ in the covisibility graph. The local map also has a reference keyframe $K_{\text{ref}} \in \mathcal{K}_1$ which shares most map points with the current frame. Now each map point seen in $\mathcal{K}_1$ and $\mathcal{K}_2$ is searched in the current frame as follows:

1. Compute the map point projection $\mathbf{x}$ in the current frame. Discard if it is outside the image bounds.

2. Compute the angle between the current viewing ray $\mathbf{v}$ and the map point mean viewing direction $\mathbf{n}$. Discard if $\mathbf{v} \cdot \mathbf{n} < \cos(60^\circ)$.

3. Compute the distance d from map point to camera center. Discard if it is out of the scale invariance region of the map point $d \notin [d_{\min}, d_{\max}]$.

4. Compute the scale in the frame by the ratio $d/d_{\min}$.

5. Compare the representative descriptor $\mathbf{D}$ of the map point with the still unmatched ORB features in the frame, at the predicted scale, and near $\mathbf{x}$, and associate the map point with the best match.

The camera pose is finally optimized with all the map points found in the frame.

4.3.5 New Keyframe Decision

The last step is to decide if the current frame is spawned as a new keyframe. As there is a mechanism in the local mapping to cull redundant keyframes, we will try to insert keyframes as fast as possible, because that makes the tracking more robust to challenging camera movements, typically rotations. To insert a new keyframe all the following conditions must be met:

1. More than 20 frames must have passed from the last global relocalization.

2. Local mapping is idle, or more than 20 frames have passed from last keyframe insertion.

3. Current frame tracks at least 50 points.

4. Current frame tracks less than 90% points than K_{ref}.

Instead of using a distance criterion to other keyframes as PTAM, we impose a minimum visual change (condition 4). Condition 1 ensures a good relocalization and condition 3 a good tracking. If a keyframe is inserted when the local mapping is busy (second part of condition 2), a signal is sent to stop local bundle adjustment, so that it can process as soon as possible the new keyframe.

4.4 Local Mapping

In this section we describe the steps performed by the local mapping with every new keyframe K_i.

4.4.1 Keyframe Insertion

At first we update the covisibility graph, adding a new node for K_i and updating the edges resulting from the shared map points with other keyframes. We then update the spanning tree linking K_i with the keyframe with most points in common. We then compute the bags of words representation of the keyframe, that will help in the data association for triangulating new points.

4.4.2 Recent Map Point Culling

Map points, in order to be retained in the map, must pass a restrictive test during the first three keyframes after creation, that ensures that they are trackable and not wrongly triangulated, i.e due to spurious data association. A point must fulfill these two conditions:

1. The tracking must find the point in more than the 25% of the frames in which it is predicted to be visible.

2. If more than one keyframe has passed from map point creation, it must be observed from at least three keyframes.

Once a map point have passed this test, it can only be removed if at any time it is observed from less than three keyframes. This can happen when keyframes are culled and when local bundle adjustment discards outlier observations. This policy makes our map contain very few outliers.

4.4.3 New Map Point Creation

New map points are created by triangulating ORB from connected keyframes $\mathcal{K}_c$ in the covisibility graph. For each unmatched ORB in K_i we search a match with other unmatched point in other keyframe. This matching is done as explained in Section 4.1.5 and discard those matches that do not fulfill the epipolar constraint. ORB pairs are triangulated, and to accept the new points, positive depth in both cameras, parallax, reprojection error and scale consistency are checked. Initially a map point is observed from two keyframes but it could be matched in others, so it is projected in the rest of connected keyframes, and correspondences are searched as detailed in section 4.3.4.

4.4.4 Local Bundle Adjustment

The local BA optimizes a local window of keyframes $\mathcal{K}_\mathcal{L}$ formed by the currently processed keyframe K_i and all the keyframes connected to it in the covisibility graph $\mathcal{K}_c$, and all the

map points $\mathcal{P}_\mathcal{L}$ seen by those keyframes. All other keyframes $\mathcal{K}_\mathcal{F}$ that see those points but are not connected to the currently processed keyframe are included in the optimization but remain fixed. The local BA optimization is a special case of the generic BA (1.5):

$$\{\mathbf{X}^p_{\mathtt{W}}, \mathbf{R}^l_{\mathtt{CW}}, {}_{\mathtt{C}}\mathbf{p}^l_{\mathtt{W}} | p \in \mathcal{P}_L, l \in \mathcal{K}_L\} = \underset{\mathbf{X}^p_{\mathtt{W}}, \mathbf{R}^l_{\mathtt{CW}}, {}_{\mathtt{C}}\mathbf{p}^l_{\mathtt{W}}}{\operatorname{argmin}} \sum_{k \in \mathcal{K}_L \cup \mathcal{K}_F} \sum_{j \in \mathcal{P}_k} \rho \left(\left\| \mathbf{x}^j_k - \pi_m \left(\mathbf{R}^k_{\mathtt{CW}} \mathbf{X}^j_{\mathtt{W}} + {}_{\mathtt{C}}\mathbf{p}^k_{\mathtt{W}} \right) \right\|^2_{\mathbf{\Sigma}^j_k} \right) \tag{4.6}$$

where $\mathbf{\Sigma}^j_k$ is the covariance of the position of the keypoint matched to 3D point $\mathbf{X}^j$ in keyframe K_k, which depends on the scale at which the keypoint was detected. $\mathcal{P}_k \subset \mathcal{P}_\mathcal{L}$ is the set of points observed in keyframe k. Observations that are marked as outliers by the Huber cost function ρ are discarded at the middle and at the end of the optimization.

4.4.5 Local Keyframe Culling

In order to maintain a compact reconstruction, the local mapping tries to detect redundant keyframes and delete them. This is beneficial as bundle adjustment complexity grows with the number of keyframes, but also because it enables lifelong operation in the same environment as the number of keyframes will not grow unbounded, unless the visual content in the scene changes. We discard all the keyframes in $\mathcal{K}_c$ whose 90% of the map points have been seen in at least other three keyframes in the same or finer scale. The scale condition ensures that map points maintain keyframes from which they are measured with most accuracy. This policy was inspired by the one proposed in the work of Tan et al. [81], where keyframes were discarded after a process of change detection.

4.5 Loop Closing

The loop closing thread takes K_i, the last keyframe processed by the local mapping, and tries to detect and close loops. The steps are next described.

4.5.1 Loop Detection

At first we compute the similarity between the bag of words vector of K_i and all its neighbors in the covisibility graph ($\theta_{min} = 30$) and retain the lowest score $s_{\min}$. Then we query the recognition database and discard all those keyframes whose score is lower than $s_{\min}$. This is a similar operation to gain robustness as the normalizing score in DBoW2, which is computed from the previous image, but here we use covisibility information. In addition all those keyframes directly connected to K_i are discarded from the results. To accept a loop candidate we must detect consecutively three loop candidates that are consistent (keyframes connected in the covisibility graph). There can be several loop candidates if there are several places with similar appearance to K_i.

4.5.2 Similarity Transformation

In monocular SLAM there are seven degrees of freedom in which the map can drift, three translations, three rotations and a scale factor [75]. Therefore to compute the error accumulated in the loop trajectory we need to estimate a similarity transformation $\mathbf{S}_{\mathsf{CC}}^{il}$ from the current keyframe K_i to the loop keyframe K_l. A similarity transformation $\mathbf{S}$ has the following structure:

$$\mathbf{S} = \begin{bmatrix} s\mathbf{R} & \mathbf{p} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{4.7}$$

where $s \in \mathbb{R}^+$ is a scale factor, $\mathbf{R} \in SO(3)$ is a rotation matrix and $\mathbf{p} \in \mathbb{R}^3$ is a translation vector. We use the computation of this similarity as geometrical validation of the loop.

We first compute correspondences between ORB associated to map points in the current keyframe and the loop candidate keyframes, following the procedure explained in section 4.1.5. At this point we have 3D to 3D correspondences for each loop candidate. We alternatively perform RANSAC iterations with each candidate, trying to find a similarity transformation using the method of Horn [29]. If we find a similarity $\mathbf{S}_{\mathsf{CC}}^{il} = [s_{il}\mathbf{R}_{il}|_i\mathbf{p}_l]$ with enough inliers, we optimize it by minimizing the reprojection error on both images:

$$s_{il}, \mathbf{R}_{il}, {}_i\mathbf{p}_l = \underset{s_{il},\mathbf{R}_{il},{}_i\mathbf{p}_l}{\operatorname{argmin}}$$
$$\sum_{j=1}^{n} \left(\rho\left(\left\| \mathbf{x}_i^j - \pi_m\left(s_{il}\mathbf{R}_{il}\mathbf{X}_l^j + {}_i\mathbf{p}_l \right) \right\|_{\mathbf{\Sigma}_i^j}^2 \right) + \rho\left(\left\| \mathbf{x}_l^j - \pi_m\left(\frac{1}{s_{il}}\mathbf{R}_{il}^T\left(\mathbf{X}_i^j - {}_i\mathbf{p}_l \right) \right) \right\|_{\mathbf{\Sigma}_l^j}^2 \right) \right) \tag{4.8}$$

where n is the number of matched ORB, ρ is the Huber function, $\mathbf{X}_i^j$ and $\mathbf{X}_l^j$ are the 3D position of matched map points in their respective camera coordinates, $\mathbf{x}$ is the keypoint location in the image, $\mathbf{\Sigma}$ is the covariance of the keypoint, and π_m is the projection function (1.1). After the optimization we perform a guided search of more correspondences. We optimize it again and, if $\mathbf{S}_{\mathsf{CC}}^{il}$ is supported by enough inliers, the loop with K_l is accepted.

4.5.3 Loop Fusion

The first step in the loop correction is to fuse duplicated map points and insert new edges in the covisibility graph that will attach the loop closure. At first the current keyframe pose is corrected with the estimated similarity transformation and this correction is propagated to all the neighbors of K_i, concatenating transformations, so that both sides of the loop get aligned. All map points seen by the loop keyframe and its neighbors are projected into K_i and its neighbors and matches are searched in a narrow area around the projection, as done in Section 4.3.4. All those map points matched and those that were inliers in the computation of the similarity transformation are fused. All keyframes involved in the fusion will update their edges in the covisibility graph effectively creating edges that attach the loop closure.

4.5.4 Essential Graph Optimization

To effectively close the loop, we perform a pose graph optimization over the *Essential Graph*, described in Section 4.1.4, that distributes the loop closing error along the graph. The

optimization is performed over similarity transformations to correct the scale drift [75]. The error terms and cost function are detailed in Section 1.3.3. After the optimization each map point is transformed according to the correction of one of the keyframes that observes it.

4.6 Experiments

We have performed an extensive experimental validation of our system in the large robot sequence of NewCollege [71], evaluating the general performance of the system, in 16 hand-held indoor sequences of the TUM RGB-D benchmark [78], evaluating the localization accuracy, relocalization and lifelong capabilities, and in 10 car outdoor sequences from the KITTI dataset [26], evaluating real-time large scale operation, localization accuracy and efficiency of the pose graph optimization.

Our system runs in real time and processes the images exactly at the frame rate they were acquired. We have carried out all experiments with an Intel Core i7-4700MQ (4 cores @ 2.40GHz) and 8Gb RAM. ORB-SLAM has three main threads, that run in parallel with other tasks from the operating system, which introduces some randomness in the results. For this reason, in some experiments, we report the median from several runs.

4.6.1 System Performance in the NewCollege Dataset

The NewCollege dataset [71] contains a 2.2km sequence from a robot traversing a campus and adjacent parks. The sequence is recorded by a stereo camera at 20 fps and a resolution of 512×382 pixels. It contains several loops and fast rotations that makes the sequence quite challenging for monocular vision. To the best of our knowledge there is no other monocular system in the literature able to process this whole sequence. For example Strasdat et al. [74], despite being able to close loops and work in large scale environments, only showed monocular results for a small part of this sequence.

As an example of our loop closing procedure we show in Fig. 4.4 the detection of a loop with the inliers that support the similarity transformation.

Fig. 4.5 shows the reconstruction before and after the loop closure. The local map is shown in red, which after the loop closure extends along both sides of the loop closure. The whole map after processing the full sequence at its real frame-rate is shown in Fig. 4.6. The

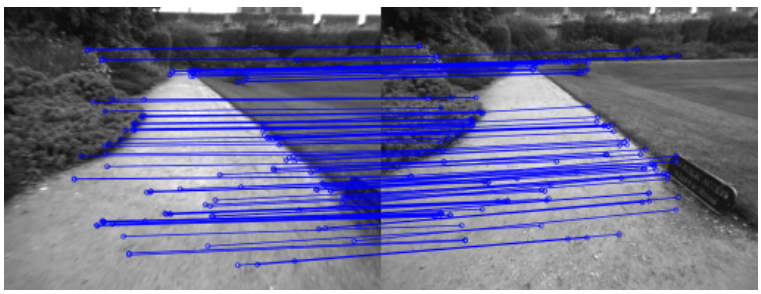


Figure 4.4: Example of loop detected in the NewCollege sequence. We draw the inlier correspondences supporting the similarity transformation found.

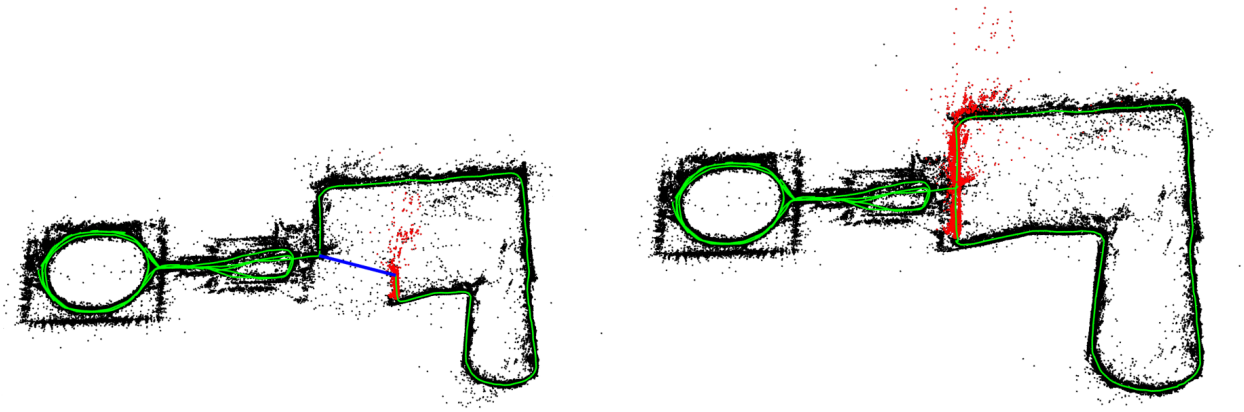


Figure 4.5: Map before and after a loop closure in the NewCollege sequence. The loop closure match is drawn in blue, the trajectory in green, and the local map for the tracking at that moment in red. The local map is extended along both sides of the loop after it is closed.

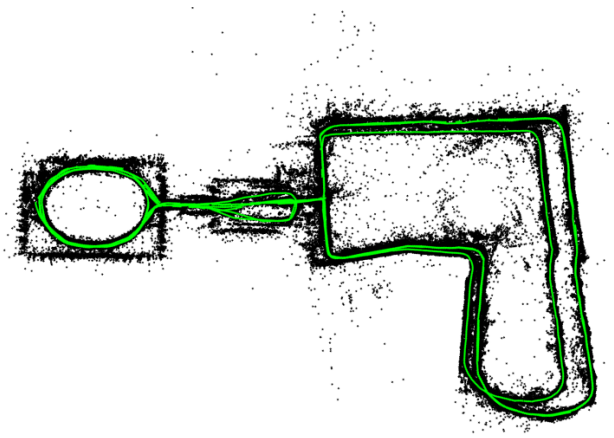


Figure 4.6: Reconstruction of the full sequence of NewCollege. The bigger loop on the right is traversed in opposite directions and no visual loop closures were found.

big loop on the right does not perfectly align because it was traversed in opposite directions and the place recognizer was not able to find loop closures.

We have extracted statistics of the times spent by each thread in this experiment. Table 4.1 shows the results for the tracking and the local mapping. Tracking works at frame-rates around 25-30Hz, being the most demanding task to track the local map. If needed this time could be reduced limiting the number of keyframes that are included in the local map. In the local mapping thread the most demanding task is local bundle adjustment. The local BA time varies if the robot is exploring or in a well mapped area, because during exploration bundle adjustment is interrupted if tracking inserts a new keyframe, as explained in section 4.3.5. In case of not needing new keyframes local bundle adjustment performs a generous number of prefixed iterations.

Table 4.2 shows the results for each of the 6 loop closures found. It can be seen how the

Table 4.1: Tracking and Mapping times in NewCollege

Thread	Operation	Median (ms)	Mean (ms)	Std (ms)
TRACKING	ORB extraction	11.10	11.42	1.61
	Initial Pose Est.	3.38	3.45	0.99
	Track Local Map	14.84	16.01	9.98
	Total	30.57	31.60	10.39
LOCAL MAPPING	KeyFrame Insertion	10.29	11.88	5.03
	Map Point Culling	0.10	3.18	6.70
	Map Point Creation	66.79	72.96	31.48
	Local BA	296.08	360.41	171.11
	KeyFrame Culling	8.07	15.79	18.98
	Total	383.59	464.27	217.89

Table 4.2: Loop closing times in NewCollege

			Loop Detection (ms)		Loop Correction (s)		
Loop	KeyFrames	Essential Graph Edges	Candidates Detection	Similarity Transf.	Fusion	Essential Graph Optim.	Total (s)
1	287	1347	4.71	20.77	0.20	0.26	0.51
2	1082	5950	4.14	17.98	0.39	1.06	1.52
3	1279	7128	9.82	31.29	0.95	1.26	2.27
4	2648	12547	12.37	30.36	0.97	2.30	3.33
5	3150	16033	14.71	41.28	1.73	2.80	4.60
6	4496	21797	13.52	48.68	0.97	3.62	4.69

loop detection increases sublinearly with the number of keyframes. This is due to the efficient querying of the database that only compare the subset of images with words in common, which demonstrates the potential of bag of words for place recognition. Our *Essential Graph* includes edges around 5 times the number of keyframes, which is a quite sparse graph.

4.6.2 Localization Accuracy in the TUM RGB-D Dataset

The TUM RGB-D dataset [78] is excellent to evaluate the accuracy of camera localization as it provides several sequences with accurate ground truth obtained with an external motion capture system. We have discarded all those sequences that we consider that are not suitable for pure monocular SLAM systems, as they contain strong rotations, no texture or no motion.

For comparison we have also executed LSD-SLAM [17] and PTAM [36]. We compare also with the trajectories generated by RGBD-SLAM [15] which are provided for some of the sequences in the website of the dataset. In order to compare ORB-SLAM, LSD-SLAM and PTAM with the ground truth, we align the keyframe trajectories using a similarity transformation, as scale is unknown, and measure the absolute trajectory error (ATE) [78]. In the case of RGBD-SLAM we align the trajectories with a rigid body transformation, but also a similarity to check if the scale was well recovered. LSD-SLAM initializes from random depth values and takes time to converge, therefore we have discarded the first 10 keyframes when comparing to the ground truth. For PTAM we manually selected two frames from which we get a good initialization. Table 4.3 shows the median results over 5 executions in each of the 16 selected sequences. Trajectories for RGBD-SLAM are aligned with 6DoF and 7DoF (results between brackets). X means that the tracking is lost at some point and a significant portion of the sequence is not processed by the system.

Table 4.3: Keyframe Localization Error Comparison in the TUM RGB-D Dataset

	Absolute KeyFrame Trajectory RMSE (cm)			
	ORB-SLAM	PTAM	LSD-SLAM	RGBD-SLAM
fr1_xyz	**0.90**	1.15	9.00	1.34 (1.34)
fr2_xyz	0.30	**0.20**	2.15	2.61 (1.42)
fr1_floor	**2.99**	X	38.07	3.51 (3.51)
fr1_desk	**1.69**	X	10.65	2.58 (2.52)
fr2_360_kidnap	3.81	**2.63**	X	393.3 (100.5)
fr2_desk	**0.88**	X	4.57	9.50 (3.94)
fr3_long_office	**3.45**	X	38.53	-
fr3_nstr_tex_far	planar ambiguity	4.92 / 34.74	18.31	-
fr3_nstr_tex_near	**1.39**	2.74	7.54	-
fr3_str_tex_far	**0.77**	0.93	7.95	-
fr3_str_tex_near	1.58	**1.04**	X	-
fr2_desk_person	**0.63**	X	31.73	6.97 (2.00)
fr3_sit_xyz	**0.79**	0.83	7.73	-
fr3_sit_halfsph	**1.34**	X	5.87	-
fr3_walk_xyz	**1.24**	X	12.44	-
fr3_walk_halfsph	**1.74**	X	X	-

It can be seen that ORB-SLAM is able to process all the sequences, except the sequence

fr3_nostructure_texture_far (fr3_nstr_tex_far). This is a planar scene that because of the camera trajectory with respect to the plane has two possible interpretations, i.e. the twofold ambiguity described in [43]. Our initialization method detects the ambiguity and for safety refuses to initialize. PTAM initializes selecting sometimes the true solution and others the corrupted one, in which case the error is unacceptable. We have not noticed two different reconstructions from LSD-SLAM but the error in this sequence is very high. In the rest of the sequences, PTAM and LSD-SLAM exhibit less robustness than our method, loosing track in eight and three sequences respectively. In terms of accuracy ORB-SLAM and PTAM are similar in open trajectories, while ORB-SLAM achieves higher accuracy when detecting large loops as in *fr3_nostructure_texture_near_withloop* (fr3_nstr_tex_near). The most surprising results is that both PTAM and ORB-SLAM are clearly more accurate than LSD-SLAM and RGBD-SLAM. One of the possible causes can be that they reduce the map optimization to a pose-graph optimization were sensor measurements are discarded, while we perform bundle adjustment and jointly optimize cameras and map over sensor measurements, which is the gold standard algorithm to solve structure from motion [28]. We further discuss this result in Section 4.7.2. Another interesting result is that LSD-SLAM seems to be less robust to dynamic objects than our system as seen in *fr2_desk_with_person* and *fr3_walking_xyz*.

We have noticed that RGBD-SLAM has a bias in the scale in *fr2* sequences, as aligning the trajectories with 7 DoF significantly reduces the error. Finally it should be noted that Engel et al. [17] reported that PTAM has less accuracy than LSD-SLAM in *fr2_xyz* with an RMSE of 24.28cm. However, the paper does not give enough details on how those results were obtained, and we have been unable to reproduce them.

4.6.3 Relocalization in the TUM RGB-D Dataset

We perform two relocalization experiments in the TUM RGB-D dataset. In the first experiment we build a map with the first 30 seconds of the sequence *fr2_xyz* and perform global relocalization with every successive frame and evaluate the accuracy of the recovered poses. We perform the same experiment with PTAM for comparison. Fig. 4.7 shows the keyframes used to create the initial map, the poses of the relocalized frames and the ground truth for those frames. It can be seen that PTAM is only able to relocalize frames which are near to the keyframes due to the little invariance of its relocalization method. Table 4.4 shows the recall and the error with respect to the ground truth. ORB-SLAM accurately relocalizes more than the double of frames than PTAM. In the second experiment we create an initial map with sequence *fr3_sitting_xyz* and try to relocalize all frames from *fr3_walking_xyz*. This is a challenging experiment as there are big occlusions due to people moving in the scene. Here PTAM finds no relocalizations while our system relocalizes 78% of the frames, as can be seen in Table 4.4. Fig. 4.8 shows some examples of challenging relocalizations performed by our system in these experiments.

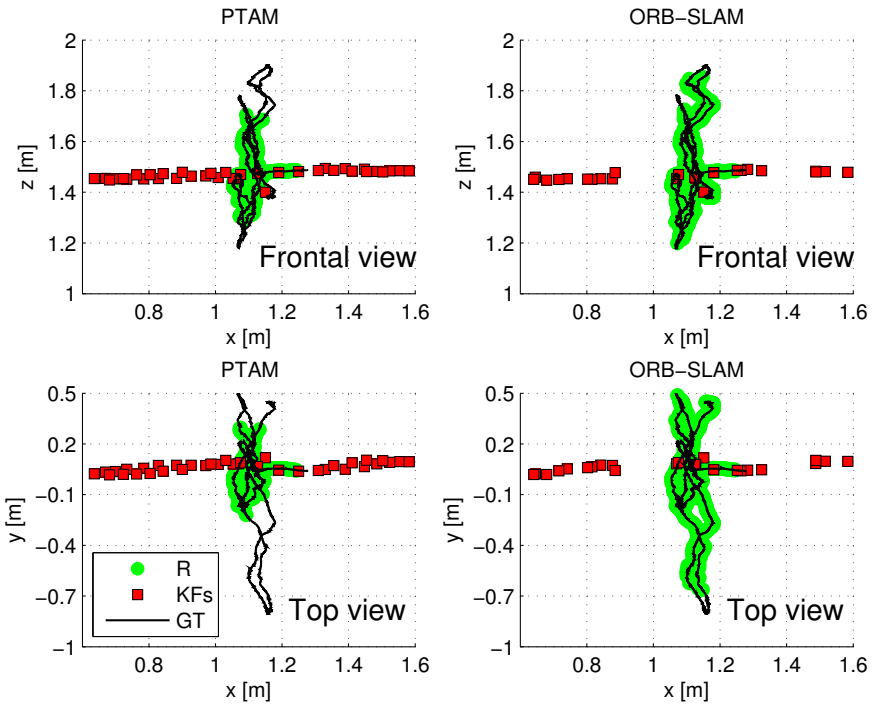


Figure 4.7: Relocalization experiment in *fr2_xyz*. Map is initially created during the first 30 seconds of the sequence (KFs). The goal is to relocalize subsequent frames. Successful relocalizations (R) of our system and PTAM are shown. The ground truth (GT) is only shown for the frames to relocalize.

Table 4.4: Results for the relocalization experiments

	Initial Map		Relocalization		
System	KFs	RMSE (cm)	Recall (%)	RMSE (cm)	Max. Error (cm)
fr2_xyz. 2769 frames to relocalize					
PTAM	37	0.19	34.9	0.26	1.52
ORB-SLAM	24	0.19	**78.4**	0.38	1.67
fr3_walking_xyz. 859 frames to relocalize					
PTAM	34	0.83	0.0	-	-
ORB-SLAM	31	0.82	**77.9**	1.32	4.95

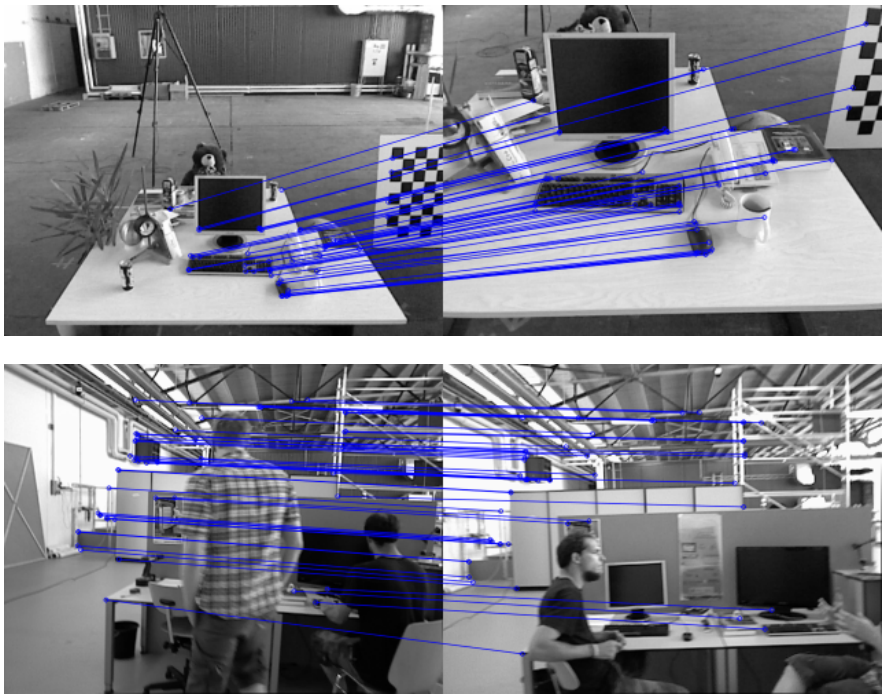


Figure 4.8: Example of challenging relocalizations (severe scale change, dynamic objects) that our system successfully found in the relocalization experiments.

4.6.4 Lifelong Experiment in the TUM RGB-D Dataset

Previous relocalization experiments have shown that our system is able to localize in a map from very different viewpoints and robustly under moderate dynamic changes. This property in conjunction with our keyframe culling procedure allows to operate lifelong in the same environment under different viewpoints and some dynamic changes.

In the case of a completely static scenario our system is able to maintain the number of keyframes bounded even if the camera is looking at the scene from different viewpoints. We demonstrate it in a custom sequence were the camera is looking at the same desk during 93 seconds but performing a trajectory so that the viewpoint is always changing. We compare the evolution of the number of keyframes in our map and those generated by PTAM in Fig. 4.9. It can be seen how PTAM is always inserting keyframes, while our mechanism to prune redundant keyframes makes its number to saturate.

While the lifelong operation in a static scenario should be a requirement of any SLAM system, more interesting is the case where dynamic changes occur. We analyze the behavior of our system in such scenario by running consecutively the dynamic sequences from *fr3*: *sitting_xyz*, *sitting_halfsphere*, *sitting_rpy*, *walking_xyz*, *walking_halfspehere* and *walking_rpy*. All the sequences focus the camera to the same desk but perform different trajectories, while people are moving and change some objects like chairs. Fig. 4.10a shows the evolution of the total number of keyframes in the map, and Fig. 4.10b shows for each keyframe its frame of creation and destruction, showing how long the keyframes have survived in the map. It can be seen that during the first two sequences the map size grows as all the views of the scene are

being seen for the first time. In Fig. 4.10b we can see that several keyframes created during these two first sequences are maintained in the map during the whole experiment. During the sequences *sitting_rpy* and *walking_xyz* the map does not grow, because the map created so far explains well the scene. In contrast, during the last two sequences, more keyframes are inserted showing that there are some novelties in the scene that were not yet represented, due probably to dynamic changes. Finally Fig. 4.10c shows a histogram of the keyframes according to the time they have survived with respect to the remaining time of the sequence from its moment of creation. It can be seen that most of the keyframes are destroyed by the culling procedure soon after creation, and only a small subset survive until the end of the experiment. On one hand, this shows that our system has a generous keyframe spawning policy, which is very useful when performing abrupt motions in exploration. On the other hand the system is eventually able to select a small representative subset of those keyframes.

In these lifelong experiments we have shown that our map grows with the content of the scene but not with the time, and that is able to store different version of the scene when there are changes, which could be useful to analyze dynamic changes.

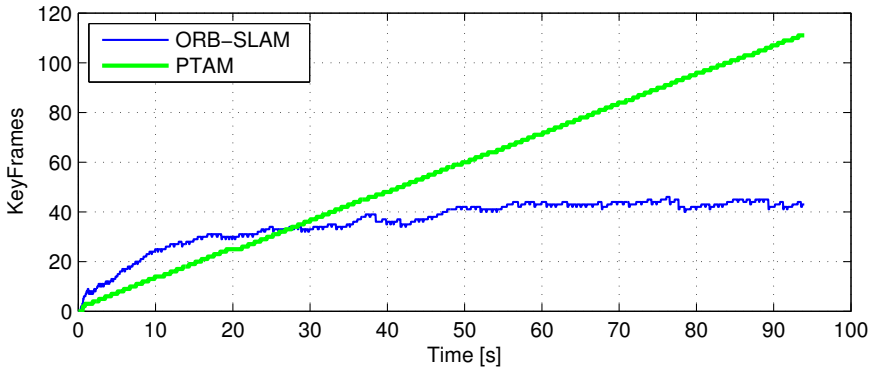
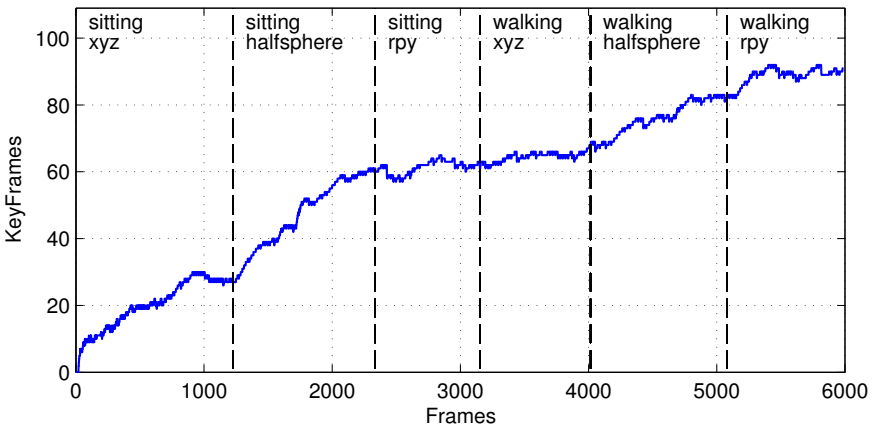
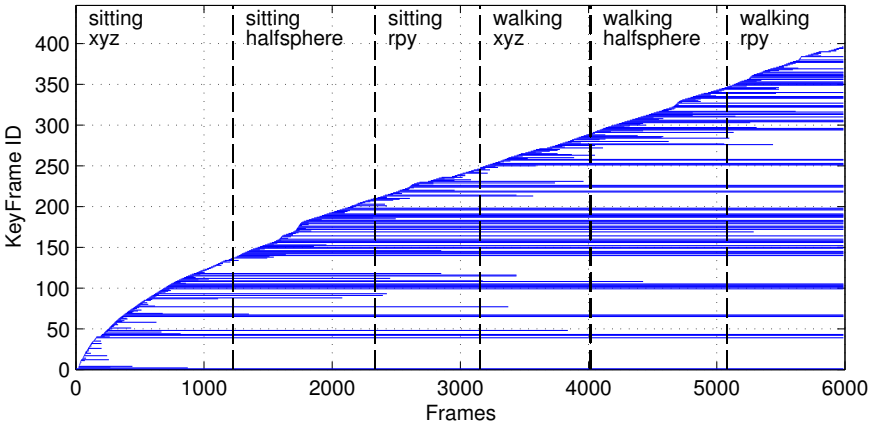


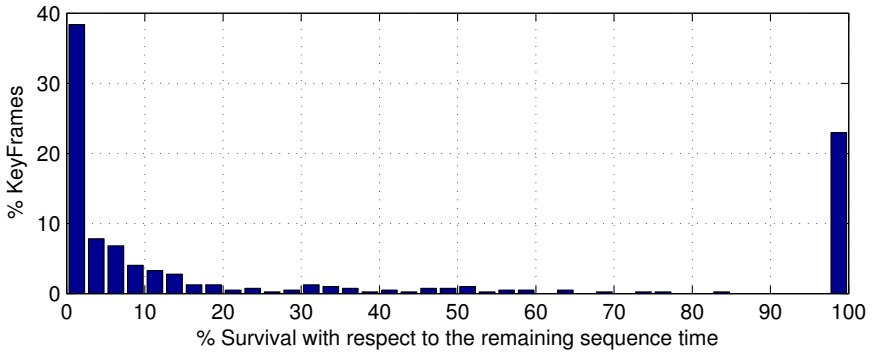
Figure 4.9: Lifelong experiment in a static environment where the camera is always looking at the same place from different viewpoints. PTAM is always inserting keyframes, while ORB-SLAM is able to prune redundant keyframes and maintains a bounded-size map.



(a) Evolution of the number of keyframes in the map



(b) Keyframe creation and destruction. Each horizontal line corresponds to a keyframe, from its creation frame until its destruction



(c) Histogram of the survival time of all spawned keyframes with respect to the remaining time of the experiment

Figure 4.10: Lifelong experiment in a dynamic environment from the TUM RGB-D Benchmark.

4.6.5 Large Scale and Large Loop Closing in the KITTI Dataset

The odometry benchmark from the KITTI dataset [26] contains 11 sequences from a car driven around a residential area with accurate ground truth. This is a very challenging dataset for monocular vision due to fast rotations, areas with lot of foliage, which make more difficult data association, and relatively high car speed, being the sequences recorded at 10 fps. We play the sequences at the real frame-rate they were recorded and ORB-SLAM is able to process all the sequences by the exception of sequence 01 which is a highway with few trackable close objects. Sequences 00, 02, 05, 06, 07, 09 contain loops that were correctly detected and closed by our system. Sequence 09 contains a loop that can be detected only in a few frames at the end of the sequence, and our system not always detects it (the results provided are for the executions in which it was detected).

Qualitative comparisons of our trajectories and the ground truth are shown in Fig. 4.11 and Fig. 4.12. As in the TUM RGB-D benchmark we have aligned the keyframe trajectories of our system and the ground truth with a similarity transformation. We can compare qualitatively our results from Fig. 4.11 and Fig. 4.12 with the results provided for sequences 00, 05, 06, 07 and 08 by the recent monocular SLAM approach of Lim et al. [42] in their figure 10. ORB-SLAM produces clearly more accurate trajectories for all those sequences by the exception of sequence 08 in which they seem to suffer less drift.

Table 4.5 shows the median RMSE error of the keyframe trajectory over five executions in each sequence. We also provide the dimensions of the maps to put in context the errors. The results demonstrate that our system is very accurate being the trajectory error typically around the 1% of its dimensions, sometimes less as in sequence 03 with an error of the 0.3% or higher as in sequence 08 with the 5%. In sequence 08 there are no loops and drift cannot be corrected, which makes clear the need of loop closures to achieve accurate reconstructions.

In this experiment we have also checked how much the reconstruction can be improved by performing 20 iterations of full BA, see Section 1.3.1 for details, at the end of each sequence. We have noticed that some iterations of full BA slightly improves the accuracy in the trajectories with loops but it has negligible effect in open trajectories, which means that the output of our system is already very accurate. In any case if the most accurate results are needed our algorithm provides a set of matches, which define a strong camera network, and an initial guess, so that full BA converge in few iterations.

Finally we show the efficacy of our loop closing approach and the influence of the $\theta_{\min}$ to include edges in the essential graph. We have selected the sequence 09 (a very long sequence with a loop closure at the end), and in the same execution we have evaluated different loop closing strategies. In table 4.6 we show the keyframe trajectory RMSE and the time spent in the optimization in different cases: without loop closing, if we directly apply a full BA (20 or 100 iterations), if we apply only pose graph optimization (10 iterations with different number of edges) and if we apply pose graph optimization and full BA afterwards. The results clearly show that before loop closure, the solution is so far from the optimal, that BA has convergence problems. Even after 100 iterations still the error is very high. On the other hand essential graph optimization shows fast convergence and more accurate results. It can be seen that the choice of $\theta_{\min}$ has not significant effect in accuracy but decreasing the number of edges the time can be significantly reduced. Performing an additional BA after

the pose graph optimization slightly improves the accuracy while increasing substantially the time.

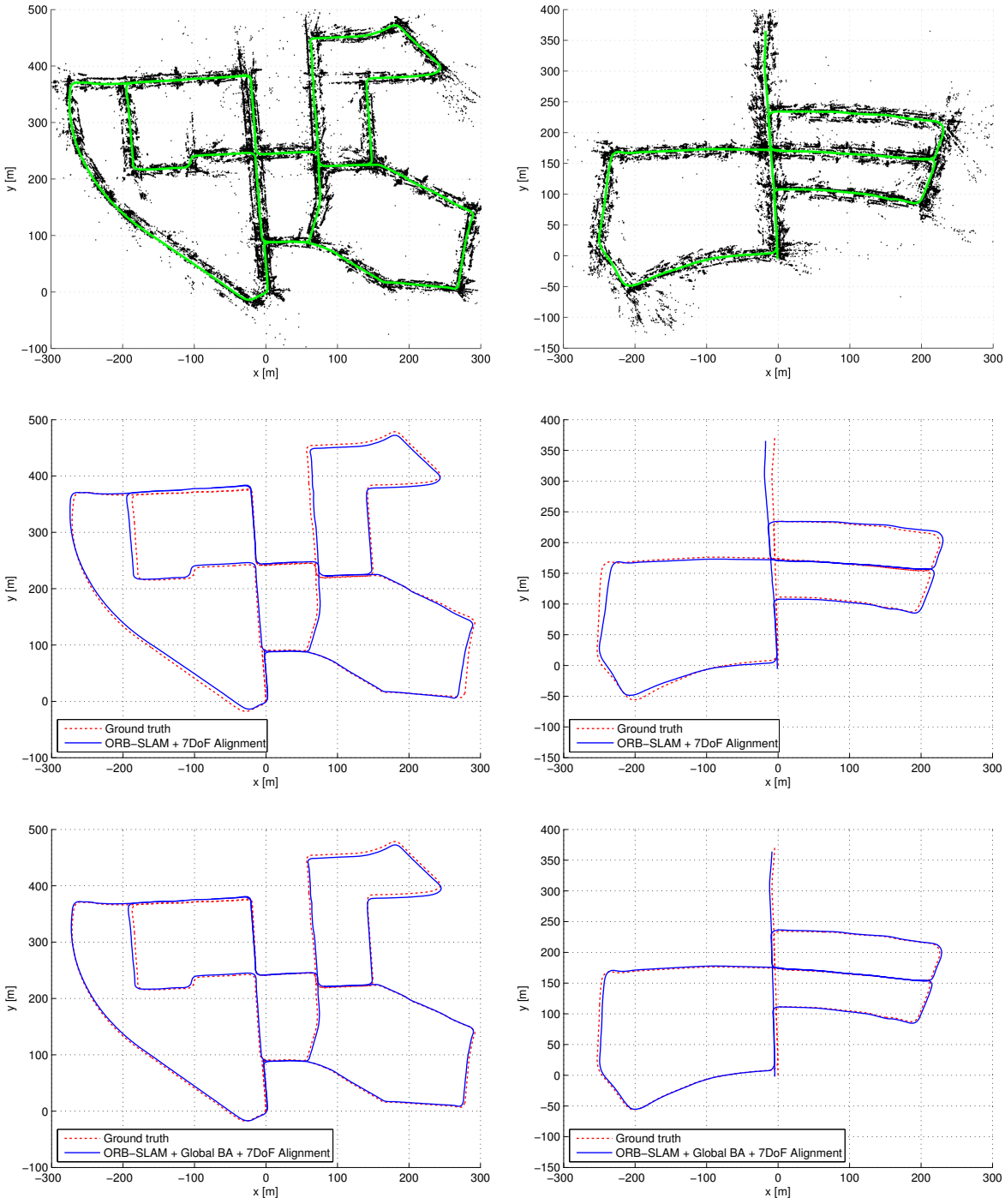
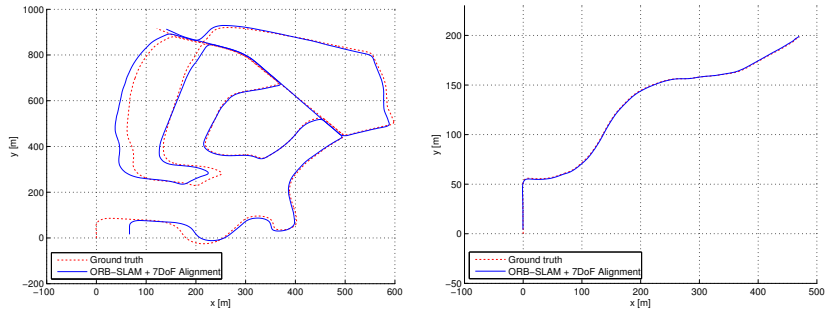
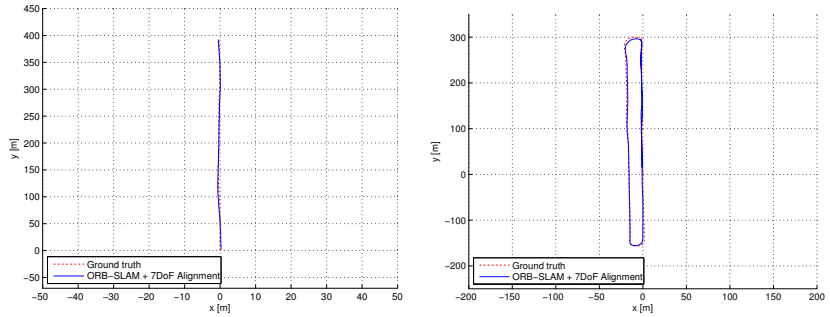


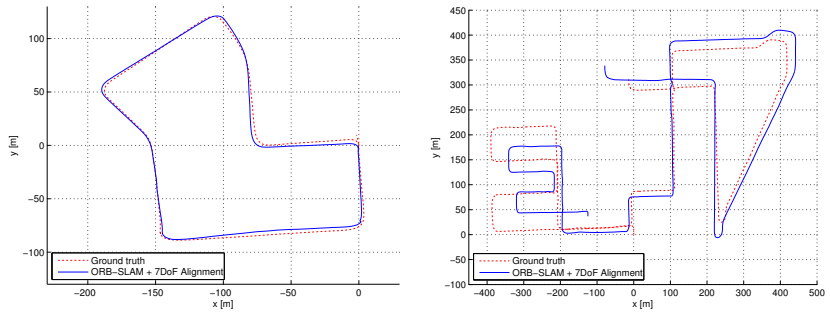
Figure 4.11: Sequences 00 and 05 from the KITTI dataset. Top: points and keyframe trajectory. Center: trajectory and ground truth. Bottom: trajectory after 20 iterations of full BA.



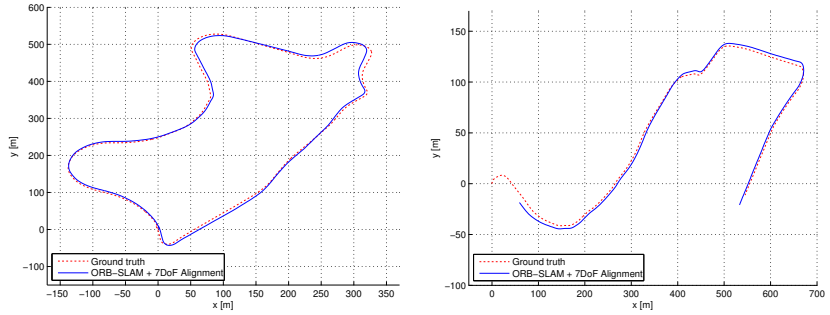
(a) Sequence 02 (b) Sequence 03



(c) Sequence 04 (d) Sequence 06



(e) Sequence 07 (f) Sequence 08



(g) Sequence 09 (h) Sequence 10

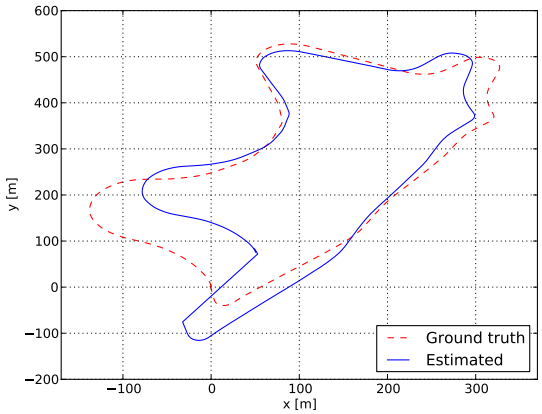
Figure 4.12: ORB-SLAM keyframe trajectories in the KITTI dataset.

Table 4.5: Results of our system in the KITTI dataset.

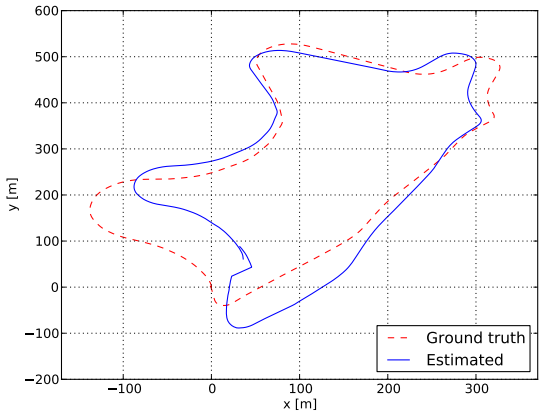
		ORB-SLAM		+ full BA (20 its.)	
Sequence	Dimension (m×m)	KFs	RMSE (m)	RMSE (m)	BA time (s)
KITTI 00	564×496	1391	6.68	5.33	24.83
KITTI 01	1157×1827	X	X	X	X
KITTI 02	599×946	1801	21.75	21.28	30.07
KITTI 03	471×199	250	1.59	1.51	4.88
KITTI 04	0.5×394	108	1.79	1.62	1.58
KITTI 05	479×426	820	8.23	4.85	15.20
KITTI 06	23×457	373	14.68	12.34	7.78
KITTI 07	191×209	351	3.36	2.26	6.28
KITTI 08	808×391	1473	46.58	46.68	25.60
KITTI 09	465×568	653	7.62	6.62	11.33
KITTI 10	671×177	411	8.68	8.80	7.64

Table 4.6: Comparison of loop closing strategies in KITTI 09.

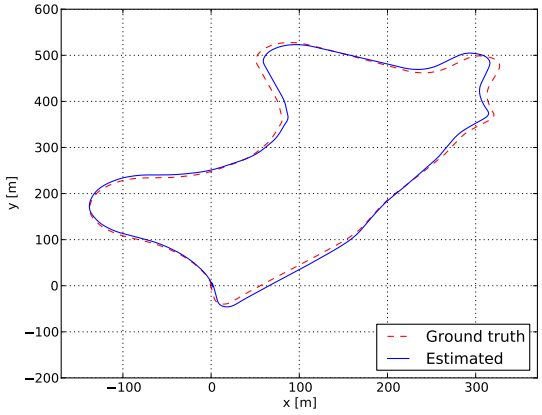
Method	Time (s)	Graph Edges	RMSE (m)
Without Loop Closure	-	-	48.77
Full BA (20 iterations)	14.64	-	49.90
Full BA (100 iterations)	72.16	-	18.82
Essential Graph ($\theta_{min} = 200$)	0.38	890	8.84
Essential Graph ($\theta_{min} = 100$)	0.48	1979	8.36
Essential Graph ($\theta_{min} = 50$)	0.59	3583	8.95
Essential Graph ($\theta_{min} = 15$)	0.94	6663	8.88
Essential Graph ($\theta_{min} = 100$) + full BA (20 iterations)	13.40	1979	7.22



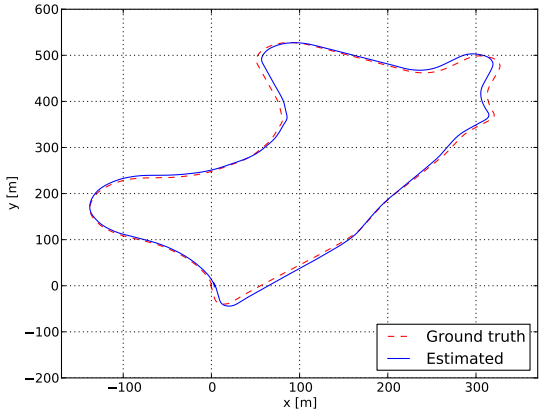
(a) Without Loop Closing



(b) Full BA (20 iterations)



(c) Essential Graph ($\theta_{min} = 100$)



(d) Essential Graph ($\theta_{min} = 100$) + Full BA (20 iterations)

Figure 4.13: Comparison of different loop closing strategies in KITTI 09.

4.7 Discussion

4.7.1 Conclusions

In this chapter we have presented a new monocular SLAM system with a detailed description of its building blocks and an exhaustive evaluation in public datasets. Our system has demonstrated that it can process sequences from indoor and outdoor scenes and from car, robot and hand-held motions. The accuracy of the system is typically below 1 cm in small indoor scenarios and of a few meters in large outdoor scenarios (once we have aligned the scale with the ground truth).

PTAM by Klein and Murray [36] has been considered the most accurate SLAM method from monocular video in real time. It is not coincidence that the backend of PTAM was bundle adjustment, which is well known to be the gold standard method for the offline

Structure From Motion problem [28]. One of the main successes of PTAM, and the earlier work of Mouragnon [51], was to bring that knowledge into the robotics SLAM community and demonstrate its real time performance. The main contribution of our work is to expand the versatility of PTAM to environments that are intractable for that system. To achieve this, we have designed from scratch a new monocular SLAM system with some new ideas and algorithms, but also building on excellent works developed in the past few years, such as the loop detection of Gálvez-López and Tardós [25], the loop closing procedure and covisibility graph of Strasdat et al. [74, 75], the optimization framework g2o by Kuemmerle et al. [38] and ORB features by Rubble et al. [69]. To the best of our knowledge, no other system has demonstrated to work in as many different scenarios and with such accuracy. Therefore our system is currently the most reliable and complete solution for monocular SLAM. Our novel policy to spawn and cull keyframes, permits to create keyframes every few frames, which are eventually removed when considered redundant. This flexible map expansion is really useful in poorly conditioned exploration trajectories, i.e. close to pure rotations or fast movements. When operating repeatedly in the same environment, the map only grows if the visual content of the scene changes, storing a history of its different visual appearances. Interesting results for long-term mapping could be extracted analyzing visual changes among keyframes of the same place inserted at different times.

Finally we have also demonstrated that ORB features have enough recognition power to enable place recognition from severe viewpoint change. Moreover they are so fast to extract and match (without the need of multi-threading or GPU acceleration) that enable real time accurate tracking and mapping.

4.7.2 Sparse/Feature-based vs. Dense/Direct Methods

Recent real-time monocular SLAM algorithms such as DTAM [60] and LSD-SLAM [17] are able to perform dense or semi dense reconstructions of the environment, while the camera is localized by optimizing directly over image pixel intensities. These direct approaches do not need feature extraction and thus avoid the corresponding artifacts. They are also more robust to blur, low-texture environments and high-frequency texture like asphalt [44]. Their denser reconstructions, as compared to the sparse point map of our system or PTAM, could be more useful for other tasks than just camera localization.

However, direct methods have their own limitations. Firstly, these methods assume a surface reflectance model that in real scenes produces its own artifacts. The photometric consistency limits the baseline of the matches, typically narrower than those that features allow. This has a great impact in reconstruction accuracy, which requires wide baseline observations to reduce depth uncertainty. Direct methods are quite affected by rolling-shutter, auto-gain and auto-exposure artifacts (as in the TUM RGB-D Benchmark), unless they are correctly modeled. In addition direct methods are in general very computationally demanding, therefore the map is just incrementally expanded as in DTAM, or map optimization is reduced to a pose graph, discarding all sensor measurements as in LSD-SLAM. Finally as direct methods rely on intensity gradients, initial solutions for the optimization need to be close to the true solution to converge, which reduce the robustness in case of low frame-rate or sudden movements.

In contrast, feature-based methods are able to match features with a wide baseline, thanks to their good invariance to viewpoint and illumination changes. Bundle adjustment jointly optimizes camera poses and points over sensor measurements. In the context of structure and motion estimation, Torr and Zisserman [83] already pointed the benefits of feature-based against direct methods. In this work we provide experimental evidence (see Section 4.6.2) of the superior accuracy of feature-based methods in real-time SLAM.

A very promising work is the recent work of Engel et al. [16] which proposes a sparse direct approach for visual odometry. Not using a geometry prior, compared to DTAM or LSD-SLAM, it is able to perform for the first time photometric bundle adjustment over a sparse and well-distributed set of points on selected keyframes. The method has excellent robustness and accuracy when using photometric calibration with global shutter cameras.

We consider that the future of monocular SLAM should incorporate the best of both feature-based and direct approaches.

Chapter 5

Probabilistic Semi-Dense Mapping

In the previous chapter we have presented ORB-SLAM, a novel feature-based monocular SLAM system, which operates in real-time in indoor and outdoor environments, being able to relocalize and close loops from significantly different viewpoints and with a robust map bootstrapping. After the excellent DTAM [60] and LSD-SLAM [17] direct SLAM systems, there is the extended believe in the community that direct methods are more robust as not relying on feature detection, and are more accurate because they used more information from the images. Surprisingly, our evaluation in Chapter 4 shows the opposite. However ORB-SLAM as a feature-based method, produces a sparse map of the environment which is very good for camera localization but of little use to describe the environment. Therefore in this chapter we aim to densify the reconstruction of ORB-SLAM.

Some previous works have proposed dense reconstruction methods using GPUs, built over feature-based SLAM [58, 77] or visual odometry algorithms [67]. Following a similar approach, in this chapter we propose a probabilistic semi-dense mapping module to perform in real-time, without requiring GPU acceleration, rich semi-dense reconstructions. One of the main novelties of our semi-dense mapping method is that instead of using many subsequent frames to filter the inverse depth of a reference frame [17, 19, 67], we perform the reconstruction over keyframes, which are very well localised by local bundle adjustment, and pose graph optimization after a loop closure. This allows to obtain high quality and accurate reconstructions. If the highest accuracy is desired, the reconstruction can also be performed at the end of the session in few seconds after a full bundle adjustment.

Our stereo correspondence search and inverse depth uncertainty derivation is based on [19]. However we search correspondences between keyframes (wider baselines) and therefore we have to deal with potentially more outliers, due to occlusions or multiple similar pixels. To gain robustness, in addition to the photometric similarity, we compare the magnitude and orientation of the image gradient, and propose a novel measurement fusion. We also propose an *inter*-keyframe depth consistency check that discards most of the outliers, see an example in Fig. 5.3. In contrast to [19], our formulation does not make assumptions of small rotations in the derivation of the inverse depth uncertainty.

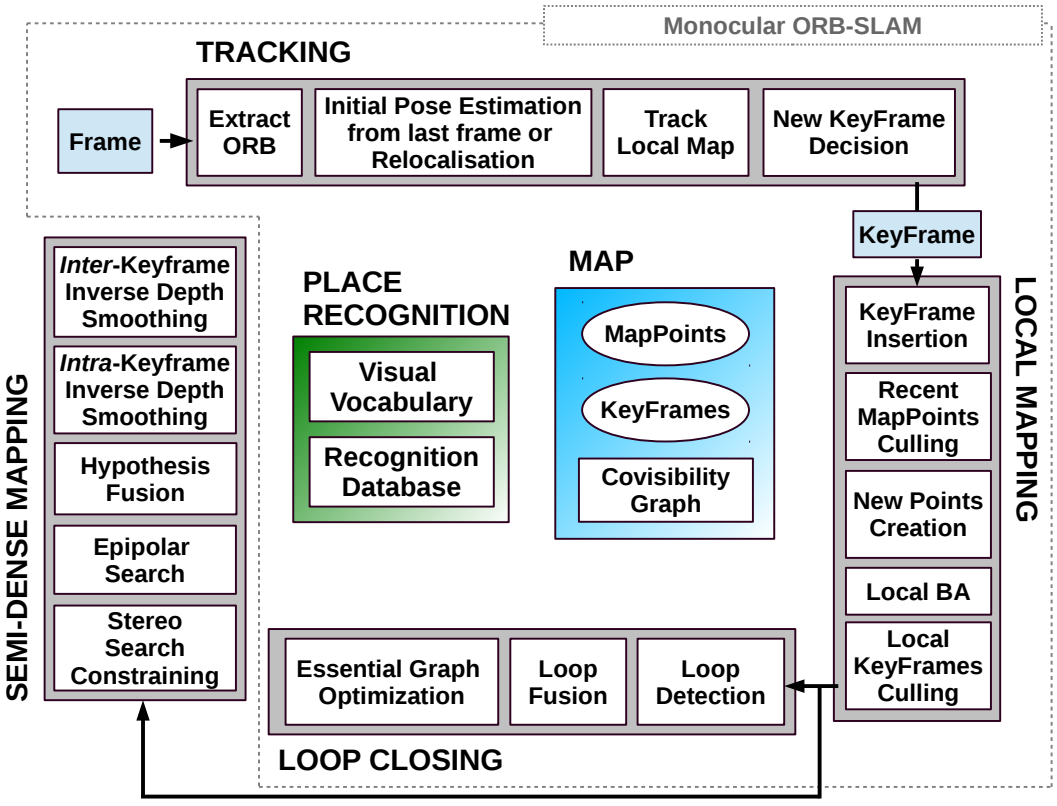


Figure 5.1: Our system including the three threads of ORB-SLAM, tracking, local mapping and loop closing, and the semi-dense mapping thread proposed in this chapter.

5.1 Method

Our system includes the three original threads of ORB-SLAM, which compute the camera trajectory and create a sparse map that is used for localization. In this chapter we propose a semi-dense mapping thread, as seen in Fig. 5.1, that processes the keyframes to create a semi-dense reconstruction of the environment. This semi-dense reconstruction is used to describe the environment but not for camera localization. The outline of our semi-dense mapping method is the following:

1. Each keyframe K_i is processed from scratch. Every pixel in a high gradient area is searched along the epipolar line on N neighbor keyframes, yielding N inverse depth hypotheses.

2. Each inverse depth hypothesis is represented by a gaussian distribution that takes into account the image noise, the parallax and the ambiguity in the matching. We consider that the keyframe poses are well localised and do not take into account their uncertainty.

3. Because the baseline is wide between keyframes the search range along the epipolar line is large. To deal with outlier measurements, due to similar pixels or occlusions, we fuse

the maximum subset of the N hypotheses that are mutually compatible. Each pixel p of the inverse depth map is then characterized with a gaussian distribution $\mathcal{N}(\rho_p, \sigma^2_{\rho_p})$.

4. As proposed in [19], a smoothing step is then applied to the inverse depth map so that a pixel is averaged with its neighbors. If a pixel inverse depth distribution is not compatible with its neighbors it is discarded.

5. After the neighbor keyframes have also computed their respective inverse depth maps, consistency in the per-pixel depths is checked across neighbor keyframes to discard outliers and the final depth is refined by optimization.

5.1.1 Stereo Search Constraints

Our feature-based SLAM system provides useful information to constrain the search of pixel correspondences to compute the inverse depth map. On one hand keyframes have associated tracked ORB features with known depth, which renders us the maximum $\rho_{\max}$ and minimum $\rho_{\min}$ expected inverse depths of the scene. This provides a prior $\mathcal{N}(\rho_0, \sigma^2_{\rho_0})$, with $\rho_{\max} = \rho_0 + 2\sigma_{\rho_0}$ and $\rho_{\min} = \rho_0 - 2\sigma_{\rho_0}$, for the inverse depth search, as illustrated in Fig. 5.2.

In addition using the covisibility graph we can retrieve the set of N keyframes $\mathcal{K}$, which share most map point observations with K_i, and focus the stereo search in those keyframes. Keyframes are processed with a small delay (around 10 keyframes) so that they can be reconstructed using also *future* keyframes to get a better reconstruction. This is also convenient as local BA optimizes recent keyframes, potentially interfering with this semi-dense mapping thread.

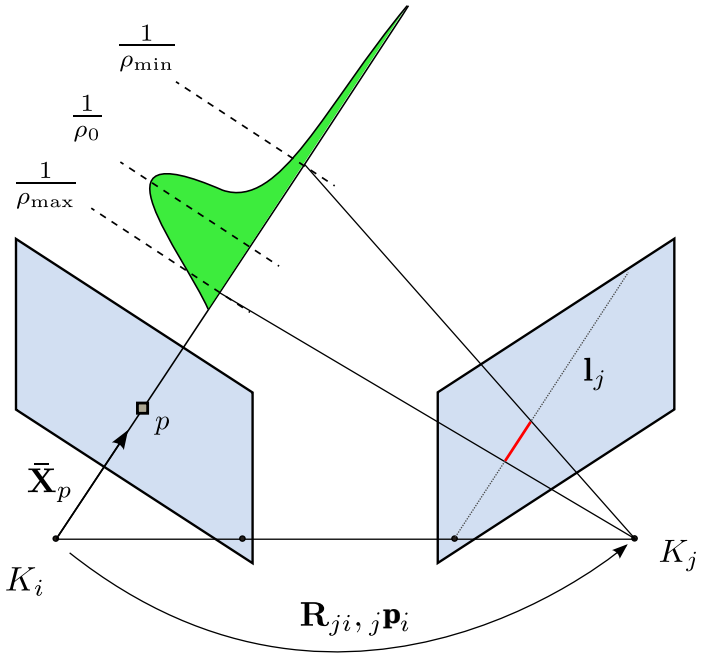


Figure 5.2: Epipolar constrained search of a pixel in a neighbor keyframe given a prior inverse depth distribution.

5.1.2 Epipolar Search

Each pixel p of K_i with gradient magnitude greater than a threshold λ_G is searched along the epipolar line $\mathbf{l}_j$ on each keyframe $K_j \in \mathcal{K}$, constrained to the segment between $\rho_{\min}$ and $\rho_{\max}$. The epipolar line is computed from the fundamental matrix $\mathbf{F}_{ji}$ [28], and for the sake of simplicity in the rest we parametrize it as a function of the horizontal coordinate u_j:

$$\mathbf{x}_j^\top \mathbf{F}_{ji} \mathbf{x}_p = \mathbf{x}_j^\top \mathbf{l}_j = 0 \quad \rightarrow \quad v_j = m \cdot u_j + n \tag{5.1}$$

In contrast to [19] (narrow baseline frames), our search along the epipolar line is larger (wider baseline keyframes) and we need to take special care of outliers. Therefore, in addition to comparing the intensity I, we propose to compare the magnitude G and orientation Θ of the image gradient.

The pixel p is characterized by an intensity value I_p, a gradient magnitude G_p and orientation Θ_p, and the goal is to find its best correspondence on $\mathbf{l}_j$. Firstly the pixels $p_j \in \mathbf{l}_j$ not fulfilling the following conditions are not considered:

- p_j must lie in a high gradient area, that is $G(u_j) > \lambda_G$.
- The ambiguity of a match is related to the intensity gradient along the epipolar line [19]. Therefore the gradient direction must not be perpendicular to the epipolar line, that is $|\Theta(u_j) - \Theta_L \pm \pi| < \lambda_L$, with Θ_L the epipolar line angle (considering both directions).
- The gradient orientation of p_j must be similar, that is $|\Theta(u_j) - (\Theta_{p_i} + \Delta\theta_{j,i})| < \lambda_\theta$, where $\Delta\theta_{j,i}$ is the in-plane rotation between keyframe images, which is computed from the median rotation of corresponding ORB between both keyframes.

These conditions discard most of the points of the epipolar line, reducing potential mismatches. To compare the remaining points we define a similarity error $e(u_j)$:

$$e(u_j) = \frac{r_I^2}{\sigma_I^2} + \frac{r_G^2}{\sigma_G^2}, \quad r_I = I_p - I(u_j), \quad r_G = G_p - G(u_j) \tag{5.2}$$

where r_I is the photometric error and r_G is the gradient magnitude error; σ_I and σ_G are the standard deviation of the intensity and gradient respectively. Because the gradient is a function of the intensity their noise are related $\sigma_G^2 = \theta\sigma_I^2$ with $\theta = 0.23$ if using the Scharr operator to compute the image derivatives ($\theta < 1$ as the Scharr operator performs an average reducing the noise). With this relation the similarity error is:

$$e(u_j) = (r_I^2 + \frac{1}{\theta} r_G^2) \frac{1}{\sigma_I^2} \tag{5.3}$$

We select the pixel at coordinate u_0 that minimizes this error, with residuals r_{I_0} and r_{G_0}. We can then compute the derivate of the error:

$$\frac{\partial e}{\partial u_j} = \frac{-2(r_I\, g + \frac{1}{\theta} r_G\, q)}{\sigma_I^2} \tag{5.4}$$

where g is the intensity gradient and q is the derivate of the intensity gradient magnitude, both along the epipolar line:

$$g \approx \frac{I(u_j+1) - I(u_j-1)}{2}, \quad q \approx \frac{G(u_j+1) - G(u_j-1)}{2} \tag{5.5}$$

Performing a first order taylor approximation of the residuals (5.2) and equaling to zero the similarity error derivate (5.4) we can retrieve the pixel correspondence with subpixel precision:

$$u_0^* = u_0 + \frac{g(u_0) r_I(u_0) + \frac{1}{\theta} q(u_0) r_G(u_0)}{g^2(u_0) + \frac{1}{\theta} q^2(u_0)} \tag{5.6}$$

Now we can derive the uncertainty of u_0^* from the intensity noise σ_I^2 by error propagation, for simplicity considering only the noise in the residuals $r_I(u_0)$ and $r_G(u_0)$:

$$\sigma_{u_0^*}^2 = \frac{2\sigma_I^2}{g^2(u_0) + \frac{1}{\theta} q^2(u_0)} \tag{5.7}$$

This uncertainty tell us that a match is more reliable as higher is the gradient along the epipolar of the quantities involved in the similarity measure (5.2), in our case the intensity and the image gradient magnitude.

Now we need to propagate the uncertainty of the match $\sigma_{u_0^*}^2$ to the uncertainty in the inverse depth $\sigma_{\rho_p}^2$. The inverse depth ρ_p of pixel p in K_i is a function of the position in the epipolar line u_j (which can be derived from the formula of the projection of a 3D world point into a camera image [28]):

$$\rho_p(u_j) = \frac{\mathbf{r}_z^{ji} \bar{\mathbf{X}}_p (u_j - c_x) - f_x \mathbf{r}_x^{ji} \bar{\mathbf{X}}_p}{-t_z^{ji} (u_j - c_x) + f_x t_x^{ji}} \tag{5.8}$$

where $\mathbf{r}_z^{ji}$ and $\mathbf{r}_x^{ji}$ are the third and first row of the rotation $\mathbf{R}_{ji}$, t_z^{ji} and t_x^{ji} are the third and first elements of the translation ${}_j\mathbf{p}_i$, $\bar{\mathbf{X}}_p = \mathbf{K}^{-1} \mathbf{x}_p$ is the unary ray trough pixel p as seen in Fig. 5.2, being $\mathbf{K}$ the calibration matrix, and f_x and c_x are the focal and the principal point. Using equation (5.8) we form the inverse depth hypothesis $\mathcal{N}(\rho_j, \sigma_{\rho_j}^2)$, as follows:

$$\begin{aligned} \rho_j &= \rho_p(u_0^*) \\ \sigma_{\rho_j} &= \max(|\rho_p(u_0^* + \sigma_{u_0^*}) - \rho_j|, |\rho_p(u_0^* - \sigma_{u_0^*}) - \rho_j|) \end{aligned} \tag{5.9}$$

Note that our uncertainity propagation is general, in contrast to the assumption of small rotations in [19].

5.1.3 Inverse Depth Hypothesis Fusion

At this point we have a set of inverse depth hypotheses for the pixel p. The number of hypotheses can be less than N as the epipolar line segment between $\rho_{\min}$ and $\rho_{\max}$ could lie entirely out of some of the keyframes or no pixel fulfills the conditions described in section 5.1.2. In addition some of the hypotheses can be outliers due to several similar pixels or

occlusions. We therefore search for at least λ_N compatible hypotheses. The compatibility between two hypotheses a, b is tested with the χ^2 test at 95%:

$$\frac{(\rho_a - \rho_b)^2}{\sigma_a^2} + \frac{(\rho_a - \rho_b)^2}{\sigma_b^2} < 5.99 \tag{5.10}$$

Selecting at each time a hypothesis we check the compatibility with the rest of hypotheses. If the best combination gives $n > \lambda_N$ compatible measures, they are fused, yielding the inverse depth distribution $\mathcal{N}(\rho_p, \sigma_{\rho_p}^2)$ for the pixel p:

$$\rho_p = \frac{\sum_n \frac{1}{\sigma_{\rho_j}^2}\rho_j}{\sum_n \frac{1}{\sigma_{\rho_j}^2}}, \quad \sigma_{\rho_p}^2 = \frac{1}{\sum_n \frac{1}{\sigma_{\rho_j}^2}} \tag{5.11}$$

5.1.4 Intra-Keyframe Depth Checking, Smoothing and Growing

After we have computed the semi-dense inverse depth map of the keyframe we perform an outlier removal, smoothing and growing step as proposed in [19]. To retain the inverse depth measurement of a pixel its inverse depth distribution must be supported by at least 2 of its 8 pixel neighbors $p_{i,n}$ as described in (5.10). The inverse depth of those retained pixels is averaged by their compatible neighbors using (5.11), but fixing the standard deviation to the minimum of the neighbors. This step smooths the reconstruction, while preserving edges, as only compatible measurements are averaged. Those pixels, in a high gradient area that do not have an inverse depth measurement but are surrounded by at least two pixels with compatible distributions, are also assigned an average inverse depth (with the minimum standard deviation). This grows the reconstruction getting more density.

5.1.5 Inter-Keyframe Depth Checking and Smoothing

Once the inverse depth maps of the neighbors of K_i have been computed, we check with them the consistency of each inverse depth distribution in the inverse depth map of K_i. For each pixel p of K_i with an associated inverse depth ρ_p, we project the corresponding 3D point into each neighbor keyframe K_j and propagate the inverse depth as follows:

$$\begin{aligned} \mathbf{x}_j &= \pi_m \left(\mathbf{R}_{ji} \frac{1}{\rho_p} \bar{\mathbf{X}}_p + {}_j\mathbf{p}_i \right) \\ \rho_j &= \frac{\rho_p}{\mathbf{r}_z^{ji} \bar{\mathbf{X}}_p + \rho_p \, t_z^{ji}} \end{aligned} \tag{5.12}$$

As the projection $\mathbf{x}_j$ will not coincide with an integer pixel coordinate, we look in the 4 neighbor pixel $p_{j,n}$ around $\mathbf{x}_j$ for a compatible inverse depth as follows:

$$\frac{(\rho_j - \rho_{j,n})^2}{\sigma_{\rho_{j,n}}^2} < 3.84 \tag{5.13}$$

To retain the inverse depth distribution of a pixel p, at least one compatible pixel $p_{j,n}$ must be found in at least λ_N neighbor keyframes.

Finally we perform a gauss-newton step that minimizes the depth difference in all compatible pixels:

$$d_p^* = \min_{d_p} \sum_{j,n} (d_{j,n} - d_p \, \mathbf{r}_z^{ji} \bar{\mathbf{X}}_p - t_z^{ji})^2 \frac{1}{d_{j,n}^4 \sigma_{\rho_{j,n}}^2} \tag{5.14}$$

We optimize the depth instead of its inverse because the propagation equation (5.12) is linear in depth an the optimal d_p^* is reached in one iteration. The denominator of (5.14) comes from uncertainity propagation of the inverse depth to depth.

5.2 Experimental Evaluation

In this section we present several experiments to show the performance of our semi-dense mapping approach. We have performed all experiments in a laptop with an Intel i7-4700MQ processor, which allows to run simultaneously 8 threads. Our feature-based ORB-SLAM and the semi-dense module are implemented in C++ and run in the Robot Operating System (ROS). The feature-based SLAM system uses 3 threads (the tracking, local mapping and loop closing), while ROS will probably make use of at least 1 additional thread. Therefore in the *online* setting, the semi-dense mapping module makes use of 4 threads for multi-threading optimization. All operations described in section 5.1 are independent for each pixel and therefore can be parallelized. The values for the parameters of the semi-dense module were set as follows: $N = 7$, $\sigma_I = 20$, $\lambda_G = 8$, $\lambda_L = 80°$, $\lambda_\theta = 45°$ and $\lambda_N = 3$.

5.2.1 The importance of removing outliers

One of the main characteristics of our semi-dense mapping method is that stereo correspondences are searched between keyframes with wide baseline. This can produce the appearance of outliers due to occlusions or multiple similar pixels. Although each inverse depth value has an associated uncertainty, imposing a restrictive variance threshold is not enough to remove outliers that could have similar uncertainty than inliers. This motivated the inclusion of an inter-keyframe depth consistency checking, see Section 5.1.5, to detect and remove outliers. Fig. 5.3 shows a semi-dense reconstruction of a planar scene that consists of several posters on the floor. First row of the figure shows the top and side views of the reconstruction without applying any outlier detection, which results in a solution with many outliers. Second row is the same reconstruction, but retaining only the pixels that have an inverse depth variance below a threshold, which reduces the number of outliers but not completely. Third row is the original reconstruction with the inter-keyframe outlier removal, but without a variance threshold, and almost all outliers have been removed. The best solution is shown in the fourth row with both a variance threshold and the inter-keyframe checking.

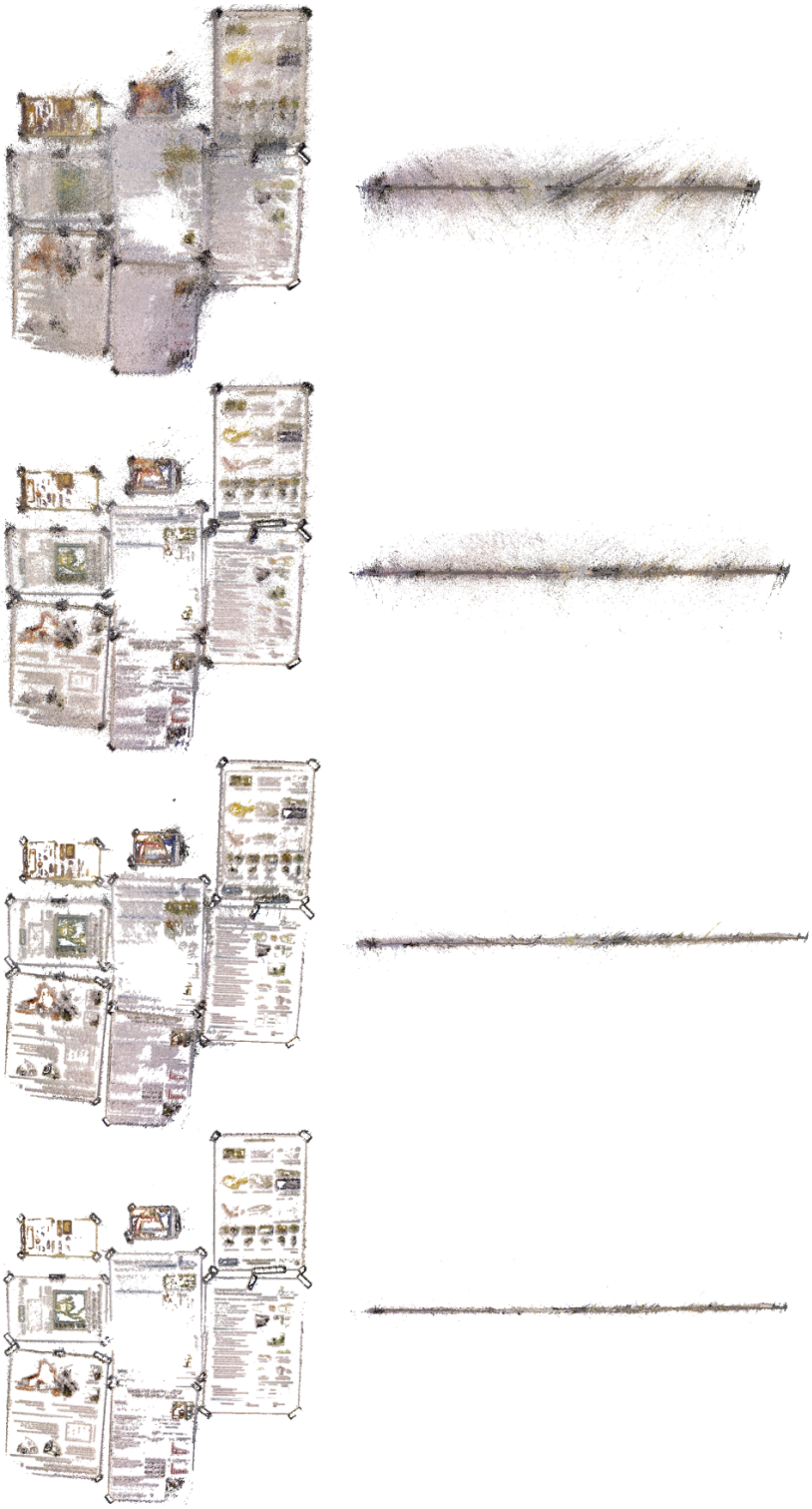


Figure 5.3: Example of outlier removal in *fr2_nostructure_texture_near_with_loop* (TUM RGB-D Dataset [78]). Description in text.

5.2.2 Accuracy

In Chapter 4 we performed an extensive evaluation of ORB-SLAM, in terms of keyframe pose accuracy. We used the TUM RGB-D Benchmark [78] as it provided several sequences with accurate ground-truth camera localization from an external motion capture system. Despite the dataset is quite exigent for monocular SLAM (e.g. limited field of view, blur, strong rotations), we achieved a typical RMSE error in the keyframe position around 1cm, and in some sequences as the *fr2_xyz* only 3 mm. We compared our results in 16 sequences with the state-of-the-art direct SLAM, LSD-SLAM [17], and surprisingly we obtained higher accuracy (around 5 times better) and higher robustness, as LSD-SLAM was not able to process all sequences.

To test our semi-dense mapping method we have selected 4 of the sequences in which the camera motion allows to recover a detailed reconstruction. Still those sequences where not recorded with care for semi-dense/dense reconstruction as it was not the goal. An analysis of suitable camera movements for this kind of reconstructions can be found in [22]. Left and middle columns of Fig. 5.4 shows different viewpoints of each reconstruction. It can be seen how the reconstruction contains very few outliers, while the point density is enough to recognize different objects as seen in Fig.5.5. The accuracy of the reconstruction can be noticed in the straight contours of the the desk in *fr2_desk* (first row), the scene planarity in *fr3_nostructure_texture_near_with_loop* (second row), and the readable text in sequence *fr3_structure_texture_near* (last row). In the sequence *fr3_long_office_household* (third row) there is a close approximation of the camera to the teddy bear that introduce more error than normal drift accumulation. Therefore the pose graph optimization performed at the loop closure at the end of the sequence cannot completely compensate this error. The result is that the desk contours do not completely align, despite in general the reconstruction being quite accurate. Running the reconstruction offline after full BA yields a perfectly aligned and accurate solution.

Table 5.1 shows the median times per keyframe and total reconstruction times for each sequence. It can be seen that the system operates in real-time as the total time spent by the semi-dense mapping module is less than the total sequence length. However it is important to remind that the reconstruction is always with some seconds delay to permit the reconstruction of a keyframe with *future* keyframes and to avoid interferences of the local BA. Variations in time depend mainly on the amount of high gradient pixels in the keyframes.

To compare we have executed LSD-SLAM in the same sequences (right column of Fig. 5.4). The reconstruction of *fr2_desk* is similar to ours. In *fr3_nostructure_texture_near_with_loop* the loop at the end of the sequence is not closed and posters do not perfectly align. The reconstruction of *fr3_long_office_household* is broken in one of the sides of the desk as it is highlighted, because the reconstruction is severely corrupted after the close camera approximation to the teddy bear and the loop closure can only partially mitigate the error. Finally in sequence *fr3_structure_texture_near* the tracking fails after a rotation at the beginning of the sequence.

Figure 5.4: Left and middle columns: our semi-dense reconstructions in TUM RGB-D Dataset [78]. Right: LSD-SLAM [17]. Variance thresholds have been adapted in both systems to show the reconstructions as clean as possible. Reconstructions from LSD-SLAM have been taken from its grayscale visualizer subtracting the background color.

Table 5.1: Online reconstruction times for the sequences in Fig. 5.4 (in row order).

Time per Keyframe	seq.1	seq. 2	seq.3	seq. 4
Inverse Depth Map Estimation (ms)	234	232	268	170
Intra-Keyframe Smoothing (ms)	28	25	31	24
Inter-Keyframe Smoothing (ms)	151	128	154	119
Total (ms)	425	376	451	308
Reconstruction Time (s)	65.2	37.0	67.1	14.3
Sequence Length (s)	98.8	56.6	87.1	37.0

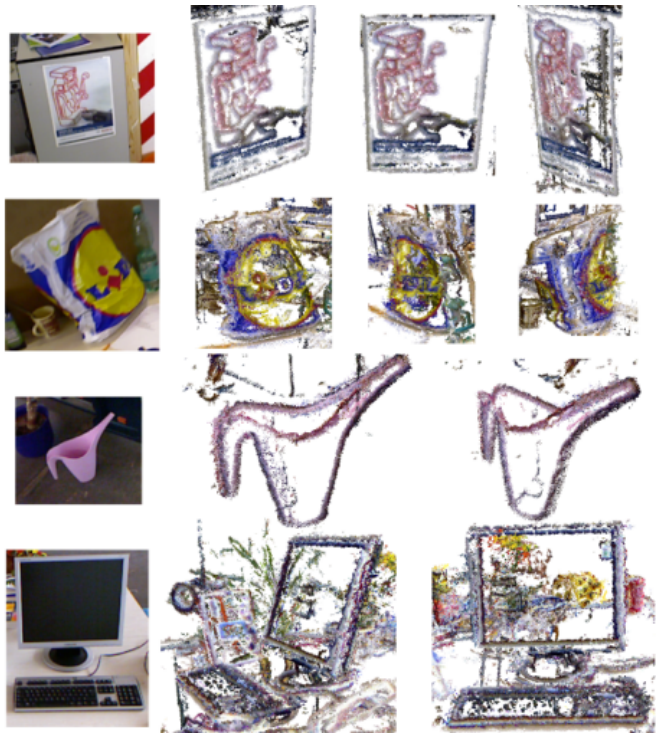


Figure 5.5: Example of reconstructed objects that are easily recognizable.

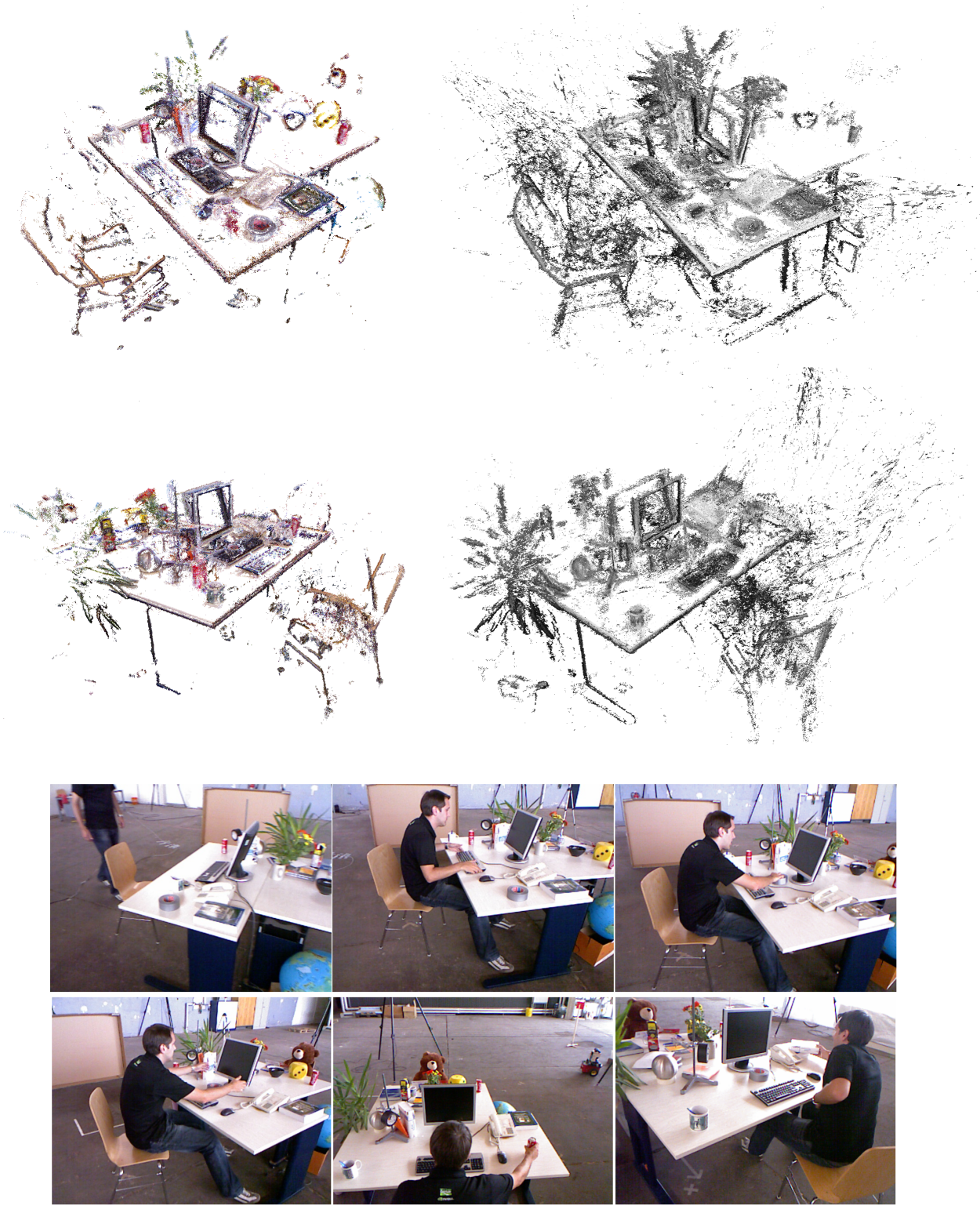


Figure 5.6: Example of a dynamic scene. At the bottom it can be seen that there is a person changing object positions. Both our reconstruction and LSD-SLAM are shown for comparison.

5.2.3 Dynamic Scenes

In this experiment we have run our system in the sequence *fr2_desk_with_person*. This is a desk sequence where a person is moving and changes some object positions. Our SLAM system is robust under those dynamic elements, achieving a RMSE error in the keyframe positions of 6.3mm. Because the semi-dense mapping operates over the keyframes, only objects that have remained static in several keyframes are reconstructed. The whole reconstruction is shown in Fig. 5.6. It can be seen that low dynamic changes (final positions of an object) are present in the reconstruction while static elements (e.g. the desk contour) are well defined.

We have also executed LSD-SLAM in this sequence to compare. It can be seen in Table 4.3 the low accuracy achieved in this sequence. The reconstruction is also shown in Fig. 5.6, where the point clouds of the first 30 keyframes of the sequence are not shown as they were very wrongly positioned. Still the overall reconstruction contains many outliers.

5.3 Discussion

In this chapter we have presented a novel probabilistic semi-dense mapping module for our ORB-SLAM framework to perform in real-time, in a conventional computer and without GPU, rich semi-dense reconstructions. The semi-dense mapping operates over keyframes, which are very well localised due to local BA and pose graph optimization at loop closing, allowing to obtain high quality reconstructions. The search of pixel correspondences in wide baseline keyframes motivated a novel inverse-depth hypothesis fusion and an *inter*-keyframe outlier detection mechanism, which checks the depth consistency across keyframes, resulting in clean reconstructions with very few outliers. Our correspondence search and inverse depth uncertainty derivation is based on [19], adding the image gradient magnitude and orientation in the comparison, and deriving the equations without narrow baseline assumptions, as we operate on keyframes. Figure 5.3 showed that our probabilistic uncertainty model and the novel inter-keyframe outlier detection significantly improves the reconstruction quality, irrespective of the keyframe poses, which is one of the main contributions of this paper. The main limitation of our approach is that the semi-dense reconstruction is obtained with a few keyframes delay, and it is not used for camera tracking.

Chapter 6

Stereo and RGB-D ORB-SLAM (ORB-SLAM2)

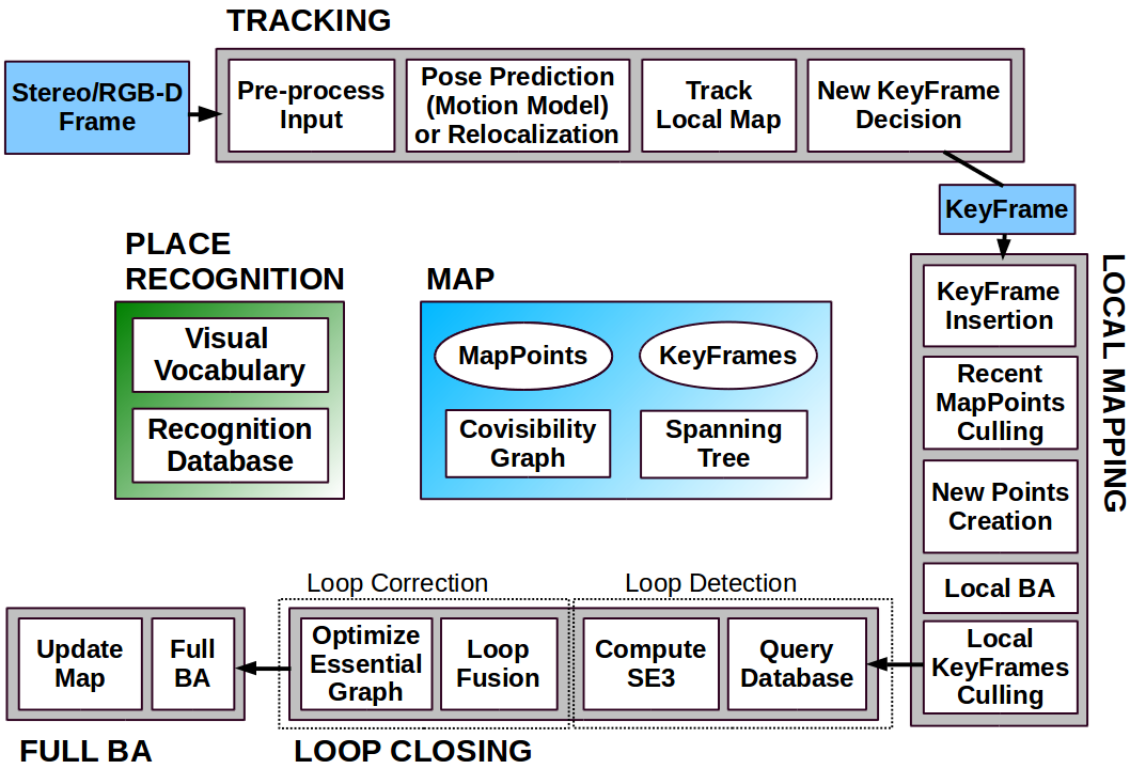
In previous chapters we have seen that visual SLAM can be performed by using just a monocular camera, which is the cheapest and smallest sensor setup. However as depth is not observable from just one camera, the scale of the map and estimated trajectory is unknown. In addition the system bootstrapping require multi-view or filtering techniques to produce an initial map as it cannot be triangulated from the very first frame. Last but not least, monocular SLAM suffers from scale drift and may fail if performing pure rotations in exploration. By using a stereo or an RGB-D camera all these issues can be solved, allowing for the most reliable visual SLAM solutions.

In this chapter we built on our monocular ORB-SLAM, described in Chapter 4, and propose ORB-SLAM2 with the following contributions:

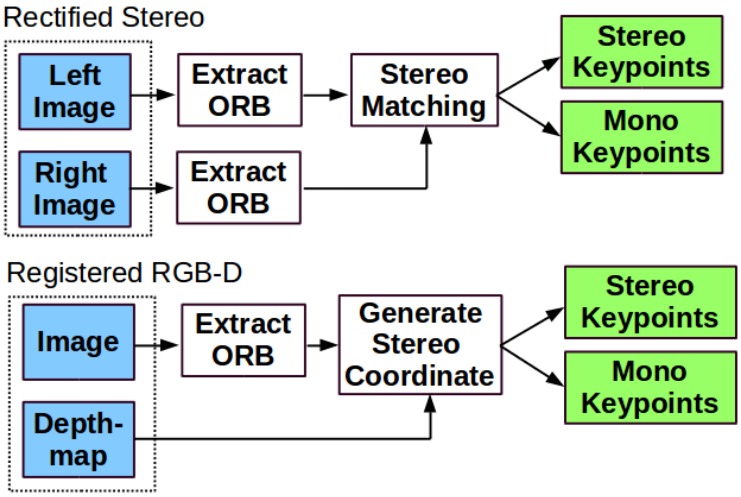
- The first open-source SLAM system for monocular, stereo and RGB-D cameras, including loop closing, relocalization and map reuse.
- Our RGB-D results show that by using bundle adjustment we achieve more accuracy than state-of-the-art methods based on ICP or photometric and depth error minimization.
- By using close and far stereo points and monocular observations our stereo results are more accurate than the state-of-the-art direct stereo SLAM.
- A lightweight localization mode that can effectively reuse the map with mapping disabled.

6.1 System description

Fig. 6.1 shows an overview of ORB-SLAM2 for stereo and RGB-D cameras, which is built on our monocular feature-based ORB-SLAM, described in Chapter 4. The system has three main parallel threads: 1) the Tracking to localize the camera with every frame by finding feature matches to the local map and minimizing the reprojection error applying motion-only



(a) System Threads and Modules.



(b) Input pre-processing

Figure 6.1: ORB-SLAM2 overview. The tracking thread pre-process the stereo or RGB-D input so that the rest of the system operates independently of the input sensor. Although it is not shown here, ORB-SLAM2 also works with monocular as described in Chapter 4.

BA, 2) the Local Mapping to manage the local map and optimize it, performing local BA, 3) the Loop Closing to detect large loops and correct the accumulated drift by performing a pose-graph optimization. This thread launches a fourth thread to perform full BA after the

pose-graph optimization, to compute the optimal structure and motion solution.

The system has embedded a Place Recognition module based on DBoW2 [25] for relocalization, in case of tracking failure (e.g. an occlusion) or for reinitialization in an already mapped scene, and for loop detection. The system maintains a covisibiliy graph [74] that links any two keyframes observing common points and a minimum spanning tree connecting all keyframes. These graph structures allow to retrieve local windows of keyframes, so that Tracking and Local Mapping operate locally, allowing to work on large environments, and serve as structure for the pose-graph optimization performed when closing a loop.

The system uses the same ORB features [69] for tracking, mapping and place recognition tasks. These features are robust to rotation and scale and present a good invariance to camera auto-gain and auto-exposure, and illumination changes. Moreover they are fast to extract and match allowing for real-time operation and show good precision/recall performance in bag-of-word place recognition, as shown in Chapter 3.

In the rest of this section we present how stereo/depth information is exploited and which elements of the system are affected.

6.1.1 Monocular, Close Stereo and Far Stereo Keypoints

ORB-SLAM2, as a feature-based method, preprocesses the input to extract features at salient keypoint locations, as shown in Fig. 6.1b. The input images are then discarded and all system operations are based on these features, so that the system is independent on the sensor being stereo or RGB-D. Our system handles monocular and stereo keypoints, which are further classified as close or far.

Stereo keypoints are defined by three coordinates $\mathbf{x}_s = (u_L, v_L, u_R)$, being (u_L, v_L) the coordinates on the left image and u_R the horizontal coordinate in the right image. For stereo cameras, we extract ORB in both images and for every left ORB we search for a match in the right image. This can be done very efficiently assuming stereo rectified images, so that epipolar lines are horizontal. We then generate the stereo keypoint with the coordinates of the left ORB and the horizontal coordinate of the right match, which is subpixel refined by patch correlation. For RGB-D cameras, we extract ORB features on the image channel and, as proposed by Strasdat et al. [74], we synthesize a right coordinate u_R for each feature, using the associated depth value d in the registered depth map channel, and the baseline b_{rgbd} between the structured light projector and the infrared camera, which for Kinect and Asus Xtion cameras we approximate to 8cm:

$$u_R = u - \frac{f_x b_{rgbd}}{d} \tag{6.1}$$

where u is the undistorted horizonatal coordinate of the keypoint in the image and f_x is the horizontal focal length.

A stereo keypoint is classified as **close** if its associated depth is less than 40 times the stereo/RGB-D baseline, as suggested in [64], otherwise it is classified as **far**. Close keypoints can be safely triangulated from one frame, as depth is reliably estimated, and provide scale, translation and rotation information. On the other hand far points provide accurate rotation

information but weaker scale and translation information. We triangulate far points when they are supported by multiple views.

Monocular keypoints are defined by two coordinates $\mathbf{x}_{\mathrm{m}} = (u_L, v_L)$ on the left image and correspond to all those ORB for which a stereo match could not be found or that have an invalid depth value in the RGB-D case. These points are only triangulated from multiple views and do not provide scale information, but contribute to the rotation and translation estimation.

6.1.2 System Bootstrapping

One of the main benefits of using stereo or RGB-D cameras is that, by having depth information from just one frame, we do not need a specific structure from motion initialization as in the monocular case. At system startup we create a keyframe with the first frame, set its pose to the origin, and create an initial map from all stereo keypoints.

6.1.3 Bundle Adjustment with Monocular and Stereo Constraints

Our system performs bundle adjustment to optimize the camera pose in the Tracking (motion-only BA), to optimize a local window of keyframes and points in the Local Mapping (local BA), and after a loop closure to optimize all keyframes and points (full BA). We use the Levenberg-Marquadt implementation in g2o [38].

Motion-only BA optimizes the camera orientation $\mathbf{R}_{\mathtt{cw}} \in SO(3)$ and position ${}_{\mathtt{c}}\mathbf{p}_{\mathtt{w}} \in \mathbb{R}^3$, minimizing the reprojection error between matched 3D points $\mathbf{X}^i_{\mathtt{w}} \in \mathbb{R}^3$ in world coordinates and keypoints $\mathbf{x}^i_{(\cdot)}$, either monocular $\mathbf{x}^i_{\mathrm{m}} \in \mathbb{R}^2$ or stereo $\mathbf{x}^i_{\mathrm{s}} \in \mathbb{R}^3$:

$$\{\mathbf{R}_{\mathtt{cw}}, {}_{\mathtt{c}}\mathbf{p}_{\mathtt{w}}\} = \underset{\mathbf{R}_{\mathtt{cw}}, {}_{\mathtt{c}}\mathbf{p}_{\mathtt{w}}}{\operatorname{argmin}} \sum_i \rho\left(\left\| \mathbf{x}^i_{(\cdot)} - \pi_{(\cdot)}\left(\mathbf{R}_{\mathtt{cw}}\mathbf{X}^i_{\mathtt{w}} + {}_{\mathtt{c}}\mathbf{p}_{\mathtt{w}}\right) \right\|^2_{\mathbf{\Sigma}_i} \right) \tag{6.2}$$

where ρ is the robust Huber cost function and Σ_i the covariance matrix associated to the scale of the keypoint. The projection functions $\pi_{(\cdot)}$, monocular π_{m} and rectified stereo π_{s}, are defined in Section 1.2.

Local BA optimizes a set of covisible keyframes $\mathcal{K}_L$ and all points seen in those keyframes $\mathcal{P}_L$. All other keyframes $\mathcal{K}_F$, not in $\mathcal{K}_L$, observing points in $\mathcal{P}_L$ contribute to the cost function but remain fixed in the optimization. Defining $\mathcal{P}_k$ as the set of points in $\mathcal{P}_L$ observed in keyframe k, the optimization problem is the following:

$$\{\mathbf{X}^p_{\mathtt{w}}, \mathbf{R}^l_{\mathtt{cw}}, {}_{\mathtt{c}}\mathbf{p}^l_{\mathtt{w}} | p \in \mathcal{P}_L, l \in \mathcal{K}_L\} = \underset{\mathbf{X}^p_{\mathtt{w}}, \mathbf{R}^l_{\mathtt{cw}}, {}_{\mathtt{c}}\mathbf{p}^l_{\mathtt{w}}}{\operatorname{argmin}} \sum_{k \in \mathcal{K}_L \cup \mathcal{K}_F} \sum_{j \in \mathcal{P}_k} \rho\left(\left\| \mathbf{x}^j_k - \pi_{(\cdot)}\left(\mathbf{R}^k_{\mathtt{cw}}\mathbf{X}^j_{\mathtt{w}} + {}_{\mathtt{c}}\mathbf{p}^k_{\mathtt{w}}\right) \right\|^2_{\mathbf{\Sigma}^j_k} \right) \tag{6.3}$$

where $\pi_{(\cdot)}$ is the monocular or stereo projection function depending on the matched keypoint $\mathbf{x}^j_k$ being monocular or stereo.

Full BA is the specific case of local BA, where all keyframes and points in the map are optimized, except the origin keyframe that is fixed to eliminate the gauge freedom.

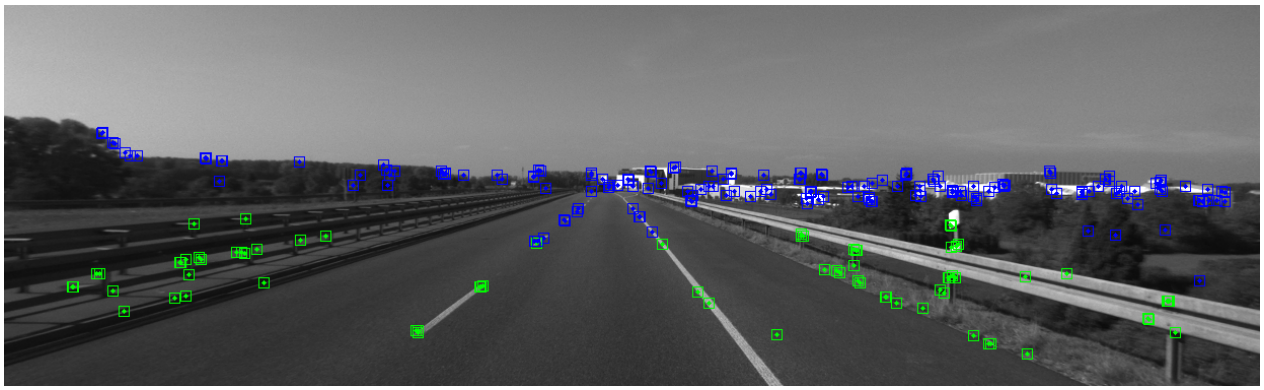


Figure 6.2: Green points have a depth less than 40 times the stereo baseline, while blue points are further away. In this kind of sequences it is important to insert keyframes very often so that the amount of close points allows for accurate translation estimation. Far points contribute to estimate orientation but provide weak information for translation and scale.

6.1.4 Loop Closing and Full BA

Loop closing is performed in two steps, firstly a loop has to be detected and validated, and secondly the loop is corrected optimizing a pose-graph. In contrast to monocular ORB-SLAM, where scale drift may occur [75], the stereo/depth information makes scale observable and the geometric validation and pose-graph optimization no longer require dealing with scale drift and are based on rigid body transformations instead of similarities.

In ORB-SLAM2 we have incorporated a full BA optimization after the pose-graph to achieve the optimal solution. This optimization might be very costly and therefore we perform it in a separate thread, allowing the system to continue creating map and detecting loops. However this brings the challenge of merging the bundle adjustment output with the current state of the map. If a new loop is detected while the optimization is running, we abort the optimization and proceed to close the loop, which will launch the full BA optimization again. When the full BA finishes, we need to merge the updated subset of keyframes and points optimized by the full BA, with the non-updated keyframes and points that where inserted while the optimization was running. This is done by propagating the correction of updated keyframes (i.e. the transformation from the non-optimized to the optimized pose) to non-updated keyframes through the spanning tree. Non-updated points are transformed according to the correction applied to their reference keyframe.

6.1.5 Keyframe Insertion

ORB-SLAM2 follows the policy introduced in monocular ORB-SLAM of inserting keyframes very often and culling redundant ones afterwards. The distinction between close and far stereo points allows us to introduce a new condition for keyframe insertion, which can be critical in challenging environments where a big part of the scene is far from the stereo sensor, as shown in Fig. 6.2. In such environment we need to have a sufficient amount of close points to accurately estimate translation, therefore if the number of tracked close points drops below

τ_t and the frame could create at least τ_c new close stereo points, the system will insert a new keyframe. We empirically found that $\tau_t = 100$ and $\tau_c = 70$ works well in all our experiments.

6.1.6 Localization Mode

We incorporate a localization mode which can be useful for lightweight long-term localization in well mapped areas, as long as there are not significant changes in the environment. In this mode the Local Mapping and Loop Closing threads are deactivated and the camera is continuously localized by the Tracking using relocalization if needed. In this mode the tracking leverages visual odometry matches and matches to map points. Visual odometry matches are matches between ORB in the current frame and 3D points created in the previous frame from the stereo/depth information (i.e. we do not create these point when using a monocular input). These matches make the localization robust to unmapped regions, but drift can be accumulated. Map point matches ensure drift-free localization to the map.

6.2 Evaluation

We have evaluated ORB-SLAM2 in three popular datasets and compared to other state-of-the-art SLAM systems, using always results published by the original authors. We have run ORB-SLAM2 in an Intel Core i7-4790 desktop computer with 16Gb RAM, being the average processing time of the tracking always below the sensor's frame-rate. We have run each sequence 5 times and show always median results, to account for the non-deterministic nature of the multi-threading system. Our open-source implementation includes calibration and instructions to run the system in all these datasets.

6.2.1 KITTI Dataset

The KITTI dataset [26] contains stereo sequences recorded from a car in urban and highway environments. The stereo sensor has a ∼54cm baseline and works at 10Hz with a resolution before rectification of 1392×512 pixels. Sequences 00, 02, 05, 06, 07 and 09 contain loops. Our ORB-SLAM2 detects all loops and is able to reuse its map afterwards, except for sequence 09 where the loop happens in very few frames at the end of the sequence. Table 6.1 shows results in the 11 training sequences, which have public ground-truth, compared to the state-of-the-art Stereo LSD-SLAM [18], to our knowledge the only stereo SLAM showing detailed results for all sequences. We use two different metrics, the absolute translation RMSE t_{abs} proposed in [78], and the average relative translation t_{rel} and rotation r_{rel} errors proposed in [26]. Our system outperforms Stereo LSD-SLAM in most sequences, and achieves in general a relative error lower than 1%. The sequence 01, see Fig. 6.2, is the only highway sequence in the training set and the translation error is slightly worse. Translation is harder to estimate in this sequence because very few close points can be tracked, due to highspeed and low frame-rate. However orientation can be accurately estimated, achieving an error of 0.21 degrees per 100 meters, as there are many far point that can be long tracked. Fig. 6.3 shows some examples of estimated trajectories.

Table 6.1: Comparison of accuracy in the KITTI Dataset.

	ORB-SLAM2 (Stereo)			Stereo LSD-SLAM		
Error (Units)	t_{rel} (%)	r_{rel} (deg/100m)	t_{abs} (m)	t_{rel} (%)	r_{abs} (deg/100m)	t_{abs} (m)
00	0.70	**0.25**	1.3	**0.63**	0.26	**1.0**
01	**1.39**	**0.21**	10.4	2.36	0.36	**9.0**
02	**0.76**	0.23	5.7	0.79	0.23	**2.6**
03	**0.71**	**0.18**	**0.6**	1.01	0.28	1.2
04	0.48	**0.13**	0.2	**0.38**	0.31	0.2
05	**0.40**	**0.16**	**0.8**	0.64	0.18	1.5
06	**0.51**	**0.15**	**0.8**	0.71	0.18	1.3
07	**0.50**	**0.28**	0.5	0.56	0.29	0.5
08	**1.05**	0.32	**3.6**	1.11	**0.31**	3.9
09	**0.87**	0.27	**3.2**	1.14	**0.25**	5.6
10	**0.60**	**0.27**	**1.0**	0.72	0.33	1.5

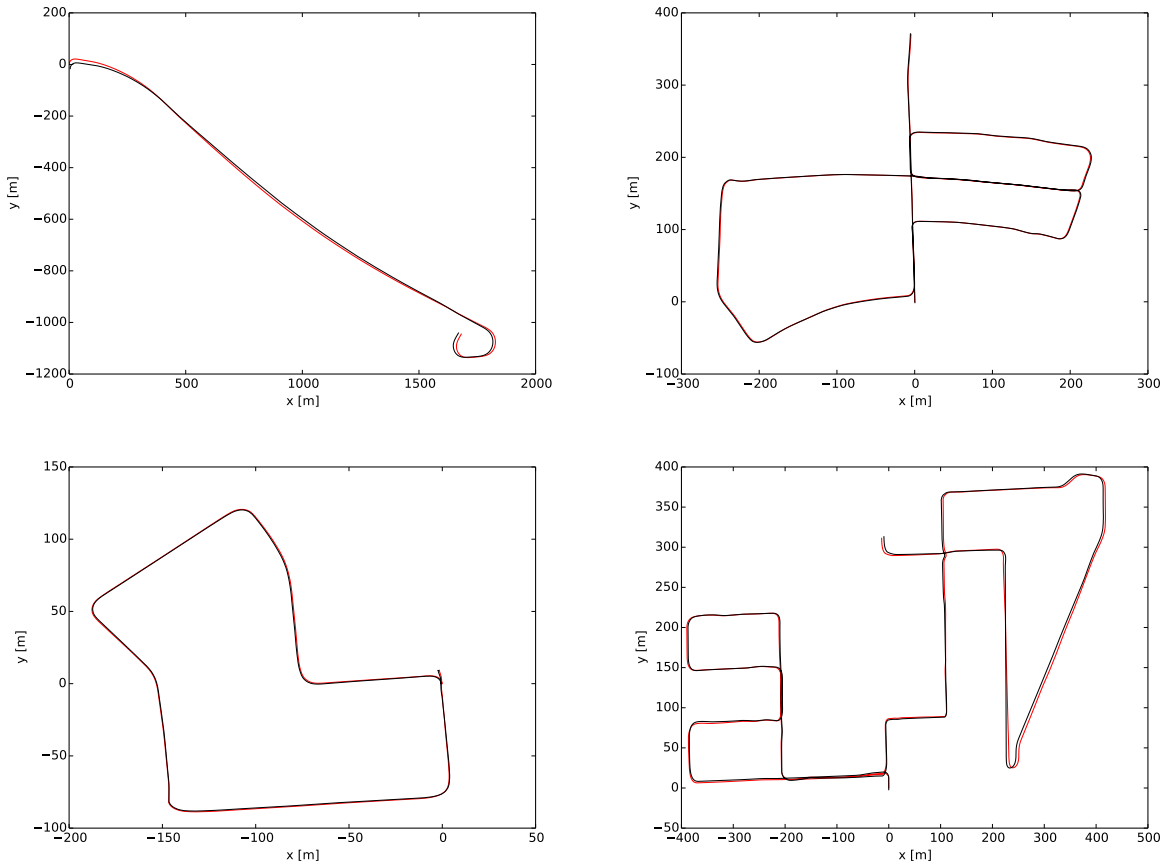


Figure 6.3: Estimated trajectory (black) and ground-truth (red) in KITTI 01, 05, 07 and 08.

6.2.2 EuRoC Dataset

The recent EuRoC dataset [6] contains 11 stereo sequences recorded from a micro aerial vehicle (MAV) flying around two different rooms and a large industrial environment. The stereo sensor has a ∼11cm baseline and provides WVGA images at 20Hz. The sequences are classified as *easy* , *medium* and *difficult* depending on MAV's speed, illumination and scene texture. In all sequences the MAV revisits the environment and ORB-SLAM2 is able to reuse its map, closing loops when necessary. Table 6.2 shows absolute translation RMSE of ORB-SLAM2 for all sequences, comparing to Stereo LSD-SLAM, for the results provided in [18]. ORB-SLAM2 achieves a localization precision of a few centimeters and is more accurate than Stereo LSD-SLAM. Our tracking get lost in some parts of *V2_03_difficult* due to abrupt motions. As shown in Chapter 7, this sequence can be processed using IMU information. Fig. 6.4 shows examples of computed trajectories compared to the ground-truth.

6.2.3 TUM RGB-D Dataset

The TUM RGB-D dataset [78] contains indoors sequences from RGB-D sensors grouped in several categories to evaluate object reconstruction and SLAM/odometry methods under different texture, illumination and structure conditions. We show results in a subset of sequences where most RGB-D methods are usually evaluated. In Table 6.3 we compare our accuracy to the following state-of-the-art methods: ElasticFusion [87], Kintinuous [86], DVO-SLAM [35] and RGB-D SLAM [15]. Our method is the only one based on bundle adjustment and outperforms the other approaches in most sequences. As we already noticed for RGB-D SLAM results in Section 4.6.2, depthmaps for *freiburg2* sequences have a 4% scale bias, probably coming from miscalibration, that we have compensated in our runs and could partly explain our significantly better results. Fig. 6.5 shows the point clouds that result from backprojecting the sensor depth maps from the computed keyframe poses in four sequences. The good definition and the straight contours of desks and posters proves the high accuracy localization of our approach.

Table 6.2: EuRoC Dataset. Comparison of Translation RMSE (m).

	ORB-SLAM2 (Stereo)	Stereo LSD-SLAM
V1_01_easy	**0.035**	0.066
V1_02_medium	**0.020**	0.074
V1_03_difficult	**0.048**	0.089
V2_01_easy	0.037	-
V2_02_medium	0.035	-
V2_03_difficult	X	-
MH_01_easy	0.035	-
MH_02_easy	0.018	-
MH_03_medium	0.028	-
MH_04_difficult	0.119	-
MH_05_difficult	0.060	-

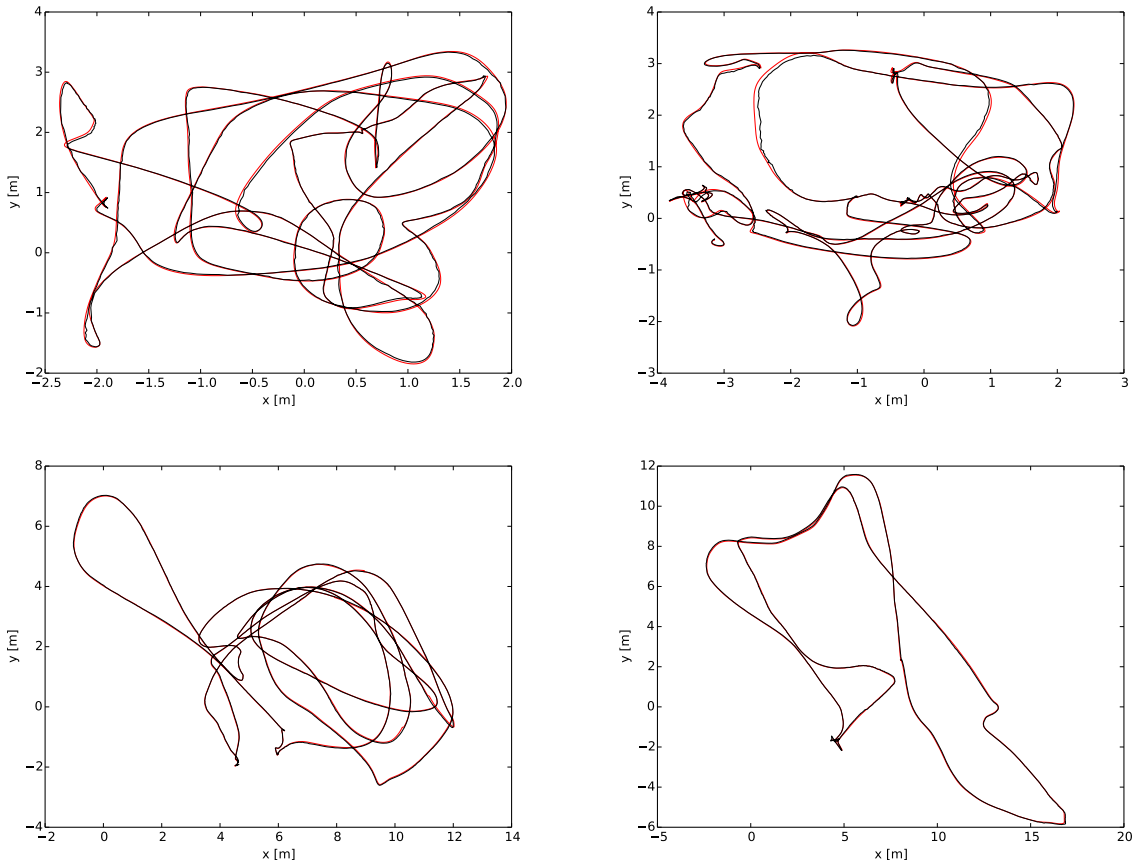


Figure 6.4: Estimated trajectory (black) and groundtruth (red) in EuRoC V1_02_medium, V2_02_medium, MH_03_medium and MH_05_difficutlt.

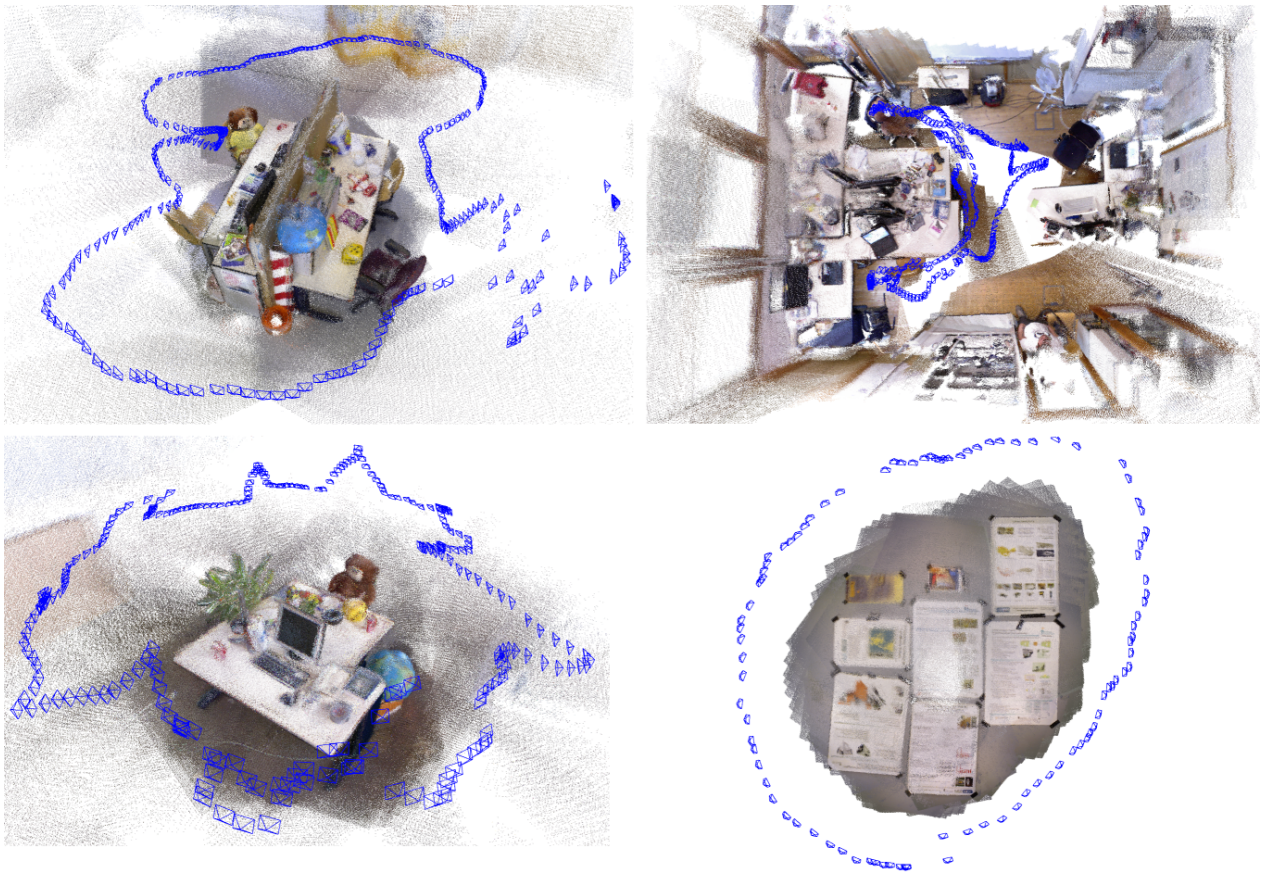


Figure 6.5: Dense pointcloud reconstructions from estimated keyframe poses and sensor depth maps in TUM RGB-D *fr3_office*, *fr1_room*, *fr2_desk* and *fr3_nst*.

Table 6.3: TUM RGB-D Dataset. Comparison of Translation RMSE (m).

	ORB-SLAM2 (RGB-D)	ElasticFusion	Kintinuous	DVO-SLAM	RGB-D SLAM
fr1/desk	**0.016**	0.020	0.037	0.021	0.026
fr1/desk2	**0.022**	0.048	0.071	0.046	-
fr1/room	0.047	0.068	0.075	**0.043**	0.087
fr2/desk	**0.009**	0.071	0.034	0.017	0.057
fr2/xyz	**0.004**	0.011	0.029	0.018	-
fr3/office	**0.010**	0.017	0.030	0.035	-
fr3/nst	0.019	**0.016**	0.031	0.018	-

6.3 Discussion

We have presented a full SLAM system for monocular, stereo and RGB-D sensors, able to perform relocalization, loop closing and reuse its map in real-time in standard CPUs. We focus on building globally consistent maps for reliable localization in a wide range of environments as demonstrated in the experiments. The comparison to the state-of-the-art shows very competitive accuracy of ORB-SLAM2, being in most cases the most accurate solution. Surprisingly our RGB-D results demonstrate that if the most accurate camera localization is desired, bundle adjustment performs better than direct methods or ICP, with the additional advantage of being less computationally expensive. We have released the source code of our system, with examples and instructions so that it can be easily used by other researchers. We are aware that it has been already used out-of-the-box in [79]. Future extensions might include, to name some examples, non-overlapping multi-camera, fisheye or omnidirectional cameras support, large scale dense fusion, cooperative mapping or increased motion blur robustness.

Chapter 7

Visual-Inertial Monocular ORB-SLAM

In previous chapters we addressed visual SLAM using monocular, stereo and RGB-D cameras. Among different sensor modalities, visual-inertial setups provide a cheap solution with great potential. On the one hand, cameras provide rich information of the environment, which allows to build 3D models, localize the camera and recognize already visited places. On the other hand, IMU sensors provide self-motion information, allowing to recover metric scale for monocular vision, and to estimate gravity direction, rendering absolute pitch and roll of the sensor observable. In this chapter we present a novel visual-inertial SLAM system fusing information from monocular vision and IMU sensors, being to the best of our knowledge the first keyframe-based visual-inertial SLAM that is able to close loops and reuse its map.

7.1 IMU Preintegration

In Section 1.2.4 we have already introduced the IMU sensor and how to integrate measurements to compute the motion of the sensor. It is also very useful to describe the motion between two consecutive keyframes in terms of the preintegration $\Delta\mathbf{R}$, $\Delta\mathbf{v}$ and $\Delta\mathbf{p}$ from all measurements in-between as proposed by Lupton and Sukkarieh [46]. We use the recent preintegration on SO(3) manifold for rotations, as described by Forster et al. [21]:

$$\begin{aligned}
\mathbf{R}_{\mathtt{WB}}^{i+1} &= \mathbf{R}_{\mathtt{WB}}^{i}\Delta\mathbf{R}_{i,i+1}\mathrm{Exp}\left(\left(\mathbf{J}_{\Delta R}^{g}\mathbf{b}_{g}^{i}\right)\right) \\
{}_{\mathtt{W}}\mathbf{v}_{\mathtt{B}}^{i+1} &= {}_{\mathtt{W}}\mathbf{v}_{\mathtt{B}}^{i} + \mathbf{g}_{\mathtt{W}}\Delta t_{i,i+1} + \mathbf{R}_{\mathtt{WB}}^{i}\left(\Delta\mathbf{v}_{i,i+1} + \mathbf{J}_{\Delta v}^{g}\mathbf{b}_{g}^{i} + \mathbf{J}_{\Delta v}^{a}\mathbf{b}_{a}^{i}\right) \\
{}_{\mathtt{W}}\mathbf{p}_{\mathtt{B}}^{i+1} &= {}_{\mathtt{W}}\mathbf{p}_{\mathtt{B}}^{i} + {}_{\mathtt{W}}\mathbf{v}_{\mathtt{B}}^{i}\Delta t_{i,i+1} + \frac{1}{2}\mathbf{g}_{\mathtt{W}}\Delta t_{i,i+1}^{2} + \mathbf{R}_{\mathtt{WB}}^{i}\left(\Delta\mathbf{p}_{i,i+1} + \mathbf{J}_{\Delta p}^{g}\mathbf{b}_{g}^{i} + \mathbf{J}_{\Delta p}^{a}\mathbf{b}_{a}^{i}\right)
\end{aligned} \tag{7.1}$$

where the Jacobians $\mathbf{J}_{(\cdot)}^{a}$ and $\mathbf{J}_{(\cdot)}^{g}$ account for a first-order approximation of the effect of changing the biases without explicitly recomputing the preintegrations. Both preintegrations and Jacobians can be efficiently computed iteratively as IMU measurements arrive [21].

7.2 System Description

The base of our visual-inertial system is our monocular ORB-SLAM, following the ORB-SLAM2 framework described in Chapter 6, including the Full BA thread and the localization mode. In this section we detail the main changes with respect to the original system.

7.2.1 Initialization

Our tracking and local BA fix states in their optimizations, which could potentially bias the solution. For this reason we need a reliable visual-inertial initialization that provides accurate state estimations before we start fixing states. To this end we propose to perform a visual-inertial full BA that provides the optimal solution for structure, camera poses, scale, velocities, gravity, and gyroscope and accelerometer biases. This full BA is a non-linear optimization that requires a good initial seed to converge. We propose in Section 7.3 a divide and conquer approach to compute this initial solution. We firstly process a few seconds of video with our pure monocular ORB-SLAM to estimate an initial solution for structure and several keyframe poses, up to an unknown scale factor. We then compute the bias of the gyroscope, which can be easily estimated from the known orientation of the keyframes, so that we can correctly rotate the accelerometer measurements. Then we solve scale and gravity without considering the accelerometer bias, using an approach inspired by Lupton and Sukkarieh [46]. To facilitate distinguishing between gravity and accelerometer bias, we use the knowledge of the magnitude of the gravity and solve for accelerometer bias, refining scale and gravity direction. At this point it is straightforward to retrieve the velocities for all keyframes. Our experiments validate that this is an efficient, reliable and accurate initialization method. Moreover it is general, could be applied to any keyframe-based monocular SLAM, does not assume any initial condition, and just require a movement of the sensor that make all variables observable [48]. While previous approaches [33, 48, 91] jointly solve vision and IMU, either ignoring gyroscope or accelerometer biases, we efficiently compute all variables subdividing the problem in simpler steps.

7.2.2 Tracking

Our visual-inertial tracking is in charge of tracking sensor pose, velocity and IMU biases, at frame-rate. This allows us to predict the camera pose very reliably, instead of using an ad-hoc motion model as in the original monocular system. Once the camera pose is predicted, the map points in the local map are projected and matched with keypoints on the frame. We then optimize current frame j by minimizing the feature reprojection error of all matched points and an IMU error term. This optimization is different depending on the map being updated or not by the Local Mapping or the Loop Closing thread, as illustrated in Fig. 7.1.

When tracking is performed just after a map update (Fig. 7.1a) the IMU error term links

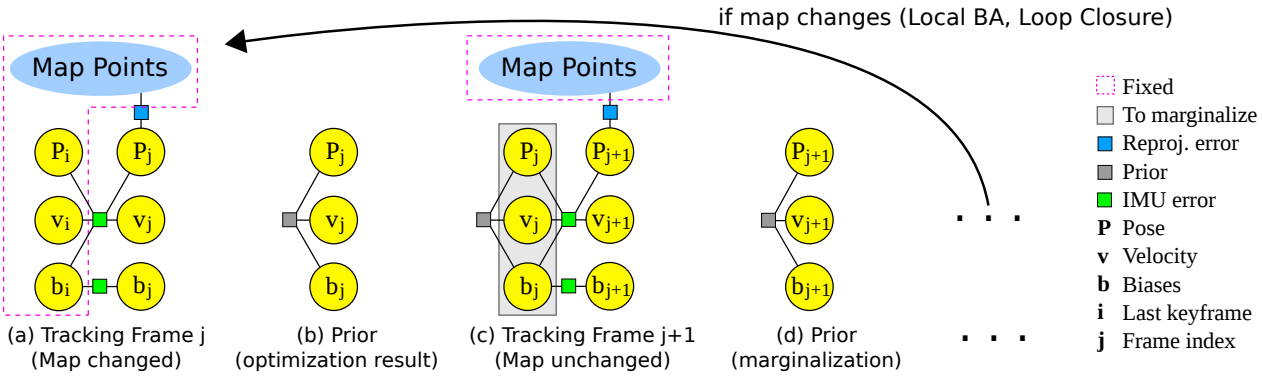


Figure 7.1: Evolution of the optimization in the Tracking thread. (a) We start optimizing the frame j linked by an IMU constraint to last keyframe i. (b) The result of the optimization (estimation and Hessian matrix) serves as prior for next optimization. (c) When tracking next frame $j+1$, both frames j and $j+1$ are jointly optimized, being linked by an IMU constraint, and having frame j the prior from previous optimization. (d) At the end of the optimization, the frame j is marginalized out and the result serves as prior for following optimization. This process is repeated until there is a map update from the Local Mapping or Loop Closing thread. In such case the optimization links the current frame to last keyframe discarding the prior, which is not valid after the map change.

current frame j with last keyframe i:

$$\begin{aligned} \theta &= \left\{ \mathbf{R}^j_{\mathtt{WB}}, {}_{\mathtt{W}}\mathbf{p}^j_{\mathtt{B}}, {}_{\mathtt{W}}\mathbf{v}^j_{\mathtt{B}}, \mathbf{b}^j_g, \mathbf{b}^j_a \right\} \\ \theta^* &= \underset{\theta}{\operatorname{argmin}} \left(\sum_k \mathbf{E}_{\text{proj}}(k,j) + \mathbf{E}_{\text{IMU}}(i,j) \right) \end{aligned} \tag{7.2}$$

where the feature reprojection error $\mathbf{E}_{\text{proj}}$ for a given match k, is defined as follows:

$$\begin{aligned} \mathbf{E}_{\text{proj}}(k,j) &= \rho \left(\left(\mathbf{x}^k - \pi(\mathbf{X}^k_{\mathtt{C}}) \right)^T \boldsymbol{\Sigma}_{\boldsymbol{k}} \left(\mathbf{x}^k - \pi(\mathbf{X}^k_{\mathtt{C}}) \right) \right) \\ \mathbf{X}^k_{\mathtt{C}} &= \mathbf{R}_{\mathtt{CB}} \mathbf{R}^j_{\mathtt{BW}} \left(\mathbf{X}^k_{\mathtt{W}} - {}_{\mathtt{W}}\mathbf{p}^j_{\mathtt{B}} \right) + {}_{\mathtt{C}}\mathbf{p}_{\mathtt{B}} \end{aligned} \tag{7.3}$$

where $\mathbf{x}^k$ is the keypoint location in the image, $\mathbf{X}^k_{\mathtt{W}}$ the map point in world coordinates, and $\boldsymbol{\Sigma}_{\boldsymbol{k}}$ the information matrix associated to the keypoint scale. The IMU error term $\mathbf{E}_{\text{IMU}}$ is:

$$\begin{aligned} \mathbf{E}_{\text{IMU}}(i,j) &= \rho \left(\left[\mathbf{e}^T_R \, \mathbf{e}^T_v \, \mathbf{e}^T_p \right] \boldsymbol{\Sigma}_I \left[\mathbf{e}^T_R \, \mathbf{e}^T_v \, \mathbf{e}^T_p \right]^T \right) + \rho \left(\mathbf{e}^T_b \boldsymbol{\Sigma}_R \mathbf{e}_b \right) \\ \mathbf{e}_R &= \text{Log} \left(\left(\Delta\mathbf{R}_{ij} \text{Exp} \left(\mathbf{J}^g_{\Delta R} \mathbf{b}^j_g \right) \right)^T \mathbf{R}^i_{\mathtt{BW}} \mathbf{R}^j_{\mathtt{WB}} \right) \\ \mathbf{e}_v &= \mathbf{R}^i_{\mathtt{BW}} \left({}_{\mathtt{W}}\mathbf{v}^j_{\mathtt{B}} - {}_{\mathtt{W}}\mathbf{v}^i_{\mathtt{B}} - \mathbf{g}_{\mathtt{W}} \Delta t_{ij} \right) - \left(\Delta\mathbf{v}_{ij} + \mathbf{J}^g_{\Delta v} \mathbf{b}^j_g + \mathbf{J}^a_{\Delta v} \mathbf{b}^j_a \right) \\ \mathbf{e}_p &= \mathbf{R}^i_{\mathtt{BW}} \left({}_{\mathtt{W}}\mathbf{p}^j_{\mathtt{B}} - {}_{\mathtt{W}}\mathbf{p}^i_{\mathtt{B}} - {}_{\mathtt{W}}\mathbf{v}^i_{\mathtt{B}} \Delta t_{ij} - \frac{1}{2} \mathbf{g}_{\mathtt{W}} \Delta t^2_{ij} \right) - \left(\Delta\mathbf{p}_{ij} + \mathbf{J}^g_{\Delta p} \mathbf{b}^j_g + \mathbf{J}^a_{\Delta p} \mathbf{b}^j_a \right) \\ \mathbf{e}_b &= \mathbf{b}^j - \mathbf{b}^i \end{aligned} \tag{7.4}$$

where $\boldsymbol{\Sigma}_I$ is the information matrix of the preintegration and $\boldsymbol{\Sigma}_R$ of the bias random walk [21], and ρ is the Huber robust cost function. We solve this optimization problem with

Gauss-Newton algorithm implemented in g2o [38]. After the optimization (Fig. 7.1b) the resulting estimation and Hessian matrix serves as prior for next optimization.

Assuming no map update (Fig. 7.1c), the next frame $j+1$ will be optimized with a link to frame j and using the prior computed at the end of the previous optimization (Fig 7.1b):

$$\begin{aligned} \theta &= \left\{ \mathbf{R}^j_{\mathtt{WB}}, \mathbf{p}^j_{\mathtt{W}}, \mathbf{v}^j_{\mathtt{W}}, \mathbf{b}^j_g, \mathbf{b}^j_a, \mathbf{R}^{j+1}_{\mathtt{WB}}, \mathbf{p}^{j+1}_{\mathtt{W}}, \mathbf{v}^{j+1}_{\mathtt{W}}, \mathbf{b}^{j+1}_g, \mathbf{b}^{j+1}_a \right\} \\ \theta^* &= \underset{\theta}{\operatorname{argmin}} \left(\sum_k \mathbf{E}_{\text{proj}}(k, j+1) + \mathbf{E}_{\text{IMU}}(j, j+1) + \mathbf{E}_{\text{prior}}(j) \right) \end{aligned} \tag{7.5}$$

where $\mathbf{E}_{\text{prior}}$ is a prior term:

$$\begin{aligned} \mathbf{E}_{\text{prior}}(j) &= \rho \left(\left[\mathbf{e}_R^T \, \mathbf{e}_v^T \, \mathbf{e}_p^T \, \mathbf{e}_b^T \right] \boldsymbol{\Sigma}_p \left[\mathbf{e}_R^T \, \mathbf{e}_v^T \, \mathbf{e}_p^T \, \mathbf{e}_b^T \right]^T \right) \\ \mathbf{e}_R &= \operatorname{Log} \left(\bar{\mathbf{R}}^j_{\mathtt{BW}} \mathbf{R}^j_{\mathtt{WB}} \right) \quad \mathbf{e}_v = {}_{\mathtt{W}}\bar{\mathbf{v}}^j_{\mathtt{B}} - {}_{\mathtt{W}}\mathbf{v}^j_{\mathtt{B}} \\ \mathbf{e}_p &= {}_{\mathtt{W}}\bar{\mathbf{p}}^j_{\mathtt{B}} - {}_{\mathtt{W}}\mathbf{p}^j_{\mathtt{B}} \qquad \mathbf{e}_b = \bar{\mathbf{b}}^j - \mathbf{b}^j \end{aligned} \tag{7.6}$$

where $\bar{(\cdot)}$ and $\boldsymbol{\Sigma}_p$ are the estimated states and Hessian matrix resulting from previous optimization (Fig. 7.1b). After this optimization (Fig. 7.1d), frame j is marginalized out [41]. This optimization linking two consecutive frames and using a prior is repeated until a map change, when the prior will be no longer valid and the tracking will link again the current frame to the last keyframe (Fig. 7.1a). Note that this is the optimization, Fig 7.1 (c-d), that is always performed in *Localization Mode*, as the map is not updated.

7.2.3 Local Mapping

The Local Mapping thread performs local BA after a new keyframe insertion. It optimizes the last N keyframes (local window) and all points seen by those N keyframes. All other keyframes that share observations of local points (i.e. are connected in the covisibility graph to any local keyframe), but are not in the local window, contribute to the total cost but are fixed during optimization (fixed window). The keyframe $N+1$ is always included in the fixed window as it constraints the IMU states. Fig. 7.2 illustrates the differences between local BA in original ORB-SLAM and Visual-Inertial ORB-SLAM. The cost function is a combination of IMU error terms (7.4) and reprojection error terms (7.3). Note that the visual-inertial version, compared to the vision-only, is more complex as there are 9 additional states (velocity and biases) to optimize per keyframe. A suitable local window size has to be chosen for real-time performance.

The Local Mapping is also in charge of keyframe management. The original ORB-SLAM policy discards redundant keyframes, so that map size does not grow if localizing in a well mapped area. This policy is problematic when using IMU information, which constrains the motion of consecutive keyframes. The longer the temporal difference between consecutive keyframes, the weaker information IMU provides. Therefore we allow the mapping to discard redundant keyframes, if that does not make two consecutive keyframes in the local window of local BA to differ more than 0.5s. To be able to perform full BA, after a loop closure or at any time to refine a map, we do not allow any two consecutive keyframes to differ more than 3s. If we switched-off full BA with IMU constraints, we would only need to restrict the temporal offset between keyframes in the local window.

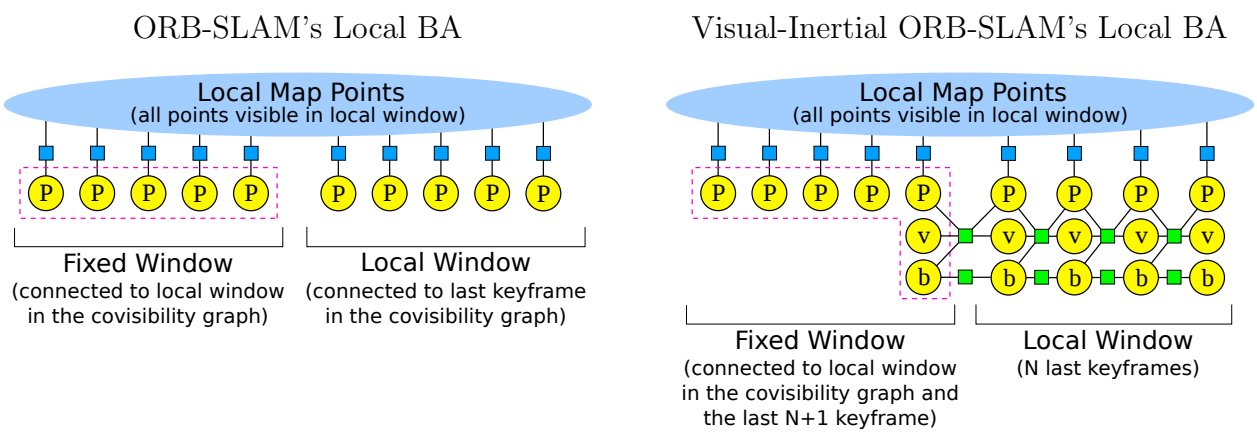


Figure 7.2: Comparison of Local Bundle Adjustment between original ORB-SLAM (left) and proposed Visual-Inertial ORB-SLAM (right). The local window in Visual-Inertial ORB-SLAM is retrieved by temporal order of keyframes, while in ORB-SLAM is retrieved using the covisibility graph.

7.2.4 Loop Closing

The loop closing thread aims to reduce the drift accumulated during exploration, when returning to an already mapped area. The place recognition module matches a recent keyframe with a past keyframe. This match is validated computing a rigid body transformation that aligns matched points between keyframes [29]. Finally an optimization is carried out to reduce the error accumulated in the trajectory. This optimization might be very costly in large maps, therefore the strategy is to perform a pose-graph optimization, which reduces the complexity, as structure is ignored, and exhibits good convergence as shown in Section 4.6.5. In contrast to the original ORB-SLAM, we perform the pose-graph on 6 Degrees of Freedom (DoF) instead of 7 DoF [75], as scale is observable. This pose-graph ignores IMU information, not optimizing velocity or IMU biases. Therefore we correct velocities by rotating them according to the corrected orientation of the associated keyframe. While this is not optimal, biases and velocities should be locally accurate to continue using IMU information right after pose-graph optimization. We perform afterwards a full BA in a parallel thread that optimizes all states, including velocities and biases.

7.3 IMU Initialization

We propose in this section a method to compute an initial estimation for a visual-inertial full BA of the scale, gravity direction, velocity and IMU biases, given a set of keyframes processed by a monocular SLAM algorithm. The idea is to run the monocular SLAM for a few seconds, assuming the sensor performs a motion that makes all variables observable. While we build on ORB-SLAM, any other SLAM could be used. The only requirement is that any two consecutive keyframes are close in time (see Section 7.2.3), to reduce IMU noise integration.

The initialization is divided in simpler subproblems: (1) gyroscope bias estimation, (2)

scale and gravity approximation, considering no accelerometer bias, (3) accelerometer bias estimation, and scale and gravity direction refinement, and (4) velocity estimation.

7.3.1 Gyroscope Bias Estimation

Gyroscope bias can be estimated just from the known orientation of two consecutive keyframes. Assuming a negligible bias change, we optimize a constant bias $\mathbf{b}_g$, which minimizes the difference between gyroscope integration and relative orientation computed from ORB-SLAM, for all two consecutive keyframes:

$$\underset{\mathbf{b}_g}{\operatorname{argmin}} \sum_{i=1}^{N-1} \left\| \operatorname{Log} \left(\left(\Delta\mathbf{R}_{i,i+1} \operatorname{Exp} \left(\mathbf{J}^g_{\Delta R} \mathbf{b}_g \right) \right)^T \mathbf{R}^{i+1}_{\mathtt{BW}} \mathbf{R}^{i}_{\mathtt{WB}} \right) \right\|^2 \tag{7.7}$$

where N is the number of keyframes. $\mathbf{R}^{(\cdot)}_{\mathtt{WB}} = \mathbf{R}^{(\cdot)}_{\mathtt{WC}} \mathbf{R}_{\mathtt{CB}}$ is computed from the orientation $\mathbf{R}^{(\cdot)}_{\mathtt{WC}}$ computed by ORB-SLAM and calibration $\mathbf{R}_{\mathtt{CB}}$. $\Delta\mathbf{R}_{i,i+1}$ is the gyroscope integration between two consecutive keyframes. We solve (7.7) with Gauss-Newton with a zero bias seed. Analytic jacobians for a similar expression can be found in [21].

7.3.2 Scale and Gravity Approximation (no accelerometer bias)

Once we have estimated the gyroscope bias, we can preintegrate velocities and positions, rotating correctly the acceleration measurements compensating the gyroscope bias.

The scale of the camera trajectory computed by ORB-SLAM is arbitrary. Therefore we need to include a scale factor s when transforming between camera $\mathtt{C}$ and IMU $\mathtt{B}$ coordinate systems:

$$_{\mathtt{W}}\mathbf{p}_{\mathtt{B}} = s\, _{\mathtt{W}}\mathbf{p}_{\mathtt{C}} + \mathbf{R}_{\mathtt{WC}}\, _{\mathtt{C}}\mathbf{p}_{\mathtt{B}} \tag{7.8}$$

Substituting (7.8) into the equation relating position of two consecutive keyframes (7.1), and neglecting at this point accelerometer bias, it follows:

$$s\, _{\mathtt{W}}\mathbf{p}^{i+1}_{\mathtt{C}} = s\, _{\mathtt{W}}\mathbf{p}^{i}_{\mathtt{C}} + _{\mathtt{W}}\mathbf{v}^{i}_{\mathtt{B}} \Delta t_{i,i+1} + \frac{1}{2}\mathbf{g}_{\mathtt{W}} \Delta t^2_{i,i+1} + \mathbf{R}^{i}_{\mathtt{WB}} \Delta\mathbf{p}_{i,i+1} + \left(\mathbf{R}^{i}_{\mathtt{WC}} - \mathbf{R}^{i+1}_{\mathtt{WC}} \right) _{\mathtt{C}}\mathbf{p}_{\mathtt{B}} \tag{7.9}$$

The goal is to estimate s and $\mathbf{g}_{\mathtt{W}}$ by solving a linear system of equations on those variables. To avoid solving for N velocities, and reduce complexity, we consider two relations (7.9) between three consecutive keyframes and use velocity relation in (7.1), which results in the following expression:

$$\begin{bmatrix} \boldsymbol{\lambda}(i) & \boldsymbol{\beta}(i) \end{bmatrix} \begin{bmatrix} s \\ \mathbf{g}_{\mathtt{W}} \end{bmatrix} = \boldsymbol{\gamma}(i) \tag{7.10}$$

where, writing keyframes $i, i+1, i+2$ as $1, 2, 3$ for clarity of notation, we have:

$$\begin{aligned} \boldsymbol{\lambda}(i) &= \left(_{\mathtt{W}}\mathbf{p}^2_{\mathtt{C}} - _{\mathtt{W}}\mathbf{p}^1_{\mathtt{C}} \right) \Delta t_{23} - \left(_{\mathtt{W}}\mathbf{p}^3_{\mathtt{C}} - _{\mathtt{W}}\mathbf{p}^2_{\mathtt{C}} \right) \Delta t_{12} \\ \boldsymbol{\beta}(i) &= \frac{1}{2}\mathbf{I}_{3\times 3} \left(\Delta t^2_{12} \Delta t_{23} + \Delta t^2_{23} \Delta t_{12} \right) \\ \boldsymbol{\gamma}(i) &= \left(\mathbf{R}^2_{\mathtt{WC}} - \mathbf{R}^1_{\mathtt{WC}} \right) _{\mathtt{C}}\mathbf{p}_{\mathtt{B}} \Delta t_{23} - \left(\mathbf{R}^3_{\mathtt{WC}} - \mathbf{R}^2_{\mathtt{WC}} \right) _{\mathtt{C}}\mathbf{p}_{\mathtt{B}} \Delta t_{12} \\ &\quad + \mathbf{R}^2_{\mathtt{WB}} \Delta\mathbf{p}_{23} \Delta t_{12} + \mathbf{R}^1_{\mathtt{WB}} \Delta\mathbf{v}_{12} \Delta t_{12} \Delta t_{23} \\ &\quad - \mathbf{R}^1_{\mathtt{WB}} \Delta\mathbf{p}_{12} \Delta t_{23} \end{aligned} \tag{7.11}$$

We stack then all relations of three consecutive keyframes (7.10) into a system $\mathbf{A}_{3(N-2)\times 4}\,\mathbf{x}_{4\times 1} = \mathbf{B}_{3(N-2)\times 1}$ which can be solved via Singular Value Decomposition (SVD) to get the scale factor s^* and gravity vector $\mathbf{g}^*_{\mathtt{W}}$. Note that we have $3(N-2)$ equations and 4 unknowns, therefore we need at least 4 keyframes.

7.3.3 Accelerometer Bias Estimation, and Scale and Gravity Refinement

So far we have not considered accelerometer bias when computing scale and gravity. Just incorporating accelerometer biases in (7.10) will heavily increase the chance of having an ill-conditioned system, because gravity and accelerometer biases are hard to distinguish [48]. To increase observability we introduce new information we did not consider so far, which is the gravity magnitude G. Consider an inertial reference I with the gravity direction $\hat{\mathbf{g}}_{\mathtt{I}} = \{0, 0, -1\}$, and the already computed gravity direction $\hat{\mathbf{g}}_{\mathtt{W}} = \mathbf{g}^*_{\mathtt{W}}/\|\mathbf{g}^*_{\mathtt{W}}\|$. We can compute rotation $\mathbf{R}_{\mathtt{WI}}$ as follows:

$$\begin{aligned} \mathbf{R}_{\mathtt{WI}} &= \mathrm{Exp}(\hat{\mathbf{v}}\theta) \\ \hat{\mathbf{v}} = \frac{\hat{\mathbf{g}}_{\mathtt{I}} \times \hat{\mathbf{g}}_{\mathtt{W}}}{\|\hat{\mathbf{g}}_{\mathtt{I}} \times \hat{\mathbf{g}}_{\mathtt{W}}\|}, &\quad \theta = \mathrm{atan2}\left(\|\hat{\mathbf{g}}_{\mathtt{I}} \times \hat{\mathbf{g}}_{\mathtt{W}}\|, \hat{\mathbf{g}}_{\mathtt{I}} \cdot \hat{\mathbf{g}}_{\mathtt{W}}\right) \end{aligned} \tag{7.12}$$

and express now gravity vector as:

$$\mathbf{g}_{\mathtt{W}} = \mathbf{R}_{\mathtt{WI}}\,\hat{\mathbf{g}}_{\mathtt{I}}\,G \tag{7.13}$$

where $\mathbf{R}_{\mathtt{WI}}$ can be parametrized with just two angles around x and y axes in $\mathtt{I}$, because a rotation around z axis, which is aligned with gravity, has no effect in $\mathbf{g}_{\mathtt{W}}$. This rotation can be optimized using a perturbation $\boldsymbol{\delta\theta}$:

$$\begin{aligned} \mathbf{g}_{\mathtt{W}} &= \mathbf{R}_{\mathtt{WI}}\mathrm{Exp}(\boldsymbol{\delta\theta})\,\hat{\mathbf{g}}_{\mathtt{I}}\,G \\ \boldsymbol{\delta\theta} = \left[\boldsymbol{\delta\theta}^{\boldsymbol{T}}_{\boldsymbol{xy}}\ 0\right]^T, &\quad \boldsymbol{\delta\theta}_{\boldsymbol{xy}} = [\delta\theta_x\ \delta\theta_y]^T \end{aligned} \tag{7.14}$$

with a first-order approximation:

$$\mathbf{g}_{\mathtt{W}} \approx \mathbf{R}_{\mathtt{WI}}\,\hat{\mathbf{g}}_{\mathtt{I}}\,G - \mathbf{R}_{\mathtt{WI}}\,(\hat{\mathbf{g}}_{\mathtt{I}})_{\times} G\,\boldsymbol{\delta\theta} \tag{7.15}$$

Substituting (7.15) in (7.9) and including now the effect of accelerometer bias, we obtain:

$$\begin{aligned} s\,{}_{\mathtt{W}}\mathbf{p}^{i+1}_{\mathtt{C}} = s\,{}_{\mathtt{W}}\mathbf{p}^{i}_{\mathtt{C}} + {}_{\mathtt{W}}\mathbf{v}^{i}_{\mathtt{B}}\Delta t_{i,i+1} - \frac{1}{2}\mathbf{R}_{\mathtt{WI}}\,(\hat{\mathbf{g}}_{\mathtt{I}})_{\times} G \Delta t^2_{i,i+1}\,\boldsymbol{\delta\theta} \\ + \mathbf{R}^{i}_{\mathtt{WB}}\left(\Delta\mathbf{p}_{i,i+1} + \mathbf{J}^{a}_{\Delta p}\mathbf{b}_a\right) + \left(\mathbf{R}^{i}_{\mathtt{WC}} - \mathbf{R}^{i+1}_{\mathtt{WC}}\right){}_{\mathtt{C}}\mathbf{p}_{\mathtt{B}} \\ + \frac{1}{2}\mathbf{R}_{\mathtt{WI}}\,\hat{\mathbf{g}}_{\mathtt{I}}\,G\Delta t^2_{i,i+1} \end{aligned} \tag{7.16}$$

Considering three consecutive keyframes as in (7.10) we can eliminate velocities and get the following relation:

$$\begin{bmatrix} \boldsymbol{\lambda}(i) & \boldsymbol{\phi}(i) & \boldsymbol{\zeta}(i) \end{bmatrix} \begin{bmatrix} s \\ \boldsymbol{\delta\theta}_{\boldsymbol{xy}} \\ \mathbf{b}_a \end{bmatrix} = \boldsymbol{\psi}(i) \tag{7.17}$$

where $\boldsymbol{\lambda}(i)$ remains the same as in (7.11), and $\boldsymbol{\phi}(i)$, $\boldsymbol{\zeta}(i)$, and $\boldsymbol{\psi}(i)$ are computed as follows:

$$\begin{aligned}
\boldsymbol{\phi}(i) &= \left[\frac{1}{2}\mathbf{R}_{\mathtt{WI}}\left(\hat{\mathbf{g}}_{\mathtt{I}}\right)_{\times} G\left(\Delta t_{12}^2 \Delta t_{23} + \Delta t_{23}^2 \Delta t_{12}\right)\right]_{(:,1:2)} \\
\boldsymbol{\zeta}(i) &= \mathbf{R}_{\mathtt{WB}}^2 \mathbf{J}_{\Delta p_{23}}^a \Delta t_{12} + \mathbf{R}_{\mathtt{WB}}^1 \mathbf{J}_{\Delta v_{23}}^a \Delta t_{12} \Delta t_{23} \\
&\quad - \mathbf{R}_{\mathtt{WB}}^1 \mathbf{J}_{\Delta p_{12}}^a \Delta t_{23} \\
\boldsymbol{\psi}(i) &= \left(\mathbf{R}_{\mathtt{WC}}^2 - \mathbf{R}_{\mathtt{WC}}^1\right) {}_{\mathtt{C}}\mathbf{p}_{\mathtt{B}} \Delta t_{23} - \left(\mathbf{R}_{\mathtt{WC}}^3 - \mathbf{R}_{\mathtt{WC}}^2\right) {}_{\mathtt{C}}\mathbf{p}_{\mathtt{B}} \Delta t_{12} \\
&\quad + \mathbf{R}_{\mathtt{WB}}^2 \Delta \mathbf{p}_{23} \Delta t_{12} + \mathbf{R}_{\mathtt{WB}}^1 \Delta \mathbf{v}_{12} \Delta t_{12} \Delta t_{23} \\
&\quad - \mathbf{R}_{\mathtt{WB}}^1 \Delta \mathbf{p}_{12} \Delta t_{23} + \frac{1}{2}\mathbf{R}_{\mathtt{WI}}\, \hat{\mathbf{g}}_{\mathtt{I}}\, G \Delta t_{ij}^2
\end{aligned} \tag{7.18}$$

where $[\,]_{(:,1:2)}$ means the first two columns of the matrix. Stacking all relations between three consecutive keyframes (7.17) we form a linear system of equations $\mathbf{A}_{3(N-2)\times 6}\,\mathbf{x}_{6\times 1} = \mathbf{B}_{3(N-2)\times 1}$ which can be solved via SVD to get the scale factor s^*, gravity $\mathbf{g}_{\mathtt{W}}^*$ by using the correction $\boldsymbol{\delta\theta}_{\boldsymbol{xy}}^*$ in (7.14), and accelerometer bias $\mathbf{b}_a^*$. In this case we have $3(N-2)$ equations and 6 unknowns and we need again at least 4 keyframes to solve the system. We can compute the condition number (i.e. the ratio between the maximum and minimum singular value) to check if the problem is well-conditioned (i.e. the sensor has performed a motion that makes all variables observable). We could relinearize (7.15) and iterate the solution, but in practice we found that a second iteration does not produce a noticeable improvement.

7.3.4 Velocity Estimation

We considered relations of three consecutive keyframes in equations (7.10) and (7.17), so that the resulting linear systems do not have the $3N$ additional unknowns corresponding to velocities. The velocities for all keyframes can now be computed using equation (7.16), as scale, gravity and bias are known.

$${}_{\mathtt{W}}\mathbf{v}_{\mathtt{B}}^i = \frac{s^*\left({}_{\mathtt{W}}\mathbf{p}_{\mathtt{C}}^{i+1} - {}_{\mathtt{W}}\mathbf{p}_{\mathtt{C}}^i\right) - \frac{1}{2}\mathbf{g}_{\mathtt{W}}^* \Delta t_{i,i+1}^2 - \mathbf{R}_{\mathtt{WB}}^i\left(\Delta \mathbf{p}_{i,i+1} + \mathbf{J}_{\Delta p}^a \mathbf{b}_a^*\right) - \left(\mathbf{R}_{\mathtt{WC}}^i - \mathbf{R}_{\mathtt{WC}}^{i+1}\right) {}_{\mathtt{C}}\mathbf{p}_{\mathtt{B}}}{\Delta t_{i,i+1}} \tag{7.19}$$

To compute the velocity of the most recent keyframe, we use the velocity relation (7.1):

$${}_{\mathtt{W}}\mathbf{v}_{\mathtt{B}}^N = {}_{\mathtt{W}}\mathbf{v}_{\mathtt{B}}^{N-1} + \mathbf{g}_{\mathtt{W}}^* \Delta t_{N-1,N} + \mathbf{R}_{\mathtt{WB}}^{N-1}\left(\Delta \mathbf{v}_{N-1,N} + \mathbf{J}_{\Delta v}^a \mathbf{b}_a^*\right) \tag{7.20}$$

7.3.5 Bias Reinitialization after Relocalization

When the system relocalizes after a long period of time, using place recognition, we reinitialize gyroscope biases by solving (7.7). The accelerometer bias is estimated by solving a simplified (7.17), where the only unknown is the bias, as scale and gravity are already known. We use 20 consecutive frames localized with only vision to estimate both biases.

7.4 Experiments

We evaluate the proposed IMU initialization method, detailed in Section 7.3 and our Visual-Inertial ORB-SLAM in the EuRoC dataset [6]. It contains 11 sequences recorded from a micro aerial vehicle (MAV), flying around two different rooms and an industrial environment. Sequences are classified as *easy*, *medium* and *difficult*, depending on illumination, texture, fast/slow motions or motion blur. The dataset provides synchronized global shutter WVGA stereo images at 20Hz with IMU measurements at 200Hz and trajectory ground-truth. These characteristics make it a really useful dataset to test visual-inertial SLAM systems. The experiments were performed processing left images only, in an Intel Core i7-4700MQ computer with 8Gb RAM.

7.4.1 IMU Initialization

We evaluate the IMU initialization in sequences *V1_01_easy* and *V2_01_easy*. We run the IMU initialization from scratch every time a new keyframe is inserted by ORB-SLAM. We run the sequences at a lower frame-rate so that the repetitive initialization does not interfere with the normal behavior of the system. The goal is to analyze the convergence of the variables as more keyframes, i.e. longer trajectories, are processed by the initialization algorithm. Fig. 7.3 shows the estimated scale and IMU biases. It can be seen that between 10 and 15 seconds all variables have already converged to stable values and that the estimated scale factor is really close to its optimal value. This optimal scale factor is computed aligning the estimated trajectory with the ground-truth by a similarity transformation [29]. Fig. 7.3 also shows the condition number of (7.17), indicating that some time is required to get a well-conditioned problem. This confirms that the sensor has to perform a motion that makes all variables observable, especially the accelerometer bias. The last row in Fig. 7.3 shows the total time spent by the initialization algorithm, which exhibits a linear growth. This complexity is the result of not including velocities in (7.10) and (7.17), which would have resulted in a quadratic complexity when using SVD to solve these systems. Subdividing the initialization in simpler subproblems, in contrast to [33, 48], results in a very efficient method.

The proposed initialization allows to start fusing IMU information, as gravity, biases, scale and velocity are reliably estimated. For the EuRoC dataset, we observed that 15 seconds of MAV exploration gives always an accurate initialization. As a future work we would like to investigate an automatic criterion to decide when we can consider an initialization successful, as we observed that an absolute threshold on the condition number is not reliable enough.

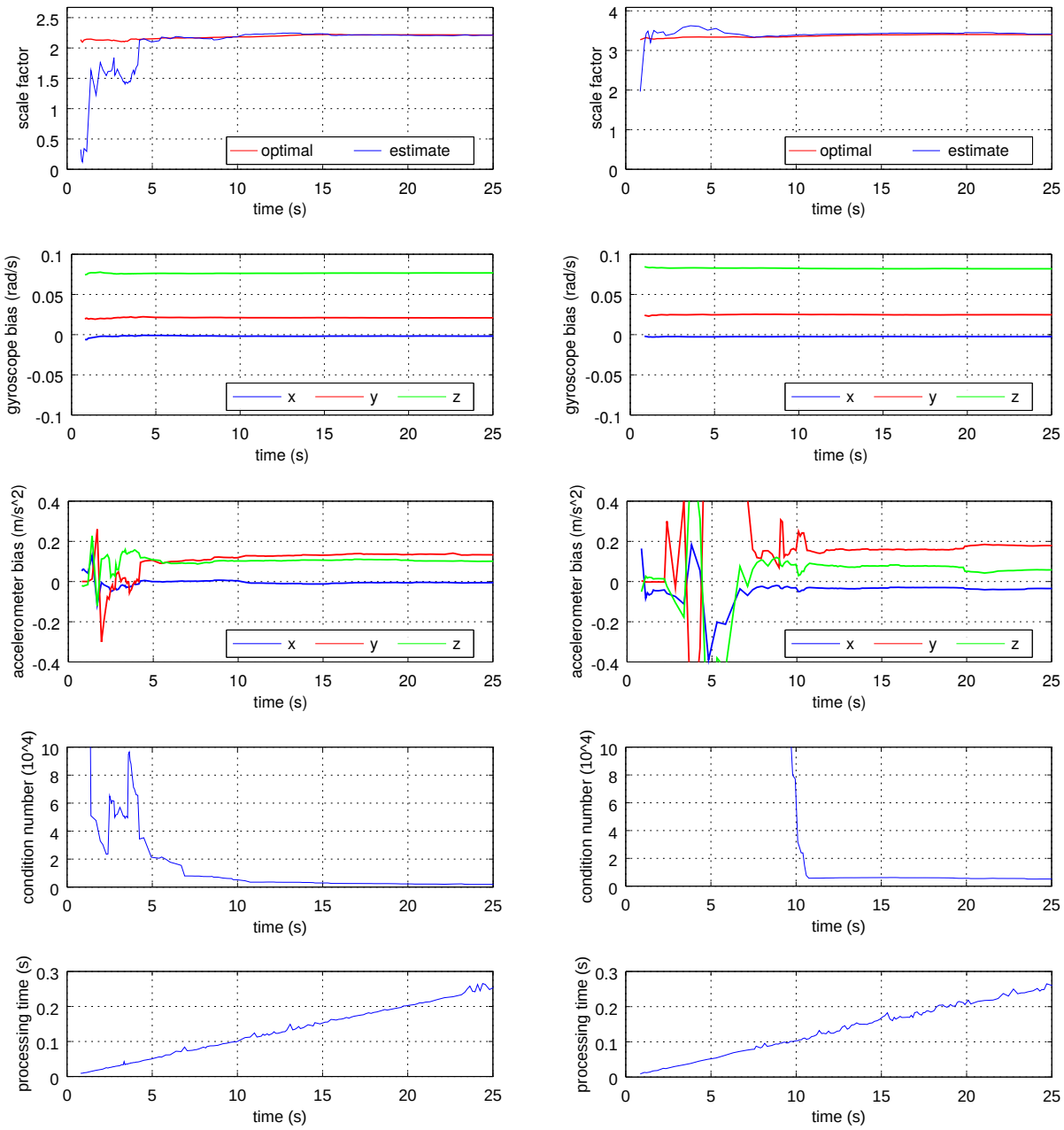


Figure 7.3: IMU initialization in the sequences *V1_01_easy* (left) and *V2_01_easy* (right). To generate these plots, we have run our initialization algorithm with all keyframes, each time a new keyframe is inserted. Note that the algorithm do not reuse any information from previous runs. However it can be seen how the scale (first row), gyroscope biases (second row) and accelerometer biases(third row) converge to stable values as the algorithm include more keyframes. Fourth row shows the condition number of (7.17). ORB-SLAM is able to erase keyframes, which explains that the condition number not always decrease if an informative keyframe for IMU initialization is erased. The last row is the time spent by the initialization.

7.4.2 SLAM Evaluation

We evaluate the accuracy of our Visual-Inertial ORB-SLAM in the 11 sequences of the EuRoC dataset. We start processing the sequences when the MAV starts exploring. The local window size for the local BA is set to 10 keyframes and the IMU initialization is performed after 15 seconds from monocular ORB-SLAM initialization. The system performs a full BA just after IMU initialization. Table 7.1 shows the translation Root Mean Square Error (RMSE) of the keyframe trajectory for each sequence, as proposed in [78]. We use the raw Vicon and Leica ground-truth as the post-processed one already used IMU. We observed a time offset between the visual-inertial sensor and the raw ground-truth of $-0.2s$ in the *Vicon Room 2* sequences and $0.2s$ in the *Machine Hall*, that we corrected when aligning both trajectories. We also measure the ideal scale factor that would align optimally the estimated trajectory and ground-truth. This scale factor can be regarded as the residual scale error of the trajectory and reconstruction. Our system successfully processes all these sequences in real-time, except *V1_03_difficult*, where the movement is too extreme for the monocular system to survive 15 seconds. Our system is able to recover motion with metric scale, with a scale error typically below 1%, achieving a typical precision of 3cm for $30m^2$ room environments and of 8cm for $300m^2$ industrial environments. To show the loss in accuracy due to scale error, we also show the RMSE if the system would be able to recover the true scale, see *GT scale* columns. We also show that the precision and scale estimation can be further improved by performing a visual-inertial full BA at the end of the execution, see Full BA columns. The reconstruction

Table 7.1: Keyframe trajectory accuracy in EuRoC dataset (raw ground-truth)

	Visual-Inertial ORB-SLAM						Monocular ORB-SLAM	
	No Full BA			Full BA			No Full BA	Full BA
	RMSE (m)	Scale Error (%)	RMSE(m) *GT scale**	RMSE (m)	Scale Error (%)	RMSE (m) *GT scale**	RMSE(m) *GT scale**	RMSE(m) *GT scale**
V1_01_easy	0.027	0.9	0.019	0.023	0.8	0.016	0.015	0.015
V1_02_medium	0.028	0.8	0.024	0.027	1.0	0.019	0.020	0.020
V1_03_difficult	X	X	X	X	X	X	X	X
V2_01_easy	0.032	0.2	0.031	0.018	0.2	0.017	0.021	0.015
V2_02_medium	0.041	1.4	0.026	0.024	0.8	0.017	0.018	0.017
V2_03_difficult	0.074	0.7	0.073	0.047	0.6	0.045	X	X
MH_01_easy	0.075	0.5	0.072	0.068	0.3	0.068	0.071	0.070
MH_02_easy	0.084	0.8	0.078	0.073	0.4	0.072	0.067	0.066
MH_03_medium	0.087	1.5	0.067	0.071	0.1	0.071	0.071	0.071
MH_04_difficult	0.217	3.4	0.081	0.087	0.9	0.066	0.082	0.081
MH_05_difficult	0.082	0.5	0.077	0.060	0.2	0.060	0.060	0.060

**GT scale*: the estimated trajectory is scaled so that it perfectly matches the scale of the ground-truth. These columns are included for comparison purposes but do not represent the output of a real system, but the output of an *ideal* system that could estimate the true scale.

for sequence *V1_02_medium* can be seen in Fig. 7.4.

To contextualize our results, we include as baseline the results of our vision-only system in Table 7.1. Our visual-inertial system is more robust as it can process *V2_03_difficult*, it is able to recover metric scale and does not suffer scale drift. The accuracy of the visual-inertial system is similar to the accuracy that would obtain the vision-only version if it could ideally recover the true scale. However the visual-inertial bundle adjustment is more costly, as explained in Section 7.2.3, and the local window of the local BA has to be smaller that in the vision-only case. This explains the slightly worse results of the *GT scaled* visual-inertial results without full BA. In fact the visual-inertial full BA typically converges in 15 iterations in 7 seconds, while the vision-only full BA converges in 5 iterations in less than 1 second.

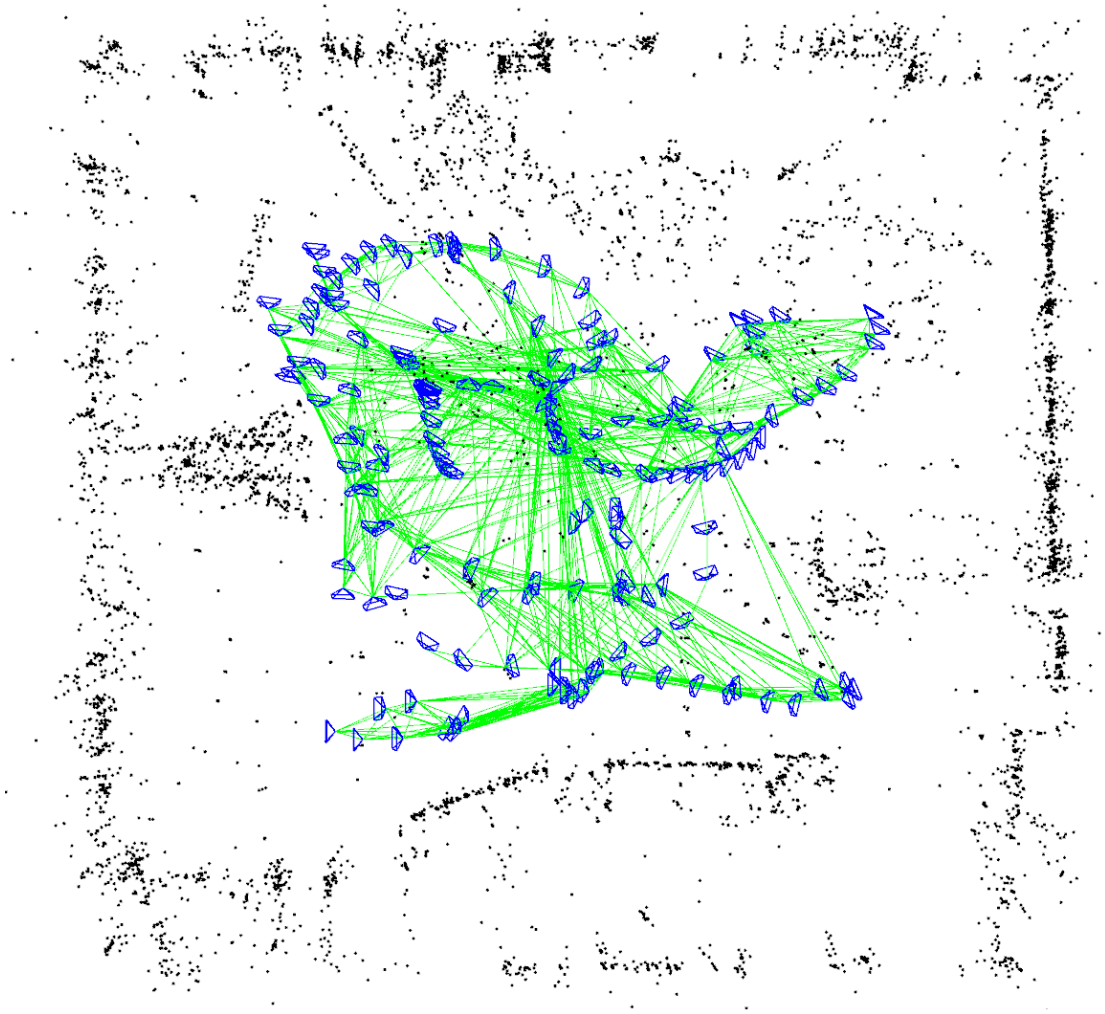


Figure 7.4: Top view of the reconstruction built by our system from sequence *V1_02_medium*. This top view was aligned using the gravity direction computed by Visual-Inertial ORB-SLAM. The green lines connect keyframes that share more than 100 point observations and are a proof of the capability of the system to reuse the map. This reuse capability, in contrast to visual-inertial odometry, allows drift-free localization when continually revisiting.

In order to test the capability of Visual-Inertial ORB-SLAM to reuse a previous map, we run in a row all sequences of the same environment. We process the first sequence and perform a full BA. Then we run the rest of the sequences, where our system relocalizes and continue doing SLAM. We then compare the accumulated keyframe trajectory with the ground-truth. As there exists a previous map, our system is now able to localize the camera in sequence *V1_03_difficult*. The RMSE in meters for V1, V2 and MH environments are 0.037, 0.027 and 0.076 respectively, with an scale factor error of 1.2%, 0.1% and 0.2%. A final full BA has a negligible effect as we have already performed a full BA at the end of the first sequence. These results show that there is no drift accumulation when revisiting the same scene, as the RMSE for all sequences is not larger than for individual sequences.

Finally we have compared Visual-Inertial ORB-SLAM to the state-of-the-art direct visual-inertial odometry for stereo cameras [85], which also showed results in *Vicon Room 1* sequences, allowing for a direct comparison. Fig. 7.5 shows the Relative Pose Error (RPE) [27]. To compute the RPE for our method, we need to recover the frame trajectory, as only keyframes are optimized by our backend. To this end, when tracking a frame we store a relative transformation to a reference keyframe, so that we can retrieve the frame pose from the estimated keyframe pose at the end of the execution. We have not run a full BA at the end of the experiment. We can see that the error for the visual-inertial odometry method grows with the traveled distance, while our visual-inertial SLAM system does not accumulate error due to map reuse. The stereo method [85] is able to work in *V1_03_difficult*, while our monocular method fails. Our monocular SLAM successfully recovers metric scale, and achieves comparable accuracy in short paths, where the advantage of SLAM is negligible

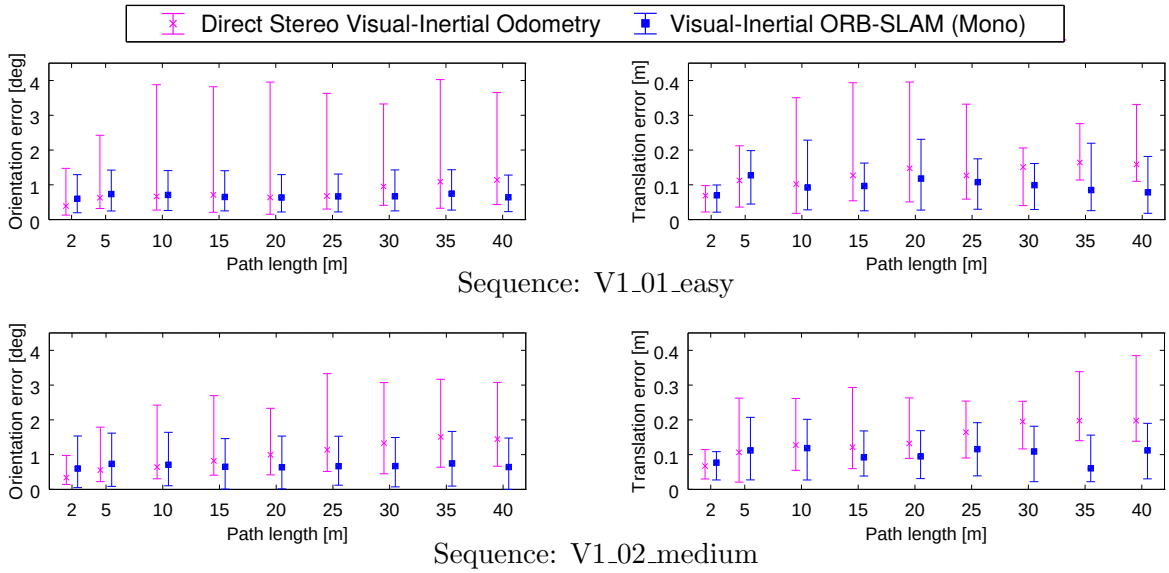


Figure 7.5: Relative Pose Error [27] comparison between our approach and the state-of-the-art direct stereo visual-inertial odometry [85]. The error for our SLAM system does not grow for longer paths, due to map reuse, in contrast to the visual-inertial odometry method where drift cannot be compensated. Note that [85] uses stereo, while our results are monocular. We thank the authors of [85] for providing their results for this comparison.

compared to odometry. This is a remarkable result of our feature-based monocular method, compared to [85] which is direct and stereo.

7.5 Discussion

We have presented a novel tightly coupled visual-inertial SLAM system, that is able to close loops in real-time and localize the sensor reusing the map in already mapped areas. This allows to achieve a *drift-free* localization, in contrast to visual odometry approaches where drift grows unbounded. The experiments show that our monocular SLAM recovers metric scale with high precision, and achieves better accuracy than the state-of-the-art in stereo visual-inertial odometry when continually localizing in the same environment. We consider this *drift-free* localization of particular interest for virtual/augmented reality systems, where the predicted user viewpoint must not drift when the user operates in the same workspace. Moreover we expect to achieve better accuracy and robustness by using stereo or RGB-D cameras, which would also simplify IMU initialization as scale is known. The main weakness of our proposed IMU initialization is that it relies on the initialization of the monocular SLAM. We would like to investigate the use of the gyroscope to make the monocular initialization faster and more robust.

Chapter 8

Conclusions

8.1 Discussion

In this thesis we have addressed the problem of Simultaneous Localization and Mapping using the most common vision sensors. We have focused on developing robust and accurate solutions that scale well in large environments, that operate in real-time in standard CPUs, and that provide a drift-free localization when continually operating in the same environment.

A core element of our visual SLAM solutions is place recognition to close loops and relocalize the vision sensor. We have proposed in Chapter 3 a place recognition method based on DBoW2 and ORB features with a novel orientation consistency test, and applied it to loop detection in image sequences and relocalization. Guided by the excellent recall and precision of the loop detector, and the significant invariance of the relocalization to viewpoint, we have integrated an improved version of this place recognition in our visual SLAM in Chapter 4. The main improvement and contribution is the use of covisibility information between images instead of temporal order. Covisibility information links images of the same scene independently of the time at which the images were captured. This is important in visual SLAM where keyframes of the same scene can be inserted at very different times when the sensor revisits the environment. Our approach has demonstrated to work in real-time for loop closing and relocalization, achieving perfect precision and high recall in popular datasets like NewCollege, KITTI, EuRoC and TUM RGB-D. We have also made live hand-held demonstrations in a wide variety of environments with excellent results. We have always used the same ORB vocabulary of 1 million words, demonstrating that a big vocabulary work well in almost any situation. While we have rarely seen false positive loop closures (which have a catastrophic effect), they happen, and constitute an open challenge. The use of robust loop closing techniques [39, 80] can help to mitigate false loop detections.

In Chapter 4 we have presented ORB-SLAM, which is currently the most reliable and complete open-source monocular SLAM solution. We use the same ORB features for all system tasks, unifying front-end, back-end and place recognition. The system is able to close large loops and relocalize the camera in real-time, and includes an automatic and robust system initialization. The result is a system that build maps that can be reused, allowing a drift-free localization in mapped environments, in contrast to pure visual odometry approaches. This is a desirable capability for robot navigation or virtual and augmented reality

systems. Our exhaustive evaluation in popular public datasets has shown that ORB-SLAM is capable in real-time to work with unprecedented accuracy in small and large environments, indoors and outdoors, and from hand-held, car, ground robot, and drone motions. The evaluation also showed the better accuracy and robustness of our feature-based solution compared to a state-of-the-art direct SLAM implementation. While direct approaches should ideally be more accurate and robust, as not relying on features and potentially exploiting all image information, in practice unmodeled effects like rolling shutter, auto gain or auto exposure have a major impact. In addition a joint direct optimization of dense or semi-dense structure and camera trajectory is more expensive than sparse bundle adjustment and is not feasible in real-time, which we believe is essential to achieve the most accurate results. The recently proposed sparse direct optimization [16] supports this claim and seems to be the means to achieve a comparable accuracy to feature-based methods.

One of the main drawbacks of feature-based approaches is their sparse reconstruction. While a sparse reconstruction has the benefit of being cheap in terms of memory footprint, its description of the environment is rather poor for robotics tasks like navigation or interaction. Therefore in Chapter 5 we have integrated a novel probabilistic semi-dense mapping module in ORB-SLAM to build in real-time semi-dense reconstructions. In contrast to fully direct approaches, our system builds a feature-based map that is used for camera localization, and a semi-dense map to describe the environment using a direct method. In this way the resulting system combines the high accuracy, robustness and place recognition capabilites of our feature-based SLAM with the semi-dense reconstruction of direct methods.

In Chapter 6 we have extended our ORB-SLAM framework to stereo and RGB-D cameras, re-designing some modules to best exploit the stereo and depth information from these kind of sensors. The result is a system that outperforms in accuracy the state-of-the-art stereo and RGB-D SLAM systems. Surprisingly our results shows that by using bundle adjustment, our solution is more accurate than methods using standard ICP or photometric error. We have also incorporated a new lightweight *Localization Mode* where the system is able to localize (and relocalize) the camera in an already mapped environment, by having active only the tracking thread of the system, saving CPU resources. Moreover when the input is stereo or RGB-D the tracking in this mode, leverages visual odometry and map point matches so that tracking can shortly survive in unmapped regions. This novel system, called ORB-SLAM2, is the first open-source SLAM solution for monocular, stereo and RGB-D cameras, and might benefit researchers working on a wide variety of environments.

Finally in Chapter 7 we have presented Visual-Inertial Monocular ORB-SLAM, which fuses monocular vision and inertial information. This is a novel tightly coupled visual-inertial SLAM that is able to close loops and build globally consistent maps in real-time. The system is able to reuse its map and achieve drift-free localization in contrast to pure visual-inertial odometry methods. While the visual-inertial bundle adjustment is more costly than its vision-only counterpart, our visual-inertial system is able to estimate the true scale of the map, also avoiding scale drift, and is more robust under extreme motions, thanks to the prediction of the inertial sensor.

8.2 Sensors

We have worked with monocular, stereo, RGB-D and IMU sensors. We have firstly addressed monocular SLAM which is the most general and challenging visual SLAM problem. Monocular SLAM can be useful in situations where size or power consumption is critical, as in endoscopy surgery or in mobile devices. It can also be the most efficient solution for some augmented reality applications where absolute scale information is not needed. Apart from an up-to-scale estimation, monocular SLAM has several drawbacks as a delicate initialization, scale drift, or needing translation to reconstruct the environment. All these issues are easily solved by using stereo and RGB-D and therefore we consider stereo or RGB-D SLAM the most reliable solutions for visual SLAM. Choosing between stereo and RGB-D will mainly depend on the application and environment, but as a rule of thumb stereo SLAM is the most general solution. Our stereo and RGB-D SLAM solutions were based on a monocular SLAM approach, and therefore they have a strong emphasis in multi-view geometry which we consider important to exploit information out of the range of these sensors.

Finally we have developed a tightly-coupled visual-inertial monocular SLAM. By fusing IMU information we are able to have a better prediction of the camera that can help under abrupt movements of the camera, where a motion model would fail. The IMU also allows to retrieve the true scale of the map and reduce scale drift. However, the cost is a more complex system and where visual-inertial optimizations are more computationally demanding.

8.3 Future Work

We believe that our detailed descriptions of our state-of-the-art solutions and open-source implementations provide a solid base for future developments in the field. We find interesting the following lines of research:

- Extend our solutions to include support for wide field of view cameras and non-overlapping camera systems, cooperative mapping and rolling shutter correction.
- Long-term mapping in dynamic environments so that outdated keyframes and points are forgotten keeping the map size bounded. This will allow lifelong SLAM in dynamic environments.
- Semantic mapping so that maps are augmented with semantic annotations that can be useful for robot navigation or user interaction in augmented or virtual reality applications. Moreover semantic information can also boost the precision and recall of place recognition.
- Dynamic object tracking so that moving objects are tracked and reconstructed. This can increase the robustness of the camera tracking as correspondences to moving objects are avoided, and be useful for robot or user interaction.
- Auto-calibration to increase the accuracy of visual SLAM.

- Hybrid solutions combining strengths of feature-based and direct methods. Direct methods are robust to motion blur and can better exploit the image information when there is little texture. On the other hand feature-based methods are useful for place recognition, loop closing, relocalization and map reuse, and exhibit better convergence.

Bibliography

[1] P. F. Alcantarilla, J. Nuevo, and A. Bartoli. Fast explicit diffusion for accelerated features in nonlinear scale spaces. In *British Machine Vision Conference (BMVC)*, Bristol, UK, 2013.

[2] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded Up Robust Features. In *European Conference on Computer Vision (ECCV)*, pages 404–417. Graz, Austria, 2006.

[3] J. L. Blanco, F. A. Moreno, and J. Gonzalez. A collection of outdoor robotic datasets with centimeter-accuracy ground truth. *Autonomous Robots*, 27(4):327–351, 2009.

[4] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart. Robust visual inertial odometry using a direct EKF-based approach. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 298–304, Hamburg, Germany, 2015.

[5] A. Bonarini, W. Burgard, G. Fontana, M. Matteucci, D. G. Sorrenti, and J. D. Tardós. RAWSEEDS: Robotics Advancement through Web-publishing of Sensorial and Elaborated Extensive Data Sets. In *Intelligent Robots and Systems (IROS) Workshop on Benchmarks in Robotics Research*, Beijing, China, 2006.

[6] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart. The EuRoC micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35(10):1157–1163, 2016.

[7] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF: Binary Robust Independent Elementary Features. In *European Conference on Computer Vision (ECCV)*, pages 778–792. Hersonissos, Greece, 2010.

[8] A. Chiuso, P. Favaro, H. Jin, and S. Soatto. Structure from motion causally integrated over time. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):523–535, 2002.

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

[10] A. Concha, G. Loianno, V. Kumar, and J. Civera. Visual-inertial direct SLAM. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1331–1338, Stockholm, Sweden, 2016.

[11] M. Cummins and P. Newman. FAB-MAP: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, 27(6):647–665, 2008.

[12] M. Cummins and P. Newman. Appearance-only SLAM at large scale with FAB-MAP 2.0. *The International Journal of Robotics Research*, 30(9):1100–1123, 2011.

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

[14] E. Eade and T. Drummond. Scalable monocular SLAM. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 469–476, New York, USA, 2006.

[15] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard. 3-D mapping with an RGB-D camera. *IEEE Transactions on Robotics*, 30(1):177–187, 2014.

[16] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. In *arXiv:1607.02565*, July 2016.

[17] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *European Conference on Computer Vision (ECCV)*, pages 834–849. Zurich, Switzerland, 2014.

[18] J. Engel, J. Stueckler, and D. Cremers. Large-scale direct SLAM with stereo cameras. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, 2015.

[19] J. Engel, J. Sturm, and D. Cremers. Semi-dense visual odometry for a monocular camera. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1449–1456, Sidney, Australia, 2013.

[20] O. D. Faugeras and F. Lustman. Motion and structure from motion in a piecewise planar environment. *International Journal of Pattern Recognition and Artificial Intelligence*, 2(03):485–508, 1988.

[21] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza. IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. In *Robotics: Science and Systems (RSS)*, Rome, Italy, 2015.

[22] C. Forster, M. Pizzoli, and D. Scaramuzza. Appearance-based active, monocular, dense reconstruction for micro aerial vehicle. In *Robotics: Science and Systems (RSS)*, Berkeley, USA, 2014.

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

[24] P. Furgale, J. Rehder, and R. Siegwart. Unified temporal and spatial calibration for multi-sensor systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1280–1286, Tokio, Japan, 2013.

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

[26] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

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

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

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

[30] V. Indelman, S. Williams, M. Kaess, and F. Dellaert. Information fusion in navigation systems via factor graph based incremental smoothing. *Robotics and Autonomous Systems*, 61(8):721–738, 2013.

[31] E. S. Jones and S. Soatto. Visual-inertial navigation, mapping and localization: A scalable real-time causal approach. *The International Journal of Robotics Research*, 30(4):407–430, 2011.

[32] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert. iSAM2: Incremental smoothing and mapping using the bayes tree. *The International Journal of Robotics Research*, 31(2):216–235, 2012.

[33] J. Kaiser, A. Martinelli, F. Fontana, and D. Scaramuzza. Simultaneous state initialization and gyroscope bias calibration in visual inertial aided navigation. *IEEE Robotics and Automation Letters*, 2(1):18–25, 2017.

[34] C. Kerl, J. Stueckler, and D. Cremers. Dense continuous-time tracking and mapping with rolling shutter RGB-D cameras. In *IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile, 2015.

[35] C. Kerl, J. Sturm, and D. Cremers. Dense visual SLAM for RGB-D cameras. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Tokio, Japan, 2013.

[36] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 225–234, Nara, Japan, 2007.

[37] G. Klein and D. Murray. Improving the agility of keyframe-based SLAM. In *European Conference on Computer Vision (ECCV)*, pages 802–815. Marseille, France, 2008.

[38] R. Kuemmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3607–3613, Shanghai, China, 2011.

[39] Y. Latif, C. Cadena, and J. Neira. Robust loop closing over time for pose graph SLAM. *The International Journal of Robotics Research*, 32(14):1611–1626, 2013.

[40] V. Lepetit, F. Moreno-Noguer, and P. Fua. EPnP: An accurate O(n) solution to the PnP problem. *International Journal of Computer Vision*, 81(2):155–166, 2009.

[41] 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*, 34(3):314–334, 2015.

[42] H. Lim, J. Lim, and H. J. Kim. Real-time 6-DOF monocular visual SLAM in a large-scale environment. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1532–1539, Hong Kong, China, 2014.

[43] H. Longuet-Higgins. The reconstruction of a plane surface from two perspective projections. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 227(1249):399–410, 1986.

[44] S. Lovegrove, A. J. Davison, and J. Ibanez-Guzmán. Accurate visual odometry from a rear parking camera. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 788–793, Baden-Baden, USA, 2011.

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

[46] T. Lupton and S. Sukkarieh. Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions. *IEEE Transactions on Robotics*, 28(1):61–76, 2012.

[47] S. Lynen, T. Sattler, M. Bosse, J. Hesch, M. Pollefeys, and R. Siegwart. Get out of my lab: Large-scale, real-time visual-inertial localization. In *Robotics: Science and Systems (RSS)*, Rome, Italy, 2015.

[48] A. Martinelli. Closed-form solution of visual-inertial structure from motion. *International Journal of Computer Vision*, 106(2):138–152, 2014.

[49] 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*, 94(2):198–214, 2011.

[50] C. Mei, G. Sibley, and P. Newman. Closing loops without places. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3738–3744, Taipei, Taiwan, 2010.

[51] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. Real time localization and 3d reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 363–370, New York, USA, 2006.

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

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

[54] R. Mur-Artal and J. D. Tardós. Fast relocalisation and loop closing in keyframe-based SLAM. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 846–853, Hong Kong, China, 2014.

[55] R. Mur-Artal and J. D. Tardos. Probabilistic semi-dense mapping from highly accurate feature-based monocular SLAM. In *Robotics: Science and Systems (RSS)*, Rome, Italy, 2015.

[56] R. Mur-Artal and J. D. Tardós. ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *arXiv preprint arXiv:1610.06475*, 2016.

[57] R. Mur-Artal and J. D. Tardós. Visual-inertial monocular SLAM with map reuse. *IEEE Robotics and Automation Letters*, 2017. (Accepted for publication).

[58] R. A. Newcombe and A. J. Davison. Live dense reconstruction with a single moving camera. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1498–1505, San Francisco, USA, 2010.

[59] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, Basel, Switzerland, 2011.

[60] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2320–2327, Barcelona, Spain, 2011.

[61] D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–770, 2004.

[62] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 2161–2168, New York, USA, 2006.

[63] A. Patron-Perez, S. Lovegrove, and G. Sibley. A spline-based trajectory representation for sensor fusion and rolling shutter cameras. *International Journal of Computer Vision*, 113(3):208–219, 2015.

[64] L. M. Paz, P. Piniés, J. D. Tardós, and J. Neira. Large-scale 6-DOF SLAM with stereo-in-hand. *IEEE Transactions on Robotics*, 24(5):946–957, 2008.

[65] T. Pire, T. Fischer, J. Civera, P. De Cristóforis, and J. J. Berlles. Stereo parallel tracking and mapping for robot localization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1373–1378, Hamburg, Germany, 2015.

[66] K. Pirker, M. Ruther, and H. Bischof. CD SLAM-continuous localization and mapping in a dynamic world. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3990–3997, San Francisco, USA, 2011.

[67] M. Pizzoli, C. Forster, and D. Scaramuzza. REMODE: Probabilistic, monocular dense reconstruction in real time. In *International Conference on Robotics and Automation (ICRA)*, pages 2609–2616, Hong Kong, China, 2014.

[68] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision (ECCV)*, pages 430–443. Graz, Austria, 2006.

[69] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: an efficient alternative to SIFT or SURF. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2564–2571, Barcelona, Spain, 2011.

[70] D. Scaramuzza, A. Martinelli, and R. Siegwart. A toolbox for easily calibrating omnidirectional cameras. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5695–5701, Beijing, China, 2006.

[71] M. Smith, I. Baldwin, W. Churchill, R. Paul, and P. Newman. The new college vision and laser data set. *The International Journal of Robotics Research*, 28(5):595–599, 2009.

[72] S. Song, M. Chandraker, and C. C. Guest. Parallel, real-time monocular visual odometry. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4698–4705, Karlsruhe, Germany, 2013.

[73] H. Strasdat. *Local Accuracy and Global Consistency for Efficient Visual SLAM*. PhD thesis, Imperial College, London, October 2012.

[74] H. Strasdat, A. J. Davison, J. M. M. Montiel, and K. Konolige. Double window optimisation for constant time visual SLAM. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2352–2359, Barcelona, Spain, 2011.

[75] H. Strasdat, J. M. M. Montiel, and A. J. Davison. Scale drift-aware large scale monocular SLAM. In *Robotics: Science and Systems (RSS)*, Zaragoza, Spain, 2010.

[76] H. Strasdat, J. M. M. Montiel, and A. J. Davison. Visual SLAM: Why filter? *Image and Vision Computing*, 30(2):65–77, 2012.

[77] J. Stühmer, S. Gumhold, and D. Cremers. Real-time dense geometry from a handheld camera. In *Proc. 32nd Annual Symp. German Association for Pattern Recognition (DAGM)*, pages 11–20, Darmstadt, Germany, 2010.

[78] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 573–580, Vilamoura, Portugal, 2012.

[79] N. Sünderhauf, T. T. Pham, Y. Latif, M. Milford, and I. Reid. Meaningful maps - object-oriented semantic mapping. *arXiv preprint arXiv:1609.07849*, 2016.

[80] N. Sünderhauf and P. Protzel. Switchable constraints for robust pose graph SLAM. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1879–1884, Vilamoura, Portugal, 2012.

[81] W. Tan, H. Liu, Z. Dong, G. Zhang, and H. Bao. Robust monocular SLAM in dynamic environments. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 209–218, Adelaide, Australia, 2013.

[82] P. H. Torr, A. W. Fitzgibbon, and A. Zisserman. The problem of degeneracy in structure and motion recovery from uncalibrated image sequences. *International Journal of Computer Vision*, 32(1):27–44, 1999.

[83] P. H. Torr and A. Zisserman. Feature based methods for structure and motion estimation. In *Vision Algorithms: Theory and Practice*, pages 278–294. Springer, 2000.

[84] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment a modern synthesis. In *Vision algorithms: theory and practice*, pages 298–372. 2000.

[85] V. Usenko, J. Engel, J. Stueckler, and D. Cremers. Direct visual-inertial odometry with stereo cameras. In *IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden, 2016.

[86] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, and J. McDonald. Real-time large-scale dense RGB-D SLAM with volumetric fusion. *The International Journal of Robotics Research*, 34(4-5):598–626, 2015.

[87] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger. ElasticFusion: Real-time dense SLAM and light source estimation. *The International Journal of Robotics Research*, 2016.

[88] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. D. Tardós. A comparison of loop closing techniques in monocular SLAM. *Robotics and Autonomous Systems*, 57(12):1188–1197, 2009.

[89] K. Wu, A. Ahmed, G. Georgiou, and S. Roumeliotis. A square root inverse filter for efficient vision-aided inertial navigation on mobile devices. In *Robotics: Science and Systems (RSS)*, Rome, Italy, 2015.

[90] X. Yang and K.-T. Cheng. LDB: An ultra-fast feature for scalable augmented reality on mobile devices. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 49–57, Atlanta, USA, 2012.

[91] Z. Yang and S. Shen. Monocular visual-inertial state estimation with online initialization and camera-IMU extrinsic calibration. *IEEE Transactions on Automation Science and Engineering*, 2016. DOI: 10.1109/TASE.2016.2550621 (to appear).