TESIS DE LA UNIVERSIDAD

DE ZARAGOZA

2021

365

Carlos Campos Martínez

Precise and Robust Visual SLAM with Inertial Sensors and Deep Learning.

Director/es

Dr. D. Juan Domingo Tardós Solano

ISSN 2254-7606

Prensas de la Universidad
Universidad Zaragoza

Servicio de Publicaciones

ISSN 2254-7606

Universidad Zaragoza
1542

Tesis Doctoral

PRECISE AND ROBUST VISUAL SLAM WITH INERTIAL SENSORS AND DEEP LEARNING.

Autor

Carlos Campos Martínez

Director/es

Dr. D. Juan Domingo Tardós Solano

UNIVERSIDAD DE ZARAGOZA
Escuela de Doctorado

2021

Tesis Doctoral

Precise and Robust Visual SLAM with Inertial Sensors and Deep Learning

Autor

Carlos Campos Martínez

Director

Juan Domingo Tardós Solano

Escuela de Ingeniería y Arquitectura
2021

Ph.D. Thesis

Precise and Robust Visual SLAM with Inertial Sensors and Deep Learning

Carlos Campos Martínez

Supervised by

Juan Domingo Tardós Solano

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

Universidad de Zaragoza

September 24, 2021

Precise and Robust Visual SLAM with Inertial Sensors and Deep Learning

Carlos Campos Martínez

Supervisor

Juan Domingo Tardós Solano	Universidad de Zaragoza

Dissertation Committee

Javier Gonzalez Jimenez	Universidad de Málaga
Luca Carlone	Massachusetts Institute of Technology
Javier Civera Sancho	Universidad de Zaragoza
Joan Solà Ortega	Institut de Robòtica i Informàtica Industrial
Luis Miguel Bergasa Pascual	Universidad de Alcalá de Henares

Funding

HF201704003	Huawei
DPI2015-67275	Spanish Government
DPI2017-91104-EXP	Spanish Government
PGC2018-096367-B-I00	Spanish Government
DGA_T45-17R	Government of Aragón

A Raquel
A mis padres y hermano

Agradecimientos

First of all I would like to thank my supervisor, Professor Juan Domingo Tardós, for introducing me to this world which has become my passion. All his support and comments have always motivated and inspired me to continue working hard in this field, guiding me through these years. Without a doubt I am the researcher that I am thanks to him. I would also like to thank Professor José María Martínez Montiel for the fruitful discussions we had. They have helped me to advance in my research, especially during the first stages of my PhD.

After these years I take with me an incredible group of labmates and friends with whom I have shared memorable moments, inside and outside the lab. I would like to mention Richard, Juanjo and Berta, researchers with whom I have had the pleasure to collaborate with and learn from them.

Muchas gracias a mis padres y hermano por estar siempre ahí, desde los comienzos, así como al resto de mi familia y amigos, en especial a mis abuelos y tías que les hubiera encantado vivir este momento. Muchas gracias Raquel por todo el apoyo y cariño incondicional de estos años, tú has sido la motivación y la compañera de viaje de esta aventura ¡Seguro que el futuro nos depara grandes momentos juntos!

Por último, gracias a todos los que de una u otra manera me han ayudado a ser lo que soy, permitiéndome llegar a este momento.

Resumen

Dotar a los robots con el sentido de la percepción destaca como el componente más importante para conseguir máquinas completamente autónomas. Una vez que las máquinas sean capaces de percibir el mundo, podrán interactuar con él. A este respecto, la localización y la reconstrucción de mapas de manera simultánea, SLAM (por sus siglas en inglés) comprende todas las técnicas que permiten a los robots estimar su posición y reconstruir el mapa de su entorno al mismo tiempo, usando únicamente el conjunto de sensores a bordo. El SLAM constituye el elemento clave para la percepción de las máquinas, estando ya presente en diferentes tecnologías y aplicaciones como la conducción autónoma, la realidad virtual y aumentada o los robots de servicio. Incrementar la robustez del SLAM expandiría su uso y aplicación, haciendo las máquinas más seguras y requiriendo una menor intervención humana.

En esta tesis hemos combinado sensores inerciales (IMU) y visuales para incrementar la robustez del SLAM ante movimientos rápidos, oclusiones breves o entornos con poca textura. Primero hemos propuesto dos técnicas rápidas para la inicialización del sensor inercial, con un bajo error de escala. Estas han permitido empezar a usar la IMU tan pronto como 2 segundos después de lanzar el sistema. Una de estas inicializaciones ha sido integrada en un nuevo sistema de SLAM visual inercial, acuñado como ORB-SLAM3, el cual representa la mayor contribución de esta tesis. Este es el sistema de SLAM visual-inercial de código abierto más completo hasta la fecha, que funciona con cámaras monoculares o estéreo, estenopeicas o de ojo de pez, y con capacidades multimapa. ORB-SLAM3 se basa en una formulación de Máximo a Posteriori, tanto en la inicialización como en el refinamiento y el ajuste de haces visual-inercial. También explota la asociación de datos en el corto, medio y largo plazo. Todo esto hace que ORB-SLAM3 sea el sistema SLAM visual-inercial más preciso, como así demuestran nuestros resultados en experimentos públicos.

Además, hemos explorado la aplicación de técnicas de aprendizaje profundo para mejorar la robustez del SLAM. En este aspecto, primero hemos propuesto DynaSLAM II, un sistema SLAM estéreo para entornos dinámicos. Los objetos dinámicos son segmentados mediante una red neuronal, y sus puntos y medidas son incluidas eficientemente en la optimización de ajuste de haces. Esto permite estimar y hacer seguimiento de los objetos en movimiento, al mismo tiempo que se mejora la estimación de la trayectoria de la cámara. En segundo lugar, hemos desarrollado un SLAM monocular y directo basado en predicciones de profundidad a través de redes neuronales. Optimizamos de manera conjunta tanto los residuos de predicción de profundidad como los fotométricos de distintas vistas, lo que da lugar a un sistema monocular capaz de estimar la escala. No sufre el problema de deriva de escala, siendo más robusto y varias veces más preciso que los sistemas monoculares clásicos.

Abstract

Endowing robots with a perception module arises as the most important component to create fully autonomous machines. Once machines perceive the world they will be able to interact with it. In this respect, Simultaneous Localization and Mapping (SLAM) embraces all techniques which allow robots to simultaneously retrieve their pose and reconstruct the environment using measurements from a set of on-board sensors. SLAM constitutes the key component of a machine perception, being already present at different technologies and applications such as autonomous driving, augmented and virtual reality or service robots. Increasing its robustness will expand its use to new applications, making machines more safe and requiring less human intervention.

In this thesis, we have combined inertial (IMU) and visual sensors to increase SLAM robustness against fast motions, short occlusions or textureless environments. We have first proposed two techniques to quickly initialize the inertial sensors with very low scale errors. These have allowed the system to start using the IMU as early as 2 seconds after launching it, avoiding running less robust pure monocular SLAM for longer periods. We have integrated it in a new SLAM system, coined ORB-SLAM3, which represents the major contribution of this thesis. This is the most complete open-source visual-inertial SLAM system up-to-date, working in monocular or stereo inertial setups, pinhole and fisheye cameras, and with multi-map capabilities. ORB-SLAM3 fully relies on a Maximum a Posteriori formulation, from initialization, to refinement and visual-inertial Bundle Adjustment. It also exploits data-associations, in the short, mid and long-term. All this together makes ORB-SLAM3 the most accurate visual-inertial SLAM system, as proven by our results on public datasets.

We have also explored the application of deep learning techniques to improve SLAM robustness. In this regard, we have first proposed DynaSLAM II, a stereo SLAM system for dynamic environments. The dynamic objects are segmented by means of a neural network, and their points and measurements are efficiently included in the Bundle Adjustment optimization. This allows us to estimate and track these moving objects as well as improve the camera trajectory estimation. Second, we have developed a direct monocular SLAM method based on depth prediction from neural networks. We optimize together photometric and depth prediction residuals from multiple views, leading to a scale-aware monocular SLAM that does not suffer from scale drift, being more robust and several times more accurate than classic monocular systems.

Contents

Chapter 1

Introduction

1.1 Simultaneous Localization and Mapping

Robots and humans have been living together for a while. They have allowed people to get rid of hard, boring and unsatisfactory tasks, letting humans to focus on more creative, intellectual and rational jobs. While robots have been successfully used in industry for repetitive, well defined, and invariant tasks, their lack of an understanding about the underlying mechanisms which govern the world has limited their application to more general and unpredictable situations. Some of these potential applications lie in the transportation industry, like autonomous driving or last-mile delivery. Others comprehend Augmented and Virtual Reality (AR/VR) or service robots such as those for cleaning, health-care or security and surveillance. Despite recent developments in electronics, manufacturing, and electrical energy storing, that are boosting the use of robots, those have not come with similar improvements in the perception nor self-decision modules. Robots are still not able to perceive the world as rational beings do, nor to take important decisions based on this information. As an example, this remains the main reason why there is still a driver in a car; to endow it with perception and decision capacities.

This is where Simultaneous Localization and Mapping (SLAM) appears like a keystone. SLAM tries to solve the problem of building a map or reconstruction of the environment and track the robot pose at the same time, using a set of on-board sensors. This endows robots with sensory capacity, being aware of themselves and their environment, and enabling human interaction with them. This perceived information may later be used by decision modules, usually deep-learning based, like Reinforcement Learning algorithms, to generate an action, adapted to surrounding conditions, aiming to achieve a preset goal. These two components render a robot fully autonomous.

To achieve this goal, a gold standard SLAM system should stand out in the next features:

- **Localization accuracy**: It should be able to estimate the pose of the agent with low error for different circumstances. It should have a low drift for exploring trajectories, where the robot is continuously visiting new places, and no drift for revisiting trajectories where the agent moves in already mapped regions.

- **Mapping accuracy**: The reconstructed environment should be precise and contain as much information as possible, in order to ease the interaction between the agent and the environment. This accuracy should not be limited to geometrical precision, but also include more abstract information like semantics.

- **Robustness**: It should be able to deal with a wide-range of motions and environments (generality), preserving its performance (reliability).
- **Efficiency**: The system should be computationally efficient, able to work on real-time, ideally on-board, and also have a limited memory usage.

Depending on the application, SLAM systems will be more oriented and devoted to one point or another. A SLAM system used for AR/VR should be focused on efficiency, since it has to run in a light on-board set-up. A vacuum cleaner robot should prioritize localization and mapping accuracy, while it does not usually require a low latency. Furthermore, one used for autonomous driving should perform robustly in all situations since human lives may depend on it.

In the same way humans heavily depend on their senses to perceive the world, SLAM systems performance also deeply depends on the used sensors. According to the nature of the acquired data, sensors may be classified in two groups:

- **Proprioceptive** sensors, which measure quantities internal to the system, which do not depend on the environment. These are for example global position measured with a GPS, wheel velocity measured by an encoder, or linear acceleration and angular velocity measured by an Inertial Measurement Unit (IMU). This kind of sensor is typically used for localization accuracy and robustness, and comes with low computational requirements.
- **Exteroceptive** sensors, those that acquire data from the environment, which allows reconstructing it and localizing the agent. The most common examples are monocular, stereo and RGB-D cameras, as well as other sensors like radars, lidar or event cameras.

Sensors from both groups complement each other. Proprioceptive sensors provide small measurements at a high frequency, which are quickly processed and used for estimating agent pose at a high rate, while exteroceptive sensors measurements usually contain much more information, which is acquired at a lower rate, but which allows to reconstruct the environment and compute the agent pose with a higher precision.

During this thesis, both kinds of sensors will be employed. We will be specially interested in SLAM with only visual sensors, **Visual SLAM**, which comprises the biggest group along the SLAM research community. This sensor configuration covers most of the real-world applications, since most existing devices count with at least one monocular camera. The goal of Visual SLAM is to use the cameras on-board a mobile agent to build a map of the environment and compute in real-time the pose of the agent in that map. In contrast, Visual Odometries (VO) put their focus on computing the agent's ego-motion, not on building a map. The big advantage of a SLAM map is that it allows matching and using in estimation problem previous observations performing three types of data association (extending the terminology used by [17]):

- **Short-term data association**, matching map elements obtained during the last few seconds. This is the only data association type used by most VO systems, that forget environment elements once they get out of view, resulting in continuous estimation drift even when the system moves in the same area.

- **Mid-term data association**, matching map elements that are close to the camera whose accumulated drift is still small. These can be matched and used in the estimation problem in the same way than short-term observations and allow to reach zero drift when the systems moves in mapped areas. They are the key of the better accuracy obtained by SLAM systems compared against VO systems with loop detection.

- **Long-term data association**, matching observations with elements in previously visited areas using a place recognition technique, regardless of the accumulated drift (loop detection) or even if the tracking was lost (relocation). Long term matchings allow to reset the drift and to correct the loop using pose-graph (PG) optimization, or more accurately, using Bundle Adjustment (BA). This is the key of SLAM accuracy in medium and large loopy environments.

In addition to visual SLAM, we will also combine visual sensors with IMUs, which is named as **Visual-Inertial SLAM**. IMUs are high frequency, very compact, energetically efficient and inexpensive sensors, which can be mounted on hand-held or wearable devices. These features make them suitable for Augmented and Virtual Reality applications for smart-phones or AR/VR headsets.

Solving the SLAM problem consists in finding the optimal estimates for the agent and map state given a set of measurements [23, 38]. There exist two approaches for solving it [17]. The first one is a **filter based** approach, where SLAM is stated as an optimal estimation problem where only current agent pose and map variables are optimized, while previous ones are marginalized out. Under Gaussian noise assumption this leads to systems based on Kalman Filter (KF) formulation, namely extended KF for quasi linear systems, unscented KF for non linear systems, or particle filters for non Gaussian noise models. They have the advantage of not only estimating the system state, but also the uncertainty of the estimation. Marginalizing previous states have two side effects. First, it removes the sparse nature of the problem which makes filter approaches scale cubically (or quadratically for some efficient implementations such as [104]) with the size of the entire state vector in terms of computational cost. Second, it fixes the linearization point not being possible to change it later and leading to inconsistencies as early noticed by [1, 24]. These features make filter based approaches suitable for real-time applications mainly focused on robot localization.

The second approach is the **keyframe based** [73] or Bundle Adjustment (BA) optimization based solutions, where a set of robot poses and map elements are optimized together. Independent map elements assumption confers a sparse structure to the underlying optimization problem which can be easily exploited, leading to a computational cost which scales only with the number of optimized poses. To reduce the computational burden, not all robot poses are used, but only a smaller set of them, named as keyframes. In addition, as the estimation improves iteration after iteration, linearization points are also updated, making the linear approximation more accurate than filter based approaches. Comparing both solutions, keyframe based ones offer a more precise solution in terms of robot localization as reported by [125], as well as a more complete and precise map. During this thesis we will only focus on optimization based approaches.

1.2 Related Work

This related work section is joint work with the rest of the authors of [22]

In this thesis we will be focused on visual and visual-inertial SLAM. Table 1.1 presents a summary of both groups, showing the main techniques used for estimation and data association. The qualitative accuracy and robustness ratings included in this table are based on the results presented in section 4.4, and the comparison between PTAM, LSD-SLAM and ORB-SLAM reported by [99]

1.2.1 Visual SLAM

Monocular SLAM was first solved in MonoSLAM [34] being later improved in [28, 35]. It used an Extended Kalman Filter (EKF) and Shi-Tomasi points that were tracked in subsequent images doing a guided search by correlation. Mid-term data association was significantly improved using techniques that guarantee that the feature matches used are consistent, achieving hand-held visual SLAM for the first time [30, 31].

In contrast, keyframe-based approaches estimate the map using only a few selected frames, discarding the information coming from intermediate frames. This allows to perform the more costly, but more accurate, BA optimization at keyframe rate. The most representative system was PTAM [73], that split camera tracking and mapping in two parallel threads. Keyframe-based techniques are more accurate than filtering for the same computational cost as highlighted by [125], becoming the gold standard in visual SLAM and Visual-Odometry (VO). Large scale monocular SLAM was achieved at [126] using sliding-window BA, and at [124] using a double-window optimization and a covisibility graph.

Building on these ideas, ORB-SLAM [97, 99] uses ORB features, whose descriptor provides short-term and mid-term data association, builds a covisibility graph to limit the complexity of tracking and mapping, and performs loop-closing and relocalization using the bag-of-words library DBoW2 [54], achieving long-term data association. To date it is the only visual SLAM system integrating the three types of data association, which we believe is the key to its excellent accuracy.

Direct methods do not extract features, but directly use the pixel intensities in the images, and estimate motion and structure by minimizing a photometric error. LSD-SLAM, [45], was able to build large scale semi-dense maps using high gradient pixels. However, map estimation was reduced to pose-graph, achieving lower accuracy than PTAM and ORB-SLAM. The hybrid system SVO, [50, 53], extracts FAST features, uses a direct method to track features and any pixel with nonzero intensity gradient, from frame to frame, and optimizes camera trajectory and 3D structure using reprojection error. SVO is extremely efficient, but being a pure VO method, it only performs short-term data association, which limits its accuracy. Direct Sparse Odometry DSO, [46], is able to compute accurate camera poses in situations where point detectors perform poorly, enhancing robustness in low textured areas or against blurred images. It introduces local photometric BA that simultaneously optimizes a window of 7 recent keyframes and the inverse depth of the points. Extensions of this work include stereo [140], loop-closing using features and DBoW2 [56, 78], visual-inertial odometry [137] and monocular with depth prediction [145, 146]. Direct Sparse Mapping DSM [154] introduces the idea of

Table 1.1: Summary of the most representative visual (top) and visual-inertial (bottom) systems, in chronological order.

	SLAM or VO	Pixels used	Data association	Estimation	Relocation	Loop closing	Multi Maps	Mono	Stereo	Mono IMU	Stereo IMU	Fisheye	Accuracy	Robustness	Open source
Mono-SLAM [34, 35]	SLAM	Shi Tomasi	Correlation	EKF	-	-	-	✓	-	-	-	-	Fair	Fair	[72][1]
PTAM [73, 74, 75]	SLAM	FAST	Pyramid SSD	BA	Thumbnail	-	-	✓	-	-	-	-	Very Good	Fair	[76]
LSD-SLAM [42, 45]	SLAM	Edgelets	Direct	PG	-	FABMAP PG	-	✓	✓	-	-	-	Good	Fair	[44]
SVO [50, 53]	VO	FAST+ Hi.grad.	Direct	Local BA	-	-	-	✓	✓	-	-	✓	Very Good	Very Good	[49][2]
ORB-SLAM2 [97, 99]	SLAM	ORB	Descriptor	Local BA	DBoW2	DBoW2 PG+BA	-	✓	✓	-	-	-	Exc.	Very Good	[100]
DSO [46, 91, 140]	VO	High grad.	Direct	Local BA	-	-	-	✓	✓	-	-	✓	Fair	Very Good	[43]
DSM [154]	SLAM	High grad.	Direct	Local BA	-	-	-	✓	-	-	-	-	Very Good	Very Good	[153]
MSCKF [82, 94, 102, 103]	VO	Shi Tomasi	Cross correlation	EKF	-	-	-	✓	-	✓	✓	-	Fair	Very Good	[25][3]
OKVIS [79, 80]	VO	BRISK	Descriptor	Local BA	-	-	-	-	-	✓	✓	✓	Good	Very Good	[81]
ROVIO [12, 13]	VO	Shi Tomasi	Direct	EKF	-	-	-	-	-	✓	✓	✓	Good	Very Good	[11]
ORBSLAM-VI [98]	SLAM	ORB	Descriptor	Local BA	DBoW2	DBoW2 PG+BA	-	✓	-	✓	-	-	Very Good	Very Good	-
VINS-Fusion [108, 110]	VO	Shi Tomasi	KLT	Local BA	DBoW2	DBoW2 PG	✓	-	✓	✓	✓	✓	Good	Exc.	[109]
VI-DSO [137]	VO	High grad.	Direct	Local BA	-	-	-	-	-	✓	-	-	Very Good	Exc.	-
BASALT [136]	VO	FAST	KLT (LSSD)	Local BA	-	ORB BA	-	-	-	-	✓	✓	Very Good	Exc.	[135]
Kimera [114]	VO	Shi Tomasi	KLT	Local BA	-	DBoW2 PG	-	-	-	-	✓	-	Good	Exc.	[113]
ORB-SLAM3 (ours)	SLAM	ORB	Descriptor	Local BA	DBoW2	DBoW2 PG+BA	✓	✓	✓	✓	✓	✓	Exc.	Exc.	[20]

[1] Last source code provided by a different author. Original software is available at [33].

[2] Source code available only for the first version, SVO 2.0 is not open source.

[3] MSCKF is patented [116], only a re-implementation by a different author is available as open source.

map reusing in direct methods, showing the importance of mid-term data association. In all cases, the lack of integration of short, mid, and long-term data association results in lower accuracy than our SLAM proposal (see section 4.4).

1.2.2 Visual-Inertial SLAM

The combination of visual and inertial sensors provides robustness to poor texture, motion blur and occlusions, and in the case of monocular systems, makes scale observable. These features make visual-inertial SLAM the most suitable solution for AR/VR applications, being therefore a very active field of research in both the academia and industry.

Research in tightly coupled approaches can be traced back to MSCKF [94], where the EKF quadratic cost in the number of features is avoided by feature marginalization. The initial system was perfected in [82] and extended to stereo in [102, 103]. The first tightly coupled visual-inertial odometry system based on keyframes and bundle adjustment was OKVIS [79, 80], that is also able to use monocular and stereo vision. While these systems rely on features, ROVIO [12, 13] feeds an EFK with photometric error using direct data association.

ORB-SLAM-VI, [98], presented for the first time a visual-inertial SLAM system able to reuse a map with short-term, mid-term and long-term data associations, using them in an accurate local visual-inertial BA based on IMU preintegration [52, 87]. However, its IMU initialization technique was too slow, taking 15 seconds, which harmed robustness and accuracy. In addition, this system only managed pinhole monocular cameras, which limited its applications. VINS-Mono [108] is a very accurate and robust monocular-inertial odometry system, with loop-closing using DBoW2 and 4 DoF pose-graph optimization, and map-merging. Feature tracking is performed with Lucas-Kanade tracker, being slightly more robust than descriptor matching. In VINS-Fusion [110] it has been extended to stereo and stereo-inertial configurations.

VI-DSO [137] extends DSO to visual-inertial odometry. They propose a bundle adjustment which combines inertial observations with the photometric error in selected high gradient pixels, which renders very good accuracy. As the information in high gradient pixels is successfully exploited, the robustness in scene regions with poor texture is also boosted. Their initialization method relies on visual-inertial BA and takes 20-30 seconds to converge within 1% scale error.

The recent BASALT [136] is a stereo visual-inertial odometry system that extracts non-linear factors from visual-inertial odometry to use them in BA, and closes loops matching ORB features, achieving very good to excellent accuracy. Kimera [114] is a novel outstanding metric-semantic mapping system, but its metric part consists in stereo-inertial odometry plus loop closing with DBoW2 and pose-graph optimization, achieving similar accuracy to VINS-Fusion.

1.3 Contributions

This thesis aims to develop solutions for making visual and visual inertial SLAM systems more robust to real situations. In this sense, our contributions are:

- **A joint visual-inertial initialization method** [19]: We build on the initialization method proposed by Martinelli [90] and extended by Kaiser et al. [69], modifying it to be more general and efficient. We improve accuracy with several rounds of visual-inertial bundle adjustment, and robustify the method with novel observability and consensus tests that discard erroneous solutions. Our results show that our method is able to initialize in less than two seconds with scale errors around 5%, which can be further reduced to less than 1% performing visual-inertial bundle adjustment after ten seconds. This contribution is detailed in section 3.2

- **An inertial-only optimization method for IMU initialization** [21]: We formulate for the first time Visual-Inertial initialization as an optimal estimation problem, in the sense of maximum-a-posteriori (MAP) estimation. This allows us to properly take into account IMU measurement uncertainty, which was neglected in previous methods that solved sets of algebraic equations or minimized ad-hoc cost functions using least squares. Our exhaustive initialization tests on EuRoC dataset show that our proposal outperforms the best methods in the literature, being able to initialize in 2 seconds in almost any point of the trajectory, with a scale error lower than 5% on average. This contribution is introduced in section 3.3.

- **ORB-SLAM3: A robust visual-inertial system** [22]: We propose ORB-SLAM3, the first system able to perform visual, visual-inertial and multi-map SLAM with monocular, stereo and RGB-D cameras, using pin-hole and fisheye lens models. The main contribution from this thesis is a feature-based tightly-integrated visual-inertial SLAM system that fully relies on Maximum-a-Posteriori (MAP) estimation, even during the IMU initialization phase. The result is a system that operates robustly in real time, in small and large, indoor and outdoor environments, and is two to ten times more accurate than previous approaches. Our detailed experiments show that, in all sensor configurations, ORB-SLAM3 is as robust as the best systems available in the literature, and significantly more accurate. Notably, our stereo-inertial SLAM achieves an average accuracy of 3.6 cm on the EuRoC drone and sub-centimeter accuracy under quick hand-held motions in the room of TUM-VI dataset, a setting representative of AR/VR scenarios. This contribution is presented in chapter 4.

- **DynaSLAM II: A SLAM system with dynamic objects** [9]: The assumption of scene rigidity is common in visual SLAM algorithms. However, it limits their applicability in populated real-world environments. We propose DynaSLAM II, a visual SLAM system for stereo and RGB-D configurations that tightly integrates multi-object tracking capability. The specific contribution of this thesis is the formulation of a BA where, structure of the static scene and of the dynamic objects is jointly optimized with the trajectories of both the camera and the moving agents. We demonstrate that tracking dynamic objects does not only provide rich clues for scene understanding but is also beneficial for camera tracking. This contribution is described in section 5.1

- **A scale-aware direct monocular odometry using depth predictions** [18]: Pure monocular SLAM has scale ambiguity, being not possible to recover it. We introduce a monocular scale-aware system, based on depth predictions from a deep neural network. We propose a tightly-coupled optimization that combines photometric and predicted depth residuals from multiple views and significantly improves accuracy with respect to monocular solutions, completely removing the scale drift issue. This is validated in the KITTI odometry dataset, compared against state-of-the-art monocular systems and other similar solutions, outperforming them. This contribution is described in section 5.2

1.4 Dissemination

1.4.1 Peer-reviewed publications

The main results of this thesis have resulted in the following publications:

- Campos, C., Montiel, J. M., & Tardós, J. D.: Fast and robust initialization for visual-inertial SLAM. In IEEE International Conference on Robotics and Automation (ICRA), pp. 1288-1294, May 2019

- Campos, C., Montiel, J. M., & Tardós, J. D.: Inertial-only optimization for visual-inertial initialization. In IEEE International Conference on Robotics and Automation (ICRA), pp. 51-57, May 2020

- Bescos, B., Campos, C., Tardós, J. D., & Neira, J.: DynaSLAM II: Tightly-coupled multi-object tracking and SLAM. IEEE Robotics and Automation Letters, 6(3): 5191-5198, July 2021

- Campos, C., Elvira, R., Rodríguez, J. J. G., Montiel, J. M., & Tardós, J. D.: ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM. IEEE Transactions on Robotics, published online, May 2021

There is a publication currently under review, submitted to the IEEE International Conference on Robotics and Automation (ICRA) 2022, corresponding to the latest work in a monocular direct odometry based on depth-predictions.

- Carlos Campos and Juan D Tardós. Scale-aware direct monocular odometry. arXiv preprint arXiv:2109.10077, 2021.

1.4.2 Licensed software

The University of Zaragoza has licensed to companies in Asia, America and Europe two software libraries jointly developed during this thesis, for their use in commercial products:

- Juan D. Tardós, José M. M. Montiel, Carlos Campos, Richard Elvira: Biblioteca Software ORB-SLAM Stereo-Inertial, Registro General de la Propiedad Intelectual 10/2019/571, 27-Nov-2019. Licensed to a company in China.

- Juan D. Tardós, José M. M. Montiel, Carlos Campos, Richard Elvira, Juan J. Gómez: Biblioteca Software ORB-SLAM3, Registro General de la Propiedad Intelectual 10/2021/364, 5-Jul-2021. Licensed to 5 companies in UK, Norway, Germany, Israel and USA
  - code: `https://github.com/UZ-SLAMLab/ORB_SLAM3`
  - multimedia: `https://www.youtube.com/channel/UCXVt-kXG6T95Z4tVaYlU80Q`

Chapter 2

Optimization based SLAM

In this chapter we present the necessary fundamentals for optimization based visual and visual-inertial SLAM.

2.1 Visual SLAM

Visual SLAM stands for systems which use information from cameras to solve the localization of the robot and mapping of the environment in real time. Information acquired from cameras is represented by images I, which are functions from a two dimensional domain Ω to a subset of the positive real space (for gray scale images), usually $[0, 255]$, such that:

$$I : \Omega \rightarrow [0, 255] \tag{2.1}$$

A 3D point $\mathbf{x}$ in the world reference frame is projected into the camera according to a surjective map $\pi : \mathbb{R}^3 \rightarrow \Omega$:

$$\mathbf{u} = \pi(\mathbf{T}_{\mathtt{CW}} \oplus \mathbf{x}) \tag{2.2}$$

being $\mathbf{u} \in \Omega$ the map point projection and $\mathbf{T}_{\mathtt{CW}} \in \mathrm{SE}(3)$ the rigid body transformation from world ($\mathtt{W}$) reference frame to camera ($\mathtt{C}$) frame. This is composed of a rotational part $\mathbf{R}_{\mathtt{CW}} \in \mathrm{SO}(3)$ and a translation $\mathbf{p}_{\mathtt{CW}} \in \mathbb{R}^3$. $\mathbf{T}_{\mathtt{CW}}$ allows us to transform 3D points from world to camera reference, defining the following transformation operator $\oplus$:

$$\oplus : \mathrm{SE}(3) \times \mathbb{R}^3 \rightarrow \mathbb{R}^3 \tag{2.3}$$

$$\mathbf{T} \oplus \mathbf{x} \rightarrow \mathbf{R}\mathbf{x} + \mathbf{p} \tag{2.4}$$

Along this text, when there is not possible confusion and for clarity sake, we will also omit the $\oplus$ operator. In addition, we will indistinctly use $\oplus$ also for SE(3) compositions, such that $\oplus : \mathrm{SE}(3) \times \mathrm{SE}(3) \rightarrow \mathrm{SE}(3)$, which is equivalent to usual matrix multiplication.

The most simple way of parametrizing the map is by means of a set of 3D points, also named as map points or landmarks. Map points can be directly parameterized by its 3D position $\mathbf{x} = (x, y, z)$, or by mean of its inverse depth [29] ρ and its image observation $\mathbf{u} \in \Omega$ in a reference frame, usually named as host (h) or anchor. In that case, point's world coordinates are computed as:

$$\mathbf{x} = \mathbf{T}_{\mathtt{W}h} \oplus \pi^{-1}(\mathbf{u}, \rho) \tag{2.5}$$

where $\mathbf{T}_{\mathtt{W}h}$ is the relative transformation from host (h) keyframe to world, and π^{-1} : $\Omega \times \mathbb{R} \rightarrow \mathbb{R}^3$ stands for the injective inverse function, mapping an image point and its inverse depth to the respective 3D point at camera reference frame. Map π, and its inverse π^{-1}, may take different forms depending on the camera model, and it is defined by its calibration parameters. Along this thesis we will keep an abstract formulation for π, making it general for any camera model. For details about specific camera models, see appendix B.1

There exist other ways of parametrizing the geometry of the environment than 3D points. Some of them make use of other geometry entities like lines [60, 106] or planes [67, 148]. Other estimate 3D points and use them to build meshes [114], having a more complete and dense reconstruction but without modifying the underlying representation. When there is a prior knowledge about objects and entities that may exist in the scene, it is possible to parametrize all points belonging to the same object with just a SE(3) transformation, which supposes a big compression [55, 120].

Recently, leveraging the advancements on compact and implicit representation using deep-learning models, new parametrizations have been adopted for SLAM. In this sense, some works [14, 32, 92] propose to use a encoder-decoder structure, where the scene is encoded in a latent space with a low dimensionality, being possible to estimate it on real time. Recent work on volumetric neural representation [93] allows using a Multi Layer Perceptron (MLP) to render very realistic new views of the scene. This has an immediate application to the SLAM problem, opening a new line of research for the SLAM community, which has lead to the first results [129]

Other information than geometry, such as semantics, may be used to characterize the map. In this thesis we will be limited to map representation with 3D points. For this case, depending on how image information is exploited we find two big visual SLAM groups which are presented in the following sections.

2.1.1 Feature based SLAM

From each image I, feature based methods extract a set of interest points, also known as features or keypoints, usually corresponding with corners, which have intensity gradients in two directions. Exploiting these characteristic points allows to perform robust data association, as well as making possible place recognition [85] for relocalization and loop closing based on Bag of Words [54]. Feature based SLAM leads to sparse maps, typically having a lower computational cost than its counterparts direct methods. These keypoints may be tracked along different images in two different way:

- Extracting and matching **descriptors**. Each keypoint and its surrounding pixels in the image are summed up in an array of binary or real values (i.e. SIFT, SURF or ORB), which have some nice properties like being translation and rotational invariant, or robust against luminosity changes. For a pair of descriptors a distance is defined, thus matching may be performed by finding the descriptors with the lowest distance along the different images.

- By **patch correlation**. A patch surrounding each point is tracked along images by minimizing the intensity difference of all its pixels with respect to the reference image (i.e. Lucas-Kanade). This allows to exploit more information from the image since the condition for extracting keypoints may be relaxed and points' extraction

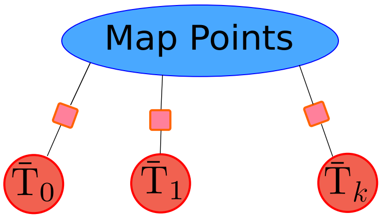


Figure 2.1: Factor graph representation for pure visual Bundle Adjustment. Reprojection or photometric residuals (red boxes) relate map points' positions and camera poses.

does not need to be done for each frame. However, it is less robust to luminosity or point of view changes and has a higher computational cost.

In both cases, we end up with a set of matches $\{\mathbf{u}_1^j \dots \mathbf{u}_n^j | \mathbf{u}_i \in \Omega\}$ for a given 3D point $\mathbf{x}^j$. For each of this matches, a residual error $\mathbf{r}_{i,\mathrm{proj}}^j$ based on reprojection error is defined as:

$$\mathbf{r}_{i,\mathrm{proj}}^j = \mathbf{u}_i^j - \pi(\mathbf{T}_{i\mathtt{W}} \oplus \mathbf{x}^j) \tag{2.6}$$

This error is the difference in pixels between the estimation $\pi(\mathbf{T}_{i\mathtt{W}} \oplus \mathbf{x}^j)$ and the measurement $\mathbf{u}_i^j$. Although we have assumed a xyz parametrization, an equivalent expression may be derived for inverse depth. Considering all keyframes $\mathcal{K}$, all points $\mathcal{P}$ and their matches, and being $\mathcal{P}_i$ the set of points observed from keyframe i, we can define the following least-square optimization problem:

$$\{\mathbf{T}_{i\mathtt{W}}, \mathbf{x}^j | \forall i \in \mathcal{K}, j \in \mathcal{P}\} = \underset{\mathbf{T}_{i\mathtt{W}}, \mathbf{x}^j}{\arg\min} \sum_{i \in \mathcal{K}} \sum_{j \in \mathcal{P}_i} \rho_{\mathrm{Hub}} \left(\left\| \mathbf{r}_{i,\mathrm{proj}}^j \right\|_{\Sigma_i^j}^2 \right) \tag{2.7}$$

This least-squares optimization problem is known as Bundle Adjustment (BA), [133], and it is the main SLAM back-end process which needs to run at keyframe frequency. This optimization can be stated as a factor graph [77], as represented in figure 2.1, and under Gaussian noise assumption it is equivalent to finding the state which maximizes the posterior distribution given the visual matches.

A Mahalanobis norm is used to weight measurements with the inverse of its covariance matrix Σ_i^j, usually composed with a robust kernel such as Huber (ρ_{Hub}). This prevents quadratically weighting outliers, reducing its influence and rendering the optimization more robust. Common iterative methods for solving this optimization are Levenberg-Marquardt and Gauss-Newton, which entails solving a linear system with a symmetric definite matrix, in such a way that efficient and stable Cholesky decomposition may be used (See A.1 for more details).

2.1.2 Direct SLAM

The second approach consists in using the image information as it is, without extracting keypoints nor descriptors. A subsampling is performed to obtain points with high

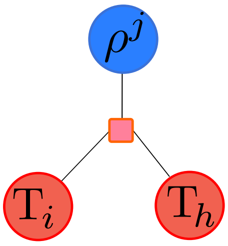


Figure 2.2: Factor representation for a photometric residual given inverse depth representation. In contrast with reprojection error and xyz parametrization, host is also included. This requires derivatives to be computed also with respect to host pose, but this may be efficiently done from observer derivatives (see section B.3)

intensity gradient, as well as a pyramid representation is built to increase the basin of convergence. The Bundle Adjustment optimization is built formulating a photometric cost, which consists in directly comparing intensity values from the image at different observer keyframes and that one of reference (host). This usually implies an inverse depth parametrization. The photometric residual for point j seen from keyframe i and hosted at keyframe h is defined as:

$$\mathbf{r}^j_{i,\text{photo}} = I_h(\mathbf{u}^j_h) - \alpha_{hi} I_i(\pi(\mathbf{T}_{ih} \oplus \pi^{-1}(\mathbf{u}^j_h, \rho^j))) - \beta_{hi} \tag{2.8}$$

Terms α_{hi} and β_{hi} define a relative affine transformation which is used to remove the effect of light changes or exposure value between keyframes. This residual can be extended to patches [46], where instead of comparing the intensity of single pixels, we use a set $\mathcal{N}_{\mathbf{u}_j}$ of surrounding pixels, such that:

$$\mathbf{r}^j_{i,\text{photo}} = \sum_{\mathbf{u}' \in \mathcal{N}_{\mathbf{u}_j}} I_h(\mathbf{u}') - \alpha_{hi} I_i(\pi(\mathbf{T}_{ih} \oplus \pi^{-1}(\mathbf{u}', \rho^j))) - \beta_{hi} \tag{2.9}$$

Under the assumption that all patch pixels have the same inverse depth and same geometric derivatives, this does not imply an important increment in the computational cost, while it makes points to be better conditioned. Regarding the factor graph representation, the photometric residual does not only relate observer pose and map point position, but also host pose, as represented in figure 2.2

With this defined photometric residual, one can build an equivalent optimization problem to equation 2.7, such that:

$$\{\mathbf{T}_{i\mathtt{W}}, \alpha_i, \beta_i, \rho^j | \forall i \in \mathcal{K}, j \in \mathcal{P}\} = \underset{\mathbf{T}_{i\mathtt{W}}, \alpha_i, \beta_i, \rho^j}{\arg\min} \sum_{i \in \mathcal{K}} \sum_{j \in \mathcal{P}_i} \rho_{\text{Hub}} \left(\left\| \mathbf{r}^j_{i,\text{photo}} \right\|^2_{\Sigma^j_i} \right) \tag{2.10}$$

Since this formulation may contain much more minima and a smaller basin of convergence compared with feature based BA, a pyramidal implementation is commonly used. This consists of subsampling the image at different levels, getting lower resolution images. Optimization stars at the highest level and continues until original image resolutions. This multistage optimization increases the basin of convergence, not being necessary to have such a precise initial seed. It is worth noting that this formulation does not have a matching procedure like that one for featured based SLAM. Instead, the correspondences are found while solving the optimization problem, checking if the intensity difference, between reference patch and that one corresponding with the reprojection, is below some threshold.

This photometric formulation allows in theory to exploit the whole set of pixels from an image. However, there are some limitations which make it not feasible, such as points with no image gradient or computational complexity. For these reasons, most direct SLAM systems do not work with all pixels but with a smaller set of points with high gradient. This set is larger than that one from featured based systems, leading to a more complete representation on the environment as we will see in section 5.2, but not necessarily to a more precise robot localization as shown in section 4.4.

There exist some approaches [78] which use both formulation, using features and high gradient points, combining reprojection and photometric residuals. While photometric points allow to exploit more information from the image, features eases data association and place recognition.

2.2 Visual-Inertial SLAM

Cameras provide very rich information about the environment. However, pure visual SLAM systems are prone to fail in challenging situations, namely fast motion leading to motion blur, temporal occlusions or featureless scenes. In addition, there is information, such as the global pose, scale for monocular sensors, instant acceleration or angular velocity, which can not be directly retrieved from visual sensors. In this sense, coming back to the BA equation 2.7, it is worth saying that the optimum for this minimization is not unique. In fact, once we have found an optimal solution, if we apply an arbitrary SE(3) transformation to all keyframes and points, the global cost is the same. This is denoted as gauge freedom, and usually removed by arbitrarily fixing the 6 degrees of freedom (DoF) of the first keyframe of the map. In particular, for a pure monocular SLAM system, scale is also not observable, having an extra DoF, which also leads to the well-known problem of scale drift, as reported by [126]. To remove its gauge freedom one can fix the distance between the first two keyframes.

All these drawbacks may be mitigated by using an Inertial Measurement Unit (IMU). This is a proprioceptive sensor which measures the linear acceleration and the angular velocity in the sensor reference frame, at a much higher frequency than visual sensors. For short periods, it is possible to accurately estimate the pose and velocity changes of the robot using just a single IMU. In addition, these measurements allow to render observable the scale of the map for a monocular camera, as well as the gravity direction. This reduces the gauge freedom of the problem to just 4 degrees, namely translation and yaw.

IMU is composed of two sensors, the accelerometer and gyroscope, which do not measure the true linear acceleration $\mathbf{a}$ nor angular velocity $\boldsymbol{\omega}$, but quantities $\tilde{\mathbf{a}}$ and $\tilde{\boldsymbol{\omega}}$, which are affected by noises (η^a, η^g), biases $(\mathbf{b}^a, \mathbf{b}^g)$ and gravity $\mathbf{g}$. For an instant i measurements take the following expressions:

$$\begin{aligned} \tilde{\mathbf{a}}_i &= \mathbf{R}_i^{\mathsf{T}} \left(\mathbf{a}_i - \mathbf{g} \right) + \mathbf{b}_i^a + \eta_i^a \in \mathbb{R}^3 \\ \tilde{\boldsymbol{\omega}}_i &= \boldsymbol{\omega}_i + \mathbf{b}_i^g + \eta_i^g \in \mathbb{R}^3 \end{aligned} \tag{2.11}$$

where $\mathbf{R}_i \in \mathrm{SO}(3)$ stands for the rotation from world `W` to IMU or body reference `B` at time i. Biases are assumed to evolve according to random walks η_{rw}^a and η_{rw}^g. Given an

IMU measurement, two consecutive states relate as:

$$\begin{aligned}
\mathbf{R}_{i+1} &= \mathbf{R}_i \mathrm{Exp}\left(\left(\tilde{\boldsymbol{\omega}}_i - \mathbf{b}_i^g\right)\Delta t\right) \\
\mathbf{v}_{i+1} &= \mathbf{v}_i + \mathbf{g}\Delta t + \mathbf{R}_i\left(\tilde{\mathbf{a}}_i - \mathbf{b}_i^a\right) \\
\mathbf{p}_{i+1} &= \mathbf{p}_i + \mathbf{v}_i\Delta t + \frac{1}{2}\mathbf{g}\Delta t^2 + \frac{1}{2}\mathbf{R}_i\left(\tilde{\mathbf{a}}_i - \mathbf{b}_i^a\right) \\
\mathbf{b}_{i+1}^a &= \mathbf{b}_i^a + \eta_{rw,i}^a \\
\mathbf{b}_{i+1}^g &= \mathbf{b}_i^g + \eta_{rw,i}^g
\end{aligned} \tag{2.12}$$

Here, $\mathbf{T}_i = [\mathbf{R}_i, \mathbf{p}_i] \in \mathrm{SE}(3)$ stands for the transformation from IMU reference at time i to world, while $\mathbf{v}_i$ for its velocity in the world frame at instant i. Exp maps a vector from the tangent space $\mathfrak{so}(3)$, isomorphic to $\mathbb{R}^3$, to the Lie group SO(3), as explained in appendix section D.2.3. From these equations, which describe the dynamics of the system, one can see that state needs to be expanded to also include keyframe velocities and biases, in addition to camera poses and map points positions from pure visual SLAM.

2.2.1 IMU integration and residuals

Since IMU works at a much higher rate than visual sensors, there will exist several inertial measurements between each pair of consecutive frames or keyframes. The naive approach for estimating keyframe j state from keyframe i state would be to iteratively use equations 2.12. However, this expensive procedure should be repeated each time state i is modified, which is too expensive.

Instead, as proposed by [87] and further developed by [52], IMU measurements may be summarized in preintegrated terms. These need to be computed just once and may be efficiently used to recompute state j if state i is modified. These integrated terms are: $\Delta\mathbf{R}_{ij}$ accounting for rotational change, $\Delta\mathbf{v}_{ij}$ for velocity change, and $\Delta\mathbf{p}_{ij}$ for position change. They only depend on IMU measurements between time i and j as well as their biases, and can be used to estimate state j as follows:

$$\begin{aligned}
\mathbf{R}_j &\approx \mathbf{R}_i \Delta\mathbf{R}_{ij}(\mathbf{b}_i^g) \\
\mathbf{v}_j &\approx \mathbf{v}_i + \mathbf{g}\Delta t_{ij} + \mathbf{R}_i \Delta\mathbf{v}_{ij}(\mathbf{b}_i^g, \mathbf{b}_i^a) \\
\mathbf{p}_j &\approx \mathbf{p}_i + \mathbf{v}_i \Delta t_{ij} + \frac{1}{2}\mathbf{g}\Delta t_{ij}^2 + \mathbf{R}_i \Delta\mathbf{p}_{ij}(\mathbf{b}_i^g, \mathbf{b}_i^a) \\
\mathbf{b}_j^a &\approx \mathbf{b}_i^a \\
\mathbf{b}_j^g &\approx \mathbf{b}_i^g
\end{aligned} \tag{2.13}$$

where $\approx$ is used to emphasize that right hand sides are affected by measurement and model noises. With these new expressions, inertial measurements should be reintegrated only when biases are updated. Instead of doing that and noticing that biases do not suffer big variations, [87] proposed to use a linear approximation. In this way, when a bias is updated by $\delta\mathbf{b}$, the new integrated term can be computed as:

$$\Delta\mathbf{p}_{ij}(\mathbf{b}_i^a + \delta\mathbf{b}^a, \mathbf{b}_i^g + \delta\mathbf{b}^g) = \Delta\mathbf{p}_{ij}(\mathbf{b}_0) + \left.\frac{\partial\Delta\mathbf{p}_{ij}}{\partial\mathbf{b}_i^a}\right|_{\mathbf{b}_i^a} \delta\mathbf{b}^a + \left.\frac{\partial\Delta\mathbf{p}_{ij}}{\partial\mathbf{b}_i^g}\right|_{\mathbf{b}_i^g} \delta\mathbf{b}^g \tag{2.14}$$

with equivalent expressions for integrated rotation and velocity. Thanks to this formulation, inertial measurements need to be integrated just once, lately updating preintegrated

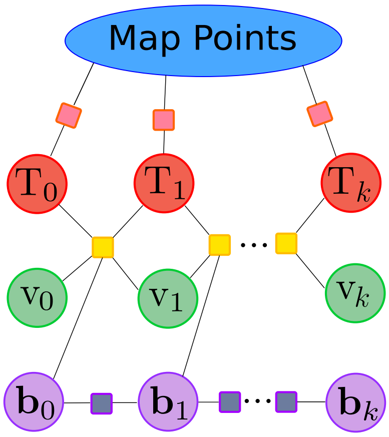


Figure 2.3: Factor graph representation for Visual-Inertial Bundle Adjustment. Visual residuals (red boxes) relates map points positions and body poses, while IMU preintegrated residuals (yellow boxes) connect consecutive keyframes states.

terms by mean of previous expressions. Details about how to efficiently compute these preintegrated terms and their derivatives are provided at [52].

In addition, from (2.13), [52] formulates a set of inertial residuals taking the difference from the left and right hand side, such that:

$$\begin{aligned}
\mathbf{r}^{ij}_{\Delta\mathbf{R}} &= \mathrm{Log}\left(\mathbf{R}_j^{\mathtt{T}}\mathbf{R}_i\Delta\mathbf{R}_{ij}\right)\\
\mathbf{r}^{ij}_{\Delta\mathbf{v}} &= \mathbf{v}_j - \mathbf{v}_i - \mathbf{g}\Delta t_{ij} - \mathbf{R}_i\Delta\mathbf{v}_{ij}\\
\mathbf{r}^{ij}_{\Delta\mathbf{p}} &= \mathbf{p}_j - \mathbf{p}_i - \mathbf{v}_i\Delta t_{ij} - \frac{1}{2}\mathbf{g}\Delta t_{ij}^2 - \mathbf{R}_i\Delta\mathbf{p}_{ij} \qquad (2.15)\\
\mathbf{r}^{ij}_{\Delta\mathbf{b}^a} &= \mathbf{b}^a_j - \mathbf{b}^a_i\\
\mathbf{r}^{ij}_{\Delta\mathbf{b}^g} &= \mathbf{b}^g_j - \mathbf{b}^g_i
\end{aligned}$$

where Log : SO(3) → $\mathfrak{so}(3)$ is the inverse map of Exp, as described in appendix section D.2.3. All these residuals are three dimensional vectors, and covariances for all of them are provided at [52], making possible to combine them together with residuals from other sensors. The foundations of these residuals are deeply related with matrix Lie Groups theory, whose most used concepts are covered in the appendix D.1. We finally define the full inertial residual $\mathbf{r}^{ij}_{\mathcal{I}}$ as:

$$\mathbf{r}^{ij}_{\mathcal{I}} = [\mathbf{r}^{ij}_{\Delta\mathbf{R}}, \mathbf{r}^{ij}_{\Delta\mathbf{v}}, \mathbf{r}^{ij}_{\Delta\mathbf{p}}, \mathbf{r}^{ij}_{\Delta\mathbf{b}^a}, \mathbf{r}^{ij}_{\Delta\mathbf{b}^g}] \qquad (2.16)$$

and combined with visual measurements, a new optimization problem is found, such that:

$$\begin{aligned}
&\{\mathbf{T}_{i\mathtt{W}}, \mathbf{v}_i, \mathbf{b}^a_i, \mathbf{b}^g_i, \mathbf{x}^j | \forall i \in \mathcal{K}, j \in \mathcal{P}\} =\\
&\qquad \underset{\mathbf{T}_{i\mathtt{W}},\mathbf{v}_i,\mathbf{b}^a_i,\mathbf{b}^g_i,\mathbf{x}^j}{\arg\min} \sum_{i\in\mathcal{K}} \left\{ \sum_{j\in\mathcal{P}_i} \rho_{\mathrm{Hub}}\left(\left\|\mathbf{r}^j_{i,\mathrm{proj}}\right\|^2_{\Sigma^j_i}\right) + \left\|\mathbf{r}^{i,i+1}_{\mathcal{I}}\right\|_{\Sigma^{i,i+1}_{\mathcal{I}}} \right\} \qquad (2.17)
\end{aligned}$$

This problem is named as visual-inertial Bundle Adjustment, and replaces the visual BA presented in the previous section. Similarly, it can also be represented as a factor graph, as illustrated in figure 2.3. Since we are solving an iterative optimization problem (equations 2.7, 2.10 and 2.17) we need an initial seed for all unknowns. For visual variables, those unknowns are the two first body poses and an initial set of 3D landmark positions. Finding these initial guesses is known as visual initialization. On the other hand, for pure inertial variables such as velocities, biases, scale (for monocular SLAM) and gravity direction, finding their initial guesses is known as IMU initialization. Both initializations may be solved together, or vision initialization can be solved first and then be used to initialized the IMU. Section 3 of this thesis will be devoted to the IMU initialization.

Chapter 3

Inertial Initialization

3.1 Introduction

Launching a SLAM system from scratch is one of the trickiest parts since we do not have any estimation yet, but just measurements. For pure monocular visual SLAM, estimating the camera motion requires an estimation of the 3D structure, while for initially inferring the structure we need to have an estimated motion. To solve this problem, named as **visual initialization**, there exist several approaches. Assuming a feature based system, correspondences between points in the first two keyframes may be used to retrieve an homography model for planar scenes, or fundamental matrix model for the general case, [99, 131]. These two models encode the motion between both frames, which allows to triangulate the tracked points, having an initial estimate good enough to make converge equation 2.7. On the other hand, if a photometric approach is used, there are no initial correspondences and this procedure may not be applied. Some works like [45, 46] propose to randomly initialize points' inverse depth and set both camera poses to origin, and try to directly solve equation 2.10. Other proposals, [154], take the same approach as features based methods, and track some points just for initialization purpose.

For visual-inertial SLAM systems, this initial estimation includes new variables which need to be computed in the **inertial initialization**. From equation 2.11, we can see that gravity and IMU biases need to be known to properly compensate the IMU measurements. In addition, since the IMU obeys the dynamic laws, we need to work with true scale poses, being necessary to compute scale factor during the initialization step. Finally, the initial velocity needs also to be computed. Since we need to integrate the IMU measurements between each pair of keyframes, the IMU estimation is very sensitive to noise and errors in these parameters. In fact, the IMU model implies a double integration of the accelerometer measurements, which leads to double integration of these errors. This makes errors grow quadratically with integration time, thus being necessary a fairly good initial guess for IMU variables. For optimization based SLAM, initialization is required to have an initial seed in the basin of convergence of the optimal solution. Equivalent visual-inertial optimization problem (equation 2.17) is not globally convex and it is plagued with local minima.

Depending on whether both initializations are solved together or separately, we can differentiate two approaches. First, we distinguish the **joint visual-inertial initialization** which solves for all visual and inertial parameters at one step. They are very appealing as they allow the immediate launch of the visual-inertial SLAM. These methods are usually built on some strong assumptions, such as noiseless measurements, which allow to obtain elegant closed form solutions but limit their application. Solving these initializations typically consists of solving a set of algebraic equations. Within this group, we find some ad-hoc solutions. Most of visual-inertial filter based methods like [65] and [94] assume strong priors, like steady initial state. This reduces the requirement of the initialization, even making it not necessary, but may invalidate the linearization approximation for EKF formulation if the actual situation differs from assumption. [48] proposed an ad hoc multistage solution for aggressive flights with quadrotors, where the initialization algorithm leverages the fact that motion can be directly controlled and thus it is possible to perform one suitable for this procedure.

A more complete joint visual-inertial initialization, was introduced by [89, 90], who proposed a closed-form solution to jointly retrieve the scale, gravity, accelerometer bias and initial velocity, as well as visual features depth. This method was built on the assumption that camera poses can be roughly estimated from IMU readings. The method tracks several points, required to be seen in all the images, and builds a system of equations stating that the 3D point coordinates, as seen from any camera pair, should be the same. This system is solved by linear least squares. This paper also provides a theoretical analysis of the conditions for the problem to be solvable and presents simulated results. This work was extended by [69] that builds a similar linear algebraic system which is solved using non-linear least squares to also find gyroscope bias and to take gravity magnitude into account by means of a constrained optimization. Simulation and real results showed that it is possible to find accurate initial solutions in less than 3 seconds, with relative errors around 15% on speed and gravity. Crucially, the original and modified methods ignore IMU noise properties, and minimize the 3D error of points in space, and not their reprojection errors, that is the gold-standard in feature-based computer vision.

On the other hand, we have **disjoint (loosely-coupled) visual-inertial initialization**, where the divide and conquer approach is followed and two separate steps are performed. First, we solve for the initial motion and structure up to scale, and secondly, from these estimations and inertial measurements, we find the initial guesses for gravity, scale and biases. Between both steps, a visual SLAM may be performed, since motion used to initialize vision may not be enough for making IMU parameters observable. These methods require fewer assumptions and have a simpler formulation, but come with an important drawback. The IMU initialization heavily depends on visual initialization and pure visual SLAM. If the visual part is not able to initialize, or if it gets lost within a short period of time, the inertial system will not initialize. We also remark that joint methods do not necessarily imply a faster initialization, since the amount of data (length of the trajectory) for solving the problem is typically the same as for disjoint methods. As modern visual-odometry and visual SLAM systems perform local bundle adjustment and provide trajectories with much higher precision than IMU integration, these solutions are becoming more and more popular. Along these methods, we highlight the pioneering proposal by Mur-Artal and Tardós in ORB-SLAM-VI [98], which was later adopted by Qin et al. in VINS-Mono [107, 108]. In both cases, the inertial parameters are found in different steps by solving a set of linear equations using least-squares optimization. In [98] a linear system is built by eliminating the velocities for each frame. However, these algebraic manipulations make the minimized errors to be meaningless and unrelated to sensor noise properties. In order to obtain accurate estimations, including accelerometer bias, the method requires 15 seconds for initialization. In [108] the accelerometer bias is assumed to be zero, requiring only 2 seconds to initialize, depending on the motion. In both methods, IMU measurements are manipulated and mixed in the same linear system, where the residuals of all equations are considered with the same weight, ignoring sensor uncertainties. In addition, the different inertial parameters are solved separately in different steps, not all at once, ignoring the correlations between them. All this leads to an estimation which is not optimal in the sense of Maximum-a-Posteriori (MAP) estimation.

Sometimes, the initialization is also carried out together with the extrinsic calibration, which consists of finding camera-IMU relative transformation, necessary for fussing visual and inertial measurements. In this sense, Yang and Shen [149] formulated an online initialization and calibration, not making use of any pattern, showing a time of convergence of more than 30 seconds, considerably greater than only initialization methods. Similarly Dong-Si and Mourikis [39] propose an efficient least square formulation which also estimates the extrinsic calibration, works with a low number of images and features, but makes the strong assumption that biases are previously known and results are prone to errors for short initialization times.

In sections 3.2 and 3.3 we respectively present a joint and a disjoint visual-inertial initialization.

3.2 Joint Visual-Inertial initialization

In this section, we build on the previously mentioned *Martinelli-Kaiser solution* [90] [69] (or simply *MK-solution*), modifying it to be more general, efficient, robust and precise. This new formulation address the main existing issues from the original work, which we identify as:

- It assumes that all features are tracked in all frames used for initializing, and that all tracks provided are correct. In case of spurious tracks, it can provide arbitrarily bad solutions.

- The initialization accuracy is low, compared to [98]. To improve it, a lot of tracks and frames are needed, increasing its computational cost, and making it unfeasible in real time.

- With noisy sensors, trajectories that are close to the unsolvable cases analyzed by Martinelli [90] give weak observability of some of the variables. The method lacks robust criteria to decide when an initialization is accurate enough.

The main novelties of our proposed initialization algorithm are:

1. **Generality**: we generalize the method to use partial tracks and to take into account the camera-IMU relative pose.

2. **Efficiency**: we reduce the running time by using a fixed number of m features and n keyframes carefully chosen, and adopting the preintegration method proposed in [51, 87].

3. **Observability test**: after *MK-solution*, we perform visual-inertial BA with the m points and n keyframes, and apply a novel observability test to check the accuracy of the solution. If the test fails, the initialization is discarded.

4. **Consensus test**: we try to initialize additional tracked features by triangulating the 3D point and checking the reprojection error. If enough consensus is found, we perform a second visual-inertial BA with all points and keyframes, and initialize the SLAM map with them. Otherwise, the initialization is discarded.

We present exhaustive initialization tests on the EuRoC dataset [16] showing that the method is able to consistently initialize in less than 2s with a scale error of 5%. This error converges to 1% performing visual-inertial BA after 10s, when the trajectory is long enough to give higher observability of all variables.

3.2.1 Initial Solution

3.2.1.1 Feature extraction and tracking

We use the multiscale ORB extractor [117] to find M uniformly distributed points to be tracked. As points with high resolution are preferable, we only extract ORB points at the three finest image scales, using a scale factor of 1.2. We have found that the extraction of FAST features and matching with ORB descriptors is not stable enough to obtain long tracks, as most of the features were lost in some of the frames. Hence, to solve this problem, we have performed feature tracking using the pyramidal implementation of the well known Lucas-Kanade algorithm [86].

However, this tracking is not perfect and could lead to some tracks which are not correct, particularly in areas of repetitive or low texture or apparent contour, finally corrupting our solution. To attack this problem, the geometric consistency of these tracks is checked by finding a fundamental matrix using a RANSAC algorithm. The combination of FAST keypoints, Lucas-Kanade tracking (KLT), and the fundamental matrix check, leads to long tracks for a high number of features, as shown in figure 3.1. When some track is lost or rejected by the RANSAC algorithm, new FAST points are extracted and start to be tracked, in order to keep a constant number of M tracks. Within this set of M points, each time we detect that there are at least m tracked points with a track length of at least l pixels, we launch our initialization method. This first test to decide whether to attempt initialization or not is called *track-length test.*

3.2.1.2 Modified Martinelli-Kaiser solution

In this part we extend the method proposed by Martinelli et al. [69, 90] to be able to deal with features not seen by all cameras, and computing terms in an efficient way using inertial pre-integration proposed by Lupton and Sukkarieh [87] and latter formulated in manifolds by Forster et al. [51]. Let m be the number of tracked features used for initialization, placed at $(\mathbf{x}^1 \dots \mathbf{x}^m)$ in the global reference frame, and $\mathcal{C} = \{C_1 \dots C_n\}$ the set of n cameras indexes used for initialization, also referred to as keyframes, which are chosen to be uniformly distributed along time. Let's also call $\mathcal{C}^i = \{C_{1_i} \dots C_{n_i}\}$ the set of n_i cameras indexes seeing feature i-th, where C_{1_i} stands for the first camera seeing this point. Here we remark that, in contrast to [90], in our formulation not all features have to be observed by all cameras, and the transformation from camera to body (IMU), obtained from calibration, is taken into account. Without loss of generality we use as global reference the first body pose used for initialization.

From figure 3.2, we can write the set of equalities:

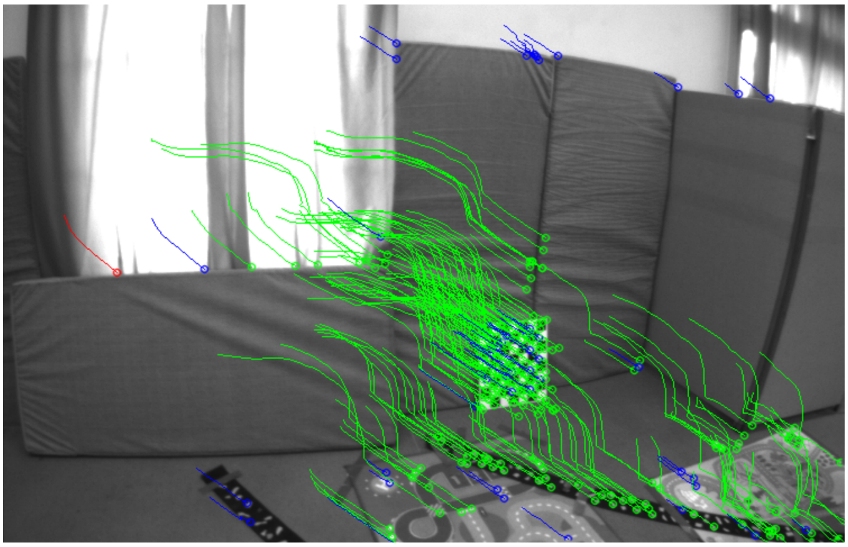


Figure 3.1: Example of obtained tracks. Green for tracks which have been checked by mean of fundamental matrix; Red for tracks which are inconsistent with computed fundamental matrix; Blue for tracks which are too short and have not been checked yet.

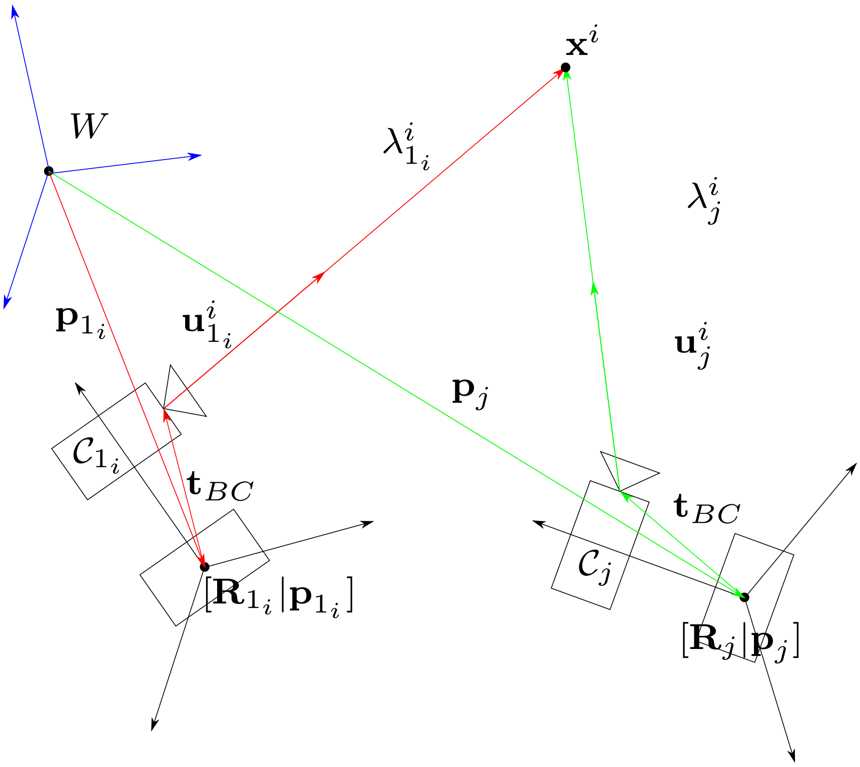


Figure 3.2: Relationships between two Body (IMU) and Camera poses observing the same feature.

$$\begin{aligned}\mathbf{p}_{1_i} + \mathbf{R}_{1_i}\mathbf{t}_{BC} + \lambda^i_{1_i}\mathbf{u}^i_{1_i} &= \mathbf{p}_j + \mathbf{R}_j\mathbf{t}_{BC} + \lambda^i_j\mathbf{u}^i_j \\ i &= 1\dots m, \quad \forall j \in \mathcal{C}^i \setminus 1_i\end{aligned} \tag{3.1}$$

where:

- $\mathbf{p}_j \in \mathbb{R}^3$: position of the j-th body in the global reference frame.
- $\mathbf{R}_j \in \mathrm{SO}(3)$: rotation matrix from j-th body to global reference frame.
- λ^i_j: depth of i-th feature at j-th camera reference.
- $\mathbf{u}^i_j$: unitary vector from j-th camera to i-th feature in the global reference frame. It can be computed as $\mathbf{u}^i_j = \mathbf{R}_j\mathbf{R}_{BC}\, {}_{c_j}\mathbf{u}^i_j$, being ${}_{c_j}\mathbf{u}^i_j$ the unitary vector in the j-th camera reference frame for the i-th feature, which is directly obtained from the tracked images.
- $\mathbf{T}_{BC} \equiv [\mathbf{R}_{BC}|\mathbf{t}_{BC}]$: transformation from Camera to Body (IMU).

$\mathbf{R}_j$ and $\mathbf{p}_j$ can be computed by integrating the inertial measurements. Considering constant biases during the initialization time and repetitively taking equations 2.12, it leads to [51]:

$$\mathbf{R}_{1,j}(\mathbf{b}^g) = \prod_{k=1}^{t_j-1} \mathrm{Exp}((\tilde{\boldsymbol{\omega}}_k - \mathbf{b}^g)\Delta t) \tag{3.2}$$

$$\mathbf{v}_j(\mathbf{b}) = \mathbf{v}_1 + \mathbf{g}\Delta t_{1,j} + \sum_{k=1}^{t_j-1} \mathbf{R}_{1,k}(\tilde{\mathbf{a}}_k - \mathbf{b}^a)\Delta t \tag{3.3}$$

$$\mathbf{p}_j(\mathbf{b}) = \mathbf{p}_1 + \sum_{k=1}^{t_j-1} \left(\mathbf{v}_k\Delta t + \frac{1}{2}\mathbf{g}\Delta t^2 + \frac{1}{2}\mathbf{R}_{1,k}(\tilde{\mathbf{a}}_k - \mathbf{b}^a)\Delta t^2\right) \tag{3.4}$$

where:

- Exp stands for the exponential map Exp : $\mathbb{R}^3 \rightarrow \mathrm{SO}(3)$, as stated in appendix section D.2.3.
- $\tilde{\mathbf{a}}_k$ and $\tilde{\boldsymbol{\omega}}_k$ are k-th acceleration and angular velocity IMU measurement
- $\mathbf{b} = (\mathbf{b}^a, \mathbf{b}^g)$ are their corresponding accelerometer and gyro biases
- $\mathbf{g}$ stands for gravity in the global frame
- $\mathbf{v}_j$ is the velocity at the j-th body.
- $\Delta t_{i,j}$ denotes time difference between i-th and j-th poses.

As stated in section 2.2.1 and following previous work [51, 87], these expressions may be split into IMU measurement and bias dependent and non-dependent components, using delta terms $\Delta\mathbf{R}_{1,j}$, $\Delta\mathbf{v}_{1,j}$, $\Delta\mathbf{p}_{1,j}$, which leads to equation 2.13 from section 2.2.1. Each time biases are modified, we can simply update delta terms, which can be made in a very efficient way using a linear approximation.

If during intermediate steps $\mathbf{b}^g$ changes more than 0.2 rad/sec from the value used for preintegration, the preintegration is recomputed with this new bias, otherwise, delta terms are directly updated using their Jacobians w.r.t. biases $\left(\frac{\partial \Delta \mathbf{R}_{1,j}}{\partial \mathbf{b}^g}, \frac{\partial \Delta \mathbf{v}_{1,j}}{\partial \mathbf{b}^g}, \frac{\partial \Delta \mathbf{v}_{1,j}}{\partial \mathbf{b}^a}, \frac{\partial \Delta \mathbf{p}_{1,j}}{\partial \mathbf{b}^g}, \frac{\partial \Delta \mathbf{p}_{1,j}}{\partial \mathbf{b}^a}\right)$, which can be found in [51]. In this way, we relinearize each time we get too far from the linearization point. We rewrite equation (3.1) using preintegrated terms:

$$\begin{aligned} \lambda_{1_i}^i \mathbf{u}_{1_i}^i - \lambda_j^i \mathbf{u}_j^i - \mathbf{v}_1 \Delta t_{1_i,j} - \mathbf{g}\left(\frac{\Delta t_{1,j}^2 - \Delta t_{1,1_i}^2}{2}\right) &= \\ \Delta \mathbf{p}_{1,j} - \Delta \mathbf{p}_{1,1_i} + (\Delta \mathbf{R}_{1,j} - \Delta \mathbf{R}_{1,1_i})\mathbf{t}_{BC}& \\ \forall i = 1 \dots m, \quad \forall j \in \mathcal{C}^i \setminus 1_i& \end{aligned} \tag{3.5}$$

Now, we can add the gravity magnitude information. Instead of adding it as a constraint of the gravity magnitude by mean of a Lagrange Multiplier formulation or as done in [69], we prefer to model the gravity by mean of a rotation matrix parametrized by only two angles (α, β) (rotation around z-axis has no effect) and the constant vector $\mathbf{g}_I = (0, 0, -g)$, thus we remain in an unconstrained problem:

$$\mathbf{g} = \mathrm{Exp}(\alpha, \beta, 0)\mathbf{g}_I \tag{3.6}$$

Equation (3.5) becomes:

$$\lambda_{1_i}^i \mathbf{u}_{1_i}^i - \lambda_j^i \mathbf{u}_j^i - \mathbf{v}_1 \Delta t_{1_i,j} = \mathbf{s}_{1_i,j}(\mathbf{b}^g, \mathbf{b}^a, \alpha, \beta) \tag{3.7}$$

where:

$$\begin{aligned} \mathbf{s}_{1_i,j}(\mathbf{b}^g, \mathbf{b}^a, \alpha, \beta) = \mathrm{Exp}(\alpha, \beta, 0)\mathbf{g}_I \left(\frac{\Delta t_{1,j}^2 - \Delta t_{1,1_i}^2}{2}\right) +& \\ \Delta \mathbf{p}_{1,j} - \Delta \mathbf{p}_{1,1_i} + (\Delta \mathbf{R}_{1,j} - \Delta \mathbf{R}_{1,1_i})\mathbf{t}_{BC}& \end{aligned} \tag{3.8}$$

Neglecting accelerometer bias as in [69], the only unknowns are λ_j^i ($\forall i = 1 \dots m$, $j \in \mathcal{C}^i$), $\mathbf{v}_1$, α, β and $\mathbf{b}^g$. Stacking equations for all possible values of i and j we build an overdetermined sparse linear system, with only three non-zero elements per row, such as:

$$\mathbf{A}(\mathbf{b}^g)\mathbf{x} = \mathbf{s}(\mathbf{b}^g, \alpha, \beta) \tag{3.9}$$

where $\mathbf{x} = (\mathbf{v}_1, \{\lambda_j^i\})$ is the unknown vector with linear dependence. To jointly find linear and no-linear dependent parameters, we solve the next unconstrained minimization problem:

$$(\mathbf{b}^g, \alpha, \beta) \triangleq \underset{\mathbf{b}^g, \alpha, \beta}{\arg\min} \left(\min_{\mathbf{x}} \|\mathbf{A}(\mathbf{b}^g)\mathbf{x} - \mathbf{s}(\mathbf{b}^g, \alpha, \beta)\|_2^2\right) \tag{3.10}$$

Cost function $c(\mathbf{b}^g, \alpha, \beta) = \min_{\mathbf{x}} \|\mathbf{A}(\mathbf{b}^g)\mathbf{x} - \mathbf{s}(\mathbf{b}^g, \alpha, \beta)\|_2^2$ is evaluated for each $(\mathbf{b}^g, \alpha, \beta)$ using the following scheme:

1. **Update $\Delta \mathbf{R}_{1,j}$ and $\Delta \mathbf{p}_{1,j}$**: Using [51], we don't need to reintegrate all IMU measurements each time that $\mathbf{b}^g$ changes. We simply update delta terms using their Jacobians w.r.t. bias. That supposes an important computational saving.

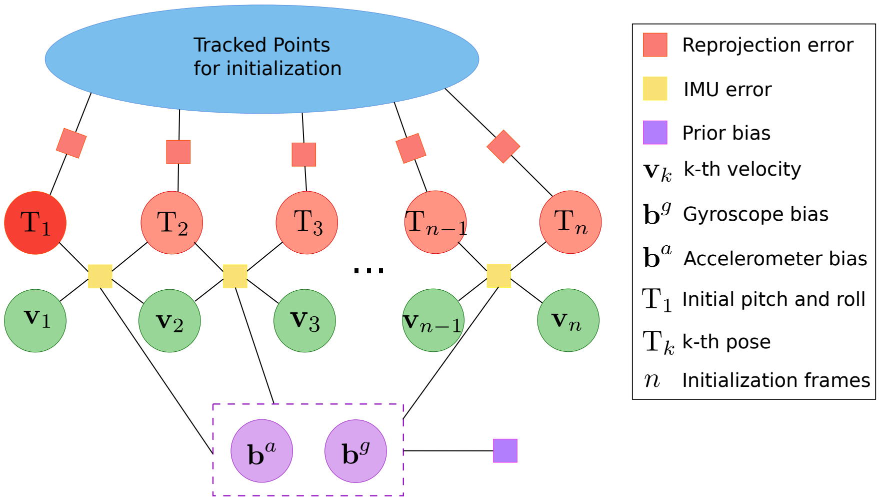


Figure 3.3: Graph for the first visual-inertial BA. The body poses and points included in the optimization are the same used in the initial solution.

2. **Compute $\mathbf{A}(\mathbf{b}^g)$ and $\mathbf{s}(\mathbf{b}^g, \alpha, \beta)$** and build the linear system.

3. **Solve $\mathbf{x} = \mathbf{A}(\mathbf{b}^g) \backslash \mathbf{s}(\mathbf{b}^g, \alpha, \beta)$** using conjugate gradient, which is very efficient for sparse systems like this.

4. **Compute $c(\mathbf{b}^g, \alpha, \beta) = \|\mathbf{A}(\mathbf{b}^g)\mathbf{x} - \mathbf{s}(\mathbf{b}^g, \alpha, \beta)\|_2^2$.**

The computational cost of evaluating $c(\mathbf{b}^g, \alpha, \beta)$ comes, first, from solving a sparse system with no more than $3 + n \times m$ unknowns and $3 \times (n-1) \times m$ equations and, second, from integrating inertial measurement along the initialization time, which is only computed once.

To optimize $c(\mathbf{b}^g, \alpha, \beta)$ and find the correct gyro bias and gravity direction we use Levenberg–Marquardt algorithm, where Jacobians of the cost function are computed numerically. As result, not only IMU initialization parameters are found ($\mathbf{g}$, $\mathbf{b}^g$ and $\mathbf{v}_1$) but also the position of tracked points ($\lambda_j^i \mathbf{u}_j^i$). We highlight that not all M tracked features have been used during this initialization, but only a small set of m features, aiming to reduce computational complexity. However, the solutions found after this step may not be accurate enough to launch the system, and further intermediate stages are required, as explained in next sections.

3.2.2 Improved Solution

3.2.2.1 First BA and observability test

After finding the initial parameters ($\mathbf{g}$, $\mathbf{b}^g$, $\mathbf{v}_1$, $\{\lambda_j^i\}$) we build a visual-inertial BA problem with the same n body poses and m points from the previous step (see figure 3.3). This optimization contains reprojection errors and IMU residuals leading to a similar problem to equation 2.17. We set the z axis in the estimated gravity direction. All body poses have six optimizable variables $(\phi, \mathbf{t}) \in \mathfrak{se}(3)$ except the first one, which has only two (pitch and roll) since translation and yaw have been fixed in order to remove the four gauge freedoms inherent to the visual inertial problem. Body velocities are also included in the optimization, and they evolve according to the inertial measurements. Initial estimations

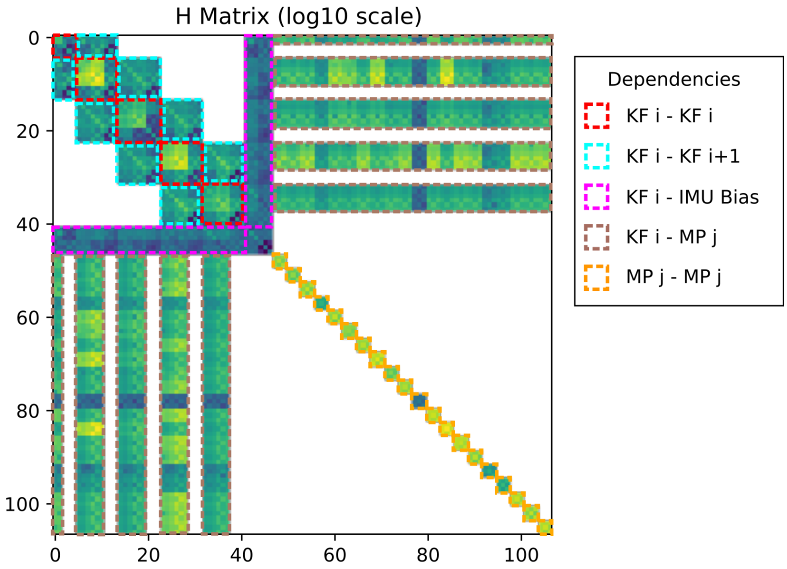


Figure 3.4: Example of Hessian matrix for an initial map with 5 keyframes (KF) and 20 map points (MP). One can distinguish different blocks, outlined with dashed lines. In the top-left part, we have the diagonal blocks of each keyframe (red), blocks relating consecutive keyframes, due to the IMU measurements (blue), and blocks relating keyframes and IMU biases (pink). In the bottom-right part, there are only the diagonal blocks of the map points (orange). Out-of-diagonal terms relate map points with the keyframes that observe them (brown). In this example all cameras observe all features.

for each vertex are added using results from the *MK-solution*. In addition, accelerometer bias $\mathbf{b}^a$ is included in this optimization, but similarly to $\mathbf{b}^g$ it is assumed to be constant for all frames. Previous $\mathbf{b}^g$ estimation from *MK-solution* step is included by means of a prior, as well as $\mathbf{b}^a$ is forced to be close to zero. We call this optimization *first BA* or simply *BA1*. Analytic expression for Jacobians, found in [51], are used for IMU residuals, while Jacobians for the reprojection error have been derived analytically, taking into account that we are optimizing body pose and not camera pose (See appendix B.2).

Usually this optimization provides a better initialization solution. However, if the motion performed gives low observability of the IMU variables, the optimization can converge to arbitrarily bad solutions. For example this happens in case of pure rotational motion or non-accelerated motions [90]. In order to detect these failure cases we propose an *observability test*, where we analyze the uncertainty associated to estimated variables. This could be done by analyzing the covariance matrix of the estimated variables and checking if its singular values are small enough. However, this would require to invert the information matrix, i.e. the Hessian matrix from *first BA*, which is a square matrix of size $(3m + 6 + 9n - 4)$, being computationally too expensive. Instead, we perform the observability test imposing a minimal threshold to all singular values of the Hessian matrix associated to our *first BA*. The Hessian can be built from the Jacobian matrices associated to each edge in the graph from figure 3.3, as explained next.

Suppose we have $\{1 \dots p\}$ states, and denote $\{\mathbf{e}_1 \dots \mathbf{e}_q\}$ the set of q measurements which appear in the *first BA*. Let's call $\mathcal{E}_i$ the set of measurements where state i is involved. The Hessian block matrix for states i and j, taking a first order approximation (see appendix A.1), can be built as follows:

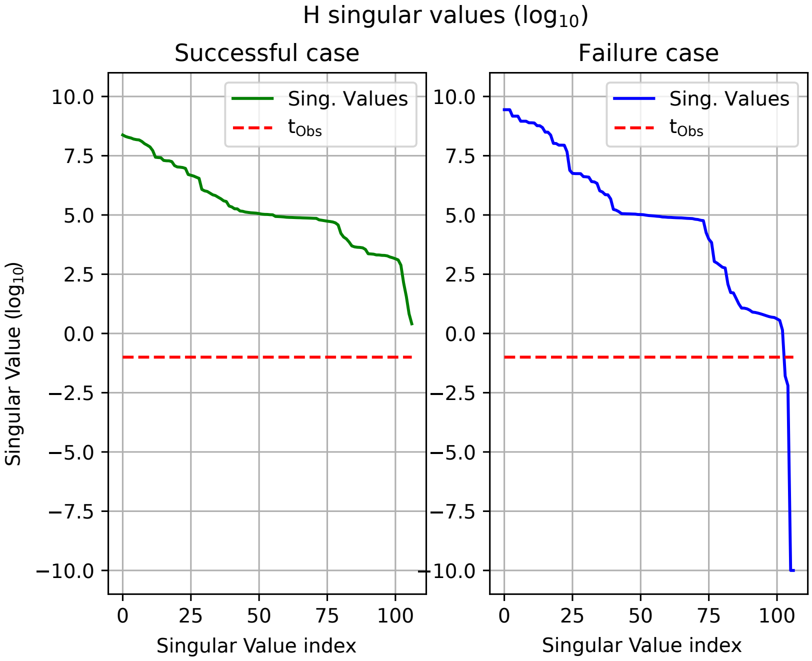


Figure 3.5: Singular values of the information matrix for a successful initialization and a failure case on the EuRoC V103 sequence. The successful case has a RMSE ATE error of 3.16 % in the initialization trajectory, and corresponds to a translation and rotation motion. The failure case has an error of 64.99 % and corresponds to an almost pure rotational motion. We draw the observability threshold used $t_{obs} = 0.1$

$$\mathbf{H}_{i,j} \approx \sum_{\mathbf{e}\in\mathcal{E}_i\cap\mathcal{E}_j} \mathbf{J}_{i,\mathbf{e}}^T \Omega_{\mathbf{e}} \mathbf{J}_{j,\mathbf{e}} \tag{3.11}$$

where $\Omega_{\mathbf{e}}$ stands for the information matrix of the $\mathbf{e}$ measurement, and $\mathbf{J}_{i,\mathbf{e}}$ for the Jacobian of the $\mathbf{e}$ measurement w.r.t. i-th state. In order to have a non-zero (i, j) block matrix, there must to be an edge between i and j node in the graph (measurement depending on both variables) as shown in figure 3.4.

Applying the SVD decomposition to $\mathbf{H}$ and looking at the smallest singular value one can determine if the performed motion guarantees observability of all the IMU variables. Hence, we discard all initializations where the smallest Hessian singular value falls below a threshold denoted by t_{obs}. If this *observability test* is not passed, we discard the initialization attempt. Examples of a successful and a rejected case are shown in figure 3.5.

3.2.2.2 Consensus test and second BA

As we have noted before, not all M tracked features have been used in *MK-solution* and *first BA* steps, but only m features. To take advantage of these extra unused tracked points, we propose to perform a *consensus test* in order to detect initializations which have been performed using spurious data, such as bad tracked features.

First, the 3D point position of each unused track is triangulated between the two most distant frames which saw the point, minimizing the reprojection error [131]. Only tracks with parallax greater than 0.01 radians are used. Then we re-project each 3D point into all the frames which observe it, compute the residual re-projection error, and perform a

Table 3.1: Parameters of our initialization algorithm

Total number of tracks	M	200
Track-length test (in pixels)	l	200
Tracks used for *MK-solution*	m	20
Keyframes used for *MK-solution*	n	5
Observability test: Singular value threshold	t_{obs}	0.1
Consensus test: Inlier threshold	t_{cons}	90%

$\chi^2(95\%)$ test with $2n_i - 3$ degrees of freedom, where n_i is the number of frames which observe this point. The *consensus test* is performed counting the percentage of inliers: if it is bigger than a threshold t_{cons} we consider that the proposed solution is accurate, if not, we discard the initialization attempt.

If the *consensus test* is successful, we perform a *second BA* (or simply *BA2*) including the m points used in the initial solution plus all the points which have been triangulated and detected as inliers, having a total of M' points. The graph for this optimization is similarly built than in case of *BA1* but with more points.

3.2.2.3 Map initialization

After this second BA, the keyframe poses are accurate enough, but we only have a few points to initialize the map. Before launching the whole visual-inertial SLAM system, we triangulate new points aiming to densify the point cloud and to ease the posterior tracking operation. Since we already have the keyframe poses, we extract ORB features in each keyframe and perform an epipolar search in each other, using the ORB descriptor. All these new points, together with the M' points from *BA2*, are promoted to map points, and the n frames used for initialization are promoted to map keyframes. The covisiblity graph [99] of this new map is also created, taking into account the observations of points.

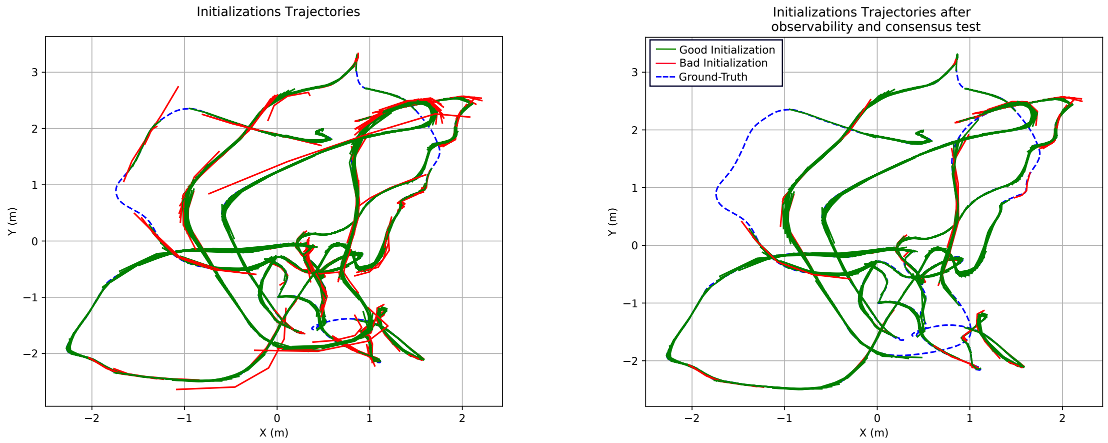
3.2.3 Experiments

The most important parameters of our method are shown in table 3.1. Our implementation uses ORB-SLAM visual-inertial [98] with its three threads for tracking, mapping and loop closing. Initialization is performed in a parallel thread, thus it has no effect in the real time tracking thread. For *MK-solution* we use Eigen C++ library, while for graph optimization of *BA1* and *BA2* we use g2o C++ library [77]. Experiments have been run in V1 dataset from EuRoC [16] using a Intel Core i7-7700 computer with 32 GB of memory.

3.2.3.1 Results

EuRoC dataset provides stereo images and synchronized IMU measures for three different indoor environments, with different complexity. We have tested our method for environment V1 from EuRoC at three difficulty levels. We run two different experiments.

In a first experiment, we try to initialize as often as possible in real time. Along the whole trajectory, every time the tracking thread has m tracks with length l, if the initialization thread is idle, a new initialization attempt is launched. We measure the RMS ATE and its scale error with respect to the ground-truth. To measure scale factor, and thus scale error, we align the initialization and ground-truth trajectories using a


Initializations Trajectories

Initializations Trajectories after observability and consensus test

Figure 3.6: Experiment on EuRoC V101 dataset. In blue, ground truth trajectory; in green, estimated initialization trajectories whose RMSE ATE error is lower than 5%; in red, those with a bigger error. On the **left**, found trajectories without applying any test. On the **right**, trajectories which passed *observability and consensus tests*. Our method was able to find 511 correct initializations along the whole trajectory, running in real time.

7DoF Horn alignment, such that for an instant t, the estimated $\hat{\mathbf{p}}(t)$ and ground-truth $\mathbf{p}_{\mathrm{GT}}(t)$ trajectories are related by:

$$\hat{\mathbf{p}}(t) = \mathbf{T} \oplus \mathbf{p}_{\mathrm{GT}}(t) \quad \text{where} \quad \mathbf{T} \in \mathrm{Sim}(3) \tag{3.12}$$

We proceed in a similar way for ATE, but using a SE(3) alignment.

Figure 3.6 shows initializations found for trajectory V101 before and after the *observability and consensus test*. We show in red trajectories which have a RMSE ATE [128] error bigger than 5% of the initialization trajectory length. We can see in that figure that our initialization algorithm successfully estimates the trajectory along almost all the sequence, and both proposed test efficiently reject worst found solutions.

In table 3.2 we show the main numerical results of these experiments with the three V1 sequences. RMSE ATE [128] is expressed in percentage over the length of the initialization trajectory. Below each sequence name we show successful initializations over the total number. First thing to notice is the large number of initialization attempts. For example, in sequence V101 which lasts 130 seconds, up to 728 initializations are computed, and 511 of them have passed the *observability and consensus test*. The table shows that the original Martinelli-Kaiser solution obtains average scale errors between 32.9% and 156.7% on these sequences. This error can be reduced until 8.8% to 24.5% applying the two rounds of visual-inertial BA proposed here. More interestingly, applying the novel *observability and consensus tests*, inaccurate initializations are consistently rejected, and the average scale error is reduced to around 5% for all sequences, a very significant improvement over the original method. The ATE error is also drastically reduced after both tests. Considering the initialization time we see an evident difference between V101, that requires initialization trajectories of 2.2 seconds in average, and V102 and V103 where 1 second is enough. In these two last sequences motion is faster and the *track-length test* is satisfied in less time than in the first sequence, where the quadrotor is flying at low speed.

Regarding the computational cost, the average CPU required to solve the initialization is less than 85ms for sequences V102 and V103, and around 121ms for V101, due to the longer preintegration period. In all cases, the *MK-solution* step takes around 75% of the total initialization CPU time.

Table 3.2: Results of exhaustive initialization tests over the three V1 EuRoC sequences.

V1 EuRoC Dataset

		After *track-length test*		After *Observ + Cons. test*			
Seq. Name		RMSE ATE (%)	Scale error (%)	RMSE ATE (%)	Scale error (%)	CPU time (ms)	Trajectory time (s)
V101 (511/728)	*MK-solution*	9.176	32.998	7.749	25.104	95.082	2.235
	MK-solution + BA1	3.977	10.719	2.352	6.471	104.114	2.235
	MK-solution + BA1&2	3.270	8.816	**2.036**	**5.496**	120.983	2.235
V102 (101/395)	*MK-solution*	12.025	156.751	6.760	48.926	60.285	0.968
	MK-solution + BA1	6.338	25.252	2.541	7.195	70.963	0.968
	MK-solution + BA1&2	5.149	20.341	**1.935**	**5.497**	84.443	0.968
V103 (71/336)	*MK-solution*	47.928	128.008	6.634	21.691	62.160	1.070
	MK-solution + BA1	71.774	28.160	2.475	6.836	73.301	1.070
	MK-solution + BA1&2	71.068	24.556	**1.870**	**5.259**	84.676	1.070

In table 3.3 we show computational times for our method which uses preintegration with first order bias correction from [51]. Compared with using the original formulation from Martinelli and Kaiser, computing time is reduced by 60%.

In a second experiment, we launch visual-inertial ORB-SLAM [98] and we retrieve the RMSE ATE and the scale error just after the proposed initialization, and after performing full visual-inertial BA at 5 seconds and 10 seconds from the first keyframe timestamp. We can see in table 3.4 that all three sequences converge to scale error smaller than 1% after 10 seconds, confirming that our initialization method is accurate enough to launch visual-inertial SLAM. Compared with the initialization method proposed in [98], our method requires trajectories of 1 or 2 seconds instead of 15 seconds, uses less CPU time, and is able to successfully initialize in sequence V103, where the previous method failed.

Table 3.3: Comparison of running time for *MK Solution+BA1+BA2* repeating IMU integration in each iteration and using preintegration with first order bias correction [51].

V101 EuRoC Dataset

	CPU time (ms)		
	Mean	Std	Max
Reintegrating each time	301.302	91.974	678.886
Using first order correction	120.983	27.609	214.989

Table 3.4: Results of VI-SLAM using our initialization (average errors on five executions are shown).

V1 EuRoC Dataset

Seq. Name	After initialization		After BA 5s		After BA 10s	
	RMSE ATE (m)	Scale error (%)	RMSE ATE (m)	Scale error (%)	RMSE ATE (m)	Scale error (%)
V1_01_easy	0.0183	4.99	0.0200	1.85	**0.0170**	**0.84**
V1_02_medium	0.0364	7.38	0.0076	3.67	**0.0162**	**0.71**
V1_03_difficult	0.0043	4.80	0.0129	2.50	**0.0120**	**0.27**

3.3 Inertial-only optimization

Our work from previous section 3.2 on IMU initialization renders the original method from Kaiser et al. usable, obtaining on the public EuRoC dataset [16] successful visual-inertial initializations in 2 seconds with scale error around 5%. However, for some studied sequences, this method can barely work in 20% of the trajectory. Such a low initialization recall can be a problem for AR/VR or drone applications where the system is desired to be launched immediately. For this reason, in this section, we explore the use of disjoint or loosely coupled visual inertial initialization methods.

Thus, we propose a novel algorithm by formulating the initialization as an optimal estimation problem, in the sense of Maximum-a-Posteriori (MAP) estimation. For this, we build on the excellent work of Forster et al. [51] that allows to preintegrate IMU readings and, taking into account the probabilistic characterization of sensor noises and properly compute the covariances of the preintegrated terms. Assuming that the error of the monocular SLAM trajectory is negligible compared with the IMU errors, we derive a very efficient MAP estimator for inertial-only parameters, and use it to initialize a visual-inertial SLAM system. The main contributions of our work are:

- The formulation of the visual-inertial initialization as an inertial-only optimal estimation problem, in the sense of MAP estimation, taking properly into account the probabilistic model of IMU noises.

- We solve for all inertial parameters at once, in a single step, avoiding the inconsistencies derived from decoupled estimation. This makes all estimations jointly consistent.

- We do not make any assumptions about initial velocity or attitude, being the method suitable for any initialization case.

- We do not assume any IMU parameter to be zero, instead we code the known information about them as probabilistic priors that are exploited by our MAP estimation.

In the next sections we present the theory and in-depth details behind our proposal. Later, we evaluate and compare it against the best examples of joint and disjoint initialization methods, proving to outperform them.

3.3.1 Maximum-a-Posteriori Initialization

As it has been stated in section 2.2, the gold-standard method for feature-based visual-inertial SLAM is visual-inertial Bundle Adjustment (VI-BA), that takes properly into account the noise properties in all the sensors, and obtains a maximum-a-posteriori joint estimation of all variables. This estimation problem is equivalent to the optimization solved at equation 2.17. The main limitation of VI-BA is that it requires a good initial seed to quickly converge and avoid getting stuck in local minima, due to its strong non-linear nature. Our previously presented initialization, and the disjoint from Mur and Tardos [98] based on least-squares optimization showed that VI-BA largely improves the initial solutions.

Here, our main goal is going a step further and use also MAP estimation in the initialization, making proper use of sensor noise models. Our novel initialization method is based on the following ideas:

- Despite the non-linear nature of BA, Monocular SLAM (or visual odometry) is mature and robust enough to obtain very accurate initial solutions for structure and motion, with the only caveat that their estimations are up-to-scale.

- The uncertainty of visual SLAM trajectory is much smaller than the IMU uncertainties and can be ignored while obtaining a first solution for the IMU variables. So, we perform inertial-only MAP estimation, taking the up-to-scale visual SLAM trajectory as constant.

- Inspired on the work of [126], we adopt a parametrization that explicitly represents and optimizes the scale factor of the monocular SLAM solution.

- Differently from [98] [108], we jointly optimize all the IMU variables in one step, taking into account the cross-covariances between the preintegrated terms for position, and linear and angular velocities [51].

Our initialization method can be split in three steps:

1. **Vision-only MAP estimation**: Initialize and run monocular ORB-SLAM [99] for a short period (typically 2 s) using BA to obtain a vision-only MAP estimation up-to-scale. At the same time, compute IMU preintegrations between keyframes and their covariances [51].

2. **Inertial-only MAP estimation**: Inertial-only optimization to align the IMU trajectory and ORB-SLAM trajectory, finding the scale, keyframes' velocities, gravity direction and IMU biases.

3. **Visual-inertial MAP estimation**: Use the solution from the previous step as seed for a full VI-BA to obtain the joint optimal solution.

After the initialization, we launch ORB-SLAM Visual-Inertial [98], that performs local VI-BA. We have observed that scale estimation accuracy can be further improved after 5-10 seconds performing a full VI-BA or, with much lower computational cost, repeating the inertial-only optimization.

The three initialization steps are further detailed next.

3.3.1.1 Vision-only MAP Estimation

We initialize pure monocular SLAM, using the same procedure as in ORB-SLAM to find the initial motion. Matching of FAST points, using ORB descriptor, is performed between two initial frames. Fundamental matrix and homography models are found and scored [99]. That one with a higher score is used to find the initial motion and triangulate the features. Once the structure and motion are initialized, we do pure monocular SLAM for 2 seconds. The only difference from ORB-SLAM is that we enforce keyframe insertion at a constant higher frequency (4Hz to 10Hz). In that way, IMU preintegration between keyframes has low uncertainty, since integration times are very short. After this period, we have an up-to-scale map composed of ten keyframes and hundreds of points, that has been optimized using BA by the ORB-SLAM mapping thread.

The up-to-scale keyframe poses are transformed to the body (or IMU) reference using visual-inertial calibration. These body poses are denoted as $\bar{\mathbf{T}}_{0:k} = [\mathbf{R}, \bar{\mathbf{p}}]_{0:k}$, where $\mathbf{R}_i \in \mathrm{SO}(3)$ is rotation matrix from i-th body to world reference, and $\bar{\mathbf{p}}_\mathbf{i} \in \mathbb{R}^3$ is its up-to-scale position.

3.3.1.2 Inertial-only MAP Estimation

The goal of this step is to obtain an optimal estimation of the inertial parameters, in the sense of MAP estimation, using the up-to-scale trajectory obtained by vision. As we don't have a good guess of the inertial parameters, using at this point a full VI-BA would be too expensive and prone to get stuck in local minima. An intermediate solution would be to marginalize out the points to obtain a prior for the trajectory and its (fully dense) covariance matrix, and use it while optimizing the IMU parameters. We opt for a more efficient solution, considering the trajectory as fixed, and perform an inertial-only optimization. The inertial parameters to be found are:

$$\mathcal{X}_k = \{s, \mathbf{R}_{\mathrm{w}g}, \mathbf{b}, \bar{\mathbf{v}}_{0:k}\} \tag{3.13}$$

where $s \in \mathbb{R}^+$ is the scale factor of the vision-only solution, $\mathbf{R}_{\mathrm{w}g} \in \mathrm{SO}(3)$ is the gravity direction, parameterized by two angles, such that gravity in world reference frame is expressed as $\mathbf{g} = \mathbf{R}_{\mathrm{w}g}\mathbf{g}_\mathrm{I}$, with $\mathbf{g}_\mathrm{I} = (0, 0, G)^\mathrm{T}$ being G the magnitude of gravity, $\mathbf{b} = (\mathbf{b}^a, \mathbf{b}^g) \in \mathbb{R}^6$ are the accelerometer and gyroscope biases, and $\bar{\mathbf{v}}_{0:k} \in \mathbb{R}^3$ the up-to-scale body velocities from first to last keyframe.

We prefer to use up-to-scale velocities $\bar{\mathbf{v}}_i$, instead of true ones $\mathbf{v}_i = s\bar{\mathbf{v}}_i$, since it eases obtaining an initial guess. Biases are assumed constant for all involved keyframes as initialization period is just 1-2 seconds, and random walk would have almost no effect. It is worth noting that this formulation takes into account gravity magnitude from the beginning, as opposed to [108] and [98] that require a separate step to fix its value.

In our case, the only measurements used come from IMU, and are summarized in the IMU preintegrated terms defined in [51]. We denote by $\mathcal{I}_{i,j}$ the preintegration of inertial measurements between i-th and j-th keyframes, and by $\mathcal{I}_{0:k}$ the set of IMU preintegrations between successive keyframes in our initialization window.

With the state and measurements defined, we can formulate a MAP estimation problem, where the posterior distribution is:

$$p(\mathcal{X}_k|\mathcal{I}_{0:k}) \propto p(\mathcal{I}_{0:k}|\mathcal{X}_k)p(\mathcal{X}_k) \tag{3.14}$$

where $p(\mathcal{I}_{0:k}|\mathcal{X}_k)$ is the likelihood distribution of the IMU measurements given the IMU states, and $p(\mathcal{X}_k)$ the prior for the IMU states. Considering independence of measurements, the likelihood can be factorized as:

$$p(\mathcal{I}_{0:k}|\mathcal{X}_k) = \prod_{i=1}^{k} p(\mathcal{I}_{i-1,i}|s, \mathbf{g}_{dir}, \mathbf{b}, \mathbf{v}_{i-1}, \mathbf{v}_i) \tag{3.15}$$

To obtain the MAP estimator, we need to find the parameters which maximize the posterior distribution, that is equivalent to minimize its negative logarithm, thus:

$$\begin{aligned}\mathcal{X}_k^* = \arg\max_{\mathcal{X}_k} p(\mathcal{X}_k|\mathcal{I}_{0:k}) = \arg\min_{\mathcal{X}_k} \Bigg(&-\log(p(\mathcal{X}_k)) \\ &- \sum_{i=1}^{k} \log\left(p(\mathcal{I}_{i-1,i}|s, \mathbf{g}_{dir}, \mathbf{b}, \mathbf{v}_{i-1}, \mathbf{v}_i)\right)\Bigg)\end{aligned} \tag{3.16}$$

Assuming Gaussian error for IMU preintegration and prior distribution, the MAP problem is equivalent to:

$$\mathcal{X}_k^* = \arg\min_{\mathcal{X}_k} \left(\|\mathbf{r}_{\mathrm{p}}\|^2_{\Sigma_p} + \sum_{i=1}^{k} \|\mathbf{r}_{\mathcal{I}_{i-1,i}}\|^2_{\Sigma_{\mathcal{I}_{i-1,i}}} \right) \tag{3.17}$$

where $\mathbf{r}_{\mathrm{p}}$ and Σ_p are the residual of the prior and its covariance, while $\mathbf{r}_{\mathcal{I}_{i-1,i}}$ are the residuals of the integrated IMU measurements between consecutive keyframes, as defined in equation 2.15, and $\Sigma_{\mathcal{I}_{i-1,i}}$ its covariance.

In this optimization, vision reprojection errors do not appear, only inertial residuals. As IMU measurements do not suffer from data association errors, the use of robust cost function, like the Huber norm, does not make sense, since it would slow down the optimization.

These inertial residuals can be rewritten making then dependent on initialization state $\mathcal{X}_k = \{s, \mathbf{R}_{\mathrm{w}g}, \mathbf{b}, \bar{\mathbf{v}}_{0:k}\}$ as follows,

$$\mathbf{r}_{\mathcal{I}_{i,j}} = [\mathbf{r}_{\Delta \mathrm{R}_{ij}}, \mathbf{r}_{\Delta \mathrm{v}_{ij}}, \mathbf{r}_{\Delta \mathrm{p}_{ij}}] \tag{3.18}$$

$$\mathbf{r}_{\Delta \mathrm{R}_{ij}} = \mathrm{Log}\left(\Delta\mathbf{R}_{ij}(\mathbf{b}^g)^{\mathrm{T}}\mathbf{R}_i^{\mathrm{T}}\mathbf{R}_j\right)$$

$$\mathbf{r}_{\Delta \mathrm{v}_{ij}} = \mathbf{R}_i^{\mathrm{T}}\left(s\bar{\mathbf{v}}_j - s\bar{\mathbf{v}}_i - \mathbf{R}_{\mathrm{w}g}\mathbf{g}_{\mathrm{I}}\Delta t_{ij}\right) - \Delta\mathbf{v}_{ij}(\mathbf{b}^g, \mathbf{b}^a)$$

$$\mathbf{r}_{\Delta \mathrm{p}_{ij}} = \mathbf{R}_i^{\mathrm{T}}\left(s\bar{\mathbf{p}}_j - s\bar{\mathbf{p}}_i - s\bar{\mathbf{v}}_i\Delta t_{ij} - \frac{1}{2}\mathbf{R}_{\mathrm{w}g}\mathbf{g}_{\mathrm{I}}\Delta t_{ij}^2\right) - \Delta\mathbf{p}_{ij}(\mathbf{b}^g, \mathbf{b}^a)$$

Since we assume that biases can be considered constant during the initialization window, IMU residuals do not include random walk for biases. We assume that the residuals follow Gaussian distributions, and their covariances can be computed as proposed in [51].

As we are optimizing in a manifold for gravity direction, we need to define a retraction to update the its estimation during the optimization. In this sense, we update it according to:

$$\mathbf{R}_{\mathrm{wg}}^{\mathrm{new}} = \mathbf{R}_{\mathrm{wg}}^{\mathrm{old}}\mathrm{Exp}(\delta\alpha_{\mathbf{g}}, \delta\beta_{\mathbf{g}}, 0) \tag{3.19}$$

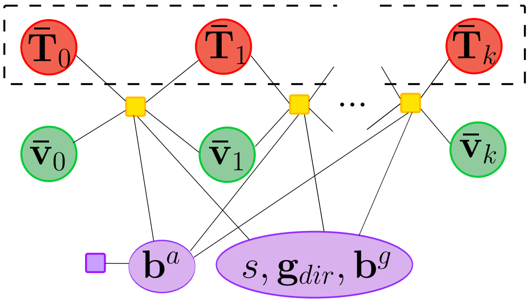


Figure 3.7: Underlying graph representation of the inertial-only optimization problem. Yellow boxes represent IMU residuals, while the purple one represents prior information for accelerometer bias. Dashed lines point out that keyframe poses are assumed to be perfectly known up-to-scale from visual SLAM.

where Exp is the SO(3) exponential map as explained in appendix section D.2.3.

In addition, to guarantee that scale factor remains positive during optimization we define its update as:

$$s^{\text{new}} = s^{\text{old}} \exp(\delta s) \tag{3.20}$$

Biases and velocities are updated additively. If we define $\delta \mathbf{g}_{dir} = (\delta\alpha_{\mathbf{g}}, \delta\beta_{\mathbf{g}})$, inertial parameters updates, used during optimization, are $(\delta s, \delta \mathbf{g}_{dir}, \delta \mathbf{b}^g, \delta \mathbf{b}^a, \{\delta \bar{\mathbf{v}}_i\})$. Derivatives for IMU residuals w.r.t. these parameters can be found in the appendix B.5.

The final optimization problem, represented in figure 3.7, is implemented and solved using g2o C++ library [77], using analytic derivatives and Levenberg-Marquardt algorithm.

As it is well known in the literature, gravity and accelerometer bias tend to be coupled, being difficult to distinguish both of them in most cases. To avoid, or at least mitigate, that problem, some techniques neglect accelerometer bias during the initialization assuming a zero value [108], while others wait for a long time to guarantee that it is observable [98]. Here we adopt a sound and pragmatic approach: we include $\mathbf{b}^a$ as a parameter to be optimized, but adding a prior residual for it: $\mathbf{r}_p = \|\mathbf{b}^a\|^2_{\Sigma_p}$. If the motion performed does not contain enough information to estimate the bias, the prior will keep its estimation close to zero. If the motion makes $\mathbf{b}^a$ observable, its estimation will converge towards its true value. A prior for $\mathbf{b}^g$ is not needed as it is always well observable from keyframe orientations and gyroscope readings.

Since we have to solve a non-linear optimization problem, we need an initial guess for inertial parameters. Hence, we initialize biases equal to zero, while gravity direction is initialized along the average of accelerometer measurements, as accelerations are usually much smaller than gravity.

The scale factor needs to be initialized sufficiently close to its true value to guarantee convergence, but we do not have any initial guess. Taking advantage of our very efficient inertial-only optimization (5ms), we launch the optimization with three initial scale values, that correspond to median scene depth of 1, 4 and 16 meters, keeping the solution that provides the lowest residual as defined in equation 3.17. Our results show that, using this range of scale values, our method is able to converge in a wide variety of scenes.

Once the inertial-only optimization is finished, the frame poses and velocities and the 3D map points are scaled with the value of s found, and rotated to align the z axis with the estimated gravity direction. Biases are updated and IMU preintegration is repeated

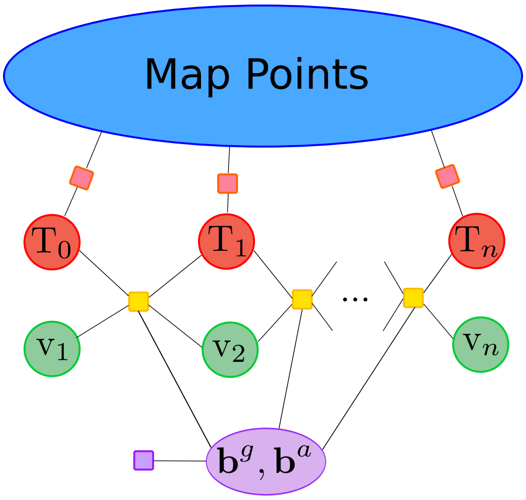


Figure 3.8: First visual-inertial Bundle Adjustment. Here, reprojection errors are also included, represented with red boxes, linking keyframes and map points.

with the new bias estimations, aiming to reduce future linearization errors.

3.3.1.3 Visual-Inertial MAP Estimation

Inertial-only optimization provides an accurate enough estimation to be used as a seed for a first joint visual-inertial Bundle Adjustment, ensuring its convergence. In this optimization, shown in figure 3.8, pure inertial parameters like $\mathbf{g}_{dir}$ and s do not appear, but they are implicitly included in keyframe poses. Compared with [19], this step replaces the BA1&2 steps. In fact, the optimization is exactly the same, it only differs in the initial seed, which previously was computed solving a linear system, and now is computed by means of a MAP estimator. A similar optimization is also done in VINS-Mono initialization, before launching VI odometry.

We remark that all initialization steps are performed in a parallel thread, without having any effect on the real time tracking thread. Once the optimization is finished, the system is already initialized, and we switch from visual to visual-inertial SLAM.

3.3.2 Experimental Results

To analyze the capability to initialize under different sensor trajectories, we run an exhaustive initialization test. We launch an initialization every 0.5 seconds (one out of 10 frames) in every trajectory of the EuRoC dataset, what results on testing 2248 different initialization trajectories. To compare, we run the same exhaustive test with our previous joint initialization method [19] and the loosely coupled initialization of VINS-Mono [108], using the software provided by the authors. As a baseline, we also try to initialize using only visual-inertial bundle adjustment with the same initial guesses for gravity direction, velocities and biases, and the same three initial values for scale our proposal uses, keeping the solution with smaller residual.

The performance is measured in terms of the scale error before and after applying full VI-BA. We also report the duration of the initialization trajectory, denoted as t_{Init}, as well as the total time, t_{Tot}, until a successful initialization is achieved. For all methods, if a bad initialization is detected, a new one is attempted with the next batch of available data. This, together with the time needed for visual initialization, makes $t_{Tot} \geq t_{Init}$.

Results are summarized in table 3.5. The first two blocks compare our method with our previous joint initialization method [19], based on the work of Martinelli [90] and Kaiser et al. [69], improved by VI-BA and two rejection tests. The proposed initialization beats the joint initialization by a wide margin, both in accuracy and needed time, being able to initialize in less than 4 seconds with scale error of 5.29%, using trajectories of 2.16 seconds on average. The method of [19] was able to obtain a scale error only slightly worse, but it was at the expense of a t_{Tot} of 13 seconds, owing to the high rejection rate of the proposed tests. The baseline VI-BA initialization, using the same trajectories and initial guesses, obtains an average scale error of 13.82%, which is even higher than the 11.69% error obtained by just applying our inertial-only optimization, which is also much more efficient (5 ms per run, compared with 133 ms, as shown in table 3.6).

The last blocks compare our method with the loosely-coupled initialization of VINS-Mono [108]. To ease comparison, we have configured our system to run with similar-sized trajectories (t_{Init}) of around 1.25 seconds. With these shorter trajectories, our method beats the baseline VI-BA initialization doubling its accuracy and more than doubling the accuracy of VINS-Mono initialization, with a t_{Tot} 0.31 seconds higher. This slightly higher t_{Tot} is the result of the visual initialization used in our system, that one from ORB-SLAM, which in difficult sequences can struggle to success. We remark that reducing the scale error by using longer initialization trajectories, i.e. increasing t_{Init}, may not be easy for VINS-Mono. Since this system is not prepared to work as a pure visual odometry system, visual and inertial initializations have to be solved simultaneously for the same set of frames, and increasing time for inertial initialization would also increase the visual initialization time. This entails that points have to be tracked along more frames, which may be not feasible in case of camera rotation or fast motion.

For VINS-Mono, there is sharp contrast between the 22.05% scale error found in our exhaustive initialization tests and the low RMS ATE error reported in [108] (in the range of 0.080-0.320 m) when the whole trajectories are processed. This may be explained because when launched from the beginning, the initialization is performed while the drone is taking off, which entails big accelerations, making inertial parameters more observable, while in our experiment, initialization is performed along the whole sequence where other motions that give lower observability are present. Moreover, since VINS-Mono marginalizes old states, not fixing them as our SLAM system does, this initial error can be further reduced as the drone progresses.

In figure 3.9, we plot the scale factor distribution for every studied method, along the whole EuRoC dataset. Results before visual-inertial BA show that all methods tend to underestimate the true scale. This bias is worse in VINS-Mono initialization, where there is a high number of initial solutions whose scale is close to zero. In contrast, the bias is much lower for inertial-only optimization at 4 Hz, that uses 2.15 s trajectories, whose mean is close to one. After visual-inertial BA, the bias almost disappears, having all methods a distribution with mean close to one, but with different variances, being VINS-Mono with 1s trajectories the worst and our inertial-only optimization with 2 s trajectories the best.

Finally, considering the computing times in table 3.6, inertial-only optimization is extremely efficient, taking around 5 ms per run, rotating and scaling points, frames and velocities takes 11 ms, and full VI-BA requires 132 ms. The inertial-only optimization time is much lower than the time required by the Martinelli-Kaiser closed-form solution,

Table 3.5: Results of exhaustive initialization attempts every 0.5s in EuRoC dataset. The first two blocks compare our proposal with the best joint initialization method [19] using trajectories of ~ 2 seconds (t_{Init}), while the last two blocks compare it with the loosely-coupled initialization of VINS-mono [108] using trajectories of ~ 1.3 seconds.

Seq. Name	Joint Initialization [19]				Inertial-only Optimization (10 KFs @ 4Hz)				VI BA 4 Hz	VINS-Mono Initialization [108]				Inertial-only Optimization (10 KFs @ 10Hz)				VI BA 10 Hz
	scale error (%)				scale error (%)					scale error (%)				scale error (%)				
	MK	MK+ BA1&2	t_{Init} (s)	t_{Tot} (s)	Inert. Only	Inert. Only+BA	t_{Init} (s)	t_{Tot} (s)	scale error (%)	VI Align.	VI Align. + BA	t_{Init} (s)	t_{Tot} (s)	Inert. Only	Inert.-Only + BA	t_{Init} (s)	t_{Tot} (s)	scale error (%)
V1_01	21.11	5.39	2.23	3.18	10.41	4.99	2.16	2.78	15.04	50.2	27.67	1.15	1.54	20.34	7.69	1.26	1.89	26.04
V1_02	31.37	6.11	0.97	3.79	11.27	4.48	2.15	2.88	21.59	69.65	36.36	1.46	2.46	29.98	9.83	1.27	1.82	19.95
V1_03	35.62	4.65	1.07	5.06	14.19	4.25	2.13	4.48	19.74	78.79	31.53	1.23	3.23	36.58	9.24	1.25	3.07	21.79
V2_01	23.76	6.72	2.26	6.83	8.66	5.29	2.19	2.88	9.77	42.65	17.32	1.12	1.86	16.09	8.09	1.28	1.98	22.55
V2_02	28.65	7.00	0.93	4.49	6.27	3.33	2.19	2.82	20.02	53.27	18.61	1.02	1.98	20.27	6.27	1.26	1.73	23.09
V2_03	32.36	7.46	0.89	12.11	22.24	8.04	2.23	6.23	17.13	65.28	22.7	1.17	3.81	35.57	9.32	1.28	4.37	25.95
MH_01	29.23	7.65	2.98	12.87	6.21	4.48	2.14	5.03	11.78	20.98	16.86	2.78	3.08	18.12	6.2	1.26	3.97	18.90
MH_02	21.62	5.71	2.93	10.57	7.07	4.31	2.14	3.77	18.02	19.89	13.15	1.64	2.03	19.76	6.82	1.27	2.61	23.08
MH_03	28.75	6.38	2.06	15.51	9.88	4.77	2.16	3.32	5.27	39.3	16.09	1.30	1.75	31.32	9.89	1.25	2.29	10.34
MH_04	28.65	5.23	2.41	36.57	16.3	7.86	2.14	3.59	6.77	57.19	20.28	1.11	2.11	43.78	13.78	1.26	2.82	11.15
MH_05	25.28	3.51	2.73	38.00	16.05	6.37	2.17	3.48	6.84	55.6	21.96	1.34	1.72	41.64	12.88	1.26	2.4	12.13
Mean Values	27.85	5.98	1.95	13.54	11.69	5.29	2.16	3.75	13.82	50.25	22.05	1.39	2.32	28.50	9.09	1.26	2.63	19.60

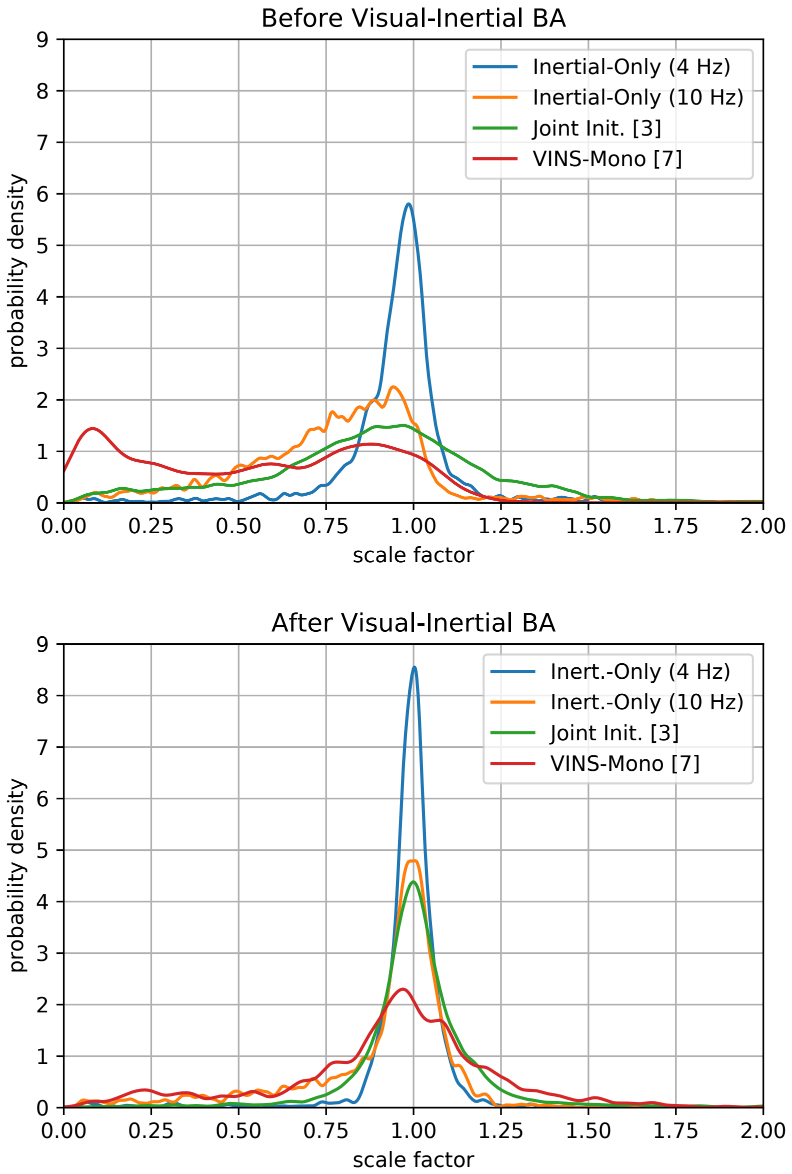


Figure 3.9: Experimental distribution of the scale factor (ratio between estimated and true scales) obtained by the different initialization methods along all sequences of the EuRoC dataset, before and after visual-inertial BA. A total of 2248 initializations have been launched.

which is around 60 ms [19]. Compared with the baseline VI-BA which requires three runs with different scales for a total 400 ms, our complete method only takes 160 ms, and doubles the scale accuracy.

Once verified that our inertial-only optimization performs better than previous initialization methods, we have made a second experiment which consists in launching ORB-SLAM Visual-Inertial [98] using our new initialization. As in [19], we perform two visual-inertial bundle adjustment 5 and 10 seconds after initialization. We check three different sequences of EuRoC dataset, with different difficulty degrees, running 5 experiments for each one. We align both SLAM and GT trajectories, and Absolute Trajectory Error (ATE) is measured.

Results from table 3.7 show that ORB-SLAM VI reaches in sequences V1_01 and

Table 3.6: Computing time of our method for the exhaustive initialization experiment in sequence V1_02.

Step	mean (ms)	median (ms)	max (ms)
Inertial-Only	3×5.24	3×4.92	3×6.39
Map update	11.18	11.25	13.98
Visual-Inertial BA	132.78	136.43	198.07
Total	159.68	163.92	228.24

Table 3.7: Results for ORBSLAM-VI with the proposed initialization (median values on five executions are shown) compared with results reported for original ORBSLAM-VI [98] and VINS-Mono in [110].

Seq. Name	ORBSLAM-VI + our initialization		ORBSLAM-VI [98]		VINS-Mono [110]
	Scale error (%)	RMSE ATE (m)	Scale error (%)	RMSE ATE (m)	RMSE ATE (m)
V1_01	0.4	0.023	0.9	0.027	0.060
V1_02	0.3	0.026	0.8	0.024	0.090
V1_03	1.7	0.059	-	-	0.180

V1_02 similar accuracy levels using the proposed initialization and the original initialization from [98]. In addition, sequence V1_03, which previously could not be processed, because the original initialization failed, can now be successfully processed. This is because the new initialization takes just 2 seconds, being possible to immediately use the IMU, avoiding tracking loss during subsequent fast motions. Our results show that the combination of our initialization method with ORB-SLAM VI gives a very robust system that is significantly more accurate than VINS-Mono.

3.4 Discussion

In section 3.2 we have proposed a fast joint monocular-inertial initialization method, based on the work of [90] and [69]. We have adapted it to be more general, allowing incomplete feature tracks, and more computationally efficient using the IMU preintegration method from [51]. Our results show that the original Martinelli-Kaiser technique does not provide a good enough initialization in most practical scenarios leading to large unpredictable errors, hence we have proposed two novel tests to detect bad initializations and two visual-inertial BA steps to improve the solution. These techniques have proven to be worth it to reduce scale error down to 5% and reject all bad initializations, despite this could lead to a lower recall. Solutions found after those steps are good enough to launch Visual-Inertial ORB-SLAM and converge to very accurate maps. In summary, we have developed a fast method for joint initialization of monocular-inertial SLAM, using trajectories of 1 to 2 seconds, that is much more accurate and robust than the original technique [69], with a maximum computational cost of 215ms.

Aiming at a more efficient initialization method with a higher recall, we have proposed a disjoint visual-inertial initialization. The proposal, stated as a MAP estimation problem, has shown to be more accurate than the top-performing methods in the literature, including our previous proposal, for the same initialization trajectory length, with a very low computing time. This confirms that optimal estimation theory is able to make a proper use of the probabilistic models of sensor noises, obtaining more accurate results than solving linear systems of equations or using non-weighted least squares. Full visual-inertial BA is a very non-linear problem, plagued with local minima. We have split it in a fully observable up-to-scale visual problem, followed by an inertial-only optimization phase that can be solved very efficiently, producing an initial guess that alleviates the local minima problem.

As future work we would like to study the possibility of making inertial-only optimization information self-aware. This means not to use a fixed initialization time, but an adaptive trajectory length which would depend on the estimated information of inertial parameters. On the other hand, initialization trajectories may contain motions with low information (IMU almost steady, not excited) along with more aggressive motions. Instead of using the entire trajectory, taking only parts of the trajectory with higher information could be interesting to reduce the computational time and improve the convergence of the optimization.

Chapter 4

ORB-SLAM3: A full visual-inertial system

In this chapter, we present ORB-SLAM3, the first system able to perform visual, visual-inertial and multi-map SLAM with monocular, stereo and RGB-D cameras, using pin-hole and fisheye lens models.

The first main novelty is a feature-based tightly-integrated visual-inertial SLAM system that fully relies on Maximum-a-Posteriori (MAP) estimation, even during the IMU initialization phase. The result is a system that operates robustly in real time, in small and large, indoor and outdoor environments, and is two to ten times more accurate than previous approaches.

The second main novelty is a multiple map system that relies on a new place recognition method with improved recall. Thanks to it, ORB-SLAM3 is able to survive long periods of poor visual information: when it gets lost, it starts a new map that will be seamlessly merged with previous maps when revisiting mapped areas. Compared with visual odometry systems that only use information from the last few seconds, ORB-SLAM3 is the first system able to reuse in all the algorithm stages all previous information. This allows us to include in bundle adjustment co-visible keyframes that provide high parallax observations, boosting accuracy, even if they are widely separated in time or if they come from a previous mapping session.

Our experiments show that, in all sensor configurations, ORB-SLAM3 is as robust as the best systems available in the literature, and significantly more accurate. Notably, our stereo-inertial SLAM achieves an average accuracy of 3.6 cm on the EuRoC drone and 9 mm under quick hand-held motions in the room of TUM-VI dataset, a setting representative of AR/VR scenarios. For the benefit of the community we make public the source code [20].

In this chapter, we will focus on the novelties and work related with the visual-inertial SLAM, which is the contribution of this thesis, while those related with the multimap system will be just mentioned or even skipped, since they come from work by Elvira et al. [41]

4.1 Introduction

Intense research on Visual Simultaneous Localization and Mapping systems (SLAM) and Visual Odometry (VO), using cameras alone or in combination with inertial sensors, has produced during the last two decades excellent systems, with increasing accuracy and robustness. As presented in section 2.1, modern systems rely on Maximum a Posteriori (MAP) estimation, which in the case of visual sensors corresponds to Bundle Adjustment (BA), either geometric BA that minimizes feature reprojection error, in feature-based methods, or photometric BA that minimizes the photometric error of a set of selected pixels, in direct methods.

With the recent emergence of VO systems that integrate loop-closing techniques, the frontier between VO and SLAM is more diffuse. A complete discussion of most significant VO/SLAM systems may be found in section 1.2.

In this work we build on ORB-SLAM, [97, 99], and ORB-SLAM Visual-Inertial, [98], the first visual and visual-inertial systems able to take full profit of short-term, mid-term and long-term data association, as exlpained in section 1.2.1, reaching zero drift in mapped areas. Here we go one step further providing **multi-map data association**, which allows us to match and use in BA map elements coming from previous mapping sessions, achieving the true goal of a SLAM system: building a map that can be used later to provide accurate localization.

The most important contribution is the ORB-SLAM3 library itself, the most complete and accurate visual, visual-inertial and multi-map SLAM system to date (see table 1.1). The main novelties of ORB-SLAM3 are:

- **A monocular and stereo visual-inertial SLAM system** that fully relies on Maximum-a-Posteriori (MAP) estimation, even during the IMU (Inertial Measurement Unit) initialization phase. The initialization method proposed was previously presented in [21], here we extend it to the stereo case and use it to refine the scale and gravity direction once system is initialized.

- **A monocular and stereo visual-inertial SLAM system**, built on top of ORB-SLAM VI, and extended to rectified and non-rectified stereo cameras. After a thorough evaluation in public datasets, results show that the monocular and stereo visual-inertial systems are extremely robust and significantly more accurate than other visual-inertial approaches, even in sequences without loops.

- **ORB-SLAM Atlas** [41], the first complete multi-map SLAM system able to handle visual and visual-inertial systems, in monocular and stereo configurations.

- **An abstract camera representation** making the SLAM code agnostic of the camera model used, and allowing to add new models by providing their projection, un-projection and Jacobian functions. We provide the implementations of pinhole [134] and fisheye [70] models.

All these novelties, together with a few code improvements make ORB-SLAM3 the new reference visual and visual-inertial open-source SLAM library, being as robust as the best systems available in the literature, and significantly more accurate, as shown by our experimental results in section 4.4. We also provide comparisons between monocular, stereo, monocular-inertial and stereo-inertial SLAM results that can be of interest for practitioners.

4.2 System Overview

ORB-SLAM3 is built on ORB-SLAM2 [97] and ORB-SLAM-VI [98]. It is a full multi-map and multi-session system able to work in pure visual or visual-inertial modes with

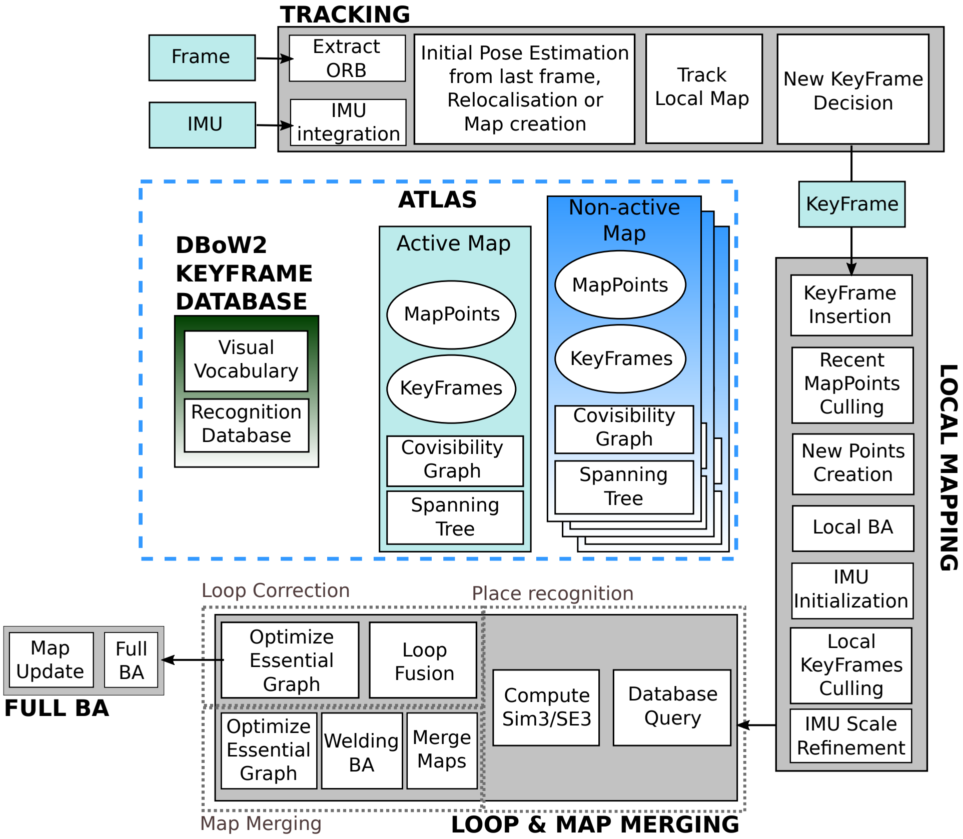


Figure 4.1: Main system components of ORB-SLAM3. This figure was original from [41] and adapted to our system

monocular, stereo or RGB-D sensors, using pin-hole and fisheye camera models. Figure 4.1 shows the main system components, that are parallel to those of ORB-SLAM2 with some significant novelties, that are summarized next:

- **Atlas** is a multi-map representation composed of a set of disconnected maps. There is an active map where the tracking thread localizes the incoming frames, and is continuously optimized and grown with new keyframes by the local mapping thread. We refer to the other maps in the Atlas as the non-active maps. The system builds a unique DBoW2 database of keyframes that is used for relocalization, loop closing and map merging.

- **Tracking thread** processes sensor information and computes the pose of the current frame with respect to the active map in real-time, minimizing the reprojection error of the matched map features. It also decides whether the current frame becomes a keyframe. In visual-inertial mode, the body velocity and IMU biases are estimated by including the inertial residuals in the optimization. When tracking is lost, the tracking thread tries to relocate the current frame in all the Atlas' maps. If relocated, tracking is resumed, switching the active map if needed. Otherwise, after a certain time, the active map is stored as non-active, and a new active map is initialized from scratch.

- **Local mapping thread** adds keyframes and points to the active map, removes the redundant ones, and refines the map using visual or visual-inertial bundle adjustment, operating in a local window of keyframes close to the current frame. Additionally, in the inertial case, the IMU parameters are initialized and refined by the mapping thread using our novel MAP-estimation technique.

- **Loop and map merging thread** detects common regions between the active map and the whole Atlas at keyframe rate. If the common area belongs to the active map, it performs loop correction; if it belongs to a different map, both maps are seamlessly merged into a single one, that becomes the active map. After a loop correction, a full BA is launched in an independent thread to further refine the map without affecting real-time performance.

4.3 Visual-Inertial SLAM

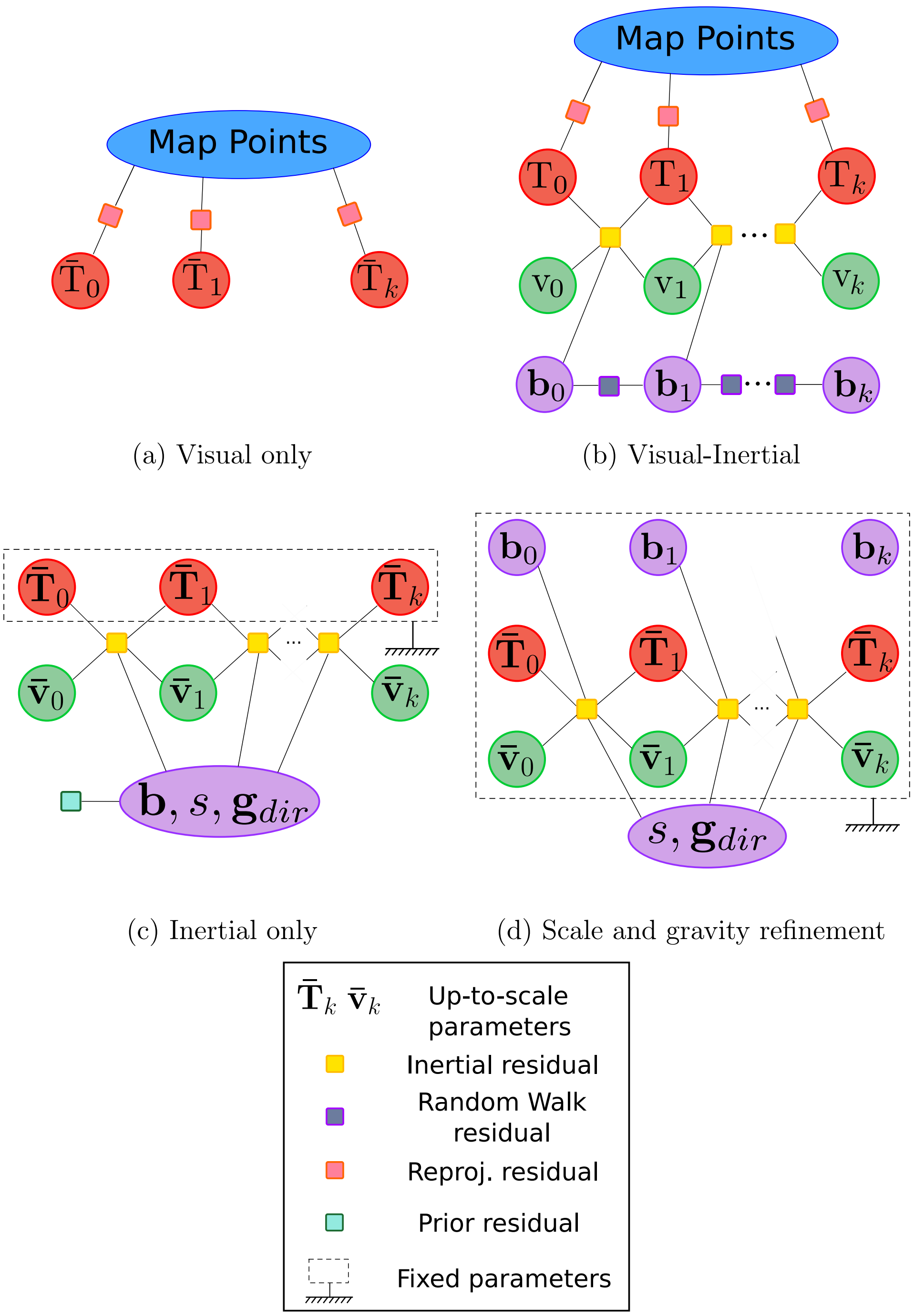
ORB-SLAM-VI was the first true visual-inertial SLAM system capable of map reusing. However, it was limited to pinhole and monocular cameras, and its initialization was too slow, failing in some challenging scenarios. In this work, we build on ORB-SLAM-VI providing a fast an accurate IMU initialization technique, and an open-source SLAM library capable of monocular-inertial and stereo-inertial SLAM, with pinhole and fisheye cameras.

4.3.1 IMU Initialization and scale refinement

As introduced in chappter 3 the goal of this step is to obtain good initial values for the inertial variables: body velocities, gravity direction, and IMU biases. Some systems like VI-DSO, [137], try to solve from scratch visual-inertial BA, sidestepping a specific initialization process, obtaining slow convergence for inertial parameters (up to 30 seconds). For ORB-SLAM3 we propose to use our previous work, based on an inertial-only optimization and detailed in section 3.3, to initialize the IMU. This initialization consists in solving an optimization problem equivalent to figure 4.2c, where only inertial parameters are optimized. This contrasts with pure visual BA, figure 4.2a, where only visual measurements are considered for camera pose and map point estimation, as well as visual-inertial BA, figure 4.2b, where IMU measurements are also included to estimate gravity direction, camera velocities and true scale poses.

First, we have easily extended it to the stereo-inertial initialization by fixing the scale factor to one and taking it out from the inertial-only optimization variables. Having a known scale boosts its convergence, leading to a lower number of iterations. In addition, it also removes the need of optimizing with three scale initial guesses as we did for the monocular case. This also entails an important computational cost reduction.

Second, in some specific cases, when slow motion does not provide good observability of the inertial parameters, the initialization may fail to converge to accurate solutions. If this error is not corrected by Local Bundle Adjustment before some keyframes get out of the optimization window, it will lead to a scale offset in the whole map. To get robustness against this situation, we propose a novel scale refinement technique, based on a modified inertial-only optimization, where all inserted keyframes are included but scale and gravity direction are the only parameters to be estimated (figure 4.2d). Notice that in that case, the assumption of constant biases would not be correct. Instead, we use the values estimated from mapping, and we fix them. This optimization, which is very computationally efficient, is performed in the Local Mapping thread every ten seconds,



(a) Visual only

(b) Visual-Inertial

(c) Inertial only

(d) Scale and gravity refinement

Figure 4.2: Factor graph representation for different optimizations along the system

until the map has more than 100 keyframes or more than 75 seconds have passed since initialization.

4.3.2 Tracking

For tracking we adopt the scheme proposed in [98] and extend it to the stereo-inertial case. At ORB-SLAM3, visual-inertial tracking consists of estimating at frame rate the body state, composed by pose, velocity and biases, while map points remain fixed. This can be split into two steps,

- **Pose prediction**, instead of using an ad-hoc constant velocity model, we directly use integrated inertial measurements to predict current frame pose using equations 2.13. We project the local map [99] into this estimated pose and look for ORB matches with a window search around the projection point smaller than the pure visual case.

- **State optimization**. Given visual and inertial measurements we build a simplified visual-inertial optimization where only the states of the last two frames are optimized, while map points remain fixed.

Visual and inertial errors appearing in the state optimization for the monocular-inertial case have been introduced in sections 2.1 and 2.2. Here we describe the necessary modifications for the stereo-inertial case. At ORB-SLAM3 we consider two stereo cases (See appendix E). First, for a pinhole rectified stereo (rs) camera [131] we fuse left and right camera observations in an equivalent 3 components residual which describes the whole stereo observation. Given point j observed from camera i, this can be written as:

$$\mathbf{r}_{i,\mathrm{rs}}^{j} = \begin{bmatrix} \mathbf{u}_{L_i}^{j} \\ u_{R_i}^{j} \end{bmatrix} - \pi_{rs}(\mathbf{x}_i^j) = \begin{bmatrix} u_{L_i}^{j} \\ v_{L_i}^{j} \\ u_{R_i}^{j} \end{bmatrix} - \begin{bmatrix} f_x x_i^j / z_i^j + c_x \\ f_y y_i^j / z_i^j + c_y \\ f_x (x_i^j - b)/z_i^j + c_x \end{bmatrix} \tag{4.1}$$

where the position for point j at camera i reference, $\mathbf{x}_i^j$, is computed as:

$$\mathbf{x}_i^j = (x_i^j, y_i^j, z_i^j)^T = \mathbf{T}_{\mathtt{CB}} \oplus \mathbf{T}_{i\mathtt{W}} \oplus \mathbf{x}^j \tag{4.2}$$

where $\mathbf{T}_{i\mathtt{W}}$ stands for the i-th Body position, (u, v) are the horizontal and vertical observations of 3D point in the image plane, and b stands for the stereo baseline only along the x camera axis. More details about this residual, including its derivatives w.r.t body pose and point position are given in appendix B.2. The second case corresponds to a non-rectified stereo camera (nrs), which is used for fisheye lenses to avoid rectification issues due to large field of view, but could be also used for pinhole cameras. In this case, left and right observations are treated as two monocular observations, $\mathbf{u}_{L_i}^{j}$ and $\mathbf{u}_{R_i}^{j}$, of the same point. Concatenating both leads to the following residual:

$$\mathbf{r}_{i,\mathrm{nrs}}^{j} = \begin{bmatrix} \mathbf{u}_{L_i}^{j} \\ \mathbf{u}_{R_i}^{j} \end{bmatrix} - \pi_{nrs}(\mathbf{T}_{i\mathtt{W}}, \mathbf{x}^j) = \begin{bmatrix} \mathbf{u}_{L_i}^{j} \\ \mathbf{u}_{R_i}^{j} \end{bmatrix} - \begin{bmatrix} \pi_m(\mathbf{T}_{\mathtt{C}_L\mathtt{B}} \oplus \mathbf{T}_{i\mathtt{W}} \oplus \mathbf{x}^j) \\ \pi_m(\mathbf{T}_{\mathtt{C}_R\mathtt{B}} \oplus \mathbf{T}_{i\mathtt{W}} \oplus \mathbf{x}^j) \end{bmatrix} \tag{4.3}$$

Relative transformations $\mathbf{T}_{\mathtt{C}_L\mathtt{B}}$ and $\mathbf{T}_{\mathtt{C}_R\mathtt{B}}$ between cameras and body (IMU) are supposed to be known and provided with the calibration. π_m stands for the monocular projection, pinhole or fisheye.

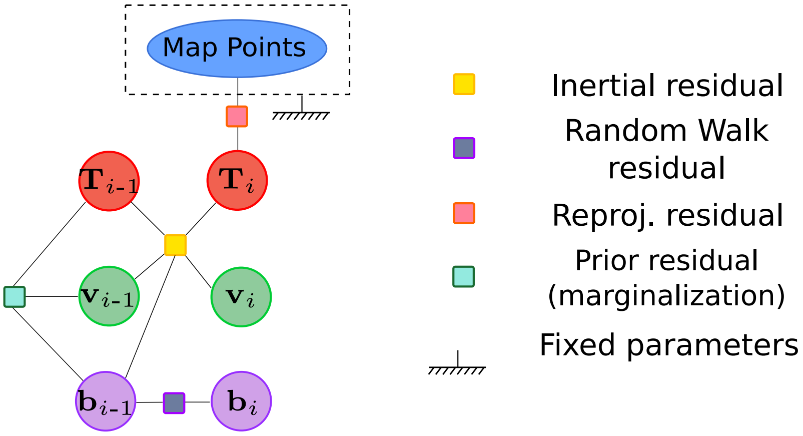


Figure 4.3: Visual-Inertial tracking factorgraph. Two last frames are optimized together while map points remain fixed. Previous frame state is marginalized out at the end of tracking to have a new prior for next optimization.

With these stereo errors and IMU residuals previously introduced in section 2.2, we can build the state optimization as done by [98]. This optimization, represented in figure 4.3, consists of refining states from the last two frames, to make possible to include IMU measurements. While map points contribute to the final cost by means of reprojection error, they remain fixed during minimization. At the end of this optimization, the previous frame is marginalized, leading to a prior residual that will be used for the next optimization. This process is repeated until mapping or loop-closing threads update the map. At this moment, the linearization point from marginalization is not valid anymore. Hence, we perform this optimization but with respect to the last keyframe, which has been effectively updated, as proposed at [98]. Then, we continue with the normal procedure optimizing the last two frames states (figure 4.3)

4.3.3 Mapping

Mapping is the back-end process which performs all map modifications, optimizing map elements(map points and keyframes), as well as creating and removing them. This process is required to run at keyframe frequency, usually 1-5Hz. For this reason, trying to solve the optimization from equation 2.17 for the whole map would be intractable. Two approaches may be found in the visual-inertial literature, the first one comprises marginalizing old variables and was firstly stated by [80]. They proposed a set of heuristics to marginalize old variables and discard measurements aiming to keep the sparse structure of point block, which is the key for an efficient optimization. This solution may be close to optimal but prevents from reusing variables and fixes the linearization point. The second approach, proposed by [98], is to fix old variables, being possible to use and refine them later if the same region is visited. This is not optimal in the statistical sense, but entails a big improvement by setting mid-term data associations, which is the key for accurate non exploratory visual-inertial SLAM. At ORB-SLAM3 we follow this later approach and we show in section 4.4 how our system outperforms existing visual-inertial system.

The back end optimization is similar to that one from [98] and it is summarized in figure 4.4. We use as optimizable variables a sliding window of the N last keyframes and their observed points. We also include covisible keyframes but keeping them fixed. The

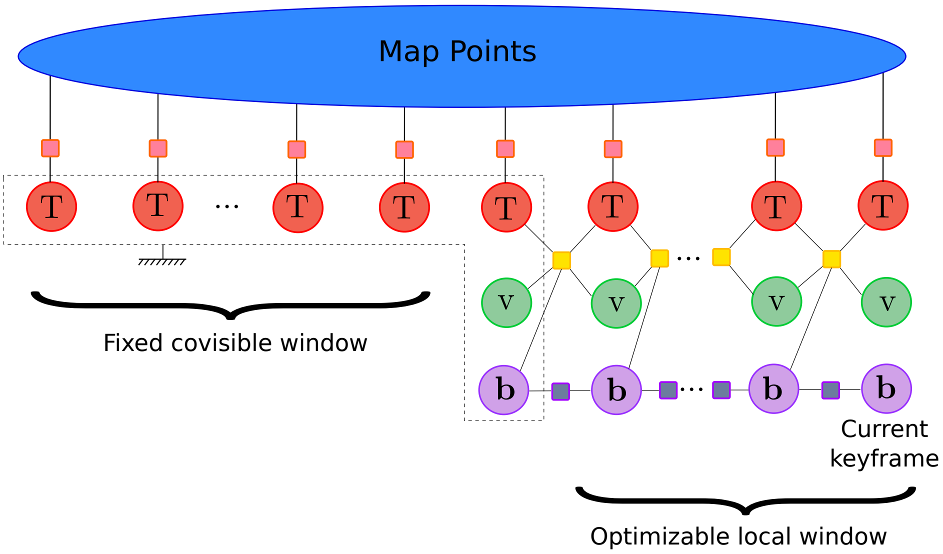


Figure 4.4: Factor graph representation of visual-inertial Local Bundle Adjustment.

optimizable window size is set to 10. However, if the number of tracked map points is high (150 points or more) and keyframes are being created at a lower rate, we have more time to perform this optimization and we double its size.

4.3.4 Robustness to tracking loss

In pure visual SLAM or VO systems, temporal camera occlusion and fast motions result in losing track of visual elements, getting the system lost. ORB-SLAM pioneered the use of fast relocation techniques based on bag-of-words place recognition, but they proved insufficient to solve difficult sequences in the EuRoC dataset [97]. Our visual-inertial system enters into *visually lost* state when less than 15 point maps are tracked, and achieves robustness in two stages:

- *Short-term lost*: the current body state is estimated from IMU readings, and map points are projected in the estimated camera pose and searched for matches within a large image window. The resulting matches are included in visual-inertial optimization. In most cases this allows recovery of visual tracking. Otherwise, after 5 seconds, we pass to the next stage.

- *Long-term lost*: A new visual-inertial map is initialized as explained above, and it becomes the active map. Two competing processes are launched in parallel, multi-map relocalization and new map initialization. If relocalization is successful, we perform pure visual tracking and IMU preintegration for 1 second, and perform a simplified inertial-only optimization, fixing scale and gravity terms, obtaining estimations for velocities and biases, and launch again Visual-Inertial SLAM. If a new map is initialized before relocalization succeeds, the mapping proceeds with the new map. If the system gets lost before 15 seconds after IMU initialization, the map is discarded. This prevents the accumulation of inaccurate and meaningless maps.

4.3.5 Visual-Inertial Loop Closing and Map Merging

Short-term and mid-term data-associations between a frame and the active map are routinely found by the tracking and mapping threads by projecting map points into the estimated camera pose and searching for matches in an image window of just a few pixels. To achieve long-term data association for relocation and loop detection, ORB-SLAM3 uses the DBoW2 bag-of-words place recognition system [54, 96]. This method has been also adopted by most recent VO and SLAM systems that implement loop closures (Table 1.1). Here we present the novelties at ORB-SLAM3 for the visual-inertial case, which are also works of this thesis, and its differences with respect to the pure visual case by Elvira et al. [41].

The place recognition module detects candidate keyframes (c) for the active keyframe (a), based on bag-of-words similarities. If the candidate keyframe belongs to the same map, it is a loop candidate, if not, it is a map merging candidate. A first alignment of pointclouds from both keyframes results in a relative transformation $\mathbf{T}_{ac} \in \mathrm{SE}(3)$. In case the IMU is not initialized in the active map, the matching transformation also includes the scale change $\mathbf{T}_{ac} \in \mathrm{Sim}(3)$. We highlight this rarely happens, since the initialization only takes two seconds.

At this stage, we can use inertial information to discard potential false positives. Since gravity direction is observable for a visual-inertial system, it is not possible to have drift in scale nor pitch and roll. In this sense, we define the following transformation:

$$\mathbf{T}_{\mathtt{W}\mathtt{W}'} = \mathbf{T}_{\mathtt{W}a}\mathbf{T}_{ac}\mathbf{T}_{c\mathtt{W}'} \tag{4.4}$$

and we take the logarithm of its rotational part and check that rotations around x and y are below some threshold. If this condition is not fulfilled, we discard that candidate.

After a candidate has been validated, correction needs to be propagated to the remaining map, performing a first pose-graph optimization (PGO). This consists of finding the absolute body poses given a set of relative transformations $\{\Delta\mathbf{T}_{ij}\}$. At visual SLAM these relative measurements may come from three different situations. First, loop closing or map merging constraints as a result of place recognition and point cloud alignment. Second, covisibility constraints, relating keyframes with a high number of common observations. Third, spanning tree constraints [99], to make sure the graph is connected. In addition, for the inertial case, we add an extra constraint which relates consecutive keyframes where there exists an inertial integrated measurement between them. For each one of these relative measurements between keyframes i and j, we define the following residual:

$$\mathbf{r}_{ij} = \mathrm{Log}_{SE(3)}(\mathbf{T}_{\mathtt{W}i}\Delta\mathbf{T}_{ij}\mathbf{T}_{j\mathtt{W}}) \in \mathbb{R}^6 \tag{4.5}$$

Contrary to the pure visual case [126], where 7 or 6 DoF exist for respectively monocular and stereo cases, in visual-inertial estimation we only have 4. DoFs for scale, roll and pitch are observable [71]. This is directly reflected in the pose-graph optimization, updating keyframe poses only with these 4 DoF, as proposed by [108]. This supposes an important reduction on the number of optimizable variables, and therefore in the computational cost.

In pure visual SLAM, a final full bundle adjustment is performed after the PGO, to improve map accuracy. In visual-inertial SLAM, keyframe states consist on 15 variables, instead of only 6, giving a much higher computational cost, since time scales cubically with the number of variables. However, the drift of our visual-inertial SLAM is much

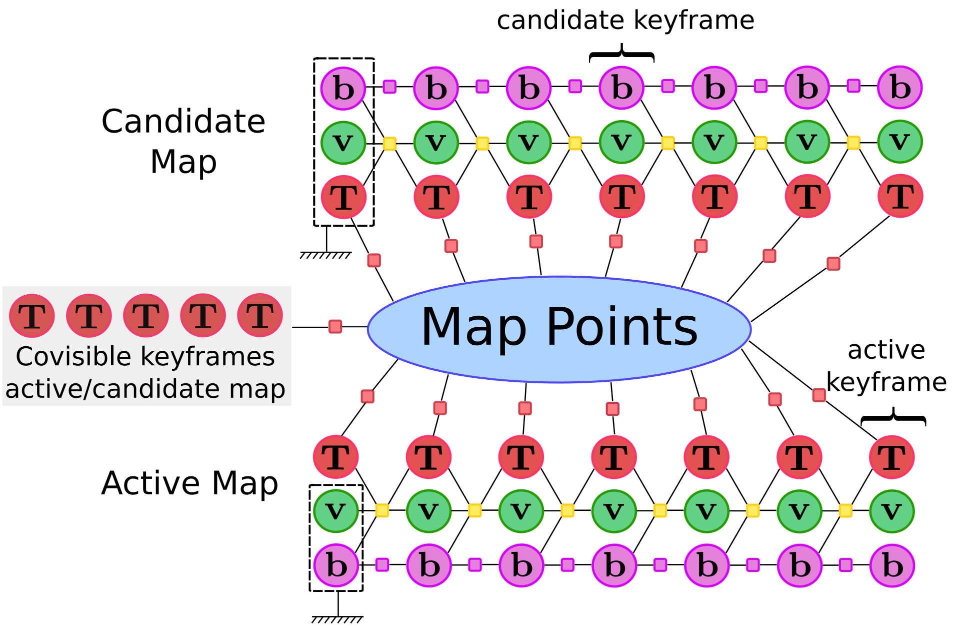


Figure 4.5: Factor graph representation for the merging. Inertial measurements are included only in the welding area, where there are observations from points belonging to both maps.

lower than in the visual case, and we can safely skip the costly full bundle adjustment, as it does not provide a remarkable improvement.

When a successful place recognition produces multi-map data association, after PGO we perform a visual-inertial Bundle Adjustment involving welding parts from both maps, whose equivalent factor graph is represented in figure 4.5. Poses, velocities and biases of the active keyframe a and its 5 last temporal keyframes are included as optimizable. These variables are related by IMU preintegration terms. For the candidate map, we proceed similarly, including poses, velocities and biases of the candidate keyframe c and its 5 temporal neighbours, as shown in Figure 4.5. For the candidate map, the keyframe immediately before the local window is included but fixed, while for the active map the similar keyframe is included with fixed IMU terms, but its pose remains optimizable, to avoid overconstraints. All points seen by all these keyframes, and the keyframes' poses observing these points are also optimized. All the keyframes and points are related by means of reprojection error.

4.4 Experimental Results

The evaluation of the whole system is split in:

- Single session experiments in EuRoC [16]: each of the 11 sequences is processed to produce a map, with the four sensor configurations: Monocular, Monocular-Inertial, Stereo and Stereo-Inertial.

- Performance of monocular and stereo visual-inertial SLAM with fisheye cameras, in the challenging TUM VI Benchmark [121].

- Multi-session experiments in both datasets.

Table 4.1: Performance comparison in the EuRoC dataset (RMS ATE in m., scale error in %). Except where noted, we show results reported by the authors of each system, for all the frames in the trajectory, comparing with the processed GT.

			MH01	MH02	MH03	MH04	MH05	V101	V102	V103	V201	V202	V203	Avg[1]
Monocular	ORB-SLAM [98]	ATE[2,3]	0.071	0.067	0.071	0.082	**0.060**	**0.015**	0.020	-	0.021	**0.018**	-	0.047*
	DSO [46]	ATE	0.046	0.046	0.172	3.810	0.110	0.089	0.107	0.903	0.044	0.132	1.152	0.601
	SVO [53]	ATE	0.100	0.120	0.410	0.430	0.300	0.070	0.210	-	0.110	0.110	1.080	0.294*
	DSM [154]	ATE	0.039	**0.036**	0.055	**0.057**	0.067	0.095	0.059	0.076	0.056	0.057	**0.784**	**0.126**
	ORB-SLAM3 (ours)	ATE	**0.031**	0.045	**0.026**	0.096	0.083	0.033	**0.015**	**0.037**	**0.019**	0.019	-	0.041*
Stereo	ORB-SLAM2 [97]	ATE	0.035	**0.018**	0.028	0.119	0.060	**0.035**	**0.020**	**0.048**	**0.037**	0.035	-	0.044*
	VINS-Fusion [110]	ATE	0.540	0.460	0.330	0.780	0.500	0.550	0.230	-	0.230	0.200	-	0.424*
	SVO [53]	ATE	0.040	0.070	0.270	0.170	0.120	0.040	0.040	0.070	0.050	0.090	0.790	0.159
	ORB-SLAM3 (ours)	ATE	**0.033**	0.019	**0.025**	**0.116**	**0.049**	**0.035**	0.021	0.094	0.047	**0.019**	**0.426**	**0.084**
Monocular Inertial	MCSKF [94]	ATE[5]	0.420	0.450	0.230	0.370	0.480	0.340	0.200	0.670	0.100	0.160	1.130	0.414
	OKVIS [80]	ATE[5]	0.160	0.220	0.240	0.340	0.470	0.090	0.200	0.240	0.130	0.160	0.290	0.231
	ROVIO [13]	ATE[5]	0.210	0.250	0.250	0.490	0.520	0.100	0.100	0.140	0.120	0.140	0.140	0.224
	ORBSLAM-VI [98]	ATE[2,3] scale error[2,3]	0.075 0.5	0.084 0.8	0.087 1.5	0.217 3.5	0.082 0.5	**0.027** 0.9	0.028 0.8	- -	**0.032** 0.2	0.041 1.4	0.074 0.7	0.075* 1.1*
	VINS-Mono [108]	ATE[4]	0.084	0.105	0.074	0.122	0.147	0.047	0.066	0.180	0.056	0.090	0.244	0.110
	VI-DSO [137]	ATE scale error	0.062 1.1	**0.044** 0.5	0.117 0.4	0.132 0.2	0.121 0.8	0.059 1.1	0.067 1.1	0.096 0.8	0.040 1.2	0.062 0.3	0.174 0.4	0.089 0.7
	ORB-SLAM3 (ours)	ATE scale error	**0.043** 1.0	0.066 1.5	**0.042** 0.2	**0.091** 0.9	**0.062** 0.5	0.042 1.4	**0.013** 0.3	**0.031** 1.7	0.038 0.4	**0.022** 0.1	**0.018** 0.3	**0.043** 0.8
Stereo Inertial	VINS-Fusion [110]	ATE[4]	0.166	0.152	0.125	0.280	0.284	0.076	0.069	0.114	0.066	0.091	0.096	0.138
	BASALT [136]	ATE[3]	0.080	0.060	0.050	0.100	**0.080**	0.040	0.020	0.030	**0.030**	0.020	-	0.051*
	Kimera [114]	ATE	0.080	0.090	0.110	0.150	0.240	0.050	0.110	0.120	0.070	0.100	0.190	0.119
	ORB-SLAM3 (ours)	ATE scale error	**0.038** 0.7	**0.027** 0.3	**0.028** 0.3	**0.055** 0.3	0.097 1.1	**0.037** 0.6	**0.014** 0.6	**0.023** 0.7	0.034 1.2	**0.014** 0.3	**0.032** 0.7	**0.036** 0.6

[1] Average error of the successful sequences. Systems that did no complete all sequences are denoted by * and are not marked in bold.
[2] Errors reported with raw GT instead of processed GT.
[3] Errors reported with keyframe trajectory instead of full trajectory.
[4] Errors obtained by ourselves, running the code with its default configuration.
[5] Errors reported at [37].

As usual in the field, we measure accuracy with RMS ATE [127], aligning the estimated trajectory with ground truth using a Sim(3) transformation in the pure monocular case, and a SE(3) transformation in the rest of sensor configurations. Scale error is computed using s from Sim(3) alignment, as $|1-s|$. All experiments have been run in an Intel Core i7-7700 CPU, at 3.6GHz, with 32 GB memory, using only CPU.

4.4.1 Single-session SLAM on EuRoC

Table 4.1 compares the performance of ORB-SLAM3 using its four sensor configurations with the most relevant systems in the state-of-the-art. Our reported values are the median after 10 executions. As shown in the table, ORB-SLAM3 achieves in all sensor configurations more accurate result than the best systems available in the literature, in most cases by a wide margin.

In monocular and stereo configurations our system is more precise than ORB-SLAM2 due to the better place recognition algorithm that closes loops earlier and provides more mid-term matches. Interestingly, the next best results are obtained by DSM that also uses mid-term matches, even though it does not close loops.

In monocular-inertial configuration, ORB-SLAM3 is five to ten times more accurate than MCSKF, OKVIS and ROVIO, and more than doubles the accuracy of VI-DSO and VINS-Mono, showing again the advantages of mid-term and long-term data association. Compared with ORB-SLAM VI, our novel fast IMU initialization allows ORB-SLAM3 to calibrate the inertial sensor in a few seconds and use it from the very beginning, being able to complete all EuRoC sequences, and obtaining better accuracy. As shown in figure 4.6, ORB-SLAM3 mono-inertial is not only able to process V103 sequence from the very beginning, but it also performs better than any other system. Regarding scale drift, as defined in [152], we see estimation is unbiased and its error is always below 1.5%. We also remark that on average, our monocular-inertial configuration has a better performance than any other stereo or stereo inertial, with the only exception of our own stereo-inertial system.

In stereo-inertial configuration, ORB-SLAM3 is four times more accurate than VINS-Fusion and Kimera. It's accuracy is only approached by the recent BASALT that, being a native stereo-inertial system, was not able to complete sequence V203, where some frames from one of the cameras are missing. Comparing our monocular-inertial and stereo-inertial systems, the latter performs better in most cases. Only for one Machine Hall (MH5) sequence a lower accuracy is obtained. We hypothesize that greater depth scene for MH sequences may lead to less accurate stereo triangulation and hence a less precise scale.

To summarize performance, we have presented the median of ten executions for each sensor configuration. For a robust system, the median represents accurately the behavior of the system. But a non-robust system will show high variance in its results. This can be analyzed using figure 4.7 that shows with colors the error obtained in each of the ten executions. Comparison with the figures for DSO, ROVIO and VI-DSO published in [137] confirms the superiority of our method.

In pure visual configurations, the multi-map system adds some robustness to fast motions by creating a new map when tracking is lost, that is merged later with the global map. This can be seen in sequences V103 monocular and V203 stereo that could not be solved by ORB-SLAM2 and are successfully solved by our system in most executions. As expected, stereo is more robust than monocular thanks to its faster feature initialization,

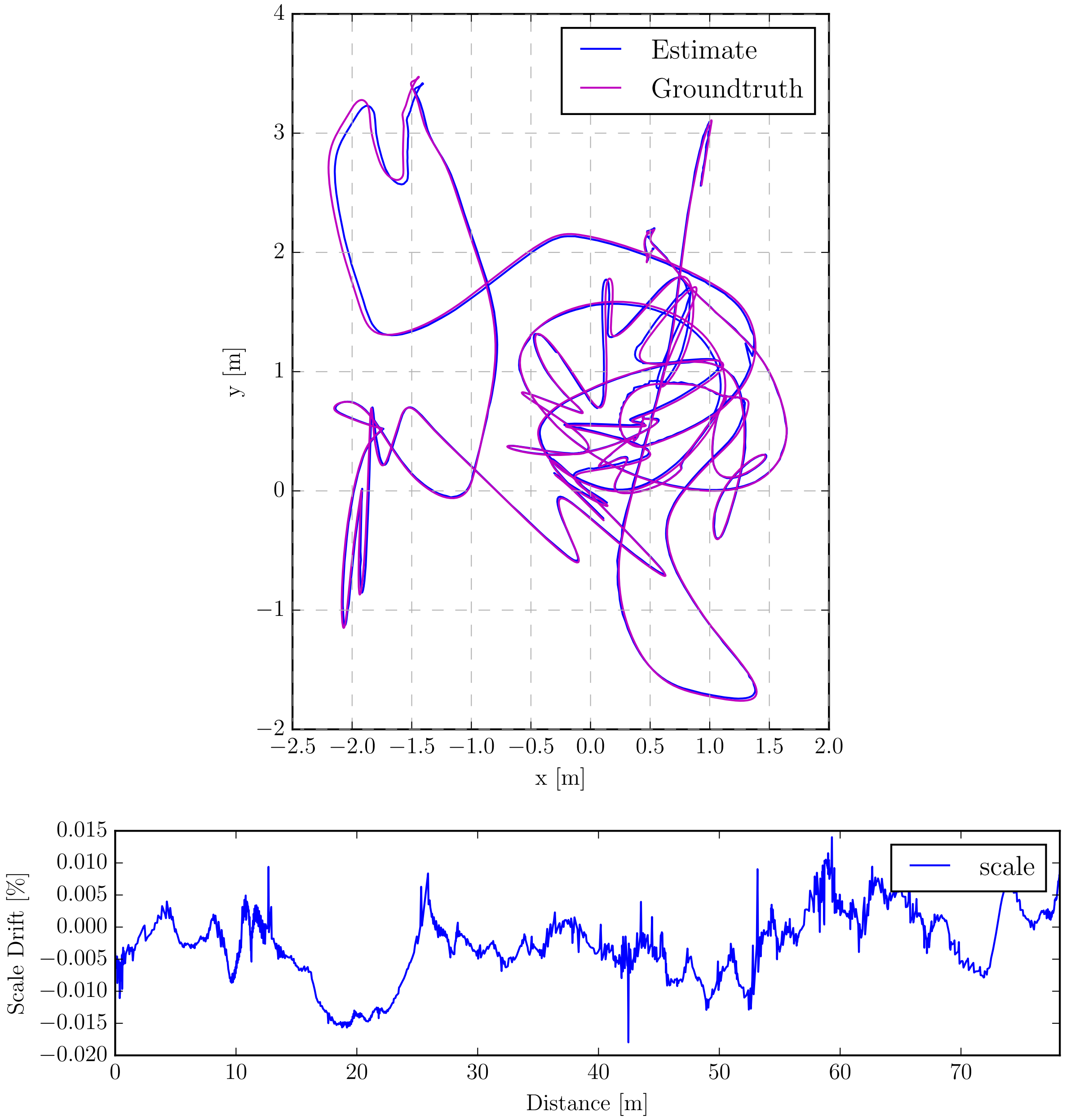


Figure 4.6: Top view of estimated trajectory for V103 difficult sequence using mono-inertial ORB-SLAM3. Results plotted with software from [152]

with the additional advantage that the real scale is estimated.

However, the big leap in robustness is obtained by our novel visual-inertial SLAM system, both in monocular and stereo configurations. With regard to inertial configurations, the stereo-inertial system comes with a very slight advantage, particularly in the most challenging V203 sequence. Comparing visual and visual-inertial configuration, figure 4.7 points out the greater stability of inertial solutions. The randomness of SLAM lies in some existing random processes, such as RANSAC, as well as processor scheduling for multi-thread systems, which leads to slightly different estimations. For the visual case, this has influence in which points are projected, which visual matches are found or what keyframes are included in the local bundle adjustment, affecting final results. However, using an IMU sensor drastically mitigates this effect. First, IMU provides measurements

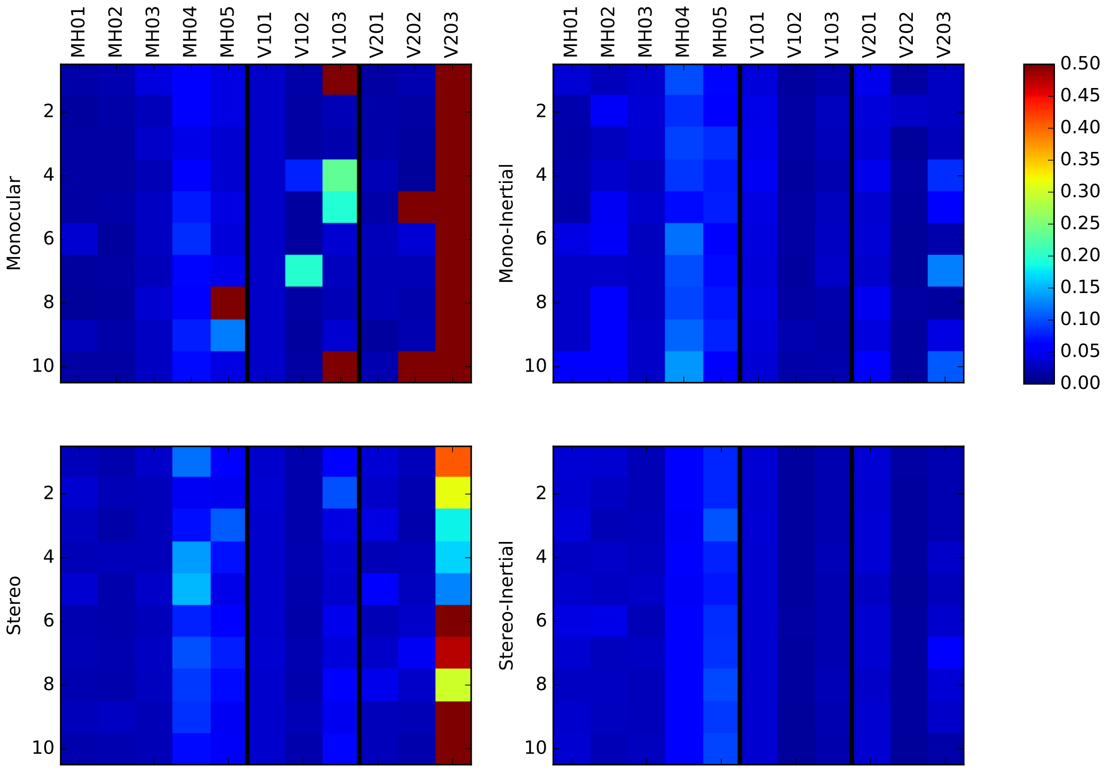


Figure 4.7: Colored squares represent the RMS ATE for ten different execution in each sequence of the EuRoC dataset.

which do not depend on the estimation and always relate the same variables, i.e. two consecutive keyframes. Second, it produces a more accurate frame estimation for tracking, where map points projection is performed, leading to a more repetitive set of visual measurements. These two points surprisingly reduce the system randomness as shown in figure 4.7. We can conclude that inertial integration not only boosts accuracy, reducing the median ATE error compared to pure visual solutions, but it also endows the system with excellent robustness, having a much more stable performance.

4.4.2 Visual-Inertial SLAM on TUM-VI Benchmark

The TUM-VI dataset [121] consists of 28 sequences in 6 different environments, recorded using a hand-held fisheye stereo-inertial rig. Ground-truth for the trajectory is only available at the beginning and at the end of the sequences, which for most of them represents a very small portion of the whole trajectory. Many sequences in the dataset do not contain loops. Even if the starting and ending point are in the same room, point of view directions are opposite and place recognition cannot detect any common region. Using this ground-truth for evaluation amounts to measuring the accumulated drift along the whole trajectory. In fact, the RMS ATE error computed after GT alignment is about half the accumulated drift for large exploratory datasets: slides, magistrales and outdoors.

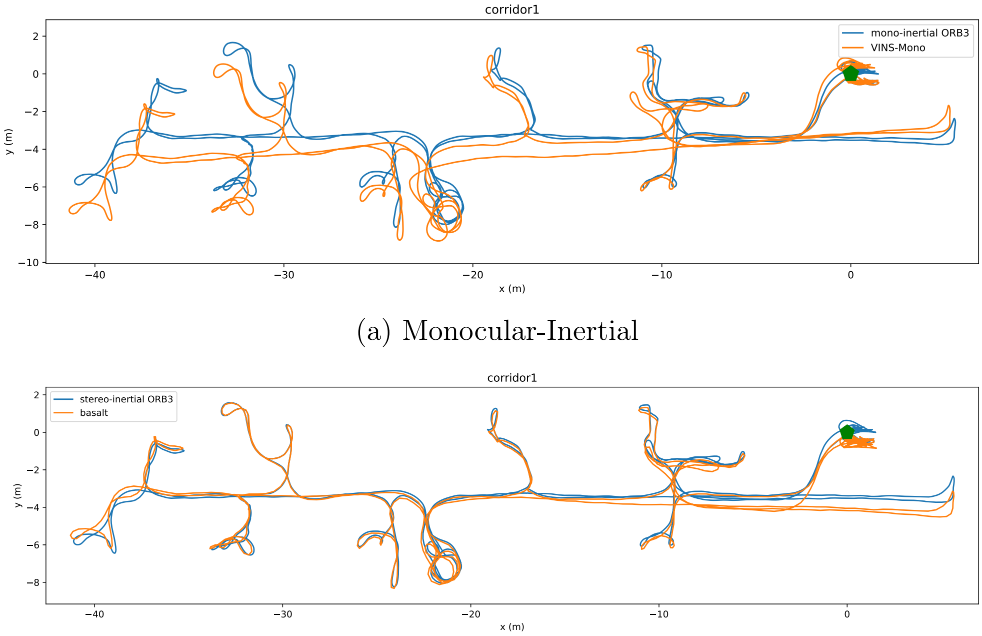
We extract 1500 ORB points per image in monocular-inertial setup, and 1000 points per image in stereo-inertial, after applying CLAHE equalization to address under and over exposure found in the dataset. For outdoors sequences, our system struggles with very far points coming from the cloudy sky, that is very visible in fisheye cameras. These points may have slow motion that can introduce drift in the camera pose. For preventing this, we discard points further than 20 meters from the current camera pose, only for outdoors sequences. A more sophisticated solution would be to use an image segmentation algorithm to detect and discard the sky.

Table 4.2: TUM VI Benchmark [121]: RMS ATE (m) for regions with available ground-truth data.

	Mono-Inertial		Stereo-Inertial					
Seq.	VINS-Mono	ORB-SLAM3	OKVIS	ROVIO	BASALT	ORB-SLAM3	Length (m)	LC
corridor1	0.63	**0.04**	0.33	0.47	0.34	**0.01**	305	✓
corridor2	0.95	**0.02**	0.47	0.75	0.42	**0.02**	322	✓
corridor3	1.56	**0.02**	0.57	0.85	0.35	**0.01**	300	✓
corridor4	0.25	**0.10**	0.26	0.13	0.21	**0.10**	114	
corridor5	0.77	**0.01**	0.39	2.09	0.37	**0.01**	270	✓
magistrale1	2.19	**0.40**	3.49	4.52	1.20	**1.14**	918	✓
magistrale2	3.11	**0.64**	2.73	13.43	1.11	**0.32**	561	✓
magistrale3	**0.40**	5.20	1.22	14.80	**0.74**	1.09	566	
magistrale4	5.12	**0.14**	0.77	39.73	1.58	**0.16**	688	✓
magistrale5	**0.85**	1.41	1.62	3.47	**0.60**	0.77	458	✓
magistrale6	2.29	**2.11**	3.91	X	3.23	**0.76**	771	
outdoors1	74.96	**60.65**	X	101.95	255.04	**33.9**	2656	
outdoors2	133.46	**14.24**	73.86	21.67	64.61	**14.14**	1601	
outdoors3	**36.99**	49.37*	32.38	**26.10**	38.26	48.18	1531	
outdoors4	**16.46**	19.02	19.51	X	17.53	**7.23**	928	
outdoors5	130.63	**17.71**	13.12	54.32	**7.89**	8.36	1168	✓
outdoors6	133.60	**14.51**	96.51	149.14	65.50	**8.31**	2045	
outdoors7	21.90	**9.18**	13.61	49.01	**4.07**	4.84	1748	✓
outdoors8	83.36	**25.26**	16.31	36.03	13.53	**11.02**	986	
room1	0.07	**0.01**	0.06	0.16	0.09	**0.01**	146	✓
room2	0.07	**0.02**	0.11	0.33	0.07	**0.01**	142	✓
room3	0.11	**0.01**	0.07	0.15	0.13	**0.01**	135	✓
room4	0.04	**0.01**	0.03	0.09	0.05	**0.01**	68	✓
room5	0.20	**0.04**	0.07	0.12	0.13	**0.01**	131	✓
room6	0.08	**0.01**	0.04	0.05	0.02	**0.01**	67	✓
slides1	**0.68**	0.85	0.86	13.73	**0.32**	0.50	289	
slides2	**0.84**	0.90	2.15	0.81	0.32	**0.31**	299	
slides3	**0.69**	1.02	2.58	4.68	0.89	**0.39**	383	

Ours are median of three executions.
For other systems, we provide values reported at [122]
* : One out of three runs has not been successful
LC: Loop Closing may exist in that sequence

The results obtained are compared with the most relevant systems in the literature in table 4.2, that clearly shows the superiority of ORB-SLAM3 both in monocular-inertial and stereo-inertial. The closest systems are VINS-Mono and BASALT, that are essentially visual-inertial odometry systems with loop closures, and miss mid-term data associations.



(a) Monocular-Inertial

(b) Stereo-Inertial

Figure 4.8: Comparative of ORB-SLAM3, VINS-Mono and Basalt at corridor1, a medium size indoors scenario. Starting (green pentagon) and ending points are in the same room. Trajectories have been aligned with the ground-truth at the starting point (green pentagon).

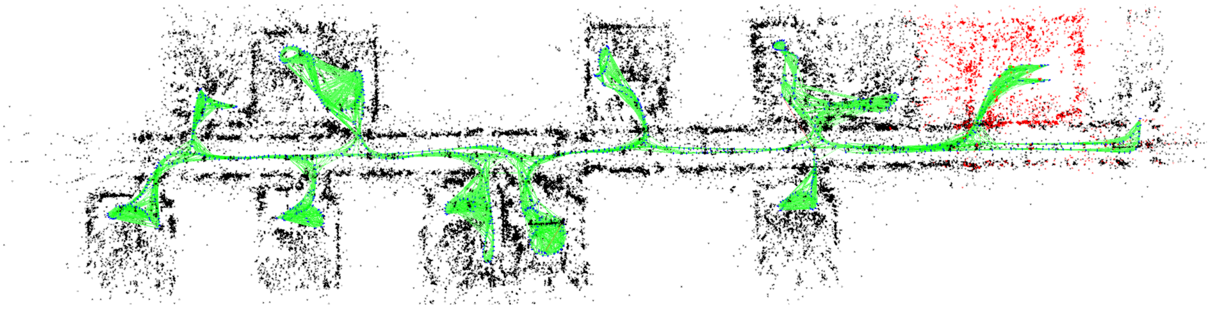
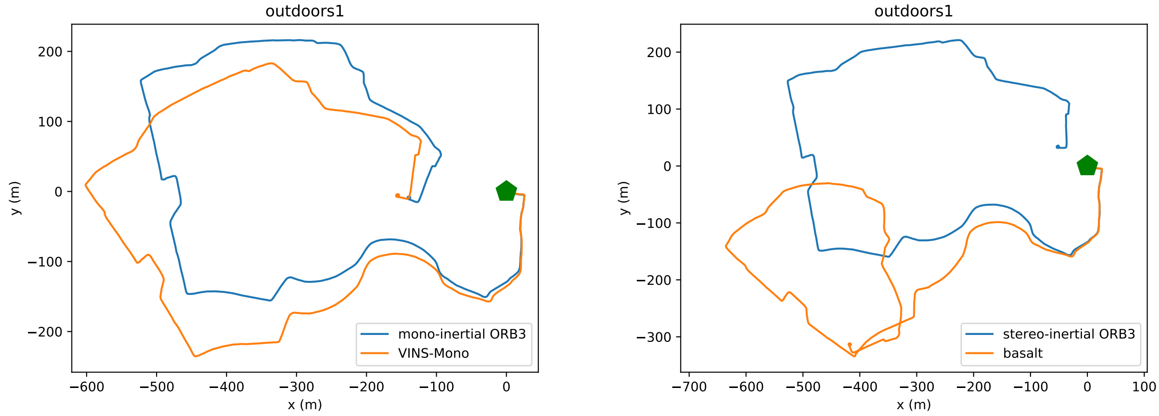


Figure 4.9: Top view map reconstruction, estimated trajectory and covisibility graph for our stereo-inertial solution at corridor1 TUM-VI sequence.



(a) Monocular-Inertial

(b) Stereo-Inertial

Figure 4.10: Comparative of ORB-SLAM3, VINS-Mono and Basalt at outdoors1 dataset, a 2.6 km long trajectory. Starting (green pentagon) and ending points are in the same room

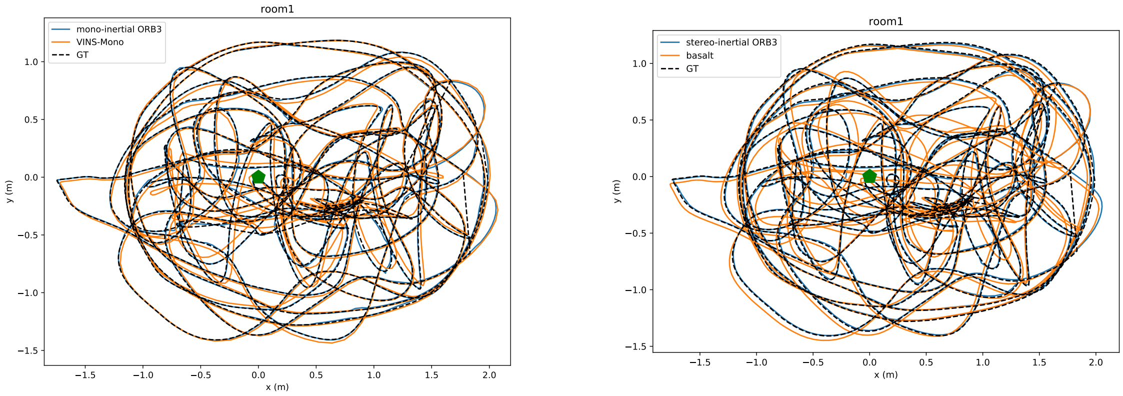
Analyzing more in detail the performance of our system, it gets lowest error in small and medium indoor environments, *room* and *corridor* sequences, with errors below 10 cm for most of them. In these trajectories, the system is continuously revisiting and reusing previously mapped regions, which is one of the main strengths of ORB-SLAM3. Also, tracked points are typically closer than 5 m, what makes easier to estimate inertial parameters, preventing them from diverging. This is also the reason why our monocular and stereo-inertial systems get very similar results at these sequences. Figure 4.8 shows the trajectories for the best monocular and stereo inertial systems in corridors1 dataset. Camera travels along corridor (x axis) while visiting different rooms along it, as shown in our reconstruction in figure 4.9. Both, VINS-Mono and Basalt struggle to detect mid-term associations, which leads to a duplicated corridor and a final drift of 0.6 m and 0.3 m respectively. In contrast, both ORB-SLAM3 inertial configurations have negligible drift, with errors below 0.04 m.

In *magistrale* indoors sequences, that are up to 900 m long, most tracked points are relatively close, and monocular-inertial ORB-SLAM3 obtains errors around 1 m except in one sequence that goes close to 5 m. Similarly, our stereo-inertial system gives errors around 1 m or less. In contrast, in some long *outdoors* sequences, the scarcity of close visual features may cause divergence of the inertial parameters, notably scale and accelerometer bias, which leads to errors in the order of 10 to 60 meters for our monocular-inertial SLAM. This is evident from figure 4.10a, and affects also VINS-Mono, as it was mentioned in results from [121]. Using a stereo camera in these situations provides a big improvement, since it constraints the scale estimation and avoids accelerometer bias drift. For ORB-SLAM3, stereo reduces the error by 50% on average, in contrast with other stereo systems where error remains high. We conclude that ORB-SLAM3 is the best performing system in the outdoor sequences for both configurations.

This dataset also contains three really challenging *slides* sequences, where the user descends though a dark tubular slide with almost total lack of visual features. In this situation, a pure visual system, at least ORB-SLAM, would be lost, but our visual-inertial system is able to process the whole sequence with competitive error, even if no loop-closures can be detected. Interestingly, VINS-Mono and BASALT, that track

Table 4.3: RMS ATE (m) obtained by ORB-SLAM3 with four sensor configurations in the room sequences, representative of AR/VR scenarios (median of 3 executions).

Seq.	Mono	Stereo	Mono-Inertial	Stereo-Inertial
room1	0.042	0.077	0.009	0.008
room2	0.026	0.055	0.018	0.012
room3	0.028	0.076	0.008	0.011
room4	0.046	0.071	0.009	0.008
room5	0.046	0.066	0.036	0.010
room6	0.043	0.063	0.006	0.006
Avg.	0.039	0.068	0.014	0.009



(a) Monocular-Inertial

(b) Stereo-Inertial

Figure 4.11: Comparative of ORB-SLAM3, VINS-Mono and Basalt at room1 dataset, a typical indoors AR/VR scenario.

features using Lucas-Kanade, obtain in some of these sequences better accuracy than ORB-SLAM3, that matches ORB descriptors. This could be pointing out that Lucas-Kanade successes to track some features while we are not able to match ORB points (see video).

Finally, the *room* sequences can be representative of typical AR/VR applications, where the user moves with a hand-held or head-mounted device in a small environment. For these sequences ground-truth is available for the entire trajectory. Table 4.2 and figure 4.11 show that ORB-SLAM3 is significantly more accurate that competing approaches in both configurations. The results obtained using our four sensors configurations are compared in table 4.3. The better accuracy of pure monocular compared with stereo is only apparent: the monocular solution is up-to-scale and is aligned with ground-truth with 7DoF, while stereo provides the true scale, and is aligned with 6DoF. Using monocular-inertial, we further reduce the average RMS ATE error below 2 cm, also obtaining the true scale. Finally, our stereo-inertial SLAM brings error below 1 cm, making it an excellent choice for AR/VR applications.

We have also performed some multi-session experiments on the TUM-VI dataset. Figure 4.12 shows the result after processing several sequences inside the TUM building[1].

[1] Videos of this and other experiments can be found at `https://www.youtube.com/channel/`

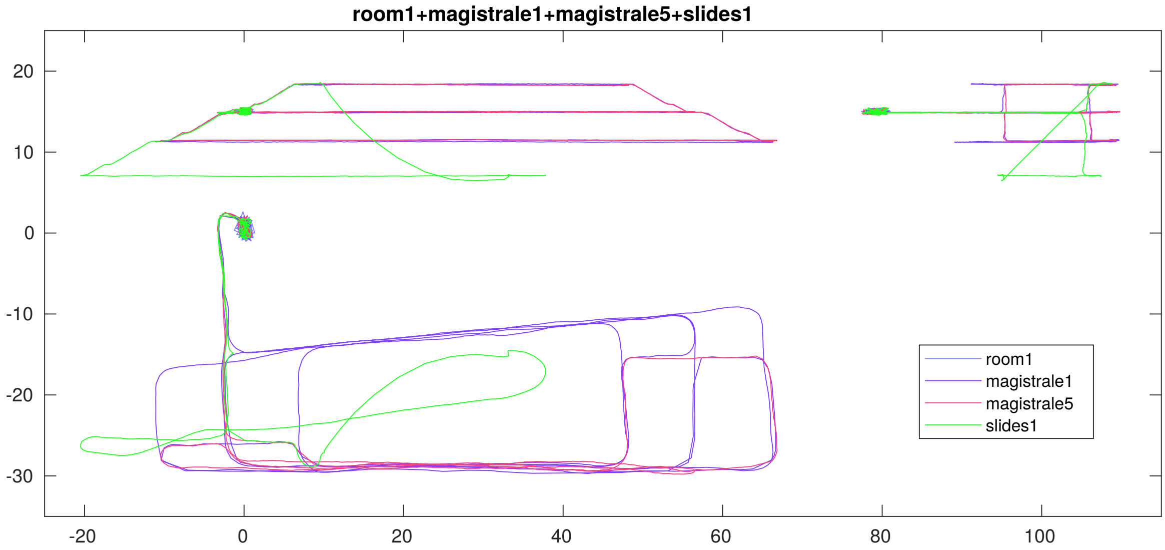


Figure 4.12: Multi-session stereo-inertial result with several sequences from TUM-VI dataset (front, side and top views).

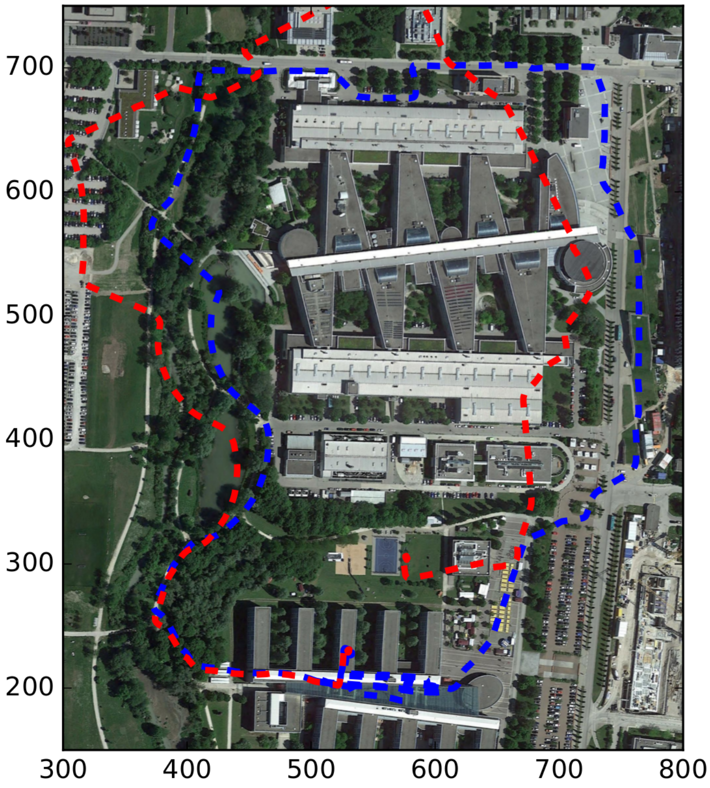


Figure 4.13: Multi-session stereo-inertial. In red, the trajectory estimated after single-session processing of outdoors1. In blue, multi-session processing of magistrale2 first, and then outdoors1.

In this case, the small *room* sequence provides loop closures that were missing in the longer sequences, bringing all errors to centimeter level. Although ground truth is not available outside the *room*, comparing the figure with the figures published in [122] clearly shows our point: our multi-session SLAM system obtains far better accuracy that existing visual-inertial odometry systems. This is further exemplified in Figure 4.13. Although ORB-SLAM3 ranks higher in stereo inertial single-session processing of outdoors1, there is still a significant drift (≈ 60 m). In contrast, if outdoors1 is processed after magistrale2 in a multi-session manner, this drift is significantly reduced, and the final map is much more accurate.

4.4.3 Computing Time

Table 4.4 summarizes the running time of the main operations performed in the tracking and mapping threads, showing that our system is able to run in real time at 50 frames and 3-4 keyframes per second for monocular configurations. Tracking for stereo configurations, including stereo rectification for EuRoC dataset, can process 30 frames and 7 keyframes per second. The inertial part takes negligible time during tracking and, in fact can render the system more efficient as the frame rate could be safely reduced. In the mapping thread, the higher number of variables per keyframe has been compensated with a smaller number of keyframes in the inertial local BA, achieving better accuracy, with a slightly lower running time. As the tracking and mapping threads work always in the active map, multi-mapping does not introduce significant overhead. ORB-SLAM3 in all its configurations largely works on real-time for common camera frame rates (<30 Hz). In blue, we have highlighted time reductions in ORB-SLAM3 v1.0, with respect to times published in [22] that were obtained with previous version ORB-SLAM3 v0.4 beta. We can observe a consistent reduction in BA cost for all sensors configurations. For pure visual SLAM this reduction is around 10%, while for visual-inertial configurations it goes from 20% to 30%. The major change between both versions was removing dependencies on opencv matrices in pose and map point representation, avoiding castings and reducing dynamic memory allocations. Poses are now represented using *Sophus*[2] library, while all algebraic computations are performed using *Eigen*[3] matrices.

Although it would be interesting, we do not compare computational time against other systems, since this would require a lot of work which is beyond the scope of this thesis. For a detailed timing analysis of some previous methods, we refer readers to [37].

4.5 Discussion

Building on ORB-SLAM2 [97] and ORB-SLAM-VI [98], we have presented ORB-SLAM3, the most complete open-source library for visual, visual-inertial and multi-session SLAM, with monocular, stereo, RGB-D, pin-hole and fisheye cameras. Our main contributions, apart from the integrated library itself [20], are the fast and accurate IMU initialization technique, as well as the multi-session map-merging functions.

Our experimental results show that ORB-SLAM3 is the first visual and visual-inertial system capable of effectively exploiting short-term, mid-term, long-term and multi-map

UCXVt-kXG6T95Z4tVaYlU8OQ

[2] https://github.com/strasdat/Sophus

[3] https://eigen.tuxfamily.org

Table 4.4: Running time of the main parts of our tracking and mapping threads compared to ORB-SLAM2, on EuRoC V202 (mean time and standard deviation in ms).

	System	ORB-SLAM2	ORB-SLAM3	ORB-SLAM3	ORB-SLAM3	ORB-SLAM3
Settings	Sensor	Stereo	Monocular	Stereo	Mono-Inertial	Stereo-Inertial
	Resolution	752×480	752×480	752×480	752×480	752×480
	Cam. FPS	20Hz	20Hz	20Hz	20Hz	20Hz
	IMU	-	-	-	200Hz	200HZ
	ORB Feat.	1200	1000	1200	1000	1200
	RMS ATE	0.035	0.029	0.028	0.021	0.014
Tracking	Stereo rect.	3.07±0.80	-	1.22±0.30	-	1.23±0.20
	ORB extract	11.20±2.00	11.24±5.04	11.25±3.56	10.25±3.69	10.91±2.38
	Stereo match	10.38±2.57	-	11.41±3.12	-	11.10±2.66
	IMU integr.	-	-	-	0.09±0.05	0.14±0.07
	Pose pred	2.20±0.72	1.36±0.50	1.95±0.54	0.07±0.34	0.10±0.45
	LM Track	9.89±4.95	4.78±2.07	5.23±2.32	6.86±2.35	7.74±1.92
	New KF dec	0.20±0.43	0.04±0.02	0.09±0.12	0.04±0.02	0.12±0.11
	Total	37.87±7.49	19.18±7.34	32.57±6.35	19.14±11.33	31.45±6.52
Mapping	KF Insert	8.72±3.60	9.25±4.62	6.40±2.02	12.02±7.23	6.94±2.04
	MP Culling	0.25±0.09	0.09±0.04	0.27±0.11	0.07±0.03	0.16±0.05
	MP Creation	36.88±14.53	21.94±9.87	14.49±6.74	30.35±12.11	15.59±5.62
	LBA	139.61±124.92	193.75±214.32	116.48±109.19	98.00±38.23	101.843±26.27
	KF Culling	4.37±4.73	15.34±13.04	6.43±4.63	25.73±16.87	9.23±6.10
	Total	173.81±139.07	238.11±233.47	140.25±118.30	166.38±71.90	134.04±37.10
Map Size	KFs	278	294	279	340	140
	MPs	14593	10804	12792	10946	10468

data associations, reaching an accuracy level that is beyond the reach of all studied existing systems. Our results also suggest that, regarding accuracy, the capability of using all these types of data association overpowers other choices such as using direct methods instead of features, or performing keyframe marginalization for local BA, instead of assuming an outer set of static keyframes as we do.

The main failure case of ORB-SLAM3 is low-texture environments. This may be partially overcome using the IMU for short periods of time. Direct methods are more robust to low-texture, but are limited to short-term [46] and mid-term [154] data association. On the other hand, matching feature descriptors successfully solves long-term and multi-map data association, but seems to be less robust for tracking than Lucas-Kanade, that uses photometric information. An interesting line of research could be developing photometric techniques adequate for the four data association problems.

About the four different sensor configurations, there is no question, stereo-inertial SLAM provides the most robust and accurate solution. Furthermore, the inertial sensor allows to estimate pose at IMU rate, which is orders of magnitude higher than frame rate, being a key feature for some use cases. For applications where a stereo camera is undesirable because of its higher bulk, cost, or processing requirements, you can use monocular-inertial without missing much in terms of robustness and accuracy. Only keep in mind that pure rotations during exploration would not allow to estimate depth.

In applications with slow motions, or without roll and pitch rotations, such as a car in a flat area, IMU sensors can be difficult to initialize. In those cases, if possible, use stereo SLAM. Otherwise, recent advances on depth estimation from a single image with CNNs offer good promise for reliable and true-scale monocular SLAM [145, 146], at least in the same type of environments where the CNN has been trained, as we will see in the next chapter.

Chapter 5

Deep Learning for SLAM

Most SLAM approaches assume a static scene and can only handle small fractions of dynamic content by labeling them as outliers to such static model. Whilst the static premise holds for some robotic applications, it limits its use in populated situations for autonomous driving, service robots or AR/VR. IMU, besides rendering observable scale and gravity direction, provides some robustness against dynamic objects since it allows to distinguish between ego motion and surrounding objects movement. However, there are situations where use of IMU is not possible or entails some problems. This is the case for autonomous-driving scenarios, where the vehicle does not perform a 6DoF motion, reducing the observability of the inertial parameters and hindering their initialization. For these IMU denied situations, recent deep learning based methods have emerged as an effective solution, being currently explored by the community.

In this chapter we demonstrate the application of deep learning techniques to SLAM problems which can not be solved by means of classical methods. First, in section 5.1, we present **DynaSLAM II**, a stereo and RGB-D SLAM system for dynamic environments, able to track and estimate dynamic objects. This is a joint work, where the contribution of this thesis is the formulation of a novel Bundle Adjustment for including dynamic objects. Second, in section 5.2, we present a **scale-aware direct monocular odometry**, based on neural network depth prediction. In contrast with existing solutions, we formulate a multi-view depth-photometric Bundle Adjustment with depth measurements from all observer keyframes as well as a truncated robust cost function which prevents from incorporating inconsistent depth observations. This significantly improves accuracy with respect to classical monocular methods, removing scale-drift and making scale observable.

5.1 Deep Learning for dynamic environments

5.1.1 Introduction

This introduction is joint work adapted from [9].

The problem of dealing with dynamic objects in SLAM has been widely targeted in recent years. The biggest part of the literature tackles this problem by detecting moving regions within the observed scene and rejecting such areas for the SLAM problem [6, 84, 130, 142]. Some works process the image streams outside of the localization pipeline by translating the images that show dynamic content into realistic images with only static content [7, 8, 10]. On the other hand, a small but growing part of the robotics community has addressed this issue by incorporating the dynamics of moving objects into the problem [64, 83, 147, 151]. Whereas the two first groups mostly focus on achieving an accurate ego-motion estimation from the static scene, the objective of the last group is twofold: they do not only solve the SLAM problem but also provide information about the poses and trajectories of other dynamic agents.

Understanding surrounding dynamic objects is of crucial importance for the frontier requirements of emerging applications within AR/VR or autonomous navigation. Whereas it is tolerable to rule out minor movements in quasi-static environments, most scenarios including autonomous driving, multi-robot collaboration and AR/VR require explicit motion information of the surroundings to aid in decision-making and scene understanding. For example, in VR, dynamic objects need to be explicitly tracked to allow the interaction of virtual objects with real-world moving instances. In autonomous driving scenarios, a car must not only localize itself but must also reliably perceive other vehicles and passers-by to avoid collisions.

The vast majority of the literature that specifically addresses this issue detects moving objects and tracks them separately from the SLAM formulation by using traditional multi-target tracking approaches [4, 112, 115, 138, 141]. Their accuracy highly depends on the camera pose estimation, which is susceptible to failure in complex dynamic environments where the presence of reliable static structure is not guaranteed. In recent years, the robotics community has made its first steps towards addressing object tracking jointly with SLAM, adding an extra layer of complexity to the problem. These systems are often tailored for special use cases and several priors are exploited to constraint the space solutions: road planar structure and planar object movement in driving scenarios [147], or even the use of object 3D models [62].

At this point, we take the opportunity to introduce DynaSLAM II. DynaSLAM II is an RGB-D and stereo SLAM system for dynamic scenes which simultaneously estimates the poses of the camera, the map and the trajectories of the scene moving objects. The contributions of this thesis to DynaSLAM II are:

- A cost-efficient bundle adjustment solution with new measurements between cameras, points and dynamic objects.
- A decoupled optimization for bounding boxes to find out a common reference across objects of the same class.
- Our experiments demonstrate that camera motion estimation and multi-object tracking can be mutually beneficial.

5.1.2 Related Work

This related work is joint work adapted from [9].

5.1.2.1 Loosely-Coupled Multi-Object Tracking and SLAM

The traditional manner of addressing 3D multi-object tracking implies detecting and tracking the moving objects separately from the SLAM formulation [4, 112, 115, 138, 141]. Among them, Wang et al. [138] derived the Bayes formula of the SLAM with tracking of moving objects and provided a solid basis for understanding and solving this problem. Wangsiripitak et al. [141] proposed the parallel implementation of SLAM with a 3D object tracker: the SLAM provides the tracker with information to register map objects, and the tracker allows to mark features on objects. Rogers et al. [112] applied an Expectation Maximization technique to a graph based SLAM approach and allowed landmarks to be dynamic. More recently, Barsan et al. [4] presented a stereo-based dense mapping algorithm for urban environments that simultaneously reconstructs the static background and the moving objects. There is a new work by Rosinol et al. [115] that reconciles visual-inertial SLAM and dense mesh tracking, focusing mostly on humans, that shows impressive results in simulation. The main drawback of these approaches is that their accuracy is highly correlated with that of the camera pose estimation. That is, if the camera pose estimation fails, which is quite likely in complex dynamic environments, multi-object tracking also directly fails.

The idea of simultaneously estimating camera motion and multiple moving objects motion originated from the SLAMMOT work [139]. They established a mathematical framework to integrate a filtering-based SLAM and moving object tracking demonstrating that it satisfied navigation and safety requirements in autonomous driving. Later on, works using RGB-D cameras followed up this idea to densely reconstruct static indoors scenes along with moving objects using pixel-wise instance segmentation, showing impressive results [118, 119, 143]. Since it is of crucial importance for dense approaches to obtain accurate segmentation, Mask-Fusion [119] and MID-Fusion [143] refine it by assuming that human-made objects are convex.

5.1.2.2 Tightly-Coupled Multi-Object Tracking and SLAM

Among the feature-based approaches, as is ours, few aim to merge information from static and dynamic objects into a single framework to boost estimation accuracy. Henein et al. [61] were among the first ones to tightly combine the problems of tracking dynamic objects and the camera ego motion. However, they only reported experiments on synthetic data showing limited real results. Li et al. [83] use a CNN trained in an end-to-end manner to estimate the 3D pose and dimensions of cars, which is further refined together with camera poses. The use of data-driven approaches often provides excellent accuracy in 6 DoF object pose estimation, but also a loss of generality and thus, they can only track cars. Huge amounts of data would be required to track generic objects with their approach. The authors of CubeSLAM [147] showed impressive results with only a monocular camera by making use of a 3D bounding box proposal generation based on 2D bounding boxes and vanishing points. They assume that objects have a constant velocity within a hard-coded duration time interval and exploit object priors such as car sizes, road structure and planar non-holonomic object wheel motion models. Moreover, they only track objects whose 3D bounding box is observable, *i.e.*, only once two or more faces of the *cuboid-shape* object are seen. On the other hand ClusterSLAM [63] proposes a SLAM back-end with no scene priors to discover individual rigid bodies and compute their motions in dynamic environments. Since it acts as a back end instead of as a full system, its performance relies heavily on the landmark tracking and association quality. The same authors recently developed the full system ClusterVO [64], which models the object points with a probability of object belonging to deal with segmentation inaccuracies. Given that they assume no priors, they obtain good tracking results in indoor and outdoor scenes, but with an inaccurate estimation of the 3D bounding boxes. VDO-SLAM [151] is a recent work that uses dense optical flow to maximise the number of tracked points on moving objects. They implement a bundle adjustment with cameras, objects and points that gives good results but is computationally complex to run in real time.

5.1.2.3 A Common Reference for each Object Class

Some approaches in the current literature do consider the tracking of dynamic objects to be complete with the tracking of dynamic feature points. Examples of these works are ClusterSLAM [63] and VDO-SLAM [151]. However, we believe that it is also of key importance to find a common spatial reference for objects of the same semantic class, as well as an estimate of their dimensions and space occupancy.

Alternatively, the basis of CubeSLAM [147] and of the work by Li et al. [83] is the discovery of object 3D bounding boxes. Only once bounding boxes are discovered, are objects tracked along frames. That is, if the camera viewing angle does not allow to estimate an object bounding box (partial view), the object tracking does not take place. Whereas this is not a problem for Li et al. [83] because CNNs are by nature robust to partial views of objects, CubeSLAM struggles to initialize bounding boxes from views of occluded objects.

In light of these advances, it is apparent that the feature-based SLAM community is searching for the best optimization formulation to combine cameras, objects and structure points. In our proposal, we use a tightly-coupled bundle adjustment formulation with new measurements between cameras, objects and points giving special attention to its computationally complexity and number of parameters involved without introducing hard-coded priors. For this, we integrate instance semantic priors together with sparse image features. This formulation allows the estimation of both the camera, the map structure and the dynamic objects to be mutually beneficial at a low computational cost. On the other hand, part of the current literature focuses on the estimation of the point cloud structure of dynamic objects and of the trajectory of a random object reference [63, 64, 151], whereas another part of the literature seeks to find a common reference for objects of the same class as well as a more informative occupancy volume [83, 147]. We intend to carry out these two tasks independently in order to leverage the benefits of both and not suffer their disadvantages.

5.1.3 Method

DynaSLAM II builds on the popular ORB-SLAM2 [97]. It takes synchronized and calibrated stereo/RGB-D images as input, and outputs the camera and the dynamic-object poses for each frame, as well as a spatial/temporal map containing the dynamic objects. For each incoming frame, pixel-wise semantic segmentation is computed and ORB features [117] are extracted and matched across stereo image pairs. We first associate the static and dynamic features with the ones from the previous frame and the map assuming a constant velocity motion for both the camera and the observed objects. Object instances are then matched based on the dynamic feature correspondences. The static matches are used to estimate the initial camera pose, and the dynamic ones yield the object's SE(3) transform. Finally, the camera and objects trajectories, as well as the objects bounding boxes and 3D points are optimized over a sliding window with marginalization and a soft smooth motion prior. The different contributions and building blocks of DynaSLAM II are explained in the following subsections.

5.1.3.1 Notation

We would like to point out that there are two types of dynamic objects in terms of their nature and influence on visual SLAM:

1. The former type consists of objects that inherently possess the capability to move (people, animals and vehicles). We name these objects *a priori dynamic* or *movable.* While the SLAM sensor observes them, they can either move or remain static.

2. The latter type consists though of objects that are moving while the visual SLAM sensor observes them regardless of their semantic class. We name them *moving* objects.

.

Regarding the scene geometry we will use the following notation: a stereo/RGB-D camera i has a pose $\mathbf{T}^{i}_{\mathtt{CW}} \in \mathrm{SE}(3)$ in the world coordinates $\mathbf{W}$ at time i (see Fig. 5.1). The camera i observes

1. static 3D map points, index l, $\mathbf{x}^{l}_{\mathtt{W}} \in \mathbb{R}^3$, and

2. dynamic objects, index k, with pose $\mathbf{T}^{k,i}_{\mathtt{WO}} \in \mathrm{SE}(3)$ and linear and angular velocity $\mathbf{v}^{k}_{i}, \mathbf{w}^{k}_{i} \in \mathbb{R}^3$ at time i, in the object reference frame.

Each observed object k contains dynamic-object points $\mathbf{x}^{j,k}_{\mathtt{O}} \in \mathbb{R}^3$.

5.1.3.2 Object Data Association

Object Data Association was mainly developed by Berta Bescós within this joint work. We keep this section in this thesis since it may be necessary to understand the following parts

For each incoming frame the below procedure is followed:

1. Pixel-wise semantic segmentation is computed and ORB features [117] are extracted and matched across stereo pairs. We hypothesize that dynamic features are those belonging to *a priori* dynamic instances, regardless of their motion.

2. We first associate the static features with the ones from the previous frame and the map to initially estimate the camera pose, following the ORB-SLAM implementation.

3. A parallel instance-to-instance matching between consecutive frames is built with the Munkres algorithm [95] using the 2D bounding boxes Intersection over Union as cost.

4. Next, dynamic features are associated with the dynamic points from the local map in two different ways:

   (a) if the map objects velocity is known, the matches are searched by reprojection assuming an inter-frame constant velocity motion. The instance matching results are used to discover outliers.

   (b) if the objects velocity is not initialized or not enough matches are found following (a), we constrain the brute force matching to those features belonging to the most overlapping instance in consecutive frames.

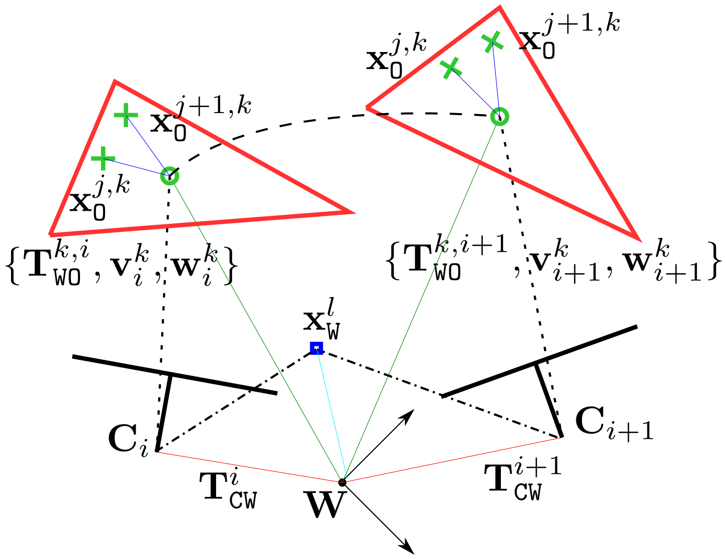

$\mathbf{x}_{\mathtt{O}}^{j,k}$
$\mathbf{x}_{\mathtt{O}}^{j+1,k}$
$\mathbf{x}_{\mathtt{O}}^{j+1,k}$
$\mathbf{x}_{\mathtt{O}}^{j,k}$
$\{\mathbf{T}_{\mathtt{WO}}^{k,i}, \mathbf{v}_i^k, \mathbf{w}_i^k\}$
$\{\mathbf{T}_{\mathtt{WO}}^{k,i+1}, \mathbf{v}_{i+1}^k, \mathbf{w}_{i+1}^k\}$
$\mathbf{x}_{\mathtt{W}}^l$
$\mathbf{C}_i$
$\mathbf{C}_{i+1}$
$\mathbf{T}_{\mathtt{CW}}^i$
$\mathbf{W}$
$\mathbf{T}_{\mathtt{CW}}^{i+1}$

Figure 5.1: Notation used to model the dynamic structure. The cameras i and $i+1$ observe the dynamic object k (···) and the static structure (-·-·-). The objects with poses $\mathbf{T}_{\mathtt{WO}}^{k,i}$ and $\mathbf{T}_{\mathtt{WO}}^{k,i+1}$ are the same moving body at consecutive observations (- - -).

5. A higher level association between instances and objects is also required. If most of the new instance key points are matched with points belonging to one same map object, the instance is attributed the same object track id.

6. If an instance corresponding to an *a priori* dynamic class contains more than seven unobserved key points whose stereo 3D projection is close to the camera (less than 55 times the stereo baseline), a new object instance is created. Key points are then assigned to the corresponding object.

Note that our framework handles occlusions that last less than two seconds. This threshold can be extended though if needed: our framework handles occlusions if the object velocity remains constant or almost constant. Because of this feature, extending the threshold might not always lead to better results if the velocity of the tracked object changes in this window.

The SE(3) pose of the first object of a track is initialized with the center of mass of the 3D points and with the identity rotation. To predict the poses of further objects from a track, we use a constant velocity motion model and refine the object pose estimate by minimizing the matches reprojection error.

The reprojection error formulation in multi-view geometry problems for a camera i with pose $\mathbf{T}_{\mathtt{CW}}^i \in \mathrm{SE}(3)$ and a 3D map point l with coordinates $\mathbf{x}_{\mathtt{W}}^l \in \mathbb{R}^3$ in reference $\mathtt{W}$ with a stereo key point correspondence $\mathbf{u}_i^l = [u, v, u_R] \in \mathbb{R}^3$ is:

$$\mathbf{e}_{\mathbf{repr}}^{i,l} = \mathbf{u}_i^l - \pi_i(\mathbf{T}_{\mathtt{CW}}^i \mathbf{x}_{\mathtt{W}}^l), \tag{5.1}$$

where π_i is the reprojection function for a rectified stereo/RGB-D camera that projects a 3D point in the camera coordinates into the camera frame pixel. Unlike this formulation, which is valid for static representations, we propose to restate the reprojection error as:

$$\mathbf{e}_{\mathbf{repr}}^{i,j,k} = \mathbf{u}_i^j - \pi_i(\mathbf{T}_{\mathtt{CW}}^i \mathbf{T}_{\mathtt{WO}}^{k,i} \mathbf{x}_{\mathtt{O}}^{j,k}), \tag{5.2}$$

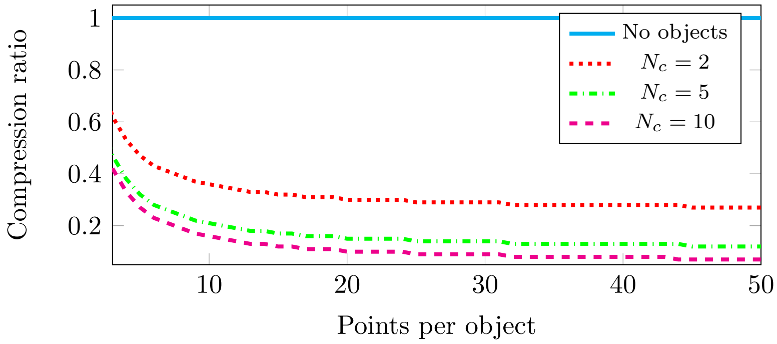


Figure 5.2: Relationship between the number of parameters that is required when objects are used and when points belonging to objects are tracked independently (No objects).

where $\mathbf{T}_{\mathtt{WO}}^{k,i} \in \mathrm{SE}(3)$ is the inverse pose of the object k in the world coordinates when the camera i is observing it, and $\mathbf{x}_{\mathtt{O}}^{j,k} \in \mathbb{R}^3$ represents the 3D coordinates of the point j in its object reference k with observation in the camera i $\mathbf{u}_i^j \in \mathbb{R}^3$. This formulation enables us to optimize either jointly the poses of the cameras and of the different moving objects, as well as the positions of their 3D points.

5.1.3.3 Object-Centric Representation

Given the extra complexity and mainly the extra number of parameters that the task of tracking moving objects implies on top of the SLAM ones, it is of high importance to keep this number as reduced as possible to maintain a real-time performance. Modeling dynamic points as repeated 3D points by forming independent point clouds as in usual dynamic SLAM implementations results in a prohibitive amount of parameters. Given a set of N_c cameras, N_o dynamic objects with N_{op} 3D points each observed in all cameras, the number of parameters needed to track dynamic objects becomes $N = 6N_c + N_c \times N_o \times 3N_{op}$ as opposed to $N = 6N_c + N_o \times 3N_{op}$ in conventional static SLAM representations. This number of parameters becomes prohibitive for long –and not so long– operations and deployment. If the concept of objects is introduced, 3D object points become unique and can be referred to their dynamic object. Therefore it is the pose of the object that is modelled along time and the number of required parameters shifts to $N' = 6N_c + N_c \times 6N_o + N_o \times 3N_{op}$. Fig. 5.2 shows the parameter compression ratio defined as $\frac{N'}{N}$ for 10 objects. This modelling of dynamic objects and points brings great savings in the number of utilized parameters.

5.1.3.4 Bundle Adjustment with Objects

Bundle Adjustment (BA) is known to provide accurate estimates of camera poses and sparse geometrical reconstruction, given a strong network of matches and good initial guesses. We hypothesize that BA might bring similar benefits if object poses are also jointly optimized (Fig. 5.3). Static map point 3D locations $\mathbf{x}_{\mathtt{W}}^l$ and camera poses $\mathbf{T}_{\mathtt{CW}}^i$ are optimized by minimizing the reprojection error with respect to the matched key points $\mathbf{u}_i^l$ (Eqn. 5.1). Similarly, for dynamic representations, object points $\mathbf{x}_{\mathtt{O}}^{j,k}$, camera poses $\mathbf{T}_{\mathtt{CW}}^i$ and object poses $\mathbf{T}_{\mathtt{WO}}^{k,i}$ can be refined by minimizing the reprojection error formulation in Eqn. 5.2.

In our implementation, a keyframe can be inserted in the map for two different reasons:

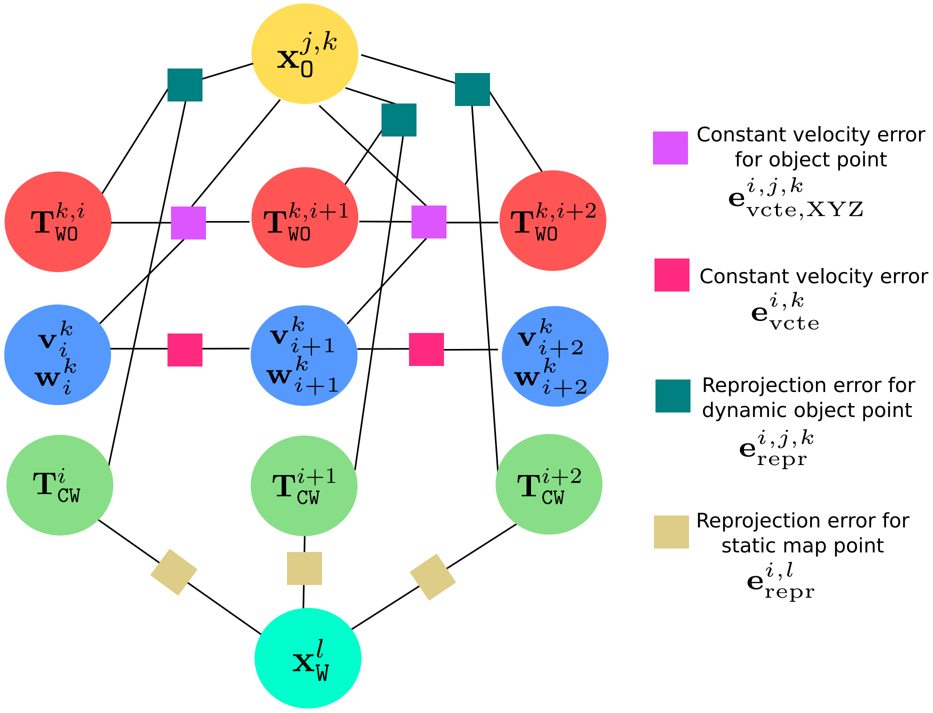

$\mathbf{x}_{\mathtt{O}}^{j,k}$
$\mathbf{T}_{\mathtt{WO}}^{k,i}$
$\mathbf{T}_{\mathtt{WO}}^{k,i+1}$
$\mathbf{T}_{\mathtt{WO}}^{k,i+2}$
$\mathbf{v}_i^k$ $\mathbf{w}_i^k$
$\mathbf{v}_{i+1}^k$ $\mathbf{w}_{i+1}^k$
$\mathbf{v}_{i+2}^k$ $\mathbf{w}_{i+2}^k$
$\mathbf{T}_{\mathtt{CW}}^{i}$
$\mathbf{T}_{\mathtt{CW}}^{i+1}$
$\mathbf{T}_{\mathtt{CW}}^{i+2}$
$\mathbf{x}_{\mathtt{W}}^{l}$
Constant velocity error for object point $\mathbf{e}_{\text{vcte,XYZ}}^{i,j,k}$
Constant velocity error $\mathbf{e}_{\text{vcte}}^{i,k}$
Reprojection error for dynamic object point $\mathbf{e}_{\text{repr}}^{i,j,k}$
Reprojection error for static map point $\mathbf{e}_{\text{repr}}^{i,l}$

Figure 5.3: BA factor graph representation with dynamic objects.

1. the camera tracking is weak, which is measured with the number of tracked static points.

2. the tracking of any scene object is weak, which is measured with the number of tracked points from the given object.

The reasons for the former are the same ones than in ORB-SLAM. The latter though happens if a dynamic instance with a relatively large amount of features has few points tracked in the current frame. The following optimization scenarios can then appear:

- If a keyframe is inserted only because the camera tracking is weak, the local BA optimizes the currently processed keyframe, all the keyframes connected to it in the covisibility graph, and all the map points seen by those keyframes, following the implementation of ORB-SLAM.

- If a keyframe is inserted only because an instance tracking is weak, a new object with new object points is created. This keyframe does not introduce new static structure, and if the rest of dynamic objects have a stable tracking, new objects for these tracks are not created. In such case the local BA optimizes the pose, velocity and object points of this object and the camera along a 2 seconds temporal tail.

- Finally, if a keyframe is inserted because both camera and object tracking is weak, camera poses, map structure, object poses, velocities and points are jointly optimized.

To avoid non-physically feasible object dynamics, a smooth trajectory is forced by assuming a constant velocity in consecutive observations. The linear and angular velocity of an object k at observation i are respectively denoted as $\mathbf{v}_i^k \in \mathbb{R}^3$ and $\mathbf{w}_i^k \in \mathbb{R}^3$. We define the following error term:

$$\mathbf{e}_{\mathbf{vcte}}^{i,k} = \begin{pmatrix} \mathbf{v}_{i+1}^k - \mathbf{v}_i^k \\ \mathbf{w}_{i+1}^k - \mathbf{w}_i^k \end{pmatrix} \tag{5.3}$$

An additional error term is needed to couple the object velocities and poses with their corresponding 3D points. This term can be seen in Eqn. 5.4, where $\Delta\mathbf{T}_{\mathtt{O}_k}^{i,i+1}$ is the pose transformation that the object k undergoes in the time interval $\Delta t_{i,i+1}$ between consecutive observations i and $i+1$:

$$\mathbf{e}_{\mathbf{vcte},\mathbf{XYZ}}^{i,j,k} = \left(\mathbf{T}_{\mathtt{WO}}^{k,i+1} - \mathbf{T}_{\mathtt{WO}}^{k,i}\Delta\mathbf{T}_{\mathtt{O}_k}^{i,i+1}\right)\mathbf{x}_{\mathtt{O}}^{j,k} \tag{5.4}$$

The term $\Delta\mathbf{T}_{\mathtt{O}_k}^{i,i+1}$ is defined from the linear and angular velocity of the object k at time i ($\mathbf{v}_i^k$ and $\mathbf{w}_i^k$) as in Eqn. 5.5, where we use the exponential map $\mathrm{Exp} : \mathbb{R}^3 \rightarrow \mathrm{SO}(3)$ as defined in appendix section D.2:

$$\Delta\mathbf{T}_{\mathtt{O}_k}^{i,i+1} = \begin{pmatrix} \mathrm{Exp}(\mathbf{w}_i^k \Delta t_{i,i+1}) & \mathbf{v}_i^k \Delta t_{i,i+1} \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix} \tag{5.5}$$

The derivative of the term $\mathbf{e}_{\mathbf{vcte},\mathbf{XYZ}}^{i,j,k}$ in Eqn. 5.4 with respect to the angular velocity is approximated as follows. Let us apply an additive perturbation $\delta\mathbf{w}_i^k$ to the object angular velocity, such that:

$$\begin{aligned}
\frac{\partial \mathbf{e}_{\mathbf{vcte},\mathbf{XYZ}}^{i,j,k}}{\partial \delta\mathbf{w}_i^k} &= -\frac{\partial}{\partial \delta\mathbf{w}_i^k}\left[\begin{pmatrix} \mathbf{R}_{\mathtt{WO}}^{k,i}\mathrm{Exp}((\mathbf{w}_i^k + \delta\mathbf{w}_i^k)\Delta t_{i,i+1}) & \mathbf{R}_{\mathtt{WO}}^{k,i}\mathbf{v}_i^k \Delta t_{i,i+1} + \mathbf{t}_{\mathtt{WO}}^{k,i} \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix}\mathbf{x}_{\mathtt{O}}^{j,k}\right] && (5.6)\\
&= -\mathbf{R}_{\mathtt{WO}}^{k,i}\frac{\partial}{\partial \delta\mathbf{w}_i^k}\left[\mathrm{Exp}((\mathbf{w}_i^k + \delta\mathbf{w}_i^k)\Delta t_{i,i+1})\mathbf{x}_{\mathtt{O}}^{j,k}\right] && (5.7)\\
&\approx -\mathbf{R}_{\mathtt{WO}}^{k,i}\frac{\partial}{\partial \delta\mathbf{w}_i^k}\left[\mathrm{Exp}(\mathbf{w}_i^k \Delta t)\mathrm{Exp}(\mathbf{J}_r(\mathbf{w}_i^k \Delta t)\delta\mathbf{w}_i^k \Delta t)\mathbf{x}_{\mathtt{O}}^{j,k}\right] && (5.8)
\end{aligned}$$

where we have used the first order approximation, where the term $\mathbf{J}_r$ is the right Jacobian of SO(3) (see Appendix section D.2.6) and relates additive increments in the tangent space to multiplicative increments applied on the right-hand side. Assuming that the term $\mathbf{J}_r(\mathbf{w}_i^k \Delta t)\delta\mathbf{w}_i^k \Delta t$ is small, we can rewrite the second term as:

$$\mathrm{Exp}(\mathbf{w}_i^k \Delta t)\mathrm{Exp}(\mathbf{J}_r(\mathbf{w}_i^k \Delta t)\delta\mathbf{w}_i^k \Delta t) \approx \mathrm{Exp}(\mathbf{w}_i^k \Delta t)(\mathbf{I} + [\mathbf{J}_r(\mathbf{w}_i^k \Delta t)\delta\mathbf{w}_i^k \Delta t]_\times), \tag{5.9}$$

where $[\cdot]_\times$ is the skew operator that transforms a vector in $\mathbb{R}^3$ into a skew symmetric matrix. Derivative takes next form:

$$\frac{\partial \mathbf{e}_{\mathbf{vcte},\mathbf{XYZ}}^{i,j,k}}{\partial \delta\mathbf{w}_i^k} \approx -\mathbf{R}_{\mathtt{WO}}^{k,i}\mathrm{Exp}(\mathbf{w}_i^k \Delta t)\Delta t \frac{\partial}{\partial \delta\mathbf{w}_i^k}\left([\mathbf{J}_r(\mathbf{w}_i^k \Delta t)\delta\mathbf{w}_i^k]_\times \mathbf{x}_{\mathtt{O}}^{j,k}\right) \tag{5.10}$$

Skew matrix multiplication by a vector is equivalent to cross product and we know derivative of cross product can be expanded as:

$$\frac{d}{dt}(\mathbf{a}\times\mathbf{b}) = \frac{d\mathbf{a}}{dt}\times\mathbf{b} + \mathbf{a}\times\frac{d\mathbf{b}}{dt} \tag{5.11}$$

In our case, this leads to:

$$\frac{\partial \mathbf{e}^{i,j,k}_{\mathbf{vcte,XYZ}}}{\partial \delta \mathbf{w}^k_i} = -\mathbf{R}^{k,i}_{\mathtt{WO}} \mathrm{Exp}(\mathbf{w}^k_i \Delta t)\Delta t \left([\mathbf{J}_r]_{col1} \times \mathbf{x}^{j,k}_{\mathtt{O}}, [\mathbf{J}_r]_{col2} \times \mathbf{x}^{j,k}_{\mathtt{O}}, [\mathbf{J}_r]_{col3} \times \mathbf{x}^{j,k}_{\mathtt{O}} \right) \tag{5.12}$$

Derivatves with respect to linear velocity can be straightforward derived. Coming back to the general optimization problem, we define the following BA for a set of cameras in the optimizable local window $\mathcal{K}$ with each camera i observing a set of map points $\mathcal{MP}_i$ and an object set $\mathcal{O}_i$ containing each object k the set of object points $\mathcal{OP}_k$:

$$\begin{aligned} \min_{\theta} \sum_{i \in \mathcal{K}} (\sum_{l \in \mathcal{MP}_i} \rho \left(\|\mathbf{e}^{i,l}_{\mathbf{repr}}\|^2_{\Sigma^l_i} \right) + \sum_{k \in \mathcal{O}_i} (\rho \left(\|\mathbf{e}^{i,k}_{\mathbf{vcte}}\|^2_{\Sigma_{\Delta t}} \right) \\ + \sum_{j \in \mathcal{OP}_k} (\rho (\|\mathbf{e}^{i,j,k}_{\mathbf{repr}}\|^2_{\Sigma^j_i}) + \rho \left(\|\mathbf{e}^{i,j,k}_{\mathbf{vcte,XYZ}}\|^2_{\Sigma_{\Delta t}} \right)))), \end{aligned} \tag{5.13}$$

where ρ is the robust Huber cost function to downweigh outlier correspondences and Σ is the covariance matrix. In the case of the reprojection error Σ is associated to the scale of the key point in the camera i observing the points l and j respectively. For the two other error terms Σ is associated to the time interval between two consecutive observations of an object, *i.e.*, the longer time the more uncertainty there is about the constant velocity assumption. The parameters to be optimized are $\theta = \{\mathbf{T}^i_{\mathtt{CW}}, \mathbf{T}^{k,i}_{\mathtt{WO}}, \mathbf{x}^l_{\mathtt{W}}, \mathbf{x}^{j,k}_{\mathtt{O}}, \mathbf{v}^k_i, \mathbf{w}^k_i\}$.

Fig. 5.4 shows the boolean Hessian matrix ($\mathbf{H}$) of the problem described. The Hessian can be built from the Jacobian matrices associated to each edge in the factor graph. In order to have a non-zero (i, j) block matrix, there must be an edge between i and j node in the factor graph. Notice the difference in the sparsity patterns of the map points and the object points. The size of the Hessian matrix is dominated by the number of map points N_{mp} and object points, which in typical problems is several orders of magnitude larger than the number of cameras and objects. Applying Schur complement trick and solving the system has a run-time complexity of $\mathcal{O}(N_c^3 + N_c^2 N_{mp} + N_c N_o N_{op})$, where either the second or third term will dominate the cost depending on the number of static and dynamic points.

5.1.3.5 Bounding Boxes

We propose to decouple the estimation of the trajectories and the bounding boxes of the dynamic objects. The former provides the system tracking with rich clues for ego-motion estimation, and the conjunction of both are useful to understand the dynamics of the surroundings. The output of the data association and the BA stages contains the camera poses, the structure of the static scene and the dynamic objects, and the 6 DoF trajectory of one point for each object. This one point is the center of mass of the object 3D points when it is first observed. Even though the center of mass changes along time with new points observations, the object pose that is tracked and optimized is referred to this first center of mass. To have a full understanding of the moving surroundings, it is of high importance to know the objects dimensions and space occupancy. Tackling the two problems independently allows to track dynamic objects from the first frame in which they appear independently of the camera-object view point.

We initialize an object bounding box by searching two perpendicular planes that fit roughly the majority of the object points. We hypothesize that many man-made objects can approximately fit a 3D bounding box. In the case in which only one plane is found,

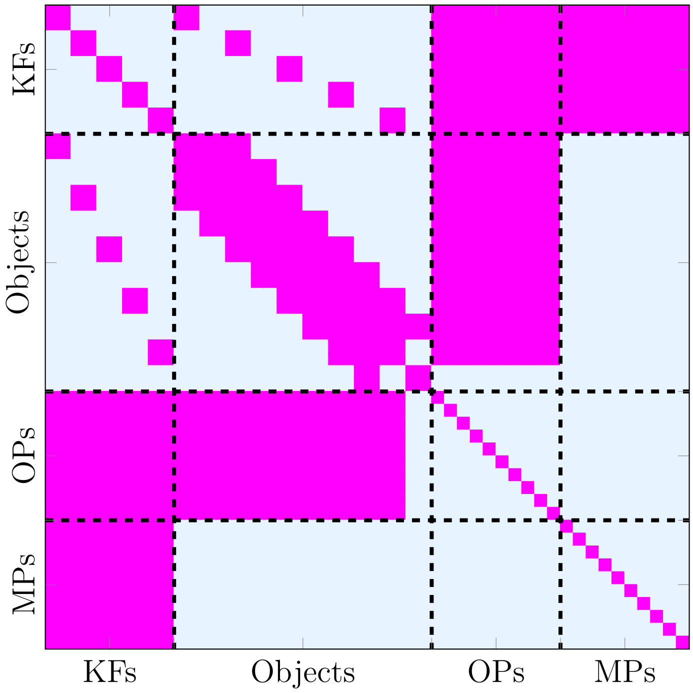


Figure 5.4: Hessian matrix for 5 keyframes (KFs), 1 object with 10 object points (OPs) and 10 static map points (MPs).

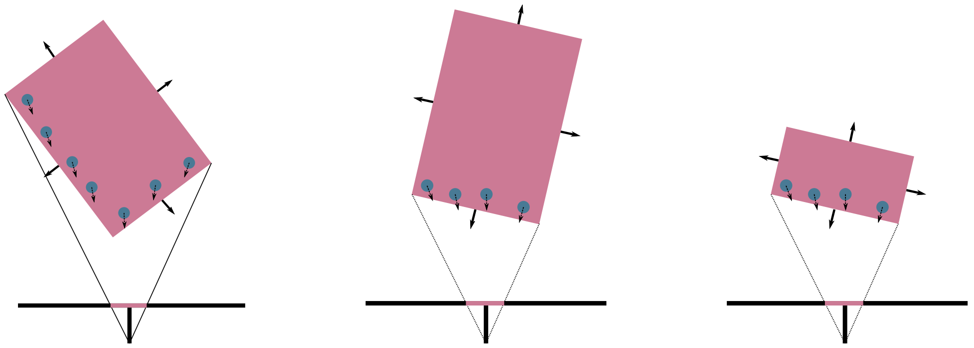
we add a prior on the rough dimensions of the non-observable direction that is related to the object class. This procedure is done within a RANSAC scheme: we choose the computed 3D bounding box that has the largest IoU of its image projection with the CNN 2D bounding box. This bounding box is computed once for every object track.

To refine the bounding box dimensions and its pose relative to the object tracking reference, an image-based optimization is performed within a temporal window. This optimization seeks to minimize the distance between the 3D bounding box image projection and the CNN 2D bounding box prediction. Also, to constraint the solution space in case the view of an object makes this problem non-observable (*e.g.*, a car observed from the back), a soft prior about the object dimensions is included. Since this prior is tightly related to the object class, we believe that adding this soft prior does not mean a loss of generality. Finally, the initial bounding box pose is set as a prior so that the optimization solution remains close.

One can see in Fig. 5.5 the effect of the different errors and priors we use in our optimization. First, all three objects (Figs. 5.5a, 5.5b and 5.5c) yield the same 2D image projection so, unless an object is observed in at least three frames at different view points, more constraints are required to render the problem observable. Second, forcing the 3D points to be near the found planes constraints most cases. Third, a prior about the object's dimensions is needed, otherwise, cases such as the ones in Figs. 5.5b and 5.5c are not fully constrained.

5.1.4 Experiments

In this section we detail the experiments carried out to test DynaSLAM II. It is divided in two main blocks: one that assesses the effect of tracking objects on the estimation of camera motion (section 5.1.4.1), and one that analyzes the multi-object tracking performance



(a) (b) (c)

Figure 5.5: Examples of camera and bounding box configurations.

seq	ORB-SLAM2 [97]			DynaSLAM [6]			VDO-SLAM [151]			Ours		
	ATE	RPE$_\mathrm{t}$	RPE$_\mathrm{R}$	ATE	RPE$_\mathrm{t}$	RPE$_\mathrm{R}$	ATE	RPE$_\mathrm{t}$	RPE$_\mathrm{R}$	ATE	RPE$_\mathrm{t}$	RPE$_\mathrm{R}$
	(m)	(m/f)	$(^\circ/f)$	(m)	(m/f)	$(^\circ/f)$	(m)	(m/f)	$(^\circ/f)$	(m)	(m/f)	$(^\circ/f)$
0000	1.32	**0.04**	0.06	1.35	**0.04**	0.06	-	0.05	**0.05**	**1.29**	**0.04**	0.06
0001	**1.95**	**0.05**	0.04	2.42	**0.05**	0.04	-	0.12	0.04	2.31	**0.05**	0.04
0002	0.95	0.04	0.03	1.04	0.04	0.03	-	0.04	**0.02**	**0.91**	0.04	**0.02**
0003	0.74	0.07	0.04	0.78	0.07	0.04	-	0.09	0.04	**0.69**	**0.06**	0.04
0004	1.44	**0.07**	0.06	1.52	**0.07**	0.06	-	0.11	**0.05**	**1.42**	**0.07**	0.06
0005	1.23	**0.06**	0.03	**1.22**	**0.06**	0.03	-	0.10	**0.02**	1.34	**0.06**	0.03
0006	0.19	0.02	**0.04**	0.19	0.02	**0.04**	-	0.02	0.05	0.19	0.02	**0.04**
0007	**2.47**	0.05	0.07	2.69	0.05	0.07	-	-	-	3.10	0.05	0.07
0008	1.40	**0.08**	0.04	**1.29**	**0.08**	0.04	-	-	-	1.68	0.10	0.04
0009	4.00	0.06	**0.05**	**3.55**	0.06	**0.05**	-	-	-	5.02	0.06	0.06
0010	1.68	0.07	0.04	1.84	0.07	0.04	-	-	-	**1.30**	0.07	**0.03**
0011	**0.97**	0.04	0.03	1.05	0.04	0.03	-	-	-	1.03	0.04	0.03
0013	1.18	0.04	0.05	1.18	0.04	0.05	-	-	-	**1.10**	0.04	**0.04**
0014	0.13	0.03	0.08	0.13	0.03	0.08	-	-	-	**0.12**	0.03	0.08
0018	**0.89**	**0.05**	0.03	1.00	**0.05**	0.03	-	0.07	**0.02**	1.09	**0.05**	**0.02**
0019	2.31	0.05	0.03	2.35	0.05	0.03	-	-	-	**2.25**	0.05	0.03
0020	16.80	0.11	0.07	**1.10**	**0.05**	0.04	-	0.16	**0.03**	1.36	0.07	0.04
mean	2.33	0.055	0.046	**1.45**	**0.051**	0.045	-	0.084	**0.036**	1.54	0.053	0.043

Table 5.1: Egomotion comparison on the KITTI tracking dataset. Results of sequences without egomotion are not shown.

(section 5.1.4.2).

5.1.4.1 Visual Odometry

For the visual odometry experiments we have chosen the KITTI tracking (Table 5.1) and raw (Table 5.2) datasets [57]. They contain several gray-scale and RGB stereo sequences of urban and road scenes recorded from a car perspective with circulating vehicles and pedestrians, as well as its GPS data.

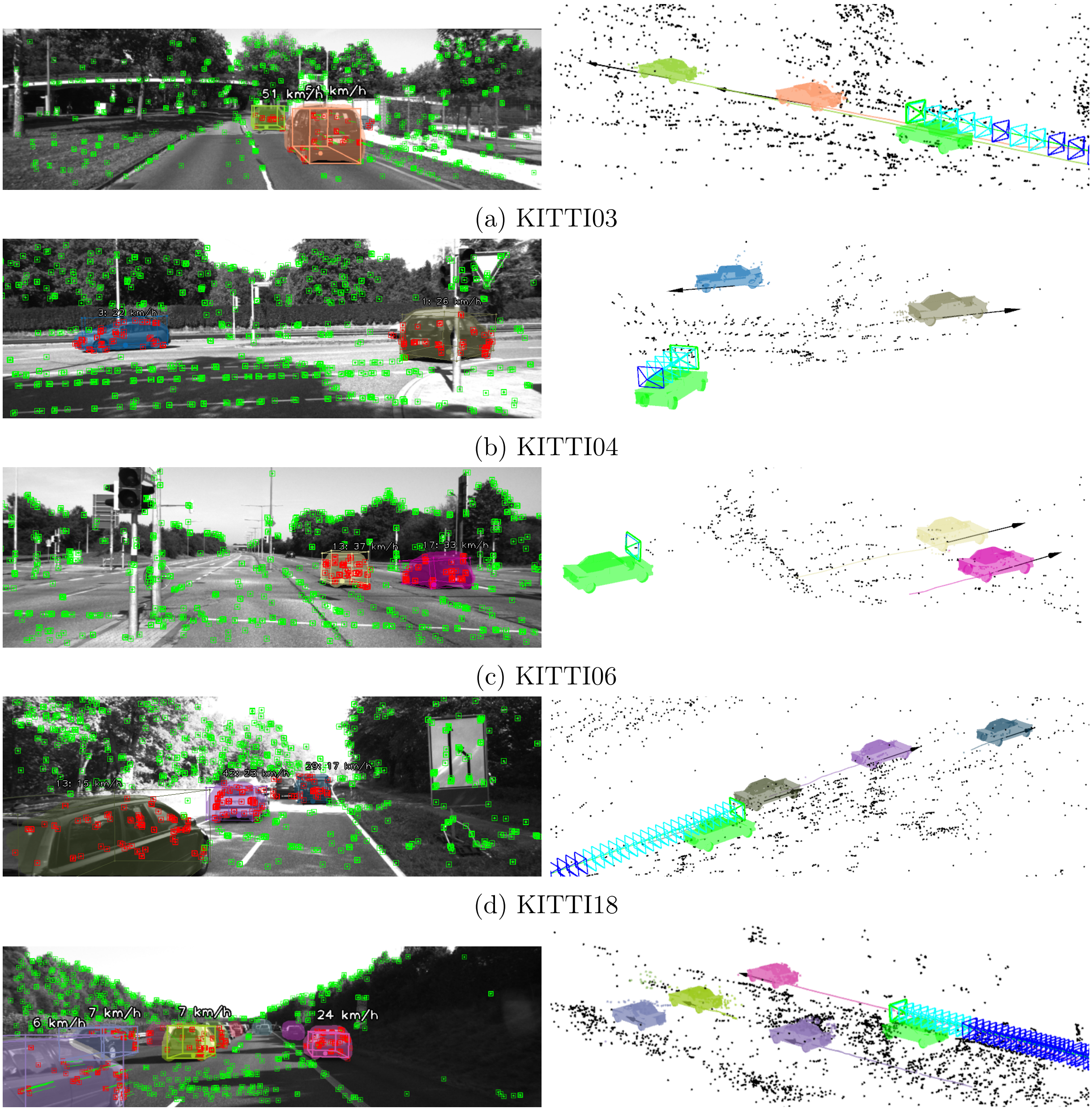
Tables 5.1 and 5.2 detail comparisons of our system's performance against ORB-SLAM2 and our previous work DynaSLAM [6]. We also provide some qualitative samples of our algorithm for these datasets in figure 5.6. ORB-SLAM2 is the base SLAM system on which we build DynaSLAM II, and does not specifically address dynamic objects. DynaSLAM adds ORB-SLAM2 the capability to detect the features belonging to dynamic objects and classes but uniquely ignores them and does not track them. Both

seq	ORB-SLAM2 [97]			DynaSLAM [6]			ClusterSLAM [63]			ClusterVO [64]			Ours		
	ATE	RPE$_\mathrm{t}$	RPE$_\mathrm{R}$	ATE	RPE$_\mathrm{t}$	RPE$_\mathrm{R}$	ATE	RPE$_\mathrm{t}$	RPE$_\mathrm{R}$	ATE	RPE$_\mathrm{t}$	RPE$_\mathrm{R}$	ATE	RPE$_\mathrm{t}$	RPE$_\mathrm{R}$
	(m)	(m)	(rd)	(m)	(m)	(rd)	(m)	(m)	(rd)	(m)	(m)	(rd)	(m)	(m)	(rd)
0926-0009	0.83	1.85	**0.01**	0.81	**1.80**	**0.01**	0.92	2.34	0.03	**0.79**	2.98	0.03	0.85	1.87	**0.01**
0926-0013	0.32	1.04	0.01	0.30	0.99	0.01	2.12	5.50	0.07	**0.26**	1.16	0.01	0.29	**0.93**	**0.00**
0926-0014	0.50	1.22	**0.01**	0.60	1.62	**0.01**	0.81	2.24	0.03	**0.48**	**1.04**	**0.01**	**0.48**	1.35	**0.01**
0926-0051	**0.38**	1.16	**0.00**	0.46	1.17	**0.00**	1.19	1.44	0.03	0.81	2.74	0.02	0.44	**1.14**	**0.00**
0926-0101	**2.97**	13.63	0.03	3.52	15.14	0.03	4.02	**12.43**	**0.02**	3.18	12.78	**0.02**	4.33	15.02	0.04
0929-0004	0.62	1.38	**0.01**	0.56	**1.36**	**0.01**	1.12	2.78	0.02	**0.40**	1.77	0.02	0.64	1.41	**0.01**
1003-0047	20.49	32.59	0.08	**2.87**	**5.95**	**0.02**	10.21	8.94	0.06	4.79	6.54	0.05	3.03	6.85	**0.02**
mean	3.73	7.55	0.02	**1.30**	**4.00**	**0.01**	2.91	5.10	0.04	1.53	4.14	0.02	1.44	4.08	**0.01**

Table 5.2: Egomotion comparison on the KITTI raw dataset.

DynaSLAM I and II use instance semantic priors. However, DynaSLAM I uses them to ignore information belonging to dynamic objects and DynaSLAM II uses them to track the different dynamic objects in the scene and have additional clues for the camera ego motion estimation. The difference in the results of ORB-SLAM2 and DynaSLAM gives an idea of how dynamic each sequence is. Theoretically, if dynamic objects are representative in the scene and they are in circulation, DynaSLAM has better performance, as can be seen in sequences 0020 and 1003-0047 in Tables 5.1 and 5.2 respectively. However, if dynamic objects are representative in the scene but not in motion, *e.g.*, parked cars, DynaSLAM shows a larger trajectory error. This happens because the features belonging to the static nearby vehicles, useful for pose estimation, are not used. This is seen for example in the sequence 0001 in Table 5.1. Besides that, DynaSLAM II achieves a performance better than both ORB-SLAM and DynaSLAM in these two types of scenarios in many of the evaluated sequences. On the one hand, when dynamic instances are moving, DynaSLAM II estimates the velocity of the corresponding objects and provide the BA with rich clues for camera pose estimation when the static representation is not sufficient. This occurs when dynamic objects occlude nearby scene regions and thus static features only provide valuable hints for accurately estimating the camera rotation. Note that, we have not evaluated the estimated speed of the vehicle because there is no ground truth for this. On the other hand, when dynamic classes instances are static, DynaSLAM II tracks their features estimating that their velocity is close to zero, *i.e.*, object points act much like static points. However, our camera tracking performance is seen slightly degraded compared to ORB-SLAM since we allow for more flexibility when estimating the dynamic-object motion status.

Tables 5.1 and 5.2 present our ego motion results compared to those of state-of-the-art systems that also track dynamic objects in a joint SLAM framework. ClusterSLAM [63] acts as a back end rather than a SLAM system and is highly dependent on the camera poses initial estimates. ClusterVO [64] and VDO-SLAM [151] are SLAM systems as ours with the multi-object tracking capability. The former can handle stereo and RGB-D data, whereas the latter only handles RGB-D. The reported errors are given with different metrics so that we can directly use the values that the authors provide. DynaSLAM II achieves in all sequences a lower translational relative error (RPE$_\mathrm{t}$) than that of VDO-SLAM. However, VDO-SLAM usually achieves a lower rotational pose error (RPE$_\mathrm{R}$). Since far points are the ones that provide the richest clues for rotation estimation, we believe that this difference in accuracy does not depend on the object tracking performance and is therefore due to the underlying camera pose estimation algorithm and sensor suite. Regarding the performance of ClusterVO, it achieves an accuracy which is in most sequences quite similar to ours.



(a) KITTI03

(b) KITTI04

(c) KITTI06

(d) KITTI18

(e) KITTI20

Figure 5.6: Qualitative results with the KITTI tracking dataset. On the left, 3D bounding box and the speed of the objects are inferred in the image. Static and dynamic key points are in green and red respectively. On the right, joint estimation of the camera ego motion (green car), the sparse static 3D map (black points) and the trajectories of the dynamic objects. The cyan keyframes allow to optimize the map dynamic structure, whereas the blue ones only optimize the camera pose and the static structure.

5.1.4.2 Multi-Object Tracking

Once the utility of tracking dynamic objects for ego motion estimation is demonstrated, we have chosen again the KITTI tracking [57] and Oxford Multimotion [66] datasets to validate our multi-object tracking results. On the KITTI dataset, dynamic-object trajectories and 3D bounding boxes are provided thanks to expensive manual annotations on LIDAR point clouds.

sequence		0003	0005	0010	0011		0018		0019		0020		
object id (class)		1 (car)	31 (car)	0 (car)	0 (car)	35 (car)	2 (car)	3 (car)	63 (car)	72 (car)	0 (car)	12 (car)	122 (car)
ATE $[m]$		0.69	0.51	0.95	1.05	1.25	1.10	1.13	0.86	0.99	0.56	1.18	0.87
RPE_t $[m/m]$		0.34	0.26	0.40	0.43	0.89	0.30	0.55	1.45	1.12	0.45	0.40	0.72
RPE_R $[°/m]$		1.84	13.50	2.84	12.51	16.64	9.27	20.05	48.80	3.36	1.30	6.19	5.75
2D	TP (%)	50.00	28.96	81.63	72.65	53.17	86.36	53.33	35.26	29.11	63.68	42.77	34.90
	MOTP [%]	71.79	60.30	73.51	74.78	65.25	74.81	70.94	63.50	62.59	78.54	76.77	78.76
BV	TP (%)	39.34	14.48	70.41	61.66	19.05	67.05	21.75	29.48	29.43	43.78	37.64	34.51
	MOTP [%]	56.61	46.84	47.60	50.74	31.95	45.47	41.45	45.69	55.48	45.00	49.29	48.05
3D	TP (%)	38.53	11.45	68.37	52.28	6.35	62.12	16.84	26.48	29.43	31.84	36.23	29.02
	MOTP [%]	48.20	34.20	40.28	47.35	26.02	34.80	35.80	33.89	39.81	46.15	40.81	44.43

Table 5.3: Objects motion comparison on the KITTI tracking dataset.

First of all, we would like to draw the attention of the reader to Fig. 5.6 to have a look at our qualitative results on this dataset. The bounding boxes of the two purple cars on the left are well estimated despite their partial view. This scene is also challenging because the other two front cars are far from the camera and yet are correctly tracked.

In the last decade Bernardin *et al.* [5] introduced the CLEAR MOT metrics to allow for objective comparison of tracker characteristics, focusing on their precision in estimating object locations, their accuracy in recognizing object configurations and their ability to consistently label objects over time. Whereas these metrics are well established in the computer vision and robotics communities and provide valuable insights about the per-frame performance of trackers, they do not take into account the quality of the tracked object trajectories. We suggest that to correctly evaluate multi-object tracking within a SLAM framework, one needs to report the CLEAR MOT metric MOTP [1] as well as the common trajectory error metrics. Most related works on SLAM and multi-object tracking only report the CLEAR MOT metric MOTP [64, 83, 147] and besides that, the authors of VDO-SLAM [151] uniquely report the relative pose error of all objects trajectories of one sequence as a single ensemble. We think that, to facilitate comparison, this metric should be instead reported for individual trajectories.

Table 5.4 shows an evaluation of all object detections in the KITTI tracking dataset with the KITTI 3D object detection benchmark. This allows us to directly compare our multi-object tracking results to those of state-of-the-art similar systems (Table 5.4). The CNNs of Chen *et al.* [26] and specially of Li *et al.* [83] achieve excellent results thanks to the single-view network accuracy itself and the multi-view refinement approach of the latter one, to the detriment of a generality loss. On the other hand, the accuracy of Barsan *et al.* [4] and Huang *et al.* [64] in detecting bounding boxes is remarkable but very sensitive to object truncation and occlusion. Our results show that we handle objects truncation and occlusion with a minor loss in precision. However, less bounding boxes are usually discovered. Our intuition is that our system feature-based nature renders this step specially challenging, opposite to the work by Barsan *et al.* [4], which computes dense stereo matching.

To evaluate our estimation of object trajectories, in Table 5.3 we have chosen the 12 longest sequences of the KITTI tracking dataset whose 2D detections are neither occluded nor truncated, and whose height is at least of 40 pixels. These chosen objects are labeled with their ground-truth object id. For each of these ground truth trajectories we look for the most overlapping bounding boxes in our estimations (the overlapping has to be of at least 25 %). In the case of the trajectory metrics (ATE and RPE) and the 2D MOTP, this

[1] MOTP stands for multiple object tracking precision. It is the predictions precision computed with any given cost function over the number of TPs.

	MOTP$_{BV}$			MOTP$_{3D}$		
	Easy	Moderate	Hard	Easy	Moderate	Hard
[26]	81.34 %	70.70 %	66.32 %	80.62 %	70.01 %	65.76 %
[83]	**88.07 %**	**77.83 %**	**72.73 %**	**86.57 %**	**74.13 %**	**68.96 %**
[4]	71.83 %	47.16 %	40.30 %	**64.51** %	43.70 %	37.66 %
[64]	**74.65 %**	49.65 %	45.62 %	55.85 %	38.93 %	33.55 %
Ours	64.69 %	**58.75 %**	**58.36 %**	53.14 %	**48.66 %**	**48.57 %**

Table 5.4: MOTP evaluation on the KITTI tracking dataset. The categories Easy, Moderate and Hard are based on the 2D bounding boxes height, occlusion and truncation level.

System	Ego Camera	Obj. 1	Obj. 2	Obj. 3	Obj. 4
MVO [66]	0.93	0.36	0.64	0.45	5.94
ClusterVO [64]	0.62	**0.24**	0.45	**0.24**	4.69
Ours	**0.21**	0.41	**0.37**	1.09	**0.28**

Table 5.5: ATE [m] for multi-object tracking and egomotion in swinging_4_unconstrained (Oxford Multimotion dataset).

overlapping is computed as the IoU of the 3D bounding boxes projected over the current frame. For the other two evaluations (BV and 3D), the overlapping is computed as the IoU of the bounding boxes in bird view and in 3D respectively. This evaluation gives an idea of our framework tracking performance and our bounding boxes quality. Regarding the true positives percentage, we can see that objects are tracked for the majority of their trajectory. Missing detections occur because the objects lay far from the camera and the stereo matching does not provide enough features for a rich tracking. Note that, the passersby tracking accuracy is lower than that of the cars due to their non-rigid shape (seq. 0017). The trajectory errors of the cars are acceptable but they are far from the ego-motion estimation performance. Our intuition is that our algorithm feature-based nature renders the bounding box estimation specially challenging. A larger amount of 3D points would always provide richer clues for object tracking.

Finally, to underline the robustness and generality of the presented approach and not to only focus on an outdoor driving scenario where constant velocity models are pretty convenient, we have also evaluated the multi-object tracking performance of DynaSLAM II on the swinging_4_unconstrained sequence from the Oxford Multimotion dataset [66]. This sequence is recorded with a RGB-D and a stereo camera in an indoor environment with four textured boxes hanging from the ceiling and balancing with non-constant velocities. The egomotion and tracking results of DynaSLAM II and of other similar systems on this dataset can be observed in Table 5.5. Our system achieves a significantly higher accuracy for ego motion estimation than compared methods, while objects tracking seems to be more robust with a lower highest error. Experiments also evince that using a soft constant velocity prior for object motion does not restrict our approach to tailored cases. Comparison against VDO-SLAM has been omitted since there exists no feasible way to compute the RPE for object tracking on this dataset.

Sequence	Building block	Time [ms]
KITTI tracking 0003	Tracking thread	80.10 ± 0.78
	Local BA	61.37 ± 6.70
	Bounding Boxes BA	0.07 ± 0.01
KITTI tracking 0020	Tracking thread	94.56 ± 1.27
	Local BA	65.03 ± 17.72
	Bounding Boxes BA	0.60 ± 0.05

	[83]	[151]	[63]	[64]	Ours
fps	5.8	5 - 8	7	8	**10 - 12**

Table 5.6: DynaSLAM II average computational time.

5.1.4.3 Timing Analysis

To complete our proposal evaluation, Table 5.6 shows the average computational time for its different building blocks. The timing of DynaSLAM II is highly dependent on the number of objects to be tracked. In sequences like KITTI tracking 0003 there are only two objects at a time as maximum and runs thus at 12 fps. However, the sequence 0020 can have up to 20 objects at a time and its performance is seen slightly compromised, but still achieves a real time performance at $\sim$ 10 fps. We do not include within these numbers the computational time of the semantic segmentation CNN since it depends on the GPU power and CNN model complexity. Algorithms such as YOLACT [15] can run in real time and provide high-quality instance masks.

Finally, the last rows of Table 5.6 collect the average timing results for systems that jointly perform SLAM and multi-object tracking in the KITTI dataset. DynaSLAM II is the only system that can provide at present a real-time solution.

5.1.5 Discussion

We have proposed an object-level SLAM system with novel measurement functions between cameras, objects and 3D map points. This allows us to track dynamic objects and tightly optimize the trajectories of self and surroundings to let both estimations be mutually beneficial. We decouple the problem of object tracking from that of bounding boxes estimation and, differently from other works, we do not make any assumptions about the objects motion, pose or model. Our experiments show that DynaSLAM II achieves a state-of-the-art accuracy at real time performance, which renders our framework suitable for a large number of real world applications.

The feature-based core of our system limits its ability to discover accurate 3D bounding boxes, and also to track objects with low texture. Fully exploiting the dense visual information would certainly push these limits forward. We would also like to explore the –even more– challenging task of multi-object tracking and SLAM with only a monocular camera. This is an interesting direction since dynamic-object tracking can provide rich clues about the scale of the map.

5.2 Deep Learning for scale-aware monocular SLAM

In the last few years, advances in machine learning have shaken the entire computer vision community. Compared with classical methods, learning based solutions have proved to be outstanding in some tasks like object detection, scene representation or monocular depth prediction. These have direct application to the Simultaneous Localization and Mapping (SLAM) problem which have been and continue to be studied by the community.

With respect to single-view depth prediction, proposed Convolutional Neural Networks (CNN) are able to accurately estimate pixelwise depth for images close to the training domain. Having such an estimation allows pure monocular SLAM/odometries [46, 99] to estimate the true scale of the map, as stereo [97] or visual-inertial [22] systems do. In addition, it mitigates and removes most important pure monocular issues like scale drift [126] or need of an ad-hoc map initialization process. Furthermore, for autonomous driving situations, using a monocular-inertial odometry may not be feasible. Vehicles do not perform 6DoF motion, making inertial parameters, such as IMU biases or scale, have low or null observability.

In this work we leverage these latest advancements and propose a direct monocular odometry pipeline, to tightly integrate information from intensity images and depth prediction inferred from existing CNNs, building a scale-aware system. Using only as inputs the intensity image and the predicted depth allows us to use a large set of existing neural networks, increasing the applicability of our proposal. Next, we enumerate the main contributions of our work,

- A novel tightly-coupled optimization for photometric and depth prediction measurements. In contrast with previous work [132, 145, 146], depth prediction residuals are formulated independently of the intensity image and are included for all observer frames, not only the first one (host/anchor). This allows us to use image points with lower intensity gradient and fully exploit depth prediction measurements, without an increment in computational cost.

- A robust optimization which makes use of *Truncated Least Square* (TLS) cost [144] for depth-prediction residual. This prevents considering inconsistent depth measurements during optimization.

- A general system that can be used with any existing or future depth prediction neural network, performing better than the similar solution from DF-VO [150].

5.2.1 Related Work

Scene scale depth remains ambiguous from monocular images and can not be directly recovered from a single camera. However, it is clear that there exists some relation between intensity image and image depth. Given a large enough dataset, learning based methods are able to learn this relation, allowing us to infer pixel's depth from gray scale images.

One of the first successful works on single-view depth estimation was presented by Eigen et al. [40]. They proposed a CNN with two components, one for the global structure of the image, and the other one to recover fine details, being trained in a semi-supervised way. Latter work from Godard et al. [58] extended the used of CNNs for

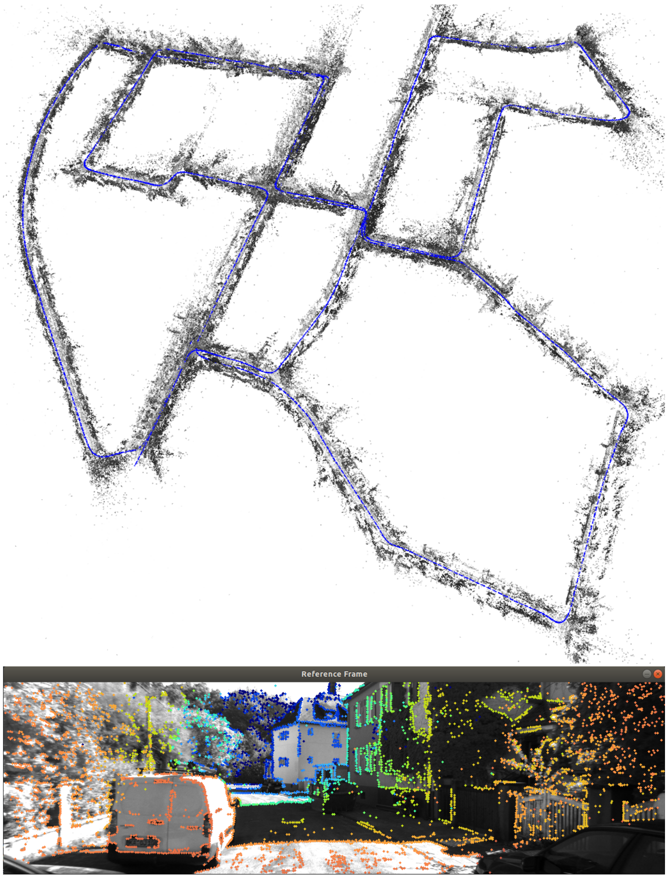


Figure 5.7: Our reconstructed point-cloud and estimated sparse depth map from visual odometry for KITTI00

this task with some important improvements. They formulated different loss functions which allowed unsupervised training, extending the applicability of this approach. More recent work CAM-Convs [47] from Fácil et al. generalizes the solution to different camera calibrations. They present a new kind of convolution which sidesteps the need of training from scratch a new network when camera parameters are modified. In this work we will use an evolution of [58], coined as *monodepth2* [59]. This presents some improvement regarding occlusion and outliers detection, and its implementation is publicly available[2].

The first time depth prediction from a CNN was used for SLAM or odometry was at CNN-SLAM [132] by Tateno et al, where consecutive frames were aligned using photometric and depth prediction measurements. Once obtained these relative transformations, the entire set of keyframe poses was optimized in a pose-graph fashion. More recent solutions include DVSO [145] and D3VO [146] by Yang et al. Both are carefully designed to make them work specifically with DSO [46], which is used as the odometry system as well as for the neural-network supervised training. At DVSO a neural network is trained to estimate not only pixel's depth, but also the disparity in the virtual right camera, which is also used as an odometry input to discard inconsistent points. At D3VO, depth prediction uncertainty, relative pose and brightness transformations between consecutive frames are also computed. This allows to accordingly weight measurements and include pose-graph constraint in the photometric Bundle Adjustment. This approach obtains outstanding results which are similar, if not better, than state-of-the-art stereo solutions.

[2] `https://github.com/nianticlabs/monodepth2`

At DF-VO [150] a more general approach is followed. Depth prediction neural network and its respective odometry are decoupled, being both independently developed. This allows to combine that odometry with any existing or future depth prediction module, gaining flexibility.

In this work we follow a similar approach to DF-VO, in contrast with DVSO or D3VO where the depth prediction module and the odometry system are tightly related. Differently from DF-VO, which also estimates optical flow between consecutive images, our proposal only requires predicted depth as input, keeping our solution even more general. Below we present our proposal.

5.2.2 Direct visual odometry with depth prediction

Map data for our odometry consists of map points and keyframes. Map points are represented using an inverse depth parametrization [29]. For the first observer keyframe, named as *host* or *anchor*, we keep azimuth and elevation fixed, having only inverse depth ρ as optimizable parameter [46]. A keyframe will be parametrized with its pose $\mathbf{T} \in \mathrm{SE}(3)$ and its brightness affine transformation (a, b). For tracking purposes, we define a temporal active window consisting of the last $N_a = 5$ keyframes. All points seen from these keyframes will define the set of active map points, which are those probably observed from current frame. We also define a larger optimization window, consisting of the $N_o = 7$ last keyframes. These will be optimized in a back-end photometric-depth Bundle Adjustment along with map points seen or hosted by them.

In the following points we will describe in detail our proposal. First, at 5.2.2.1, we explain how map points and keyframes are created and removed. We continue with frame photometric tracking, at point 5.2.2.2. Finally, at 5.2.2.3, we present our novel multi-view and tightly-coupled photometric-depth optimization.

5.2.2.1 Map point and Keyframe management

In contrast with pure monocular approaches, having a single-view depth estimation allows to initialize points just from one view. This is also especially important for map initialization, since we can boost our system with a single frame, as stereo odometries do. The keyframe where the point is initialized will be set as the anchor or host keyframe. When a new keyframe is inserted, we initialize points in image regions where there exist no observations. To this end, we compute the mean μ and standard deviation σ of intensity values at each cell along a 16×32 grid in the image. For each cell, we extract points whose gradient is above $\mu + f\sigma$, where f is an adaptive factor. We make several extractions, starting with a high f value and decreasing it until we have at least 2000 hosted or observed points in the current keyframe. Each time we extract a new point, we mask its neighbors within a 5×5, window to avoid extracting overlapping points.

We also perform a map point culling process. In this way, a map point can be removed for three reasons:

1. When a map point has been created outside of the active window and it has less than 2 observations. This removes points which are difficult to track.

2. When the mean photometric residual of its observations is higher than 9 intensity values. This removes probably bad estimated points.

3. When it has at least one observation and its inverse depth information, quantified with Hessian block from previous BA, drops below some threshold. In this sense even a point with several observations may contain little information. This is the case when its geometric derivatives (very far points) or image gradient (textureless points) are small, or they are orthogonal to each other (see appendix B.3).

For keyframe creation we follow a simple heuristic: when the number of inliers in frame tracking (see section 5.2.2.2) drops below 70%, a new keyframe is created. Each time a keyframe is inserted, all active points are assumed to be observed. It is the photometric-depth optimization which will manage and discard outlier observations, as explained in 5.2.2.3. When a keyframe gets out of the active window and more than 80% observed and hosted points have at least 3 observations, we discard that keyframe since we consider it contains too redundant information.

5.2.2.2 Photometric Tracking

This task consists in estimating the current frame estate, relative pose and brightness affine transformation with respect to the last keyframe, named as reference keyframe. We will follow a procedure similar to DSO [46].

For this goal, when the reference keyframe is updated, we first build a sparse depth map, $D : \Omega_D \to \mathbb{R}$, by projecting active points into reference image Ω and dilating them to get a denser set $\Omega_D \subset \Omega$. Since map points are created from a single view, projected points may have from one (low accuracy) to multiple (high accuracy) observations. In contrast with DSO and to take into account this disparate amount of information and accordingly weight each projected point, we propose to weight them using the information value of its inverse depth, computed from the last photometric-depth bundle adjustment. This limits the influence of recently created points with higher uncertainty while points with more observations and better conditioned will dominate the solution.

We remark we build this depth map D from our estimated point cloud instead of directly using the predicted depth map from the neural network. Since our estimated points have been refined combining photometric and depth-prediction residuals they are much more precise than network output. Once map D is built, we solve the following optimization problem:

$$\underset{\{\mathbf{T},a,b\}_{i,\text{ref}}}{\arg\min} \sum_{\mathbf{u}\in\Omega_D} \rho_{\text{Hub}} \left(\| I_{\text{ref}}(\mathbf{u}) - \; e^{-a_{i,\text{ref}}} \{ I_i(\pi(\mathbf{T}_{i,\text{ref}} \pi^{-1}(\mathbf{u}, D(\mathbf{u})))) - b_{i,\text{ref}} \} \|^2 \right) \tag{5.14}$$

where images are defined as $I : \Omega \to [0, 255]$ and $\pi : \mathbb{R}^3 \to \Omega$ and $\pi^{-1} : \Omega \times \mathbb{R} \to \mathbb{R}^3$ are the camera projection map and its inverse. Unknowns to be found are $\mathbf{T}_{i,\text{ref}} \in$ SE(3) transformation from reference keyframe to current frame and $\{a_{i,\text{ref}}, b_{i,\text{ref}}\}$ for its relative brightness affine transformation as explained in [46]. Relative motion is initialized assuming a constant velocity model, while affine parameters are set to values from the last frame. In addition, we use a Huber robust norm to downweight outlier observations. We run this optimization in a multiscale fashion, starting at the coarsest scale level (we use five scales with scale factor 2) and going down until the original image resolution. We optimize with Levenberg-Marquardt until convergence at each level, with a maximum of 20 iterations per level, since this optimization is very efficient. If we detect that the optimization has not converged in the coarsest level, usually due to big rotations, we update the initial attitude estimate with small rotations. We keep the solution which provides the lowest mean photometric residual. Along this optimization we do not include depth residuals which avoids running the depth-prediction depth-prediction at frame rate, bringing an important computaational saving.

Instead of solving (5.14), we follow the more efficient inverse compositional approach as presented in [2], also adopted in DSO. Details about this approach are given in appendix C.

5.2.2.3 Photometric-Depth optimization

Each time a new keyframe is inserted, we run a windowed optimization for the last $N_o = 7$ keyframes, denoted as $\mathcal{K}$. Variables to be optimized are keyframe poses and affine parameters inside the optimizable window, as well as inverse depth from all their observed points $\mathcal{P} = \{\mathcal{P}_1 \cup \cdots \cup \mathcal{P}_{N_o}\}$. All observations for these points are included while observer and host keyframes outside the optimization window remain fixed. For this optimization we include two kinds of residuals, with a tunable parameter k weighting between them, leading to the following optimization problem:

$$\underset{\{\mathbf{T}_i,\rho^j\}_{i,j}}{\arg\min} \sum_{i\in\mathcal{K}} \sum_{j\in\mathcal{P}_i} \rho_{\text{Hub}}\left(\left\|r^j_{i,\text{photo}}\right\|^2\right) + k\,\rho_{\text{TLS}}\left(\left\|r^j_{i,\text{depth}}\right\|^2\right) \tag{5.15}$$

A simplified factor graph representation for this optimization problem is given in figure 5.8.

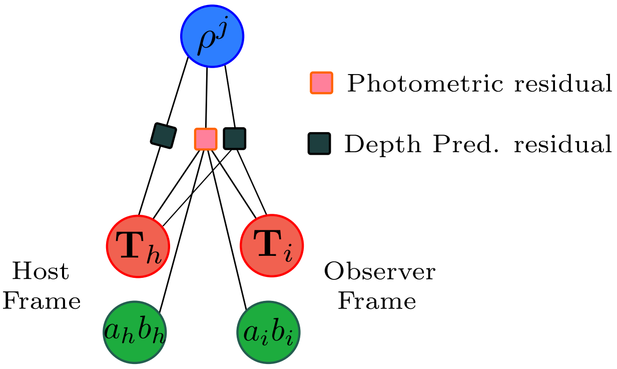


Figure 5.8: Factor graph representation for photometric-depth BA. Only one point, its host and one observer keyframe are shown for simplicity.

The first residual, $r^j_{i,\text{photo}}$, is the photometric one, which relates point j and observer keyframe i as follows:

$$r^j_{i,\text{photo}} = \sum_{\mathbf{u}^{j\prime}_h \in \mathcal{N}_{\mathbf{u}_j}} I_h\left(\mathbf{u}^{j\prime}_h\right) - b_h - \frac{e^{a_h}}{e^{a_i}}\left(I_i\left(\mathbf{u}^{j\prime}_i\right) - b_i\right) \tag{5.16}$$

such that

$$\mathbf{u}^{j\prime}_i = \pi\left(\mathbf{x}^{j\prime}_i\right) \quad \text{with} \quad \mathbf{x}^{j\prime}_i = \mathbf{T}_{ih}\pi^{-1}\left(\mathbf{u}^{j\prime}_h, \rho^j\right) \tag{5.17}$$

where ρ^j is the inverse depth and $\mathbf{u}^{j\prime}_h$ are the image coordinates of neighbor pixels in the host for point j. We use the same patch $\mathcal{N}_{\mathbf{u}_j}$ as proposed at DSO [46] which allows fast vectorized computation. In contrast with DSO, where the $\mathbf{T}_i$ update is performed in the local reference, we prefer to apply it on the global reference. This makes derivatives with respect to the host and observer frame only differ on sign, as shown in the appendix section B.3.1, resulting in a computational reduction. The drawback of this formulation is derivatives depend on distance to origin, which may cause stability issues when far from it. To remove this, we apply a translation offset to the entire set of keyframes, bringing the last keyframe to the origin. Since keyframes involved in optimization are spatially close, this issue disappears. Once solved the optimization, we take all keyframe poses back to the original reference. We use a Huber kernel to linearly weight outliers, with its threshold set to 9 for pixel.

The second residual accounts for depth prediction measurements. From the neural network, we have an inverse depth estimation $D^i_{NN} : \Omega \rightarrow \mathbb{R}^+$ for a given keyframe i. For each point j observed from keyframe i we define the following depth prediction error:

$$r^j_{i,\text{depth}} = D^i_{NN}(\mathbf{u}^j_i) - \rho^j_i \quad \text{s.t.} \quad \rho^j_i = [\mathbf{x}^j_i]^{-1}_z \tag{5.18}$$

where $\mathbf{x}^j_i$ and $\mathbf{u}^j_i$ can be computed similar to (5.17). $[\cdot]_z$ takes the z component. For this error we do not use a patch of pixels, since depth map is usually much more smooth than intensity image and close pixels usually contain redundant depth information.

In addition, we use a *Truncated Least Square* (TLS) cost function [144], also known as threshold cost [88]. When the depth prediction residual is above a threshold, its gradient vanishes, which may be seen as setting its weights to zero, removing its influence. There are two reasons for using this TLS cost. First, when photometric and depth prediction measurements do not agree, we rely on the former and neglect the later which is more prone to inconsistencies. Second, for each map point we have prediction-depth residuals from each observer keyframe, which may not be consistent between them. Prediction accuracy may depend on the point of view, on the region of the image where the prediction is made or on how far the point is. Using a TLS robust cost function allows to activate only depth measurements which are consistent between them or agree with photometric information. The threshold of this TLS cost function is set to 0.01 m^{-1}. We also define the depth prediction residual for the host keyframe, which takes a simpler form:

$$r^j_{h,\text{depth}} = D^h_{NN}(\mathbf{u}^j_h) - \rho^j \tag{5.19}$$

This residual proves to be very useful to constraint points with lower photometric information. We remark we are including depth prediction residuals for all observer keyframes, which contrasts with [145, 146], where only host depth measurement is considered. Including this error does not suppose a big computational cost increment, since a lot of terms may be reused from photometric residual (see appendix B.4).

Affine parameters a and b are prone to drift during optimization and they also add extra degrees of freedom to the optimization problem. To avoid this, in addition to these residuals, we add a strong prior to keep them close to zero. As for photometric tracking, this optimization is solved in a multiscale way. However, since previous optimizations have been run for keyframes and map points, estimates are close to minima. Thus, it is not necessary to start the optimization from the coarsest level. Instead, we start at the third finest level, which gives a basin of convergence of 4 pixels.

Once solved the optimization, we discard outlier observations based on two criteria. First, if the average photometric residual for pixels in $\mathcal{N}_{\mathbf{u}_j}$ is over a threshold, 9 intensity values in our implementation, we discard it. Second, if the number of pixels in $\mathcal{N}_{\mathbf{u}_j}$ with a high residual (15 intensity values or more in our implementation) is greater than 40%, we discard that observation.

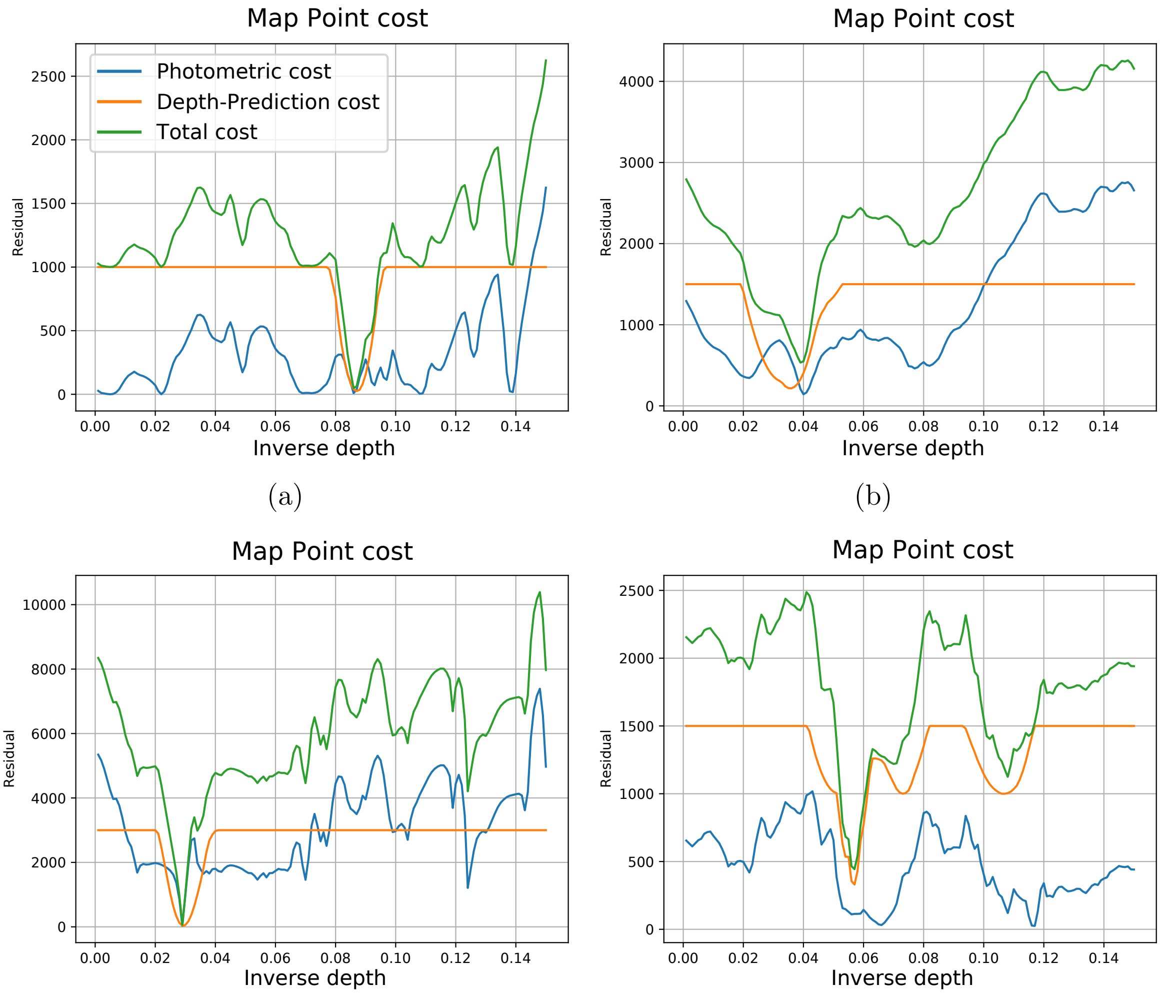
Comparing how photometric and depth prediction residuals are combined together we find differences with previous works. At CNN-SLAM [132], depth prediction residual is not explicitly used. Instead, the photometric residual is made dependent on the depth map prediction. DF-VO [150] solves a PnP problem, using predicted depth. Other methods like DVSO [145] or D3VO [146] convert the inverse depth prediction to an equivalent stereo observation, defining a photometric error for the virtual stereo system. For all these approaches, the residual and its derivatives depend on intensity image and its gradient, which is more prone to contain noise and is much less smooth than predicted depth gradients. In addition, it restricts the use of predicted depth to regions with high visual texture.

In figure 5.9, we plot photometric, depth prediction and total costs for some map points during photometric-depth BA. In figure 5.9d, predicted depth cost (orange) has multiple minima since not all predicted depths are consistent. However, the lowest one also corresponds with one photometric minimum (blue), leading to a clear absolute minimum in the total cost (green). Adding predicted depth also increases convergence region of the total cost (Fig. 5.9b). In some cases, all depth predictions and photometric measurements are consistent with an equivalent minimum, as shown in figure 5.9c. As we have previously stated, for low gradient points or repetitive regions, whose photometric cost may be plagued with multiples shallow minima, predicted depth increases the convergence region for point's depth (Fig. 5.9a).

5.2.3 Results

We evaluate our proposal on KITTI odometry [57], a self-driving oriented dataset. This includes some of the sequences where *monodepth2* has been trained as well as others with similar characteristics not used for training, following the Eigen split proposal [40]. This contains challenging sequences for pure monocular odometries since there exist almost pure turns which lead to big scale drift. In addition, it has a low frame rate (10 fps) which makes tracking more difficult, and has important luminosity changes which may cause brightness affine parameters to easily diverge.

We compare our system against monocular DSO [46], monocular and stereo versions of ORB-SLAM [97, 99], with the loop-closing thread deactivated, for a more fair comparison, and also against DF-VO [150] a monocular odometry based on depth prediciton. We measure the RMSE of Absolute Trajectory Error (ATE) [128] which are reported at table 5.7. Compared with pure monocular systems, our method based on depth prediction has an accuracy 5 times higher than ORB-SLAM and 9 times higher than DSO. Compared with the similar system DF-VO, which is the closest system to ours also using depth prediction, our proposal achieves a higher accuracy for 7 out of 11 sequences, with a lower average error. We use bold characters for better results among this comparison. Using a stereo system gives better performance than our system, with a 57% error reduction. We also detect that our method has quite diverse behaviour, being very dependent on the sequence type. The best results are obtained for sequences moving along urban environments with close building at both sides, as sequences 00, 02, 04, 06, 07, 08. Other urban sequences like 03, 05 or 10, with further buildings and more vegetation produce worse results. For highway environments, namely sequence 01, our system does not get good results at all, with stereo ORB-SLAM being the only system able to obtain low error. We hypothesize that these differences may be due to training dataset distribution, where more urban environments are used, while highway sequences are much less frequent.



(a)

(b)

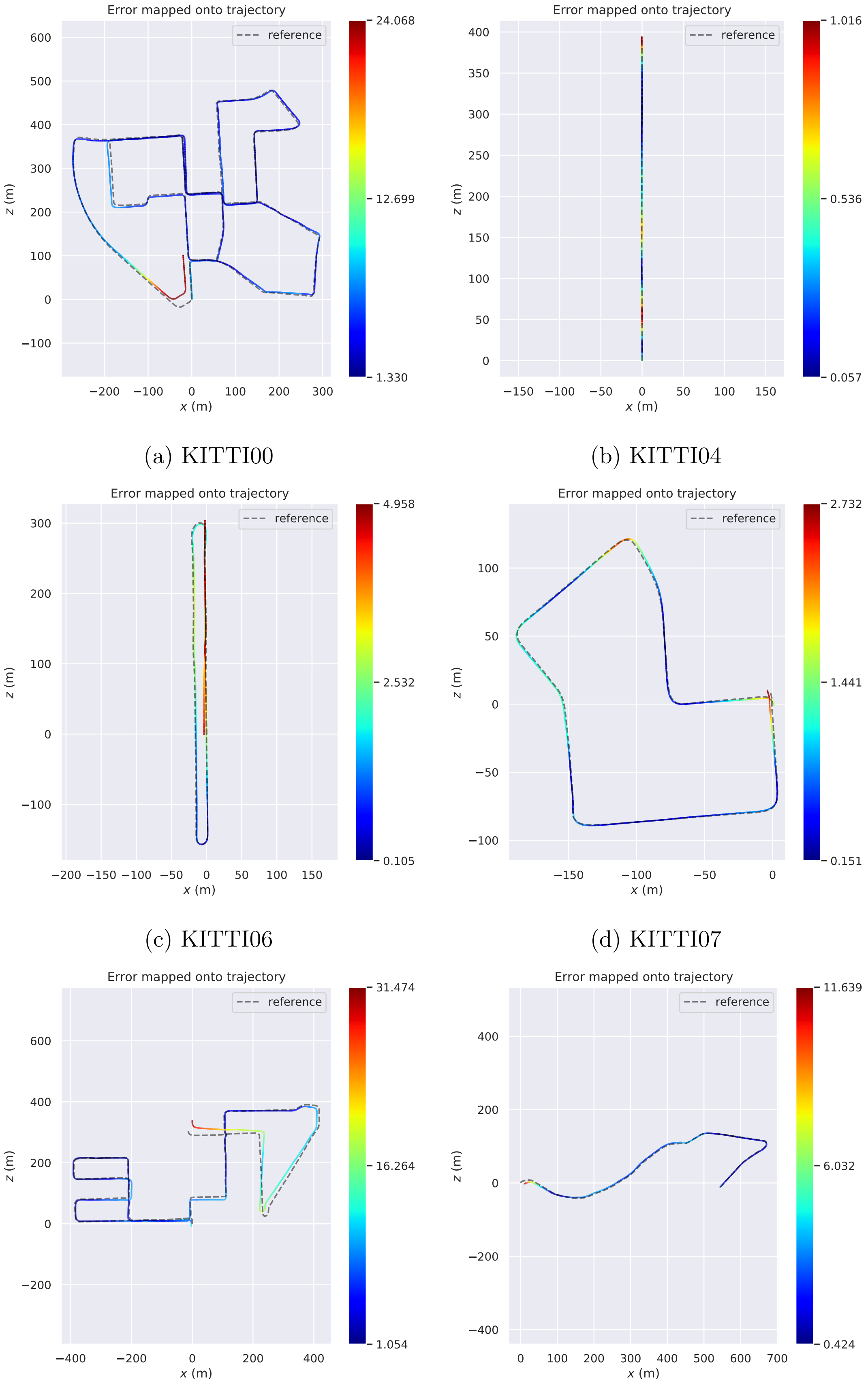
(c)

(d)

Figure 5.9: Optimization cost w.r.t. ρ^j for different situations

Several results of our proposal are shown in figure 5.10. Major source of error for monocular odometries, scale drift, has completely disappeared leading to much more accurate results. More detailed results for sequence 00 are presented in figure 5.11. We highlight that despite the lack of a loop closing module in our solution, we achieve zero drift for most of this sequence, as shown in figure 5.11a, where multiple paths along the same streets can barely be differentiated. In this sense, regarding figure 5.11b, we can see that our proposal accumulates most of the error in a small section of the trajectory at the end of the sequence. In fact, this part corresponds with a non-urban environment, where both sides of the road are covered with vegetation, leading to less accurate results from *monodepth2* as previously hypothesized. If this last part were not considered, the RMS ATE would be much closer to the median ATE, equal to 4.63m. An example of the reconstructed point cloud, as well as the sparse depth map used for frame tracking (see section 5.2.2.2) are shown in figure 5.7, top page. More detailed point clouds are shown in figure 5.12

For a more precise solution, a more complex neural network could be used as other methods do, but the solution would be less general. At DVSO paper [145], they report results obtained with its odometry (DSO) using *monodepth2* for the depth prediction, ob-



(a) KITTI00

(b) KITTI04

(c) KITTI06

(d) KITTI07

(e) KITTI08

(f) KITTI10

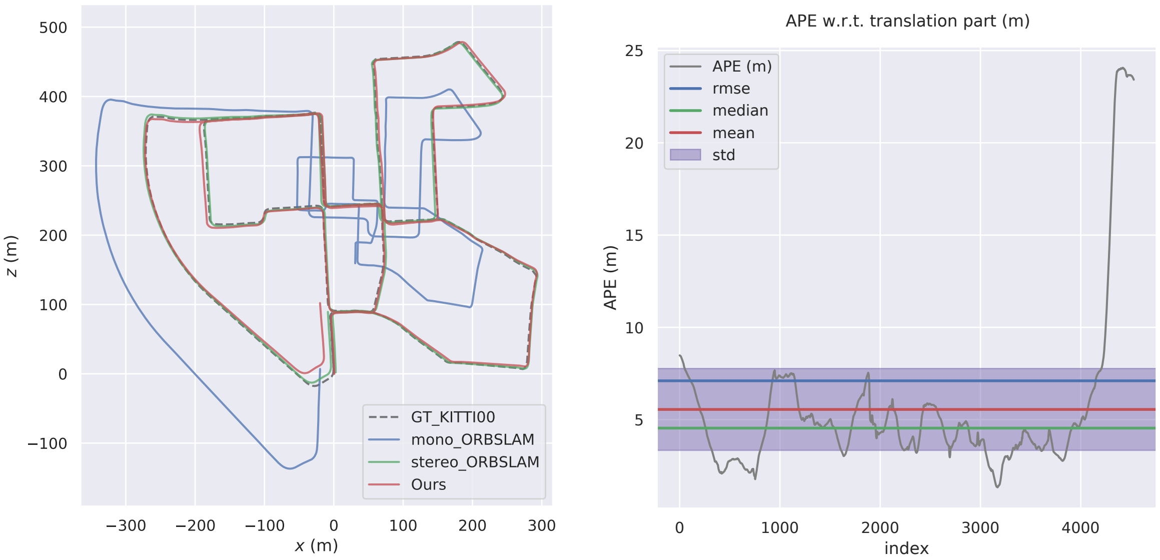
Figure 5.10: Some sample trajectories at KITTI dataset coloured by ATE error with respect to ground-truth. Graphics from *evo* software `https://github.com/MichaelGrupp/evo`

Table 5.7: RMSE ATE (m) errors for KITTI odometry dataset.

	ORB-SLAM mono (No LC)	DSO*	DF-VO [150]	Ours	ORB-SLAM stereo (No LC)
KITTI00	77.19	113.18	11.34	**7.10**	3.99
KITTI01	110.12	-	484.86	**25.38**	1.38
KITTI02	34.32	116.81	21.16	**14.12**	8.82
KITTI03	0.90	1.39	2.04	**1.71**	0.25
KITTI04	0.72	0.42	0.86	**0.29**	0.22
KITTI05	36.29	47.46	**3.63**	7.51	2.18
KITTI06	52.61	55.62	**2.53**	4.05	1.81
KITTI07	17.04	16.72	**1.72**	2.41	1.43
KITTI08	56.42	111.08	**5.66**	10.43	3.22
KITTI09	55.74	52.23	10.88	**9.15**	3.26
KITTI10	8.44	11.09	3.72	**3.40**	0.88
Avg.†	30,54	52.6	6.35	**6,02**	2.60

*: Results for DSO are extracted from [150]

†: KITTI01 is not used for average to ease comparison between methods



(a) Aligned trajectories

(b) ATE evolution for our method

Figure 5.11: Comparative results for KITTI00 dataset. Loop closing thread has been deactivated for ORB-SLAM solutions. Notice the huge scale drift of pure monocular solution, which is overcome with our solution based on depth prediction.

taining results with an accuracy close to ours. On the other hand, D3VO gets impressive results, but they do not report results using depth prediction from other neural networks and their implementation is not open-source and cannot be adopted in our work.

In addition, for the supervised training of these two works, they use a stereo version of DSO, which computes a very accurate true scale point cloud for high gradient points. Its neural networks learn to estimate very precisely this kind of points, which are also used along their odometries based on DSO. Coupling the odometry and the depth prediction in this way probably reduces the generality, which is one of our goals, but leads to very

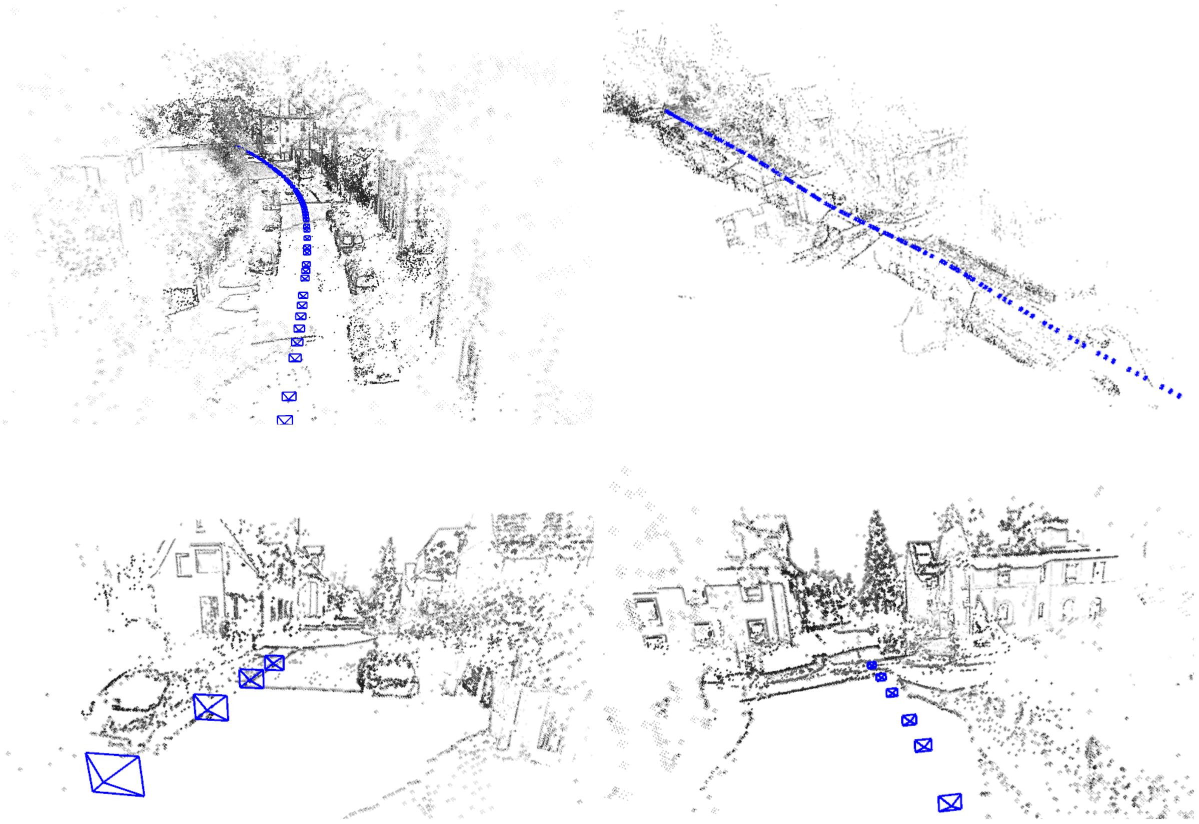


Figure 5.12: Close looks for reconstructed point clouds.

accurate results.

5.2.4 Discussion

In this work we have presented a direct monocular odometry based on depth prediction from neural networks. We have shown that combining multi-view depth prediction and photometric residuals in a single optimization makes scale observable, removes scale-drift and leads to a much more accurate estimation than pure monocular solutions. Using a truncated robust cost (*TLS*) for depth residuals allows us to consider only consistent measurements, making our optimization robust against spurious depth data. Our solution only requires the predicted depth estimation from the neural network, making our solution very general and enabling to integrate it along with most of existing networks. Using *monodepth2* [59] as it is, we have evaluated our system in multiple sequences and compared against monocular SLAM and DF-VO, system similar to ours. Our system gets results 5 to 9 times more precise than monocular solutions and is more accurate than DF-VO for 64% of the sequences. The experiments demonstrate the validity of our proposal.

As future work we identify updating this odometry to a complete SLAM system, with a covisible and not temporal optimization window, which would boost its performance. This would entail some unsolved challenges for long term SLAM since images, depth prediction and its gradients and pyramidal decomposition should be stored in memory. In addition, including a loop-closing module would also be a big improvement. Making it work directly on images, without using features, is also an open problem. The generality of our approach opens the way to use other depth neural networks as well as other datasets.

Chapter 6

Conclusions and future work

6.1 Conclusions

This thesis aims to increase the robustness of Simultaneous Localization and Mapping (SLAM), making it able to deal with a wider variety of situations. For this goal, we have explored different sensor configurations, with especial interest in visual-inertial setups, proposing new methods to make the most of them. We have also made use of deep-learning tools to solve some SLAM related problems which may not be solved in a classic geometric way, such as SLAM in dynamic-environments or monocular scale-aware SLAM.

First, we have focused on initializing visual-inertial SLAM, aiming to find faster and more accurate solutions than existing ones. In section 3, we have presented two different inertial initialization techniques. The first one, section 3.2, was a joint visual-inertial initialization which computes the 3D landmarks, camera poses and inertial parameters in a single step. Since some simplifications are needed to find a closed form solution, we have proposed two observability and consensus checks to discard inaccurate results. This allowed us to find motions with consistent scale errors lower than 5%, using trajectories of 2 seconds or less. These tests remove all bad solutions but lead to lower recall, being only able to initialize in some parts of the trajectory. In section 3.3, we have explored the use of disjoint visual-inertial initialization, where first vision parameters are found, and later the inertial sensor (IMU) is initialized. We have formulated the entire initialization as a Maximum A Posteriori estimation problem, proposing a novel inertial-only optimization which properly deals with the measurements uncertainties. This method has proved to be more efficient, accurate and with a higher recall than any other solution in the state-of-the-art. This confirms that solving first vision (up to scale) and taking into account IMU uncertainty is crucial to obtain quick and precise visual-inertial solutions. Having such a fast initialization allows using IMU from the very beginning.

Second, in section 4, building on the well known ORB-SLAM2 system, we have developed ORB-SLAM3, a new real-time, multi-map and visual-inertial SLAM system. Currently this is the most complete open source SLAM system, which expands new opportunities for the research community. We have used and further developed our disjoint IMU initialization method, not only for launching ORB-SLAM3, but also to refine the scale and gravity direction after initialization, in a very efficient way. We have extended previous work to the monocular and stereo inertial configurations, for pinhole and fisheye cameras, covering the most used sensor setups. Our exhaustive and detailed experiments have demonstrated that ORB-SLAM3 is the most precise system, specially within inertial configurations. Our stereo-inertial SLAM gets errors below 1cm for typical small AR/VR scenarios. Furthermore, compared to competing methods, it is one order of magnitude more precise for medium size scenarios, where the camera is revisiting common regions and mid-term data association may be exploited.

Finally, we have also explored the use of deep learning techniques to improve the robustness of SLAM for dynamic environments and monocular configurations. First, in section 5.1, we have introduced DynaSLAM II, a stereo feature-based SLAM system. We first segment dynamic objects by means of the deep neural network Mask-RCNN, and instead of neglecting them, we incorporate their points and measurements to the estimation problem, formulating an efficient Bundle Adjustment where dynamic objects' pose and velocity are optimized. We also present a procedure to compute dynamic objects' bounding boxes, which enhance the volumetric understanding of the scene. All these together makes DynaSLAM II a robust and efficient system for dynamic environments, showing that ego-motion estimation may also benefit from dynamic objects. Second, in section 5.2, we have explored the application of depth-prediction neural networks to monocular odometry. We have proposed a direct method which combines multiview photometric and depth-prediction residuals, in a tightly coupled optimization. Using robust cost functions, we can discard inaccurate depth prediction residuals while keeping those consistent between them and with photometric measurements. Including also the depth prediction measurements allows to include points with lower photometric gradient. This leads to a monocular system which does not suffer from scale drift, being several times more precise than monocular systems, while getting closer to stereo accuracy.

All the work developed in this thesis goes in the same direction, increasing SLAM robustness.

Conclusiones

Esta tesis tiene como objetivo el incremento de la robustez de la localización y reconstrucción simultánea de mapas (SLAM, por sus siglas en inglés) que le haga capaz de lidiar con una mayor variedad de situaciones. Con este fin, hemos explorado distintas configuraciones de sensores, con especial interés en los equipos visual-inerciales, proponiendo nuevos métodos para sacarles el máximo provecho. También hemos usado herramientas de aprendizaje profundo para resolver algunos de los problemas que no pueden ser resueltos mediante geometría clásica, como el SLAM en entornos dinámicos o estimación de escala para SLAM monocular.

Primero, nos hemos centrado en la inicialización del SLAM visual-inercial, con el objetivo de encontrar soluciones más rápidas y precisas que las que ya existen. En la sección 3, hemos presentado dos técnicas diferentes de inicialización. La primera, sección 3.2, era una inicialización conjunta visual-inercial que calcula en un único paso la posición 3D de los puntos, la posición de las cámaras y los parámetros inerciales. Ya que se asumen algunas simplificaciones para la obtención de una expresión cerrada, hemos propuesto dos comprobaciones de observabilidad y consenso para descartar los resultados inadecuados. Esto nos permite encontrar de manera consistente soluciones con un error de escala por debajo del 5% usando trayectorias de 2 segundos de duración. Las comprobaciones propuestas eliminan todas las soluciones malas pero conllevan una exhaustividad baja, siendo únicamente capaz de inicializar en determinadas partes de la trayectoria. En la sección 3.3, hemos explorado el uso de una inicialización visual-inercial disjunta, donde se obtienen primero los parámetros visuales para posteriormente inicializar el sensor inercial (IMU). Hemos formulado la inicialización como un problema de estimación de Máximo a Posteriori, proponiendo una nueva optimización únicamente inercial que considera adecuadamente las incertidumbres de medida. Este método ha demostrado ser más eficiente y con una mayor exhaustividad que el estado del arte. Esto confirma que resolver primero la visión (a un factor de escala) y tener en cuenta las incertidumbres inerciales es crucial para una rápida y precisa solución. Tener una inicialización así de rápida y exhaustiva permite empezar a usar la IMU antes.

Segundo, en la sección 4, construido en el popular sistema ORB-SLAM2, hemos desarrollado ORB-SLAM3, un nuevo sistema SLAM en tiempo real, multimapa y visual-inercial. Actualmente, este es el sistema SLAM de código abierto más completo, el cual abre nuevas oportunidades para la comunidad investigadora. Hemos usado nuestra inicialización de la IMU disjunta previamente desarrollada no solo para inicializar ORB-SLAM3, sino también para refinar de manera muy eficiente la escala y dirección de gravedad una vez inicializado el sistema. Hemos extendido el trabajo previo para hacerlo funcionar con cámaras monoculares y estéreo, tanto para lentes estenopeica como de ojo de pez, cubriendo la mayor parte de sensores utilizados. Nuestros detallados y minuciosos experimentos han demostrado que ORB-SLAM3 es el sistema más preciso, especialmente en sus configuraciones inerciales. Nuestro sistema de SLAM estéreo-inercial obtiene errores por debajo del centímetro para escenas típicas de realidad virtual y aumentada. Además, comparado con métodos similares, nuestro sistema es un orden de magnitud más preciso en entornos de tamaño medio, donde la cámara revisita zonas comunes y la asociación de datos a medio plazo puede ser explotada. Finalmente, hemos explorado el uso de técnicas de aprendizaje profundo para la mejora de la robustez del SLAM en entornos dinámicos y en configuraciones monoculares. Primero, en la sección 5.1, hemos presentado DynaSLAM II, un sistema SLAM estéreo basado en puntos característicos. Primero segmentamos los objetos dinámicos mediante la red profunda Mask-RCNN y en lugar de descartarlos, incorporamos sus puntos al problema de estimación, formulando un ajuste de rayos eficiente donde la posición y velocidad de los objetos dinámicos son optimizadas. También presentamos un procedimiento para calcular el volumen ocupado por el objeto dinámico. Todo esto hace de DynaSLAM II un sistema robusto y eficiente para entornos dinámicos. En segundo lugar, en la sección 5.2, hemos explorado la aplicación de redes de predicción de profundidad para odometría monocular. Hemos propuesto un método directo que combina errores fotométricos y de predicción de profundidad desde distintas vistas, en una misma optimización. Usando funciones de coste robusto se pueden descartar las predicciones de profundidad incongruentes mientras se mantienen las consistentes con la fotometría y con el resto de predicciones. El añadir errores de predicción de profundidad también permite incluir puntos con menor gradiente fotométrico. Todo esto da lugar a un sistema monocular que no sufre deriva de escala, varias veces más preciso que los sistemas monoculares, con una precisión que se acerca a la de sistemas estéreo.

Como se puede ver todo el trabajo desarrollado en esta tesis va en la misma dirección: incrementar la robustez de los sistemas SLAM.

6.2 Future work

SLAM is a very active field of research, with a lot of ongoing and future work, being still an open problem. In this thesis we have covered some of the most relevant topics, with especial interest in robustness. In this respect, we identify as work to be done the next points ordered by relevance,

- More complete geometric map representations. We have seen that ORB-SLAM3 gets very impressive results in terms of trajectory accuracy, with sub-centimeter precision for stereo-inertial setup in room-size environments. However, as a feature based method, map representation is very sparse and can be merely used for robot localization. In this sense, a dense representation would allow a higher interaction with the environment. Recent works on neural implicit representation are very promising [93, 101], offering a compact, dense and differentiable representation, potentially allowing dense SLAM with a small memory and computational footprint. How to integrate these methods along the SLAM pipeline, reformulate the Bundle Adjustment, as well as making them computationally efficient for real time SLAM are the biggest challenges.

- Include semantic information along with the map. In section 5.1, we have classified objects as dynamic and non-dynamic, but this classification could be further developed, leading to a full semantic SLAM. Relations between different objects could be included in the estimation problem.

- Visual-inertial SLAM for dynamic and non-rigid scenarios. When most of the scene is dynamic, visual sensors are not enough to distinguish between scene and ego motion. For non-rigid environments, specially for non-isometric deformation, something similar happens. Including an inertial sensor in dynamic and non-rigid SLAM would help to decouple the scene motion/deformation and the ego motion.

- Multi-IMU visual-inertial SLAM. Combining more than one IMU in the same setup could bring some important benefits. Two non-aligned IMUs could increase the observability of the inertial parameters, reducing initialization complexity and making it faster. They could also reduce the signal-noise ratio of the IMU measurements, which could be also achieved with a single better IMU. It would be interesting to study the most efficient spatial distributions of these IMUs to get the less redundant information.

Finally, making ORB-SLAM3 open-source we provide the research community with the most complete state-of-the-art visual-inertial SLAM system. It can be used by researchers as a baseline for comparison and new tools can be built on top of it. We wish the community to push ORB-SLAM3 beyond its limits, showing its limitations and failure cases, pointing out where more research effort is needed. This will definitely help to advance SLAM to new horizons.

Appendices

Appendix A

Bundle Adjustment

A.1 Non-linear optimization

In Bundle Adjustment we have a state vector $\mathbf{x}$, whose components relate between them by mean of measurements (residuals) vectors $\{\mathbf{r}_1(\mathbf{x}) \dots \mathbf{r}_n(\mathbf{x})\}$, defining the squared errors $\{c_1(\mathbf{x}) \dots c_n(\mathbf{x})\}$ such that $c_i(\mathbf{x}) = \mathbf{r}_i^T \mathbf{r}_i$. Our problem can be stated as finding the state $\mathbf{x}$ which minimizes $c(\mathbf{x})$, the sum of all squared errors:

$$c(\mathbf{x}) = \sum_l c_l(\mathbf{x}) = \sum_l \frac{1}{2} \mathbf{r}_l^T(\mathbf{x}) \mathbf{r}_l(\mathbf{x}) = \sum_l \sum_k \frac{1}{2} r_{l,k}^2(\mathbf{x}) \tag{A.1}$$

This optimization problem is known as least-squares problem. Residuals are weighted with the inverse of the measurement covariance matrix Σ_l. Here, for clarity sake we have assumed an identity matrix. If we take a second order approximation of $c(\mathbf{x})$ around the current estimate $\mathbf{x}_0$, we get:

$$c(\mathbf{x}_0 + \Delta\mathbf{x}) \approx c(\mathbf{x}_0) + \left.\frac{\partial c}{\partial \mathbf{x}}\right|_{\mathbf{x}_0} \Delta\mathbf{x} + \frac{1}{2} \Delta\mathbf{x}^T \left.\frac{\partial^2 c}{\partial \mathbf{x}^2}\right|_{\mathbf{x}_0} \Delta\mathbf{x} \tag{A.2}$$

Taking the first derivative with respect to the update $\Delta\mathbf{x}$ and setting it to zero to find the singular points (maximum or minimum), we get:

$$\left.\frac{\partial c}{\partial \mathbf{x}}\right|_{\mathbf{x}_0} + \left.\frac{\partial^2 c}{\partial \mathbf{x}^2}\right|_{\mathbf{x}_0} \Delta\mathbf{x} = 0 \tag{A.3}$$

which is the well-known normal equation. If we particularize for our least-squares problem, the first and second derivative of $c(\mathbf{x})$ are computed as:

$$\left(\frac{\partial c}{\partial \mathbf{x}}\right)_i = \frac{\partial c}{\partial x_i} = \sum_l \sum_k \frac{\partial r_{l,k}}{\partial x_i} r_{l,k} \tag{A.4}$$

$$\left(\frac{\partial^2 c}{\partial \mathbf{x}^2}\right)_{i,j} = \sum_l \sum_k \frac{\partial r_{l,k}}{\partial x_i} \frac{\partial r_{l,k}}{\partial x_j} + \underbrace{\frac{\partial^2 r_{l,k}}{\partial x_i \partial x_j} r_{l,k}}_{\text{negligible}} \tag{A.5}$$

These define the gradient vector $\mathbf{b}$ and the Hessian matrix $\mathbf{H}$, such that:

$$\boxed{\mathbf{b} = \frac{\partial c}{\partial \mathbf{x}} = \sum_l \left(\frac{\partial \mathbf{r}_l}{\partial \mathbf{x}}\right)^T \mathbf{r}_l = \sum_l \mathbf{J}_l \mathbf{r}_l} \tag{A.6}$$

$$\boxed{\mathbf{H} = \frac{\partial^2 c}{\partial \mathbf{x}^2} \approx \sum_l \left(\frac{\partial \mathbf{r}_l}{\partial \mathbf{x}}\right)^T \frac{\partial \mathbf{r}_l}{\partial \mathbf{x}} = \sum_l \mathbf{J}_l^T \mathbf{J}_l} \tag{A.7}$$

For the Hessian matrix, we have made the assumption that the second term in (A.5) is negligible with respect to the first one, which is usually true when close to the minimum. This approximation avoids computing second derivatives of residual vectors $\mathbf{r}_l$. We also remark that $\mathbf{H}$ is symmetric by definition.

This renders the optimization problem equivalent to solving the following normal equation:

$$\Delta\mathbf{x} = -\mathbf{H}^{-1}\mathbf{b} \tag{A.8}$$

Both gradient and Hessian are computed at current estimate $\mathbf{x}_0$. Once $\Delta\mathbf{x}$ has been found, we update the current estimate as $\mathbf{x}_0 = \mathbf{x}_0 + \Delta\mathbf{x}$, where + stands for the usual sum for variables belonging to euclidean spaces, while it stands for increment in the tangent space for non-euclidean variables (see appendix D).

Equation A.8 is iteratively solved, recomputing $\mathbf{b}$ and $\mathbf{H}$ whilst the linearization point is updated. This leads to the Gauss-Newton optimization algorithm, which is the gold standard for non-linear least squares optimization. When the Hessian is rank deficient or the estimate is far away from the minimum and second order approximation is inaccurate, this method may fail. Instead of solving equation A.8, a damped least-squares method is used, which interpolates between Gauss-Newton and gradient descent. This is known as Levenberg-Marquardt and should be used when the Hessian matrix is ill-conditioned or residual error is high. This method uses a regularized version of the Hessian matrix, which is equivalent to solving the following equation:

$$\Delta\mathbf{x} = -(\mathbf{H} + \lambda\mathbf{I})^{-1}\mathbf{b} \tag{A.9}$$

where λ is the damping factor which controls its behaviour. When λ is big compared with the Hessian spectral norm, we have a gradient descent method, whilst being small it is equivalent to Gauss-Newton optimization. The Levenberg-Marquardt algorithm describes how to update λ depending on the cost increase/decrease.

Under Gaussian noise assumption (with residuals affected by Gaussian noise), it can be shown that solving this non-linear least-square problem is equivalent to find the Maximum Likelihood estimate of the state vector, which has a strong statistical meaning.

A.2 Bundle Adjustment optimization

In the Bundle Adjustment problem (see sections 2.1 and 2.2), residuals usually depend on a small number of optimizable variables. This makes $\mathbf{J}_l = \partial\mathbf{r}_l/\partial\mathbf{x}$ term to have a very sparse structure which may be exploited to compute the also sparse Hessian $\mathbf{H}_l$, such that $\mathbf{H} = \sum_l \mathbf{H}_l$ (See figure A.1). One should notice that the Hessian component $(\mathbf{H})_{ij}$ will be different from zero only if there exists a measurement which relates i and j variables. These optimizations can be represented by means of factor graphs [36, 68], which depict the variable interaction in a very intuitive way. They have been extensively used along this thesis.

For the specific case of visual SLAM we can split our vector state as $\mathbf{x} = (\mathbf{x}_c, \mathbf{x}_p)$, where $\mathbf{x}_c$ stands for cameras or keyframes parameters and $\mathbf{x}_p$ for points parameters. By accumulating all residuals' Hessians $\mathbf{H}_l$ we get a Hessian matrix $\mathbf{H}$ as depicted in central part of figure A.1. We can identify three blocks:

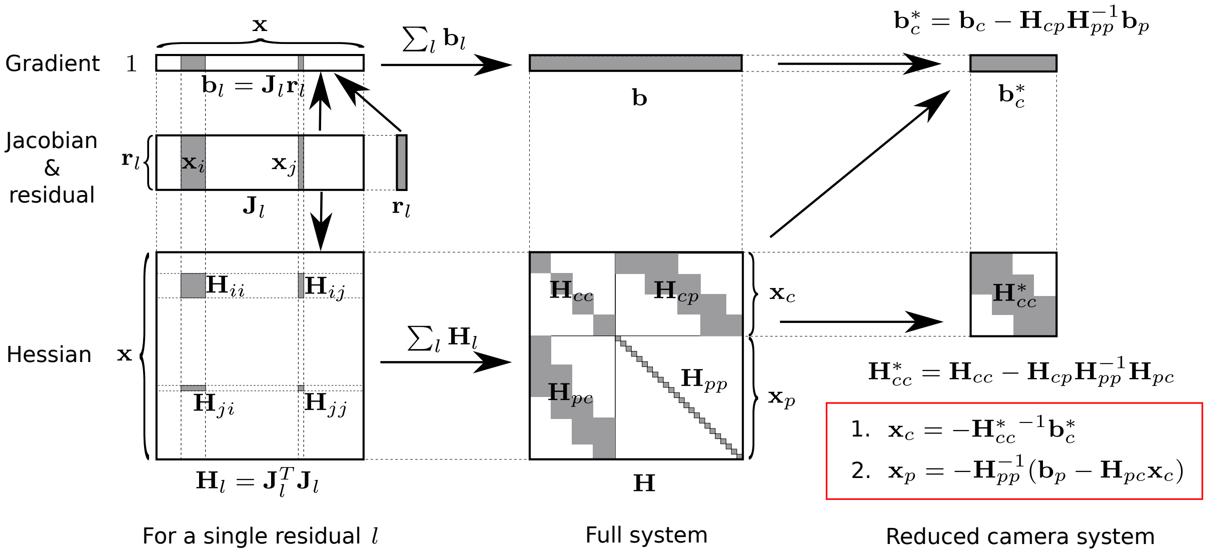

x
Gradient
1
$\mathbf{b}_l = \mathbf{J}_l \mathbf{r}_l$
$\sum_l \mathbf{b}_l$
$\mathbf{b}_c^* = \mathbf{b}_c - \mathbf{H}_{cp}\mathbf{H}_{pp}^{-1}\mathbf{b}_p$
b
$\mathbf{b}_c^*$
Jacobian & residual
$\mathbf{r}_l$
$\mathbf{x}_i$
$\mathbf{x}_j$
$\mathbf{J}_l$
$\mathbf{r}_l$
Hessian
x
$\mathbf{H}_{ii}$
$\mathbf{H}_{ij}$
$\mathbf{H}_{ji}$
$\mathbf{H}_{jj}$
$\sum_l \mathbf{H}_l$
$\mathbf{H}_{cc}$
$\mathbf{H}_{cp}$
$\mathbf{H}_{pc}$
$\mathbf{H}_{pp}$
$\mathbf{x}_c$
$\mathbf{x}_p$
$\mathbf{H}_{cc}^*$
$\mathbf{H}_{cc}^* = \mathbf{H}_{cc} - \mathbf{H}_{cp}\mathbf{H}_{pp}^{-1}\mathbf{H}_{pc}$
1. $\mathbf{x}_c = -\mathbf{H}_{cc}^{*\,-1}\mathbf{b}_c^*$
2. $\mathbf{x}_p = -\mathbf{H}_{pp}^{-1}(\mathbf{b}_p - \mathbf{H}_{pc}\mathbf{x}_c)$
$\mathbf{H}_l = \mathbf{J}_l^T \mathbf{J}_l$
H
For a single residual l
Full system
Reduced camera system

Figure A.1: Non-zero components are grey coloured. **Left**: Given a visual residual depending only on keyframe i and point j variables, we can use its residual $\mathbf{r}_l$ and Jacobian $\mathbf{J}_l$ to compute its sparse gradient $\mathbf{b}_l$ and Hessian $\mathbf{H}_l$. **Center**: Accumulating for all residual, we get a sparse Hessian matrix $\mathbf{H}$ and a dense gradient vector $\mathbf{b}$ which define the complete system $\mathbf{Hx} = -\mathbf{b}$. **Right**: Instead of solving the complete system, we apply the Schur complement to obtain the reduced camera system. We solve it for the keyframe variables and later obtain the point variables by back-substitution.

- $\mathbf{H}_{cc}$ stands for the camera-camera block. When using a xyz parameterization, this block is diagonal by blocks (as assumed in figure A.1), since a camera only shares residuals with itself. However, if an inverse depth parameterization is used, with an anchor or host keyframe, this block will have off-diagonal terms.

- $\mathbf{H}_{cp} = \mathbf{H}_{pc}^T$ are camera-point blocks. The sparsity of these blocks depends on how meshed is the covisibility graph. For Structure from Motion, where points may be observed from all cameras, they are dense. However, for SLAM applications, where points are only observed from a small set of keyframes, these blocks are sparse, as assumed in figure A.1.

- $\mathbf{H}_{pp}$ stands for the point-point block. Since a point only shares residuals with cameras but not with other points, this block is diagonal. If xyz parameterization, it contains 3×3 blocks along the diagonal. This fact will be exploited to efficiently compute the Hessian inverse.

A.2.1 Schur complement for an efficient solution

For SLAM applications, the number of points' parameters, N_p, is much larger than the number of keyframes' parameters, N_c. Typically we optimize hundreds or thousands of points while only a few tens of keyframes. Directly solving equation A.8 would yield a high computational cost, $\mathcal{O}((N_c + N_p)^3)$. However, we have seen that $\mathbf{H}_{pp}$ has a diagonal layout that can be taken advantage of using inversion by blocks. In fact, an efficient solution can be easily derived using the matrix inversion lemma [133], obtaining a reduced camera system where only camera variables are involved:

$$\Delta \mathbf{x}_c = -{\mathbf{H}_{cc}^{*}}^{-1} \mathbf{b}_c^{*} \tag{A.10}$$

with:

$$\mathbf{H}_{cc}^{*} = \mathbf{H}_{cc} - \mathbf{H}_{cp}\mathbf{H}_{pp}^{-1}\mathbf{H}_{pc} \tag{A.11}$$

$$\mathbf{b}_c^{*} = \mathbf{b}_c - \mathbf{H}_{cp}\mathbf{H}_{pp}^{-1}\mathbf{b}_p \tag{A.12}$$

where $\mathbf{H}_{cc}^{*}$ is known as the Schur complement of $\mathbf{H}_{cc}$. The diagonal structure of $\mathbf{H}_{pp}$ allows us to bypass most of the cost of this reduced system construction. Computing $\mathbf{H}_{pp}^{-1}$ has a $\mathcal{O}(N_p)$ cost, while computing the matrix product $\mathbf{H}_{cp}\mathbf{H}_{pp}^{-1}\mathbf{H}_{pc}$ takes less than $\mathcal{O}(N_c^2 N_p)$ considering it is not fully dense. This yields an important cost reduction with respect to original solution since $N_p \gg N_c$. Once solved the reduced system, we can retrieve point parameters as:

$$\mathbf{x}_p = -\mathbf{H}_{pp}^{-1}(\mathbf{b}_p - \mathbf{H}_{pc}\mathbf{x}_c) \tag{A.13}$$

This has a $\mathcal{O}(N_p N_c)$ cost. Altogether, using the Schur complement trick, we have reduced the Bundle Adjustment cost from $\mathcal{O}((N_c + N_p)^3)$ to $\mathcal{O}(N_c^2 N_p)$, which is a substantial reduction since $N_p \gg N_c$. The biggest cost comes from building the reduced camera system, while the exact cost will depend on how meshed is the covisibility graph, i.e. how dense matrices $\mathbf{H}_{cc}^{*}$ and $\mathbf{H}_{cp}$ are. Building the optimization problem has also an important cost that can not be neglected.

Appendix B

Bundle Adjustment residuals

In this appendix we introduce the most important residuals used for Bundle Adjustment. We first present some camera models and their projection functions to later compute their derivatives, necessary for the optimization. Next, we introduce the two most common visual residuals used for BA, namely reprojection and photometric residuals, and we compute their derivatives. To cover the two point parameterizations, *xyz* and inverse depth, the former will be used for reprojection residual while the latter for photometric residual. We finally present depth prediction residuals from section 5.2 as well as inertial preintegrated terms used for section 3.3.

B.1 Camera models

We analyze here different camera models, mainly interested in their projection functions $\pi : \mathbb{R}^3 \rightarrow \Omega$ as well as their derivatives with respect to the 3D point position $\mathbf{x}$ in the camera reference.

B.1.1 Pinhole camera

The projection function for a distortionless pinhole camera is:

$$\boxed{\mathbf{u} = \pi(\mathbf{x}) = \begin{pmatrix} f_x x/z + c_x \\ f_y y/z + c_y \end{pmatrix}} \tag{B.1}$$

where $\mathbf{x}$ stands for the 3D point coordinates in the camera reference. We immediately obtain the derivative wrt the 3D point:

$$\boxed{\frac{\partial \pi(\mathbf{x})}{\partial \mathbf{x}} = \begin{pmatrix} f_x/z & 0 & -f_x x/z^2 \\ 0 & f_y/z & -f_y y/z^2 \end{pmatrix}} \tag{B.2}$$

B.1.2 Fisheye camera

The projection function for a fisheye camera, using the Kannala-Brandt model [70] is:

$$\mathbf{u} = \pi(\mathbf{x}) = \begin{pmatrix} f_x x'/z + c_x \\ f_y y'/z + c_y \end{pmatrix} \tag{B.3}$$

where the following intermediate terms are used:

$$\begin{aligned}
\bar{x} &= x/z \\
\bar{y} &= y/z \\
r &= \sqrt{\bar{x}^2 + \bar{y}^2} \\
\theta &= atan(r) \\
\theta_d &= \theta(1 + k_1\theta^2 + k_2\theta^4 + k_3\theta^6 + k_4\theta^8) \\
x' &= \theta_d x/r \\
y' &= \theta_d y/r \\
z' &= z
\end{aligned} \tag{B.4}$$

We compute now derivatives with respect to the 3D point $\mathbf{x}$ at camera reference:

$$\frac{\partial \pi(\mathbf{x})}{\partial \mathbf{x}} = \frac{\partial \pi(\mathbf{x})}{\partial \mathbf{x}'} \frac{\partial \mathbf{x}'}{\partial \mathbf{x}} \tag{B.5}$$

where $\partial \pi(\mathbf{x})/\partial \mathbf{x}'$ is the Jacobian of the pinhole camera, while the 3×3 matrix $\partial \mathbf{x}'/\partial \mathbf{x}$ needs to be computed. For simplicity, we will use index notation, such that $\mathbf{x} = (x_1, x_2, x_3)$. In this sense, for the third component we have:

$$\frac{\partial x_i'}{\partial x_j} = \delta_j^i \quad \text{for} \quad i = 3 \tag{B.6}$$

while for the first two:

$$\begin{aligned}
\frac{\partial x_i'}{\partial x_j} &= \frac{\partial \theta_d}{\partial x_j} \frac{x_i}{r} + \theta_d \left(\frac{\partial x_i}{\partial x_j} \frac{1}{r} - \frac{x_i}{r^2} \frac{\partial r}{\partial x_j} \right) \\
&= \frac{\partial \theta_d}{\partial \theta} \frac{\partial \theta}{\partial r} \frac{\partial r}{\partial x_j} \frac{x_i}{r} + \theta_d \left(\frac{\delta_j^i}{r} - \frac{x_i}{r^2} \frac{\partial r}{\partial x_j} \right) \quad \text{for} \quad i = 1, 2
\end{aligned} \tag{B.7}$$

The intermediate derivatives are computed as:

$$\begin{aligned}
\frac{\partial \theta_d}{\partial \theta} &= 1 + 3k_1\theta^2 + 5k_2\theta^4 + 7k_3\theta^6 + 9k_4\theta^8 \\
\frac{\partial \theta}{\partial r} &= \frac{1}{1 + r^2} \\
\frac{\partial r}{\partial x_j} &= \frac{1}{r} \left(\bar{x} \frac{\partial \bar{x}}{\partial x_j} + \bar{y} \frac{\partial \bar{y}}{\partial x_j} \right)
\end{aligned} \tag{B.8}$$

B.2 Reprojection residual

For keyframe i observing point j, we define the reprojection error:

$$\boxed{\mathbf{r}_{i,\text{proj}}^j = \mathbf{u}_i^j - \pi(\mathbf{x}_i^j)} \in \mathbb{R}^2 \quad \text{with} \quad \mathbf{x}_i^j = \mathbf{T}_{i\mathtt{W}} \oplus \mathbf{x}^j \tag{B.9}$$

where $\mathbf{x}_i^j$ denotes j point position in the i camera reference. This error may be seen as the difference in pixels between the predicted projection $\pi(\mathbf{x}_i^j)$ and the measurement $\mathbf{u}_i^j$. Projection map π may take any expression, and particularly those presented in the previous section B.1.

B.2.1 Reprojection residual derivatives

Assuming a *xyz* representation, this residual will depend on the observer keyframe pose $\mathbf{T}_{i\mathtt{W}} = [\mathbf{R}_{i\mathtt{W}}, \mathbf{t}_{i\mathtt{W}}]$ and the global j point position $\mathbf{x}^j$. Next, we compute the derivatives with respect to them.

B.2.1.1 Derivatives w.r.t. point position

For global point position $\mathbf{x}^j$ we obtain the following expression:

$$\begin{aligned}
\frac{\partial \mathbf{r}^j_{i,\text{proj}}}{\partial \mathbf{x}^j} &= -\left.\frac{\partial \pi(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}^j_i} \frac{\partial \mathbf{T}_{i\mathtt{W}} \oplus \mathbf{x}^j}{\partial \mathbf{x}^j} \\
&= -\left.\frac{\partial \pi(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}^j_i} \frac{\partial (\mathbf{R}_{i\mathtt{W}}\mathbf{x}^j + \mathbf{t}_{i\mathtt{W}})}{\partial \mathbf{x}^j}
\end{aligned}$$

$$\boxed{\frac{\partial \mathbf{r}^j_{i,\text{proj}}}{\partial \mathbf{x}^j} = -\left.\frac{\partial \pi(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}^j_i} \mathbf{R}_{i\mathtt{W}}} \in \mathbb{R}^{2\times 3} \tag{B.10}$$

where $\left.\frac{\partial \pi(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}^j_i}$ has been derived in the previous section.

B.2.1.2 Derivatives w.r.t. observer pose

Poses $\mathbf{T}_{i\mathtt{W}}$ belong to Lie group SE(3), which is a smooth manifold, locally but not globally Euclidean, as explained in appendix D. This implies that we can not directly compute the derivative with respect to absolute pose. Instead, we apply a small update $\xi = (\xi_\phi, \xi_t)$ in the local reference (left hand side of the pose), and we compute derivatives with respect to it. ξ_ϕ stands for the rotational part whilst ξ_t stands for the translation update, obtaining:

$$\begin{aligned}
\frac{\partial \mathbf{r}^j_{i,\text{proj}}}{\partial \xi} &= -\left.\frac{\partial \pi(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}^j_i} \left.\frac{\partial (\text{Exp}(\xi) \oplus \mathbf{T}_{i\mathtt{W}} \oplus \mathbf{x}^j)}{\partial \xi}\right|_{\xi=\mathbf{0}} \\
&= -\left.\frac{\partial \pi(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}^j_i} \left.\frac{\partial (\text{Exp}(\xi) \oplus \mathbf{x}^j_i)}{\partial \xi}\right|_{\xi=\mathbf{0}}
\end{aligned} \tag{B.11}$$

The exponential map (See appendix section D.1.2) for SE(3) is defined as follows [123]:

$$\text{Exp}(\xi) = \begin{pmatrix} \text{Exp}(\xi_\phi) & \mathbf{V}(\xi_\phi)\xi_t \\ \mathbf{0} & 1 \end{pmatrix} \in \text{SE}(3) \tag{B.12}$$

If ξ is a small update, as it is in our case, we can approximate the SO(3) exponential map as (See equation D.32):

$$\text{Exp}(\xi_\phi) \approx \mathbf{I} + [\xi_\phi]_\times \tag{B.13}$$

and (see [123]):

$$\mathbf{V}(\xi_\phi) \approx \mathbf{I} - 1/2[\xi_\phi]_\times \tag{B.14}$$

where $[\cdot]_\times$ is the skew operator from $\mathbb{R}^3$ to $\mathfrak{so}(3)$, as defined in equation D.28. With these approximations, we can finally compute derivatives such that:

$$\begin{aligned}\frac{\partial \mathbf{r}^j_{i,\text{proj}}}{\partial \xi} &= -\left.\frac{\partial \pi(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}^j_i} \left.\frac{\partial((\mathbf{I}+[\xi_\phi]_\times)\mathbf{x}^j_i + (\mathbf{I}-1/2[\xi_\phi]_\times)\xi_t)}{\partial \xi}\right|_{\xi=\mathbf{0}} \\ &= -\left.\frac{\partial \pi(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}^j_i} \left(\left.\frac{\partial[\xi_\phi]_\times(\mathbf{x}^j_i - 1/2\xi_t)}{\partial \xi_\phi}\right|_{\xi=\mathbf{0}}, \left.\frac{\partial(\mathbf{I}-1/2[\xi_\phi]_\times)\xi_t}{\partial \xi_t}\right|_{\xi=\mathbf{0}} \right)\end{aligned}$$

$$\boxed{\frac{\partial \mathbf{r}^j_{i,\text{proj}}}{\partial \xi} = -\left.\frac{\partial \pi(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}^j_i} \left(-[\mathbf{x}^j_i]_\times, \mathbf{I}\right)} \in \mathbb{R}^{2\times 6} \tag{B.15}$$

where we have used $\partial[\xi]_\times \mathbf{x}/\partial\xi = -[\mathbf{x}]_\times$ (See appendix section D.2.5 for derivation). Given a camera model, an explicit expression may be found for equation B.15, probably leading to some computational saving since simplifications may exist.

For visual-inertial case, where the pose $\mathbf{T}_{i\mathtt{W}}$ to be optimized corresponds with the IMU body $\mathtt{B}$, a transformation $\mathbf{T}_{\mathtt{BC}}$ between body and camera is required. In this case, we have:

$$\begin{aligned}\frac{\partial \mathbf{r}^j_{i,\text{proj}}}{\partial \xi} &= -\left.\frac{\partial \pi(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}^j_i} \left.\frac{\partial(\mathbf{T}_{\mathtt{CB}} \oplus \text{Exp}(\xi) \oplus \mathbf{x}^j_{\mathtt{B}_i})}{\partial \xi}\right|_{\xi=\mathbf{0}} \\ &= -\left.\frac{\partial \pi(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}^j_i} \mathbf{R}_{\mathtt{CB}} \left.\frac{\partial(\text{Exp}(\xi) \oplus \mathbf{x}^j_{\mathtt{B}_i})}{\partial \xi}\right|_{\xi=\mathbf{0}} \\ &= -\left.\frac{\partial \pi(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}^j_i} \mathbf{R}_{\mathtt{CB}} \left(-[\mathbf{x}^j_{\mathtt{B}_i}]_\times, \mathbf{I}\right)\end{aligned} \tag{B.16}$$

where $\mathbf{x}^j_{\mathtt{B}_i}$ stands for the 3D point position in the i body reference. Equivalent expression but with point coordinates in camera reference could be derived using the adjoint matrix $\mathbf{Ad}_{\mathbf{T}_{\mathtt{CB}}}$. In the particular case when $\mathbf{R}_{\mathtt{CB}}$ is the identity, we retrieve the previous expression B.15, as expected.

B.3 Photometric residual

The second visual residual is the photometric error. This can be seen as the difference of intensity values between image pixels which should correspond to the same 3D point. For a map point j observed from frame i we can define the following $r^j_{i,\text{photo}}$ photometric error:

$$\boxed{r^j_{i,\text{photo}} = I_h(\mathbf{u}^j_h) - b_h - \frac{e^{a_h}}{e^{a_i}}(I_i(\mathbf{u}^j_i) - b_i)} \in \mathbb{R} \tag{B.17}$$

where:

- I_h, I_i: Images at host h and i observer frames.
- $\mathbf{u}^j_h, \mathbf{u}^j_i$: Point projection at host and i observer frames. We remark $\mathbf{u}^j_h$ is constant since we follow parameterization from [46] instead of [29], while $\mathbf{u}^j_i = \pi(\mathbf{x}^j_i)$ it is not.

- a_h, b_h, a_i, b_i: Affine brightness parameters for host and i observer frames, as defined in [46], and necessary for luminosity changes.

Here we will use an inverse depth parametrization for the 3D map point. Thereby, $\mathbf{x}^j$ can be computed as a function of the unitary vector $\bar{\mathbf{x}}_h^j$ in the host/anchor (h) reference pointing to the point, and the inverse depth ρ^j in the h reference, such that $\mathbf{x}_h^j = \bar{\mathbf{x}}_h^j {\rho^j}^{-1}$. This finally yields:

$$\mathbf{u}_i^j = \pi(\mathbf{x}_i^j) = \pi(\mathbf{T}_{ih} \oplus (\bar{\mathbf{x}}_h^j {\rho^j}^{-1})) = \pi({\rho^j}^{-1}\mathbf{R}_{ih}\bar{\mathbf{x}}_h^j + \mathbf{t}_{ih}) \tag{B.18}$$

where $\mathbf{T}_{ih} = [\mathbf{R}_{ih}, \mathbf{t}_{ih}]$ stands for transformation from host to i observer frame. $\mathbf{T}_{ih} = \mathbf{T}_{i\mathtt{W}} \oplus \mathbf{T}_{h\mathtt{W}}^{-1}$. The problem of equation B.18 lies in the division by the inverse depth, which leads to stability issues for far points. One way to solve this consists in leveraging the fact that $\pi(\mathbf{x}) = \pi(c\mathbf{x})$. In this way, we can scale $\mathbf{x}_i^j$ with ρ^j, such that:

$$\boxed{\mathbf{u}_i^j = \pi(\rho^j\mathbf{x}_i^j) = \pi(\rho^j[\mathbf{T}_{ih} \oplus (\bar{\mathbf{x}}_h^j {\rho^j}^{-1})]) = \pi(\mathbf{R}_{ih}\bar{\mathbf{x}}_h^j + \rho^j\mathbf{t}_{ih})} \tag{B.19}$$

where inverse depth ρ^j does not appear dividing. Hence, we will use expression B.19 for computing equation B.17 and its derivatives in the following sections.

We have that this photometric residual depends on point inverse depth ρ^j, host $\mathbf{T}_{h\mathtt{W}}$ and observer $\mathbf{T}_{i\mathtt{W}}$ poses, as well as their affine parameters a_h, b_h, a_i, b_i. We will need to compute the derivatives with respect to each of them to build our photometric Bundle Adjustment.

B.3.1 Photometric residual derivatives

In this section we derive Jacobians for each of the optimizable variables.

B.3.1.1 Derivatives w.r.t. affine brightness parameters

We start with the brightness affine parameters a_h, b_h, a_i, b_i. It can be immediately found that:

$$\frac{\partial r_{i,\text{photo}}^j}{\partial a_h} = -\frac{e^{a_h}}{e^{a_i}}(I_i(\mathbf{u}_i^j) - b_i) \tag{B.20}$$

$$\frac{\partial r_{i,\text{photo}}^j}{\partial b_h} = -1 \tag{B.21}$$

$$\frac{\partial r_{i,\text{photo}}^j}{\partial a_i} = -\frac{\partial r_{i,\text{photo}}^j}{\partial a_h} \tag{B.22}$$

$$\frac{\partial r_{i,\text{photo}}^j}{\partial b_i} = \frac{e^{a_h}}{e^{a_i}} \tag{B.23}$$

B.3.1.2 Derivatives w.r.t. point inverse depth

Now, we derive Jacobians for inverse depth ρ^j, such that applying chain rule we obtain:

$$\begin{aligned}\frac{\partial r_{i,\text{photo}}^j}{\partial \rho^j} &= -\frac{e^{a_h}}{e^{a_i}}\frac{\partial I_i(\mathbf{u}_i^j)}{\partial \rho^j} \\ &= -\frac{e^{a_h}}{e^{a_i}}\left.\frac{\partial I_i}{\partial \mathbf{u}}\right|_{\mathbf{u}_i^j}\left.\frac{\partial \pi(\mathbf{x})}{\partial \mathbf{x}}\right|_{\rho^j\mathbf{x}_i^j}\frac{\partial(\mathbf{R}_{ih}\bar{\mathbf{x}}_h^j + \rho^j\mathbf{t}_{ih})}{\partial \rho^j}\end{aligned}$$

$$\boxed{\frac{\partial r^j_{i,\text{photo}}}{\partial \rho^j} = -\frac{e^{a_h}}{e^{a_i}} \left.\frac{\partial I_i}{\partial \mathbf{u}}\right|_{\mathbf{u}^j_i} \left.\frac{\partial \pi(\mathbf{x})}{\partial \mathbf{x}}\right|_{\rho^j\mathbf{x}^j_i} \mathbf{t}_{ih}} \in \mathbb{R} \tag{B.24}$$

where $\partial I_i / \partial \mathbf{u} \in \mathbb{R}^2$ stands for the image gradient, while the derivative of the projection map has been already derived.

B.3.1.3 Derivatives w.r.t. frame poses

In contrast with equation B.15, where the pose update was applied in the local reference frame (right hand side of the pose), here, we decide to apply it in the global reference frame, which will lead to some computational saving. In particular, when computing the Hessian, several terms will only defer on sign, making its construction much more efficient. First, we compute with respect to host pose:

$$\frac{\partial r^j_{i,\text{photo}}}{\partial \xi_h} = -\frac{e^{a_h}}{e^{a_i}} \left.\frac{\partial I_i}{\partial \mathbf{u}}\right|_{\mathbf{u}^j_i} \left.\frac{\partial \pi(\mathbf{x})}{\partial \mathbf{x}}\right|_{\rho^j\mathbf{x}^j_i} \left.\frac{\partial (\rho^j \mathbf{x}^j_i)}{\partial \xi_h}\right|_{\xi=0} \tag{B.25}$$

where:

$$\begin{aligned}
\left.\frac{\partial (\rho^j \mathbf{x}^j_i)}{\partial \xi_h}\right|_{\xi=0} &= \rho^j \left.\frac{\partial (\mathbf{T}_{ih} \oplus \mathbf{x}^j_h)}{\partial \xi_h}\right|_{\xi=0} \\
&= \rho^j \left.\frac{\partial (\mathbf{T}_{i\mathtt{W}} \oplus (\mathbf{T}_{h\mathtt{W}} \oplus \operatorname{Exp}(\xi_h))^{-1} \oplus \mathbf{x}^j_h)}{\partial \xi_h}\right|_{\xi=0} \\
&= \rho^j \left.\frac{\partial (\mathbf{T}_{i\mathtt{W}} \oplus \operatorname{Exp}(-\xi_h) \oplus \mathbf{T}_{\mathtt{W}h} \oplus \mathbf{x}^j_h)}{\partial \xi_h}\right|_{\xi=0} \\
&= \rho^j \left.\frac{\partial (\operatorname{Exp}(-\mathbf{Ad}_{\mathbf{T}_{i\mathtt{W}}}\xi_h) \oplus \mathbf{x}^j_i)}{\partial \xi_h}\right|_{\xi=0} \\
&= -\rho^j (\mathbf{I}_3 | - [\mathbf{x}^j_i]_\times) \mathbf{Ad}_{\mathbf{T}_{i\mathtt{W}}} \in \mathbb{R}^{3\times 6}
\end{aligned}$$

This finally leads to:

$$\boxed{\frac{\partial r^j_{i,\text{photo}}}{\partial \xi_h} = \rho^j \frac{e^{a_h}}{e^{a_i}} \left.\frac{\partial I_i}{\partial \mathbf{u}}\right|_{\mathbf{u}^j_i} \left.\frac{\partial \pi(\mathbf{x})}{\partial \mathbf{x}}\right|_{\rho^j\mathbf{x}^j_i} (\mathbf{I}_3 | - [\mathbf{x}^j_i]_\times) \mathbf{Ad}_{\mathbf{T}_{i\mathtt{W}}}} \in \mathbb{R}^6 \tag{B.26}$$

First, we have used the following property $\operatorname{Exp}(\xi)^{-1} = \operatorname{Exp}(-\xi)$ (See equation D.33 for derivation). Second, we have leverage the adjoint matrix $\mathbf{Ad}_{\mathbf{X}}$ (see section D.1.3):

$$\operatorname{Exp}(\mathbf{a}) \oplus \mathbf{X} = \mathbf{X} \oplus \operatorname{Exp}(\mathbf{b}) \longrightarrow \mathbf{a} = \mathbf{Ad}_{\mathbf{X}} \mathbf{b} \tag{B.27}$$

For $\mathbf{T} = [\mathbf{R}, \mathbf{t}] \in \mathrm{SE}(3)$, the adjoint matrix takes following expression [123]:

$$\mathbf{Ad}_{\mathbf{T}} = \begin{bmatrix} \mathbf{R} & [\mathbf{t}]_\times \mathbf{R} \\ \mathbf{0} & \mathbf{R} \end{bmatrix} \tag{B.28}$$

Now, we compute the derivative with respect to observer frame. In the same way, we write:

$$\frac{\partial r^j_{i,\text{photo}}}{\partial \xi_i} = -\frac{e^{a_h}}{e^{a_i}} \left.\frac{\partial I_i}{\partial \mathbf{u}}\right|_{\mathbf{u}^j_i} \left.\frac{\partial \pi(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}^j_i} \left.\frac{\partial (\rho^j \mathbf{x}^j_i)}{\partial \xi_i}\right|_{\xi=0} \tag{B.29}$$

where:

$$\begin{aligned}
\left.\frac{\partial(\rho^j \mathbf{x}_i^j)}{\partial \xi_i}\right|_{\xi=0} &= \rho^j \left.\frac{\partial(\mathbf{T}_{ih} \oplus \mathbf{x}_h^j)}{\partial \xi_h}\right|_{\xi=0} \\
&= \rho^j \left.\frac{\partial(\mathbf{T}_{i\mathtt{W}} \oplus \operatorname{Exp}(\xi_i) \oplus \mathbf{T}_{\mathtt{W}h} \oplus \mathbf{x}_h^j)}{\partial \xi_h}\right|_{\xi=0} \\
&= \rho^j \left.\frac{\partial(\operatorname{Exp}(\mathbf{Ad}_{\mathbf{T}_{i\mathtt{W}}}\xi_i) \oplus \mathbf{x}_i^j)}{\partial \xi_h}\right|_{\xi=0} \\
&= \rho^j (\mathbf{I}_3 | -[\mathbf{x}_i^j]_\times)\mathbf{Ad}_{\mathbf{T}_{i\mathtt{W}}}
\end{aligned} \tag{B.30}$$

finally leading to:

$$\boxed{\frac{\partial r^j_{i,\text{photo}}}{\partial \xi_i} = -\rho^j \frac{e^{a_h}}{e^{a_i}} \left.\frac{\partial I_i}{\partial \mathbf{u}}\right|_{\mathbf{u}_i^j} \left.\frac{\partial \pi(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_i^j} (\mathbf{I}_3 | -[\mathbf{x}_i^j]_\times)\mathbf{Ad}_{\mathbf{T}_{i\mathtt{W}}}} \in \mathbb{R}^6 \tag{B.31}$$

Derivatives for both frames only differ on sign, which greatly simplifies computations. However, a problem arises in its linear dependence with respect to the absolute position $\mathbf{t}_{i\mathtt{W}}$, by means of the adjoint matrix. For frames far from the origin, this may cause problems. To avoid this, all keyframes may be translated to a new reference close to the origin.

If a pattern of neighbors is used, as explained in section 2.1.2, instead of a scalar residual, we will have a vector of residuals. For simplicity, as proposed in [46], we will assume that geometrical parts of the Jacobians (B.24), (B.26) and (B.31) are constant for all pixels in the patch. We will only consider different the photometric derivative, $\left.\frac{\partial I_i}{\partial \mathbf{u}}\right|_{\mathbf{u}_i^j}$.

B.3.1.4 Derivatives with local update

If instead of applying the update in the world reference, we apply it in the camera reference, derivatives for both frames would not have so similar expressions, and more computations would be required to compute Hessian. In fact, for equations B.25 and B.29, the only changes would be in the last terms, such that:

$$\begin{aligned}
\left.\frac{\partial(\rho^j \mathbf{x}_i^j)}{\partial \xi_h}\right|_{\xi=0} &= \rho^j \left.\frac{\partial(\mathbf{T}_{i\mathtt{W}} \oplus (\operatorname{Exp}(\xi_h) \oplus \mathbf{T}_{h\mathtt{W}})^{-1} \oplus \mathbf{x}_h^j)}{\partial \xi_h}\right|_{\xi=0} \\
&= \rho^j \left.\frac{\partial(\operatorname{Exp}(-\mathbf{Ad}_{\mathbf{T}_{ih}}\xi_h) \oplus \mathbf{x}_i^j)}{\partial \xi_h}\right|_{\xi=0} \\
&= -\rho^j (\mathbf{I}_3 | -[\mathbf{x}_i^j]_\times)\mathbf{Ad}_{\mathbf{T}_{ih}} \in \mathbb{R}^{3\times 6}
\end{aligned} \tag{B.32}$$

$$\begin{aligned}
\left.\frac{\partial(\rho^j \mathbf{x}_i^j)}{\partial \xi_i}\right|_{\xi=0} &= \rho^j \left.\frac{\partial(\operatorname{Exp}(\xi_i) \oplus \mathbf{T}_{i\mathtt{W}} \oplus \mathbf{T}_{\mathtt{W}h} \oplus \mathbf{x}_h^j)}{\partial \xi_h}\right|_{\xi=0} \\
&= \rho^j (\mathbf{I}_3 | -[\mathbf{x}_i^j]_\times)
\end{aligned} \tag{B.33}$$

We can see that host and observer derivatives not only defer in sign, but also the former is multiplied by $\mathbf{Ad}_{\mathbf{T}_{ih}}$, while the latter is not. This leads to more different components along the camera i and camera h Hessian block, making its computation more expensive. The advantage is that $\mathbf{Ad}_{\mathbf{T}_{ih}}$ is well bounded, since distance between host and observer does not take very big values.

B.4 Depth-prediction residual

As described in section 5.2, we define the depth-prediction residual for point j seen from keyframe i as:

$$r^j_{i,\text{depth}} = D^i_{NN}(\mathbf{u}^j_i) - \rho^j_i = \quad \text{where} \quad \rho^j_i = [\mathbf{x}^j_i]_z^{-1} \tag{B.34}$$

B.4.1 Depth-prediction derivatives

B.4.1.1 Derivatives w.r.t. point inverse depth

For obtaining these derivatives, most important computations may be recovered from photometric residual derivatives. Following same procedure than for equation B.24, we immediately get:

$$\frac{\partial r^j_{i,\text{depth}}}{\partial \rho^j} = \left.\frac{\partial D^i_{NN}}{\partial \mathbf{u}}\right|_{\mathbf{u}^j_i} \left.\frac{\partial \pi(\mathbf{x})}{\partial \mathbf{x}}\right|_{\rho^j \mathbf{x}^j_i} \mathbf{t}_{ih} - \frac{\partial \rho^j_i}{\partial \rho^j} \tag{B.35}$$

We now compute $\partial \rho^j_i / \partial \rho^j$ as:

$$\begin{aligned}\frac{\partial \rho^j_i}{\partial \rho^j} &= \frac{\partial [\mathbf{x}^j_i]_z^{-1}}{\partial \mathbf{x}^j_i} \frac{\partial \mathbf{x}^j_i}{\partial \rho^j_i} \\ &= (0, 0, -[\mathbf{x}^j_i]_z^{-2}) \frac{\partial \left(\mathbf{R}_{ih}\bar{\mathbf{x}}^j_h/\rho^j + \mathbf{t}_{ih}\right)}{\partial \rho^j} \\ &= \left(\frac{\rho^j_i}{\rho^j}\right)^2 [\mathbf{R}_{ih}\bar{\mathbf{x}}^j_h]_z\end{aligned}$$

finally leading to:

$$\boxed{\frac{\partial r^j_{i,\text{depth}}}{\partial \rho^j} = \left.\frac{\partial D^i_{NN}}{\partial \mathbf{u}}\right|_{\mathbf{u}^j_i} \left.\frac{\partial \pi(\mathbf{x})}{\partial \mathbf{x}}\right|_{\rho^j \mathbf{x}^j_i} \mathbf{t}_{ih} - \left(\frac{\rho^j_i}{\rho^j}\right)^2 [\mathbf{R}_{ih}\bar{\mathbf{x}}^j_h]_z} \tag{B.36}$$

B.4.1.2 Derivatives w.r.t. keyframes pose

First, with respect to the host pose. Immediately from equation B.31 it follows:

$$\frac{\partial r^j_{i,\text{depth}}}{\partial \xi_h} = -\rho^j \left.\frac{\partial D^i_{NN}}{\partial \mathbf{u}}\right|_{\mathbf{u}^j_i} \left.\frac{\partial \pi(\mathbf{x})}{\partial \mathbf{x}}\right|_{\rho^j \mathbf{x}^j_i} (\mathbf{I}_3| - [\mathbf{x}^j_i]_\times)\mathbf{Ad}_{\mathbf{T}_{i\mathtt{W}}} - \frac{\partial \rho^j_i}{\partial \xi_h} \tag{B.37}$$

Now, we compute $\partial \rho^j_i / \partial \xi_h$, and following same procedure than for equation B.30:

$$\frac{\partial \rho^j_i}{\partial \xi_h} = \frac{\partial [\mathbf{x}^j_i]_z^{-1}}{\partial \mathbf{x}^j_i} \frac{\partial \mathbf{x}^j_i}{\partial \xi_h} = (0, 0, [\mathbf{x}^j_i]_z^{-2})(\mathbf{I}_3| - [\mathbf{x}^j_i]_\times)\mathbf{Ad}_{\mathbf{T}_{i\mathtt{W}}}$$

This finally leads to:

$$\boxed{\frac{\partial r^j_{i,\text{depth}}}{\partial \xi_h} = -\left(\rho^j \left.\frac{\partial D^i_{NN}}{\partial \mathbf{u}}\right|_{\mathbf{u}^j_i} \left.\frac{\partial \pi(\mathbf{x})}{\partial \mathbf{x}}\right|_{\rho^j \mathbf{x}^j_i} + (0,0,\rho^{j^2}_i)\right)(\mathbf{I}_3| - [\mathbf{x}^j_i]_\times)\mathbf{Ad}_{\mathbf{T}_{i\mathtt{W}}}} \tag{B.38}$$

For the observer, we proceed in a similar way, leading to:

$$\frac{\partial r^j_{i,\text{depth}}}{\partial \xi_i} = \rho^j \left.\frac{\partial D^i_{NN}}{\partial \mathbf{u}}\right|_{\mathbf{u}^j_i} \left.\frac{\partial \pi(\mathbf{x})}{\partial \mathbf{x}}\right|_{\rho^j \mathbf{x}^j_i} (\mathbf{I}_3| - [\mathbf{x}^j_i]_\times)\mathbf{Ad}_{\mathbf{T}_{i\mathtt{W}}} - \frac{\partial \rho^j_i}{\partial \xi_i} \tag{B.39}$$

where:

$$\frac{\partial \rho^j_i}{\partial \xi_i} = \frac{\partial [\mathbf{x}^j_i]^{-1}_z}{\partial \mathbf{x}^j_i}\frac{\partial \mathbf{x}^j_i}{\partial \xi_i} = -(0,0,[\mathbf{x}^j_i]^{-2}_z)(\mathbf{I}_3| - [\mathbf{x}^j_i]_\times)\mathbf{Ad}_{\mathbf{T}_{i\mathtt{W}}}$$

which finally leads to:

$$\boxed{\frac{\partial r^j_{i,\text{depth}}}{\partial \xi_i} = \left(\rho^j \left.\frac{\partial D^i_{NN}}{\partial \mathbf{u}}\right|_{\mathbf{u}^j_i} \left.\frac{\partial \pi(\mathbf{x})}{\partial \mathbf{x}}\right|_{\rho^j \mathbf{x}^j_i} + (0,0,\rho^{j^2}_i)\right)(\mathbf{I}_3| - [\mathbf{x}^j_i]_\times)\mathbf{Ad}_{\mathbf{T}_{i\mathtt{W}}}} \tag{B.40}$$

Once again, derivatives with respect to the host and observer only differ on sign, as for the photometric error.

B.5 Derivatives for inertial-only optimization

Necessary derivatives for IMU preintegrated residual can be found at [51]. Here we only provide derivatives for residuals used in the inertial-only optimization, defined in section 3.3, equation 3.18, which also depend on scale s and gravity direction $\mathbf{g}_{dir}$.

In this sense, using the scale update δs as defined at equation 3.20, we obtain the next derivative:

$$\frac{\partial \mathbf{r}_{\Delta \mathrm{R}_{ij}}}{\partial \delta s} = \mathbf{0}_{3\times 1} \tag{B.41}$$

$$\frac{\partial \mathbf{r}_{\Delta \mathrm{v}_{ij}}}{\partial \delta s} = \mathbf{R}^{\mathrm{T}}_{\mathtt{W}i}\left(\bar{\mathbf{v}}_j - \bar{\mathbf{v}}_i\right) s \exp(\delta s) \tag{B.42}$$

$$\frac{\partial \mathbf{r}_{\Delta \mathrm{p}_{ij}}}{\partial \delta s} = \mathbf{R}^{\mathrm{T}}_{\mathtt{W}i}\left(\bar{\mathbf{p}}_j - \bar{\mathbf{p}}_i - \bar{\mathbf{v}}_i \Delta t_{ij}\right) s \exp(\delta s) \tag{B.43}$$

All these expressions are evaluated for $\delta s = 0$. Derivatives for the gravity direction update $\delta \mathbf{g}_{dir}$, as defined in equation 3.19, are obtained as follows:

$$\frac{\partial \mathbf{r}_{\Delta \mathrm{R}_{ij}}}{\partial \delta \mathbf{g}_{dir}} = \mathbf{0}_{3\times 2} \tag{B.44}$$

$$\frac{\partial \mathbf{r}_{\Delta \mathrm{v}_{ij}}}{\partial \delta \mathbf{g}_{dir}} = -\mathbf{R}^{\mathrm{T}}_{\mathtt{W}i}\mathbf{R}_{\mathrm{wg}}\mathbf{G}\Delta t_{ij} \tag{B.45}$$

$$\frac{\partial \mathbf{r}_{\Delta \mathrm{p}_{ij}}}{\partial \delta \mathbf{g}_{dir}} = -\frac{1}{2}\mathbf{R}^{\mathrm{T}}_{\mathtt{W}i}\mathbf{R}_{\mathrm{wg}}\mathbf{G}\Delta t^2_{ij} \tag{B.46}$$

where:

$$\mathbf{G} = \begin{pmatrix} 0 & -G \\ G & 0 \\ 0 & 0 \end{pmatrix} \tag{B.47}$$

Appendix C

Photometric tracking

Here we present in more detail the tracking algorithm used for section 5.2.

Our goal is finding the optimal parameters $\mathbf{p} = [\mathbf{T}_{i,\text{ref}}, a_i, b_i]$ which align reference ref and current i images, in the sense that the photometric error is minimized. We use an inverse compositional approach, as proposed in [2], where we optimize for an update in the reference frame, and inversely update the tracked frame.

In other words, given a reference image $I_{\text{ref}} : \Omega \to \mathbb{R}$ and its estimated sparse depth map $D : \Omega_d \to \mathbb{R}$, and given also the current image I_i, we want to compute parameters $\mathbf{p}$, or equivalently, an increment $\Delta\mathbf{p} = (\xi, \delta a, \delta b)$ on it, which minimizes the next expression:

$$\varepsilon = \frac{1}{2} \sum_{\mathbf{u} \in \Omega_d} \left[e^{-\delta a} \left(I_{\text{ref}}(\mathbf{W}(\mathbf{u}, \xi)) - \delta b \right) - e^{-a_i} \left(I_i(\mathbf{W}(\mathbf{u}, \mathbf{T})) - b_i \right) \right]^2 \tag{C.1}$$

We see how the update is applied in the reference frame instead of current frame i. $\Omega_d \subset \Omega$ stands for the region of the image where the estimated depth map is defined, as explained in section 5.2.2.2. For clarity we use $\mathbf{T} = \mathbf{T}_{i,\text{ref}}$. The warp map $\mathbf{W}$ which transforms image points from reference to i is computed as:

$$\mathbf{u}_i = \mathbf{W}(\mathbf{u}, \mathbf{T}) = \pi(\mathbf{x}_i) = \pi(\mathbf{T} \oplus \mathbf{x}_{\text{ref}}) = \pi(\mathbf{T} \oplus \pi^{-1}(\mathbf{u}, D(\mathbf{u}))) \quad \text{with} \quad \mathbf{W}(\mathbf{u}, 0) = \mathbf{u} \tag{C.2}$$

Taking a linear approximation for ε:

$$\varepsilon = \frac{1}{2} \sum_{\mathbf{u} \in \Omega_d} \left\{ I_{\text{ref}}(\mathbf{u}) - e^{-a_i}(I_i(\mathbf{u}_i) - b_i) + \left[\nabla I_{\text{ref}} \left. \frac{\partial \mathbf{W}}{\partial \xi} \right|_0 , \; -I_{\text{ref}}(\mathbf{u}) \, , \; -1 \right] \begin{bmatrix} \xi \\ \delta a \\ \delta b \end{bmatrix} \right\}^2 \tag{C.3}$$

$$= \frac{1}{2} \sum_{\mathbf{u} \in \Omega_d} [\underbrace{I_{\text{ref}}(\mathbf{u}) - e^{-a_i}(I_i(\mathbf{u}_i) - b_i)}_{r(\mathbf{u},\mathbf{p})} + \mathbf{J}(\mathbf{u})\Delta\mathbf{p}]^2 \tag{C.4}$$

$$= \frac{1}{2} \sum_{\mathbf{u} \in \Omega_d} \left[r^2(\mathbf{u}, \mathbf{p}) + \Delta\mathbf{p}^T \mathbf{J}^T(\mathbf{u}) \mathbf{J}(\mathbf{u}) \Delta\mathbf{p} + 2r(\mathbf{u}, \mathbf{p}) \mathbf{J}(\mathbf{u}) \Delta\mathbf{p} \right] \tag{C.5}$$

We remark that $\mathbf{J}(\mathbf{u})$ depends only on reference frame pixels $\mathbf{u}$ and it may be precomputed and stored as long as the reference frame is not updated. This is the big computational saving of the inverse compositional approach. Now we take derivative of ε with respect to $\Delta\mathbf{p}$, such that:

$$\frac{\partial \varepsilon}{\partial \Delta\mathbf{p}} = \sum_{\mathbf{u} \in \Omega_d} r(\mathbf{u}, \mathbf{p}) \mathbf{J}^T(\mathbf{u}) + \sum_{\mathbf{u} \in \Omega_d} \mathbf{J}^T(\mathbf{u}) \mathbf{J}(\mathbf{u}) \Delta\mathbf{p} \tag{C.6}$$

$$= \mathbf{g} + \mathbf{H}\Delta\mathbf{p} \tag{C.7}$$

and we set it to zero to find the minimum (if Hessian is positive definite), leading to the expected normal equations:

$$\boxed{\Delta \mathbf{p} = -\mathbf{H}^{-1}\mathbf{g}} \tag{C.8}$$

with,

$$
\begin{aligned}
\mathbf{x}_{\text{ref}} =& \Pi^{-1}(\mathbf{u}, D(\mathbf{u})) && \text{(C.9)}\\
\mathbf{u}_i(\mathbf{u}, \mathbf{T}) =& \Pi(\mathbf{T}\mathbf{x}_{\text{ref}}) && \text{(C.10)}\\
r(\mathbf{u}, \mathbf{p}) =& I_{\text{ref}}(\mathbf{u}) - e^{-a_i}(I_i(\mathbf{u}_i) - b_i) && \text{(C.11)}\\
\left.\frac{\partial \mathbf{W}(\mathbf{u}, \xi)}{\partial \xi}\right|_0 =& \left.\frac{\partial \Pi}{\partial \mathbf{x}}\right|_{\mathbf{x}_{\text{ref}}} (\mathbf{I}_3\,,\ -[\mathbf{x}_{\text{ref}}]_\times) && \text{(C.12)}\\
\mathbf{J}(\mathbf{u}) =& \left[\nabla I_{\text{ref}} \left.\frac{\partial \mathbf{W}}{\partial \xi}\right|_0 ,\ -I_{\text{ref}}(\mathbf{u})\,,\ -1\right] && \text{(C.13)}\\
\mathbf{g}(\mathbf{u}, \mathbf{p}) =& \sum_{\mathbf{u}\in\Omega} r(\mathbf{u}, \mathbf{p})\mathbf{J}(\mathbf{u}) && \text{(C.14)}\\
\mathbf{H}(\mathbf{u}) =& \sum_{\mathbf{u}\in\Omega} \mathbf{J}(\mathbf{u})\mathbf{J}^T(\mathbf{u}) && \text{(C.15)}
\end{aligned}
$$

Equations (C.15) and (C.15) are computed once per reference frame.

Since it is an inverse approach, we are updating the reference instead of the tracked frame. For pose, we have previous ref and updated ref' which relate as $\mathbf{T}_{\text{ref',ref}} = \text{Exp}(\xi)$, having new relative transformation as $\mathbf{T}' = \mathbf{T}_{i,\text{ref}}\mathbf{T}_{\text{ref,ref'}}$. This leads to the next pose update:

$$\mathbf{T}' = \mathbf{T} \oplus \text{Exp}(-\xi) \tag{C.16}$$

Similarly, for affine parameters, we have $e^{-\delta a}(I_{\text{ref}} - \delta b) = e^{-a}(I_i - b)$, which leads to $I_{\text{ref}} = e^{-(a-\delta a)}(I_i - (b - e^{(a-\delta a)}\delta b))$ and next updates:

$$a' = a - \delta a \tag{C.17}$$

$$b' = b - e^{a'}\delta b \tag{C.18}$$

Appendix D

Matrix Lie Groups

Matrix Lie groups, such as rotation matrices and rigid body transformations, are continuously used in robotics for state parametrization, being vitally important. These groups do not belong to an Euclidean space and a different theory (differential geometry) needs to be developed.

This appendix does not aim to give a full and self-contained introduction to matrix Lie groups. There exist numerous very well written documents in the literature which present in different ways this field. Among them, we highlight the work by Chirikjian [27], giving a very detailed, exhaustive and abstract introduction to general matrix Lie groups, not only limited to robotics. A more brief and robotics oriented presentation is given by Sola et al. [123]. This is probably the most condensed and complete document with all the necessary concepts used in robotic applications. Finally, a more practical presentation is given by Barfoot [3].

All this previous work lacks some details about specific analytical expression derivations. This appendix aims to fill some of this gap. In this sense, we highlight and focus on the next two points:

- Right Jacobian ($\mathbf{J}_r$) and its inverse ($\mathbf{J}_r^{-1}$) whose derivations are not publicly available to the best of our knowledge. These Jacobians are extensively used for IMU preintegration [52], for example for relateing the gyroscope bias updates with changes in the rotation matrix. In addition, some works, such as [52], give different and not equivalent expressions for $\mathbf{J}_r^{-1}$, increasing the confusion. In sections D.2.6 and D.2.7 we provide a derivation for $\mathbf{J}_r$ and $\mathbf{J}_r^{-1}$ to clarify this inconsistency.

- Rotation matrix normalization. We derive the expression for the optimal projection into the rotation matrix group, which is particularly important when composing transformations with limited precision in a computer. This corresponds with the SVD decomposition, which is widely used in the robotics community, but its derivation can not be found in the literature.

In addition we give a very brief introduction to general matrix Lie groups (section D.1) and its main concepts and particularize them for rotation matrices (section D.2). For a self-contained and painless introduction, we invite the reader to go through [123].

D.1 General Matrix Lie groups

D.1.1 Definition

A Lie group is a smooth manifold whose elements satisfy the group axioms. A smooth manifold is a topological space which locally resembles euclidean. Differential geometry generalizes concepts of euclidean spaces to general manifolds. The group structure $(G, \circ)$ is defined by a set G, a group operation $\circ$ and the next four axioms:

- Closure: $\forall a, b \in G \, : \, a \circ b \in G$
- Associativity: $\forall a, b, c \in G \, : \, (a \circ b) \circ c = a \circ (b \circ c)$
- Identity: $\exists e \in G \, : \, \forall a \in G \, : \, e \circ a = a \circ e = a$
- Inverse: $\forall a \in G \, : \, \exists b \in G \, : \, a \circ b = b \circ a = e$

For the specific case of matrix Lie groups, elements can be stated as matrices, and the group operation is the usual matrix multiplication. Notice that this group is not supposed to be abelian, thus, in general $a \circ b \neq b \circ a$. From now on, for matrix Lie groups, we will omit $\circ$ and we will denote the identity element indistinctly as e or $\mathbf{I}$.

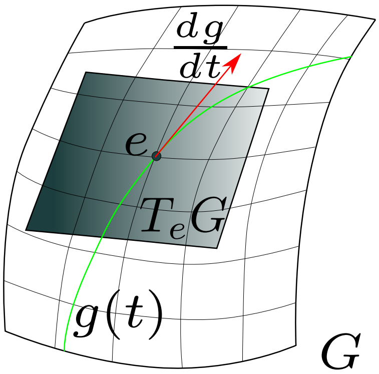


Figure D.1: Representation of a Lie group, with its underlying smooth manifold structure G, its identity element e under group operation $\circ$. Tangent space at the identity T_eG which defines the Lie algebra $\mathfrak{g}$.

D.1.1.1 Tangent space and Lie Algebra

Since matrix Lie groups are smooth manifolds, it is possible to define a tangent space at every point of the manifold. For a curve $g(t) \in G$, its velocity $\dot{g} := dg(t)/dt$ belongs to the tangent space at $g(t)$, which is denoted as $T_{g(t)}G$. From this definition of tangent space, we can define the Lie algebra $\mathfrak{g}$ of G as:

$$\mathfrak{g} := T_eG \tag{D.1}$$

being e the identity element of G. Thus, the Lie algebra is the tangent space of the Lie group (smooth manifold) at the identity, which is in fact a vector space. To compute the

Lie algebra $\mathfrak{g}$ of G, one needs to check the two conditions than an element $\mathbf{v}^\wedge \in \mathfrak{g}$ has to satisfy. The first one, from the definition of the Lie algebra as the tangent space at the identity:

$$\mathbf{v}^\wedge = \frac{d}{du}\left(g(t+u)g^{-1}(t)\right)\Big|_{u=0} = \boxed{\dot{g}g^{-1} \in \mathfrak{g}} \quad \forall t \tag{D.2}$$

And the second one from the identity element group axiom:

$$g(t)g^{-1}(t) = e \Rightarrow \dot{g}g^{-1} + g\dot{g^{-1}} = 0 \Rightarrow \boxed{\dot{g}g^{-1} = -g\dot{g^{-1}} \in \mathfrak{g}} \tag{D.3}$$

Similar procedure can be followed to show that $g^{-1}\dot{g} \in \mathfrak{g}$. These two conditions will be necessary in the future to compute the structure of the Lie Algebra for specific cases such as SO(3) or SE(3).

Since the Lie algebra is a n-dimensional vector space, it is isomorphic to $\mathbb{R}^n$, and one can define a basis $\{\mathbf{b}_i^\wedge\}$, such that any element $\mathbf{v}^\wedge \in \mathfrak{g}$ can be expressed as:

$$\mathbf{v}^\wedge = \sum_{i=1}^{n} v_i \mathbf{b}_i^\wedge$$

We will denote $\mathbf{v} = (v_1, \dots v_n)$ as the reduced coordinate vector of $\mathbf{v}^\wedge$. Elements on the Lie algebra and the reduced coordinates are related by mean of hat and vee operators, defined as:

$$\wedge : \mathbb{R}^n \to \mathfrak{g}$$
$$\vee : \mathfrak{g} \to \mathbb{R}^n$$

D.1.2 Exponential and Logarithmic map

How are elements of G and $\mathfrak{g}$ related? The answer is the equation D.2. This equation defines an ordinary differential equation, whose solution can be immediately found as:

$$\dot{g} = \mathbf{v}^\wedge g \Rightarrow g = g_0 \exp(\mathbf{v}^\wedge t)$$

where $g_0 \in G$ is a constant. Since $g, g_0 \in G$, it follows that $\exp(\mathbf{v}^\wedge t) \in G$, being the exp a map from the Lie algebra to the Lie group. For matrix Lie groups, this map is the exponential matrix, such that:

$$\exp\ :\ \mathfrak{g} \to G\ :\ \mathbf{v}^\wedge \to g = \exp(\mathbf{v}^\wedge) = \sum_{i=0}^{\infty} \frac{(\mathbf{v}^\wedge)^i}{i!} \tag{D.4}$$

For small values of $\mathbf{v}^\wedge$, one can linearly approximate the exponential map as:

$$\exp(\mathbf{v}^\wedge) \approx \mathbf{I} + \mathbf{v}^\wedge \quad \text{for} \quad \mathbf{v}^\wedge \to \mathbf{0}^\wedge \tag{D.5}$$

Where $\mathbf{I}$ is the identity matrix, which is equivalent to the identity element e for matrix Lie groups. For notational simplicity, we define a different exponential map, Exp, which maps from the reduced coordinates of the $\mathfrak{g}$ to G, such that:

$$\text{Exp} : \mathbb{R}^n \to G : \text{Exp}(\mathbf{v}) = \exp(\mathbf{v}^\wedge)$$

There exists an inverse map, denoted as logarithm map $\log\ :\ G \to \mathfrak{g}$, such that $\log(\exp(\mathbf{v}^\wedge)) = \mathbf{v}^\wedge$. For matrix Lie groups, it is defined as:

$$\log(g) = \sum_{i=1}^{\infty} (-1)^{i+1} \frac{(g - \mathbf{I})^i}{i} \tag{D.6}$$

Since exp is not a bijective, one-to-one, function, attention needs to be paid when applying logarithm.

D.1.3 Adjoint operator

An interesting operator for Lie groups is the adjoint operator $\mathrm{Ad}_{\mathbf{v}}$. It is defined for any element $\mathbf{v}^{\wedge} \in \mathfrak{g}$ and operates for elements in G. It is formally defined as follows:

$$\mathrm{Ad}_{\mathbf{v}}(g) := \frac{d}{dt}\left(g \exp(\mathbf{v}^{\wedge} t) g^{-1}\right)\Big|_{t=0} \in \mathfrak{g} \tag{D.7}$$

Since it is the derivative of a curve in G at the identity, it belongs to the Lie algebra. This expression can be extended for matrix Lie groups, taking a linear approximation of exponential map (equation D.4) at zero:

$$\mathrm{Ad}_{\mathbf{v}}(g) \approx \frac{d}{dt}\left(g(I + \mathbf{v}^{\wedge} t) g^{-1}\right)\Big|_{t=0} = g\mathbf{v}^{\wedge} g^{-1} \in \mathfrak{g} \tag{D.8}$$

This operator defines an homomorphism $\mathrm{Ad}_{\mathbf{v}} : G \to GL(\mathfrak{g})$ which takes an element of G and gives an invertible linear transformation from $\mathfrak{g}$ onto itself. It is a homomorphism because:

$$\mathrm{Ad}_{\mathrm{Ad}_{\mathbf{v}}(g_2)}(g_1) = g_1(g_2\mathbf{v}^{\wedge} g_2^{-1})g_1^{-1} = g_1 g_2 \mathbf{v}^{\wedge}(g_1 g_2)^{-1} = \mathrm{Ad}_{\mathbf{v}}(g_1 g_2)$$

And linear because:

$$\mathrm{Ad}_{c_1\mathbf{v}_1 + c_2\mathbf{v}_2}(g) = g(c_1\mathbf{v}_1^{\wedge} + c_2\mathbf{v}_2^{\wedge})g^{-1} = c_1\mathrm{Ad}_{\mathbf{v}_1}(g) + c_2\mathrm{Ad}_{\mathbf{v}_2}(g)$$

Since $\mathrm{Ad}_{\mathbf{v}}$ is a linear operator, we can find its correspondent matrix as follows:

$$\boxed{(\mathrm{Ad}_{\mathbf{v}}(g))^{\vee} = \left(\mathrm{Ad}_{\sum_i v_i \mathbf{b}_i}(g)\right)^{\vee} = \sum_i v_i \left(\mathrm{Ad}_{\mathbf{b}_i}(g)\right)^{\vee} = \mathbf{Ad}(g)\mathbf{v}} \tag{D.9}$$

being matrix $\mathbf{Ad}(g)$ defined as:

$$\boxed{\mathbf{Ad}(g) := [(\mathrm{Ad}_{\mathbf{b}_1}(g))^{\vee} \ldots (\mathrm{Ad}_{\mathbf{b}_n}(g))^{\vee}]} \tag{D.10}$$

Once we have defined and characterized the adjoint operator, the question now is, what is the adjoint useful for? From equation D.9, we can write:

$$\exp\left(\mathrm{Ad}_{\mathbf{v}}(g)\right) = \sum_{i=0} \frac{(g\mathbf{v}^{\wedge} g^{-1})^i}{i!} = g\left(\sum_{i=0} \frac{(\mathbf{v}^{\wedge})^i}{i!}\right) g^{-1} = \exp\left((\mathbf{Ad}(g)\mathbf{v})^{\wedge}\right)$$

which leads to:

$$\boxed{g\mathrm{Exp}(\mathbf{v}) = \mathrm{Exp}\left(\mathbf{Ad}(g)\mathbf{v}\right) g} \tag{D.11}$$

In plain words: given a right transformation $\mathrm{Exp}(\mathbf{v})$, we can find an equivalent left transformation as $\mathrm{Exp}\left(\mathbf{Ad}(g)\mathbf{v}\right)$. So, in the robotics field, this means that adjoint relates transformations or updates in the world and local reference frames. This will be extremely useful when computing for example visual Jacobians (see appendix section B.3).

D.1.4 Jacobians

Another important question is how increments in the tangent space $\mathfrak{g}$ relate with increments on G.

We are going to first relate increments in a curve parameter t with increments on G. In this sense, if $g : \mathbb{R} \to G$ is a curve in G, we have seen $\dot{g}g^{-1}$ and $g^{-1}\dot{g}$ belong to $\mathfrak{g}$. Using standard basis $\{\mathbf{b}_i^\wedge | (\mathbf{b}_i)_j = \delta_i^j\}$ We define the next interesting terms:

$$\omega_l^\wedge := \dot{g}g^{-1} = \sum_{i=1}^{n} \omega_{l,i}\mathbf{b}_i^\wedge \tag{D.12}$$

$$\omega_r^\wedge := g^{-1}\dot{g} = \sum_{i=1}^{n} \omega_{r,i}\mathbf{b}_i^\wedge \tag{D.13}$$

where $\omega_l^\wedge$ and $\omega_r^\wedge$ will also depend on t. These terms allow us to relate increments on the curve parameter t to increments on the Lie group G, as follows:

$$\omega_l^\wedge = \dot{g}g^{-1} = \frac{g(t+\delta t) - g(t)}{\delta t} g^{-1} = \frac{g(t+\delta t)g^{-1} - e}{\delta t}$$

leading to:

$$g(t+\delta t) \approx (e + \omega_l^\wedge \delta t)g(t) \approx \exp(\omega_l^\wedge \delta t)g(t) \quad \text{for} \quad \delta t \to 0$$

where we have used approximation from equation D.5. Similar steps can be followed for $g^{-1}\dot{g}$ leading to the two following expressions:

$$\boxed{g(t+\delta t) \approx \exp(\omega_l^\wedge \delta t)g(t) \quad \text{for} \quad \delta t \to 0} \tag{D.14}$$

$$\boxed{g(t+\delta t) \approx g(t)\exp(\omega_r^\wedge \delta t) \quad \text{for} \quad \delta t \to 0} \tag{D.15}$$

One could be more interested in relating increments on the Lie algebra with increments on the Lie group. We redefine the curve g in a compositional way, such that:

$$g(t) = \mathrm{Exp}(\mathbf{v}(t))$$

where $\mathbf{v} : \mathbb{R} \to \mathbb{R}^n$ is a curve in the reduced coordinates of $\mathfrak{g}$. Using velocity $\dot{\mathbf{v}}$, expression D.14 can be rewritten as:

$$\mathrm{Exp}(\mathbf{v}(t+\delta t)) \approx \mathrm{Exp}(\mathbf{v}(t)+\dot{\mathbf{v}}\delta t) = \mathrm{Exp}(\mathbf{v}(t)+\delta\mathbf{v}) \approx \mathrm{Exp}(\omega_l \delta t)\mathrm{Exp}(\mathbf{v}(t)) \quad \text{for} \quad \delta t, \delta\mathbf{v} \to 0$$

To do that, we rewrite $\omega_l(t)$ as a linear combination of the velocity $\dot{\mathbf{v}}$, such that:

$$\omega_l(t) = \mathbf{J}_l(\mathbf{v}(t))\dot{\mathbf{v}} \tag{D.16}$$

being $\mathbf{J}_l(\mathbf{v}(t))$ a matrix. We obtain,

$$\mathrm{Exp}(\mathbf{v}(t) + \delta\mathbf{v}) \approx \mathrm{Exp}(\mathbf{J}_l(\mathbf{v})\dot{\mathbf{v}}\delta t)\mathrm{Exp}(\mathbf{v}(t)) = \mathrm{Exp}(\mathbf{J}_l(\mathbf{v})\delta\mathbf{v})\mathrm{Exp}(\mathbf{v}(t)) \quad \text{for} \quad \delta\mathbf{v} \to 0$$

Repeating the same for equation D.15, and extending $\omega_r = \mathbf{J}_r\dot{\mathbf{v}}$, we obtain the next two final equations:

$$\boxed{\mathrm{Exp}(\mathbf{v} + \delta\mathbf{v}) \approx \mathrm{Exp}(\mathbf{J}_l(\mathbf{v})\delta\mathbf{v})\mathrm{Exp}(\mathbf{v}) \quad \text{for} \quad \delta\mathbf{v} \to 0} \tag{D.17}$$

$$\boxed{\mathrm{Exp}(\mathbf{v}+\delta\mathbf{v}) \approx \mathrm{Exp}(\mathbf{v})\mathrm{Exp}(\mathbf{J}_r(\mathbf{v})\delta\mathbf{v}) \quad \text{for} \quad \delta\mathbf{v} \to 0} \tag{D.18}$$

These expressions allow us to relate additive changes ($\delta\mathbf{v}$) on the Lie algebra $\mathfrak{g}$ (it is a vector space), with left and right changes on the Lie group, respectively $\mathrm{Exp}(\mathbf{J}_l(\mathbf{v})\delta\mathbf{v})$ and $\mathrm{Exp}(\mathbf{J}_r(\mathbf{v})\delta\mathbf{v})$. We denote matrices $\mathbf{J}_l(\mathbf{v})$ and $\mathbf{J}_r(\mathbf{v})$ as left and right Jacobians at $\mathbf{v}$.

The final question now is, how do we compute these Jacobians? From definition, equation D.12, and equation D.16 we have:

$$(\dot{g}g^{-1})^{\vee} = \mathbf{J}_l\dot{\mathbf{v}} \in \mathbb{R}^n$$

$$(\dot{g}g^{-1})^{\vee}_i = (\mathbf{J}_l\dot{\mathbf{v}})_i = \sum_j (\mathbf{J}_l)_{i,j}\frac{dv_j}{dt}$$

Applying chain rule to the left hand side:

$$(\dot{g}g^{-1})^{\vee}_i = \left(\sum_j \frac{\partial g}{\partial v_j}\frac{dv_j}{dt}g^{-1}\right)^{\vee}_i = \sum_j (\mathbf{J}_l)_{i,j}\frac{dv_j}{dt}$$

$$\boxed{(\mathbf{J}_l)_{i,j} = \left(\frac{\partial g}{\partial v_j}g^{-1}\right)^{\vee}_i} \tag{D.19}$$

In a similar way we can derive an expression for the right Jacobians:

$$\boxed{(\mathbf{J}_r)_{i,j} = \left(g^{-1}\frac{\partial g}{\partial v_j}\right)^{\vee}_i} \tag{D.20}$$

These two equations, D.19 and D.20, will be used for computing Jacobians for different Lie groups. A simple relation can be found between both. First, notice that:

$$\mathbf{J}_l = \left[\left(\frac{\partial g}{\partial v_1}g^{-1}\right)^{\vee} \quad \dots \quad \left(\frac{\partial g}{\partial v_n}g^{-1}\right)^{\vee}\right] \tag{D.21}$$

Each column can be further expanded as:

$$\left(\frac{\partial g}{\partial v_i}g^{-1}\right)^{\vee} = \left(g\left(g^{-1}\frac{\partial g}{\partial v_i}\right)g^{-1}\right)^{\vee} = \left(\mathrm{Ad}_{g^{-1}\frac{\partial g}{\partial v_i}}(g)\right)^{\vee} \tag{D.22}$$

Where we have identified the previously defined adjoint operator (eqn. D.7). We can rewrite:

$$\boxed{\mathbf{J}_l = \mathbf{Ad}(g)\mathbf{J}_r} \tag{D.23}$$

In robotics, these Jacobians will be necessary when working for example with IMU. When the IMU bias is updated, which belongs to the tangent space $\mathfrak{g}$, we are interested in seeing how the rotation matrix changes. The inverse of these Jacobians, $\mathbf{J}_r^{-1}(\mathbf{v})$ and $\mathbf{J}_l^{-1}(\mathbf{v})$, allow us to relate changes but in the other way. From equation D.17 we can write:

$$\mathrm{Exp}(\mathbf{v}+\mathbf{J}_l(\mathbf{v})^{-1}\delta\mathbf{v}) \approx \mathrm{Exp}(\delta\mathbf{v})\mathrm{Exp}(\mathbf{v}) \to \boxed{\mathrm{Log}(\mathrm{Exp}(\delta\mathbf{v})\mathrm{Exp}(\mathbf{v})) \approx \mathbf{v}+\mathbf{J}_l^{-1}(\mathbf{v})\delta\mathbf{v}} \tag{D.24}$$

We can obtain a similar expression for inverse right Jacobian:

$$\boxed{\mathrm{Log}(\mathrm{Exp}(\mathbf{v})\mathrm{Exp}(\delta\mathbf{v})) \approx \mathbf{v}+\mathbf{J}_r^{-1}(\mathbf{v})\delta\mathbf{v}} \tag{D.25}$$

D.2 SO(3): Group of rotations

We particularize previous general properties to the *special orthogonal group* SO(3), which represents 3D rotations.

D.2.1 Definition

The special orthogonal group in dimension n, denoted as SO(n), is a matrix Lie group of matrices such that:

$$\mathrm{SO}(n) = \left\{\mathbf{R} \in \mathrm{GL}(n, \mathbb{R}) \mid \mathbf{R}\mathbf{R}^T = \mathbf{R}^T\mathbf{R} = \mathbf{I}_n\ ,\ \det(\mathbf{R}) = +1\right\} \tag{D.26}$$

This group represents rotational matrices in a n dimensional space. Here, we will deal with SO(3), which is the group of interest in robotics.

D.2.2 $\mathfrak{so}(3)$

The Lie algebra of SO(3), denoted as $\mathfrak{so}(3)$, is the tangent space at the identity which can be directly computed using D.3:

$$\dot{g}g^{-1} = -g\dot{g^{-1}} \Rightarrow \dot{\mathbf{R}}\mathbf{R}^T = -\mathbf{R}\dot{\mathbf{R}}^T = -(\dot{\mathbf{R}}\mathbf{R}^T)^T \in \mathfrak{so}(3) \tag{D.27}$$

This defines the skew symmetric matrices of dimension 3, which is isomorphic to $\mathbb{R}^3$, since there exists a bijective map, denoted as $[\cdot]_\times$:

$$[\cdot]_\times\ :\ \mathbb{R}^3 \to \mathfrak{so}(3)\ :\ [(v_1, v_2, v_3)]_\times = \begin{pmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{pmatrix} \tag{D.28}$$

This operator is equivalent to the general $\wedge$ operator.

D.2.3 Exponential and Logarithmic map

To compute the exponential map, we just apply equation D.4, taking into account equation D.73 from D.3:

$$\begin{aligned} \exp([\mathbf{v}]_\times) = \sum_{i=0}^{\infty} \frac{[\mathbf{v}]_\times^i}{i!} = \mathbf{I}_3 + \sum_{i=0}^{\infty} \frac{[\mathbf{v}]_\times^{2i+1}}{(2i+1)!} + \sum_{i=1}^{\infty} \frac{[\mathbf{v}]_\times^{2i}}{(2i)!} = \\ \mathbf{I}_3 + \left(\sum_{i=0}^{\infty} \frac{(-1)^i\|\mathbf{v}\|^{2i}}{(2i+1)!}\right)[\mathbf{v}]_\times + \left(\sum_{i=1}^{\infty} \frac{(-1)^{i+1}\|\mathbf{v}\|^{2(i-1)}}{(2i)!}\right)[\mathbf{v}]_\times^2 = \end{aligned} \tag{D.29}$$

Identifying terms with Taylor series:

$$\sum_{i=0}^{\infty} \frac{(-1)^i\|\mathbf{v}\|^{2i}}{(2i+1)!} = \frac{1}{\|\mathbf{v}\|}\sum_{i=0}^{\infty} \frac{(-1)^i\|\mathbf{v}\|^{2i+1}}{(2i+1)!} = \frac{\sin\|\mathbf{v}\|}{\|\mathbf{v}\|} \tag{D.30}$$

$$\sum_{i=1}^{\infty} \frac{(-1)^{i+1}\|\mathbf{v}\|^{2(i-1)}}{(2i)!} = \frac{1}{\|\mathbf{v}\|^2}\left(\sum_{i=0}^{\infty} \frac{(-1)^{i+1}\|\mathbf{v}\|^{2i}}{(2i)!} + 1\right) = \frac{1-\cos\|\mathbf{v}\|}{\|\mathbf{v}\|^2} \tag{D.31}$$

Which finally leads to the next closed form (Rodrigues' rotation formula) for the exponential map:

$$\boxed{\exp([\mathbf{v}]_\times) = \mathrm{Exp}(\mathbf{v}) = \mathbf{I} + \frac{\sin\|\mathbf{v}\|}{\|\mathbf{v}\|}[\mathbf{v}]_\times + \frac{1-\cos\|\mathbf{v}\|}{\|\mathbf{v}\|^2}[\mathbf{v}]_\times^2} \tag{D.32}$$

From the exponential map expression we can immediately compute the inverse element $g^{-1}(\mathbf{v})$:

$$g^{-1}(\mathbf{v}) = (g(\mathbf{v}))^T = \mathbf{I}_3 - \frac{\sin\|\mathbf{v}\|}{\|\mathbf{v}\|}[\mathbf{v}]_\times + \frac{1-\cos\|\mathbf{v}\|}{\|\mathbf{v}\|^2}[\mathbf{v}]_\times^2 = g(-\mathbf{v}) \tag{D.33}$$

since $[\mathbf{v}]_\times^2$ is symmetric and $[\mathbf{v}]_\times$ skew. This expression will be useful for later computing the Jacobian matrix. Similar procedures can be followed for the logarithm map, leading to next expression:

$$\boxed{\log(\mathbf{R}) = \theta\mathbf{u} = \frac{\theta(\mathbf{R}-\mathbf{R}^T)}{2\sin\theta} \quad \text{where} \quad \theta = \cos^{-1}\left(\frac{\mathrm{trace}(\mathbf{R})-1}{2}\right) \quad \text{and} \quad \|\mathbf{u}\| = 1} \tag{D.34}$$

D.2.4 Adjoint operator

Using equations D.8 and D.9, and particularizing for SO(3), we can write:

$$[\mathbf{Ad}(\mathbf{R})\mathbf{v}]_\times = \mathbf{R}[\mathbf{v}]_\times\mathbf{R}^T \tag{D.35}$$

Together with expression D.84 from appendix:

$$[\mathbf{Ad}(\mathbf{R})\mathbf{v}]_\times = [\mathbf{R}\mathbf{v}]_\times \tag{D.36}$$

Finally leading to:

$$\boxed{\mathbf{Ad}(\mathbf{R}) = \mathbf{R}} \tag{D.37}$$

D.2.5 Derivative of a SO(3) action

Given $\mathbf{R}_{\mathtt{CW}} \in \mathrm{SO}(3)$, we define $\oplus : \mathrm{SO}(3) \times \mathbb{R}^3 \to \mathbb{R}^3$, an action of SO(3) on $\mathbb{R}^3$, which rotates a 3D vector from reference W to C. In this case, $\oplus$ stands for the usual matrix multiplication. For $\mathbf{x}_{\mathtt{C}} = \mathbf{R}_{\mathtt{CW}}\mathbf{x}_{\mathtt{W}}$, we define the next derivative wrt an increment in the local reference:

$$\frac{\partial \mathbf{x}_{\mathtt{C}}}{\partial \mathbf{R}_{\mathtt{CW}}} \triangleq \left.\frac{\partial(\mathrm{Exp}(\xi)\mathbf{R}_{\mathtt{CW}}\mathbf{x}_{\mathtt{W}})}{\partial \xi}\right|_{\xi=0} \approx \frac{\partial((\mathbf{I}_3 + [\xi]_\times)\mathbf{R}_{cw}\mathbf{x}_{\mathtt{W}})}{\partial \xi} = \frac{\partial([\xi]_\times \mathbf{x}_{\mathtt{C}})}{\partial \xi} \tag{D.38}$$

leading to:

$$\frac{\partial_c \mathbf{x}}{\partial \mathbf{R}_{\mathtt{CW}}} = \frac{\partial \begin{pmatrix} -\xi_3 y_c + \xi_2 z_c \\ \xi_3 x_c - \xi_1 z_c \\ -\xi_2 x_c + \xi_1 y_c \end{pmatrix}}{\partial \xi} = \begin{pmatrix} 0 & z_c & -y_c \\ -z_c & 0 & x_c \\ y_c & -x_c & 0 \end{pmatrix} = -[\mathbf{x}_{\mathtt{C}}]_\times \tag{D.39}$$

Similarly, for the inverse transformation $\mathbf{x}_{\mathtt{W}} = \mathbf{R}_{\mathtt{WC}}\mathbf{x}_{\mathtt{C}}$, and using the same update, we define the next derivative:

$$\frac{\partial \mathbf{x}_{\mathtt{W}}}{\partial \mathbf{R}_{\mathtt{CW}}} \triangleq \left.\frac{\partial\left((\mathrm{Exp}(\xi)\mathbf{R}_{\mathtt{CW}})^T \mathbf{x}_{\mathtt{W}}\right)}{\partial \xi}\right|_{\xi=0} \approx \frac{\partial(\mathbf{R}_{\mathtt{WC}}(\mathbf{I}_3 - [\xi]_\times)\mathbf{x}_{\mathtt{W}})}{\partial \xi} = \mathbf{R}_{\mathtt{WC}}[\mathbf{x}_{\mathtt{W}}]_\times \tag{D.40}$$

D.2.6 Jacobian derivation

We are not aware of any publicly available derivation for SO(3) Jacobians. Here we provide one for $\mathbf{J}_r$, using the expression D.20. With this in mind, we first compute $\frac{\partial g}{\partial v_j}$. By denoting $(\mathbf{b}_1^\wedge, \mathbf{b}_2^\wedge, \mathbf{b}_3^\wedge)$ the basis of the Lie algebra $\mathfrak{so}(3)$, and $(\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3)$ that one of the reduced space $\mathbb{R}^3$, we have:

$$\begin{aligned}\frac{\partial g}{\partial v_j} = \frac{\partial}{\partial v_j}\left(\mathbf{I}_3 + \frac{\sin\|\mathbf{v}\|}{\|\mathbf{v}\|}[\mathbf{v}]_\times + \frac{1-\cos\|\mathbf{v}\|}{\|\mathbf{v}\|^2}[\mathbf{v}]_\times^2\right) = \\ \left(\frac{\cos\|\mathbf{v}\|}{\|\mathbf{v}\|} - \frac{\sin\|\mathbf{v}\|}{\|\mathbf{v}\|^2}\right)\frac{\partial\|\mathbf{v}\|}{\partial v_j}[\mathbf{v}]_\times + \frac{\sin\|\mathbf{v}\|}{\|\mathbf{v}\|}\mathbf{b}_j^\wedge + \dots \\ \left(\frac{\sin\|\mathbf{v}\|}{\|\mathbf{v}\|^2} - 2\frac{1-\cos\|\mathbf{v}\|}{\|\mathbf{v}\|^3}\right)\frac{\partial\|\mathbf{v}\|}{\partial v_j}[\mathbf{v}]_\times^2 + \frac{1-\cos\|\mathbf{v}\|}{\|\mathbf{v}\|^2}\frac{\partial[\mathbf{v}]_\times^2}{\partial v_j}\end{aligned} \tag{D.41}$$

where:

$$\frac{\partial\|\mathbf{v}\|}{\partial v_j} = \frac{\partial}{\partial v_j}\left((v_1^2+v_2^2+v_3^2)^{1/2}\right) = \frac{v_j}{\|\mathbf{v}\|} \tag{D.42}$$

Using equation D.33 together with D.41, we now compute $g^{-1}\frac{\partial g}{\partial v_j} \in \mathfrak{g}$ as follows:

$$\begin{aligned}g^{-1}\frac{\partial g}{\partial v_j} = \left(\frac{\cos\|\mathbf{v}\|}{\|\mathbf{v}\|} - \frac{\sin\|\mathbf{v}\|}{\|\mathbf{v}\|^2}\right)\frac{v_j}{\|\mathbf{v}\|}[\mathbf{v}]_\times + \frac{\sin\|\mathbf{v}\|}{\|\mathbf{v}\|}\mathbf{b}_j^\wedge + \dots \\ \left(\frac{\sin\|\mathbf{v}\|}{\|\mathbf{v}\|^2} - 2\frac{1-\cos\|\mathbf{v}\|}{\|\mathbf{v}\|^3}\right)\frac{v_j}{\|\mathbf{v}\|}[\mathbf{v}]_\times^2 + \frac{1-\cos\|\mathbf{v}\|}{\|\mathbf{v}\|^2}\frac{\partial[\mathbf{v}]_\times^2}{\partial v_j} + \dots \\ -\left(\frac{\cos\|\mathbf{v}\|}{\|\mathbf{v}\|} - \frac{\sin\|\mathbf{v}\|}{\|\mathbf{v}\|^2}\right)\frac{v_j\sin\|\mathbf{v}\|}{\|\mathbf{v}\|^2}[\mathbf{v}]_\times^2 - \frac{\sin^2(\|\mathbf{v}\|)}{\|\mathbf{v}\|^2}[\mathbf{v}]_\times\mathbf{b}_j^\wedge + \dots \\ +\left(\frac{\sin\|\mathbf{v}\|}{\|\mathbf{v}\|^2} - 2\frac{1-\cos\|\mathbf{v}\|}{\|\mathbf{v}\|^3}\right)v_j\sin\|\mathbf{v}\|\,[\mathbf{v}]_\times - \frac{1-\cos\|\mathbf{v}\|}{\|\mathbf{v}\|^2}\frac{\sin\|\mathbf{v}\|}{\|\mathbf{v}\|}[\mathbf{v}]_\times\frac{\partial[\mathbf{v}]_\times^2}{\partial v_j} + \dots \\ -\left(\frac{\cos\|\mathbf{v}\|}{\|\mathbf{v}\|} - \frac{\sin\|\mathbf{v}\|}{\|\mathbf{v}\|^2}\right)\frac{v_j(1-\cos\|\mathbf{v}\|)}{\|\mathbf{v}\|}[\mathbf{v}]_\times + \frac{1-\cos\|\mathbf{v}\|}{\|\mathbf{v}\|^2}\frac{\sin\|\mathbf{v}\|}{\|\mathbf{v}\|}[\mathbf{v}]_\times^2\mathbf{b}_j^\wedge + \dots \\ -\left(\frac{\sin\|\mathbf{v}\|}{\|\mathbf{v}\|^2} - 2\frac{1-\cos\|\mathbf{v}\|}{\|\mathbf{v}\|^3}\right)\frac{v_j(1-\cos\|\mathbf{v}\|)}{\|\mathbf{v}\|}[\mathbf{v}]_\times^2 + \left(\frac{1-\cos\|\mathbf{v}\|}{\|\mathbf{v}\|^2}\right)^2[\mathbf{v}]_\times^2\frac{\partial[\mathbf{v}]_\times^2}{\partial v_j}\end{aligned} \tag{D.43}$$

If we reorder this expression and do some simplifications, we arrive to the next expression, splitted in $[\mathbf{v}]_\times$, $[\mathbf{v}]_\times^2$ and others terms:

$$\begin{aligned}g^{-1}\frac{\partial g}{\partial v_j} = v_j a(\mathbf{v})[\mathbf{v}]_\times + v_j b(\mathbf{v})[\mathbf{v}]_\times^2 + c(\mathbf{v})\mathbf{b}_j^\wedge + d(\mathbf{v})\frac{\partial[\mathbf{v}]_\times^2}{\partial v_j} + \dots \\ - c^2(\mathbf{v})[\mathbf{v}]_\times\mathbf{b}_j^\wedge - c(\mathbf{v})d(\mathbf{v})[\mathbf{v}]_\times\frac{\partial[\mathbf{v}]_\times^2}{\partial v_j} + c(\mathbf{v})d(\mathbf{v})[\mathbf{v}]_\times^2\mathbf{b}_j^\wedge + d^2(\mathbf{v})[\mathbf{v}]_\times^2\frac{\partial[\mathbf{v}]_\times^2}{\partial v_j}\end{aligned} \tag{D.44}$$

where:

$$\begin{aligned} a(\mathbf{v}) &= \frac{1}{\|\mathbf{v}\|^2}\left(1+\frac{\cos\|\mathbf{v}\|\sin\|\mathbf{v}\|}{\|\mathbf{v}\|}-\frac{2\sin\|\mathbf{v}\|}{\|\mathbf{v}\|}\right)\\ b(\mathbf{v}) &= \frac{1-\cos\|\mathbf{v}\|}{\|\mathbf{v}\|^4}\\ c(\mathbf{v}) &= \frac{\sin\|\mathbf{v}\|}{\|\mathbf{v}\|}\\ d(\mathbf{v}) &= \frac{1-\cos\|\mathbf{v}\|}{\|\mathbf{v}\|^2} \end{aligned} \tag{D.45}$$

Since $g^{-1}\frac{\partial g}{\partial v_j} \in \mathfrak{so}(3)$, and being $\mathfrak{so}(3)$ the space of skew matrices, we are just interested in skew components of matrix terms from equation D.44, as symmetric component will be null. Any square matrix $\mathbf{M}$ can be decomposed as the sum of a symmetric and skew matrices, since $\mathbf{M} = \frac{1}{2}(\mathbf{M}+\mathbf{M}^T) + \frac{1}{2}(\mathbf{M}-\mathbf{M}^T)$ being $\mathbf{M}+\mathbf{M}^T$ symmetric and $\mathbf{M}-\mathbf{M}^T$ skew. Let's denote Skew($\mathbf{M}$) the skew component of $\mathbf{M}$, we have:

$$\begin{aligned} \text{Skew}([\mathbf{v}]_\times) &= [\mathbf{v}]_\times\\ \text{Skew}([\mathbf{v}]_\times^2) &= \mathbf{0}_3\\ \text{Skew}(\mathbf{b}_j^\wedge) &= \mathbf{b}_j^\wedge\\ \text{Skew}([\mathbf{v}]_\times\mathbf{b}_j^\wedge) &= \frac{1}{2}(\mathbf{b}_j\mathbf{v}^T - \mathbf{v}\mathbf{b}_j^T)\\ \text{Skew}([\mathbf{v}]_\times^2\mathbf{b}_j^\wedge) &= -\frac{1}{2}(v_j[\mathbf{v}]_\times + \|\mathbf{v}\|^2\mathbf{b}_j^\wedge)\\ \text{Skew}\left(\frac{\partial[\mathbf{v}]_\times^2}{\partial v_j}\right) &= \mathbf{0}_3\\ \text{Skew}\left([\mathbf{v}]_\times\frac{\partial[\mathbf{v}]_\times^2}{\partial v_j}\right) &= -\frac{1}{2}(3v_j[\mathbf{v}]_\times + \mathbf{b}_j^\wedge\|\mathbf{v}\|^2)\\ \text{Skew}\left([\mathbf{v}]_\times^2\frac{\partial[\mathbf{v}]_\times^2}{\partial v_j}\right) &= -\frac{\|\mathbf{v}^2\|}{2}(\mathbf{b}_j\mathbf{v}^T - \mathbf{v}\mathbf{b}_j^T) \end{aligned} \tag{D.46}$$

where we have used properties from appendix section D.3. This expression leads to:

$$\begin{aligned} g^{-1}\frac{\partial g}{\partial v_j} =& \left(v_j a(\mathbf{v}) + v_j c(\mathbf{v})d(\mathbf{v})\right)[\mathbf{v}]_\times + \dots\\ & c(\mathbf{v})\mathbf{b}_j^\wedge + \dots\\ & -\frac{1}{2}\left(c^2(\mathbf{v}) + \|\mathbf{v}\|^2 d^2(\mathbf{v})\right)(\mathbf{b}_j\mathbf{v}^T - \mathbf{v}\mathbf{b}_j^T) \end{aligned} \tag{D.47}$$

Now, it only remains to apply the vee $\vee$ operator to $g^{-1}\frac{\partial g}{\partial v_j}$ (see equation D.20). Noticing that $(\mathbf{b}_j\mathbf{v}^T - \mathbf{v}\mathbf{b}_j^T)_i^\vee = ([\mathbf{v}]_\times)_{i,j}$, we obtain using index notation:

$$\mathbf{J}_r(\mathbf{v})_{i,j} = \left(g^{-1}\frac{\partial g}{\partial v_j}\right)_i^\vee = \left(a(\mathbf{v}) + c(\mathbf{v})d(\mathbf{v})\right)v_j v_i + c(\mathbf{v})\delta_{i,j} - \frac{1}{2}\left(c^2(\mathbf{v}) + \|\mathbf{v}\|^2 d^2(\mathbf{v})\right)([\mathbf{v}]_\times)_{i,j} \tag{D.48}$$

which in matrix notation is similar to:

$$\mathbf{J}_r(\mathbf{v}) = \left(a(\mathbf{v}) + c(\mathbf{v})d(\mathbf{v})\right)\mathbf{v}\mathbf{v}^T + c(\mathbf{v})\mathbf{I}_3 - \frac{1}{2}\left(c^2(\mathbf{v}) + \|\mathbf{v}\|^2 d^2(\mathbf{v})\right)[\mathbf{v}]_\times \tag{D.49}$$

We can now simplify factors as follows:

$$\begin{aligned} a(\mathbf{v}) + c(\mathbf{v})d(\mathbf{v}) &= \frac{\|\mathbf{v}\| - \sin\|\mathbf{v}\|}{\|\mathbf{v}\|^3} \\ c(\mathbf{v}) &= \frac{\sin\|\mathbf{v}\|}{\|\mathbf{v}\|} = \|\mathbf{v}\|^2 \frac{\sin\|\mathbf{v}\| - \|\mathbf{v}\|}{\|\mathbf{v}\|^3} + 1 \\ -\frac{1}{2}\left(c^2(\mathbf{v}) + \|\mathbf{v}\|^2 d^2(\mathbf{v})\right) &= -\frac{1 - \cos\|\mathbf{v}\|}{\|\mathbf{v}\|^2} \end{aligned} \tag{D.50}$$

Finally, remembering that $[\mathbf{v}]_\times^2 = \mathbf{v}\mathbf{v}^T - \|\mathbf{v}\|^2 \mathbf{I}_3$, one can rewrite:

$$\boxed{\mathbf{J}_r(\mathbf{v}) = \mathbf{I}_3 - \frac{1 - \cos\|\mathbf{v}\|}{\|\mathbf{v}\|^2}[\mathbf{v}]_\times + \frac{\|\mathbf{v}\| - \sin\|\mathbf{v}\|}{\|\mathbf{v}\|^3}[\mathbf{v}]_\times^2} \tag{D.51}$$

We remark that it is possible to derive this expression making use of the Baker-Campbell-Hausdorff formula [27], but we prefer not to introduce this formula to keep as simple as possible this appendix.

D.2.7 Inverse Jacobian derivation

In the literature one can find different expressions for $\mathbf{J}_r^{-1}(\mathbf{v})$ which are not equivalent, for example [27] and [52]. Aiming to give some light about this issue, we provide here a demonstration showing that [27] is the right one.

First, lets notice that at SO(3), both exponential matrix $\mathrm{Exp}(\mathbf{v})$ and right Jacobian $\mathbf{J}_r(\mathbf{v})$, are linear combination of $\mathbf{I}$, $[\mathbf{v}]_\times$ and $[\mathbf{v}]_\times^2$. For deriving the inverse Jacobian we will suppose this is also true for $\mathbf{J}_r^{-1}(\mathbf{v})$, such that:

$$\mathbf{J}_r^{-1}(\mathbf{v}) = \mathbf{I} + \alpha[\mathbf{v}]_\times + \beta[\mathbf{v}]_\times^2 \tag{D.52}$$

To find α and β, we will solve next equation:

$$\begin{aligned} \mathbf{I} = \mathbf{J}_r(\mathbf{v})\mathbf{J}_r^{-1}(\mathbf{v}) = &\mathbf{I} + \left(\alpha - \frac{1-c}{\|\mathbf{v}\|^2} + \beta(1-c) - \alpha\frac{\|\mathbf{v}\| - s}{\|\mathbf{v}\|}\right)[\mathbf{v}]_\times + \dots \\ &+ \left(\beta - \alpha\frac{1-c}{\|\mathbf{v}\|^2} + \frac{\|\mathbf{v}\| - s}{\|\mathbf{v}\|^3} - \beta\frac{\|\mathbf{v}\| - s}{\|\mathbf{v}\|}\right)[\mathbf{v}]_\times^2 \end{aligned}$$

where we have used properties from D.72 and D.72 and c and s stand for $\cos\|\mathbf{v}\|$ and $\sin\|\mathbf{v}\|$. Setting both $[\mathbf{v}]_\times$ and $[\mathbf{v}]_\times^2$ terms to zero, we get next 2 equations:

$$-\frac{1-c}{\|\mathbf{v}\|^2} + \beta(1-c) + \alpha\frac{s}{\|\mathbf{v}\|} = 0 \tag{D.53}$$

$$-\alpha\frac{1-c}{\|\mathbf{v}\|^2} + \frac{\|\mathbf{v}\| - s}{\|\mathbf{v}\|^3} + \beta\frac{s}{\|\mathbf{v}\|} = 0 \tag{D.54}$$

From D.53 we get:

$$\alpha = \frac{\|\mathbf{v}\|(1-c)}{s}(\|\mathbf{v}\|^{-2} - \beta) \tag{D.55}$$

and replacing in D.54 gives:

$$\beta\left(\frac{s}{\|\mathbf{v}\|} + \frac{(1-c)^2}{s\|\mathbf{v}\|}\right) = \left(\frac{(1-c)^2}{\|\mathbf{v}\|^3 s} - \frac{\|\mathbf{v}\| - s}{\|\mathbf{v}\|^3}\right) \tag{D.56}$$

finally leading to:

$$\beta = \frac{(1-c)^2 - s\left\|\mathbf{v}\right\| + s^2}{s^2\left\|\mathbf{v}\right\|^2 + (1-c)\left\|\mathbf{v}\right\|^2} = \frac{2(1-c) - s\left\|\mathbf{v}\right\|}{2(1-c)\left\|\mathbf{v}\right\|^2} \tag{D.57}$$

$$\beta = \frac{1}{\left\|\mathbf{v}\right\|^2} - \frac{\sin\left\|\mathbf{v}\right\|}{2\left\|\mathbf{v}\right\|(1-\cos\left\|\mathbf{v}\right\|)} = \frac{1}{\left\|\mathbf{v}\right\|^2} - \frac{1+\cos\left\|\mathbf{v}\right\|}{2\left\|\mathbf{v}\right\|\sin\left\|\mathbf{v}\right\|} \tag{D.58}$$

Replacing back into D.55 leads to $\alpha = 1/2$. We finally get the inverse right Jacobian as:

$$\boxed{\mathbf{J}_r^{-1}(\mathbf{v}) = \mathbf{I} + \frac{1}{2}[\mathbf{v}]_\times + \left(\frac{1}{\left\|\mathbf{v}\right\|^2} - \frac{1+\cos\left\|\mathbf{v}\right\|}{2\left\|\mathbf{v}\right\|\sin\left\|\mathbf{v}\right\|}\right)[\mathbf{v}]_\times^2} \tag{D.59}$$

D.2.8 SO(3) normalization

When composing rotations in a computer, the loss of precision may lead to elements which do not belong to SO(3). If using quaternion parametrization, this leads to a norm different to 1, while using matrix parametrization may lead to non-orthogonal matrices. For these reasons it is necessary to normalize the rotation element to make it belong again to the group of rotations.

For matrix representation, this is equivalent to find a projection $\mathbf{P}_R \in \mathrm{SO}(3)$ of a matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$ into the space SO(3). There are several intuitive ways of projecting a matrix into SO(3): convert to a quaternion parameterization and set the norm to 1, apply an Gram–Schmidt orthogonalization, compose exponential and logarithm map, such that $\mathbf{P}_R = \mathrm{Exp}(\log(\mathbf{R}))$. Here, we are interested in finding the optimal projection which minimizes the squared Frobenius norm of the difference, such that:

$$\mathbf{P}_R = \arg\min_{\mathbf{P}_R} \left\|\mathbf{P}_R - \mathbf{R}\right\|_F^2 \quad \text{s.t.} \quad \mathbf{P}_R \in \mathrm{SO}(3) \tag{D.60}$$

This problem can be stated as a constrained optimization problem using Lagrange multipliers. Given that $\mathbf{P}_R$ has to meet the condition $\mathbf{P}_R\mathbf{P}_R^T = \mathbf{I}$, and denoting the constraint matrix $\mathbf{C} = \mathbf{P}_R\mathbf{P}_R^T - \mathbf{I}$, we have:

$$\mathbf{P}_R = \arg\min_{\mathbf{P}_R,\mathbf{\Lambda}} \left\{\mathrm{tr}((\mathbf{P}_R - \mathbf{R})^T(\mathbf{P}_R - \mathbf{R})) + \sum_i\sum_j \lambda_{ij}C_{ij}\right\} \tag{D.61}$$

Since $\mathbf{C}$ is a symmetric matrix by definition, constraints C_{ij} are equivalent to constraints C_{ji}, which forces also symmetry in $\mathbf{\Lambda}$. One can rewrite second half part as:

$$\sum_i\sum_j \lambda_{ij}C_{ij} = \mathrm{tr}(\mathbf{\Lambda}\mathbf{C}) \tag{D.62}$$

finally leading to:

$$\mathbf{P}_R = \arg\min_{\mathbf{P}_R,\mathbf{\Lambda}} \left\{\mathrm{tr}((\mathbf{P}_R - \mathbf{R})^T(\mathbf{P}_R - \mathbf{R}) + \mathbf{\Lambda}(\mathbf{P}_R\mathbf{P}_R^T - \mathbf{I}))\right\} \tag{D.63}$$

Taking the derivatives of the cost function $c(\mathbf{P}_R, \mathbf{\Lambda})$ w.r.t. $\mathbf{P}_R$, and given that (see [105]):

$$\frac{\partial \mathrm{tr}(\mathbf{A}^T(\mathbf{X})\mathbf{A}(\mathbf{X}))}{\partial \mathbf{X}} = 2\mathbf{A}(\mathbf{X}) \tag{D.64}$$

$$\frac{\partial \mathrm{tr}(\mathbf{A}\mathbf{X}^T\mathbf{X})}{\partial \mathbf{X}} = \mathbf{X}(\mathbf{A} + \mathbf{A}^T) \tag{D.65}$$

This leads to the following derivatives of the cost function:

$$\frac{\partial c(\mathbf{P}_R, \mathbf{\Lambda})}{\partial \mathbf{P}_R} = -2(\mathbf{R} - \mathbf{P}_R) + \mathbf{P}_R(\mathbf{\Lambda} + \mathbf{\Lambda}^T) \tag{D.66}$$

Setting this expression to zero and taking the symmetry of $\mathbf{\Lambda}$, we get:

$$\mathbf{R} = \mathbf{P}_R(\mathbf{I} + \mathbf{\Lambda}) \rightarrow \mathbf{P}_R = \mathbf{R}(\mathbf{I} + \mathbf{\Lambda})^{-1} \tag{D.67}$$

Hence, $\mathbf{R}^T\mathbf{R} = (\mathbf{I} + \mathbf{\Lambda})^2$, which finally leads to:

$$\mathbf{R} = \mathbf{P}_R(\mathbf{I} + \mathbf{\Lambda}) \rightarrow \mathbf{P}_R = \mathbf{R}(\mathbf{R}^T\mathbf{R})^{-1/2} \tag{D.68}$$

Given the eigen decomposition $\mathbf{R} = \mathbf{U}\mathbf{S}\mathbf{V}^T$,

$$\mathbf{R}^T\mathbf{R} = \mathbf{V}\mathbf{S}\mathbf{S}\mathbf{V}^T \rightarrow (\mathbf{R}^T\mathbf{R})^{1/2} = \mathbf{V}\mathbf{S}\mathbf{V}^T \,\text{since}\, \mathbf{V}\mathbf{S}\mathbf{V}^T\mathbf{V}\mathbf{S}\mathbf{V}^T = \mathbf{V}\mathbf{S}\mathbf{S}\mathbf{V}^T \tag{D.69}$$

$$(\mathbf{R}^T\mathbf{R})^{-1/2} = \mathbf{V}\mathbf{S}^{-1}\mathbf{V}^T \rightarrow \mathbf{R}(\mathbf{R}^T\mathbf{R})^{-1/2} = \mathbf{U}\mathbf{S}\mathbf{V}^T\mathbf{V}\mathbf{S}^{-1}\mathbf{V}^T = \mathbf{U}\mathbf{V}^T \tag{D.70}$$

$$\boxed{\mathbf{P}_R = \mathbf{U}\mathbf{V}^T \quad \text{where} \quad \mathbf{R} = \mathbf{U}\mathbf{S}\mathbf{V}^T} \tag{D.71}$$

where D.71 is the optimal projection given the Frobenius norm.

D.3 Useful maths

Here we provide and derive some useful expressions for previous sections. We start with $\mathbf{v}]^2_\times$, such that:

$$[\mathbf{v}]^2_\times = \begin{pmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{pmatrix}^2 = \begin{pmatrix} -v_2^2 - v_3^2 & v_1v_2 & v_1v_3 \\ v_1v_2 & -v_1^2 - v_3^2 & v_2v_3 \\ v_1v_3 & v_2v_3 & -v_1^2 - v_2^2 \end{pmatrix} = \mathbf{v}\mathbf{v}^T - \|\mathbf{v}\|^2 \mathbf{I}_3 \tag{D.72}$$

Notice that $[\mathbf{v}]^2_\times$ is symmetric. Thus,

$$[\mathbf{v}]^3_\times = \begin{pmatrix} 0 & v_3(v_1^2 + v_2^2 + v_3^2) & -v_2(v_1^2 + v_2^2 + v_3^2) \\ -v_3(v_1^2 + v_2^2 + v_3^2) & 0 & v_1(v_1^2 + v_2^2 + v_3^2) \\ v_2(v_1^2 + v_2^2 + v_3^2) & -v_1(v_1^2 + v_2^2 + v_3^2) & 0 \end{pmatrix} = -\|\mathbf{v}\|^2 [\mathbf{v}]_\times \tag{D.73}$$

We also compute the derivative $\frac{\partial [\mathbf{v}]^2_\times}{\partial v_j}$ which will be usefull in the next step:

$$\frac{\partial [\mathbf{v}]^2_\times}{\partial v_j} = \mathbf{v}\mathbf{b}_j^T + \mathbf{b}_j\mathbf{v}^T - 2v_j\mathbf{I}_3 \tag{D.74}$$

as well as other multiplication terms, such that:

$$[\mathbf{v}]_\times \mathbf{b}_j^\wedge = \mathbf{b}_j\mathbf{v}^T - v_j\mathbf{I}_3 \tag{D.75}$$

$$[\mathbf{v}]^2_\times \mathbf{b}_j^\wedge = [\mathbf{v}]_\times \mathbf{b}_j\mathbf{v}^T - v_j[\mathbf{v}]_\times \tag{D.76}$$

and:

$$[\mathbf{v}]_\times \frac{\partial[\mathbf{v}]_\times^2}{\partial v_j} = [\mathbf{v}]_\times \mathbf{v}\mathbf{b}_j^T + [\mathbf{v}]_\times \mathbf{b}_j \mathbf{v}^T - 2v_j[\mathbf{v}]_\times \tag{D.77}$$

$$[\mathbf{v}]_\times^2 \frac{\partial[\mathbf{v}]_\times^2}{\partial v_j} = [\mathbf{v}]_\times^2 \mathbf{v}\mathbf{b}_j^T + [\mathbf{v}]_\times^2 \mathbf{b}_j \mathbf{v}^T - 2v_j[\mathbf{v}]_\times^2 \tag{D.78}$$

Now, we may also show that $[\mathbf{R}\mathbf{v}]_\times = \mathbf{R}[\mathbf{v}]_\times\mathbf{R}^T$ with $\mathbf{R} \in \mathrm{SO}(3)$. First, let's compute $(\mathbf{R}[\mathbf{v}]_\times\mathbf{R}^T)^2$:

$$\begin{aligned}
\left(\mathbf{R}[\mathbf{v}]_\times\mathbf{R}^T\right)^2 =& \mathbf{R}[\mathbf{v}]_\times^2\mathbf{R}^T && \text{(D.79)}\\
=& \mathbf{R}\left(\mathbf{v}\mathbf{v}^T - \|\mathbf{v}\|^2\,\mathbf{I}_3\right)\mathbf{R}^T && \text{(D.80)}\\
=& (\mathbf{R}\mathbf{v})(\mathbf{R}\mathbf{v})^T - \|\mathbf{v}\|^2\,\mathbf{I}_3 && \text{(D.81)}\\
=& (\mathbf{R}\mathbf{v})(\mathbf{R}\mathbf{v})^T - \|\mathbf{R}\mathbf{v}\|^2\,\mathbf{I}_3 && \text{(D.82)}\\
=& \left([\mathbf{R}\mathbf{v}]_\times\right)^2 && \text{(D.83)}
\end{aligned}$$

where we have used equation D.72 and the fact that $\|\mathbf{R}\mathbf{v}\| = \|\mathbf{v}\|$. We finally obtain:

$$[\mathbf{R}\mathbf{v}]_\times = \mathbf{R}[\mathbf{v}]_\times\mathbf{R}^T \tag{D.84}$$

Appendix E

Supplementary material for ORB-SLAM3

E.1 Reference systems and extrinsic calibration

Extrinsic calibration consists of the set of parameters which define how different sensors are related. When working on a multi-sensor SLAM system, different reference frames have to be defined for each sensor. This multiple definition does not imply having multiple optimizable frames. Instead, we define a unique optimizable reference frame, denoted as Body ($\mathtt{B}$). All other sensors relate to this by means of a SE(3) rigid transformation which, at ORB-SLAM3, is supposed to be known from calibration. For stereo-inertial or monocular inertial ORB-SLAM3 configurations, we make $\mathtt{B}$ to be coincident with IMU, while right and left cameras are defined w.r.t it. Thus, once IMU is initialized, we have the following reference frames (see figure E.1):

- **World** ($\mathtt{W}$): Defines a fixed reference system, whose $z_{\mathtt{W}}$ axis points in opposite direction of the gravity vector $\mathbf{g}$. Translation and yaw are freely set by the SLAM system and remain fixed once initialized.

- **Body** ($\mathtt{B}$): This is the optimizable reference and it is supposed to be coincident with the IMU. We assume the gyroscope and the accelerometer share the same reference system. Body pose $\mathbf{T}_{\mathtt{WB}}$ and velocity $\mathbf{v}_{\mathtt{B}}$ in $\mathtt{W}$ are the optimizable variables.

- **Cameras** ($\mathtt{C}_1$ and $\mathtt{C}_2$): These are coincident with visual sensors, such as $z_{\mathtt{C}}$ is pointing forward, along the optical axis. $y_{\mathtt{C}}$ points down and $x_{\mathtt{C}}$ right, both coincident with image directions u and v. Cameras and body poses relate as:

$$\mathbf{T}_{\mathtt{WC}_1} = \mathbf{T}_{\mathtt{WB}}\mathbf{T}_{\mathtt{BC}_1} \tag{E.1}$$

$$\mathbf{T}_{\mathtt{WC}_2} = \mathbf{T}_{\mathtt{WB}}\mathbf{T}_{\mathtt{BC}_1}\mathbf{T}_{\mathtt{C}_1\mathtt{C}_2} \tag{E.2}$$

  $\mathbf{T}_{\mathtt{BC}_1}, \mathbf{T}_{\mathtt{C}_1\mathtt{C}_2} \in \mathrm{SE}(3)$ are the extrinsic parameters, known from calibration and need to be included in the .yaml calibration file, as shown in listings E.1 and E.2.

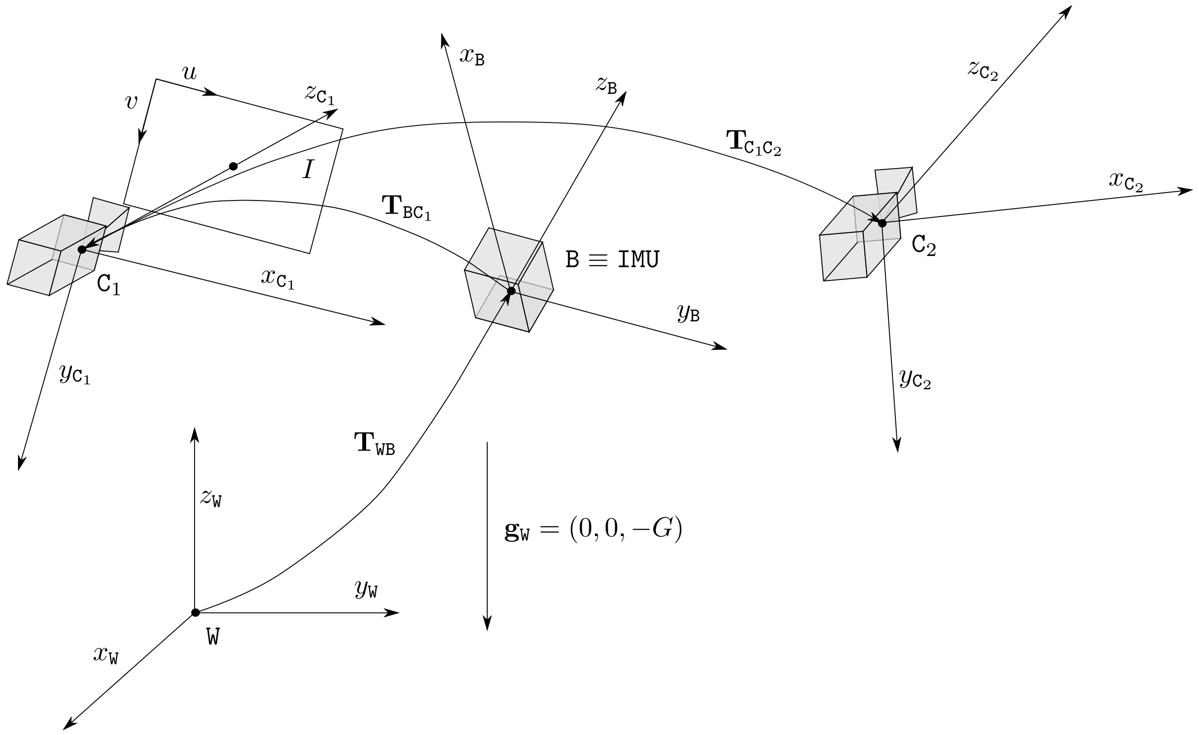


Figure E.1: Defined coordinate systems for stereo-inertial ORB-SLAM3.

```
Tbc: !!opencv-matrix
rows: 4
cols: 4
dt: f
data: [-0.9995250,  0.0075019, -0.0298901,  0.0455748,
         0.0296153, -0.0343973, -0.9989693, -0.0711618,
        -0.0085223, -0.9993800,  0.0341588, -0.0446812,
         0.0,        0.0,        0.0,        1.0]
```

Listing E.1: Transformation matrix from left camera (1) to body. Values are those from TUM-VI dataset

```
Tlr: !!opencv-matrix
rows: 3
cols: 4
dt: f
data: [ 0.9999994,   0.0007916,   0.0006940,   0.1010634,
        -0.0008233,   0.9988994,   0.0468954,   0.0019462,
        -0.0006561,  -0.0468960,   0.9988995,   0.0010153]
```

Listing E.2: Transformation matrix from right camera (2) to left camera (1). Values are those from TUM-VI dataset

These extrinsic transformations can be obtained from calibration software such as *Kalibr* [111]. For stereo pinhole cameras (i.e. EuRoC dataset), the right camera reference frame $\mathtt{C}_2$ is not used. Instead, right images are stereo rectified, such as $v_1 \equiv v_2$, and u_2 observation is directly transformed into the $\mathtt{C}_1$ reference frame. For stereo fisheye cameras, rectification is not necessary.

For pure visual solutions, $\mathtt{B}$ is coincident with the left camera $\mathtt{C}_1$. Right camera, only for non rectified stereo, is related by a SE(3) transformation. World reference is set to the first keyframe reference and remains fixed as long as no loop closing or map merging are performed.

E.2 Intrinsic calibration

Those are calibration parameters which only depend on the sensor itself. Here, we distinguish inertial and visual sensors.

E.2.1 IMU intrinsic parameters

IMU readings ($\widetilde{\mathbf{a}}$ and $\widetilde{\omega}$) are affected by measurement noise (η^a, η^g) and bias ($\mathbf{b}^a, \mathbf{b}^g$), such as:

$$\widetilde{\mathbf{a}} = \mathbf{a} + \eta^a + \mathbf{b}^a \tag{E.3}$$

$$\widetilde{\omega} = \omega + \eta^g + \mathbf{b}^g \tag{E.4}$$

where $\mathbf{a}$ and ω are the true acceleration (gravity not subtracted) and angular velocity at B. Measurement noises are assumed to follow centered normal distributions, such that:

$$\eta^a \sim \mathcal{N}(0, \sigma_a^2 \mathbf{I}_3) \tag{E.5}$$

$$\eta^g \sim \mathcal{N}(0, \sigma_g^2 \mathbf{I}_3) \tag{E.6}$$

where σ_a and σ_g are both noise densities, which are characterized in the IMU data-sheet. They need to be provided at the calibration file, with $m/s^2/\sqrt(\text{Hz})$ and $\text{rad}/s/\sqrt(\text{Hz})$ units, as shown in listing E.3.

```
IMU.NoiseGyro: 0.00016 # rad/s^-0.5
IMU.NoiseAcc: 0.0028 # m/s^-1.5
IMU.Frequency: 200 #s^-1
```

Listing E.3: Noise densities for IMU. Values are from TUM-VI dataset

When integrating the IMU measurements and estimating its covariance, used noise densities $\sigma_{a,f}$ will depend on IMU sampling frequency f, which will be also provided along the calibration file. This is internally managed by ORB-SLAM3, which computes $\sigma_{a,f} = \sigma_a / \sqrt{f}$

Regarding bias, it is supposed to evolve according to a Brownian motion. Given two consecutive instants i and $i+1$, this is characterized by:

$$\mathbf{b}^a_{i+1} = \mathbf{b}^a_i + \eta^a_{\text{rw}} \quad \text{with } \eta^a_{\text{rw}} \sim \mathcal{N}(0, \sigma^2_{a,\text{rw}} \mathbf{I}_3) \tag{E.7}$$

$$\mathbf{b}^g_{i+1} = \mathbf{b}^g_i + \eta^g_{\text{rw}} \quad \text{with } \eta^g_{\text{rw}} \sim \mathcal{N}(0, \sigma^2_{g,\text{rw}} \mathbf{I}_3) \tag{E.8}$$

where $\sigma_{a,\text{rw}}$ and $\sigma_{g,\text{rw}}$ need to be supplied with the calibration file, as shown in listing E.4.

```
IMU.GyroWalk: 0.000022 # rad/s^-1.5
IMU.AccWalk: 0.00086 # m/s^-2.5
```

Listing E.4: Random walk variances for IMU biases. Values are from TUM-VI dataset.

E.2.2 Camera intrinsic parameters

Depending on camera set-up we will need to provide different calibration parameters. At ORB-SLAM3 we distinguish next visual cases:

- Monocular pinhole camera. Camera focal length (f_x, f_y) and central point (c_x, c_y) in pixel lengths, as introduced in the appendix section B.1.1, together with 4 or 5 distortion coefficients for a radial-tangential distortion model [131] need to be provided along the calibration file as shown in listing E.5. Those can be calibrated using *opencv* [1] or *Kalibr*.

```
Camera.fx: 458.654
Camera.fy: 457.296
Camera.cx: 367.215
Camera.cy: 248.375

Camera.k1: -0.28340811
Camera.k2: 0.07395907
Camera.p1: 0.00019359
Camera.p2: 1.76187114e-05
```

Listing E.5: Left pinhole camera intrinsic parameters for Euroc dataset.

- Monocular fisheye camera. Camera parameters (f_x, f_y, c_x, c_y) similar to the pinhole camera case and 4 distortion coefficients, see the appendix section B.1.2, for an equidistant distortion model (Kanala-Brandt) need to be provided. An example is shown in code E.6.

```
Camera.k1: 0.0034823894022493434
Camera.k2: 0.0007150348452162257
Camera.k3: -0.0020532361418706202
Camera.k4: 0.00020293673591811182
```

Listing E.6: Left fisheye camera intrinsic parameters for TUM-VI dataset.

- Stereo pinhole camera. For this case we will rectify left and right images. If left and right images are not already rectified, as in the EuRoC dataset, we will need to provide rectification parameters. From extrinsic left-right transformation and distortion parameters, these can be easily computed, using for example opencv library[2]. If images are already stereo rectified, as in the KITTI dataset, only common camera parameters (f and c) and stereo baseline need to be provided.

- Stereo fisheye camera. Left and right camera matrices and equidistant distortion parameters.

[1] `https://docs.opencv.org/3.4.15/d9/d0c/group__calib3d.html`

[2] `https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html#stereorectify`

Bibliography

[1] Tim Bailey, Juan Nieto, and Eduardo Nebot. Consistency of the fastSLAM algorithm. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 424–429. IEEE, 2006.

[2] Simon Baker and Iain Matthews. Lucas-Kanade 20 years on: A unifying framework. *Int. J. Computer Vision*, 56(3):221–255, 2004.

[3] Timothy D Barfoot. *State estimation for robotics.* Cambridge University Press, 2017.

[4] Ioan Andrei Bârsan, Peidong Liu, Marc Pollefeys, and Andreas Geiger. Robust dense mapping for large-scale dynamic environments. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 7510–7517. IEEE, 2018.

[5] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: the clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008:1–10, 2008.

[6] Berta Bescos, José M Fácil, Javier Civera, and José Neira. DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes. *Robotics and Automation Letters*, 3 (4):4076–4083, 2018.

[7] Berta Bescos, José Neira, Roland Siegwart, and Cesar Cadena. Empty cities: Image inpainting for a dynamic-object-invariant space. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 5460–5466. IEEE, 2019.

[8] Berta Bescos, Cesar Cadena, and José Neira. Empty cities: A dynamic-object-invariant space for visual slam. *IEEE Trans. Robotics*, 37(2):433–451, 2020.

[9] Berta Bescos, Carlos Campos, Juan D Tardós, and José Neira. DynaSLAM II: Tightly-coupled multi-object tracking and SLAM. *Robotics and Automation Letters*, 6(3):5191–5198, 2021.

[10] Borna Bešić and Abhinav Valada. Dynamic Object Removal and Spatio-Temporal RGB-D Inpainting via Geometry-Aware Adversarial Learning. *arXiv preprint arXiv:2008.05058*, 2020.

[11] Michael Bloesch, Sammy Omari, Marco Hutter, and Roland Siegwart. ROVIO. `https://github.com/ethz-asl/rovio`, 2015.

[12] Michael Bloesch, Sammy Omari, Marco Hutter, and Roland Siegwart. Robust visual inertial odometry using a direct EKF-based approach. In *Proc. IEEE/RJS Int. Conf. Intelligent Robots and Systems (IROS)*, pages 298–304, 2015.

[13] Michael Bloesch, Michael Burri, Sammy Omari, Marco Hutter, and Roland Siegwart. Iterated extended Kalman filter based visual-inertial odometry using direct photometric feedback. *Int. J. Robotics Research*, 36(10):1053–1072, 2017.

[14] Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, and Andrew J Davison. CodeSLAM—learning a compact, optimisable representation for dense visual SLAM. In *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, pages 2560–2568, 2018.

[15] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact: Real-time instance segmentation. In *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, pages 9157–9166. IEEE, 2019.

[16] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The EuRoC micro aerial vehicle datasets. *Int. J. Robotics Research*, 35(10):1157–1163, 2016.

[17] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Trans. Robotics*, 32(6):1309–1332, 2016.

[18] Carlos Campos and Juan D. Tardós. Scale-aware direct monocular odometry. *arXiv preprint arXiv:2109.10077*, 2021.

[19] Carlos Campos, Jose M M. Montiel, and Juan D Tardós. Fast and robust initialization for visual-inertial SLAM. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 1288–1294. IEEE, 2019.

[20] Carlos Campos, Richard Elvira, Juan J. Gómez Rodríguez, Jose M. M. Montiel, and Juan D. Tardós. ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM. `https://github.com/UZ-SLAMLab/ORB_SLAM3`, 2020.

[21] Carlos Campos, José MM Montiel, and Juan D Tardós. Inertial-only optimization for visual-inertial initialization. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 51–57. IEEE, 2020.

[22] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM. *IEEE Trans. Robotics*, pages 1–17, 2021.

[23] Jose A Castellanos, José MM Montiel, José Neira, and Juan D Tardós. The SPmap: A probabilistic framework for simultaneous localization and map building. *IEEE Trans. Robotics and Automation*, 15(5):948–952, 1999.

[24] José A Castellanos, José Neira, and Juan D Tardós. Limits to the consistency of EKF-based SLAM. *IFAC Proceedings Volumes*, 37(8):716–721, 2004.

[25] Kenneth Chaney. Monocular MSCKF. `https://github.com/daniilidis-group/msckf_mono`, 2018.

[26] Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, Huimin Ma, Sanja Fidler, and Raquel Urtasun. 3D object proposals using stereo imagery for accurate object class detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 40(5):1259–1272, 2017.

[27] Gregory S Chirikjian. *Stochastic models, information theory, and Lie groups, volume 2: Analytic methods and modern applications*, volume 2. Springer Science & Business Media, 2011.

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

[29] Javier Civera, Andrew J Davison, and JM Martinez Montiel. Inverse depth parametrization for monocular SLAM. *IEEE Trans. Robotics*, 24(5):932–945, 2008.

[30] Javier Civera, Óscar G. Grasa, Andrew J. Davison, and José M. M. Montiel. 1-point RANSAC for extended Kalman filtering: Application to real-time structure from motion and visual odometry. *Journal of Field Robotics*, 27(5):609–631, 2010.

[31] Laura Clemente, Andrew J. Davison, Ian D. Reid, José Neira, and J. D. Tardós. Mapping large loops with a single hand-held camera. In *Proc. Robotics: Science and Systems*, Atlanta, GA, USA, June 2007.

[32] Jan Czarnowski, Tristan Laidlow, Ronald Clark, and Andrew J Davison. Deepfactors: Real-time probabilistic dense monocular SLAM. *Robotics and Automation Letters*, 5(2):721–728, 2020.

[33] Andrew J. Davison. SceneLib 1.0. `https://www.doc.ic.ac.uk/~ajd/Scene/index.html`.

[34] Andrew J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, pages 1403–1410, vol. 2, Oct 2003.

[35] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007.

[36] Frank Dellaert and Michael Kaess. Square root sam: Simultaneous localization and mapping via square root information smoothing. *Int. J. Robotics Research*, 25(12): 1181–1203, 2006.

[37] Jeffrey Delmerico and Davide Scaramuzza. A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 2502–2509, 2018.

[38] MWM Gamini Dissanayake, Paul Newman, Steve Clark, Hugh F Durrant-Whyte, and Michael Csorba. A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Trans. Robotics and Automation*, 17(3):229–241, 2001.

[39] Tue-Cuong Dong-Si and Anastasios I Mourikis. Estimator initialization in vision-aided inertial navigation with unknown camera-imu calibration. In *Proc. IEEE/RJS Int. Conf. Intelligent Robots and Systems (IROS)*, pages 1064–1071. IEEE, 2012.

[40] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *Advances in Neural Information Processing Systems*, 27:2366–2374, 2014.

[41] Richard Elvira, Juan D. Tardós, and Jose M. M. Montiel. ORBSLAM-atlas: a robust and accurate multi-map system. In *Proc. IEEE/RJS Int. Conf. Intelligent Robots and Systems (IROS)*, 2019.

[42] J. Engel, J. Stueckler, and D. Cremers. Large-scale direct SLAM with stereo cameras. In *Proc. IEEE/RJS Int. Conf. Intelligent Robots and Systems (IROS)*, 2015.

[43] J. Engel, V. Koltun, and D. Cremers. DSO: Direct Sparse Odometry. `https://github.com/JakobEngel/dso`, 2018.

[44] Jakob Engel, Thomas Schöps, and Daniel Cremers. LSD-SLAM: Large-scale direct monocular SLAM. `https://github.com/tum-vision/lsd_slam`.

[45] Jakob Engel, Thomas Schöps, and Daniel Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *European Conf. on Computer Vision (ECCV)*, pages 834–849, 2014.

[46] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 40(3):611–625, 2018.

[47] Jose M Facil, Benjamin Ummenhofer, Huizhong Zhou, Luis Montesano, Thomas Brox, and Javier Civera. Cam-convs: Camera-aware multi-scale convolutions for single-view depth. In *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, pages 11826–11835, 2019.

[48] Matthias Faessler, Flavio Fontana, Christian Forster, and Davide Scaramuzza. Automatic re-initialization and failure recovery for aggressive flight with a monocular vision-based quadrotor. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 1722–1729. IEEE, 2015.

[49] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. SVO. `https://github.com/uzh-rpg/rpg_svo`, 2014.

[50] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 15–22, 2014.

[51] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. In *Proc. Robotics: Science and Systems*, 2015.

[52] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. On-manifold preintegration for real-time visual–inertial odometry. *IEEE Trans. Robotics*, 33(1):1–21, 2016.

[53] Christian Forster, Zichao Zhang, Michael Gassner, Manuel Werlberger, and Davide Scaramuzza. SVO: Semidirect visual odometry for monocular and multicamera systems. *IEEE Trans. Robotics*, 33(2):249–265, 2017.

[54] Dorian Gálvez-López and Juan D. Tardós. Bags of binary words for fast place recognition in image sequences. *IEEE Trans. Robotics*, 28(5):1188–1197, October 2012. ISSN 1552-3098.

[55] Dorian Gálvez-López, Marta Salas, Juan D Tardós, and José MM Montiel. Real-time monocular object SLAM. *Robotics and Autonomous Systems*, 75:435–449, 2016.

[56] Xiang Gao, Rui Wang, Nikolaus Demmel, and Daniel Cremers. LDSO: Direct sparse odometry with loop closure. In *Proc. IEEE/RJS Int. Conf. Intelligent Robots and Systems (IROS)*, pages 2198–2204, 2018.

[57] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *Int. J. Robotics Research*, 32(11):1231–1237, 2013.

[58] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, pages 270–279, 2017.

[59] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, pages 3828–3838, 2019.

[60] Ruben Gomez-Ojeda, Francisco-Angel Moreno, David Zuñiga-Noël, Davide Scaramuzza, and Javier Gonzalez-Jimenez. PL-SLAM: a stereo SLAM system through the combination of points and line segments. *IEEE Trans. Robotics*, 35(3):734–746, 2019.

[61] Mina Henein, Gerard Kennedy, Robert Mahony, and Viorela Ila. Exploiting rigid body motion for SLAM in dynamic environments. *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2018.

[62] Mehdi Hosseinzadeh, Kejie Li, Yasir Latif, and Ian Reid. Real-time monocular object-model aware sparse SLAM. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2019.

[63] Jiahui Huang, Sheng Yang, Zishuo Zhao, Yu-Kun Lai, and Shi-Min Hu. Cluster-SLAM: A SLAM Backend for Simultaneous Rigid Body Clustering and Motion Estimation. In *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, pages 5875–5884, 2019.

[64] Jiahui Huang, Sheng Yang, Tai-Jiang Mu, and Shi-Min Hu. ClusterVO: Clustering Moving Instances and Estimating Visual Odometry for Self and Surroundings. In *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, pages 2168–2177, 2020.

[65] Eagle S Jones and Stefano Soatto. Visual-inertial navigation, mapping and localization: A scalable real-time causal approach. *Int. J. Robotics Research*, 30(4): 407–430, 2011.

[66] Kevin M Judd, Jonathan D Gammell, and Paul Newman. Multimotion visual odometry (MVO): Simultaneous estimation of camera and third-party motions. *Proc. IEEE/RJS Int. Conf. Intelligent Robots and Systems (IROS)*, 2018.

[67] Michael Kaess. Simultaneous localization and mapping with infinite planes. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 4605–4611. IEEE, 2015.

[68] Michael Kaess, Ananth Ranganathan, and Frank Dellaert. isam: Incremental smoothing and mapping. *IEEE Trans. Robotics*, 24(6):1365–1378, 2008.

[69] Jacques Kaiser, Agostino Martinelli, Flavio Fontana, and Davide Scaramuzza. Simultaneous state initialization and gyroscope bias calibration in visual inertial aided navigation. *Robotics and Automation Letters*, 2(1):18–25, 2017.

[70] Juho Kannala and Sami S Brandt. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 28(8):1335–1340, 2006.

[71] Jonathan Kelly and Gaurav S Sukhatme. Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration. *Int. J. Robotics Research*, 30(1):56–79, 2011.

[72] Hanme Kim. SceneLib2 - MonoSLAM open-source library. `https://github.com/hanmekim/SceneLib2`.

[73] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. IEEE/ACM Int. Symp. Mixed and Augmented Reality*, pages 225–234, Nara, Japan, 2007.

[74] G. Klein and D. Murray. Parallel tracking and mapping on a camera phone. In *2009 8th IEEE International Symposium on Mixed and Augmented Reality*, pages 83–86, Oct 2009.

[75] Georg Klein and David Murray. Improving the agility of keyframe-based SLAM. In *European Conf. on Computer Vision (ECCV)*, pages 802–815, 2008.

[76] Georg Klein and David Murray. PTAM-GPL. `https://github.com/Oxford-PTAM/PTAM-GPL`, 2013.

[77] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g2o: A general framework for graph optimization. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 3607–3613, 2011.

[78] Seong Hun Lee and Javier Civera. Loosely-coupled semi-direct monocular SLAM. *Robotics and Automation Letters*, 4(2):399–406, 2018.

[79] Stefan Leutenegger, Paul Furgale, Vincent Rabaud, Margarita Chli, Kurt Konolige, and Roland Siegwart. Keyframe-based visual-inertial SLAM using nonlinear optimization. *Proceedings of Robotics Science and Systems (RSS)*, 2013.

[80] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *Int. J. Robotics Research*, 34(3):314–334, 2015.

[81] Stefan Leutenegger, Andreas Forster, Paul Furgale, Pascal Gohl, and Simon Lynen. OKVIS: Open keyframe-based visual-inertial SLAM (ROS version). `https://github.com/ethz-asl/okvis_ros`, 2016.

[82] Mingyang Li and Anastasios I Mourikis. High-precision, consistent EKF-based visual-inertial odometry. *Int. J. Robotics Research*, 32(6):690–711, 2013.

[83] Peiliang Li, Tong Qin, et al. Stereo vision-based semantic 3D object and ego-motion tracking for autonomous driving. In *European Conf. on Computer Vision (ECCV)*, 2018.

[84] Shile Li and Dongheui Lee. RGB-D SLAM in dynamic environments using static point weighting. *Robotics and Automation Letters*, 2017.

[85] Stephanie Lowry, Niko Sünderhauf, Paul Newman, John J Leonard, David Cox, Peter Corke, and Michael J Milford. Visual place recognition: A survey. *IEEE Trans. Robotics*, 32(1):1–19, 2015.

[86] Lucas, Bruce D and Kanade, Takeo and others. An iterative image registration technique with an application to stereo vision. In *Proc. Int. Joint Conf. Artificial Intelligence*, pages 674–679, 1981.

[87] Todd Lupton and Salah Sukkarieh. Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions. *IEEE Trans. Robotics*, 28(1):61–76, 2012.

[88] Kirk MacTavish and Timothy D Barfoot. At all costs: A comparison of robust cost functions for camera correspondence outliers. In *Conference on Computer and Robot Vision*, pages 62–69. IEEE, 2015.

[89] Agostino Martinelli. Vision and imu data fusion: Closed-form solutions for attitude, speed, absolute scale, and bias determination. *IEEE Trans. Robotics*, 28(1):44–60, 2011.

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

[91] Hidenobu Matsuki, Lukas von Stumberg, Vladyslav Usenko, Jörg Stückler, and Daniel Cremers. Omnidirectional DSO: Direct sparse odometry with fisheye cameras. *Robotics and Automation Letters*, 3(4):3693–3700, 2018.

[92] Hidenobu Matsuki, Raluca Scona, Jan Czarnowski, and Andrew J Davison. Codemapping: Real-time dense mapping for sparse SLAM using compact scene representations. *Robotics and Automation Letters*, 2021.

[93] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conf. on Computer Vision (ECCV)*, pages 405–421. Springer, 2020.

[94] Anastasios I Mourikis and Stergios I Roumeliotis. A multi-state constraint Kalman filter for vision-aided inertial navigation. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 3565–3572, 2007.

[95] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 1957.

[96] Raúl Mur-Artal and Juan D. Tardós. Fast relocalisation and loop closing in keyframe-based SLAM. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 846–853. IEEE, 2014.

[97] Raúl Mur-Artal and Juan D. Tardós. ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras. *IEEE Trans. Robotics*, 33(5):1255–1262, 2017.

[98] Raúl Mur-Artal and Juan D Tardós. Visual-inertial monocular SLAM with map reuse. *Robotics and Automation Letters*, 2(2):796–803, 2017.

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

[100] Raúl Mur-Artal, Juan D. Tardós, Jose M. M. Montiel, and Dorian Gálvez-López. ORB-SLAM2. `https://github.com/raulmur/ORB_SLAM2`, 2016.

[101] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, pages 165–174, 2019.

[102] Mrinal K Paul and Stergios I Roumeliotis. Alternating-stereo VINS: Observability analysis and performance evaluation. In *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, pages 4729–4737, 2018.

[103] Mrinal K Paul, Kejian Wu, Joel A Hesch, Esha D Nerurkar, and Stergios I Roumeliotis. A comparative analysis of tightly-coupled monocular, binocular, and stereo VINS. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 165–172, 2017.

[104] Lina M Paz, Juan D Tardós, and José Neira. Divide and conquer: EKF SLAM in $o(n)$. *IEEE Trans. Robotics*, 24(5):1107–1120, 2008.

[105] Kaare Brandt Petersen, Michael Syskind Pedersen, et al. The matrix cookbook. *Technical University of Denmark*, 7(15):510, 2008.

[106] Albert Pumarola, Alexander Vakhitov, Antonio Agudo, Alberto Sanfeliu, and Francese Moreno-Noguer. PL-SLAM: Real-time monocular visual SLAM with points and lines. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 4503–4508. IEEE, 2017.

[107] Tong Qin and Shaojie Shen. Robust initialization of monocular visual-inertial estimation on aerial robots. In *Proc. IEEE/RJS Int. Conf. Intelligent Robots and Systems (IROS)*, pages 4225–4232, 2017.

[108] Tong Qin, Peiliang Li, and Shaojie Shen. VINS-Mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Trans. Robotics*, 34(4):1004–1020, 2018.

[109] Tong Qin, Shaozu Cao, Jie Pan, Peiliang Li, and Shaojie Shen. VINS-Fusion: An optimization-based multi-sensor state estimator. `https://github.com/HKUST-Aerial-Robotics/VINS-Fusion`, 2019.

[110] Tong Qin, Jie Pan, Shaozu Cao, and Shaojie Shen. A general optimization-based framework for local odometry estimation with multiple sensors. *arXiv preprint arXiv:1901.03638*, 2019.

[111] Joern Rehder, Janosch Nikolic, Thomas Schneider, Timo Hinzmann, and Roland Siegwart. Extending kalibr: Calibrating the extrinsics of multiple IMUs and of individual axes. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 4304–4311, 2016.

[112] John G Rogers, Alexander JB Trevor, Carlos Nieto-Granda, and Henrik I Christensen. SLAM with expectation maximization for moveable object tracking. *Proc. IEEE/RJS Int. Conf. Intelligent Robots and Systems (IROS)*, 2010.

[113] Antoni Rosinol, Marcus Abate, Yun Chang, and Luca Carlone. Kimera. `https://github.com/MIT-SPARK/Kimera`, 2019.

[114] Antoni Rosinol, Marcus Abate, Yun Chang, and Luca Carlone. Kimera: an open-source library for real-time metric-semantic localization and mapping. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2020.

[115] Antoni Rosinol, Arjun Gupta, Marcus Abate, Jingnan Shi, and Luca Carlone. 3D Dynamic Scene Graphs: Actionable Spatial Perception with Places, Objects, and Humans. *arXiv preprint arXiv:2002.06289*, 2020.

[116] Stergios I Roumeliotis and Anastasios I Mourikis. Vision-aided inertial navigation, September 19 2017. US Patent 9,766,074.

[117] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, pages 2564–2571, 2011.

[118] Martin Rünz and Lourdes Agapito. Co-fusion: Real-time segmentation, tracking and fusion of multiple objects. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 4471–4478, 2017.

[119] Martin Runz, Maud Buffier, and Lourdes Agapito. Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects. In *Proc. IEEE/ACM Int. Symp. Mixed and Augmented Reality*, 2018.

[120] Renato F Salas-Moreno, Richard A Newcombe, Hauke Strasdat, Paul HJ Kelly, and Andrew J Davison. SLAM++: Simultaneous localisation and mapping at the level of objects. In *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, pages 1352–1359, 2013.

[121] David Schubert, Thore Goll, Nikolaus Demmel, Vladyslav Usenko, Jörg Stückler, and Daniel Cremers. The TUM VI benchmark for evaluating visual-inertial odometry. In *Proc. IEEE/RJS Int. Conf. Intelligent Robots and Systems (IROS)*, pages 1680–1687, 2018.

[122] David Schubert, Thore Goll, Nikolaus Demmel, Vladyslav Usenko, Jörg Stückler, and Daniel Cremers. The TUM VI benchmark for evaluating visual-inertial odometry. *arXiv preprint arXiv:1804.06120v3*, March 2020.

[123] Joan Sola, Jeremie Deray, and Dinesh Atchuthan. A micro lie theory for state estimation in robotics. *arXiv preprint arXiv:1812.01537*, 2018.

[124] H. Strasdat, A. J. Davison, Jose M. M. Montiel, and K. Konolige. Double window optimisation for constant time visual SLAM. In *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, pages 2352–2359, 2011.

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

[126] Hauke Strasdat, J Montiel, and Andrew J Davison. Scale drift-aware large scale monocular SLAM. *Proc. Robotics: Science and Systems*, 2, 2010.

[127] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *Proc. IEEE/RJS Int. Conf. Intelligent Robots and Systems (IROS)*, Oct. 2012.

[128] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *Proc. IEEE/RJS Int. Conf. Intelligent Robots and Systems (IROS)*, pages 573–580, 2012.

[129] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J Davison. imap: Implicit mapping and positioning in real-time. *arXiv preprint arXiv:2103.12352*, 2021.

[130] Yuxiang Sun, Ming Liu, and Max Q-H Meng. Improving RGB-D SLAM in dynamic environments: A motion removal approach. *RAS*, 2017.

[131] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Verlag, London, 2011.

[132] Keisuke Tateno, Federico Tombari, Iro Laina, and Nassir Navab. CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction. In *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, pages 6243–6252, 2017.

[133] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment—a modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer, 1999.

[134] Roger Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal on Robotics and Automation*, 3(4):323–344, 1987.

[135] Vladyslav Usenko and Nikolaus Demmel. BASALT. `https://gitlab.com/VladyslavUsenko/basalt`, 2019.

[136] Vladyslav Usenko, Nikolaus Demmel, David Schubert, Jörg Stückler, and Daniel Cremers. Visual-inertial mapping with non-linear factor recovery. *Robotics and Automation Letters*, 5(2), April 2020.

[137] Lukas von Stumberg, Vladyslav Usenko, and Daniel Cremers. Direct sparse visual-inertial odometry using dynamic marginalization. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2018.

[138] Chieh-Chih Wang, Charles Thorpe, and Sebastian Thrun. Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2003.

[139] Chieh-Chih Wang, Charles Thorpe, Sebastian Thrun, Martial Hebert, and Hugh Durrant-Whyte. Simultaneous localization, mapping and moving object tracking. *Int. J. Robotics Research*, 26(9), 2007.

[140] Rui Wang, Martin Schworer, and Daniel Cremers. Stereo DSO: Large-scale direct sparse visual odometry with stereo cameras. In *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, pages 3903–3911, 2017.

[141] Somkiat Wangsiripitak and David W Murray. Avoiding moving outliers in visual SLAM by tracking moving objects. In *ICRA*. IEEE, 2009.

[142] Linhui Xiao, Jinge Wang, Xiaosong Qiu, Zheng Rong, and Xudong Zou. Dynamic-SLAM: Semantic monocular visual localization and mapping based on deep learning in dynamic environment. *RAS*, 117, 2019.

[143] Binbin Xu, Wenbin Li, Dimos Tzoumanikas, Michael Bloesch, Andrew Davison, and Stefan Leutenegger. MID-fusion: Octree-based object-level multi-instance dynamic SLAM. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 5231–5237, 2019.

[144] Heng Yang, Pasquale Antonante, Vasileios Tzoumas, and Luca Carlone. Graduated non-convexity for robust spatial perception: From non-minimal solvers to global outlier rejection. *Robotics and Automation Letters*, 5(2):1127–1134, 2020.

[145] Nan Yang, Rui Wang, Jorg Stuckler, and Daniel Cremers. Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. In *European Conf. on Computer Vision (ECCV)*, pages 817–833, 2018.

[146] Nan Yang, Lukas von Stumberg, Rui Wang, and Daniel Cremers. D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry. In *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, pages 1281–1292, 2020.

[147] Shichao Yang and Sebastian Scherer. CubeSLAM: Monocular 3-D object SLAM. *IEEE Trans. Robotics*, 35(4):925–938, 2019.

[148] Shichao Yang, Yu Song, Michael Kaess, and Sebastian Scherer. Pop-up SLAM: Semantic monocular plane SLAM for low-texture environments. In *Proc. IEEE/RJS Int. Conf. Intelligent Robots and Systems (IROS)*, pages 1222–1229. IEEE, 2016.

[149] Zhenfei Yang and Shaojie Shen. Monocular visual–inertial state estimation with online initialization and camera–imu extrinsic calibration. *IEEE Transactions on Automation Science and Engineering*, 14(1):39–51, 2016.

[150] Huangying Zhan, Chamara Saroj Weerasekera, Jia-Wang Bian, and Ian Reid. Visual odometry revisited: What should be learnt? In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 4203–4210. IEEE, 2020.

[151] Jun Zhang, Mina Henein, Robert Mahony, and Viorela Ila. VDO-SLAM: A Visual Dynamic Object-aware SLAM System. *arXiv preprint arXiv:2005.11052*, 2020.

[152] Zichao Zhang and Davide Scaramuzza. A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry. In *Proc. IEEE/RJS Int. Conf. Intelligent Robots and Systems (IROS)*, pages 7244–7251, 2018.

[153] Jon Zubizarreta, Iker Aguinaga, Juan D. Tardós, and Jose M. M. Montiel. DSM: Direct Sparse Mapping. `https://github.com/jzubizarreta/dsm`, 2019.

[154] Jon Zubizarreta, Iker Aguinaga, and Jose Maria Martinez Montiel. Direct sparse mapping. *IEEE Trans. Robotics*, 36(4):1363–1370, 2020.