UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

EDISON KLEIBER TTITO CONCHA

Map Point Optimization in Keyframe-Based SLAM using Covisibility Graph and Information Fusion

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Computer Science

Advisor: Prof. Dr. Edson Prestes e Silva Júnior

Porto Alegre
May 2018

CIP — CATALOGING-IN-PUBLICATION

Concha, Edison Kleiber Ttito

Map Point Optimization in Keyframe-Based SLAM using Covisibility Graph and Information Fusion / Edison Kleiber Ttito Concha. – Porto Alegre: PPGC da UFRGS, 2018.

67 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2018. Advisor: Edson Prestes e Silva Júnior.

1. Information Fusion. 2. Robot Vision. 3. Visual SLAM. 4. Keyframe-Based. I. Prestes e Silva Júnior, Edson. II. Título.

“Life can only be understood backwards; but it must be lived forwards.”

— Søren Kierkegaard

AGRADECIMENTOS

I would like to thank Prof. Dr. Edson Prestes e Silva Júnior for offering me the possibility to pursue my master's degree at his chair, for his continuous support and many fruitful discussions.

Optimization Point Cloud in Keyframe-Based SLAM Using Fusion Information

RESUMO

SLAM (do inglês Simultaneous Localization and Mapping) Monocular baseado em Keyframes é uma das principais abordagens de SLAM Visuais, usado para estimar o movimento da câmera juntamente com a reconstrução do mapa sobre frames selecionados. Estas técnicas representam o ambiente por pontos no mapa localizados em um espaço tri-dimensional, que podem ser reconhecidos e localizados no frame. Contudo, estas técnicas não podem decidir quando um ponto do mapa se torna um outlier ou uma informação obsoleta e que pode ser descartada, ou combinar pontos do mapa que correspondem ao mesmo ponto tri-dimensional. Neste trabalho, apresentamos um método robusto para manter um mapa refinado. Esta abordagem usa o grafo de covisibilidade e um algoritmo baseado na fusão de informações para construir um mapa probabilístico, que explicitamente modela medidas de outlier. Além disso, incorporamos um mecanismo de poda para reduzir informações redundantes e remover outliers. Desta forma, nossa abordagem gerencia a redução do tamanho do mapa, mantendo informações essenciais do ambiente. Finalmente, a fim de avaliar a performance do nosso método, ele foi incorporado ao sistema do ORB-SLAM e foi medido a acurácia alcançada em datasets publicamente disponíveis que contêm sequências de imagens de ambientes internos gravados com uma câmera monocular de mão.

Palavras-chave: Information Fusion, Robot Vision, Visual SLAM, Keyframe-Based.

ABSTRACT

Keyframe-based monocular SLAM (Simultaneous Localization and Mapping) is one of the main visual SLAM approaches, used to estimate the camera motion together with the map reconstruction over selected frames. These techniques based on keyframes represent the environment by map points located in the three-dimensional space that can be recognized and located in the frames. However, many of these techniques cannot combine map points corresponding to the same three-dimensional point or detect when a map point becomes outlier and an obsolete information. In this work, we present a robust method to maintain a refined map that uses the covisibility graph and an algorithm based on information fusion to build a probabilistic map, which explicitly models outlier measurements. In addition, we incorporate a pruning mechanism to reduce redundant information and remove outliers. In this way our approach manages the map size maintaining essential information of the environment. Finally, in order to evaluate the performance of our method, we incorporate it into an ORB-SLAM system and measure the accuracy achieved on publicly available benchmark datasets which contain indoor images sequences recorded with a hand-held monocular camera.

Keywords: Information Fusion. Robot Vision. Visual SLAM. Keyframe-Based.

LIST OF ABBREVIATIONS AND ACRONYMS

AR	Augmented Reality
ATE	Absolute Trajectory Error
BA	Bundle Adjustment
COP	Closed-Form Online Pose-Chain Optimization
DBoW2	Bags of Binary Words
DLT	Direct Linear Transform
EKF	Extended Kalman Filter
FAST	Features from Accelerated Segment Test
FOV	Field of View
iSAM	Incremental Smoothing and Mapping
LSD	Large-Scale Direct Monocular
ORB	Oriented FAST and Rotated BRIEF
PTAM	Parallel Tracking and Mapping
RPE	Relative Pose Error
SFM	Structure From Motion
SLAM	Simultaneous Localization and Mapping
SSD	Sum of Squared Differences
SVO	Fast Semi-Direct Monocular Visual Odometry
SVD	Singular Valued Decomposition
RMSE	Root Mean Square Error
VSLAM	Visual Simultaneous Localization and Mapping

LIST OF FIGURES

Figure 1.1 Tasks that need to be solved by a robot in order to acquire accurate models of the environment. The overlapping areas represent different combinations of the mapping, localization, and path planning tasks. Image extracted from (MAKARENKO et al., 2002). 13

Figure 2.1 The essential SLAM problem. A simultaneous estimation of both robot and landmarks locations is required. The true locations are never known or measured directly. Observations are made between true robot and landmark locations. Image generated by the author. 17

Figure 2.2 The SLAM problem modeled as a Bayesian Network. The observed variables (orange nodes) are the observations $z_{1:t}$ and motion controls $u_{1:t}$, whereas the hidden variables (yellow nodes) are the robot's states $x_{0:t}$ and map of the environment m. Image generated by the author. 18

Figure 2.3 Graph-based SLAM, the robot poses and the landmarks are represented as nodes in a graph. The observations and motion measurements are the constraints encoded in the edges of this graph. Image generated by the author.............20

Figure 2.4 Visual SLAM methods. (left) Features-based methods, minimize projection error of features. (right) Direct methods, minimize photometric error, thereby exploit all image information. Image generated by the author. 23

Figure 2.5 Map representation in PTAM. Keyframes are depicted as red-white-green coordinates frames, landmarks as red crosses. Image extracted from PTAM (KLEIN; MURRAY, 2007)...26

Figure 2.6 Reconstruction and graphs in the sequence *fr3 long office household* from the TUM RGB-D Benchmark (STURM et al., 2012). Image extracted from (MUR-ARTAL; MONTIEL; TARDOS, 2015). 33

Figure 3.1 Covisibility graph (STUMM; MEI; LACROIX, 2013) with nodes representing keyframes and edges representing their covisibility. Image generated by the author...41

Figure 4.1 Images of the sequence *fr3_long_office* taken at different camera poses. Image generated by the author. 49

Figure 4.2 Images of the sequence *fr2_desk* taken at different camera poses. Image generated by the author. 50

Figure 4.3 Images of the sequence *fr2_desk_person* taken at different camera poses. Image generated by the author. 51

Figure 4.4 Resulting maps for the *fr3_long_office* sequence an the *fr2_desk_person* sequence with ORB-SLAM (left) and our method (right) . Image generated by the author...53

Figure 4.5 Relative Pose Error evaluation for *fr3_long_office*, *fr2_desk* and *fr2_desk_person* sequences of TUM dataset (STURM et al., 2012), showing the results obtained by ORB-SLAM without our method (left) and with our method (right). Image generated by the author. 57

Figure 4.6 Relative Pose Error for *fr3_long_office*, *fr2_desk* and *fr2_desk_person* sequences of TUM dataset (STURM et al., 2012), showing the results obtained by ORB-SLAM without our method (left) and with our method (right). Image generated by the author. 58

Figure 4.7 Absolute Keyframe Trajectory Error evaluation for *fr3_long_office*, *fr2_desk* and *fr2_desk_person* sequences of TUM dataset (STURM et al., 2012), showing the results obtained by ORB-SLAM without our method (left) and with our method (right). Additionally, blue points represent those keyframes that are not considered by the alignment method. Image generated by the author..........59

LIST OF TABLES

Table 4.1 The values for the parameters of our method. 46
Table 4.2 Quantitative evaluations for 11 sequences from the TUM benchmark (STURM et al., 2012). From left to right, the columns show the dataset name, their length and length of filming. The number of keyframes, map points and their percentages of reduction, respectively. The root mean square error (RMSE) of absolute keyframe trajectory (ATE) and the relative pose (RPE) in terms of translation. 47
Table 4.3 Quantitative evaluations for 10 separate experiments on the *fr3_long_office* sequence. 52
Table 4.4 Quantitative evaluations for 10 separate experiments on the *fr2_desk* sequence. 52
Table 4.5 Quantitative evaluations for 10 separate experiments on the *fr2_desk_person* sequence. 52

CONTENTS

1 INTRODUCTION......12
1.1 Motivation......14
1.2 Objectives......15
1.3 Organization......15
2 SLAM FOUNDATION......16
2.1 SLAM......16
2.2 Graph-Based SLAM......20
2.3 Visual SLAM......22
2.4 Monocular Keyframe-Based SLAM......24
2.5 Related Work......24
2.5.1 PTAM......25
2.5.2 ORB-SLAM......32
3 MAP POINT OPTIMIZATION IN KEYFRAME-BASED SLAM......39
3.1 Map Representation......39
3.2 Probabilistic Depth Sensor......39
3.3 Depth Bayesian Inference......40
3.4 Map Optimization......44
4 EXPERIMENTS......46
4.1 Removing outliers......47
4.2 Relative Pose Error......54
4.3 Absolute Keyframe Trajectory Error......55
5 FINAL REMARKS AND FUTURE WORK......60
REFERENCES......62
APPENDIX A — PROBABILITY AND STATISTICS......67
A.1 Normal Distribution......67
A.2 Beta Distribution......67

1 INTRODUCTION

Humans have always dreamed of building skillful and intelligent machines that are subject to their command and can take the annoying, tedious and dangerous jobs that are repetitive, tiresome and dangerous, where the human cognitive abilities are not necessary and human lives are exposed to risks. Selection, transportation, monitoring and delivery of products as fast as possible in e-commerce companies with enormous warehouses and many customers; observation and monitoring of potentially dangerous situations such as protests or large scale sport events; inspection of damaged nuclear reactors; removal of landmine; deactivate bombs in the case of terrorist attacks; explore the extensive space and the deep sea; search, location and rescue of survivors after natural disasters such as earthquakes, fires or landslides in tunnels under the soil are some of them. For these jobs the idea of using robots is promising. These machines with certain intelligence could be interacting, assisting and exploring in homes and schools; in hospitals and factories; on other planets and in the oceans. However, developing a robot capable to replace a human on these scenarios is a very complex and challenging task.

In general, building robots with such skills requires the solution of three tasks, which are mapping, localization and path planning (THRUN; BURGARD; FOX, 2005). Firstly, mapping is the problem of interpreting and integrating the knowledge gathered with the robot's sensors into a given representation known as map, which can be manipulated. For the most part of this representation falls into one of three main classes (CADENA et al., 2016): occupancy grids, geometric maps, or landmark-based maps. The mapping task is done while the robot is navigating an unknown environment. In contrast to this, localization is the problem of estimating the pose of the robot in an already mapped area. Typically, two types of localization capabilities are distinguished (CADENA et al., 2016) : position tracking or local localization, which is the ability of tracking the correctly current pose when the initial pose is known; and global localization, where the robot should be able to localize itself when its initial pose is unknown. Finally, the path planning problem involves to answer the question of how to efficiently guide the robot to a desired location. The route must be safe, free of obstacles and if possible avoid regions with difficult traversing, such as narrow spaces. Unfortunately, these three tasks cannot be solved independently of each other. Because, before a robot can answer the question of what the environment looks like given a set of observations, it needs to know from which locations these observations have been made. At the same time, it is hard to estimate the current pose of the robot without a map. In addition, planning a path to a goal location is also tightly coupled with the information about the current robot's pose and the knowledge of what the environment looks like.

Figure 1.1: Tasks that need to be solved by a robot in order to acquire accurate models of the environment. The overlapping areas represent different combinations of the mapping, localization, and path planning tasks. Image extracted from (MAKARENKO et al., 2002).

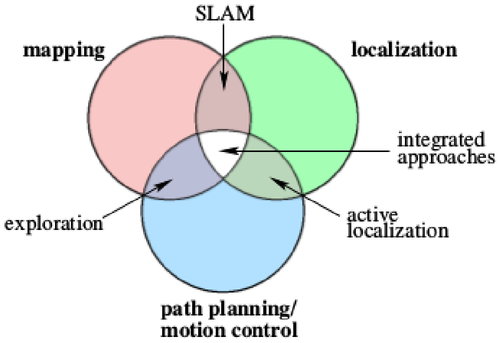


Figure 1.1 shows all possible combinations of these tasks (MAKARENKO et al., 2002). The simultaneous localization and mapping problem (SLAM) is when the robot moves in an unknown environment and incrementally building a consistent map while simultaneously determining its location within this map. Since robot motion is subject to error, the mapping problem necessarily induces a robot localization problem. In this way, SLAM is a chicken-and-egg problem: a map is needed for localization and a precise localization is needed for mapping. Next, active localization problem is when the robot, based on a known map, tries to recalculate the path to follow during its traversing in which it can choose motions intended specifically to provide the opportunity to reduce the uncertainty of its own pose. This skill is important for a robot that lacks an accurate source of odometry. In contrast to this, exploration is the robot's ability to navigate in an environment where it does not have any knowledge. The robot must move to a region to gain as much new information as possible in order to cover the environment and build a complete map in the shortest possible time. The exploration directly affects the input of the robot system. Finally, the center area of the diagram represents the so-called integrated approaches which address mapping, localization, and path planning simultaneously. This means that a robot can acquire sensor data by moving autonomously through its environment while at the same time building a map. Furthermore, whenever the robot is moving, it considers actions to improve its localization, to acquire information about unknown terrain and to improve its map model by revisiting areas where it is uncertain about. In the end, the

robot is assumed to have learned an accurate model of the whole environment as well as determined its own pose relative to this model. However, performing the three main necessary tasks to an autonomous robot is already a challenge.

1.1 Motivation

Maps serve as important resources to build truly autonomous robots by providing them with the necessary relevant information about their environment. The use of maps enables robots to perform their tasks more reliably, flexibly, and efficiently. Additionally, maps allow limiting the error produced when estimating the robot's states. In recent years the interest in using cameras as main sensors in SLAM has increased and some authors have been concentrating on building 3D maps using visual information (NEWCOMBE; LOVEGROVE; DAVISON, 2011; ENGEL; SCHOPS; CREMERS, 2014; MUR-ARTAL; MONTIEL; TARDOS, 2015). The reasons for this interest are not only because of their low power consumption, small size and cost, but also for their ability to provide rich information about the surrounding environment, such as color, texture, motion and structure.

In that sense, SLAM algorithms using visual information are known in the literature as Visual SLAM (DAVISON; KITA, 2001) and many of these methods represent the scene as a set of sparse 3D landmarks corresponding to discriminative features in the environment (e.g., points, lines, polygons) (LU; SONG, 2015; ZHOU et al., 2015; PUMAROLA et al., 2017). Moreover, a common assumption underlying these representations is that the landmarks are distinguishable and provide descriptors which establishes a data association between each measurement and the corresponding landmark. Thus, the robot can operate for an extended period of time and revisit a place several times, while new information is continuously added to the map. However, this is a problem since the size of the map grows with the mapping process duration and not only with the size of the area explored. Furthermore, new information can be repetitive or outlier and current visual SLAM techniques are vulnerable to them. The inclusion of a single outlier degrades the quality of the estimate, which in turn degrades the capability of discerning outliers later on. Therefore, it is necessary to have approaches that can deal with repetitive information and outliers to maintain a refined and accurate representation of the map.

1.2 Objectives

This work proposes a new method to maintain a refined map through the pruning of map points that can have a direct influence on the performance of the Visual SLAM process, such as outliers generated from a poor depth estimate, or map points, which have been visualized only in some frames and over time have become obsolete; and also deals with repetitive information to maintain a good quality map and counteract the effect of the frequent addition of features. Our method uses a covisibility graph (STUMM; MEI; LACROIX, 2013) similar to the one employed by ORB-SLAM (MUR-ARTAL; MONTIEL; TARDOS, 2015) and an information fusion algorithm as Forster et al. (FORSTER; PIZZOLI; SCARAMUZZA, 2014) or Pizzoli et al. (PIZZOLI; FORSTER; SCARAMUZZA, 2014), to reduce the number of outliers and combine repetitive information in the map. Then, we represent the depth information of each map point as a mixture of probability distributions and take advantage of the keyframe neighbourhoods created in the covisibility graph to update these probabilities and thus maintain the highest possible accuracy of depth estimation. Additionally to ensure a few outliers in the map and reduce the number of redundant keyframes, a pruning policy based on the depth accuracy of the map points is performed.

1.3 Organization

This dissertation is divided as follows: Chapter 2 reviews the fundamentals of SLAM and Visual SLAM, presenting two Keyframe-Based SLAM methods: PTAM (KLEIN; MURRAY, 2007) and ORB-SLAM (MUR-ARTAL; MONTIEL; TARDOS, 2015). Chapter 3 introduces the core ideas of our method. Then, an overview of the proposed algorithm is presented followed by its formal derivation to clarify in details each process of our approach. Chapter 4 presents the tested environments used to appraise the present work. Later, the results obtained by our approach are compared to the ORB-SLAM. The differences between the tested techniques are highlighted and discussed. Finally, Chapter 5 summarizes the result obtained of an extensive set of experiments and presents the conclusions of this work.

2 SLAM FOUNDATION

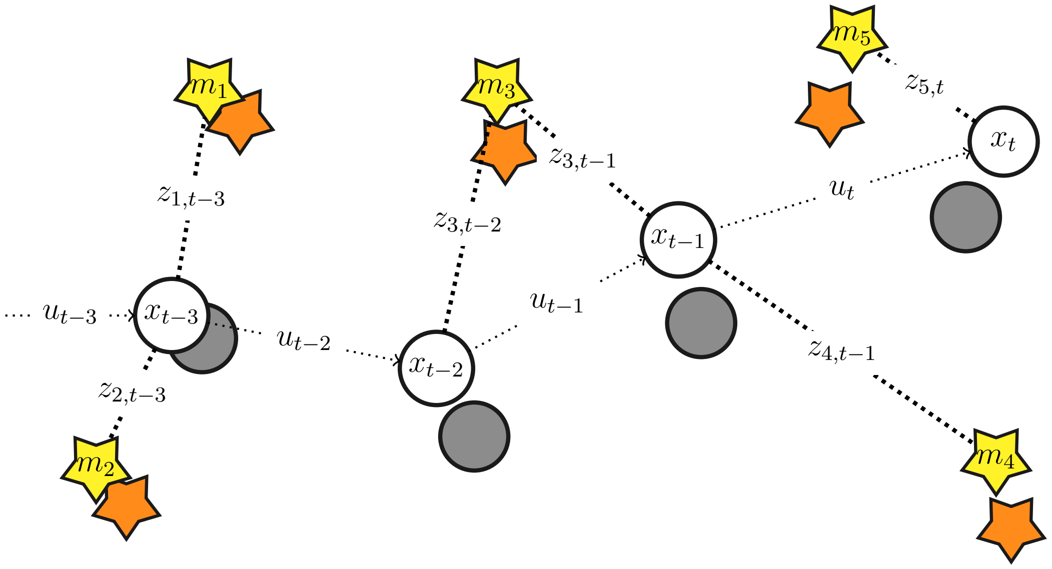
This chapter starts with a review of the foundation of simultaneous localization and mapping techniques, describing the problem and introducing its mathematical formulation. Then, the division of SLAM problem is introduced. In Section 2.2 we describe a graph-based SLAM as an optimal strategy to represent the SLAM problem. In Section 2.3 we describe the use of visual sensors in SLAM problem and its advantages. In Section 2.4 we introduce a specific type of visual SLAM problem, which uses a monocular camera. Finally, In Section 2.5 we introduce two main monocular SLAM methods. First, we introduce the PTAM (KLEIN; MURRAY, 2007) and later ORB-SLAM (MUR-ARTAL; MONTIEL; TARDOS, 2015).

2.1 SLAM

One of the most fundamental problems in robotics is the simultaneous localization and mapping problem, better known in its abbreviated form as SLAM. The SLAM problem is generally considered as key requisite in the pursuit of building truly autonomous robots. SLAM addresses the problem of a robot navigating an unknown environment. While navigating, the robot perceives its environment and constructs a coherent map, later this map is used in the deduction of its pose. The reason why SLAM is difficult arises precisely from the interaction between the localization and the map building processes. While for localization the robot needs to know a **priori** map, for map building the robot needs to know its precise pose in the map. This is an example of the chicken-and-egg problem where both actions are strongly dependent on each other.

Figure 2.1 shows the essential SLAM problem. Consider a robot in an instant of time t with a state x_t moving in an environment consisting of a population of k landmarks denoted by m. The robot is equipped with proprioceptive sensors that can measure its own motion u_t and exteroceptive sensors that can take measurement of observations $z_{k,t}$ that are defined as the relative location between robot with state x_t and a nearby landmark m_k. The objective of the SLAM problem consists in the simultaneous estimation of the robot's states and the map of the environment. The robot's state x_t also is called robot's pose because it generally represents the pose of the robot (position and orientation), although other quantities can be included, such as robot's velocity, sensor biases and calibration parameters. On the other hand the map (set of landmarks) is a representation of aspects of interest describing the environment in which the robot operates.

Figure 2.1: The essential SLAM problem. A simultaneous estimation of both robot and landmarks locations is required. The true locations are never known or measured directly. Observations are made between true robot and landmark locations. Image generated by the author.

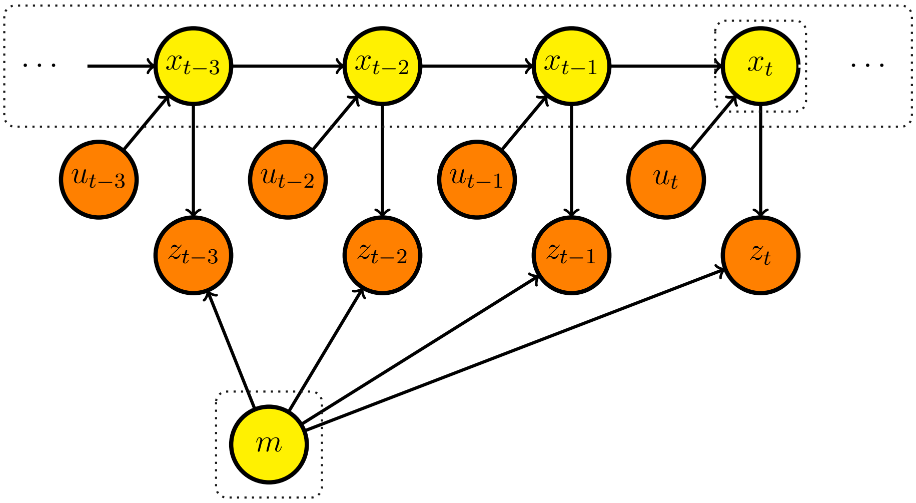


	Robot	Landmark
True Pose	○	☆
Estimated Pose	●	★

Along the time the robot moves, increasing the uncertainty over its localization due to the inherent noise in the sensor measurements and the uncertain about the motion, therefore the estimation of the robot's states and map are approximate. However, there are many ways to model this uncertainty. For example, the extended Kalman filter SLAM (EKF-SLAM) proposed by Smith et al. (SMITH; CHEESEMAN, 1986) represents the uncertainty using an approximated Gaussian distribution, while the GMapping technique proposed by Grisetti et al (GRISETTI; STACHNISS; BURGARD, 2007) uses a Rao-Blackwellized particle filter scheme. In this way, the SLAM problem usually is described by means of probabilistic tools like a Bayesian network.

Figure 2.2 shows the relations between the observed and the hidden variables of this Bayesian network. The observed variables, the set of motion controls $u_{1:t} = \{u_1, u_2, ..., u_t\}$ and the set of observations $z_{1:t} = \{z_1, z_2, ..., z_t\}$ are those ones that can be directly measured. Hidden variables, the set of robot's states $x_{0:t} = \{x_0, x_1, ..., x_t\}$ and the map $m = \{m_1, m_2, ..., m_k\}$ are the variables not directly measured by the robot and what a SLAM technique have to estimate. According to this model, the observed variables are dependent of the hidden variables which can be inferred from the observed ones.

Figure 2.2: The SLAM problem modeled as a Bayesian Network. The observed variables (orange nodes) are the observations $z_{1:t}$ and motion controls $u_{1:t}$, whereas the hidden variables (yellow nodes) are the robot's states $x_{0:t}$ and map of the environment m. Image generated by the author.



So to compute a good estimation, this Bayesian network requires two probabilistic models: a motion model and an observation model. The motion model is described in terms of a probability distribution on state transitions. That is, the state transition is assumed to be a Markov process in which the next state x_t depends only on the immediately preceding state x_{t-1} and the applied control u_t:

$$P\left(x_t \mid x_{t-1}, z_{1:t}, u_{1:t}\right) = P\left(x_t \mid x_{t-1}, u_t\right) \tag{2.1}$$

On the other hand, the observation model describes the probability of making an observation z_t based on the current information. So, once the robot's state and map are defined, observations are conditionally independent:

$$P\left(z_t \mid x_{0:t}, z_{1:t}, u_{1:t}, m\right) = P\left(z_t \mid x_t, m\right) \tag{2.2}$$

In other words, these models represent the uncertain robot motion and the sensor measurements corrupted by noise like probabilistic distributions. In this way, from a probabilistic perspective there are two main forms of the SLAM problem: online SLAM and full SLAM, which are both of equal importance.

Online SLAM formulation involves the estimation of the map and the robot's state persist only at time t. As shown at Equation 2.3, the robot's state x_t at time t and the map estimation m are the posterior probability computed from all observed states whereas $z_{1:t}$ are the observations and $u_{1:t}$ are the controls. These algorithms use the last estimation of map and robot's state to compute the next robot's state.

$$P\left(x_t, m \mid z_{1:t}, u_{1:t}\right) \tag{2.3}$$

On the other hand, full SLAM formulation involves the estimation of all robot's states that persist along the time $1 : t$, instead of just the current state x_t at time t. As shown at Equation 2.4, the robot's states $x_{1:t}$ over the entire trajectory and the map estimation m are the posterior probability computed from all observed states whereas $z_{1:t}$ are the observations and $u_{1:t}$ are the controls.

$$P\left(x_{1:t}, m \mid z_{1:t}, u_{1:t}\right) \tag{2.4}$$

In particular, online SLAM is the result of integrating out past poses estimates from the full SLAM problem (THRUN; BURGARD; FOX, 2005):

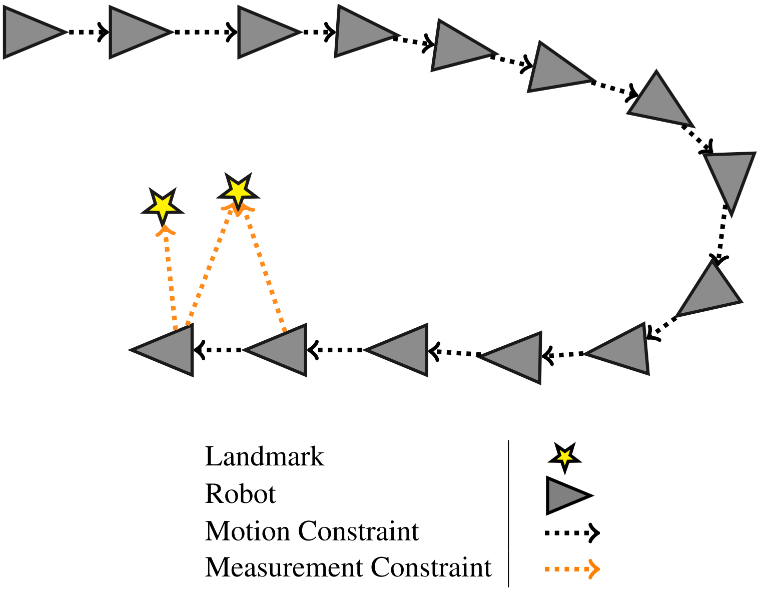
$$P\left(x_t, m \mid z_{1:t}, u_{1:t}\right) = \int\int ... \int P\left(x_{1:t}, m \mid z_{1:t}, u_{1:t}\right) dx_1 dx_2 ... dx_{t-1} \tag{2.5}$$

The aim of both formulations is to estimate the posterior which captures all there is to be known about the map and the robot's pose or the trajectory. In the literature, a large variety of solutions to the SLAM problem are available and can be classified either as filtering or smoothing (GRISETTI et al., 2010). In filtering approaches the estimation is augmented and refined by incorporating the new measurements as they become available. They are usually referred to online SLAM methods. Many techniques like Kalman, information filter (SMITH; SELF; CHEESEMAN, 1990; CASTELLANOS et al., 1999; EUSTICE; SINGH; LEONARD, 2006; THRUN et al., 2004) or particle filters (MONTEMERLO et al., 2002; HAHNEL et al., 2003; GRISETTI; STACHNISS; BURGARD, 2007) fall into this category. On the other hand, smoothing approaches address the full SLAM problem, estimating the full trajectory of the robot and map from the set of measurements (LU; MILIOS, 1997; DELLAERT; KAESS, 2006; OLSON; LEONARD; TELLER, 2006).

2.2 Graph-Based SLAM

Filtering techniques such as EKF-SLAM have led to hundreds of extensions for years. Nevertheless, a key disadvantage is that the robot's states are not stored therefore it is impossible revisit past states. Thereby, in 1997 Lu and Milios (LU; MILIOS, 1997) proposed an efficient and intuitive formulation of the full SLAM problem, but only became popular years later due to advances on error minimization techniques (THRUN; MONTEMERLO, 2006). This formulation is known as graph-based SLAM shown at Figure 2.3 and involves building a graph where the robot's states $x_{0:t}$ and the set of landmarks m are represented as nodes of the graph. The observations $z_{1:t}$ and motion measurements $u_{1:t}$ are the spatial constraints represented by the edge of this graph. These constraints represent the probability distribution over the relative transformations between two nodes that connect two robot's states (motion constraint) or the robot's state to landmark (measurement constraint). Obviously such constraints are always affected by noise, so it is necessary to find the spatial configuration of the nodes that best satisfies such constraints. This involves solving a error minimization problem.

Figure 2.3: Graph-based SLAM, the robot poses and the landmarks are represented as nodes in a graph. The observations and motion measurements are the constraints encoded in the edges of this graph. Image generated by the author.



Since the publications of graph-based SLAM, many approaches to minimizing the error have been proposed. For example Howard et al. (HOWARD; MATARIC; SUKHATME, 2001) apply relaxation to localize the robot and build a map. Frese et al. (FRESE; LARSSON; DUCKETT, 2005) propose a variant of Gauss-Seidel relaxation called multi-level relaxation which applies to different resolutions. Dellaert and Kaess (DELLAERT; KAESS, 2006) were the first to exploit sparse matrix factorizations to solve the linearized problem in offline SLAM. Kaess et al. (KAESS; RANGANATHAN; DELLAERT, 2007) presented iSAM, a fast incremental matrix factorization method. Konolige et al. (KONOLIGE et al., 2010) proposed an open source implementation of a pose-graph method which constructs the linearized system in an efficient way. Olson et al. (OLSON; LEONARD; TELLER, 2006) presented an efficient non-linear optimization algorithm that rapidly recovers the robot trajectory, which is based on the stochastic gradient descent. Later, Grisetti et al. (GRISETTI et al., 2008) proposed an extension of Olson's approach that uses a tree parametrization of the nodes in 2D and 3D, increasing the convergence speed.

Besides, graph-based SLAM has an observation model which is multi-modal. This means that a single observation z_t might result in multiple potential edges connecting different poses, and one needs to determine the most likely constraint resulting from an observation. This decision depends on the probability distribution over the robot's poses. Such strategy is weak and susceptible to false positive matches especially when the sensor used by the robot is poor. Such problem is known as data association and is usually treated in the construction of the graph.

The graph-based SLAM problem can be divided in two tasks: graph construction, usually called front-end, that seeks to interpret the sensor data to create nodes and edges of the graph representing the robot's states and their constraints between them; and graph optimization also known as back-end, which aims to minimize the error introduced by the constraints given in the edges of the graph and thus determine the most likely nodes configuration.

In this sense, we explain the common idea to compute the maximum likelihood of a graph configuration. Let $x = (x_1, x_2, ..., x_t)^T$ be a vector of parameter, where x_i describes the pose of ith node. The constraint over each edge between two nodes x_i and x_j is represented by a Gaussian distribution with mean z_{ij} and information matrix Ω_{ij} (which is the inverse of the covariance matrix Σ_{ij}). Therefore, the error observation $e\left(x_i, x_j\right)$ is computed as the difference between the measurement z_{ij} gathered by the robot and the predicted measurement $\bar{z}\left(x_i, x_j\right)$, as shown:

$$e\left(x_i, x_j\right) = z_{ij} - \bar{z}\left(x_i, x_j\right) \tag{2.6}$$

For simplicity of notation $e\left(x_i, x_j\right)$ will be represented as e_{ij}. Hence the log-likelihood l_{ij} of a measurement z_{ij} is given by:

$$l_{ij} \propto e_{ij}^T \Omega_{ij} e_{ij} \tag{2.7}$$

Using this information the back-end minimizes the error produced by all disagreements between the observations and predictions. This minimization is done by moving the nodes inside its zones of uncertainty in a manner to reach an optimized global consistency, without losing the local constraints identified by the robot. Therefore, to compute the maximum likelihood configuration of nodes x^*, we seek to solve the Equation 2.9 that minimizes $F(x)$, the sum of the log-likelihood for all pairs of indices for which observations exist in the set of constrains C.

$$F(x) = \sum_{(i,j) \in C} e_{ij}^T \Omega_{ij} e_{ij} \tag{2.8}$$

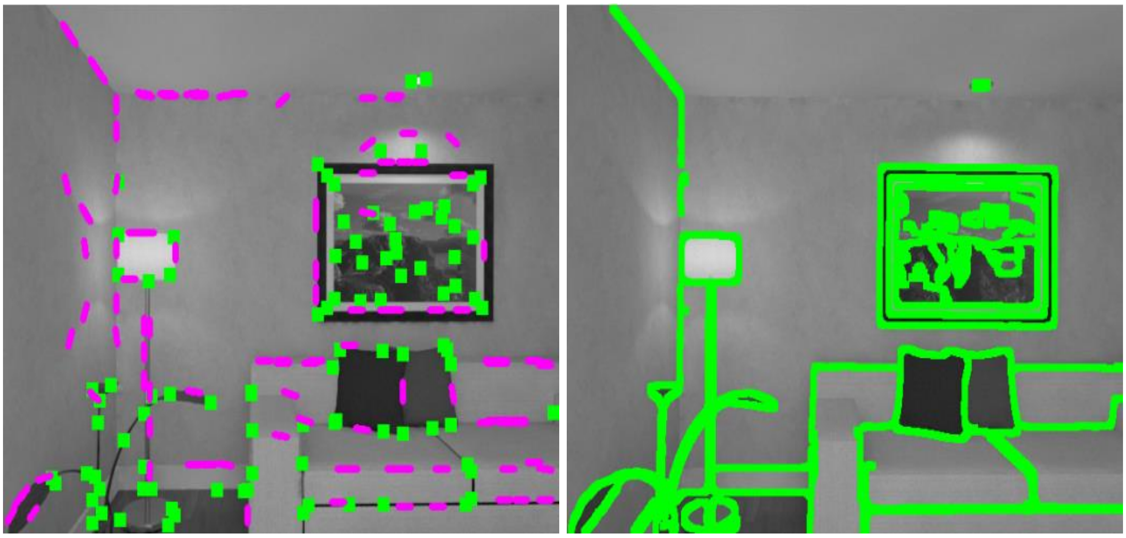
$$x^* = \underset{\mathbf{x}}{\arg\min}\, F(x) \tag{2.9}$$

To compute the maximum likelihood x^*, it is usually employed methods such as conjugate gradient, Gauss-Newton and Levenberg-Marquardt (KUMMERLE et al., 2011). Nevertheless, the approach based on a stochastic gradient descent method introduced by Olso et al. (OLSON; LEONARD; TELLER, 2006) has shown better results with less computational time.

2.3 Visual SLAM

In recent years many approaches using visual sensors (monocular, stereo and RGB-D cameras) in SLAM have been proposed as a replacement for the traditional laser range and they are known as Visual SLAM. The use of visual sensors such as cameras has several key advantages, for example, less energy requirement, low price, small size and weight, as well as also the amount of information that can be gathered about the environment. In addition, the set of consecutive images can be used to track the camera motion known as visual odometry that was introduced in the work of Nister et al. (NISTER; NARODITSKY; BERGEN, 2004).

Figure 2.4: Visual SLAM methods. (left) Features-based methods, minimize projection error of features. (right) Direct methods, minimize photometric error, thereby exploit all image information. Image generated by the author.



Nevertheless, Visual SLAM aims not only at tracking the camera motion but also at building a globally consistent map. Therefore, Visual SLAM methods are often built upon visual odometry and can be divided into two classes: feature-based and direct, which are briefly discussed next. First, feature-based SLAM are methods that extract a sufficiently large set of features such as points (MUR-ARTAL; MONTIEL; TARDOS, 2015) or lines (**??**) from each image and match them across multiple frames. These features and their corresponding matches are the input to the joint process of estimating the camera pose and map representation. The majority of Visual SLAM methods fall into this class, for example, Davison et al. (DAVISON et al., 2007) proposed a real-time algorithm which can recover the 3D trajectory of a monocular camera. Klein et al. (KLEIN; MURRAY, 2007) proposed to split tracking and mapping in two separate tasks processed in parallel. Newcombe et al. (NEWCOMBE et al., 2011) used kinect camera to rapidly create detailed 3D reconstructions of an indoor scene. Taguchi et al. (CANSIZOGLU; TAGUCHI; RAMALINGAM, 2016) used the valid pixel's depths to compute the invalid pixel's depths in kinect camera.

Second, direct SLAM. These types of methods do not require a matching step, they work directly with the intensity values of the pixels. An important advantage is the level of accuracy that they can attain due to the exploitation of all image information, such as areas where the intensity of the gradient is sufficiently large. For instance, Engel et al. (ENGEL; SCHOPS; CREMERS, 2014) proposes LSD-SLAM, a direct SLAM method based on image intensities which allows to build large-scale maps. Foster et al. (FORSTER; PIZZOLI; SCARAMUZZA, 2014) proposes SVO, a new method that combine a feature-based SLAM and direct SLAM methods. Newcombe et al. proposes DTAM (NEWCOMBE; LOVEGROVE; DAVISON, 2011) that uses the image alignment for camera tracking and dense reconstruction of the scene. Dubbelman et al. (DUBBELMAN; BROWNING, 2015) proposes COP-SLAM, a solution called closed-form online pose-chain SLAM that performs accurate visual odometry and reliable appearance-based loop detection.

2.4 Monocular Keyframe-Based SLAM

A specific problem in Visual SLAM is known as monocular SLAM, which uses a single video camera as sensor. Initially, monocular SLAM was solved by filtering-based method, where every frame is processed by the filter to jointly estimate the map representation and the camera pose. However recently keyframe-based methods have become popular. They retain a selected subset of previous observations called keyframes that explicitly represent past knowledge gained. These keyframes are snapshots taken by the camera at various instants in the time and are composed of regions of interest called keypoints, which represent small and distinguishable areas in the image. In addition, keypoints are the two dimensional projections of landmarks on the image plane. There are many approaches to compute keypoints such as SIFT (LOWE, 1999), SURF (BAY; TUYTELAARS; GOOL, 2006) or FAST (ROSTEN; DRUMMOND, 2006). One of the most representative keyframe-based SLAM is probably PTAM (KLEIN; MURRAY, 2007), because it was the first work that introduced the idea of splitting camera tracking and mapping in parallel threads, and proved to be successful for real time augmented reality (AR) applications in small environment.

2.5 Related Work

Over the last decade, numerous efforts have been made towards minimizing the computational requirements of SLAM by reducing the amount of variables (observations and poses) in the state space, while keeping the sparse structure of the problem. However recently due to the popularity of graph-based optimization solutions for SLAM, researchers investigated how to reduce the number of nodes in the SLAM graph. Some approaches focus on which node to remove from the graph and how to deal with the resulting graph. Konolige et al. (KONOLIGE; BOWMAN, 2009) clustered nodes in the graph according to their spatial distance. Among each cluster, they removed the least recently used nodes, in order to keep a limited number of nodes and still capture the dynamic nature of the environment. A similar idea has also been introduced by Eade et al. (EADE; FONG; MUNICH, 2010), who proposed to remove nodes without data or with similar observations to existing nodes in their vicinity. Moreover, Ila et al. (ILA; PORTA; ANDRADE-CETTO, 2010) introduce a principled on-line approach for Pose SLAM, which only keeps non-redundant poses and highly informative links. While Johannsson et al. (JOHANNSSON et al., 2013) reuse already existing poses in previously mapped areas, keeping the number of poses bounded by the size of the explored environment and using the new measurements to improve the map.

Similarly, keyframe-based approaches are among the first attempt to reduce the pose graph for visual SLAM. They create a sparse pose graph, where each node is a selected keyframe representing the prominent visual appearances and variations in order to keep the density of nodes constant. In this section we will detail two monocular keyframe-based SLAM methods. They will serve as comparison to our approach in Chapter 4. First an explanation and mathematical derivation of PTAM (KLEIN; MURRAY, 2007) is introduced. Later, a detailed revision of ORB-SLAM (MUR-ARTAL; MONTIEL; TARDOS, 2015) is presented. PTAM is a system in real time that infers the motion of the hand-held monocular camera from a video frames in a small AR workspace. While ORB-SLAM is a system that operates in real time, in small and large, indoor and outdoor environments.

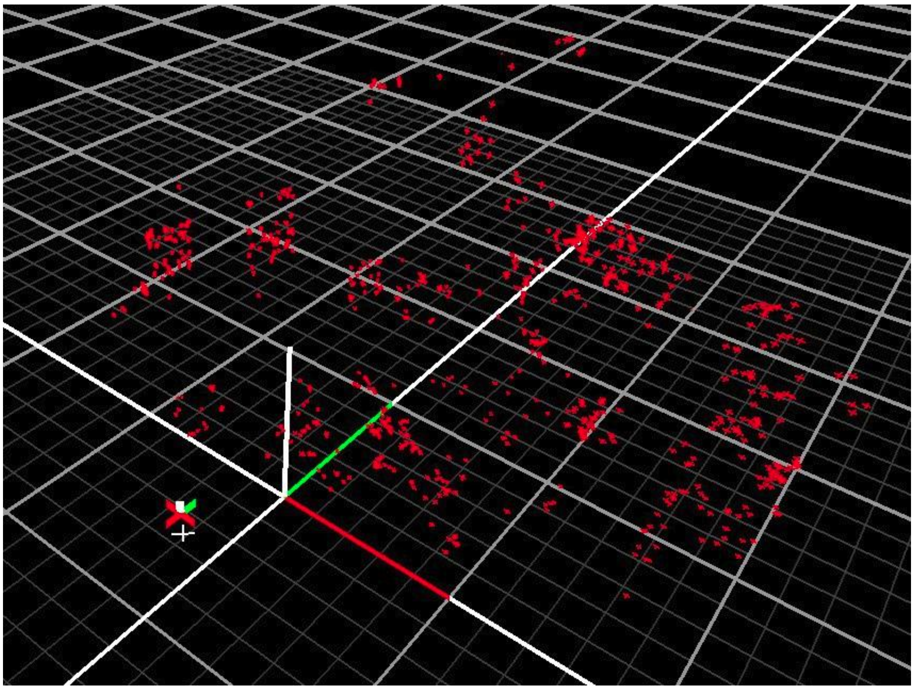
2.5.1 PTAM

Parallel Tracking and Mapping also known as PTAM was proposed in 2007 by Klein and Murray (KLEIN; MURRAY, 2007). PTAM is a method to estimate the pose of a hand-held monocular camera in a unknown scene. It is simple and efficient because it uses FAST corners, which can be detected and processed quickly. PTAM also splits tracking and mapping into two separate tasks which are processed in parallel, one thread deals with the task of robustly tracking hand-held camera motion, while the other thread produces a 3D map from previously observed video frames. In this sense, PTAM is considered an important milestone in the development of SLAM monocular methods. Nevertheless, the advantages of PTAM are also its limitations, since it is a relatively simple system that does not extract structural meaning from the reconstructed environment other than the points detected by the FAST corners . Furthermore, it is necessary to perform a stereo initialization in a very specific way.

Map Representation

The map consists of a collection of N landmarks located in a world coordinate frame, where each landmark represents a locally planar textured patch of size 8×8 pixels. The ith landmark has homogeneous coordinates ${}^{w}p_i = \left({}^{w}x_i,\ {}^{w}y_i,\ {}^{w}z_i, 1\right)^T$ in world coordinates frame w; a normal unit vector $n_i \in \mathbb{R}^3$ and a reference to the patch coordinates $\left(u_i, v_i\right)^T$ which is the center pixel of the source patch. The map also contains M keyframes, where each jth keyframe ${}^{w}K_j$ has coordinates frame associated to a central position of the camera; a matrix transformation between the coordinates frame and the world coordinates ${}^{w}E_j$. This matrix transformation contains a rotation and a translation component and is a member of the Lie group $SE\left(3\right)$, the set of 3D rigid body transformations. Furthermore, each keyframe also has a four level image pyramid, where the level

Figure 2.5: Map representation in PTAM. Keyframes are depicted as red-white-green coordinates frames, landmarks as red crosses. Image extracted from PTAM (KLEIN; MURRAY, 2007).



zero stores the full image and this is sub-sampled down to level three. Additionally, each landmark stores a reference to a source keyframe typically the first keyframe in which this landmark was observed.

Tracking Process

Tracking is the process that calculates the corresponding camera position to every new frame taken with the assumption that a map of landmarks has already been created. When a new frame is acquired, the image pyramid is built and the FAST corner detector is run on each pyramid level. Later, a **priori** pose estimate is generated from a motion model. Once the pose is estimated, the current map of landmarks can be projected into the new image plane and the patches correspondences can be searched over the neighbourhood of the projected location. Initially few landmarks are projected and the camera pose is updated from the matches found. Next, a large number of landmarks are projected, their matches are searched and the pose estimate for the frame is computed from all matches found.

Therefore, to project the map of landmarks on the image plane, they are first transformed from the world coordinates frame w to the camera centred coordinates frame c. This is done by multiplying a transformation matrix ${}^{w}E_c$ with the landmark position in world coordinates frame ${}^{w}p_i$:

$$ {}^{c}p_i = {}^{w}E_c \, {}^{w}p_i \tag{2.10}$$

Where ${}^{c}p_i$ is the landmark position in camera centred coordinates. Next, to project the landmarks, in the camera centred coordinates frame c, into image coordinates frame is used a calibrated camera projection model:

$$\begin{pmatrix} u_i \\ v_i \end{pmatrix} = \Theta\left({}^{c}p_i \right) \tag{2.11}$$

Where Θ is the pinhole camera projection model which defines the relationship between the landmark ${}^{c}p_i = \left({}^{c}x_i, \, {}^{c}y_i, \, {}^{c}z_i, 1 \right)^T$, in the camera centred coordinates frame c, and the pixel $(u_i, v_i)^T$ into image coordinates frame, as shown:

$$\Theta\left({}^{c}p_i \right) = \begin{pmatrix} u_c \\ v_c \end{pmatrix} + \begin{pmatrix} f_u & 0 \\ 0 & f_v \end{pmatrix} \frac{\hat{r}}{r} \begin{pmatrix} \frac{{}^{c}x_i}{{}^{c}z_i} \\ \frac{{}^{c}y_i}{{}^{c}z_i} \end{pmatrix}, \tag{2.12}$$

Here f_u and f_v are the focal lengths; $(u_c, v_c)^T$ is the principal point of the frame c and the field of view model (FOV) is the radial distortion model proposed by Devernay et al. (DEVERNAY; FAUGERAS, 2001). This model transforms the radial distortion r to $\hat{r}$ using the image distortion d and it is defined as:

$$r = \sqrt{\frac{{}^{c}x_i + {}^{c}y_i}{{}^{c}z_i}} \tag{2.13}$$

$$\hat{r} = \frac{1}{d}\arctan\left(2r\tan\left(\frac{d}{2}\right)\right) \tag{2.14}$$

On the other hand, a fundamental requirement of the tracking process (and also the mapping) is the ability to differentiate Equation 2.11 with respect to changes in camera pose ${}^{w}E_c$. Therefore, changes in the camera pose are represented by the multiplication of ${}^{w}E_c$ with the camera motion M:

$${}^{w}\hat{E}_c = M\ {}^{w}E_c = exp\left(\xi\right)\ {}^{w}E_c \tag{2.15}$$

Where, M is also a member of the Lie group $SE\left(3\right)$ and can be mapped to the tangent space or twist that represents the camera pose or motion as a six-dimensional vector $\xi = (w_x, w_y, w_z, v_x, v_y, v_z)^T$, with the first three elements represent the angular velocity and the latter three elements represent the linear velocity. Twist is mapped by exponential map (ETHAN, 2014) and the inverse is done by the logarithmic map (ETHAN, 2014). This representation of the camera pose and motion allow to readily obtain in a closed form, the differentiation of Equation 2.15 and Equation 2.11.

Once the projection of the landmarks into image has been made, for each landmark a set of patches (patches centered on FAST corners), lying inside a circular region around the projection of the landmark, is selected. This set is first affine warped by matrix $A \in \mathbb{R}^{2\times 2}$ to accommodate view point changes. Later, the patches are searched in the image pyramid level whose scale matches with the original patches. Next, the best match is selected evaluating SSD (sum of squared differences) scores. Then, the error projection e_k, defined by the difference between the coordinates of the kth patch and the coordinates of the projection matched ${}^{c}p_k$, is calculated:

$$e_k = \begin{pmatrix} \hat{u}_k \\ \hat{v}_k \end{pmatrix} - \Theta\left(exp\left(\xi\right)\ {}^{c}p_k\right) \tag{2.16}$$

By last, to update the camera pose, the objective function of the projection error is minimized iteratively with respect to the current camera pose ξ:

$$\hat{\xi} = \arg\min_{\xi} \sum_{k \in S} \mathcal{G}\left(\frac{\mid e_k \mid}{\sigma_k}, \mathcal{T}\right), \tag{2.17}$$

Where, $\hat{\xi}$ represents the camera pose updated; S is the set of patches successfully matches; σ_k is the assumed measurement noise; $\mathcal{T}$ is the standard deviation derived from all the residuals (median based) and $\mathcal{G}$ is the Tukey biweight objective function introduced in the work of Huber et al. (HUBER, 2011).

Eventually tracking process can failure, for this reason for each frame is estimated the quality of tracking and it is computed as the fraction of landmarks in the map that have been successfully observed. If this fraction falls below than a certain threshold, tracking quality is considered poor because such frames would likely be of poor quality. If the fraction falls below an even lower threshold for more than few frames, then tracking is considered lost. Then, when tracking is lost a recovery procedure is required. Therefore PTAM implements the recovery method of William at al. (WILLIAMS; KLEIN; REID, 2007) which tries to recover the camera pose, comparing the current frame with all existing keyframes in the map, to find an approximate match. If this method find a pose estimate, the tracking procedure continues as normal.

Mapping Process

Mapping describes the process by which the map of keyframes and landmarks (3D map points) is built. In the beginning an initial map is built from the sequence of images, then this map is continually refined and expanded whenever new keyframes and landmarks are added by the tracking process. Visual SLAM using stereo and RGB-D camera can build an initial map simply from the first image. This is not possible for monocular SLAM because the landmark positions are not fully observable. Therefore, PTAM employs a separate initialization procedure using the five-point stereo algorithm of Nister et al. (NISTER, 2004) which simulates an stereo system and requires the user cooperation. The user must place the camera above the scene and select the first keyframe to be added to the map. Using the lowest image pyramid level of the selected keyframe, 1000 FAST corners are computed. Then, the user smoothly moves the camera to a new position and selects the second keyframe to also be added to the map.

Next, FAST corners are matched and an essential matrix encoding the relation between two viewpoints is computed using the five-point algorithm (NISTER, 2004) and RANSAC (FISCHLER; BOLLES, 1981). This essential matrix encodes the rotation matrix and translation vector for the second keyframe with respect to the first, and it is also used to compute the triangulation of the initial map. Later, the resulting map is refined through non-linear optimization called bundle adjustment (TRIGGS et al., 1999). Due to numerical considerations, the initial map obtained in the previous step has an arbitrary scale factor such as the average distance between camera and landmarks.

Once the initialization is finished, the initial map contains only two keyframes and few landmarks which describe a relatively small volume of space. Nevertheless, as the camera moves away from its initial pose, one needs to add new keyframes to allow the map to expand. Thus, a new keyframe is added whenever the following conditions are met: the tracking quality is good; the frame is separated at least 20 frames since last keyframe added; the camera center is a minimum distance away from the nearest keyframe already in the map. The latter addresses the problem of a stationary camera corrupting the map, ensuring that there is enough stereo baseline for new triangulation. Later, for each image pyramid level in the keyframe the most salient FAST points are selected by computing non-maximal suppression and using the threshold based on Shi-Tomasi scores (SHI; TOMASI, 1994). The salient FAST points which are close to existing observations are discarded and each remaining salient FAST point is a candidate to be a new map point. New map point requires depth information which is not available from a single keyframe. Therefore, in order to simulate a stereo system a second view already existing in the map is selected (the closest keyframe to the current one in terms of camera position). Next, correspondences between the two views are established and in the second view patches around FAST corner and near to epipolar lines are matched using SSD (sum of squared differences). This procedure is also known as search along epipolar line. Because the epipolar line is infinite, a **priori** hypothesis on the likely of the depth of the new candidate point is used to reduce the search space. This hypothesis depends on the distribution of the depths of the observed landmarks in the new keyframe. Later, each patch is searched only in its corresponding level of the image pyramid and if a match is found, the new point is triangulated and inserted into the map as a new landmark.

Finally, all landmarks and camera poses estimate are refined using an iterative minimization procedure known as full bundle adjustment or global bundle adjustment. Full bundle adjustment is very similar to the previous pose refinement problem expressed by Equation 2.17. But now the optimization is over all landmarks positions and all keyframes except the first, which is a fixed data. Then, given a set of N landmarks ${}^{w}p_1, .., {}^{w}p_N$; a set of M keyframes ${}^{w}K_1, .., {}^{w}K_M$ in the current map and the set of patches successfully matches S_j associated with each jth keyframe, for $j = 1, ..., M$. Full bundle adjustment iteratively adjusts the map, minimizing the objective function $\mathcal{G}$:

$$\left\{\hat{\xi}_1..\hat{\xi}_M\right\}, \{ {}^{w}\hat{p}_1.. {}^{w}\hat{p}_N\} = \underset{\{\xi_1..\xi_M\},\{ {}^{w}p_1.. {}^{w}p_N\}}{\arg\min} \sum_{j=1}^{M} \sum_{k \in S_j} \mathcal{G}\left(\frac{\mid {}^{j}e_k \mid}{{}^{j}\sigma_k}, \sigma^T\right) \tag{2.18}$$

Where $\hat{\xi}_1, ..., \hat{\xi}_M$ and ${}^{w}\hat{p}_1, ..., {}^{w}\hat{p}_N$ represent the camera poses and landmarks updated in the map; $\xi_1, ..., \xi_M$ are the camera poses of all keyframes in the map; ${}^{j}\sigma_k$ and ${}^{j}e_k$ are the assumed measurement noise and the error projection of the kth patch in the jth keyframe with pose ξ_j, respectively. Full bundle adjustment is solved using the algorithm of Levenberg-Marquardt (MORE, 1978). Nevertheless, this optimization may take a long time as the map size increases. Therefore when the camera is exploring, PTAM applies a bundle adjustment only with the pose of the most recent keyframe, its 4 closest keyframes and the position of all landmarks seen by them. This last procedure is known as local bundle adjustment and in essence just limits the variables being optimized.

As discussed earlier, PTAM is the most representative keyframe-based SLAM approach, which provides simple but effective methods for keyframe selection, patches matching, point triangulation, camera localization and relocalization after tracking failure. Unfortunately several factors limit its application, such as lack of loop closure, large-scale operation or even an incorrect initialization. Probably the biggest problem is the constant insertion of keyframes even if the camera is looking at the scene from different viewpoints, generating redundant information in the map and causing an excessive growth of the map size. In next Section, ORB-SLAM is presented. This method expands the versatility of PTAM to environments that are intractable for it. ORB-SLAM integrates a very efficient place recognition system in PTAM to perform relocalization and loop closing. ORB-SLAM also adds methods for an automatically initialization and for the removal of keyframes and landmarks in order to avoid unnecessary redundancy.

2.5.2 ORB-SLAM

ORB-SLAM proposed by Mur-Artal et al. (MUR-ARTAL; MONTIEL; TARDOS, 2015), is a keyframe-based monocular SLAM which operates in real time, in small and large, indoor and outdoor environments. ORB-SLAM is composed of three main tasks: tracking, mapping and loop closure, which run in parallel. ORB-SLAM also builds a reusable map features that can be used for recognition, relocalization and loop closure. In addition, it incorporates a pruning mechanism to reduce the redundancy of keyframes and landmarks in the map.

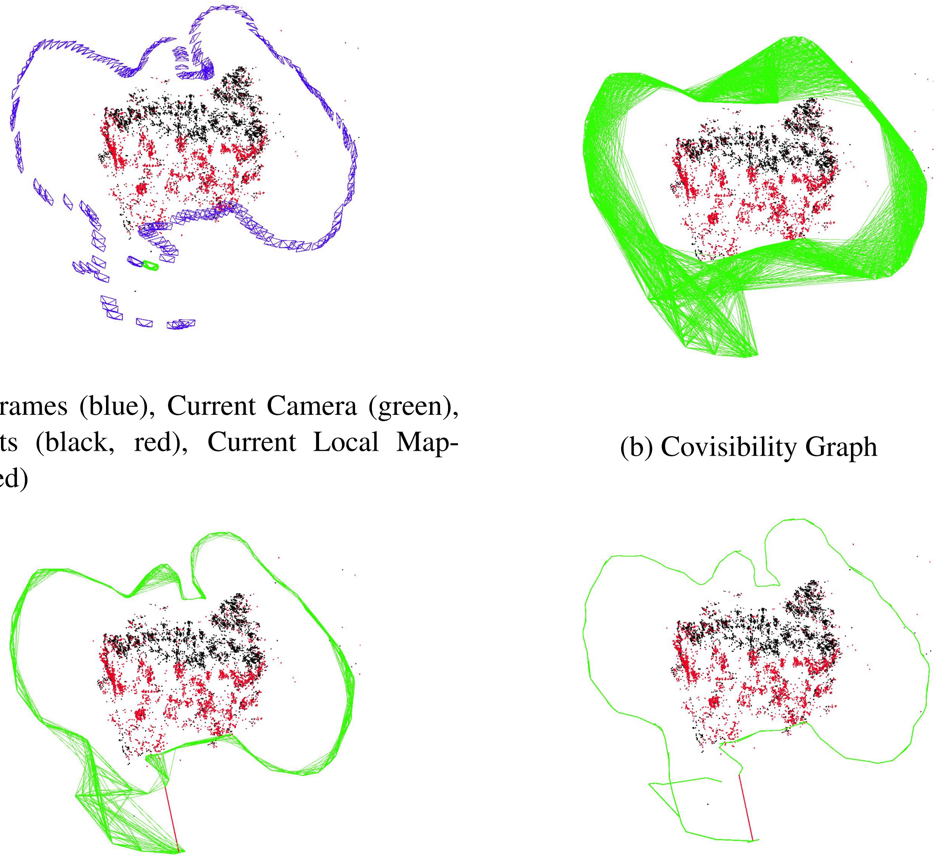
Map Representation

Just as PTAM, the map contains a collection of map points and keyframes. Additionally, ORB-SLAM builds a covisibility graph (RUBLEE et al., 2011) that allows to perform real-time operation in large environments; and to deal with the loop closure problem, ORB-SLAM builds an essential graph which is constructed from the graph of covisibility.

Each map point ${}^{w}p_i$ represents a FAST feature and stores its location in the world coordinates expressed in homogeneous coordinates ${}^{w}X_i \in \mathbb{R}^4$; an ORB descriptor D_i that provides a good invariant to changes in viewpoint and illumination (RUBLEE et al., 2011); a viewing direction $n_i \in \mathbb{R}^4$ and the maximum d_{max} and minimum d_{min} distance at which the map point can be observed. In the case of keyframes, each ${}^{w}K_i$ stores all the FAST features and ORB descriptors extracted in the frame associated or not to a map point; the camera pose ${}^{w}T_i$; a image pyramid to deal with the scale and the intrinsic parameters of the camera.

On the other hand, a covisibility graph is defined as an undirected weighted graph where each node is a keyframe. An edge between two keyframes exists if they see at least θ common map points. This covisibility graph allows to maintain the tracking and mapping focused in a local covisible area, independent of global map size. The covisibility graph also allows the construction of the essential graph containing a subset of edges with high covisibility, the loop closure edges and a spanning tree that provides a connected sub-graph with a minimum number of edges. These graphs and the structure of the map are shown in Figure 2.6

Figure 2.6: Reconstruction and graphs in the sequence *fr3 long office household* from the TUM RGB-D Benchmark (STURM et al., 2012). Image extracted from (MUR-ARTAL; MONTIEL; TARDOS, 2015).



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

(b) Covisibility Graph

(c) Essential Graph

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

Place Recognition

ORB-SLAM also integrates a place recognizer to perform loop detection and relocalization. This place recognizer is based on bags of binary words DBoW2 of Galvez Lopez et al. (GALVEZ-LOPEZ; TARDOS, 2012). So, in an offline step, using DBoW2 a visual vocabulary is built on a large set of ORB descriptors. These descriptors are extracted from a training image dataset that is collected in both indoor and outdoor areas. If the images in the training dataset are general enough, the same visual vocabulary can be used for different environments getting a good performance (MUR-ARTAL; TARDOS, 2014).

The visual vocabulary is composed of words which are just a discretization of the descriptor space. Then, each ORB descriptor is associated to a word, and for each word a list of keyframes where it has been seen is stored. This structure in the vocabulary allows queries to the recognition database can be performed very efficiently by just checking common visual words. Then, when the relocalization or the loop detector query the recognition database, a similarity score is computed between all keyframes that share visual words with the query image. Because there exists visual overlap between keyframes, there will not exist a unique keyframe with high score, rather there will be several keyframes with high scores. In addition, the bags of words representation is also used for feature matching.

Automatic Map Initialization

The purpose of the map initialization is to calculate the relative pose between two frames and create an initial set of map points. Therefore, to perform an initialization a good two-view configuration must be selected without human intervention. Then, ORB-SLAM proposed an automatic map initialization method which works independent of the scene (planar or non-planar).

Initially, the method takes a reference frame K_r and extracts a set of ORB features $\mathcal{X}$. Then, another frame K_c known as current frame is taken and the matches $\mathcal{Y}$ are searched in the the reference frame. If not enough matches are found, the reference frame is restarted. Otherwise, two geometrical models are computed: a homography model H_{cr} assuming a planar scene and a fundamental matrix model F_{cr} assuming a non-planar scene. Then a score S_M is computed for each model M (H for the homography, F for the fundamental matrix). This score is computed in each iteration of RANSAC using:

$$S_M = \sum_i \left[\rho\left(d\left(\mathcal{X}_i, \mathcal{Y}_i\right), M\right) + \rho\left(d\left(\mathcal{Y}_i, \mathcal{X}_i\right), M\right)\right] \tag{2.19}$$

Where ρ and d are the weight and the symmetric transfer error function, respectively (HARTLEY; ZISSERMAN, 2003). If no model could be found, the ORB features are extracted again and the process is repeated. Otherwise, it is necessary to select one of the two models. Then, a weight R is calculated using the previously scores, as shown:

$$R = \frac{S_H}{S_H + S_F} \tag{2.20}$$

If $R > 0.45$, the homography model is selected, otherwise, the fundamental matrix model. Once a model is selected, the relative pose is retrieved. In the case of the homography model, 8 relative poses are retrieved using the method of Faugeras et. al

(HARTLEY; ZISSERMAN, 2003). In the case of the fundamental matrix model, first it is converted in an essential matrix using the camera calibration matrix and then 4 relative poses are retrieved using the singular valued decomposition (SVD) (FAUGERAS; LUSTMAN, 1988). Later, the relative pose with most map points seen in front of both cameras and with low reprojection error is selected. Finally, the bundle adjustment is performed.

Tracking Process

Once an initial map has been created, the tracking process estimates the camera pose with every current frame incoming from the camera. First, the image pyramid is computed with FAST features. However, in order to ensure an homogeneous distribution each pyramid level is divided in a grid, trying to extract the same number of FAST features for each cell. After, the orientation and the ORB descriptor are computed on each feature. These descriptors are used in all feature matching instead of the search by patch correlation.

Then, a match in the current frame is searched for each FAST feature observed and associated a map point in the last frame. Later, with the set of correspondences previously calculated a **priori** camera pose is estimated using a constant velocity motion model. And it is optimized using the Levenberg-Marquardt algorithm with the Huber cost function (HUBER, 2011) that minimizes the projection error. Thus, with a **priori** camera pose and an initial set of correspondences between FAST features, the map is projected into the current frame for searching more correspondences.

However, to deal with the complexity in large maps, ORB-SLAM only projects a local map instead of projecting all map points. The local map is the union of $\mathcal{K}_1$ and $\mathcal{K}_2$. Where $\mathcal{K}_1$ is the set of keyframes which share map points with the current frame. While $\mathcal{K}_2$ is the set with a limited number of neighbouring keyframes in the covisibility graph to each keyframe in $\mathcal{K}_1$. Furthermore, the local map also has a reference keyframe ${}^{w}K_{ref}$ that is used in the mapping process. This keyframe belongs to set $\mathcal{K}_1$ and shares most map points with the current frame.

In this way, for each map point in the local map is calculated its projection x, the angle α between the current viewing ray v and the map point mean viewing direction n and its distance to the center of the camera d. If the map point projection x is outside the image boundaries or $v \cdot n < \cos(60°)$ or d is not within of $[d_{min}, d_{max}]$, the map point is not considered for the tracking procedure. Then, using the map point descriptor, it is associated with the still unmatched FAST features close to x in the current frame. Later, the camera pose is optimized with all the map points found in the current frame.

If the tracking was successful, it has to be checked if the current frame can be inserted as a new keyframe on the map. So, to be a new keyframe, all the following conditions must be checked: the current frame tracks at least 50 map points, this condition ensures a good tracking; the current frame tracks less than 90% map points contained in K_{ref}, this condition ensures the visual change; more than 20 frames have passed from last keyframe insertion, this condition inserts keyframes as soon as possible to make tracking more precise in front of challenging camera movements, such as rotations.

Otherwise, if the tracking is lost. First, the current frame is converted on its bag of word representation and then in order to obtain the relocalization candidates a query is made to the recognition database (place recognition). For each keyframe candidate obtained by the query, the correspondences are computed between those FAST features in the keyframe candidate associated to a map point and the FAST features extracted in the current frame. Later, PnP algorithm (LEPETIT; MORENO-NOGUER; FUA, 2009) with RANSAC iterations are performed trying to find the camera pose. The RANSAC iterations are performed until finding a camera pose with enough inliers. Finally, the camera pose is optimized and the tracking process continues.

Local Mapping Process

The mapping procedure processes every new keyframe ${}^{w}K_i$ accepted in the tracking procedure. Initially, the new keyframe is inserted in the map and its bags of words representation is computed. So, in the covisibility graph a new node for ${}^{w}K_i$ is created and new edges from the shared map points with other keyframes are also created. Furthermore in the spanning tree, the node that represents ${}^{w}K_i$ is connected to the keyframe with most map points in common. Before the creation of the new points, in current map a point culling policy (MUR-ARTAL; MONTIEL; TARDOS, 2015) is applied. This culling procedure is performed in order to retain only high quality map points. Furthermore it procedure ensures that the map points are trackable and not wrongly triangulated. To retained a map point in the map, the tracking must find it in more than the 25% of the keyframes. And if more than one keyframe has inserted since the map point creation, it must be observed from at least three keyframes. Otherwise the map point is discarded.

To create new map points. First, $\mathcal{N}_i$ a set of neighbouring keyframes of the current keyframe is extracted from the covisibility graph. Subsequently, for each unmatched feature in the current keyframe a match is searched in the set $\mathcal{N}_i$. And from each match found, a new point is created by the triangulation. However, only those map points that have parallax, scale consistency and positive depth in both keyframes are accepted on the map. Then, once the new map points have been created, each one of them is projected in the keyframes of $\mathcal{N}_i$ and its correspondences are searched, similarly to the tracking process. Later, the current keyframe ${}^{w}K_i$, the keyframes in the set $\mathcal{N}_i$ and the map points seen by those keyframes are optimized by local bundle adjustment. Finally for each map point, the normal n_i, the scale invariance distances d_i^{min} and d_i^{max} are recomputed.

Additionally, in order to maintain a compact reconstruction, ORB-SLAM attempts to detect redundant keyframes and delete them. In $\mathcal{N}_i$, those keyframes whose 90% of their map points have been seen in at least other three keyframes in the same scale are discarded. This process is beneficial for the complexity time of global bundle adjustment which depends of the number of keyframes.

Loop Closure Process

The loop closing procedure takes the last keyframe ${}^{w}K_i$ processed by the mapping procedure to attempts to detect and close loops. Initially s_i is computed, which represents the similarity between the bag of words vector of keyframe ${}^{w}K_i$ and all its neighbours in the covisibility graph. Then, the lowest score is retained in s_{min} and queries to the recognition database (place recognition) are performed. In the recognition database, those keyframes that are directly connected to ${}^{w}K_i$ or whose scores are lower than s_{min} are discarded as loop candidate. To accept a keyframe as a loop candidate, the loop closure procedure must detect consecutively three loop candidates that share at least one keyframe in the covisibility graph.

The correspondences between features associated to map points in the current keyframe and the loop candidate keyframes are computed. Then, RANSAC iterations (HORN, 1987) are performed between each loop candidate keyframe ${}^{w}K_l$ and the current keyframe ${}^{w}K_i$, trying to find the similarity transformation S_{il} with seven degrees of freedom (3 rotations, 3 translations and 1 scale factor) which informs the error accumulated in the loop. If a similarity transformation with enough inliers is found, it is optimized and a search of more correspondences between features associated to map points are performed.

Finally, if S_{il} is supported by enough inliers the loop candidate keyframe is accepted.

Once the loop is closed, it is necessary to update the map. Then, duplicate map points are replaced by the map point that was seen more times; the new edges that connects the loop closure are inserted in the covisibility graph; the current keyframe pose ${}^{w}T_i$ is corrected with the similarity transformation S_{il} and the graph is aligned by propagation of this correction over all neighbours of ${}^{w}K_i$. Later, the map points seen in the loop keyframe ${}^{w}K_l$ and its neighbours are projected into ${}^{w}K_i$ and its neighbours. Next, the matches are searched as in the tracking procedure and those that were inliers in the computation of S_{il} are fused. In the covisibility graph, new edges are created between the loop keyframe and the keyframes involved in the fusion. To correct the loop error, a global bundle adjustment is performed on the essential graph (STRASDAT; MONTIEL; DAVISON, 2010). Finally, each map point is transformed according to the correction of one of the keyframes that observes it.

3 MAP POINT OPTIMIZATION IN KEYFRAME-BASED SLAM

In the following, we introduce our approach that uses a covisibility graph (STUMM; MEI; LACROIX, 2013) and an algorithm of information fusion (PIZZOLI; FORSTER; SCARAMUZZA, 2014) in order to maintain a refined map in keyframe-based monocular SLAM process.

3.1 Map Representation

Our map is represented as a set of map points, keyframes and an undirected weighted graph known as covisibility graph, where each node is a keyframe ${}^{w}K_i$ and an edge between two keyframes exists if they share at least θ common map points. The map point ${}^{w}p_j$ corresponds to an ORB feature, which represents a textured planar patch in the world whose position has been triangulated from different views. Each map point stores its 3D position in the world coordinate system ${}^{w}X_j \in \mathbb{R}^3$ and the viewing direction $n_j \in \mathbb{R}^3$; the maximum d_{max} and minimum d_{min} distances at which it can be observed, according to the scale invariance limits of the ORB features; and an ORB descriptor D_j. On the other hand, each keyframe stores all ORB features extracted in the frame; the camera pose ${}^{w}T_i \in SE(3)$, which is a rigid body transformation that maps the map points from the world to the camera coordinates system and the intrinsic parameters of the camera P $\in \mathbb{R}^{3\times3}$.

3.2 Probabilistic Depth Sensor

To create a map in the SLAM process, the robot collects information and builds a representation of the environment where it is located. In keyframe-based SLAM, cameras are commonly used to get such information by performing depth measurements through the captured images. These measurements are always subject to errors called noise and there may also be seemingly random measurements that are caused by photometric inconsistency. In this way, when the task is to create an accurate map of the environment from such noisy measurements, a probabilistic approach is necessary. Thus, we model each depth measurement d obtained by the sensor as a distribution that mixes a good measurement model with the bad one as in the work performed by Forster et al. (FORSTER; PIZ-

ZOLI; SCARAMUZZA, 2014). The good measurement is normally distributed around the true depth $\hat{d}$ whereas the bad one is uniformly distributed in all possible depth locations in the interval $[d_{min}, d_{max}]$ which is known to contain the true depth. In this way, our mixture model distribution is defined as:

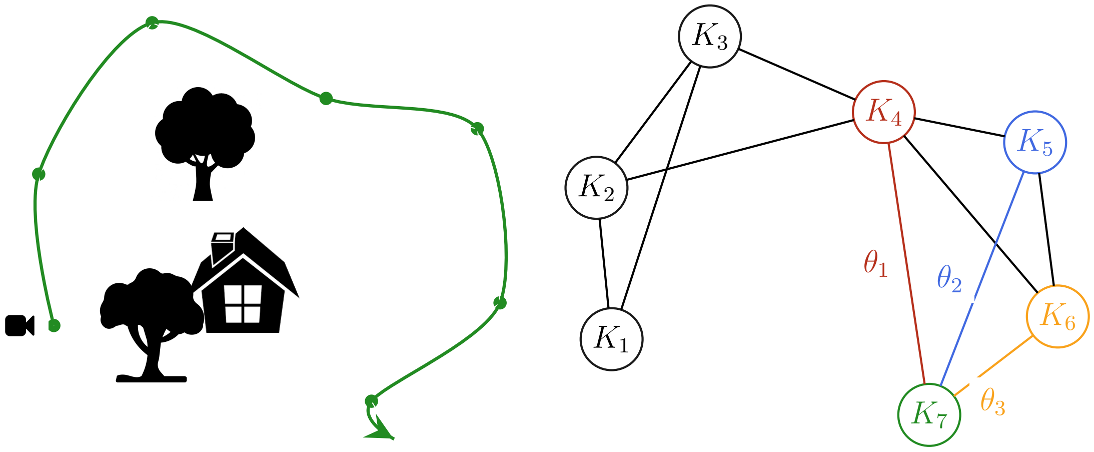
$$p\left(d \mid \hat{d}, \rho\right) = \rho \underbrace{\mathcal{N}\left(d \mid \hat{d}, \tau^2\right)}_{\text{Normal Distribution}} + (1-\rho)\underbrace{U\left(d \mid d_{min}, d_{max}\right)}_{\text{Uniform Distribution}}, \tag{3.1}$$

where d is the depth measurement, the parameter ρ measure the purity of the measurement (inlier probability) and τ^2 is the variance of a good measurement, which can be computed geometrically by assuming a fixed variance of one pixel in the image plane defined by the relative position of the cameras that produced the measurement (PIZZOLI; FORSTER; SCARAMUZZA, 2014).

3.3 Depth Bayesian Inference

The uncertainties in sensors arise not only from the imprecision and noise in the measurements, but are also caused by the ambiguities and inconsistencies present in the environment, and by the inability to distinguish them. Information fusion algorithms are able to exploit redundant data to alleviate such effects. Briefly, we can define the information fusion as the process of integrating multiple information sources to obtain improved and useful information as accurately as possible (HORNUNG et al., 2013). The creation of new map points is necessary to represent the new information every time that a new keyframe is selected. In this sense, we first update the covisibility graph by adding a new node for the new keyframe ${}^{w}K_i$ and creating the covisibility edges as shown in Figure 3.1b, where the node corresponding to keyframe ${}^{w}K_7$ is added to the covisibility graph. Due to these updates in the covisibility graph, a vicinity is generated for this new keyframe which we use to compute new depth measurements. Later, to infer a single depth estimation with the higher possible accuracy we collect and combine these measurements using information fusion, i.e, using Bayesian inference (BISHOP, 2006).

Figure 3.1: Covisibility graph (STUMM; MEI; LACROIX, 2013) with nodes representing keyframes and edges representing their covisibility. Image generated by the author.



(a)

(b)

Hence, given a new keyframe ${}^{w}K_n$ and a covisibility graph G, for each feature found in the new keyframe there is a set of noisy depth measurement denoted by d_i for $i = 1, ..., k$ obtained from the triangulation of the new keyframe with k neighbours of the covisibility graph. Furthermore, in order to simplify the problem and allow the use of Bayesian estimation we assume that all measurements $d_1, ..., d_k$ are independent. Therefore the depth posterior is approximated by the product of the marginal probabilities:

$$p\left(\hat{d}, \rho \mid d_1, ..., d_k\right) \propto p\left(\hat{d}, \rho\right) \prod_k p\left(d_k \mid \hat{d}, \rho\right), \tag{3.2}$$

with $p(\hat{d}, \rho)$ being a prior on the true depth and the inlier probability. Vogiatzis et al. (VOGIATZIS; HERNANDEZ, 2011) prove that the posterior can be approximated by the product of $Gaussian$ and $Beta$ distributions (the first for the true depth and the second for the inlier probability), to minimize the Kullback-Leibler divergence from the true posterior:

$$p\left(\hat{d}, \rho \mid d_1, ..., d_k\right) \propto \mathcal{N}\left(\hat{d} \mid \mu_k, \sigma_k^2\right) Beta\left(\rho \mid a_k, b_k\right), \tag{3.3}$$

which can be parametrized with four parameters, where the first two parameters, a_k and b_k control the Beta distribution. The other two, μ_k and σ_k^2 represent the mean and the variance of the Gaussian distribution. It leads to:

$$q\left(\hat{d}, \rho \mid a_k, b_k, \mu_k, \sigma_k^2\right) \propto \mathcal{N}\left(\hat{d} \mid \mu_k, \sigma_k^2\right) Beta\left(\rho \mid a_k, b_k\right) \tag{3.4}$$

Once defined the posterior, the Bayesian estimation allows us to integrate new measurements in order to update the posterior to describe a new knowledge state. It is given by:

$$\underbrace{q\left(\hat{d},\rho \mid a_k,b_k,\mu_k,\sigma_k^2\right)}_{\text{Posterior}} = \underbrace{q\left(\hat{d},\rho \mid a_{k-1},b_{k-1},\mu_{k-1},\sigma_{k-1}^2\right)}_{\text{Prior}}\underbrace{p\left(d_k \mid \hat{d},\rho\right)}_{\text{Likelihood}}, \tag{3.5}$$

using Equation 3.4 and 3.1 in Equation 3.5, moreover the definition of Gaussian and Beta distributions, as well as the properties of the gamma function (JAYNES, 2003), we obtain:

$$\begin{aligned}
&= q\left(\hat{d},\rho \mid a_{k-1},b_{k-1},\mu_{k-1},\sigma_{k-1}^2\right) p\left(d_k \mid \hat{d},\rho\right)\\
&= \left[\mathcal{N}\left(\hat{d} \mid \mu_{k-1},\sigma_{k-1}^2\right) Beta\left(\rho \mid a_{k-1},b_{k-1}\right)\right]\\
&\quad \left[\rho\mathcal{N}\left(d_k \mid \hat{d},\tau^2\right) + (1-\rho)\,U\left(d_k \mid d^{min},d^{max}\right)\right]\\
&= \rho\mathcal{N}\left(d_k \mid \hat{d},\tau_k^2\right)\mathcal{N}\left(\hat{d} \mid \mu_{k-1},\sigma_{k-1}^2\right) Beta\left(\rho \mid a_{k-1},b_{k-1}\right)\\
&\quad + (1-\rho)\,U\left(d_k\right)\mathcal{N}\left(\hat{d} \mid \mu_{k-1},\sigma_{k-1}^2\right) Beta\left(\rho \mid a_{k-1},b_{k-1}\right)\\
&= \mathcal{N}\left(d_k \mid \hat{d},\tau_k^2\right)\mathcal{N}\left(\hat{d} \mid \mu_{k-1},\sigma_{k-1}^2\right) Beta\left(\rho \mid a_{k-1},b_{k-1}\right)\rho\\
&\quad + U\left(d_k\right)\mathcal{N}\left(\hat{d} \mid \mu_{k-1},\sigma_{k-1}^2\right) Beta\left(\rho \mid a_{k-1},b_{k-1}\right)(1-\rho)\\
&= \frac{a_{k-1}}{a_{k-1}+b_{k-1}}\mathcal{N}\left(d_k \mid \hat{d},\tau_k^2\right)\mathcal{N}\left(\hat{d} \mid \mu_{k-1},\sigma_{k-1}^2\right) Beta\left(\rho \mid a_{k-1}+1,b_{k-1}\right)\\
&\quad + \frac{b_{k-1}}{a_{k-1}+b_{k-1}}U\left(d_k\right)\mathcal{N}\left(\hat{d} \mid \mu_{k-1},\sigma_{k-1}^2\right) Beta\left(\rho \mid a_{k-1},b_{k-1}+1\right)\\
&= \frac{a_{k-1}}{a_{k-1}+b_{k-1}}\mathcal{N}\left(d_k \mid \mu_{k-1},\tau_k^2+\sigma_{k-1}^2\right)\mathcal{N}\left(\hat{d} \mid m,s^2\right) Beta\left(\rho \mid a_{k-1}+1,b_{k-1}\right)\\
&\quad + \frac{b_{k-1}}{a_{k-1}+b_{k-1}}U\left(d_k\right)\mathcal{N}\left(\hat{d} \mid \mu_{k-1},\sigma_{k-1}^2\right) Beta\left(\rho \mid a_{k-1},b_{k-1}+1\right)\\
&= \frac{a_{k-1}}{a_{k-1}+b_{k-1}}\mathcal{N}\left(d_k \mid \mu_{k-1},\tau_k^2+\sigma_{k-1}^2\right) q\left(\hat{d},\rho \mid a_{k-1}+1,b_{k-1},m,s^2\right)\\
&\quad + \frac{b_{k-1}}{a_{k-1}+b_{k-1}}U\left(d_k\right) q\left(\hat{d},\rho \mid a_{k-1},b_{k-1}+1,\mu_{k-1},\sigma_{k-1}^2\right)\\
&= \mathrm{c}_1 q\left(\hat{d},\rho \mid a_{k-1}+1,b_{k-1},m,s^2\right) + \mathrm{c}_2 q\left(\hat{d},\rho \mid a_{k-1},b_{k-1}+1,\mu_{k-1},\sigma_{k-1}^2\right)
\end{aligned} \tag{3.6}$$

with:

$$\mathrm{c}_1 = \frac{a_{k-1}}{a_{k-1}+b_{k-1}}\mathcal{N}(d_k \mid \mu_{k-1},\tau_k^2+\sigma_{k-1}^2) \tag{3.7}$$

$$\mathrm{c}_2 = \frac{b_{k-1}}{a_{k-1}+b_{k-1}}U(d_k \mid d_{min},d_{max}) \tag{3.8}$$

where c_1 and c_2 can be calculated from the information known in a previous step. Furthermore m and s^2 represent the new mean and variance obtained from the product of two Gaussian functions and are given by:

$$m = \frac{\sigma_{k-1}^2 d_k + \tau_k^2 \mu_{k-1}}{\tau_k^2 + \sigma_{k-1}^2}, \tag{3.9}$$

$$s^2 = \frac{\tau_k^2 \sigma_{k-1}^2}{\tau_k^2 + \sigma_{k-1}^2}, \tag{3.10}$$

In Equation 3.6, we can observe that a_k and b_k can be thought of as probabilistic counters of how many inlier and outlier measurements have occurred during the lifetime of the map point. Finally, using Equation 3.6 and matching the first and second moments of the Gaussian and Beta distributions (JAYNES, 2003), we can obtain the new posterior parameters a_k, b_k, μ_k, σ_k^2, from the old parameters a_{k-1}, b_{k-1}, μ_{k-1}, σ_{k-1}^2, and the new measurement d_k. Therefore, the first and second moments with regard to $\hat{d}$ is given by:

$$\mu_k = \mathrm{c}_1 m + \mathrm{c}_2 \mu_{k-1} \tag{3.11}$$

$$\sigma_k^2 = \mathrm{c}_1 \left(s^2 + m^2\right) + \mathrm{c}_2 \left(\sigma_{k-1}^2 + \mu_{k-1}^2\right) - \mu_k^2 \tag{3.12}$$

In the same way the first and second moments with regard to ρ is given by:

$$\frac{a_k}{a_k + b_k} = \mathrm{c}_1 \frac{a_{k-1} + 1}{a_{k-1} + b_{k-1} + 1} + \mathrm{c}_2 \frac{a_{k-1}}{a_{k-1} + b_{k-1} + 1} \tag{3.13}$$

$$\begin{aligned} \frac{a_k \left(a_k + 1\right)}{\left(a_k + b_k\right)\left(a_k + b_k + 1\right)} &= \mathrm{c}_1 \frac{\left(a_{k-1} + 1\right)\left(a_{k-1} + 2\right)}{\left(a_{k-1} + b_{k-1} + 1\right)\left(a_{k-1} + b_{k-1} + 2\right)} \\ &\quad + \mathrm{c}_2 \frac{a_{k-1}\left(a_{k-1} + 1\right)}{\left(a_{k-1} + b_{k-1} + 1\right)\left(a_{k-1} + b_{k-1} + 2\right)} \end{aligned} \tag{3.14}$$

Finally, solving the system of equations formed by Equation 3.13 and 3.14, we obtained the new posterior parameters a_k and b_k:

$$a_k = \frac{\mathrm{v}_1 \left(\mathrm{v}_2 - \mathrm{v}_1\right)}{\mathrm{v}_1^2 - \mathrm{v}_2} \tag{3.15}$$

$$b_k = a_k \frac{(1 - \mathrm{v}_1)}{\mathrm{v}_1} \tag{3.16}$$

with:

$$\mathrm{v}_1 \quad = \quad \mathrm{c}_1 \frac{a_{k-1} + 1}{a_{k-1} + b_{k-1} + 1} + \mathrm{c}_2 \frac{a_{k-1}}{a_{k-1} + b_{k-1} + 1} \tag{3.17}$$

$$\begin{aligned} \mathrm{v}_2 = \mathrm{c}_1 & \frac{(a_{k-1} + 1)(a_{k-1} + 2)}{(a_{k-1} + b_{k-1} + 1)(a_{k-1} + b_{k-1} + 2)} \\ & + \mathrm{c}_2 \frac{a_{k-1}(a_{k-1} + 1)}{(a_{k-1} + b_{k-1} + 1)(a_{k-1} + b_{k-1} + 2)} \end{aligned} \tag{3.18}$$

3.4 Map Optimization

To maintain a refined map, we need a method to detect and reject outliers and obsolete information on all observations made on the new keyframe. Firstly, we define as obsolete information those map points that through the time do not contribute to the SLAM process. In this way, we have considered obsolete information those map points which have not been observed in at least three keyframes because they can be information not trusted due they are not continuously observables. Furthermore, this ensure that more than one measurement has been used in estimating the depth posterior distribution. Secondly, in order to identify outliers we use only the information contained in the depth posterior distribution. Our approach takes into account the amount of inlier measurements that have occurred and the amount of information gained. In this sense, given the depth posterior for all observations in the new keyframe we assign a state for each of them using the conditional defined as:

$$S(q_k) = \begin{cases} \text{Converged,} & \text{if } \frac{a_k}{a_k + b_k} > \eta^i \text{ and } \sigma^2 < \sigma_*^2 \\ \text{Diverged,} & \text{if } \frac{a_k - 1}{a_k + b_k - 2} < \eta^o \\ \text{Update,} & \text{otherwise,} \end{cases} \tag{3.19}$$

where q_k represent the depth posterior distribution defined in Equation 3.4, σ_* is the gained information threshold, whereas η^i and η^o are the inlier and outlier threshold, respectively. In Equation 3.19 we can observe that there are three possible outcomes. First, if the mode of the beta distribution is less than η^o then we conclude that the depth estimation has failed to converge due to unreliable measurements. Therefore this map point is removed from the map. Second, if the mean of the beta distribution is bigger than η^i and the variance of the normal distribution is less than σ_*^2 then we assume that the depth has converged to a good estimate and this map point remains stable, otherwise it waits for new measurements to be integrated.

Note that the mean of the beta distribution is used to evaluate convergence while the mode of the same distribution is used to evaluate divergence. However, the mean could also be used to evaluate the divergence but this value tends to extremes (0 or 1) rapidly as the parameters of the posterior are updated through Equation 3.5. Due to this behaviour we choose to use the mode, allowing to maintain the map points for more time waiting for new measurements to be integrated. Finally, to deal with repetitive information, the map point is searched in the neighbouring keyframes. If a match is found, it is added to the map using the method explained in the previous section and once again the criteria of (3.19) is applied.

4 EXPERIMENTS

The proposed method is evaluated using the TUM RGB-D benchmark (STURM et al., 2012). The benchmark contains 39 sequences that was captured by a Microsoft Kinect sensor. Each sequence contains both the color and depth images in full sensor resolution (640x480) at video frame rate (30 Hz), and an accurate ground truth for camera motion that was provided by a motion capture system with eight high-speed tracking cameras (100 Hz). We have selected 11 sequences, which are also used in other works (MUR-ARTAL; MONTIEL; TARDOS, 2015), (ENGEL; SCHOPS; CREMERS, 2014). For instance, some samples of the processed sequences can be seen in Figures 4.1, 4.2 and 4.3. Our experiments are performed on a desktop computer with Ubuntu 14.04, equipped with Intel Core 2 Quad processor and 4GB of RAM memory. The keyframe-based monocular SLAM system and our method are implemented in C++. The values for the parameters of our method are established as in the work of Vogiatzis et al. (VOGIATZIS; HERNANDEZ, 2011):

Table 4.1: The values for the parameters of our method.

Parameter	value
$\boldsymbol{a_0}$	10
$\boldsymbol{b_0}$	10
$\boldsymbol{\mu_0}$	$\bar{d}$
$\boldsymbol{\sigma_0}$	$(d_{max} - d_{min})$
$\boldsymbol{\eta^i}$	0.70
$\boldsymbol{\eta^o}$	0.05
$\boldsymbol{\sigma_*^2}$	$\frac{(d_{max}-d_{min})}{1000}$

In the next sections we integrate quantitative results to evaluate the resulting map; and in order to allow a quantitative comparison between trajectories obtained and ground truths, we computed two error metrics as proposed in (STURM et al., 2012): the relative pose error and the absolute keyframe trajectory error. Finally to compare our results, we executed the original ORB-SLAM system over the same sequences.

4.1 Removing outliers

To evaluate the resulting map, we used a quantitative metric such as the map size. Table 4.2 provides a summary of the ORB-SLAM performance with and without our proposed method on the 11 sequences. This table shows the number of keyframes and map points, the relative and the absolute error with respect to the trajectory which will be discussed later. In Table 4.2 we can observe the comparison of the resulting maps obtained by ORB SLAM with and without our proposed method, where we can see that in all sequences our approach manages to reduce a percentage of the amount estimated by the original ORB SLAM system.

Additionally, we performed 10 separate experiments for *fr3_long_office*, *fr2_desk* and *fr2_desk_person* sequences. In Figure 4.1, we can observe the first sequence where the camera moves along a large round through a household and office scene with textures and structures. Furthermore, the end of the trajectory overlaps with the beginning so that there is a large loop closure. In the second sequence shown in Figure 4.2, the camera moves around two tables that includes a typical office scene with two desks, a computer monitor, keyboard, phone, chairs, etc. Finally, the third sequence shown in Figure 4.3 is similar to the second, where additionally a person is sitting at a desk, moves and interacts with some of the objects (screen, phone, etc) during the recording. This sequence is intended to verify the robustness of a SLAM system against dynamic objects and people, but it can also be used for differencing maps and finding changes in the scene. In this way, Table 4.3, 4.4 and 4.5 show the results of experiments performed in these environments.

Table 4.2: Quantitative evaluations for 11 sequences from the TUM benchmark (STURM et al., 2012). From left to right, the columns show the dataset name, their length and length of filming. The number of keyframes, map points and their percentages of reduction, respectively. The root mean square error (RMSE) of absolute keyframe trajectory (ATE) and the relative pose (RPE) in terms of translation.

	Dataset	Duration [s]	Length [m]	Keyframes			Points			ATE[m]		RPE[m]	
				ORB	Ours	%	ORB	Ours	%	ORB	**Ours**	ORB	**Ours**
1	*fr1_desk*	23.40	9.263	63	60	**4.76**	3236	2785	**13.93**	0.0144	**0.0141**	0.0338	**0.0243**
2	*fr1_xyz*	30.09	7.112	31	25	**19.35**	1740	1408	**19.08**	0.0092	**0.0083**	**0.0402**	0.1257
3	*fr3_str_tex_far*	31.55	5.884	25	23	**8.00**	1911	1698	**11.14**	0.0088	**0.0076**	0.0972	**0.0702**
4	*fr3_walk_halfsph*	35.81	7.686	45	36	**20.00**	1433	1035	**27.77**	0.0164	**0.0159**	0.2690	**0.1783**
5	*fr3_str_tex_near*	36.91	5.050	49	43	**12.24**	3188	2969	**6.86**	0.0114	**0.0110**	**0.0317**	0.0401
6	*fr3_sit_halfsph*	37.15	6.503	76	69	**9.21**	2570	2096	**18.44**	0.2342	**0.0100**	0.2349	**0.1129**
7	*fr3_nstr_tex_near*	56.48	13.456	67	62	**7.46**	4542	4032	**11.22**	0.0138	**0.0132**	**0.0607**	0.0854
8	*fr3_long_office*	87.09	21.455	198	170	**14.14**	10175	8250	**18.91**	**0.0102**	0.0142	**0.1569**	0.1580
9	*fr2_desk*	99.36	18.880	177	149	**15.81**	7288	5914	**18.85**	**0.0083**	0.0170	0.0931	**0.0889**
10	*fr2_desk_person*	142.08	17.044	119	106	**10.92**	4565	3863	**15.37**	0.0096	**0.0080**	**0.0995**	0.1140
11	*fr2_xyz*	122.74	7.029	37	29	**21.62**	1582	1355	**14.34**	0.0025	**0.0024**	**0.0174**	0.0182

Table 4.3 and 4.4 are the results in static environments, where we can observe that the number of map points decrease because possibly enough of these map points are outliers, repetitive or obsolete. Furthermore, the number of keyframes also decrease due to the fact that the ORB-SLAM system has a method which attempts to detect redundant keyframes and delete them. This method is responsible for removing all keyframes whose 90% of their observations (map points) have been seen in at least three other keyframes in the map, allowing the system a long life operation in the same environment and benefiting bundle adjustment complexity. However, the real world is a dynamic environment. Thus, Table 4.5 shows the experiments performed for a dynamic environment, where we can observe that the results are still maintained, i.e., the number of map points and keyframes obtained with our approach is lower than that obtained with the ORB-SLAM system. This happens because information fusion methods manage to deal with frequent changes. In this sense, we also have to analyse how the decrease in the size of the map affects the accuracy of the estimated trajectory. Therefore, we present other measurement metrics.

Aditionally, Figure 4.4a and 4.4b show the resulting map for the *fr3_long_office* sequence whit ORB-SLAM and our approach respectively. For other hand, Figure 4.4c and 4.4d show the octree discretization of both approaches. In the same way Figure 4.4e and 4.4f Figure 4.4g and 4.4h show the resulting map for the *fr2_desk* sequence.

Figure 4.1: Images of the sequence *fr3_long_office* taken at different camera poses. Image generated by the author.

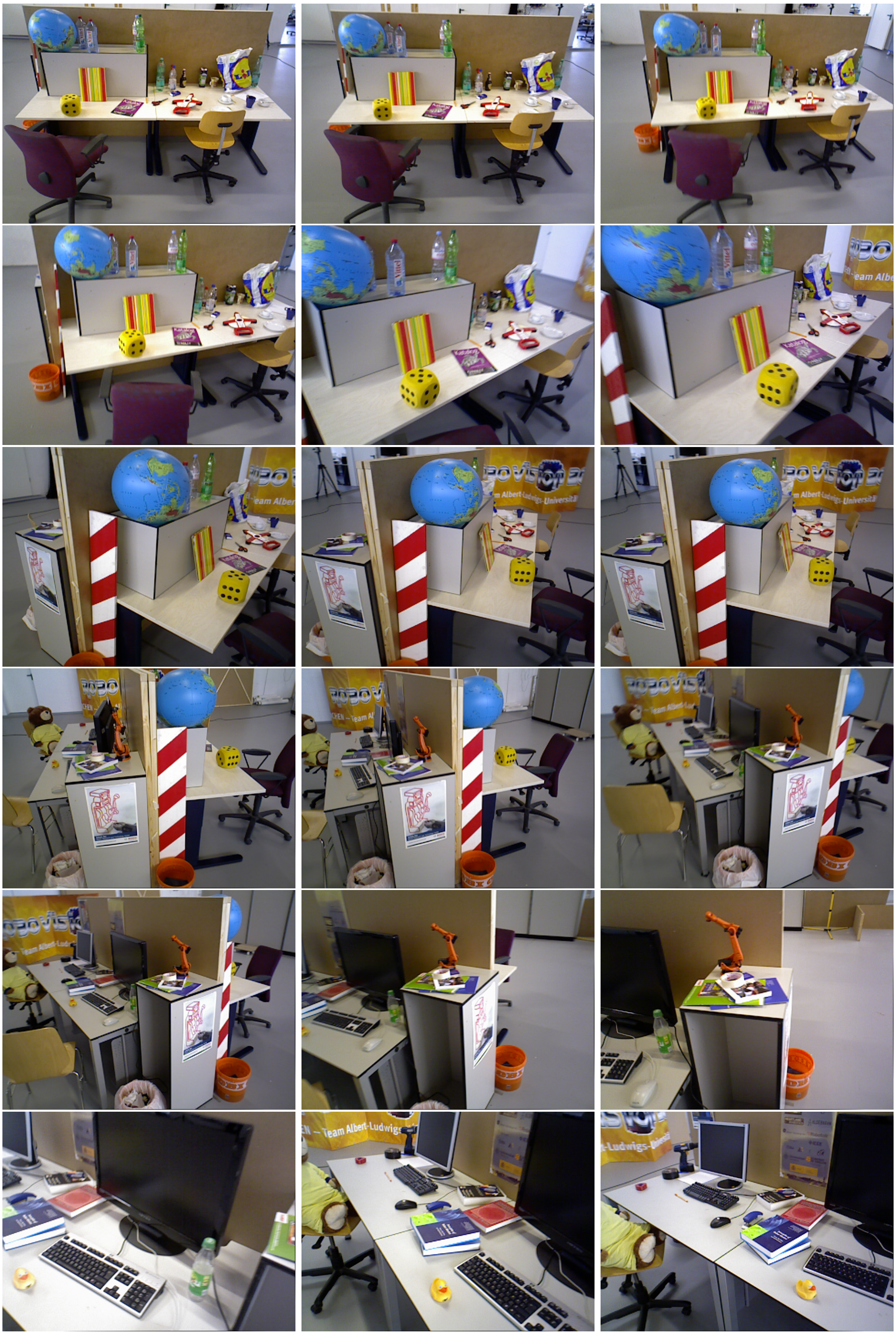


Figure 4.2: Images of the sequence *fr2_desk* taken at different camera poses. Image generated by the author.

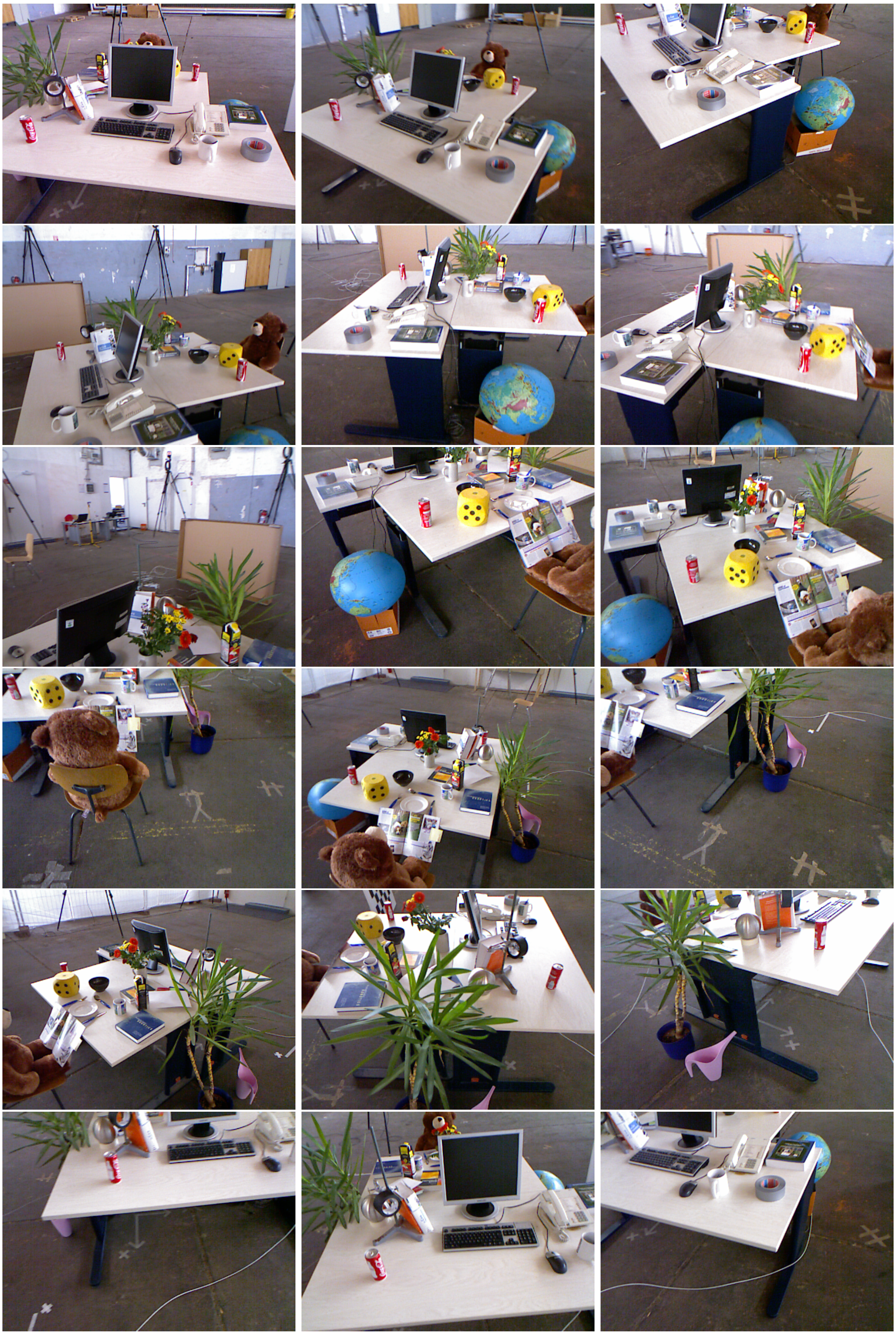


Figure 4.3: Images of the sequence *fr2_desk_person* taken at different camera poses. Image generated by the author.

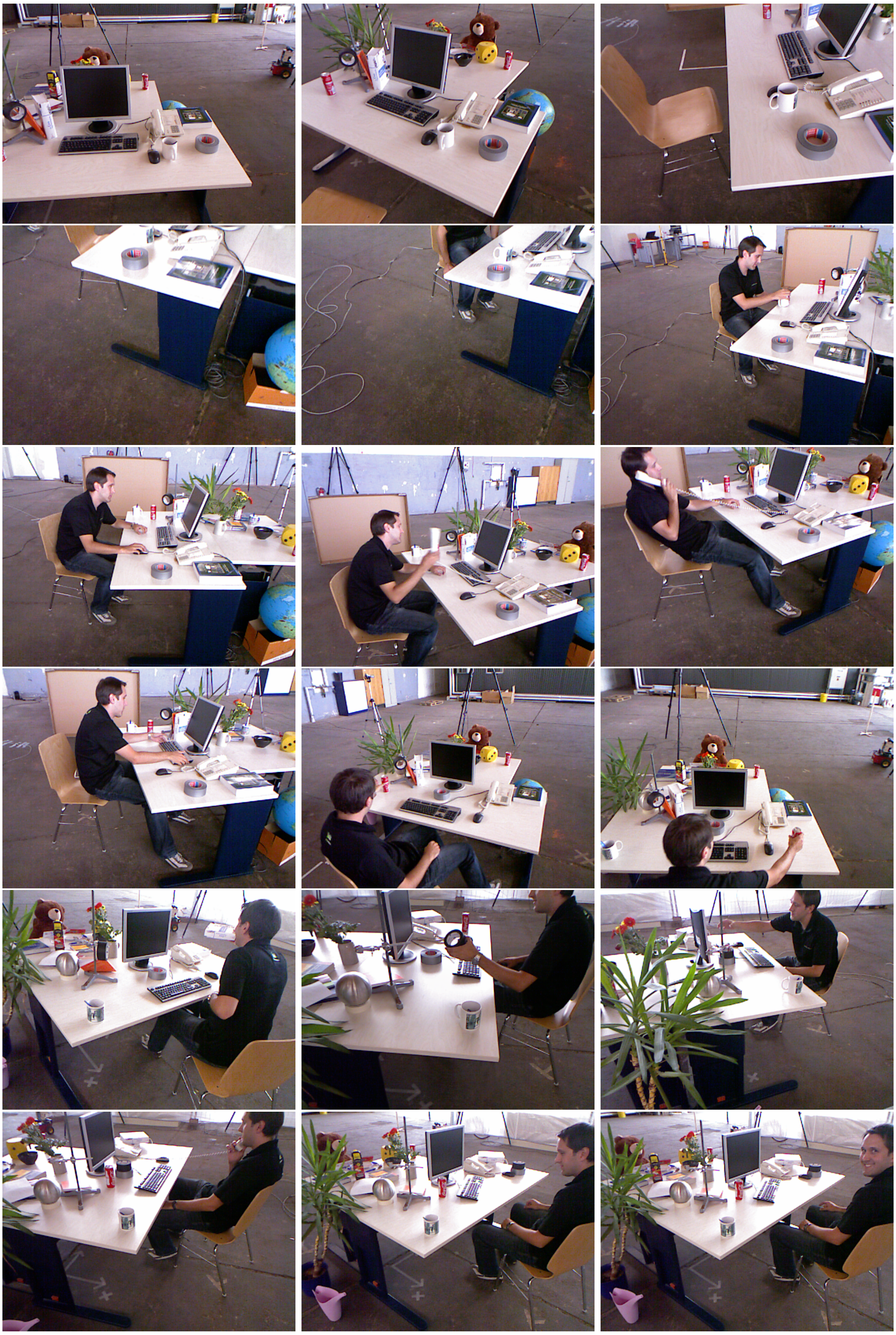


Table 4.3: Quantitative evaluations for 10 separate experiments on the *fr3_long_office* sequence.

Test	Keyframes			Points			ATE[m]						RPE[m]					
							RMSE		mean		std		RMSE		mean		std	
	ORB	Ours	**%**	ORB	Ours	**%**	ORB	**Ours**	ORB	**Ours**	ORB	**Ours**	ORB	**Ours**	ORB	**Ours**	ORB	**Ours**
1	198	170	**14.14**	10175	8250	**18.91**	**0.0102**	0.0142	**0.0095**	0.0127	**0.0039**	0.0063	**0.1569**	0.1580	0.1503	**0.1455**	**0.0450**	0.0615
2	190	165	**13.15**	9955	7881	**20.83**	0.0114	**0.0104**	0.0103	**0.0093**	0.0048	**0.0048**	**0.1534**	0.1547	0.1470	**0.1444**	**0.0437**	0.0555
3	184	157	**14.67**	9580	7794	**18.64**	**0.0098**	0.0209	**0.0090**	0.0178	**0.0039**	0.0110	0.1560	**0.1487**	0.1495	**0.1403**	**0.0447**	0.0493
4	187	155	**17.11**	9829	7634	**22.33**	**0.0119**	0.0127	**0.0107**	0.0113	**0.0052**	0.0056	**0.1538**	0.1563	0.1473	**0.1455**	**0.0442**	0.0571
5	186	160	**13.97**	9679	7620	**21.27**	**0.0101**	0.0116	**0.0090**	0.0106	**0.0044**	0.0047	**0.1551**	0.1556	0.1486	**0.1458**	**0.0444**	0.0544
6	189	159	**15.87**	9777	7652	**21.73**	**0.0110**	0.0149	**0.0101**	0.0136	**0.0043**	0.0061	0.1533	**0.1513**	0.1469	**0.1419**	**0.0438**	0.0523
7	210	159	**24.28**	8857	7640	**13.74**	1.0020	**0.0106**	0.8960	**0.0093**	0.4485	**0.0050**	0.2081	**0.1571**	0.1895	**0.1472**	0.0859	**0.0549**
8	187	167	**10.69**	9610	7973	**17.03**	**0.0114**	0.0138	**0.0104**	0.0121	**0.0046**	0.0067	**0.1551**	0.1567	0.1486	**0.1475**	**0.0444**	0.0531
9	182	159	**12.63**	9350	7880	**15.72**	**0.0109**	0.0117	**0.0099**	0.0106	**0.0047**	0.0048	0.1544	**0.1524**	0.1479	**0.1440**	**0.0443**	0.0499
10	190	161	**15.26**	9668	7864	**18.65**	**0.0108**	0.0124	**0.0102**	0.0109	**0.0037**	0.0059	**0.1517**	0.2023	**0.1454**	0.1806	**0.0434**	0.0912

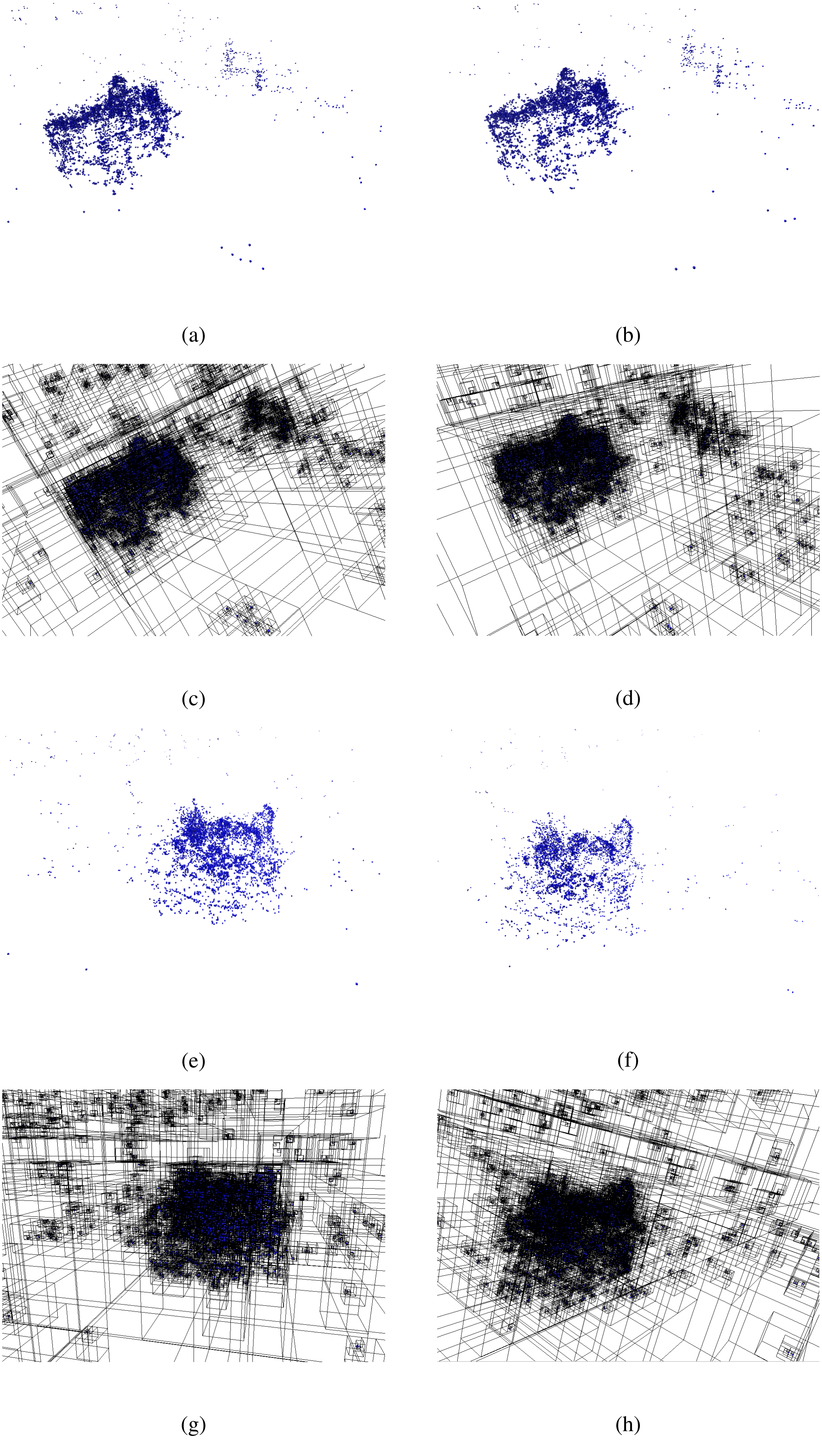
Table 4.4: Quantitative evaluations for 10 separate experiments on the *fr2_desk* sequence.

Test	Keyframes			Points			ATE[m]						RPE[m]					
							RMSE		mean		std		RMSE		mean		std	
	ORB	Ours	**%**	ORB	Ours	**%**	ORB	**Ours**	ORB	**Ours**	ORB	**Ours**	ORB	**Ours**	ORB	**Ours**	ORB	**Ours**
1	167	151	**9.58**	7060	5880	**16.71**	**0.0103**	0.0157	**0.0091**	0.0146	**0.0048**	0.0058	**0.0461**	0.1228	0.0436	**0.1105**	**0.0150**	0.0535
2	166	149	**10.24**	7001	5830	**16.72**	**0.0079**	0.0103	**0.0068**	0.0090	**0.0040**	0.0050	**0.1262**	0.1467	**0.1187**	0.1299	**0.0428**	0.0681
3	168	139	**17.26**	6998	5766	**17.60**	**0.0102**	0.0156	**0.0085**	0.0131	**0.0056**	0.0084	**0.1070**	0.1448	**0.1007**	0.1319	**0.0363**	0.0596
4	166	149	**10.24**	6968	5726	**17.82**	0.0109	**0.0083**	0.0094	**0.0070**	0.0055	**0.0045**	**0.0809**	0.1464	**0.0761**	0.1330	**0.0274**	0.0611
5	168	143	**14.88**	7032	5626	**19.99**	**0.0074**	0.0087	**0.0066**	0.0074	**0.0034**	0.0044	0.1277	**0.0647**	0.1201	**0.0599**	0.0434	**0.0243**
6	166	149	**10.24**	6898	5914	**14.26**	**0.0085**	0.0170	**0.0075**	0.0156	**0.0039**	0.0066	0.0996	**0.0889**	0.0940	**0.0794**	**0.0329**	0.0401
7	166	149	**10.24**	7031	5857	**16.69**	**0.0076**	0.0086	**0.0067**	0.0072	**0.0035**	0.0047	**0.0967**	0.1034	**0.0910**	0.0928	**0.0327**	0.0457
8	167	149	**10.77**	6934	5685	**18.01**	**0.0076**	0.0080	**0.0068**	0.0069	**0.0034**	0.0042	**0.0962**	0.1069	**0.0908**	0.0947	**0.0316**	0.0496
9	177	145	**18.07**	7288	5672	**22.17**	0.0083	**0.0069**	0.0073	**0.0063**	0.0038	**0.0028**	**0.0931**	0.1166	**0.0880**	0.1063	**0.0304**	0.0477
10	168	150	**10.71**	7076	6055	**14.42**	**0.0091**	0.0150	**0.0080**	0.0123	**0.0043**	0.0085	0.1063	**0.1016**	0.1004	**0.0887**	**0.0350**	0.0495

Table 4.5: Quantitative evaluations for 10 separate experiments on the *fr2_desk_person* sequence.

Test	Keyframes			Points			ATE[m]						RPE[m]					
							RMSE		mean		std		RMSE		mean		std	
	ORB	Ours	**%**	ORB	Ours	**%**	ORB	**Ours**	ORB	**Ours**	ORB	**Ours**	ORB	**Ours**	ORB	**Ours**	ORB	**Ours**
1	114	101	**11.40**	4564	3680	**19.36**	**0.0084**	0.0086	0.0073	**0.0068**	**0.0041**	0.0052	**0.0793**	0.1035	**0.0610**	0.0809	**0.0507**	0.0645
2	116	104	**10.34**	4632	3842	**17.05**	0.0092	**0.0082**	**0.0068**	0.0074	0.0061	**0.0034**	0.1132	**0.0874**	0.0882	**0.0635**	0.0709	**0.0601**
3	119	104	**12.60**	4717	3765	**20.18**	0.0103	**0.0075**	0.0092	**0.0064**	0.0047	**0.0040**	**0.0707**	0.0907	**0.0520**	0.0683	**0.0478**	0.0596
4	121	102	**15.70**	4677	3723	**20.39**	0.0100	**0.0094**	0.0086	**0.0076**	**0.0052**	0.0055	**0.1076**	0.1105	**0.0622**	0.0822	**0.0694**	0.0738
5	117	106	**9.40**	4650	3863	**16.92**	0.0121	**0.0080**	0.0107	**0.0068**	0.0056	**0.0042**	0.1209	**0.1140**	0.0959	**0.0903**	0.0735	**0.0695**
6	119	100	**15.96**	4643	3765	**18.91**	0.0084	**0.0068**	0.0069	**0.0059**	0.0047	**0.0033**	**0.0751**	0.0996	**0.0563**	0.0703	**0.0497**	0.0705
7	119	101	**15.12**	4656	3731	**19.86**	0.0092	**0.0081**	**0.0066**	0.0070	0.0065	**0.0040**	**0.0997**	0.1152	**0.0746**	0.0853	**0.0661**	0.0774
8	114	101	**11.40**	4479	3785	**15.49**	0.0102	**0.0069**	0.0083	**0.0055**	0.0059	**0.0042**	**0.1023**	0.1115	**0.0751**	0.0859	**0.0693**	0.0711
9	119	104	**12.60**	4663	3755	**19.47**	**0.0085**	0.0102	0.0078	**0.0073**	**0.0033**	0.0071	**0.0835**	0.1029	**0.0649**	0.0823	**0.0524**	0.0617
10	119	102	**14.28**	4600	3726	**19.00**	**0.0100**	0.8033	**0.0078**	0.5883	**0.0062**	0.5470	**0.1123**	0.1488	**0.0880**	0.1233	**0.0696**	0.0832

Figure 4.4: Resulting maps for the *fr3_long_office* sequence an the *fr2_desk_person* sequence with ORB-SLAM (left) and our method (right) . Image generated by the author.



(a)

(b)

(c)

(d)

(e)

(f)

(g)

(h)

4.2 Relative Pose Error

The relative pose error (RPE) measures the difference between the estimated and the true motion. It is used to evaluate the local accuracy or the drift of a visual odometry system over a fixed time interval Δ. To compute the RPE, the relative transformation between consecutive poses of the estimated trajectory P and the ground truth Q are compared at time step i using:

$$E_i^{\Delta} = \left(Q_i^{-1} Q_{i+\Delta}\right)^{-1} \left(P_i^{-1} P_{i+\Delta}\right), \tag{4.1}$$

Later, the root mean square error (RMSE) is computed over all translational components of these errors along the sequence with time parameter Δ equals to 1, to match consecutive frame. It leads to:

$$RMSE\left(E_{1:n}^{\Delta}\right) = \frac{1}{n} \sum_{\Delta=1}^{n} \left(\frac{1}{m} \sum_{i=1}^{m} \left\| \mathcal{T}(E_i^{\Delta}) \right\|^2 \right)^{\frac{1}{2}} \tag{4.2}$$

Where n is the number of camera poses, m is the number of individual relative pose errors calculated from Equation 4.1 and $\mathcal{T}$ is the function that extracts the translational component from the transformation matrix E_i^{Δ}. This is illustrated in Figure 4.5 where we can see the relative pose error obtained in our experiments for *fr3_long_office*, *fr2_desk* and *fr2_desk_person* sequences. Tables 4.3, 4.4 and 4.5 show the RPE for ten executions with these three sequences. Additionally, we analysed these results further by computing the error histogram shown in Figure 4.6.

Note that we measured the RPE in meters and per frame similar to earlier publication (STURM et al., 2012). Moreover, instead of the median or the standard error we measure the RMSE, which is much more influenced by large occasional errors. On the one hand, we compare our method with ORB-SLAM for static environments that contain significant amounts of noise and outliers that are partially derived from occlusions and reflections in the scene that violate the photo-consistency assumption. Table 4.3 and 4.4 show the results for these static environments, where the root mean square errors of the RPE are computed using their published resulting trajectories (STURM et al., 2012). For each sequence, we show the mean, the standard and the RMSE of the translational error. It can be seen that our approach and ORB-SLAM achieve similar results in most of the experiments and it can be verified in the histograms that are shown in Figure 4.6a, 4.6b, 4.6c and 4.6d. Furthermore, in Figure 4.5b and 4.5d, the camera trajectories estimated by our approach are compared to the ground truth trajectories. Similarly, in Fig 4.5a and 4.5c those that are estimated by the original ORB-SLAM.

On the other hand, we also calculate the RMSE of the RPE for a sequence with a dynamic environment. In Table 4.5, we compare the results between our approach and ORB-SLAM separately. For this sequence, we also show the mean, the standard and the RMSE of the translational error. As can be seen, our approach achieves similar visual odometry accuracy as ORB-SLAM does. The plots in Figure 4.6e and 4.6f confirm that the trajectory estimated by our approach and ORB-SLAM are very close to the ground truth. In general, Table 4.2 shows the quantitative results of the RMSE for the RPE computed in each sequence. The results obtained by our approach and ORB-SLAM are clearly very close in both static and dynamic environments. This implies that our approach performs similar to ORB-SLAM in terms of accuracy. Furthermore, we can conclude that the estimated trajectory by our approach is not affected by the removal of the map points that are considered as outliers, repetitive or obsolete information.

4.3 Absolute Keyframe Trajectory Error

In addition, we use the absolute keyframe trajectory error (ATE) which focuses on global consistency. It is used to evaluate the performance of visual SLAM systems. To compute the ATE, the absolute distances between the estimated keyframe trajectory P and the ground truth Q at time step i are compared using the absolute trajectory error:

$$F_i = Q_i^{-1} S P_i, \tag{4.3}$$

Where S is the rigid-body transformation corresponding to the least-square solution to the alignment problem, which maps the estimated keyframe trajectory P onto the ground truth Q. Similar to the RPE, the RMSE is computed over all translational components of the relative pose error in all time instants. It leads to:

$$RMSE\left(F_{1:n}\right) = \left(\frac{1}{n} \sum_{i=1}^{n} \left\| \mathcal{T}(F_i) \right\|^2 \right)^{\frac{1}{2}} \tag{4.4}$$

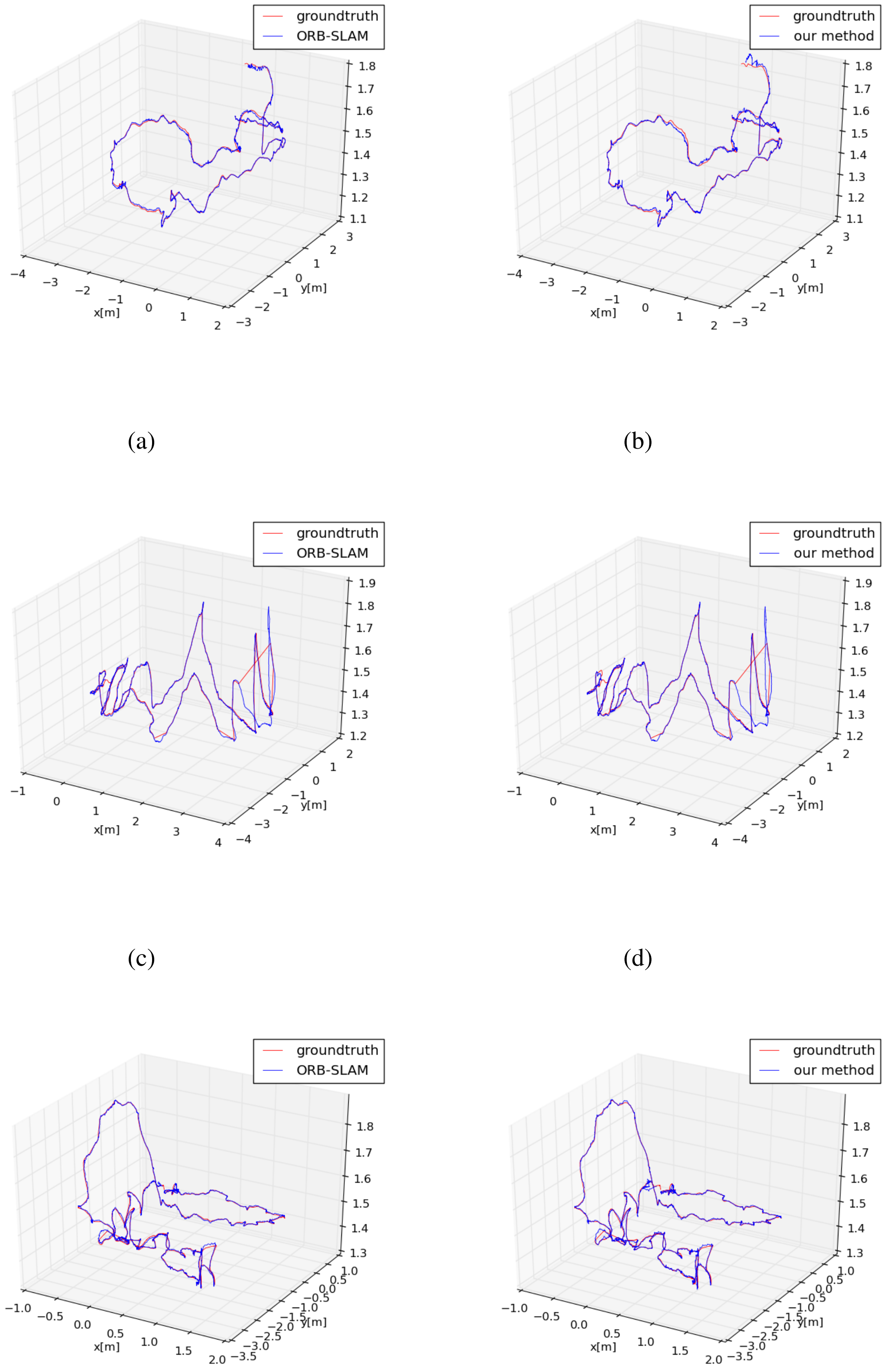
Where $\mathcal{T}$ is the function that extracts the translational component from the transformation matrix F_i. This is illustrated in Figure 4.7 which shows the ground truth and all keyframes in the map, for *fr3_long_office*, *fr2_desk* and *fr2_desk_person* sequences. In these plots, each circle represents a keyframe. On the one hand, the blue circles are those keyframes in the estimated trajectory, which are not considered for the RMSE computation because their matches were not found onto the ground truth at the time of alignment. On the other hand, the remaining cycles are those keyframes that were matches, therefore they are considered for the RMSE computation. For these circles, the filled color represents the ATE in each keyframe.

The results for these three individual sequences are given in Table 4.3, 4.4 and 4.5. Similarly as in the previous section, we calculate the RMSE of the ATE for static and dynamic environments. In Table 4.4 and 4.3, the results obtained by our approach and ORB-SLAM in static environments are shown. However, it is more interesting to observe the results in a dynamic environment. Table 4.5 shows the performance of our approach and ORB-SLAM in the sequence with a dynamic environment. It can be observed that our approach is not affected by changes in the environment, this is because our information fusion algorithm constantly updates the map points. So we can say that the dynamic map points are considered as noise that may cause the whole SLAM process to fail, therefore they are removed.

We observe that the performance of our method is good on the three sequences. We think that the reason is that the three sequences are sufficiently feature rich, allowing to generate a rich map of information and therefore a good performance of our method. For instance, if we look for the sequence *fr3_long_office* in Table 4.2, the results show that the number of keyframes in the map generated by our approach is lower than the one generated by ORB-SLAM, besides the ATE estimated by our approach is greater than the estimated by ORB-SLAM. This is also perceived in Figure 4.7a that shows regions with a high density of keyframes where there is possibly repeated information, outliers or obsolete informations. While in Figure 4.7d this density decreases. In the same way, the alignment obtained by ORB-SLAM is better than that obtained by our method.

Finally, Table 4.2 shows the quantitative results of the RMSE in meters for the absolute keyframe trajectory computed in each sequence, where we can observed that the RMSE produced by ORB-SLAM with our method is very close to the one produced by the original ORB-SLAM.

Figure 4.5: Relative Pose Error evaluation for *fr3_long_office*, *fr2_desk* and *fr2_desk_person* sequences of TUM dataset (STURM et al., 2012), showing the results obtained by ORB-SLAM without our method (left) and with our method (right). Image generated by the author.



(a)

(b)

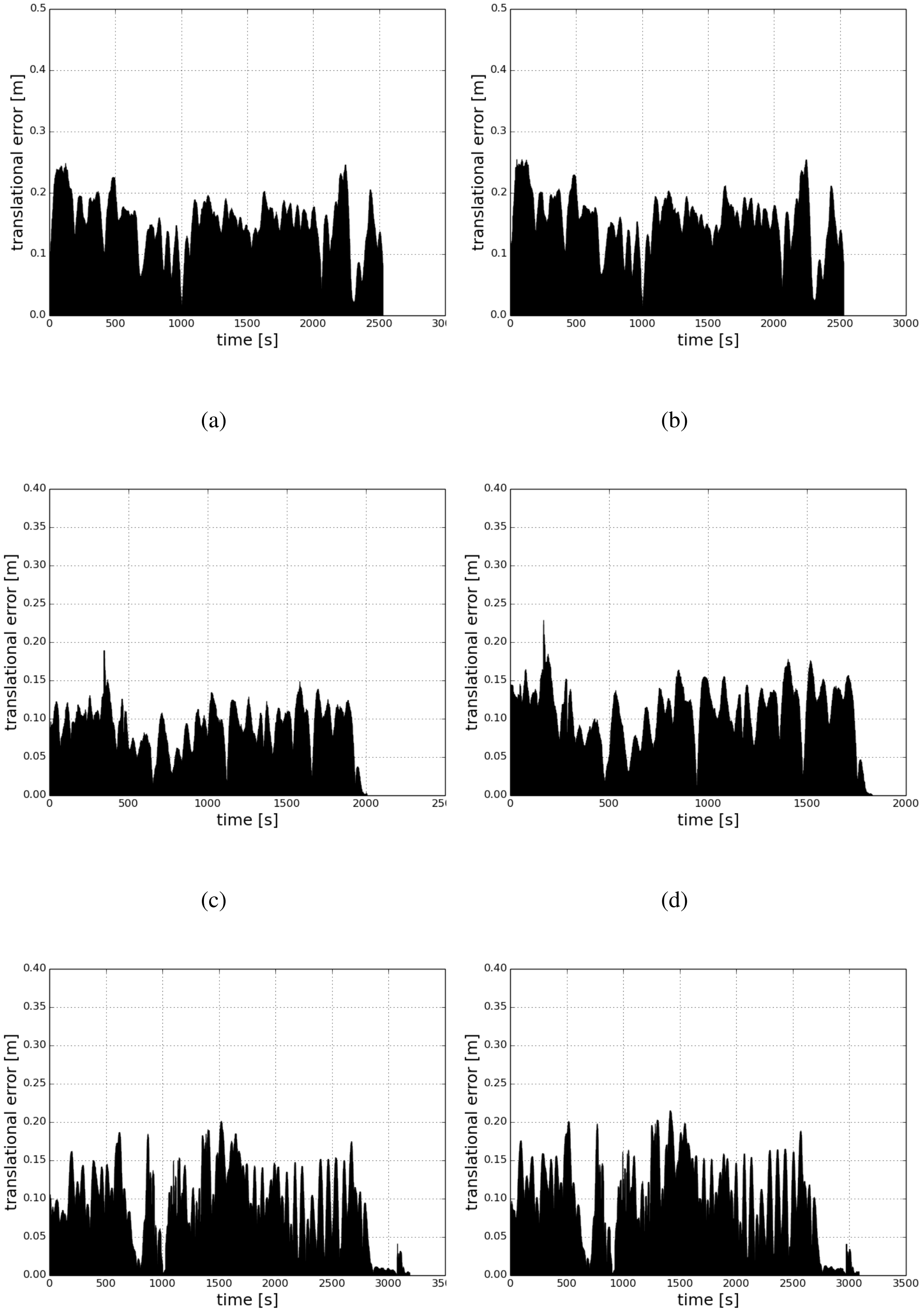
(c)

(d)

(e)

(f)

Figure 4.6: Relative Pose Error for *fr3_long_office*, *fr2_desk* and *fr2_desk_person* sequences of TUM dataset (STURM et al., 2012), showing the results obtained by ORB-SLAM without our method (left) and with our method (right). Image generated by the author.



(a)

(b)

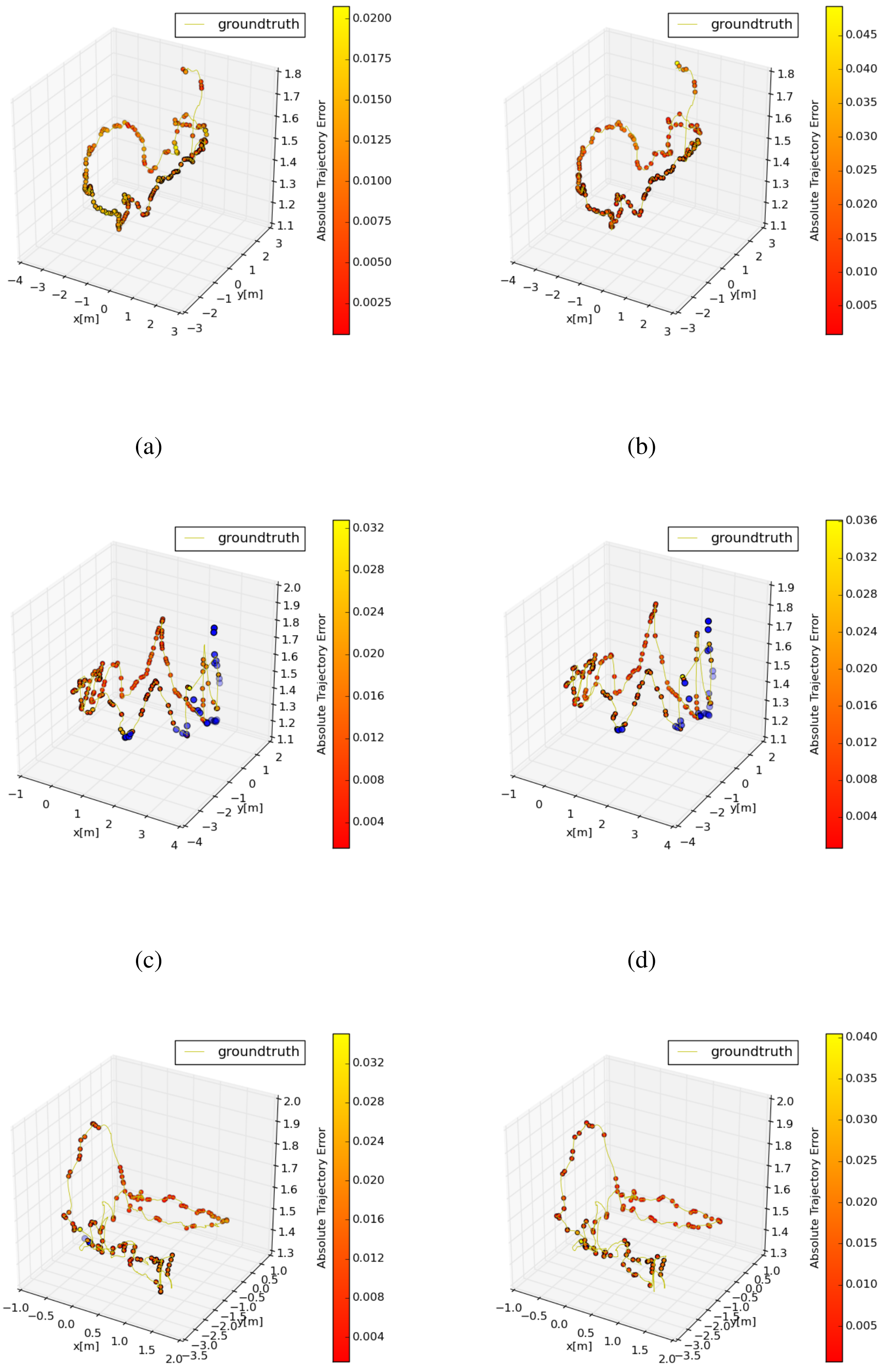
(c)

(d)

(e)

(f)

Figure 4.7: Absolute Keyframe Trajectory Error evaluation for *fr3_long_office*, *fr2_desk* and *fr2_desk_person* sequences of TUM dataset (STURM et al., 2012), showing the results obtained by ORB-SLAM without our method (left) and with our method (right). Additionally, blue points represent those keyframes that are not considered by the alignment method. Image generated by the author.



(a)

(b)

(c)

(d)

(e)

(f)

5 FINAL REMARKS AND FUTURE WORK

In this work, we developed a new method that allows to maintain a refined map in the keyframe-based SLAM. Our approach reduce the map size, removing map points that can have a direct influence on the performance of the SLAM process, such as outliers generated from poor depth estimates, or map points which have been visualized only in some frames and over time have become obsolete. Our approach also deals with repetitive information to maintain a good quality map and counteract the effect of the frequent addition of features.

To achieve this, we employ a monocular keyframe-based SLAM system to build a sparse map of the environment, and to compute a visual pose estimate within this map for each video frame. We contribute with a novel method to detect outliers and combine repetitive information within the map. This is achieved by combining a covisibility graph and an algorithm based on the information fusion to build a probabilistic map. In this sense, our approach represents the depth information of each map point as a mixture of probability distributions and takes advantage of the keyframe neighbourhoods created in the covisibility graph to update these probabilities, and thus maintain the highest possible accuracy of depth estimation. Later, to guarantee a few outliers within the map and reduce the number of redundant keyframes, a pruning policy based on the depth accuracy of the map points is performed.

Finally, the effectiveness of our approach is shown through extensive experiments on publicly available TUM dataset. On the one hand, the performance in static indoor environments achieves a considerable percentage of reduction in the size of the map. Where, these environments present structures and textures that allow to obtain a considerable quantity of information for the development of our method and the keyframe-based SLAM process. On the other hand, we can see that the performance in dynamic indoor environments is not affected by the dynamic changes and similarly as in the static, a considerable percentage of reduction in the size of the map is achieved. Because our method considers the dynamic points as outliers, therefore they are removed from the map and they do not affect the performance of the keyframe-based SLAM process.

In summary, we showed in our experiments that a Visual SLAM system based on keyframes can build a refined, accurate and reliable map representation of the environment which only increases its size if the visual content of the scene changes. This is achieved by removing repetitive, obsolete and outliers information from the map representation without compromise the quality of the trajectory estimate when the camera continuously operating in the same environment. In conclusion, we contribute with a method that facilitates the use of keyframe-based SLAM by a long time within indoor environments, guaranteeing the construction of a refined, accurate and reliable map. As a future work, we propose to investigate the performance of our method to address problems of a collaborative visual SLAM system in dynamic environments using multiple cameras, where all the cameras work together to build a global map.

REFERENCES

BAY, H.; TUYTELAARS, T.; GOOL, L. V. **SURF: Speeded Up Robust Features**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.

BISHOP, C. M. **Pattern Recognition and Machine Learning**. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

CADENA, C. et al. Simultaneous localization and mapping: Present, future, and the robust perception age. **IEEE Transactions on Robotics**, v. 32, p. 1309–1332, Dec 2016.

CANSIZOGLU, E. A.; TAGUCHI, Y.; RAMALINGAM, S. Pinpoint SLAM: A hybrid of 2D and 3D simultaneous localization and mapping for RGB-D sensors. In: **Proc. IEEE International Conference on Robotics and Automation (ICRA)**. [S.l.: s.n.], 2016. p. 1300–1307.

CASTELLANOS, J. A. et al. The SPmap: a probabilistic framework for simultaneous localization and map building. **IEEE Transactions on Robotics and Automation**, v. 15, p. 948–952, Oct 1999.

DAVISON, A. J.; KITA, N. 3D simultaneous localisation and map-building using active vision for a robot moving on undulating terrain. In: **Proc. IEEE International Conference on Intelligent Robots and Systems (IROS)**. [S.l.: s.n.], 2001. p. 384–391.

DAVISON, A. J. et al. MonoSLAM: Real-time single camera SLAM. **Proc. IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 29, p. 1052–1067, Jun 2007.

DELLAERT, F.; KAESS, M. Square root SAM: Simultaneous localization and mapping via square root information smoothing. **The International Journal of Robotics Research**, v. 25, p. 1181–1203, Oct 2006.

DEVERNAY, F.; FAUGERAS, O. Straight lines have to be straight: Automatic calibration and removal of distortion from scenes of structured enviroments. **Proc. SPIE**, v. 2567, p. 14–24, Aug 2001.

DUBBELMAN, G.; BROWNING, B. COP-SLAM: closed-form online pose-chain optimization for visual SLAM. **IEEE Transactions on Robotics**, v. 31, p. 1194–1213, Apr 2015.

EADE, E.; FONG, P.; MUNICH, M. E. Monocular graph SLAM with complexity reduction. In: **Proc. IEEE International Conference on Intelligent Robots and Systems**. [S.l.: s.n.], 2010. p. 3017–3024.

ENGEL, J.; SCHOPS, T.; CREMERS, D. LSD-SLAM: Large-scale direct monocular slam. In: **European Conference on Computer Vision**. [S.l.: s.n.], 2014. p. 834–849.

ETHAN, E. Lie groups for computer vision. In: **Proc. IEEE International Conference on Intelligent Robots and Systems**. [S.l.: s.n.], 2014. p. 3017–3024.

EUSTICE, R. M.; SINGH, H.; LEONARD, J. J. Exactly sparse delayed-state filters for view-based SLAM. **IEEE Transactions on Robotics**, v. 22, p. 1100–1114, Dec 2006.

FAUGERAS, O. D.; LUSTMAN, F. Motion and structure from motion in a piecewise planar environment. **IEEE International Journal of Pattern Recognition and Artificial Intelligence**, v. 2, p. 485–508, May 1988.

FISCHLER, M. A.; BOLLES, R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. **Commun. ACM**, v. 24, p. 381–395, Jun 1981.

FORSTER, C.; PIZZOLI, M.; SCARAMUZZA, D. SVO: Fast semi-direct monocular visual odometry. In: **Proc. IEEE International Conference on Robotics and Automation (ICRA)**. [S.l.: s.n.], 2014. p. 15–22.

FRESE, U.; LARSSON, P.; DUCKETT, T. A multilevel relaxation algorithm for simultaneous localization and mapping. **IEEE Transactions on Robotics**, v. 21, p. 196–207, Apr 2005.

GALVEZ-LOPEZ, D.; TARDOS, J. D. Bags of binary words for fast place recognition in image sequences. **IEEE Transactions on Robotics**, v. 28, p. 1188–1197, Jun 2012.

GRISETTI, G. et al. A tutorial on graph-based SLAM. **IEEE Transactions on Robotics**, v. 2, p. 31–43, Feb 2010.

GRISETTI, G.; STACHNISS, C.; BURGARD, W. Improved techniques for grid mapping with rao-blackwellized particle filters. **IEEE Transactions on Robotics**, v. 23, p. 34–46, Feb 2007.

GRISETTI, G. et al. A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. In: **Proc. Robotics: Science and Systems (RSS)**. [S.l.: s.n.], 2008. p. 65–72.

HAHNEL, D. et al. An efficient fastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In: **Proc. IEEE International Conference on Intelligent Robot and Systems (IROS)**. [S.l.: s.n.], 2003. p. 206–211.

HARTLEY, R.; ZISSERMAN, A. **Multiple view geometry in computer vision**. [S.l.]: Cambridge university, 2003.

HORN, B. K. Closed-form solution of absolute orientation using unit quaternions. **Image and Vision Computing**, v. 4, p. 629–642, Aug 1987.

HORNUNG, A. et al. Octomap: an efficient probabilistic 3D mapping framework based on octrees. **IEEE Autonomous Robots**, v. 34, p. 189–206, Apr 2013.

HOWARD, A.; MATARIC, M. J.; SUKHATME, G. Relaxation on a mesh: a formalism for generalized localization. In: **Proc. IEEE International Conference on Intelligent Robot and Systems (IROS)**. [S.l.: s.n.], 2001. p. 1055–1060.

HUBER, P. J. **Robust Statistics**. Berlin, Heidelberg: Springer, 2011.

ILA, V.; PORTA, J. M.; ANDRADE-CETTO, J. Information-based compact pose SLAM. **IEEE Transactions on Robotics**, v. 26, p. 78–93, Feb 2010.

JAYNES, E. T. **Probability theory: The logic of science**. [S.l.]: Cambridge university press, 2003.

JOHANNSSON, H. et al. Temporally scalable visual SLAM using a reduced pose graph. In: **Proc. IEEE International Conference on Robotics and Automation (ICRA)**. [S.l.: s.n.], 2013. p. 3638–3643.

KAESS, M.; RANGANATHAN, A.; DELLAERT, F. iSAM: Fast incremental smoothing and mapping with efficient data association. In: **Proc. IEEE International Conference on Robotics and Automation (ICRA)**. [S.l.: s.n.], 2007. p. 1670–1677.

KLEIN, G.; MURRAY, D. Parallel tracking and mapping for small AR workspaces. In: **Proc. IEEE International Symposium on Mixed and Augmented Reality**. [S.l.: s.n.], 2007. p. 225–234.

KONOLIGE, K.; BOWMAN, J. Towards lifelong visual maps. In: **Proc. IEEE International Conference on Intelligent Robots and Systems**. [S.l.: s.n.], 2009. p. 1156–1163.

KONOLIGE, K. et al. Efficient sparse pose adjustment for 2D mapping. In: **Proc. IEEE International Conference on Intelligent Robot and Systems (IROS)**. [S.l.: s.n.], 2010. p. 22–29.

KUMMERLE, R. et al. g2o: A general framework for graph optimization. In: **Proc. IEEE International Conference on Robotics and Automation (ICRA)**. [S.l.: s.n.], 2011. p. 3607–3613.

LEPETIT, V.; MORENO-NOGUER, F.; FUA, P. Epnp: An accurate o (n) solution to the pnp problem. **IEEE International Journal of Computer Vision**, v. 81, p. 155–159, Aug 2009.

LOWE, D. G. Object recognition from local scale-invariant features. In: **Proc. IEEE International Conference on Computer Vision (ICCV)**. [S.l.: s.n.], 1999. p. 1150–1157.

LU, F.; MILIOS, E. Globally consistent range scan alignment for environment mapping. **Autonomous Robots**, v. 4, p. 333–349, Oct 1997.

LU, Y.; SONG, D. Visual navigation using heterogeneous landmarks and unsupervised geometric constraints. **IEEE Transactions on Robotics**, v. 31, p. 736–749, Jun 2015.

MAKARENKO, A. A. et al. An experiment in integrated exploration. In: **Proc. IEEE International Conference on Intelligent Robots and Systems, (IROS)**. [S.l.: s.n.], 2002. p. 534–539.

MONTEMERLO, M. et al. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In: **Eighteenth National Conference on Artificial Intelligence**. [S.l.: s.n.], 2002. p. 593–598.

MORE, J. J. The levenberg-marquardt algorithm: implementation and theory. In: **Numerical analysis**. New York, NY: Springer, 1978.

MUR-ARTAL, R.; MONTIEL, J. M. M.; TARDOS, J. D. ORB-SLAM: A versatile and accurate monocular SLAM system. **IEEE Transactions on Robotics**, v. 31, p. 1147–1163, Oct 2015.

MUR-ARTAL, R.; TARDOS, J. D. Fast relocalisation and loop closing in keyframe-based SLAM. In: **Proc. IEEE International Conference on Robotics and Automation (ICRA)**. [S.l.: s.n.], 2014. p. 846–853.

NEWCOMBE, R. A. et al. Kinectfusion: Real-time dense surface mapping and tracking. In: **Proc. IEEE International Symposium on Mixed and Augmented Reality (ISMAR)**. [S.l.: s.n.], 2011. p. 127–136.

NEWCOMBE, R. A.; LOVEGROVE, S. J.; DAVISON, A. J. DTAM: Dense tracking and mapping in real-time. In: **Proc. IEEE International Conference on Computer Vision (ICCV)**. [S.l.: s.n.], 2011. p. 2320–2327.

NISTER, D. An efficient solution to the five-point relative pose problem. **Proc. IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 26, p. 756–770, Aug 2004.

NISTER, D.; NARODITSKY, O.; BERGEN, J. Visual odometry. In: **Proc. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2004. p. 652–659.

OLSON, E.; LEONARD, J.; TELLER, S. Fast iterative alignment of pose graphs with poor initial estimates. In: **Proc. IEEE International Conference on Robotics and Automation (ICRA)**. [S.l.: s.n.], 2006. p. 2262–2269.

PIZZOLI, M.; FORSTER, C.; SCARAMUZZA, D. REMODE: Probabilistic, monocular dense reconstruction in real time. In: **Proc. IEEE International Conference on Robotics and Automation (ICRA)**. [S.l.: s.n.], 2014. p. 2609–2616.

PUMAROLA, A. et al. PL-SLAM: Real-time monocular visual SLAM with points and lines. In: **Proc. IEEE International Conference on Robotics and Automation (ICRA)**. [S.l.: s.n.], 2017. p. 4503–4508.

ROSTEN, E.; DRUMMOND, T. Machine learning for high-speed corner detection. In: **European conference on Computer Vision**. [S.l.: s.n.], 2006. p. 430–443.

RUBLEE, E. et al. ORB: An efficient alternative to SIFT or SURF. In: **Proc. IEEE International Conference on Computer Vision (ICCV)**. [S.l.: s.n.], 2011. p. 2564–2571.

SHI, J.; TOMASI, C. Good features to track. In: **Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 1994. p. 593–600.

SMITH, R.; SELF, M.; CHEESEMAN, P. Estimating uncertain spatial relationships in robotics. In: COX, I. J.; WILFONG, G. T. (Ed.). **Autonomous Robot Vehicles**. New York, NY: Springer New York, 1990.

SMITH, R. C.; CHEESEMAN, P. On the representation and estimation of spatial uncertainty. **The International Journal of Robotics Research**, v. 5, p. 56–68, Feb 1986.

STRASDAT, H.; MONTIEL, J.; DAVISON, A. J. Scale drift-aware large scale monocular SLAM. **Robotics: Science and Systems**, v. 81, p. 155–159, May 2010.

STUMM, E.; MEI, C.; LACROIX, S. Probabilistic place recognition with covisibility maps. In: **Proc. IEEE International Conference on Intelligent Robots and Systems (IROS)**. [S.l.: s.n.], 2013. p. 4158–4163.

STURM, J. et al. A benchmark for the evaluation of RGB-D SLAM systems. In: **Proc. IEEE International Conference on Intelligent Robots and Systems (IROS)**. [S.l.: s.n.], 2012. p. 573–580.

THRUN, S.; BURGARD, W.; FOX, D. **Probabilistic robotics**. New York, NY, USA: MIT press, 2005.

THRUN, S. et al. Simultaneous localization and mapping with sparse extended information filters. **The International Journal of Robotics Research**, v. 23, p. 693–716, Dec 2004.

THRUN, S.; MONTEMERLO, M. The graph SLAM algorithm with applications to large-scale mapping of urban structures. **The International Journal of Robotics Research**, v. 25, p. 403–429, Oct 2006.

TRIGGS, B. et al. Bundle adjustment a modern synthesis. In: **Proc. IEEE International Workshop on Vision Algorithms**. [S.l.: s.n.], 1999. p. 298–372.

VOGIATZIS, G.; HERNANDEZ, C. Video-based, real-time multi-view stereo. **Image and Vision Computing**, v. 29, p. 434–441, May 2011.

WILLIAMS, B.; KLEIN, G.; REID, I. Real-time SLAM relocalisation. In: **Proc. International Conference on Computer Vision (ICCV)**. [S.l.: s.n.], 2007. p. 147–151.

ZHOU, H. et al. StructSLAM: Visual SLAM with building structure lines. **IEEE Transactions on Vehicular Technology**, v. 64, p. 1364–1375, Apr 2015.

APPENDIX A — PROBABILITY AND STATISTICS

A.1 Normal Distribution

One of the most important examples of a continuous probability distribution is the normal distribution, sometimes called the Gaussian distribution $X = \mathcal{N}(x \mid \mu, \sigma^2)$. The density function for this distribution is given by:

$$\mathcal{N}(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left\{-\frac{(\mu - x)^2}{2\sigma^2}\right\} \tag{A.1}$$

where μ and σ are the mean and the standard deviation, respectively. The first and second moment are:

$$E(X) = \mu \tag{A.2}$$

$$E(X^2) = \sigma^2 + \mu^2 \tag{A.3}$$

A.2 Beta Distribution

A random variable $X = Beta(x \mid a, b)$ is said to have the beta distribution, if the density function is:

$$Beta(x \mid a, b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{a-1}(1-x)^{b-1} \tag{??}$$

$$\Gamma(a+1) = a\Gamma(a) \tag{??}$$

where a and b are positive. The first and second moment are:

$$E(X) = \frac{a}{a+b} \tag{A.4}$$

$$E(X^2) = \frac{a(a+1)}{(a+b)(a+b+1)} \tag{A.5}$$