UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIÊNCIAS EXATAS
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Douglas Coelho Braga de Oliveira

Dynamic-Object-Aware Simultaneous Localization and Mapping for Augmented Reality Applications

Juiz de Fora

2018

UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIÊNCIAS EXATAS
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Douglas Coelho Braga de Oliveira

Dynamic-Object-Aware Simultaneous Localization and Mapping for Augmented Reality Applications

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Ciências Exatas da Universidade Federal de Juiz de Fora como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Orientador: Rodrigo Luis de Souza da Silva

Juiz de Fora

2018

Douglas Coelho Braga de Oliveira

Dynamic-Object-Aware Simultaneous Localization and Mapping for Augmented Reality Applications

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Ciências Exatas da Universidade Federal de Juiz de Fora como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Aprovada em 19 de Setembro de 2018.

BANCA EXAMINADORA

Prof. D.Sc. Rodrigo Luis de Souza da Silva - Orientador
Universidade Federal de Juiz de Fora

Prof. D.Sc. Gilson Antonio Giraldi
Laboratório Nacional de Computação Científica

Prof. D.Sc. Marcelo Bernardes Vieira
Universidade Federal de Juiz de Fora

Ficha catalográfica elaborada através do programa de geração
automática da Biblioteca Universitária da UFJF,
com os dados fornecidos pelo(a) autor(a)

Oliveira, Douglas Coelho Braga de.
Dynamic-Object-Aware Simultaneous Localization and Mapping for Augmented Reality Applications / Douglas Coelho Braga de Oliveira. -- 2018.
65 f.

Orientador: Rodrigo Luis de Souza da Silva
Dissertação (mestrado acadêmico) - Universidade Federal de Juiz de Fora, Instituto de Ciências Exatas. Programa de Pós Graduação em Ciência da Computação, 2018.

1. Object tracking. 2. Dynamic map. 3. Graph optimization. I. Silva, Rodrigo Luis de Souza da, orient. II. Título.

Dedicado à minha mãe

e à minha irmãzinha

ACKNOWLEDGMENTS

I would like to thank CAPES for funding my research and all professionals of the post-graduate program in Computer Science of the Federal University of Juiz de Fora. The professors, coordinators, secretaries, cleaning aunts, all people who have contributed in any way to make it possible for me to complete this work. I would also like to thank some special people in my native language.

Ao meu pai Pedro, à minha irmã Bárbara e, principalmente, à minha mãe Elza pelo apoio, carinho e suporte nos momentos mais difíceis. Sem essas pessoas, não restaria nenhuma motivação para eu continuar o trabalho.

Ao meu orientador, Rodrigo Luis, pela enorme paciência (eu mesmo não teria essa paciência), por todo o apoio e conhecimento passado nesses 6 anos em que trabalhamos juntos, desde a graduação até este mestrado. Muito obrigado, de verdade!

Ao professor Marcelo Bernardes pela quase co-orientação durante esse período, foram 4 disciplinas, muito árduas, mas que elevaram bastante meu conhecimento e que, sem dúvida, foram essenciais para a conclusão deste trabalho.

Por fim, agradeço à todos os colegas e amigos que fiz durante essa jornada em busca do título de mestre pelas mais variadas formas de ajuda, desde prestar suporte técnico até desperdiçar o tempo conversando, criando teorias do universo que nem Einstein sonharia e jogando (claro).

Meu muito obrigado à todos.

RESUMO

Realidade Aumentada (RA) é uma tecnologia que permite combinar objetos virtuais tridimensionais com um ambiente predominantemente real, de forma a construir um novo ambiente onde os objetos reais e virtuais podem interagir uns com os outros em tempo real. Para fazer isso, é necessário encontrar a pose do observador (câmera, HMD, óculos inteligentes, etc.) em relação a um sistema de coordenadas global. Geralmente, algum objeto físico conhecido é usado para marcar o referencial para as projeções e para a posição do observador. O problema de Localização e Mapeamento Simultâneo (SLAM) se origina da comunidade de robótica como uma condição necessária para se construir robôs verdadeiramente autônomos, capazes de se auto localizarem em um ambiente desconhecido ao mesmo tempo que constroem um mapa da cena observada a partir de informações capturadas por um conjunto de sensores. A principal contribuição do SLAM para a RA é permitir aplicações em ambientes despreparados, ou seja, sem marcadores. No entanto, ao eliminar o marcador, perdemos o referencial para a projeção dos objetos virtuais e a principal fonte de interação entre os elementos reais e virtuais. Embora o mapa gerado possa ser processado a fim de encontrar uma estrutura conhecida, como um plano predominante, para usá-la como referencial, isso ainda não resolve a questão das interações. Na literatura recente, encontramos trabalhos que integram um sistema de reconhecimento de objetos ao SLAM e incorporam tais objetos ao mapa. Frequentemente, assume-se um mapa estático, devido às limitações das técnicas envolvidas, de modo que o objeto é usado apenas para fornecer informações semânticas sobre a cena.

Neste trabalho, propomos um novo *framework* que permite estimar simultaneamente a posição da câmera e de objetos para cada quadro de vídeo em tempo real. Dessa forma, cada objeto é independente e pode se mover pelo mapa livremente, assim como nos métodos baseados em marcadores, mas mantendo as vantagens que o SLAM fornece. Implementamos a estrutura proposta sobre um sistema SLAM de última geração a fim de validar nossa proposta e demonstrar a potencial aplicação em Realidade Aumentada.

Palavras-chave: Rastreamento de objetos. Mapa dinâmico. Otimização.

ABSTRACT

Augmented Reality (AR) is a technology that allows combining three-dimensional virtual objects with an environment predominantly real in a way to build a new environment where both real and virtual objects can interact with each other in real-time. To do this, it is required to find the pose of the observer (camera, HMD, smart glasses etc) in relation to a global coordinate system. Commonly, some well known physical object, called marker, is used to define the referential for both virtual objects and the observer's position. The Simultaneous Localization and Mapping (SLAM) problem borns from robotics community as a way to build truly autonomous robots by allowing they to localize themselves while they build a map of the observed scene from the input data of their coupled sensors. SLAM-based Augmented Reality is an active and evolving research line. The main contribution of the SLAM to the AR is to allow applications on unprepared environments, i.e., without markers. However, by eliminating the marker object, we lose the referential for virtual object projection and the main source of interaction between real and virtual elements. Although the generated map can be processed in order to find a known structure, e.g. a predominant plane, to use it as the referential system, this still not solve for interactions. In the recent literature, we can found works that integrate an object recognition system to the SLAM in a way the objects are incorporated into the map. The SLAM map is frequently assumed to be static, due to limitations on techniques involved, so that on these works the object is just used to provide semantic information about the scene.

In this work, we propose a new framework that allows estimating simultaneously the camera and object positioning for each camera image in real time. In this way, each object is independent and can move through the map as well as in the marker-based methods but with the SLAM advantages kept. We develop our proposed framework over a state-of-the-art SLAM system in order to evaluate our proposal and demonstrate potentials application in Augmented Reality.

Keywords: Object tracking. Dynamic map. Graph optimization.

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

ATE Absolute Trajectory Error

BA Bundle Adjustment

BoW Bag-of-Words

BRIEF Binary Robust Independent Elementary Features

CDE Cube Displacement Error

EKF Extended Kalman Filter

FAST Features from Accelerated Segment Test

MBR Minimum Bounding Rectangle

ORB Oriented FAST and Rotated BRIEF

PnP Perspective-n-Point

RANSAC Random Sample Consensus

SIFT Scale Invariant Features Transform

SLAM Simultaneous Localization and Mapping

SURF Speeded Up Robust Features

RMSE Root-Mean-Square Error

CONTENTS

1 INTRODUCTION

We, humans, have the ability to localize ourselves in an unknown environment. We can map the area around us in our mind by recognizing patterns and identifying landmarks. These abilities make us able to explore any environment that supports life autonomously. One of the main study fields in robotics is related to replicate our exploration ability in order to make autonomous robots. The Simultaneous Localization and Mapping (SLAM) problem borns from the robotics community as a crucial task to robots become truly autonomous (DURRANT-WHYTE; BAILEY, 2006). The primordial idea is to estimate the robot pose (position and orientation) in the world at the same time it builds a map of the environment, from measurements taken by a set of sensors coupled to the robot. When only cameras are used, the problem becomes purely visual getting support from the Computer Vision field and expanding the application domain.

Augmented Reality (AR) is a technology that allows for combining virtual elements with real ones making a new environment, where real and virtual objects can interact with each other in real time (AZUMA, 1997). A crucial task for any AR system is to detect the observer's viewpoint, i.e., the position and direction from which it observes the scene. This is a localization problem. However, just localizing the observer is not enough to produce the desired effect. We need to align virtual elements over the real world, so we need to find a good referential system for them. For example, if we want to project a virtual cube over a real table, we need to know the position of the table relative to the observer. This is a tracking problem.

The first libraries to support the development of AR applications solved both problems by introducing the environment a simple marker object with known features (KATO, 2005; FIALA, 2005). The marker is the tracking target, i.e. it provides the referential system for virtual objects and also for the observer's position. With the evolution of the tracking techniques and hardware performance, more complex objects naturally presented in the environment could be used as markers (UFKES; FIALA, 2013). On these marker-based systems, localizing a camera or tracking an object is the same problem, because one is positioned relative to the other. The advantage of this approach is to easily define a good referential for the virtual objects and to provide an interface between real and virtual

environments. However, the need for the object to be visible is its main weakness.

The main contribution of SLAM to AR domain is to make possible to track the camera without any previous information about the scene, i.e., it is not necessary to have a known object visible. The first SLAM-based AR applications avoided the tracking problem by identifying on the generated map a predominant plane and using it as the referential for augmentations (KLEIN; MURRAY, 2007; TAN et al., 2013). With the development of SLAM methods integrated to object recognition systems, applications could use such recognized objects as this referential (KIM et al., 2012; RAMADASAN et al., 2015). However, one of the main issues with all SLAM systems is to deal with dynamic maps, i.e., when the environment changes over time. The difficulty in solving this problem causes many works to assume a static map as we will see in Chapter 3. This is not interesting in the context of Augmented Reality, because it limits the interaction between real and virtual environments under arbitrary circumstances.

Our main contribution in this work is the design of a novel optimization framework capable of updating dynamic elements positioning on the SLAM map at the frame rate. This means we can estimate the 6 degrees of freedom (6DoF) of both moving objects and the camera on the map without jeopardizing the performance of the SLAM system. To our knowledge, we are the first to propose an object-aware monocular SLAM system that recognizes and tracks **moving objects** using **only geometric** properties of the map. In this way, AR applications can benefit from both advantages of a marker- and a SLAM-based system. So, our contribution to AR domain is to provide an interface between real and virtual worlds, but giving more freedom for the camera movement since its tracking does not depend on the presence of a known object.

1.1 PROBLEM DEFINITION

Our problem in this work is actually three in one. First, we need to solve the SLAM, which itself is the combination of two problems: Camera Localization and Environment Mapping. In addition, we want to track moving objects on the map. All these three problems must be solved simultaneously. Moreover, we take only monocular observations of the environment using a calibrated RGB camera. So the challenge here is to distinguish camera movement from object motion on the 3D space, which it is a very hard problem. Our hypothesis is only geometric relations between the camera and the map are able to

distinguish these motions, provided there are some visible static points.

In the SLAM literature, there are three main paradigms to solve it. We will discuss with more details about these paradigms in the Section 2.1.2, but for now, we can classify them as filter-based or graph-based. The study carried out in Strasdat et al. (2012) found that to obtain better accuracy for the visual SLAM problem it is more advantageous to increase the number of points than the number of cameras. Hence, graph-based approaches, more specifically those that use a geometric formulation for the map optimization, tend to get better accuracy than those filter-based ones. Based on this study, we assume that this conclusion can be generalized to the moving object tracking problem integrated with SLAM. At first, this may be counter-intuitive because if the object is moving, it is reasonable that by observing more positions, we can get a smoother trajectory. This is true, but we argue that for Augmented Reality the important thing is the current object position, not its entire trajectory. Therefore, in one instant the whole map is static, so we have a visual SLAM problem again.

Our approach follows a geometric formulation for map optimization. In each camera image, we need to search for correspondences between the image and the current map, giving us 3D-to-2D point matches. From these matches, we want to simultaneously estimate the 6DoF pose of each visible object in the current image and the current 6DoF pose of the camera by minimizing a projective error function.

1.2 OBJECTIVES

The main objective of this work is to extend the interaction level between real and virtual objects in an Augmented Reality environment. For this, we believe it is fundamental to adopt SLAM-based methods to increase the freedom of camera movement. Moreover, a marker-based method is still important to provide an interface between the real and virtual worlds. In this way, another objective of our work is to incorporate an object tracking system to state-of-the-art SLAM software in order to combine advantages of both methods. Furthermore, we want to demonstrate potential Augmented Reality applications with the new software.

1.3 OUTLINE

The remainder of this work is organized as follows: Chapter 2 provides the technical and theoretical fundamentals needed to fully understand our work; Chapter 3 describes the SLAM problem evolution and the related works. Chapter 4 presents our proposal and describes the developed system; Chapter 5 shows our experimental results, describes the metrics and the used data sets; Finally, we close our work in Chapter 6 with our conclusions and proposals for future works.

2 FUNDAMENTALS

In this chapter it is presented the main fundamentals of this work.

2.1 SIMULTANEOUS LOCALIZATION AND MAPPING

Our SLAM problem was defined in Section 1.1, but the term SLAM comprehend a large and diversified set of problems. Therefore, to better understand the terms used throughout this work and help us to position our work in the literature, in this section, we describe the SLAM taxonomy and its main paradigms.

2.1.1 TAXONOMY

A SLAM problem may be distinguished by several aspects. When the sensors are only cameras the problem is known as *visual SLAM*, or *vision-based SLAM*, or simply *vSLAM*. The map is the representation of the environment. This representation may be *topological* or *geometric*, *dense* or *sparse* and *static* or *dynamic*. The topological map describes only qualitative relations between places whereas geometric map provides a metric description of the environment. Dense or *volumetric* maps have a high resolution that allows realistic reconstructions of the environment while sparse maps are built over a small set of *features*. *Feature-based* methods are the ones that extract a subset of data from the sensor stream and so they normally build sparse maps. Finally, a static map assumes that the environment does not change over time, which is not true for dynamic maps that must preserve its consistency even in the presence of moving artifacts.

A SLAM system can allow *short-* or *long-term* operation which implies how much space the robot can navigate without loss of map consistency. The long-term operation generally requires *loop-closure* techniques to detect when the robot passes through a previously visited area and to make corrections across the map. This is important to reduce the accumulated uncertainty during the robot navigation. The problem still may be classified by run-time requirements. If the system must generate an output for each sensor measurement input in a timely manner, then the map is built incrementally so we call this problem a *incremental SLAM* or *online SLAM*. However, in cases where only the final result matters and all sensor data is processed at once, we have a *full SLAM* problem.

2.1.2 PARADIGMS

There are three main paradigms for the SLAM problem, they are:

- EKF-based
- Particle Filter
- Graph-based

EKF-based SLAM was the first solution to the SLAM problem. This model uses the Extended Kalman Filter (KALMAN et al., 1960) to estimate a huge state vector composed by the robot position and orientation and the position of each measurement of the environment while updates a covariance matrix that increases quadratically with the number of measurements. The maintenance of this matrix is its main limitation.

The idea behind the particle filter approaches is decomposing the SLAM problem into a localization problem and a collection of landmark estimation problems conditioned by the pose estimations. All K landmark locations are estimated by M particles following a Rao-Blackwellised particle filter algorithm (MURPHY; RUSSELL, 2001). Using a tree structure it is possible to obtain an algorithm with $O(M \log K)$ runtime cost (MONTEMERLO et al., 2002).

Graph-based approaches reduce the SLAM to a least-squares optimization problem. The idea is to build a network with feature locations, robot poses, and spatial relation constraints. Feature locations and robot poses are represented by a node in this network while each spatial relationship between them is represented by an arc linking the correspondent nodes. Then, this structure is optimized by minimizing a geometric error function. The graph-based approaches are currently the state-of-the-art in terms of accuracy (STRASDAT et al., 2012). The computational cost is high but nowadays is possible to achieve real-time performance through task parallelization.

2.2 ORB DESCRIPTORS

The feature space provides a concise and invariant representation of the camera stream as it dramatically reduces the number of points needed to describe the scene and adds mechanisms to recognize the same region in different viewpoints and light conditions. The ORB (Oriented FAST and Rotated BRIEF) descriptor extractor is a robust technique

developed by Rublee et al. (2011) that achieve real-time performance for image matching task. The main difference of ORB to other state-of-the-art techniques like SIFT (LOWE, 2004) or SURF (BAY et al., 2006, 2008) is its binary representation of the feature space rather than floating-point.

The ORB algorithm is a combination of the FAST keypoint detector (ROSTEN; DRUMMOND, 2006) and the BRIEF descriptor extractor (CALONDER et al., 2010) with modifications on both to add in-plane rotation and scale invariance to the feature space. These two components will be detailed below.

2.2.1 ORIENTED FAST

The FAST algorithm detects keypoints in the image through an intensity threshold between a center pixel and those in a circular ring about the center. This threshold is the only parameter of the original FAST algorithm. RUBLEE et al. found FAST has a large response on edges so they employed the Harris corner measure (HARRIS; STEPHENS, 1988) to rank FAST points and pick the top η ones.

ORB builds a scale pyramid of the image and detects FAST keypoints at each pyramid level in order to obtain scale invariance. The authors introduce yet a measure for the corner orientation. Each keypoint k defines a circular region $\mathfrak{H}_k \subset \mathfrak{H}$ centered in k. The orientation angle θ_k is obtained from the intensity centroid of the region $\mathfrak{H}_k$ (Equation 2.1).

$$\theta_k = \arctan_2 \left(\sum_{(x,y)\in\mathfrak{H}_k} x \ \boldsymbol{\iota}(x,y), \sum_{(x,y)\in\mathfrak{H}_k} y \ \boldsymbol{\iota}(x,y) \right) \tag{2.1}$$

Where $\arctan_2$ is the quadrant-aware version of arctan function and $\boldsymbol{\iota} : \mathfrak{H} \to \mathbb{R}$ returns the intensity value of the image point $(x, y) \in \mathfrak{H}_k$.

2.2.2 ROTATED BRIEF

The BRIEF descriptor is a bit array of n binary tests over a smoothed image patch, p. A binary test $\boldsymbol{\tau}$ is defined by Equation 2.2.

$$\boldsymbol{\tau}(\boldsymbol{\iota}_p; \vec{x}, \vec{y}) = \begin{cases} 1 & : \boldsymbol{\iota}_p(\vec{x}) < \boldsymbol{\iota}_p(\vec{y}) \\ 0 & : \boldsymbol{\iota}_p(\vec{x}) \geq \boldsymbol{\iota}_p(\vec{y}) \end{cases} \tag{2.2}$$

Where $\boldsymbol{\iota}_p : \mathfrak{H}_p \to \mathbb{R}$ is the function that returns the intensity of the patch p at a point around its center and $\vec{x}, \vec{y} \in \mathfrak{H}_p$ are the test points. The BRIEF feature descriptor is then defined by Equation 2.3.

$$\boldsymbol{\delta}_{BRIEF}(p) = \sum_{i=1}^{n} 2^{i-1} \boldsymbol{\tau}(\boldsymbol{\iota}_p; \vec{x}_i, \vec{y}_i) \tag{2.3}$$

Each pair of points $\vec{x}_i, \vec{y}_i$ for a test is chosen following a probability distribution. In the paper of BRIEF, Calonder et al. (2010) consider many types of distributions. ORB authors chose a Gaussian distribution around the center of a 31×31 keypoint patch and a vector length $n = 256$. It is important that the tests have high variance and are uncorrelated to make the feature more discriminative, i.e., that responds differently to inputs. For this, ORB has a pre-built table with the n tests which have the greater variance and lower correlation among all possible tests over a 31×31 patch.

The in-plane rotation invariance is achieved by rotating the test points by the orientation angle of the keypoint. Let $\mathfrak{T}$ be the set of all test points and $\boldsymbol{R}_{f\theta_k}$ the rotation matrix correspondent to the orientation of k discretized to increments of $f^{-1} = 2\pi/30$, then we define $\mathfrak{T}_{\theta_k} = \{\boldsymbol{R}_{f\theta_k}\vec{x} \mid \vec{x} \in \mathfrak{T}\}$. Finally, the ORB feature descriptor for a patch p_k is defined by Equation 2.4.

$$\boldsymbol{\delta}_{ORB}(p_k, \theta_k) = \boldsymbol{\delta}_{BRIEF}(p_k) \mid \vec{x}_i, \vec{y}_i \in \mathfrak{T}_{\theta_k} \tag{2.4}$$

2.3 FEATURE MATCHING

Feature matching is the process of finding the best correspondences between two sets of features. The best correspondence is the one that has the smaller distance in the feature space. For floating-point descriptors, the L2-norm is often used. In the binary domain, this function is equivalent to the Hamming distance, but the last is much faster. Hamming distance is the number of positions in which two words of the same size differ. In the case of binary descriptors like ORB or even BRIEF, it can be defined as the number of bits 1 from the **xor** operation between them.

The feature matching is a fundamental step of any visual SLAM system since it is responsible to perform the crucial task for this problem: data association. A poor correspondence can cause a wrong point triangulation, a displacement in the camera pose and a

bad object recognition. So, it is very important to ensure good matching. Many filters can be applied to eliminate false-positives like the maximum distance, distance ratio, cross-validation etc. The first one simply defines a threshold for the maximum distance between the descriptors. The second one defines a threshold for the ratio between the distances of the two best correspondences. Finally, cross-validation ensures that the correspondence $a \rightarrow b$ is the best mutually, i.e., b is the best for a as a is the best for b. Nevertheless, robust estimators like RANSAC (FISCHLER; BOLLES, 1987) are important to reject potential survivor outliers while it estimates the camera model.

2.4 BAG OF WORDS

When we want to compare an image against a large set of images, for instance, to do image retrieving in a web search, it is computationally impracticable to perform the matching with each image in the set individually. So, we need to structure the database in some way to allow speedup the information retrieval process.

The Bag of Words (BoW) model was originally proposed to classify text documents from a vocabulary of keywords (HARRIS, 1954). In this model, the grammatical structure and word order are disregards. Just the set of words present in documents and their frequency are used for classification. This model was posteriorly adapted for images (CSURKA et al., 2004), where words represent the visual descriptors.

The construction of the vocabulary of visual descriptors is performed from the recursive clustering of descriptors, building a clustered tree whose internal nodes are the cluster centroid and the leaves are the visual words. For float-pointing descriptors, the k-means clustering algorithm is commonly used to find k clusters that segment the set of descriptors. In another way, for binary strings like ORB, Gálvez-López and Tardos (2012) proposed to use the k-medians algorithm, whose only difference to k-means is to compute the median of each descriptor component instead of the arithmetic mean. Once the vocabulary was created, we can classify a new image by finding the vocabulary words present on the image and checking documents that most share these words.

In this work, we use the same BoW model proposed by GÁLVEZ-LÓPEZ; TARDOS to structure our object database in order to perform object recognition on a camera image.

2.5 RIGID-BODY TRANSFORMATION

Camera and object positioning in the tridimensional space are represented by rigid-body transformations. A given 3D rigid-body transformation, $\boldsymbol{T}$, it is an element of the Lie group of the Special Euclidean transformations, $\boldsymbol{T} \in SE(3)$, defined by the matrix on the following equation.

$$\boldsymbol{T} = \left[\begin{array}{c|c} \boldsymbol{R} & \boldsymbol{t} \\ \hline \boldsymbol{0} & 1 \end{array}\right] \tag{2.5}$$

Where $\boldsymbol{R} \in SO(3)$ is a rotation 3x3 sub-matrix and $\boldsymbol{t} \in \mathbb{R}^3$ is a translation sub-vector. They represent, respectively, the object orientation and its center position. This representation is efficient when we want to compose transformations, which can be done via matrix multiplication but it is over-parameterized. For the optimization process, we need minimal representation. For this, we convert the rotation sub-matrix $\boldsymbol{R}$ to its quaternion form, $\mathbf{q} = [q_w, [q_x, q_y, q_z]]$ (Equation 2.6). Then we compose the imaginary-part components of $\mathbf{q}$ with the components of vector $\boldsymbol{t}$, building the 6-vector $\tau = [t_x, t_y, t_z, q_x, q_y, q_z] \in \mathbb{R}^6$. The real component of $\mathbf{q}$ may be recovered by $q_w = \sqrt{1 - q_x^2 - q_y^2 - q_z^2}$.

$$\begin{aligned} &\mathbf{q}(\boldsymbol{R}) = [\ \omega/2,\ [(\boldsymbol{R}_{3,2} - \boldsymbol{R}_{2,3})/2\omega,\ (\boldsymbol{R}_{1,3} - \boldsymbol{R}_{3,1})/2\omega,\ (\boldsymbol{R}_{2,1} - \boldsymbol{R}_{1,2})/2\omega]\ ] \\ &\text{where} \quad \omega = \sqrt{1 + trace(\boldsymbol{R})} \end{aligned} \tag{2.6}$$

We can restore the matrix representation by doing the inverse process. First, we decompose the 6-vector τ to the quaternion $\mathbf{q}$ and the vector $\boldsymbol{t}$. Then we convert $\mathbf{q}$ to its equivalent rotation matrix $\boldsymbol{R}$ (Equation 2.7) and we build the matrix $\boldsymbol{T}$ as defined by Equation 2.5.

$$\boldsymbol{R}(\mathbf{q}) = \begin{bmatrix} 1 - 2(q_y^2 + q_z^2) & 2(q_x q_y - q_w q_z) & 2(q_x q_z + q_w q_y) \\ 2(q_x q_y + q_w q_z) & 1 - 2(q_x^2 + q_z^2) & 2(q_y q_z - q_w q_x) \\ 2(q_x q_z - q_w q_y) & 2(q_y q_z + q_w q_x) & 1 - 2(q_x^2 + q_y^2) \end{bmatrix} \tag{2.7}$$

We use the notation $\boldsymbol{T}_{a,b}$ to denote the rigid-body transformation from the coordinate system of b to the referential of a. Its inverse is denoted as $\boldsymbol{T}_{b,a}$ that can be easily

computed as defined by the following equation.

$$\boldsymbol{T}_{b,a} = \boldsymbol{T}_{a,b}^{-1} = \left[\begin{array}{c|c} \boldsymbol{R}_{a,b} & \boldsymbol{t}_{a,b} \\ \hline \mathbf{0} & 1 \end{array}\right]^{-1} = \left[\begin{array}{c|c} \boldsymbol{R}_{a,b}^{T} & -\boldsymbol{R}_{a,b}^{T}\boldsymbol{t}_{a,b} \\ \hline \mathbf{0} & 1 \end{array}\right] \tag{2.8}$$

2.6 BUNDLE ADJUSTMENT

Bundle Adjustment (BA) is the problem of refining a visual reconstruction to produce *jointly optimal* 3D structure and viewing parameters, i.e., camera positioning and/or calibration (TRIGGS et al., 2000). The term *optimal* means the parameters are optimized by minimizing some error function. *Jointly* means the parameters are simultaneously estimated and they are optimal for both scene structure and camera positioning. The idea behind the problem is to adjust the light rays (*bundles*) leaving each 3D point and converging on each camera center (Figure 2.1).

The classical solution for the BA problem follows a nonlinear least squares formulation. Suppose that we have vectors of observations $\overline{\mathbf{z}_i}$ predicted by a model $\mathbf{z}_i = \mathbf{z}_i(\mathbf{x})$, where $\mathbf{x}$ is the state vector we want to optimize. The problem is to estimate the optimal values of $\mathbf{x}$ that minimize some cost function $\boldsymbol{\rho}$ (Equation 2.9).

$$\hat{\mathbf{x}} = \arg_{\mathbf{x}} \min \sum_i \boldsymbol{\rho}(\Delta\mathbf{z}_i(\mathbf{x})^T \ \boldsymbol{W}_i \ \Delta\mathbf{z}_i(\mathbf{x})) \tag{2.9}$$

Where $\Delta\mathbf{z}_i(\mathbf{x}) \equiv \overline{\mathbf{z}_i} - \mathbf{z}_i(\mathbf{x})$ is the measurement prediction error and $\boldsymbol{W}_i$ is an arbitrary symmetric positive definite weight matrix. In this work, we apply this formulation in different ways. We can optimize only the camera parameters or even the entire map structure composed of several cameras, points, and objects.

The objective function may be expressed as a graph. Each component of the state vector $\mathbf{x}$ defines a vertex in this graph. Each observation of a point by a camera defines an edge between their corresponding vertices, constrained by the measurement prediction error $\Delta\mathbf{z}_i$. Figure 2.2 shows how the minimization function in Equation 2.9 may be transformed into a graph structure. The optimization library g^2o designed by Kümmerle et al. (2011) is a state-of-the-art, open-source framework for optimizing nonlinear error functions based on a graph structure. The core algorithms of our system were implemented using this framework.

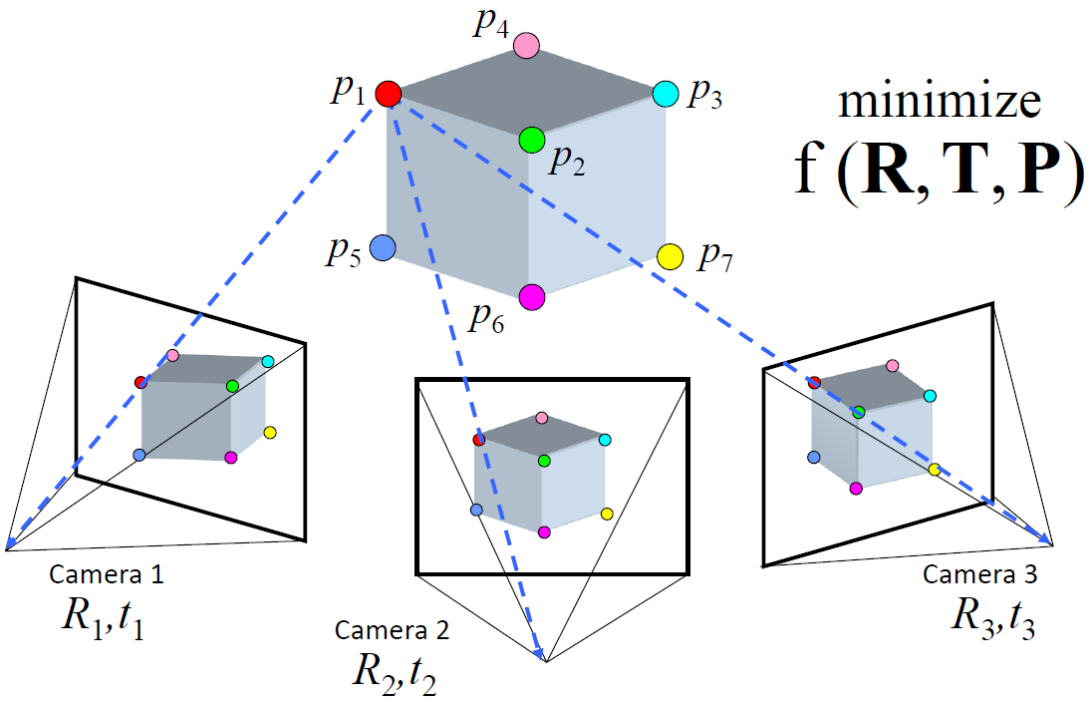


Figure 2.1: The *bundles* are the light rays leaving the scene structure and converging in the camera optical center. Bundle Adjustment is the problem to adjust these *bundles* for an optimal state both of the structure and camera parameters (FERGUS, 2012).

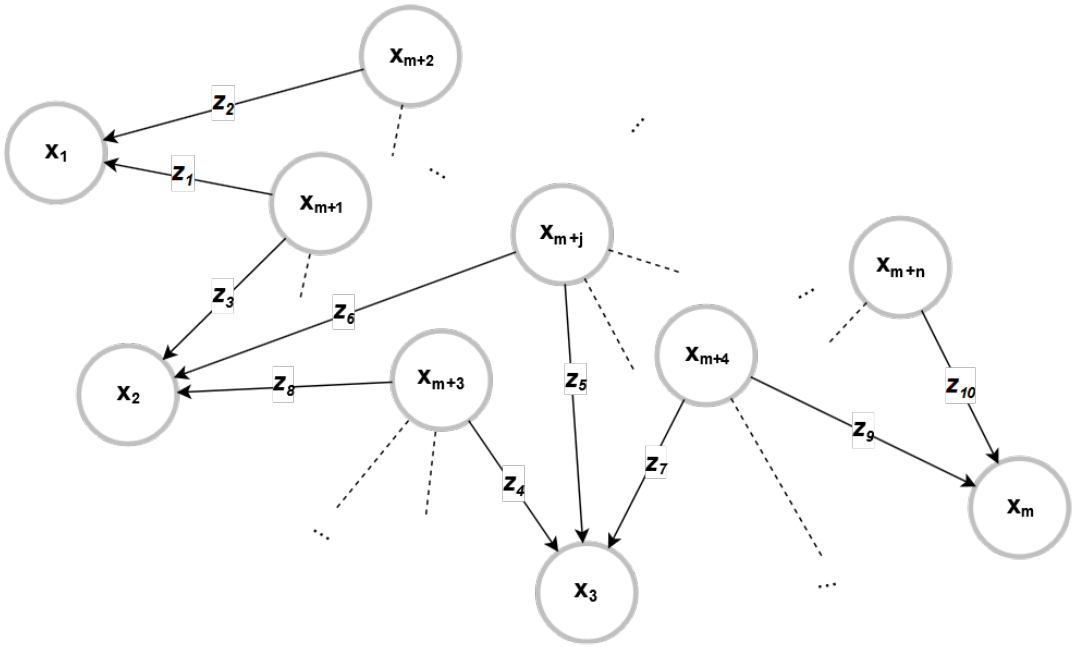


Figure 2.2: Representation of the objective function of the bundle adjustment as a graph. Each component of the state vector (camera, points etc) is a node in this graph. The arcs links a structure element to cameras that observes it. Each observation are constrained by its measurement prediction error model.

3 RELATED WORKS

As the Durrant-Whyte and Bailey (2006) tutorial work tells us, the history of the SLAM problem begins at the *1986 IEEE Robotics and Automation Conference*, where a few researchers began to discuss the application of probabilistic methods in the localization and mapping problems. From this discussion was established that conceptual and computational issues involved to a consistent probabilistic mapping needed to be addressed by robotics community. Over the next few years, relevant papers were published. Smith and Cheeseman (1986) and Durrant-Whyte (1988) showed how to represent and manipulate the geometric uncertainty between successive observations of landmarks. Smith et al. (1990) showed that these landmarks are all correlated with each other due the uncertainty in the estimated location of the robot (observer). This result implied that the combined localization and mapping problem required a huge state vector which increases the computational cost with the square of the number of the landmarks, but this work did not demonstrate the convergence properties of the map. Taking into account the computational complexity and without proof of convergence, the localization and mapping problems were dealt separately again until the Durrant-Whyte et al. (1996) survey paper finally prove the convergence of the map for the combined problem. This work was also the first to coin the acronym 'SLAM' and to define the structure of the problem.

The first full solution to the SLAM problem was taken by Dissanayake et al. (2001) which used the Extended Kalman Filter (EKF) (KALMAN et al., 1960) to estimate the state vector (robot pose and landmarks location) and update the covariance matrix. The second paradigm was born from the work of Montemerlo et al. (2002) which presented the FastSLAM algorithm. The authors proposed the use of a Rao-Blackwellised particle filter (MURPHY; RUSSELL, 2001) to recursively estimate the posterior distribution over the state vector which scales logarithmically with the number of landmarks in the map. Finally, graph-based SLAM approaches were proposed for the first time in the work of Lu and Milios (1997), but only after the work of Klein and Murray (2007), this approach became feasible for real-time applications. These three main paradigms cover a wide range of the SLAM literature as we will see in the next sections.

3.1 VISUAL SLAM

As explained in the Section 2.1.1, visual SLAM is an instance of the SLAM problem that makes use of cameras only and it has been one of the most prominent study fields in the last years. This popularity may be attributed to the fact that cameras capture information about the environment's appearance like color and texture, making possible to integrate other tasks into the pipeline like the detection and recognition of objects. Furthermore, cameras are less expensive than other exteroceptive sensors like radars or laser scanners (FUENTES-PACHECO et al., 2015). However, there are many challenges related to the data association step like illumination changes, camera motion blur due to fast movement, scene dynamics among others. In this section, we cover the visual SLAM literature.

Davison and Murray (2002) work was perhaps the first visual SLAM. They used only a set of calibrated stereo cameras attached to a mobile robot to capture information from the environment. The robot state was just ground plane position and orientation (3DoF) and each feature state was a 3D position. All these variables were estimated by a standard EKF-based SLAM. A short time later, Davison (2003) presented the first full 3D (6DoF) monocular SLAM system based on his previous stereo work. This system was later called MonoSLAM and advanced uses were demonstrated in a humanoid robot and Augmented Reality applications (DAVISON et al., 2007).

Eade and Drummond (2006) were the first to apply a FastSLAM-like particle filter to monocular SLAM reaching real-time performance while observing hundreds of landmarks. In the same year, MOURAGNON et al. were the first to introduce concepts of *keyframes* and *local mapping* for graph-based approaches in order to speedup the optimization step.

However, graph-based approaches only became really feasible for a real-time application thanks to the work by Klein and Murray (2007). This work breaks the concept of a truly simultaneous localization and mapping by proposing a new framework for graph-based SLAM called *PTAM* (Parallel Tracking and Mapping). PTAM framework is composed of a front-end and a back-end thread. The front-end thread is responsible to make data association and to estimate the current camera's pose by projecting the current map in the image frame while the back-end thread selects keyframes, adds new 3D features into map, and makes both local and global optimization on the map. PTAM is capable of running at frame rate even with large maps with thousands of 3D features and dozens

of keyframes, but it is still limited to small-term operation since it does not make loop closure. However, the system was created focusing on indoor Augmented Reality applications so long-term operation is not really needed. PTAM framework was later optimized for mobile devices (KLEIN; MURRAY, 2009). After PTAM, graph-based approaches, which already had better accuracy results, now also have equivalent or even superior performance compared to the other SLAM paradigms. From this point, the problem became improving the algorithms and including new features to the framework.

Strasdat et al. (2010) proposed 7DoF optimization to correct scale-drift when monocular cameras are used in a long-term operation. Pirker et al. (2011) addressed the environment change problem in a long-term monocular SLAM by proposing the Histogram of Oriented Cameras (HOC) to describe feature visibility information. The HOC descriptor was used both to improve data association and to detect dynamic features and remove them from the map. Tan et al. (2013) also addressed the dynamic environment problem for monocular SLAM, but their focus was on indoor Augmented Reality applications. They proposed a variant of RANSAC estimator algorithm called PARSAC that assumes that static background features are evenly distributed over the entire image while dynamic features tend to aggregate in a few small texture areas. Based on this, PARSAC looks not only for the number of inlier correspondences but also for the distribution of these inliers over the image to improve the outlier rejection and thus the camera's pose estimation too.

All works described above are feature-based methods, i.e., they extract from each camera image a small set of features like corners or high-intensity gradients regions. Therefore, the map built by these methods is a cloud of sparse points. PTAM's framework also allowed dense or semi-dense map reconstructions in a real-time system, among which we cite Newcombe et al. (2011), Engel et al. (2014), Forster et al. (2014) and Engel et al. (2017).

Our proposal is based on *ORB-SLAM* (MUR-ARTAL et al., 2015; MUR-ARTAL; TARDÓS, 2017) that is an open-source, feature-based SLAM for both monocular, stereo and RGB-D cameras and may be seen as an extension of PTAM. The authors inserted a third thread in the framework responsible to detect loops and to correct the accumulated drift allowing long-term operation. The global optimization was moved to Loop Closing thread while the Mapping one makes only local optimization based on a co-visibility graph of keyframes. Tracking failures were treated by a new re-localization module based

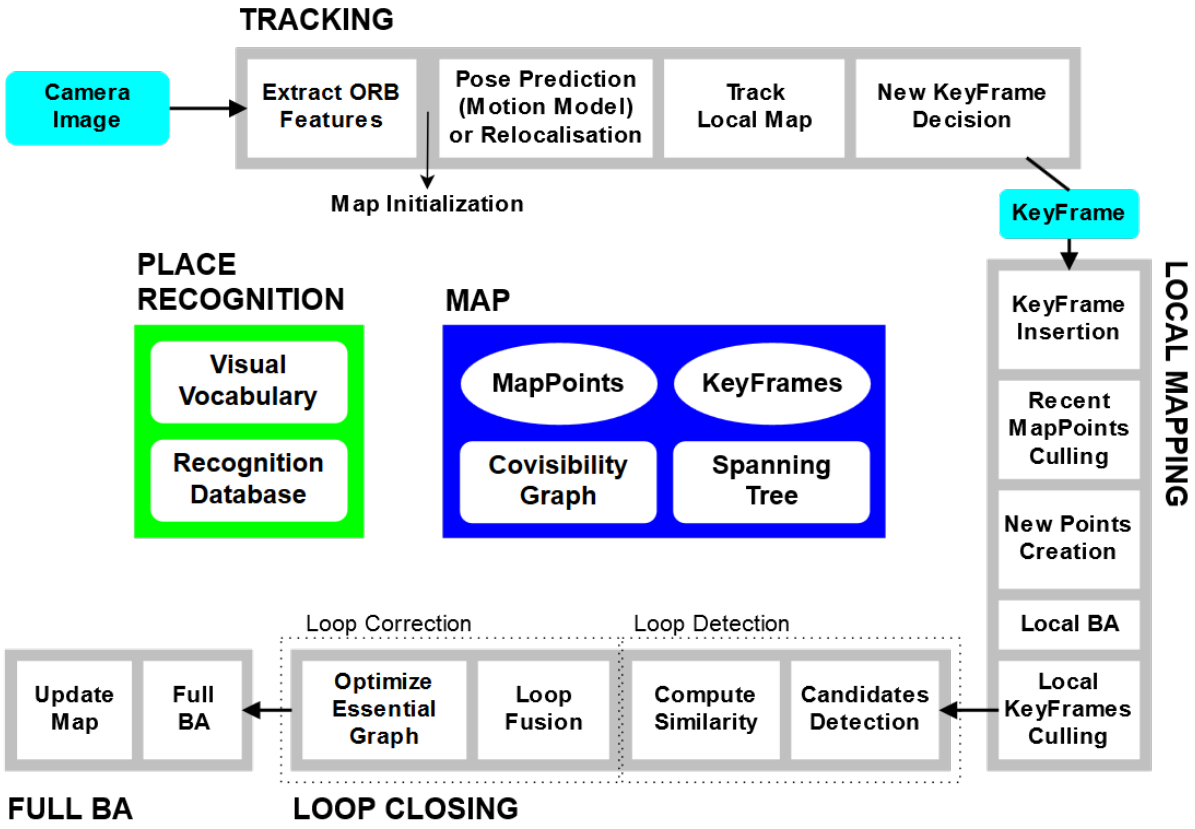


Figure 3.1: Monocular ORB-SLAM system overview. The system runs in parallel three main threads: Tracking, Local Mapping and Loop Closing, which creates a fourth temporary thread to run the global *bundle adjustment* after a loop is closed. (Adapted from (MUR-ARTAL et al., 2015; MUR-ARTAL; TARDÓS, 2017))

on state-of-the-art place recognition technique (GÁLVEZ-LÓPEZ; TARDOS, 2012). A new automatic initialization method, which takes no prior about scene structure, was implemented dispensing any kind of user interaction as PTAM required. The system used ORB features (RUBLEE et al., 2011) for all tasks, i.e., tracking, mapping, re-localization, and loop detection making it more efficient and simple. The ORB descriptors are fast to compute and provide good invariance to changes in illumination and viewpoint. An overview of ORB-SLAM system for monocular camera is illustrated in Figure 3.1. Our system extends the functionality of monocular ORB-SLAM by incorporating a moving object tracking system to its framework using the ORB features.

3.2 OBJECT SLAM

We define the *object SLAM* as an instance of the visual SLAM problem when a prior knowledge about some objects in the environment is available and we want to augment

the map with these objects. Moreover, the objects can stay static or they can be dynamic and move through the environment. In the second case, it is required the tracking of the object's trajectory to maintain the map consistency over time. In this section, we describe the most related SLAM systems with our work.

One of the first works that incorporate object recognition in the graph-based SLAM framework was the one by Castle and Murray (2011). The authors added a new thread responsible for detecting known objects to a PTAM-like framework. The object recognition was performed via SIFT features (LOWE, 2004) matching against an object database. Once two keyframes were successful matching with the same object, the corresponding pair of SIFT features were triangulated and the object's pose in the map was computed. However, objects were not added to the mapping's optimization. The object database was built from dozens of planar pictures described by SIFT.

Using a RGB-D camera, the works by Fioraio and Stefano (2013) and Salas-Moreno et al. (2013) augmented the SLAM map with 3D objects. The first one proposed a Semantic Bundle Adjustment framework for jointly optimizing cameras, points, and objects. The recognition was made by finding putative 3D-to-3D correspondences that were filtered in a RANSAC-like scheme. This recognition proposal is the main bottleneck of their system which takes a few seconds to run thus a real-time application is not possible. The SALAS-MORENO et al.'s work though is able to achieve real-time performance by exploiting GPU computation. The recognition algorithm was based on Hough voting method (BALLARD, 1987).

Employing meta-programming techniques, Ramadasan et al. (2015) proposed the Dynamically Constrained SLAM (DCSLAM). The main point about this system is that it generates, at compile time, a dedicated least-squares optimization algorithm. This allows superior performance compared to other state-of-the-art optimization libraries. Classes of objects are defined a priori so after detecting a new object it is added to optimization process with known constraints. The recognition step is a semi-automatic process where the user may select an area in the image where the object is present. The algorithm tries to fit the reconstructed points inside the selected convex hull with each 3D model of object class and the best fit is kept.

Gálvez-López et al. (2016) were the first to achieve real-time performance for object detection with large database (up to 500 objects). The object database is composed of

a visual vocabulary, an inverted index, and a direct index. The visual vocabulary is a BoW-based clustered tree built from a set of ORB features extracted from the entire object model (GÁLVEZ-LÓPEZ; TARDOS, 2012). The inverted index maps a word (tree leave) to objects that contain this word while the direct index maps an object model to its ORB features and the tree nodes, in a specific level, that contain them. With the inverted index is possible to quickly retrieve candidate objects to be present in the query image. For each candidate object, the recognition algorithm tries to estimate its pose. If the candidate's pose is successfully estimated then the object is added to the map. Finally, the back-end jointly optimizes points, cameras, objects and the map scale that is possible since object scale is known.

All object SLAM systems presented above have a common limitation: they are not able to deal with dynamic objects. In the SLAMMOT (SLAM and Moving Object Tracking) literature we found only filter-based approaches for the tracking moving features (WANGSIRIPITAK; MURRAY, 2009; MIGLIORE et al., 2009; LIN; WANG, 2010) although the SLAM itself may be graph-based (KUNDU et al., 2011). In fact, to our knowledge, we are the first to present a fully graph-based dynamic-object-aware monocular SLAM system. The Table 3.1 compares approaches cited in this section and positioning our proposal in the literature.

Table 3.1: Summary of the object-aware SLAM works. Abbreviations: OTM - Object Tracking Method; DOA - Dynamic-Object-Aware; MAO - Map Augmented by Objects.

Author	**Sensor**	**OTM**	**DOA**	**MAO**
Wangsiripitak and Murray (2009)	monocular	filtering	yes	no
Migliore et al. (2009)	monocular	filtering	yes	no
Lin and Wang (2010)	stereo	filtering	yes	no
Kundu et al. (2011)	monocular	filtering	yes	yes
Castle and Murray (2011)	monocular	filtering	no	no
Fioraio and Stefano (2013)	RGB-D	optimization	no	yes
Salas-Moreno et al. (2013)	RGB-D	optimization	no	yes
Ramadasan et al. (2015)	monocular	optimization	no	yes
Gálvez-López et al. (2016)	monocular	optimization	no	yes
Ours	**monocular**	**optimization**	**yes**	**yes**

4 PROPOSED METHOD

In this chapter, we describe our proposal to solve the problem of dynamic object tracking in a SLAM map and also the whole system developed to evaluate the proposed method. This system was developed over the second version of ORB-SLAM public source code. The main reasons to make this choice were its superior performance and accuracy compared to other state-of-the-art SLAM works, the availability of source code, and our previous experience with feature-based systems. It is important to say that the first version of ORB-SLAM is only for monocular cameras, while the second one also supports RGB-D and stereo cameras, but we use only its monocular module. We do not use the first version only for technical reasons, more specifically, due to its dependencies.

As well as ORB-SLAM and others PTAM-based systems, the combined problem for localization and mapping was broken into two concurrently processes. Front-end performs the tracking of the camera and objects in each video frame as faster as possible, while back-end is responsible to update the map, maintaining its consistency over time. Furthermore, in our system it is also necessary that the back-end do dynamic map initialization, i.e, it must detect objects and make its insertion on SLAM map.

The ORB-SLAM's back-end is distributed on two threads, being one for mapping and the other for loop closing and global optimization. Mapping searches for new feature correspondences between the current key-frame and the ones on its neighborhood in the co-visibility graph. It triangulates the new correspondences, adding new 3D points to the map. It removes points and redundant keyframes and also run Bundle Adjustment for the local map. Loop closing, in turn, detects and closes a loop in the map. After a loop is detected, a similarity transformation is computed to correct the entire map.

In our system, both operations were used, but a filter was added to the optimizers to make them ignore the object points since they may not obey the static map assumption. Besides that, we introduce in the loop closing thread the object detection module. We opted to incorporate a new module in an existing thread instead of creating a new one, not only due to the smaller computational cost involved but also because this thread gets idle on ORB-SLAM. Figure 4.1 provides an overview of the developed system in comparison with the original ORB-SLAM. The red/gray labels highlight our modifications and the

next sections detailed each of them.

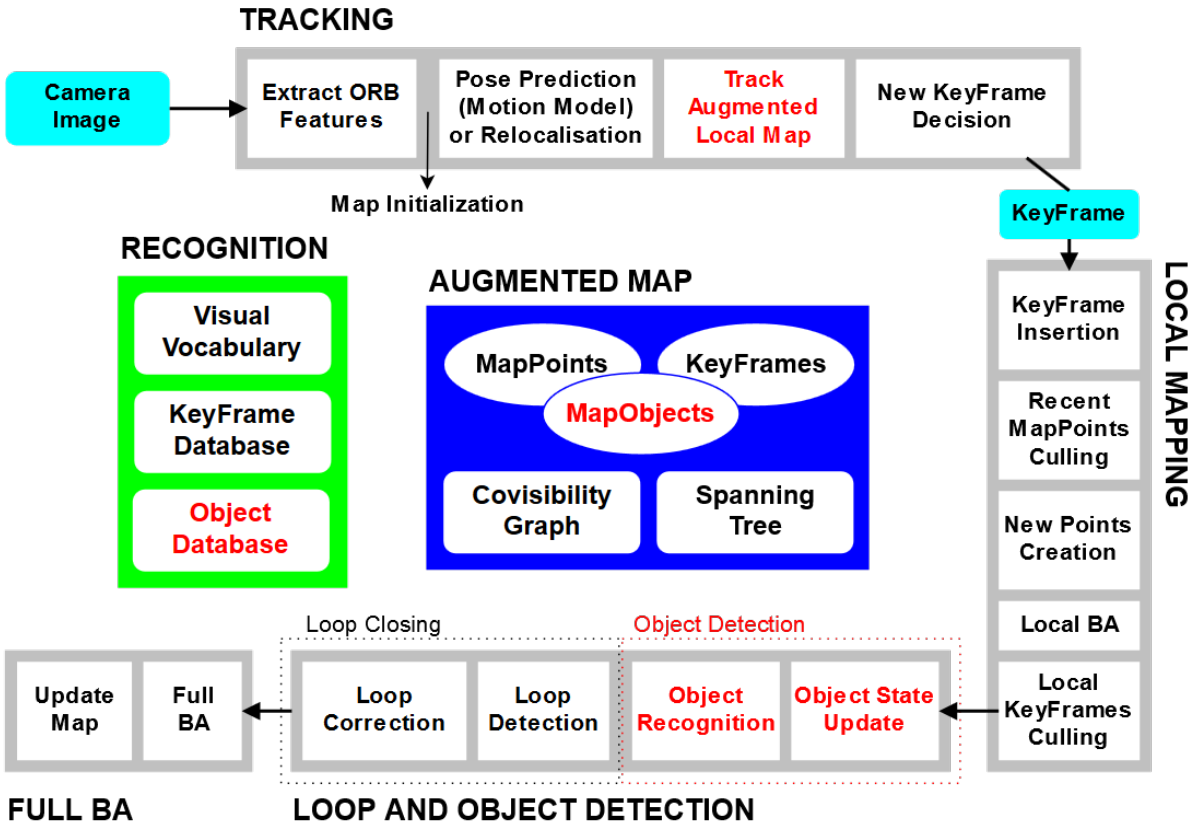


Figure 4.1: Overview of our system based on ORB-SLAM. Labels in red/gray indicates the new or modified components that our system introduced in ORB-SLAM framework. Objects are aggregate to the map; An object database stores known objects for recognition; Novel optimization framework for local map tracking; Loop Closing thread receives the Object Detection module.

4.1 SIMULTANEOUS TRACKING OF CAMERA AND OBJECTS

Our main contribution is described in this section: The 6DoF tracking of the camera and of multiple rigid objects of interest. It is performed on "Track Augmented Local Map" step illustrated by the Figure 4.1. Originally, ORB-SLAM performs a camera-only BA on this step, that is, the state vector $\mathbf{x}_i = [\tau_{i,w}]$, where $\tau_{i,w}$ is the minimal representation of the transformation from world frame w to the i-th camera frame. The error is defined as the difference between the observed image point, $\overline{\vec{x}_{i,j}}$, and the projection of its corresponding 3D point, $\vec{x}_{w,j}$, what is called re-projection error (Equation 4.1).

$$\mathbf{e}_{i,j} = \overline{\vec{x}_{i,j}} - \boldsymbol{\pi}_i(\boldsymbol{\phi}(\tau_{i,w})\ \vec{x}_{w,j}) \tag{4.1}$$

Where $\boldsymbol{\pi}_i : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ is the camera projection function obtained from the calibration. The mapping operator $\boldsymbol{\phi} : \mathbb{R}^6 \rightarrow SE(3)$ give us the corresponding transformation matrix of a minimal parametrization as described in the Section 2.5. So, we can interpret this error formula as following: First, the world 3D point $\vec{x}_{w,j}$ is transformed to the i-th camera frame by matrix multiplication; Then the resultant point is projected on the camera image and the difference with the expected point $\overline{\vec{x}_{i,j}}$ give us the error vector.

Our proposal is to add more components to the state vector $\mathbf{x}_i$ representing the 6DoF pose of each object that currently has enough matches with the camera image. We denote by $\tau_{w,k}$ the minimal representation of the pose of the k-th object in the world frame. So, our state vector is $\mathbf{x}_i = [\tau_{i,w}, \tau_{w,1}, \cdots, \tau_{w,K}]$. We use object local points to support the estimation of the object pose, so the object structure must be known a priori. We define a new re-projection error formulation specially for object point matches, $\vec{x}_{k,j_k} \rightarrow \overline{\vec{x}_{i,j_k}}$. This formulation is described by Equation 4.2.

$$\mathbf{e}_{i,j_k} = \overline{\vec{x}_{i,j_k}} - \boldsymbol{\pi}_i(\boldsymbol{\phi}(\tau_{i,w})\ \boldsymbol{\phi}(\tau_{w,k})\ \vec{x}_{k,j_k})) \tag{4.2}$$

The idea behind this formula is very similar to the previous one, but we first transform a point in the object local space to the world frame and then to the camera and then to the image domain. Our optimization framework is then defined by Equation 4.3.

$$\hat{\mathbf{x}}_i = \arg_{\mathbf{x}_i} \min \sum_{j}^{N} v_{i,j}\ \boldsymbol{\rho}_H(\mathbf{e}_{i,j}^T\ \boldsymbol{\Omega}_{i,j}^{-1}\ \mathbf{e}_{i,j}) + \sum_{k}^{K} \sum_{j_k}^{N_k} v_{i,j_k}\ \boldsymbol{\rho}_H(\mathbf{e}_{i,j_k}^T\ \boldsymbol{\Omega}_{i,j_k}^{-1}\ \mathbf{e}_{i,j_k}) \tag{4.3}$$

Where $v_{a,b}$ is a binary value to indicate if the correspondence $a \leftrightarrow b$ exist. The covariance matrix, $\boldsymbol{\Omega}_{a,b}$, models the correspondence uncertainty that it is supposed uncorrelated and follow a Gaussian distribution, thus $\boldsymbol{\Omega}_{a,b} = \sigma_{a,b}^2 \boldsymbol{I}_{2\times 2}$. The standard deviation $\sigma_{a,b}$ is defined to be proportional to the scale factor of the pyramid level where the feature was extracted, because the feature position (pixel coordinate) is less accurate in higher scales (camera distance). Finally, $\boldsymbol{\rho}_H$ is the Huber loss function defined by Equation 4.4. The threshold δ is obtained from the χ^2 distribution at 95% confidence and $f = 3$ degrees of

freedom, which in this case gives $\delta^2 = \chi^2_{0.05}(3) = 7.815$.

$$\boldsymbol{\rho}_H(\varepsilon) = \begin{cases} \varepsilon & : \varepsilon \leq \delta^2 \\ 2\delta\sqrt{\varepsilon} - \delta^2 & : \varepsilon > \delta^2 \end{cases} \tag{4.4}$$

As an optimization process, it is necessary to provide the initial values of the state vector components $\mathbf{x}_i$. Therefore, an initialization process for both static and dynamic maps must be performed. ORB-SLAM provides a robust and automatic method to initialize its static map. On our system, this process is exactly the same. Dynamic maps are initialized as they are detected.

4.2 FRONT-END OVERVIEW

In this section we describe the operation of the *Tracking* thread (refer to Figure 4.1).

After the map was initialized, ORB descriptors are extracted from the image for each new video frame. Depending on the tracking state, an initial pose for the camera is computed. If tracking is currently lost, the camera re-localization process takes place. Otherwise, a pose prediction step based on a motion model is executed. Having a rough estimation for the camera pose, the local map is updated. Then, a projection-based matching algorithm searches by correspondences between local map and the current frame. All these steps are identical to the ones performed by ORB-SLAM, with the difference that the local map may be augmented by object points if any object was already detected by the back-end. In this way, object correspondences in the camera image are automatically found.

After the matching step, our method takes place. Features matched with static (background) points add the constraint defined by Equation 4.1. On the other hand, features matched with object points add the constraint defined by Equation 4.2. The predicted camera pose is used to initialize the correspondent component of vector $\mathbf{p}_i$. The current pose of each object on the map initializes the other components. The graph structure for our proposed optimization framework (Figure 4.2) is created using the g^2o library and the optimization step begins. Originally, in this step, ORB-SLAM executes 4 optimizations. After each of them, a correspondence is classified as inlier/outlier. At the next optimization, outliers are not included, but at the end they may be classified as inliers again. On our system this process is similar, but we execute 2 more optimizations. As

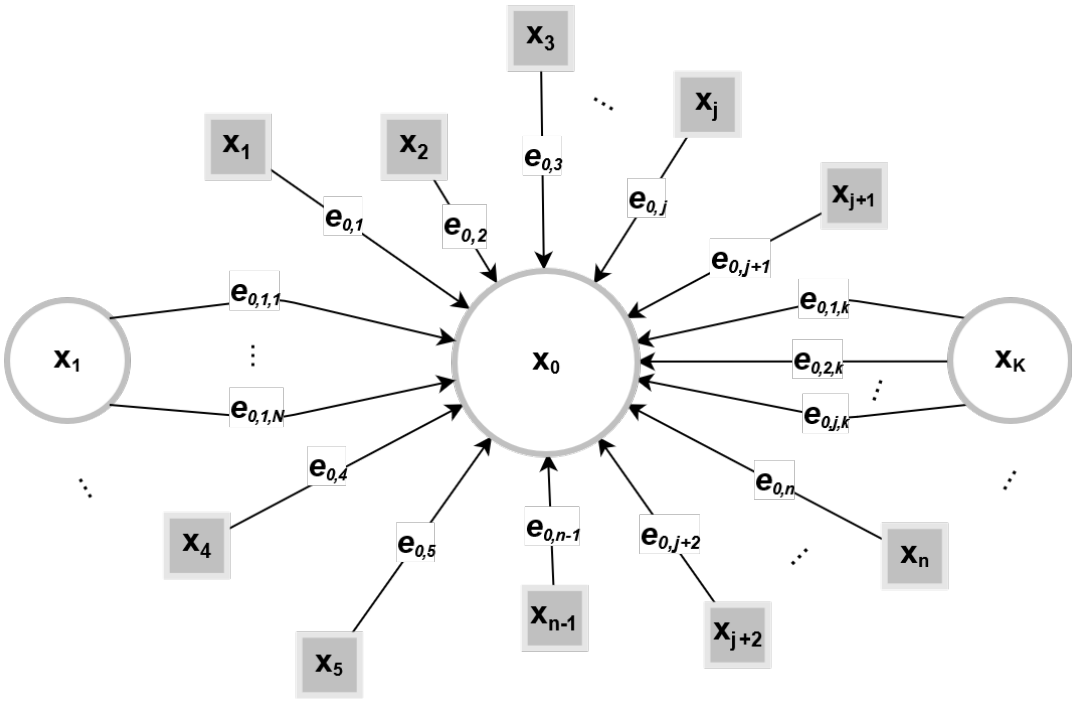


Figure 4.2: Graph structure to our optimization framework. Circles vertices are components of the state vector $\mathbf{x}_i$, while squares represents fixed vertices that just take part of the constraint, but they are not updated.

the camera already has a pose prediction for the current frame, we first perform these 2 extra optimizations over object-only. After all the optimizations have ended, we check the number of inliers for each object to define its visibility state. For this, an absolute threshold $N_i = 20$ is used. This value was chosen based on Ufkes and Fiala (2013) work and also by observing our results.

Before we update the objects in the map, the optimized transformations are filtered for a motion threshold. The idea is to reject irrelevant object motions. The reason is to minimize small variations that occur even when the object is stationary due to the matching uncertainty. The linear and rotational motions are considered separately. The first one is considered valid when the squared L2 distance between the translation vectors $\mathbf{t}_b, \mathbf{t}_b \in \mathbb{R}^3$ (Equation 4.5) is greater than the linear threshold τ_l defined as the half of the variance associated to map-point position, i.e., $\tau_l = 0.5\sigma_m^2$ where $\sigma_m = 0.01$.

$$\boldsymbol{\delta_l}(\mathbf{t}_a, \mathbf{t}_b) = \|\mathbf{t}_a - \mathbf{t}_b\|^2 \tag{4.5}$$

The metric for the rotational motion is deduced from the angle θ between two rotation quaternions. Since a quaternion is a unitary vector of the $\mathbb{R}^4$ space, the cosine of the angle

between the quaternions $\mathbf{q}_a$ and $\mathbf{q}_b$ are obtained from a simple dot product, $\cos\theta = \mathbf{q}_a \cdot \mathbf{q}_b$. So, from the identity $\cos^2\theta + \sin^2\theta = 1$ we derive the metric showed by Equation 4.6.

$$\boldsymbol{\delta_r}(\mathbf{q}_a, \mathbf{q}_b) \equiv \sin^2\theta = 1 - (\mathbf{q}_a \cdot \mathbf{q}_b)^2 \tag{4.6}$$

If there is any displacement between the rotations ($\theta = 0$) then $\boldsymbol{\delta_r} = 0$. On the other hand, if the displacement is maximum ($\theta = 180°$) then $\boldsymbol{\delta_r} = 1$. So, we define the rotational threshold as $\tau_r = \sin^2(0.5°)$. Note that the quaternion representation of a rotation uses the half of the rotation angle. Therefore, 0.5 degree between two quaternions is equivalent to 1 degree rotation.

The update of the optimized data will occur only when the following conditions are satisfied: $\boldsymbol{\delta_l}(\mathbf{t}_{w,k}, \hat{\mathbf{t}}_{w,k}) \leq \tau_l$ and $\boldsymbol{\delta_r}(\mathbf{q}_{w,k}, \hat{\mathbf{q}}_{w,k}) \leq \tau_r$.

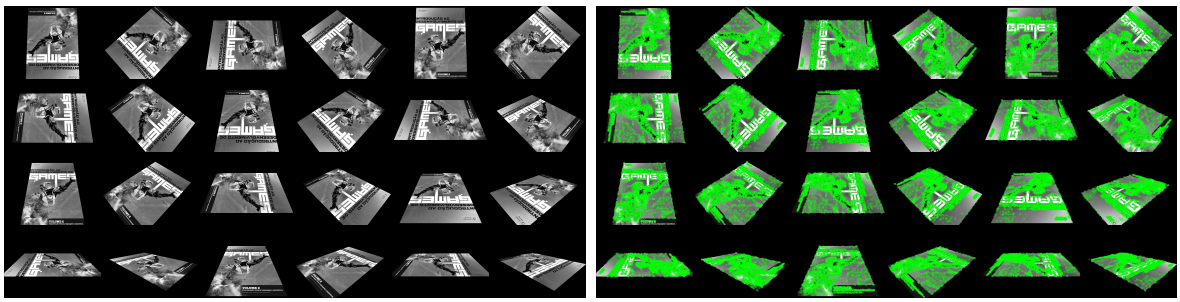
Finally, the last step of the front-end is to define when a video frame becomes a key-frame. We kept all conditions defined by ORB-SLAM.

4.3 OBJECT DATABASE

We use an object database for object recognition task similar to the one used by Gálvez-López et al. (2016). The database stores 3D reconstructed points and an ORB descriptor set extracted from several images of the object with different perspectives. The object features are transformed into a Bag of Words (BoW) representation, with an inverted index and a direct index. This representation was implemented using the open-source library proposed by Gálvez-López and Tardos (2012). The difference is on the manner which the object reconstruction was made. GÁLVEZ-LÓPEZ et al. perform a real reconstruction using third-party software to do it. In our work, in order to make this process easier for the user, we opted to perform a synthetic object reconstruction.

Our idea was to compute the 3D points from only one digital image of the object. For planar objects, like books, magazines, and photos in general, this approach is straightforward. Just retro-project the image pixels over any plane in tridimensional space. Since object dimensions are known or easily measured, it is possible to reconstruct the object for a specific metric system. It is important because the SLAM map scale is unknown, but once we have known the real measurements of the object and it is present on the map, we can estimate the real scale of the map.

(a) Retro-projected pixels (b) Virtual cameras (e) Final reconstruction



(c) Synthetic images (d) Extracted features

Figure 4.3: Object registration steps. (a) Pixels retro-projected to 3D space; (b) 24 virtual cameras are positioned over the surface of a semi-sphere around the object in order to cover several perspectives; (c) Image viewed by each virtual camera; (d) Extracted ORB features for each image; (e) 3D remaining points after filtering. In this example, the input image had 1,543,500 pixels and the database received 21,418 3D points and 53,987 ORB descriptors.

However, the feature extraction performed in a unique image limits the perspective in which the object can be detected on the video stream. So, after we compute 3D points, a predefined number of virtual cameras are automatically positioned around the entire object to cover its surface in several perspectives. For each virtual camera, we produce a synthetic image of the object from the projection of each reconstructed point. ORB feature extraction is performed on each image with 8 pyramid levels and a scale factor of 1.2, which is the default parameters of the original implementation of the technique.

Finally, we filter out points that were not associated with, at least, two ORB descriptors. Figure 4.3 illustrates each step described in this section.

4.4 OBJECT DETECTION

Tasks related to the object tracking which involves the back-end will be described from this section forward. The objective of this section it to describe how an object is recognized on the video stream, how we compute its pose, how we estimate the real scale of the map, and

how to fuse static and dynamic maps. Each of these topics is addressed in the following subsections. All process described here refer to the step named "Object Recognition" in the Figure 4.1.

Firstly, it is important to remember that the back-end process related to object tracking was incorporated into the loop closing thread because, as mentioned before, this thread was getting idle for a considerable period of time. All tasks performed by this thread operates over a key-frame already processed by the mapping. Thus, the keyframes were already matched against its neighbor keyframes in the co-visibility graph and new 3D points were triangulated from this matches. Moreover, the key-frame pose was potentially optimized by Bundle Adjustment over the local map, so we may consider this pose is reliable.

Another important note is that the developed system was planned to be applied for Augmented Reality applications. In this context, the target objects are usually known. Therefore, in order to simplify the recognition algorithm, such objects must be informed to the system before initialization. They are retrieved from the object database and loaded into the memory. Two lists store target objects, one saves the *not-found* objects and the other saves the objects already detected.

4.4.1 OBJECT RECOGNITION

Our recognition algorithm works in a similar manner as the one proposed by Gálvez-López et al. (2016), but with some simplifications. The first one is not partitioning the image features on regions of interest before querying the database. For each new key-frame, the database is queried passing all words present on the image. The query returns the K objects with better scores. The score indicates how many words of the object is present in the input image. Therefore, it can be interpreted as the possibility of the object to be visible in the current key-frame. The value of K depends on how many objects are stored by the database. GÁLVEZ-LÓPEZ et al. built a base composed up to 500 objects and defined $K = 10$. They match and estimate the pose for each returned object in order to approve the detection or not. On the other hand, we built a base with only 5 planar objects and defined $K = 1$. Then, we run the matching algorithm for the returned object if it is in the *not-found* list.

The matching algorithm is performed in the usual way, comparing the pairs of de-

scriptors of the object and key-frame. Only key-frame features that have already been triangulated to the map are compared. In this way, since an object feature has a corresponding 3D point, we can get 3D-to-3D correspondences by means of a 2D matching algorithm. Both key-frame and object features already have their BoW representation. So, we use the direct index to decrease the search space. Only descriptors whose corresponding word is under the same node at a pre-defined level on the vocabulary tree are compared. We follow the ORB-SLAM that uses a similar method to support relocalization and loop closing tasks and we define the level for the second one (top-down) in a vocabulary tree of 6 levels. When an empirical good number of matches ($N_m = 30$) are found, the algorithm pass to the next step.

4.4.2 POSE ESTIMATION

We try to estimate the object's pose relative to the camera's space by solving a Perspective-n-Point (PnP) problem. 3D object points are associated with its corresponding 2D image points. A RANSAC scheme (FISCHLER; BOLLES, 1987) run the algorithm proposed by Lepetit et al. (2008). RANSAC executes until the maximum number of iterations (100) is performed or the re-projection error is less than the threshold $\tau_e = \sqrt{\chi^2_{0.05}(2)} = \sqrt{5.991}$. If the estimated pose, $\boldsymbol{T_{c,o}}$, is supported by a minimum number of inliers (N_i) then the object is considered visible for the current key-frame. The next steps are to estimate the real scale of the map, to compute the object's pose relative to the world frame and, finally, to fuse the maps.

4.4.3 AUGMENTED MAP

In order to insert the object on the map, we need to know the object's pose relative to the world frame $\boldsymbol{T}_{w,o}$. Let $\boldsymbol{R}_{i,j}$ and $\boldsymbol{t}_{i,j}$ be the rotation and translation part of the transformation matrix $\boldsymbol{T}_{i,j}$, respectively. We roughly compute the matrices $\boldsymbol{R}_{w,o}$ and $\boldsymbol{t}_{w,o}$ as defined by Equations 4.7 and 4.8, respectively.

$$\boldsymbol{R}_{w,o} = \boldsymbol{R}_{c,w}^{T} \times \boldsymbol{R}_{c,o} \tag{4.7}$$

$$\boldsymbol{t}_{w,o} = \boldsymbol{R}_{c,w}^{T} \times (\boldsymbol{t}_{c,o} - \boldsymbol{t}_{c,w}) \tag{4.8}$$

This computation assumes the scale between world and object space as 1. In fact, the scale is unknown and we want to estimate it, but for now this is an acceptable approximation since the object is measured in meters. With a rough estimation of the pose $\boldsymbol{T}_{w,o}$, we search for new point matches between the object and the current key-frame by projecting each object point (transformed by $\boldsymbol{T}_{w,o}$) on the image and by searching the best match among the descriptors inside a window centered at the projection. Given all 3D-to-3D correspondences, we optimize the object state in the world frame by minimizing an alignment error function, described by Equation 4.9.

$$\hat{\tau}_{w,o} = \arg\min_{\tau_{w,o}} \sum_{\langle \vec{x}_{o,j}, \vec{x}_{w,j} \rangle} \boldsymbol{\rho}_H(\mathbf{a}_j^T \ \boldsymbol{\Omega}_{\boldsymbol{A}}^{-1} \ \mathbf{a}_j) \tag{4.9}$$

The object state is parameterized as a 7-vector $\tau_{w,o} = [t_x, t_y, t_z, q_x, q_y, q_z, s]$, where t_x, t_y, t_z are the object's world coordinates; q_x, q_y, q_z are the imaginary-part of the rotation quaternion such that the real-part can be recovered from $q_w = \sqrt{1 - q_x^2 - q_y^2 - q_z^2}$; and s is the scale between object and world frames. The alignment errors $\mathbf{a}_j$ defined by Equation 4.10 are supposed uncorrelated and follow a Gaussian distribution. Adopting the standard deviation of the point triangulation $\sigma = 0.01$, we define the covariance matrix as $\boldsymbol{\Omega}_{\boldsymbol{A}} = \sigma^2 I_{3x3}$.

$$\mathbf{a}_j = \vec{x}_{w,j} - (\boldsymbol{\phi}(\tau_{w,o}) \ \vec{x}_{o,j}) \tag{4.10}$$

Finally, we can fuse the maps. For this, we take each corresponding pair $\langle \vec{x}_{w,j}, \vec{x}_{o,j} \rangle$. The meta-data of the map-point $\vec{x}_{w,j}$ is copied to a new map-point created for $\vec{x}_{o,j}$. The coordinates of this new map-point is computed as $\boldsymbol{\phi}(\hat{\tau}_{w,o}) \ \vec{x}_{o,j}$. All observations to $\vec{x}_{w,j}$ is linked to the new map-point. Then we remove $\vec{x}_{w,j}$ from the SLAM map and add the map-object instance to the augmented map. Figure 4.4 shows an example of the augmented map with all objects we register in our database.

4.5 OBJECT STATE UPDATE

After detecting an object, it is removed from the *not-found* list and added to *found* list. For the objects inside the last one, the back-end updates its visibility state. The state machine diagram is illustrated in Figure 4.5 and we will detail it in the following

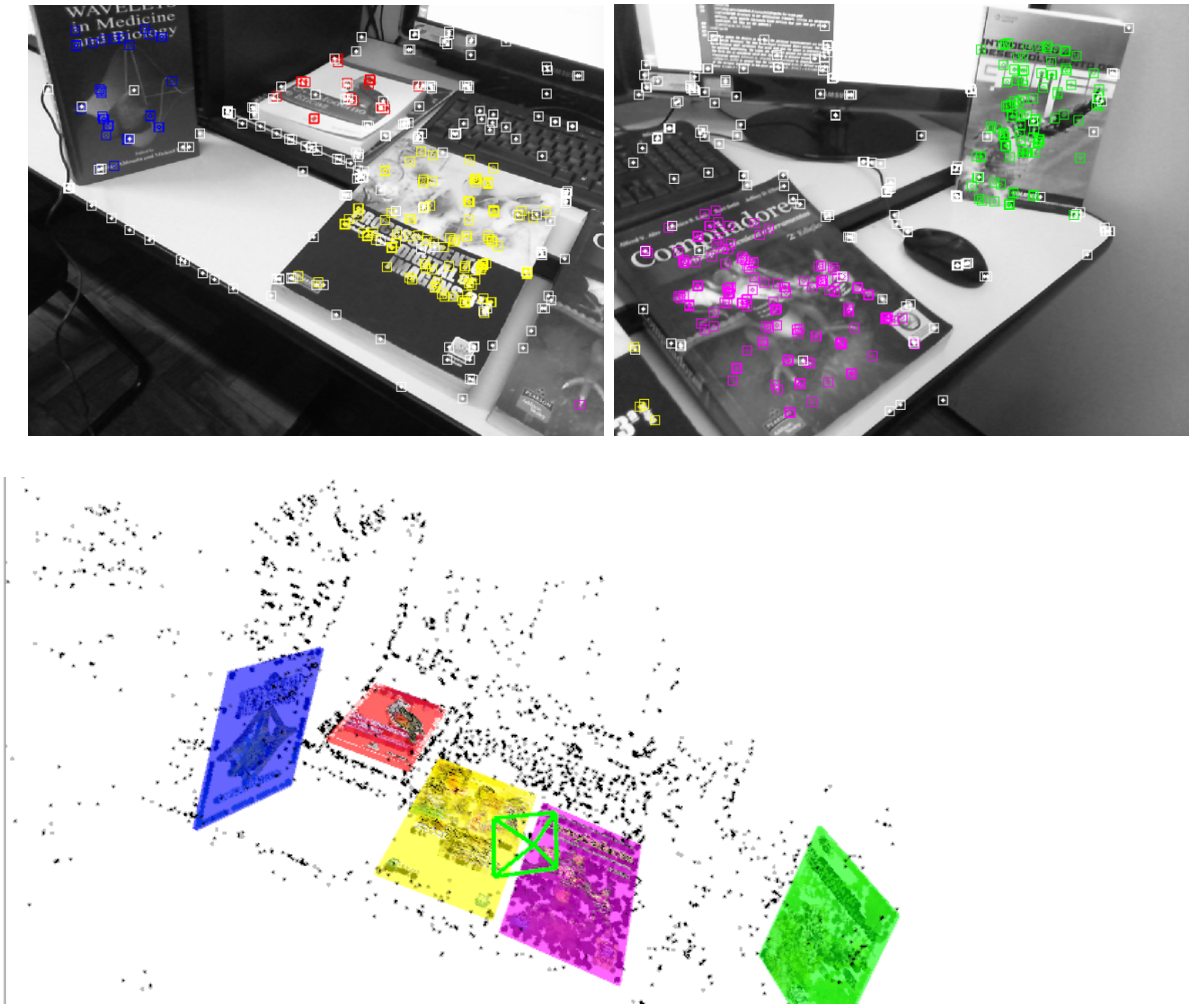


Figure 4.4: Multiple object recognition and mapping example. Our system is able to maintain the tracking of multiple objects simultaneously in real time. After an object is mapped by the back-end, the front-end just do projection-based matching to find correspondences between the camera image and the augmented map. Our proposed framework jointly optimizes the camera and visible objects. On the figures above, we associate a specific color to each object in our database.

paragraphs.

From the moment an object registered in the database is defined as a target of the system, it is retrieved from the database and an instance of a map-object is allocated and added to the *not-found* list. All objects inside this list are put on *not-found* state. While there is an object in this list, the back-end runs the recognition algorithm. When an object is detected, it changes its state to the one named as *schrödinger*.

The name of this state is a reference to the famous Schrödinger's Cat experiment (SCHRÖDINGER, 1935). Who knows this experiment, at this point, may not understand the analogy here since the transition for this state takes place when the object is detected.

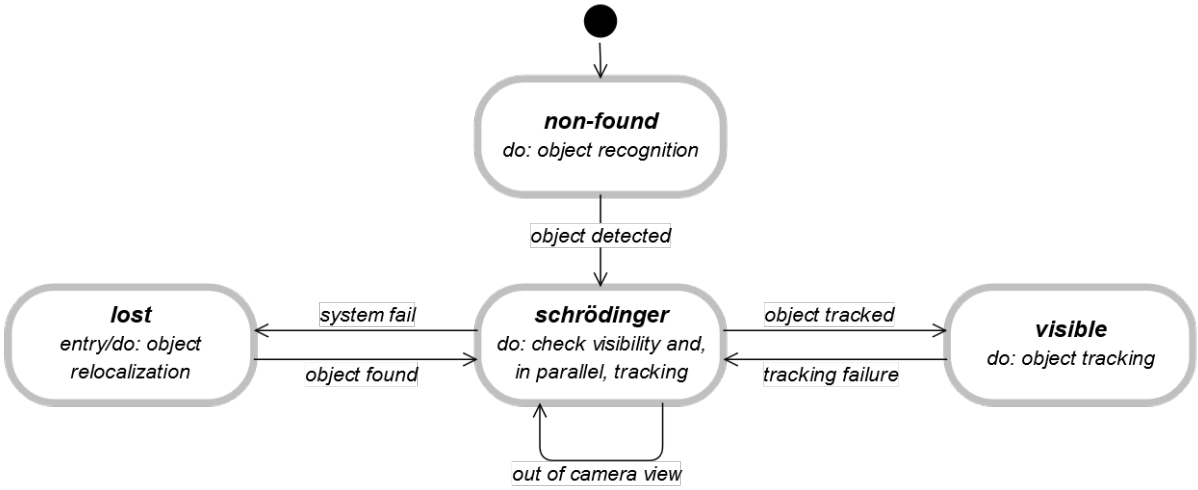


Figure 4.5: Object visibility state machine diagram.

However, the object was detected for a key-frame in the back-end thread, that is, for an instant in the past. The front-end is probably about 20 to 30 frames ahead. So we do not actually know if the object is visible or lost for the current frame. We will only know after "looking at what is inside the box".

While the object is on *schrödinger* state both front-end and back-end can change its state. If front-end can locate the object, the state is changed to *visible* and only a potential future failure in the tracking will lead to another state change, back to the *schrödinger* state. The back-end, in turn, will try to check if the object is not visible due to a system failure or simply because it is out of the camera's field of view (Section 4.5.1). It is only possible to discriminate these two cases because the camera localization is independent of the object tracking. If the system realizes the object is out of the camera's view then the object remains on *schrödinger* state. Otherwise, there was a system failure, so the object state changes to *lost*.

As soon as the object enters the *lost* state, the re-localization algorithm (Section 4.5.2) is invoked for the same key-frame that caused the transition in order to corroborate the visibility checking result. If the re-localization is successful then the object immediately changes back to *schrödinger* state. Otherwise, *lost* state is kept until the object is re-localized in some future key-frame.

While the object is on *visible* state, back-end only tries to find new 3D correspondences by projection-based matching. The new matched points are fused to map just as done after detecting. In this way, the front-end will dispose of more object points on the local

map to support tracking.

4.5.1 CHECKING OBJECT VISIBILITY

In order to verify if an object is not visible on the front-end due a system failure or because it is out the camera's field of view, we take the 8 vertices of the object's bounding box and we project them on the image. Then, we find the minimum bounding rectangle (MBR) for the projected points and we compute the intersection between it and the image. Finally, we compute the ratio between the intersection area and the whole image area. A threshold of value 0.07 was empirically defined. This implies the object will be treated as out of the camera's view if the occupied area represents, at least, 7% of the total image area. The bigger an object is, the farther it can stay from the camera without harming the tracking. One interesting thing about this technique is that it treats objects of different sizes in the same way, simplifying the state definition to a single threshold.

4.5.2 OBJECT RE-LOCALIZATION

An object on the *lost* state means its estimated world pose is probably incorrect. So we can not use this pose to perform the projection-based matching. That is why we appeal to the BoW-based matching algorithm, the same used right after the object be returned from the database querying. If the number of matches is at least N_m then we do a similarity optimization by minimizing the alignment error (Equation 4.9). If the optimized transformation is supported by the defined minimum number of inliers N_i, then the re-localization is already considered successful, but before terminate it, we perform a new round of matching, now by projection, followed by a second optimization with the new pairs. In all other cases, the re-localization fails.

5 EXPERIMENTAL RESULTS

In this chapter, we present the results obtained with our system. As we did not find in the literature works related to object-aware SLAM with available source code and the evaluated works did not present a standard for method evaluation of the techniques, we did not compare our object tracking system with another in literature. Therefore, this chapter can be seen as a proof of concept, our goal is to show the feasibility of the proposed framework. For this purpose, we evaluated the camera localization quantitative accuracy in relation to one of the main datasets. We developed an metric for the tracking of mapped objects. Finally, we demonstrate potential applications in Augmented Reality and carry out a qualitative analysis of the visual results.

5.1 DATASETS

To evaluate our system, we searched datasets in the literature. For the SLAM-based camera localization problem, we found a variety of excellent datasets with ground-truth estimated by modern motion capture systems, several sequences to evaluate different problem aspects, and objective metrics for comparison. We chose the dataset proposed by Sturm et al. (2012), which is more oriented to indoor applications and is used by many works in recent literature.

However, we did not find any dataset compatible with our SLAM-based object tracking system, i.e., for indoor textured objects with provided 6DoF ground-truth. The works in the literature which integrate objects into SLAM systems usually quantitatively evaluate camera localization results, but mapped object position is only qualitatively evaluated. Nonetheless, in order to numerically evaluate mapped object positioning, we developed an metric (Section 5.2.2) based on other works in Augmented Reality and created a small dataset composed of 6 quick image sequences, each lasting from 30 to 40 seconds. Each sequence was planned to do a specific test, with common movements in a AR application.

- *camera_translation*: Marker is fixed and the camera translates over the x- and z-axis;
- *camera_half_sphere*: Marker is fixed and the camera translates around the marker over a semi-sphere;

- *marker_translation*: Camera is fixed and the marker moves over its x- and y-axis;
- *marker_rotation_360*: Camera is fixed and marker performs an in-plane rotation around its z-axis;
- *marker_inclination_zoom*: Camera is fixed and marker performs an out-plane rotation followed by a translation toward the camera.
- *free_motion*: Free and simultaneous movement of both marker and camera.

In order to determine each sequence's ground-truth, we employed ARToolKit[1], an Augmented Reality library, one of the first libraries for this purpose presented in the literature, which still receives community updates[2]. ARToolKit's strongest point is fiducial markers tracking. Therefore, we created a textured plane object[3] with a fiducial marker in its center (Figure 5.1). The marker center was defined as the coordinate system origin, in this way both ARToolKit and our system compute the transformation of the object relative to camera on the same coordinate system.

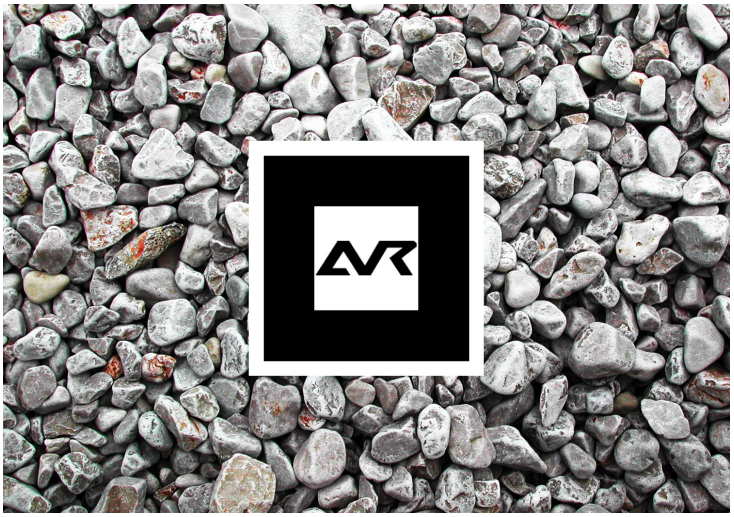


Figure 5.1: Marker object used in our sequences. The fiducial marker in the center has side $s = 70mm$, while the entire texture has dimension 24,7cm x 17,3cm.

Since our system required an initialization process, to synchronize it with the ARToolKit, we recorded for a few seconds a part of the scenario with multiple features where the marker object does not appear. Then, we move the camera until we frame the object correctly in the image and proceed with the planned test.

[1] Actually, we employed **artoolkitX** that is the currently official name of the open-source version, but for convenience we still refer it by its original name **ARToolKit**.

[2] artoolkitX repository: https://github.com/artoolkitx/artoolkitx [Online; 09-07-2018]

[3] We used the texture of one Vuforia library patterns: https://library.vuforia.com/ [Online; 09-07-2018]

5.2 EVALUATION METRICS

To evaluate the camera trajectory, we compute the Absolute Trajectory Error (ATE), whereas to evaluate the object trajectory on the map in each frame we compute the Cube Displacement Error (CDE). We detail each metric below.

5.2.1 ABSOLUTE TRAJECTORY ERROR (ATE)

The Absolute Trajectory Error directly evaluates the difference between each trajectory pose by defining a frame error as the difference between estimated and ground-truth translation vectors. Only the translation is considered because rotational errors usually lead to incorrect translations, which ends up being captured indirectly by ATE.

Still, the trajectories need to be aligned before computing error, considering that each trajectory referential is arbitrary. To do that, the algorithm proposed by Umeyama (1991) was used to find the similarity transformation $\boldsymbol{S}$ which maps the points of the estimated path $E_{1:n}$ to the ground-truth path points $G_{1:n}$. Thus, we compute the ATE for a frame i as defined in Equation 5.1.

$$ATE[\ i\ ] := trans(G_i^{-1} \times \boldsymbol{S} \times E_i) \tag{5.1}$$

In which the function $trans(T)$ returns the translation corresponding to the input transformation T. As proposed by Sturm et al. (2012), we calculated ATE's root-mean-square error (RMSE) at all frames (Equation 5.2).

$$RMSE[ATE] := \sqrt{\frac{1}{n}\sum_{i=1}^{n} \|ATE[\ i\ ]\|^2} \tag{5.2}$$

5.2.2 CUBE DISPLACEMENT ERROR (CDE)

Cube displacement error is our adaptation of the metric proposed by Siltanen et al. (2007) to evaluate calibration quality in the fiducial markers arbitrary field. To our knowledge, we are the first to use this metric to measure the accuracy of object tracking system. For each frame, the error is defined as the average displacement of the 8 vertices of a cube projected on the object's frame. In other words, let $\boldsymbol{T}_{i,o} = \boldsymbol{T}_{i,w} \times \boldsymbol{T}_{w,o}$ be an object estimate pose relative to the current camera i, let $\boldsymbol{T}^*_{i,o}$ be that transformation's ground-

truth and let $\mathfrak{V}$ be the set of vertices of a cube centered in the system origin with side length s. We define CDE for frame i as shown in Equation 5.3.

$$CDE[\ i\ ] := \frac{1}{8} \sum_{v\ \in\ \mathfrak{V}} ||\boldsymbol{T}^{*}_{i,o} v - \boldsymbol{T}_{i,o} v|| \tag{5.3}$$

Then, we compute CDE RMSE, arithmetic mean and standard deviation for all frames in which the object was considered visible. In this way, we can indirectly evaluate the rigid transformation estimated quality in each frame and also the scale computed in the detection. The cube edge length s is defined for the actual dimensions, in meters, of the fiducial marker that makes up the object of interest.

5.3 QUANTITATIVE RESULTS

In this section, we present and discuss numerical results on the datasets described in Section 5.1. To evaluate camera localization accuracy we use the metric described in Section 5.2.1, while the marker positioning accuracy results are done according to the metric described in Section 5.2.2.

5.3.1 CAMERA LOCALIZATION

In order to evaluate the camera localization accuracy, we run our system 5 times and take the median of the ATE RMSE errors. We also executed the original ORB-SLAM2 (MUR-ARTAL; TARDÓS, 2017), the base of our work, under the same conditions of our developed system. Table 5.1 shows our results for several sequences of the TUM RGB-D dataset. Our objective with this comparison is just checking whether our modifications over ORB-SLAM2 could impact in some way the performance of SLAM. However, as we can see the accuracy results are statistically equal.

5.3.2 OBJECT TRACKING

Just like the evaluation of the camera localization, we also run our system 5 times to take the median of the errors to evaluate the object tracking proposal. We assume the trajectory obtained by ARToolKit as the ground-truth trajectory and we compute the CDE error of our system. Table 5.2 exposes the average results for each of our 6 recorded

Table 5.1: Absolute Trajectory RMSE (cm) for several sequences of the TUM dataset

SEQUENCE	OURS	ORBSLAM2
fr1_xyz	0.889	1.016
fr1_desk	1.456	1.584
fr2_xyz	0.228	0.241
fr2_360_kidnap	4.552	5.380
fr2_desk	0.789	0.632
fr2_desk_with_person	0.785	0.767
fr3_long_office	1.326	1.955
fr3_nostr_tex_near_loop	1.309	1.450
fr3_str_tex_far	0.943	1.064
fr3_str_tex_near	1.450	1.321
fr3_sit_xyz	0.965	1.013
fr3_sit_halfsphere	1.858	1.781
fr3_walk_halfsphere	1.751	1.788

sequences and Figure 5.3 shows the evolution of the CDE over time.

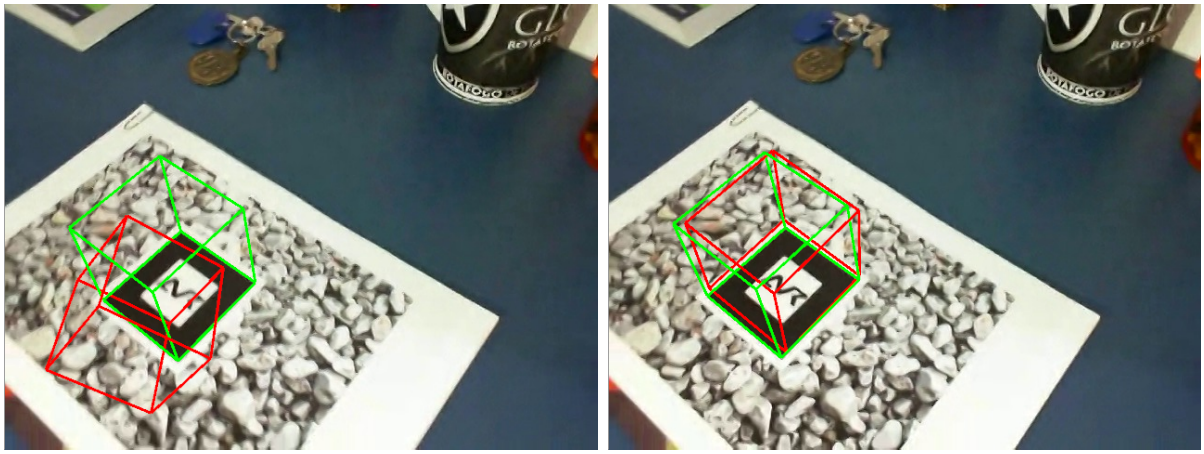
Before we can interpret the results for our proposed metric, we need to take some notes. First, CDE encapsulates both object tracking and camera localization errors, since the transformation $\boldsymbol{T}_{c,o}$ used to position the cube over the object's referential system, is the composition of the model ($\boldsymbol{T}_{o,w}$) and the view ($\boldsymbol{T}_{c,w}$) transformations. Second, although we assume ARToolKit trajectory as the ground-truth, the fact is that ARToolKit also provides just an estimation for the object's transformation. So, what we really evaluate here is how close our system is from the ARToolKit results.

Table 5.2: Cube Displacement Error (cm) in each of our recorded sequences for a projected cube of 70mm side aligned with the marker's center. We also compute the proportional error with the marker's size (value in parenthesis).

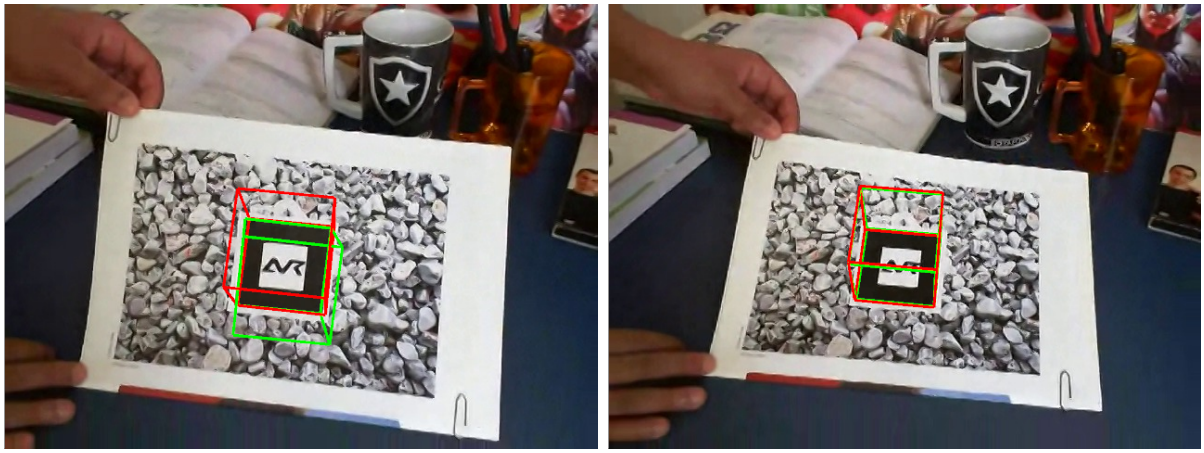
SEQUENCE	RMSE (%)	MEAN (%)	STD (%)	MIN	MAX
camera_translation	1.726 (25)	1.534 (22)	0.790 (11)	0.144	4.529
camera_half_sphere	2.769 (40)	2.417 (35)	1.352 (19)	0.250	9.298
marker_translation	1.459 (21)	1.317 (19)	0.627 (09)	0.144	4.917
marker_rotation_360	2.407 (25)	2.319 (22)	0.645 (11)	0.625	9.306
marker_inclin_zoom	2.455 (35)	2.132 (30)	1.217 (17)	0.214	8.112
free_motion	2.224 (32)	2.057 (29)	0.847 (12)	0.349	4.725

An interesting result we can note from Table 5.2 is that the maximum error in 3 sequences is clearly distorted, surpassing 100% and, therefore, totally misaligned with the marker. If we look at the graphs in Figure 5.3 we can see that this peak occurs in the first frame in which the object is visible. Thus, we can associate this result to a problem in the similarity optimizer, probably caused by mismatches. An evidence of this is illustrated

in Figure 5.2a (left), where we can see a blur effect in the first frame that the object is detected. The same figure, on the right, shows the CDE for the immediately subsequent frame, where we see the error was already corrected, giving us a demonstration of the feasibility of our front-end optimization framework.



(a) *camera_halfsphere*



(b) *free_motion*

Figure 5.2: CDE visualization for some sequences. (a) Two subsequent frames. Left frame shows a detection error (note the blur on it), it is the first frame right after the target be recognized, but as we see on the right frame, the error is immediately corrected by front-end optimization. (b) Two close frames (13 frames separates them) with the highest error difference. We choose this example to illustrate the perspective ambiguity. Is the "ground-truth" cube on the left image the truly right answer?

The CDE graphs for the sequences *camera_half_sphere* and *marker_inclination_zoom* (Figures 5.3b and 5.3e, respectively) clearly shows the effect as we distance the camera from the object. In the *camera_half_sphere* sequence, when we frame the marker, we distance the camera from it as we try to perform the rotation around it. The result, as we see in the graph, is that the error increases as we move around it. Before the end, we

approach the object again and the error decrease. In the sequence *marker_inclination_-zoom* we have the opposite situation. We start by rotating the marker around the x-axis and then we translate the marker towards the camera, decreasing the error. The question is whether this effect is most related to our system or to ARToolKit since this is a well-known problem in this library.

The graphs for the sequences *marker_translation* (Figure 5.3c) and *marker_rotation_-360* (Figure 5.3d) illustrate a problem we can identify by using the system. The translation movement is more challenging for our framework than rotation one. The interesting is that the CDE captures it. We can note that the graph for the rotation movement is more constant while the graph for translation has many peaks and valleys. When we watch the video, we note the valleys appear when the movement stops, and the peaks appear during the motion. A possible cause to this is probably related to the projection-based matching. A translation motion distances object points from its expected window more than a pure rotation motion that keeps the object's center in the same place. So fewer matches may be found or more outliers may be introduced, degrading the accuracy of the optimization.

5.4 QUALITATIVE RESULTS

We present in this section some Augmented Reality applications that are possible to create with our developed system and we discuss about the visual results.

5.4.1 TOTAL OCCLUSION

The main advantage of using SLAM-based systems to build Augmented Reality applications is the possibility to keep the projection of a virtual object even when the marker object is entirely out of the camera view. This is only possible because the camera localization is independent of the object that marks the projection referential. Our proposal still brings another advantage: The marker object can move while visible. Other object-aware SLAM works assume a static map and so the referential is kept fixed. Figure 5.4 shows some frames for the execution of the **WhiteLightingAR** application we developed.

5.4.2 VIRTUAL OBJECT OCCLUSION

An application that is facilitated by object-aware SLAM is the occlusion of a virtual object by a real one. Since the real object structure is known, we can generate its mesh and then draw it with maximum transparency on the same place as the object is. By enabling the depth buffer processing on the graphics API, it automatically computes the occluded faces by the object's mesh on the virtual projection and removes them from the scene, causing the occlusion effect.

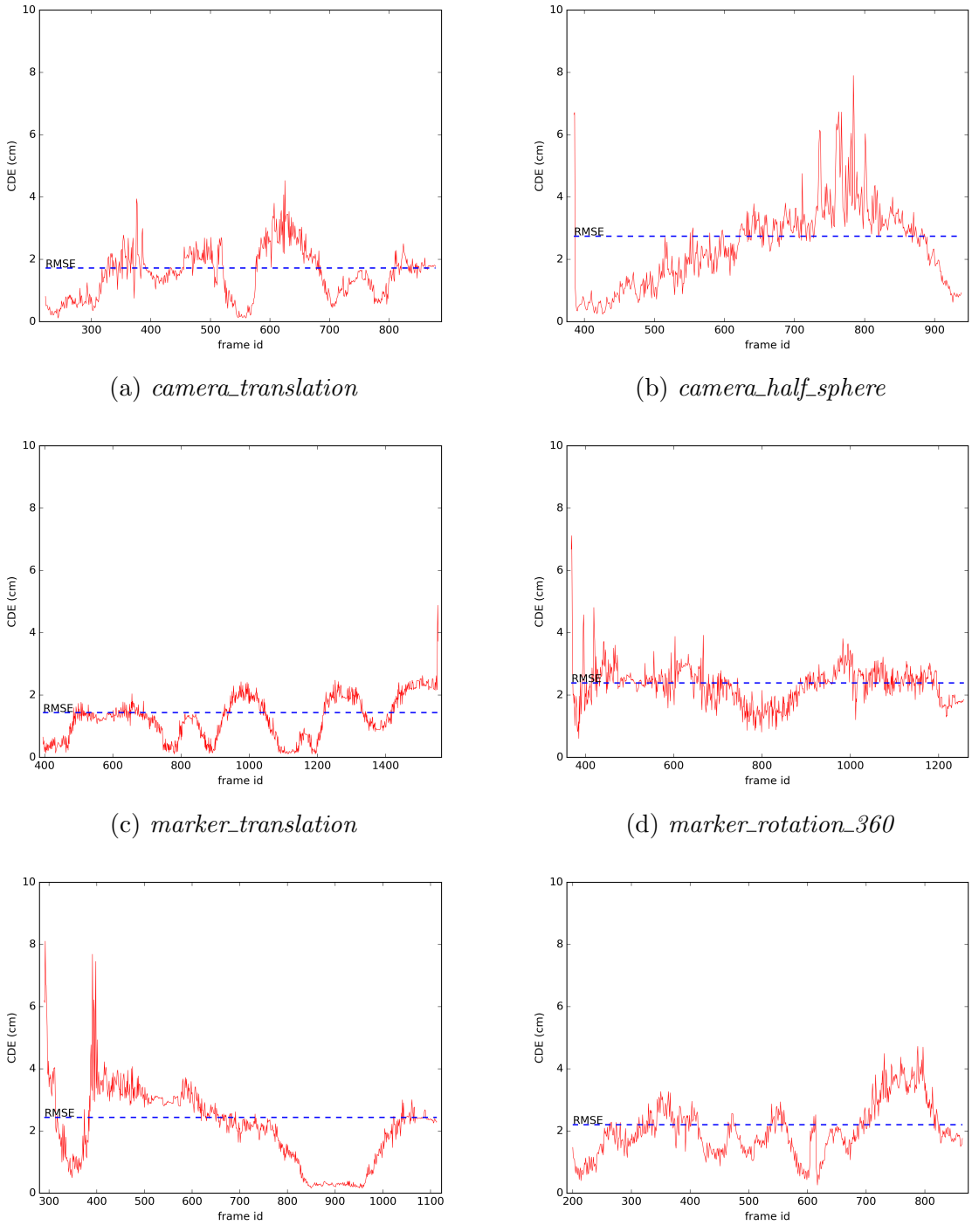
5.4.3 COLLISION

Another possible application is to deal with physical interactions like the collision between two objects. Just like the previous application, we can use the known structure of the target object to generate its mesh and treat collisions with virtual projections.

5.4.4 DISCUSSION

We have demonstrated some AR applications that our system supports. It is not a goal of this work to develop a product, so failures and limitations exist. For example, our system is slow to recover from tracking failures, sometimes the application becomes impractical,

but within the usage restrictions, the apps appear to correctly position virtual objects in the real world and have an acceptable level of stability, that is, the projection does not suffer great fluctuations. But more importantly, we demonstrate that our proposal not only allows new applications in AR but also increase the interaction level between the virtual and real environments. The interaction is a fundamental characteristic of an Augmented Reality software. It is important that virtual objects are over the same rules that act in the real world to increase users' immersion and give them the feeling this is real. So any progress in this direction is important and our work takes a small step in that way.



(a) *camera_translation*

(b) *camera_half_sphere*

(c) *marker_translation*

(d) *marker_rotation_360*

(e) *marker_inclination_zoom*

(f) *free_motion*

Figure 5.3: Evolution of the CDE error per frame on each recorded sequence.

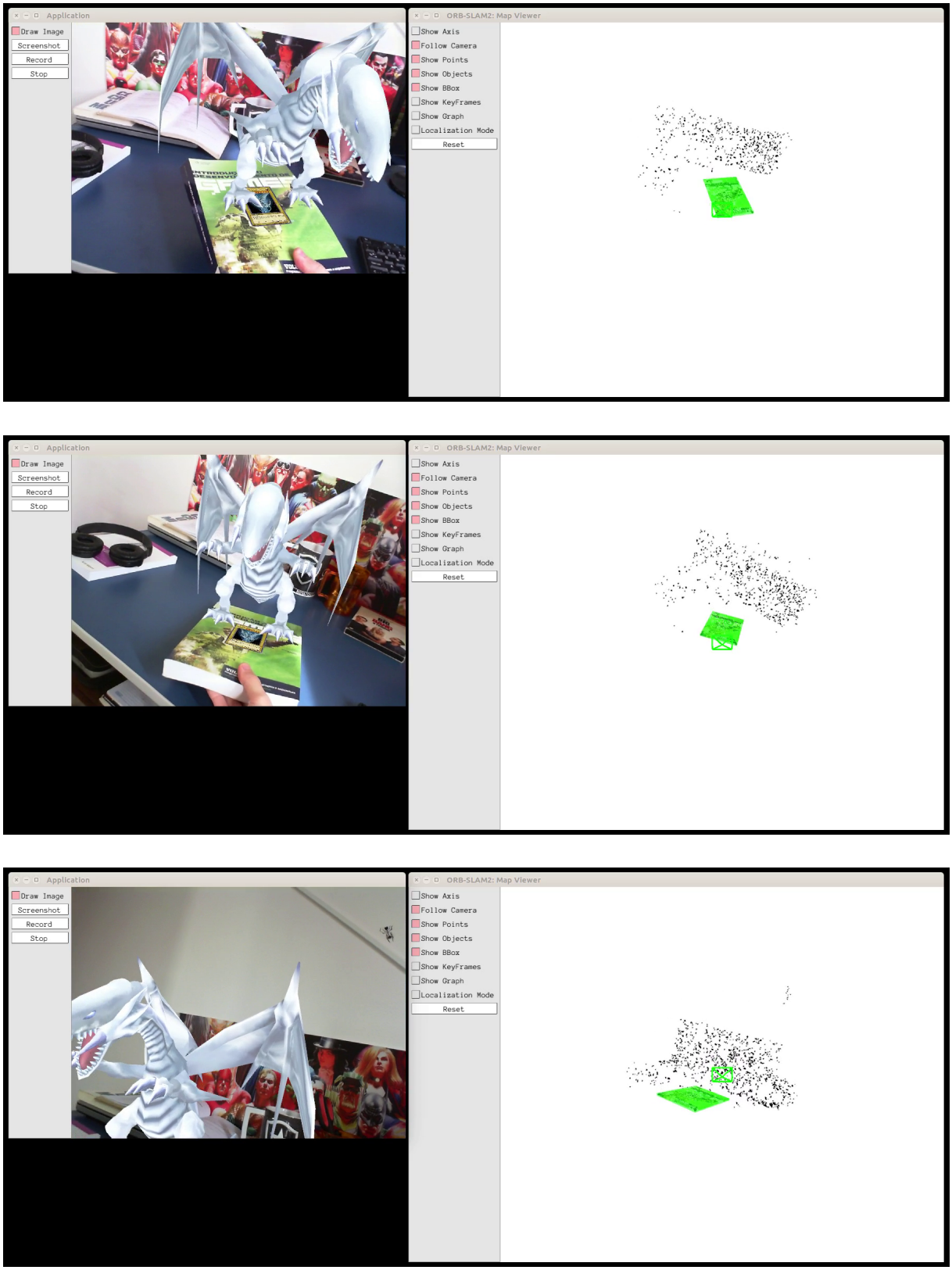


Figure 5.4: Augmented Reality application sequence. The first two rows shows the target object moving, keeping the virtual object aligned. The last row shows a full occlusion of the object, but the projection is still there. Second column images show the correspondent state of the map, updating in real-time.

6 CONCLUSION

This work presented a new graph-based approach to deal with the problem of tracking rigid mobile objects in conjunction with the problem of Simultaneous Localization and Mapping (SLAM) with a monocular camera. We propose a new optimization framework for the front-end capable of simultaneously estimating the current positions of each object visible and the camera. Our hypothesis was that the scene geometric information provides sufficient constraints to distinguish an object moving from the camera motion. However, for this, it is necessary to observe static points enough to support the camera pose estimation, which in turn acts on the delimitation of the object position on the map. As far as we know, this is the first work to propose a completely graph-based optimization approach to solve camera localization and tracking of dynamic objects at the same time. Previous works either assumed a static map and solved object placement instead of tracking or used a probabilistic filter-based approach to predict and update the object's trajectory.

We developed a monocular SLAM system integrated with the object tracking 6DoF based on state-of-the-art work developed in Mur-Artal and Tardós (2017). This work uses the ORB feature extractor for data association at all system stages. The ORB extractor was also used in Ufkes and Fiala (2013) and Gálvez-López et al. (2016) works to find matches of previously known objects in the camera image. Therefore, we kept using the ORB technique also for our system's object recognition step. We design the object visibility state machine thinking on the application to Augmented Reality. The main purpose was to solve the virtual object projection loss problem after the reference object total occlusion.

To verify the camera location accuracy results, we used the public TUM RGB-D dataset (STURM et al., 2012) and compared our results with the ones published by ORB-SLAM article. In contrast, object tracking evaluation was done using sequences recorded by us and compared to the Augmented Reality library ARToolKit. Despite the system limitations that will be discussed below, all the results obtained corroborate our proposal. The substitution of camera optimization model used by ORB-SLAM by our framework did not impact on the camera's accuracy results. In presence of dynamic objects, our

proposal can distinguish well, in some scenarios, the objects' movement from the camera movement, updating the map in real time. We have also demonstrated some applications of our system in Augmented Reality. The visual results, as well as the numerical ones, also corroborate for the proposed framework validation.

Our system has some limitations, the first one is directly related to our framework. Since the observation of background points is required to estimate object movement, our system fails when an object that is fully occupying the camera image moves. Thus, this implies in a limitation of the minimum distance that the camera can be from the object in order to be able to distinguish the scene movements. Also, the observed environment must fit the data association method limitations, in the case of ORB-SLAM, the scenario must be textured to allow feature extraction. The projection-based matching method leads to the system's main limitation: the speed at which an object can move. However, in order to not jeopardize front-end performance, we cannot perform a deeper search for matches. We consider the resolution of this problem to be fundamental to, in fact, enable our system application in Augmented Reality.

6.1 FUTURE WORKS

The system developed in this work has some limitations that can be approached by future works. We believe that the main limitation is the freedom of the objects' movement since this is in line with our proposal. Nowadays, the system can only follow an object trajectory correctly when the movement is smooth and in scenarios where background points appear in sufficient numbers throughout the course. In fact, abrupt movements continue an open problem for vision-based object tracking area, especially for the monocular case. Anyhow, an abrupt movement is one of the main causes of object loss. In this scenario, it is important to re-locate the object as quickly as possible. Thus, one proposal is to work towards re-localization technique improvement, both in quality and speed. Recognizing quickly that the object has been lost is also important, so the state machine can also be the focus of new studies. Regarding the need to have visible background points, as we discussed before, it is due to the projective space ambiguity, where it is difficult to separate the camera movement from the object movement. Therefore, the background points act as a support for the camera movement identification, which, in turn, acts on the object movement identification. One way to handle this can be through new constraints on the

object, e.g., physical constraints as pointed out in Vo et al. (2016) work.

Data association is a crucial pipeline step for any system that depends on image matching. Yet it is also the main source of errors in these systems. Indeed, if there were a hypothetical perfect data association, then many problems would not even exist. Hence, data association is an open problem and lacks new methods. We find in literature recent works that try to solve this problem through the application of Deep Learning, among which we cite Han et al. (2015); Zagoruyko and Komodakis (2015); Yi et al. (2016). In fact, Deep Learning is an area that had massive growth in the last years and is being applied to several problems related to the Computer Vision area. It even has already been applied to the monocular SLAM in the work of Tateno et al. (2017) in order to improve the map points depth estimation.

Another proposal for future work is the construction of a dataset for tracking objects that are compatible with SLAM systems. What we consider ideal for such a dataset is that there be not only objects ground-truth but also camera ground-truth, both in relation to a global fixed-world system. In that way, not only systems like ours would be benefited, but any other tracker system type. Furthermore, other proposals for future work are our framework generalization for stereo and RGB-D SLAM systems, generalization for non-textured or non-rigid objects, human skeleton tracking, among others. In short, there is still much to be done for the improvement of SLAM systems as a whole.

REFERENCES

AZUMA, R. A survey of augmented reality. **Teleoperators and Virtual Environments**, v. 6, p. 355–385, 1997.

BALLARD, D. H. Generalizing the hough transform to detect arbitrary shapes. In: **Readings in computer vision**, 1987. p. 714–725.

BAY, H.; ESS, A.; TUYTELAARS, T.; GOOL, L. V. Speeded-up robust features (surf). **Computer vision and image understanding**, Elsevier, v. 110, n. 3, p. 346–359, 2008.

BAY, H.; TUYTELAARS, T.; GOOL, L. V. Surf: Speeded up robust features. In: SPRINGER. **European conference on computer vision**, 2006. p. 404–417.

CALONDER, M.; LEPETIT, V.; STRECHA, C.; FUA, P. Brief: Binary robust independent elementary features. In: SPRINGER. **European conference on computer vision**, 2010. p. 778–792.

CASTLE, R. O.; MURRAY, D. W. Keyframe-based recognition and localization during video-rate parallel tracking and mapping. **Image and Vision Computing**, Elsevier, v. 29, n. 8, p. 524–532, 2011.

CSURKA, G.; DANCE, C.; FAN, L.; WILLAMOWSKI, J.; BRAY, C. Visual categorization with bags of keypoints. In: PRAGUE. **Workshop on statistical learning in computer vision, ECCV**, 2004. v. 1, n. 1-22, p. 1–2.

DAVISON, A. J. Real-time simultaneous localisation and mapping with a single camera. In: IEEE. **Computer Vision, 2003, Proceedings. Ninth IEEE International Conference on**, 2003. p. 1403.

DAVISON, A. J.; MURRAY, D. W. Simultaneous localization and map-building using active vision. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 24, n. 7, p. 865–880, 2002.

DAVISON, A. J.; REID, I. D.; MOLTON, N. D.; STASSE, O. Monoslam: Real-time single camera slam. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 29, n. 6, p. 1052–1067, 2007.

DISSANAYAKE, M. G.; NEWMAN, P.; CLARK, S.; DURRANT-WHYTE, H. F.; CSORBA, M. A solution to the simultaneous localization and map building (slam) problem. **IEEE Transactions on robotics and automation**, IEEE, v. 17, n. 3, p. 229–241, 2001.

DURRANT-WHYTE, H.; BAILEY, T. Simultaneous localisation and mapping (slam): Part i the essential algorithms. **IEEE Robotics and Automation Magazine**, p. 99–108, 2006.

DURRANT-WHYTE, H.; RYE, D.; NEBOT, E. Localization of autonomous guided vehicles. In: **Robotics Research**, 1996. p. 613–625.

DURRANT-WHYTE, H. F. Uncertain geometry in robotics. **IEEE Journal on Robotics and Automation**, IEEE, v. 4, n. 1, p. 23–31, 1988.

EADE, E.; DRUMMOND, T. Scalable monocular slam. In: IEEE COMPUTER SOCIETY. **Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Volume 1**, 2006. p. 469–476.

ENGEL, J.; KOLTUN, V.; CREMERS, D. Direct sparse odometry. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, 2017.

ENGEL, J.; SCHÖPS, T.; CREMERS, D. Lsd-slam: Large-scale direct monocular slam. In: SPRINGER. **European Conference on Computer Vision**, 2014. p. 834–849.

FERGUS, R. **Multi-view Stereo & Structure from Motion**. 2012. University Lecture.

FIALA, M. Artag, a fiducial marker system using digital techniques. In: **2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)**, 2005. v. 2, p. 590–596 vol. 2. ISSN 1063-6919.

FIORAIO, N.; STEFANO, L. D. Joint detection, tracking and mapping by semantic bundle adjustment. In: IEEE. **Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on**, 2013. p. 1538–1545.

FISCHLER, M. A.; BOLLES, R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In: **Readings in computer vision**, 1987. p. 726–740.

FORSTER, C.; PIZZOLI, M.; SCARAMUZZA, D. Svo: Fast semi-direct monocular visual odometry. In: IEEE. **Robotics and Automation (ICRA), 2014 IEEE International Conference on**, 2014. p. 15–22.

FUENTES-PACHECO, J.; RUIZ-ASCENCIO, J.; RENDÓN-MANCHA, J. M. Visual simultaneous localization and mapping: a survey. **Artificial Intelligence Review**, Springer, v. 43, n. 1, p. 55–81, 2015.

GÁLVEZ-LÓPEZ, D.; SALAS, M.; TARDÓS, J. D.; MONTIEL, J. Real-time monocular object slam. **Robotics and Autonomous Systems**, Elsevier, v. 75, p. 435–449, 2016.

GÁLVEZ-LÓPEZ, D.; TARDOS, J. D. Bags of binary words for fast place recognition in image sequences. **IEEE Transactions on Robotics**, IEEE, v. 28, n. 5, p. 1188–1197, 2012.

HAN, X.; LEUNG, T.; JIA, Y.; SUKTHANKAR, R.; BERG, A. C. Matchnet: Unifying feature and metric learning for patch-based matching. In: **2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, 2015. p. 3279–3286. ISSN 1063-6919.

HARRIS, C.; STEPHENS, M. A combined corner and edge detector. In: CITESEER. **Alvey vision conference**, 1988. v. 15, n. 50, p. 10–5244.

HARRIS, Z. S. Distributional structure. **Word**, Taylor & Francis, v. 10, n. 2-3, p. 146–162, 1954.

KALMAN, R. E. et al. A new approach to linear filtering and prediction problems. **Journal of basic Engineering**, v. 82, n. 1, p. 35–45, 1960.

KATO, H. **ARToolKit 2.33 Documentation (Alpha Version)**, 2005. [Online; acessado em 21-Outubro-2011].

KIM, K.; LEPETIT, V.; WOO, W. Real-time interactive modeling and scalable multiple object tracking for ar. **Computers & Graphics**, v. 36, n. 8, p. 945 – 954, 2012. ISSN 0097-8493. Graphics Interaction Virtual Environments and Applications 2012. Disponível em: <http://www.sciencedirect.com/science/article/pii/S009784931200132X>.

KLEIN, G.; MURRAY, D. Parallel tracking and mapping for small ar workspaces. In: IEEE. **Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on**, 2007. p. 225–234.

KLEIN, G.; MURRAY, D. Parallel tracking and mapping on a camera phone. In: IEEE. **Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on**, 2009. p. 83–86.

KÜMMERLE, R.; GRISETTI, G.; STRASDAT, H.; KONOLIGE, K.; BURGARD, W. g^2o: A general framework for graph optimization. In: IEEE. **Robotics and Automation (ICRA), 2011 IEEE International Conference on**, 2011. p. 3607–3613.

KUNDU, A.; KRISHNA, K. M.; JAWAHAR, C. Realtime multibody visual slam with a smoothly moving monocular camera. In: IEEE. **Computer Vision (ICCV), 2011 IEEE International Conference on**, 2011. p. 2080–2087.

LEPETIT, V.; MORENO-NOGUER, F.; FUA, P. Epnp: An accurate o(n) solution to the pnp problem. **International Journal of Computer Vision**, v. 81, p. 155–166, 2008.

LIN, K.-H.; WANG, C.-C. Stereo-based simultaneous localization, mapping and moving object tracking. In: IEEE. **Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on**, 2010. p. 3975–3980.

LOWE, D. G. Distinctive image features from scale-invariant keypoints. **International journal of computer vision**, Springer, v. 60, n. 2, p. 91–110, 2004.

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

MIGLIORE, D.; RIGAMONTI, R.; MARZORATI, D.; MATTEUCCI, M.; SORRENTI, D. G. Use a single camera for simultaneous localization and mapping with mobile object tracking in dynamic environments. In: **ICRA Workshop on Safe navigation in open and dynamic environments: Application to autonomous vehicles**, 2009. p. 12–17.

MONTEMERLO, M.; THRUN, S.; KOLLER, D.; WEGBREIT, B. et al. Fastslam: A factored solution to the simultaneous localization and mapping problem. In: **Proceedings of the 18th AAAI National Conference on Artificial Intelligence**, 2002. p. 593–598.

MOURAGNON, E.; LHUILLIER, M.; DHOME, M.; DEKEYSER, F.; SAYD, P. Real time localization and 3d reconstruction. In: IEEE. **Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on**, 2006. v. 1, p. 363–370.

MUR-ARTAL, R.; MONTIEL, J. M. M.; TARDOS, J. D. Orb-slam: a versatile and accurate monocular slam system. **IEEE Transactions on Robotics**, IEEE, v. 31, n. 5, p. 1147–1163, 2015.

MUR-ARTAL, R.; TARDÓS, J. D. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. **IEEE Transactions on Robotics**, IEEE, v. 33, n. 5, p. 1255–1262, 2017.

MURPHY, K.; RUSSELL, S. Rao-blackwellised particle filtering for dynamic bayesian networks. In: **Sequential Monte Carlo methods in practice**, 2001. p. 499–515.

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

PIRKER, K.; RÜTHER, M.; BISCHOF, H. Cd slam-continuous localization and mapping in a dynamic world. In: IEEE. **Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on**, 2011. p. 3990–3997.

RAMADASAN, D.; CHATEAU, T.; CHEVALDONNÉ, M. Dcslam: A dynamically constrained real-time slam. In: IEEE. **Image Processing (ICIP), 2015 IEEE International Conference on**, 2015. p. 1130–1134.

ROSTEN, E.; DRUMMOND, T. Machine learning for high-speed corner detection. In: SPRINGER. **European conference on computer vision**, 2006. p. 430–443.

RUBLEE, E.; RABAUD, V.; KONOLIGE, K.; BRADSKI, G. Orb: An efficient alternative to sift or surf. In: IEEE. **Computer Vision (ICCV), 2011 IEEE international conference on**, 2011. p. 2564–2571.

SALAS-MORENO, R. F.; NEWCOMBE, R. A.; STRASDAT, H.; KELLY, P. H.; DAVISON, A. J. Slam++: Simultaneous localisation and mapping at the level of objects. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**, 2013. p. 1352–1359.

SCHRÖDINGER, E. Die gegenwärtige situation in der quantenmechanik. **Naturwissenschaften**, v. 23, n. 48, p. 807–812, Nov 1935. ISSN 1432-1904. Disponível em: <https://doi.org/10.1007/BF01491891>.

SILTANEN; HAKKARAINEN; HONKAMAA. Automatic marker field calibration. In: **Virtual Reality International Conference (VRIC2007)**, 2007. p. 261–267.

SMITH, R.; SELF, M.; CHEESEMAN, P. Estimating uncertain spatial relationships in robotics. In: _____. **Autonomous Robot Vehicles**, 1990. p. 167–193. ISBN 978-1-4613-8997-2.

SMITH, R. C.; CHEESEMAN, P. On the representation and estimation of spatial uncertainty. **The international journal of Robotics Research**, Sage Publications Sage CA: Thousand Oaks, CA, v. 5, n. 4, p. 56–68, 1986.

STRASDAT, H.; MONTIEL, J.; DAVISON, A. J. Scale drift-aware large scale monocular slam. **Robotics: Science and Systems VI**, Universidad de Zaragoza Zaragoza, Spain, v. 2, 2010.

STRASDAT, H.; MONTIEL, J. M.; DAVISON, A. J. Visual slam: why filter? **Image and Vision Computing**, Elsevier, v. 30, n. 2, p. 65–77, 2012.

STURM, J.; ENGELHARD, N.; ENDRES, F.; BURGARD, W.; CREMERS, D. A benchmark for the evaluation of rgb-d slam systems. In: **Proc. of the International Conference on Intelligent Robot Systems (IROS)**, 2012.

TAN, W.; LIU, H.; DONG, Z.; ZHANG, G.; BAO, H. Robust monocular slam in dynamic environments. In: IEEE. **Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on**, 2013. p. 209–218.

TATENO, K.; TOMBARI, F.; LAINA, I.; NAVAB, N. Cnn-slam: Real-time dense monocular slam with learned depth prediction. **arXiv preprint arXiv:1704.03489**, 2017.

TRIGGS, B.; MCLAUCHLAN, P. F.; HARTLEY, R. I.; FITZGIBBON, A. W. Bundle adjustment — a modern synthesis. In: **Vision Algorithms: Theory and Practice**, 2000.

UFKES, A.; FIALA, M. A markerless augmented reality system for mobile devices. In: IEEE. **Computer and Robot Vision (CRV), 2013 International Conference on**, 2013. p. 226–233.

UMEYAMA, S. Least-squares estimation of transformation parameters between two point patterns. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 13, n. 4, p. 376–380, April 1991. ISSN 0162-8828.

VO, M.; NARASIMHAN, S. G.; SHEIKH, Y. Spatiotemporal bundle adjustment for dynamic 3d reconstruction. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**, 2016. p. 1710–1718.

WANGSIRIPITAK, S.; MURRAY, D. W. Avoiding moving outliers in visual slam by tracking moving objects. In: IEEE. **Robotics and Automation, 2009. ICRA'09. IEEE International Conference on**, 2009. p. 375–380.

YI, K. M.; VERDIE, Y.; FUA, P.; LEPETIT, V. Learning to assign orientations to feature points. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**, 2016. p. 107–116.

ZAGORUYKO, S.; KOMODAKIS, N. Learning to compare image patches via convolutional neural networks. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**, 2015. p. 4353–4361.