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

Gustavo Magalhães Moura

VEM-SLAM - Virtual Environment Modelling through SLAM

Juiz de Fora
2020

Gustavo Magalhães Moura

VEM-SLAM - Virtual Environment Modelling through SLAM

Dissertação apresentada ao Pós-Graduação em Ciência da Computação da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. D.Sc. Rodrigo Luis de Souza da Silva

Juiz de Fora

2020

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

Moura, Gustavo Magalhães.
VEM-SLAM - Virtual Environment Modelling through SLAM / Gustavo Magalhães Moura. -- 2020.
71 p.

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, 2020.

1. SLAM. 2. Detecção de objetos. 3. Realidade virtual. I. Silva, Rodrigo Luis de Souza da, orient. II. Título.

Gustavo Magalhães Moura

"VEM-SLAM – Virtual Environment Modelling through SLAM"

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do grau de Mestre em Ciência da Computação.

Aprovada em 04 de março de 2020.

BANCA EXAMINADORA

Prof. Dr. Rodrigo Luís de Souza da Silva – Orientador
Universidade Federal de Juiz de Fora

Prof. Dr. Marcelo Bernardes Vieira
Universidade Federal de Juiz de Fora

P.P.
Prof. Dr. Antônio Lopes Apolinário Júnior
Universidade Federal da Bahia

AGRADECIMENTOS

Primeiramente, agradeço à minha esposa Marluce, com quem amo compartilhar a minha vida. Obrigado pelo carinho, paciência e compreensão ao longo de toda nossa vida juntos.

Agradeço à todos os professores do DCC e do PGCC que buscaram transmitir seu conhecimento e ensinamentos de vida ao longo da minha vida acadêmica. Em especial, ao Prof. Rodrigo Luis, responsável pela orientação deste trabalho e diversas outras.

Aos colegas e amigos com quem pude contar nestes anos de convivência com quem compartilhei momentos de concentração, dedicação, alegria e nervosismo ao longo destes anos.

Aos meus familiares, em especial aos meus avós maternos, que contribuíram, cada um à sua maneira, para a minha formação moral e ética.

Agradeço aos funcionários do DCC e PGCC pelo suporte. E agradeço à CAPES pelo apoio financeiro sem o qual não haveria este trabalho.

RESUMO

O problema de mapeamento de um ambiente real e reconhecimento dos objetos contidos neste ambiente é um problema da área de Visão Computacional e tem recebido atenção com o avanço de soluções SLAM e soluções robustas de reconhecimento de objetos. O problema da robótica de Localização e Mapeamento Simultâneos (Simultaneous Localization and Mapping - SLAM) consiste em criar um mapa (geralmente geométrico) da cena ao mesmo tempo em que estima a pose do observador. As soluções para este problema são utilizadas em diversas áreas onde se deseja mapear um ambiente e extrair informações geométricas deste. O reconhecimento de objetos permite identificar o objeto na cena conforme as classes de objetos da base de dados de referência. Para o reconhecimento em imagens 2D, as melhores soluções são baseadas em redes neurais convolucionais. Entretanto, para a obtenção das informações geométricas 3D dos objetos na cena, são necessárias outras técnicas que variam conforme o modelo do objeto 3D de referência. Neste trabalho, será apresentada uma nova abordagem para lidar com a estimativa de pose de objetos 3D a partir de imagens de cenas estáticas de ambientes internos. Para isso, uma integração entre um detector de objetos em imagens e uma solução SLAM monocular baseada em *keyframes* foi desenvoldida. Como resultados, demonstramos uma melhoria na estimativa da trajetória da câmera em relação ao método original e uma utilização do sistema implementado na criação de ambientes virtuais.

Palavras-chave: SLAM. Deteção de objetos. Realidade Virtual.

ABSTRACT

The problem of mapping a real environment and recognizing the objects contained in this environment is a problem in the Computer Vision area and has received attention with the advances of SLAM solutions and robust object recognition solutions. The problem with Simultaneous Localization and Mapping (SLAM) robotics is to create a (generally geometric) map of the scene while estimating the viewer's pose. The solutions to this problem are used in several areas where a map of the environment is desirable and extract geometric information from it. Object recognition allows us to identify the object in the scene according to the object classes of the reference database. For recognition in 2D images, the best solutions are based on convolutional neural networks. However, to obtain the 3D geometric information of the objects in the scene, other techniques are necessary that vary according to the model of the reference 3D object. In this work, we present a new approach to deal with the pose estimation of 3D objects from images of static scenes of indoor environments. We also propose a new integration between an object detector and a monocular SLAM solution based on keyframes. As results we demonstrate an improvement in the estimation of the camera's trajectory in relation to the original method and a use of the system implemented in the creation of virtual environments.

Keywords: SLAM. Object detection. Virtual Reality.

LIST OF FIGURES

Figure 1 – Steps followed in this SLR. 21

Figure 2 – Steps for search for studies. 23

Figure 3 – Selected studies divided by detection method classes. 29

Figure 4 – Object detection paradigms and processing types in the selected studies. 30

Figure 5 – Examples of object detection types. 35

Figure 6 – Example of object labeling. (a) labeling all object instances present in images by a predefined set of object classes. (b) labeling the object instances by a specific object label. 36

Figure 7 – Basic steps of 3D object pose estimator. The object detector process the images and generates bounding boxes as output (a), the ellipses that fits the bounding boxes are calculated (b), and the ellipsoids are estimated (c). 37

Figure 8 – The right circular cone and example of generators. 38

Figure 9 – The three non-degenerate conics in the intersection of a double right circular cone and plans. 38

Figure 10 – Conic geometric elements exemplified in an ellipse. ξ_1 and ξ_2 are the semi-axes, θ is the angle relative to abscissa axis and $\mathbf{x}_c$ is the center vector. 40

Figure 11 – (a) Points $\mathbf{x}$ satisfying $\mathbf{x}^{\mathrm{T}}\mathtt{C}\mathbf{x} = 0$ lie on a point conic. (b) Lines $\mathbf{l}$ satisfying $\mathbf{l}^{\mathrm{T}}\mathtt{D}\mathbf{l} = 0$ are tangent to the point conic $\mathtt{C}$. The conic $\mathtt{C}$ is the envelope of the lines $\mathbf{l}$. 42

Figure 12 – Relation between dual conics and dual quadrics. The ellipse $\mathtt{D}$ is tangent to three generic lines $\mathbf{l}_1$, $\mathbf{l}_2$ and $\mathbf{l}_3$, which belong to three generic planes $\boldsymbol{\rho}_1$, $\boldsymbol{\rho}_2$ and $\boldsymbol{\rho}_3$ tangent to the ellipsoid $\mathtt{E}$. 43

Figure 13 – Overview of the system developed. The red labels highlights our developments or modifications in ORB-SLAM2. 46

Figure 14 – Example of bounding boxes (yellow) and corresponding fitted ellipses (red) for a set of objects. 47

Figure 15 – Example of geometric elements in a scene. $\mathtt{K}_f$, $\mathtt{K}_{f+1}$ and $\mathtt{K}_{f+2}$ are sequential keyframes, $\mathtt{C}_{fi}$, $\mathtt{C}_{(f+1)i}$ and $\mathtt{C}_{(f+2)i}$ are the conics related with each keyframe from a generic region i in each keyframe, and $\mathtt{Q}_j$ is the quadric related to object o_j. 52

Figure 16 – The output of Irace execution on optimization of nonlinear genetic algorithm parameters. 56

Figure 17 – Example of errors (red) on estimated trajectory (blue) and ground truth (black) in same TUM sequence. There are differences in camera trajectory initialization and loss of localization. The green rectangles emphasizes the initialization differences between the executions and the orange rectangle emphasis the localization loss. 60
Figure 18 – Objects pose and shape estimation of objects in an sequences of TUM dataset. The pose of the keyframes are represented by the green rectangle in the objects map. 61
Figure 19 – Application developed to generating a virtual environment from TUM dataset sequences. The desk was fixed in both environments. . . 62

LIST OF TABLES

Table 1 – Research questions and their aims. 22
Table 2 – Search results before and after removing duplicates. 23
Table 3 – Results of application of inclusion and exclusion criteria 25
Table 4 – Quality assessment checklist. 26
Table 5 – Justification of questions in the quality assessment. 27
Table 6 – Quality assessment result. 27
Table 7 – Quality assessment of studies. 28
Table 8 – Publication years in this SLR. 28
Table 9 – Search range and optimum values of each parameters on Irace optimization 56
Table 10 – ORB-SLAM2 and VEM-SLAM timing results of each stage in proposed pipeline in milliseconds (mean ± std deviations) and frequency of execution (per frame). Bold figures denote the lower mean time processing for each stage. 58
Table 11 – Average localization errors and distance ratios on TUM sequences. The improvement of camera trajectory estimation was significant for sequences with a high number of frames, below the dotted line. Bold figures denote the lower value of RMSE ATE of each sequence. 59

LIST OF ACRONYMS

ATE	Absolute Trajectory Error
BRIEF	Binary Robust Independent Elementary Features
CNN	Convolutional Neural Network
FAST	Features from Accelerated Segment Test
GPU	Graphics Processing Unit
ORB	Oriented FAST and Ro tated BRIEF
RMSE	Root-Mean-Square Error
SIFT	Scale-Invariant Feature Transform
SLAM	Simultaneous Localization And Mapping
SURF	Speeded Up Robust Features
SVD	Singular Vector Decomposition
TPU	Tensor Processing Unit

CONTENTS

1 INTRODUCTION 19
1.1 PROBLEM DEFINITION 20
1.2 OBJECTIVES 20
1.3 OUTLINE 20
2 RELATED WORKS 21
2.1 SYSTEMATIC LITERATURE REVIEW 21
2.1.1 Research questions 22
2.1.2 Keywords definition 22
2.1.3 Databases definition 22
2.1.4 Search for studies 22
2.1.5 Studies selection 24
2.1.6 Quality assessment (QA) 24
2.2 ANALYSIS AND DISCUSSION OF SELECTED PAPERS 25
2.2.1 RQ1: What 3D object detection methods are applied with a visual SLAM approach? 25
2.2.2 RQ2: How the methods affected the SLAM approach? 30
2.2.3 RQ3: What techniques are used with these approaches? 31
2.2.4 RQ4: What are the premisses of these approaches? 31
2.3 CONCLUSIONS 32
3 FUNDAMENTALS 35
3.1 OBJECT DETECTION 35
3.2 VISUAL SIMULTANEOUS LOCALIZATION AND MAPPING 36
3.3 BUNDLE ADJUSTMENT 37
3.4 CONIC SECTIONS AND QUADRIC SURFACES 37
3.4.1 Conic sections 37
3.4.1.1 Ellipse 39
3.4.2 Quadric Surfaces 40
3.4.2.1 Ellipsoid 41
3.4.3 Dual conics and dual quadrics 41
3.4.4 Rigid transformations 43
4 PROPOSED METHOD 45

4.1 3D OBJECT POSE ESTIMATION AND OPTIMIZATION 45
4.1.1 Normalization 48
4.1.2 Nonlinear optimization 49
4.2 OBJECT ASSOCIATION 51
4.3 FULL BUNDLE ADJUSTMENT 53
5 EXPERIMENTAL RESULTS 55
5.1 IMPLEMENTATION DETAILS 55
5.2 YOLOV2 AND ORB-SLAM2 INTEGRATION 55
5.3 NONLINEAR OPTIMIZATION THROUGH GENETIC ALGORITHM . 55
5.4 THREADS TIMING EVALUATION 57
5.5 CAMERA LOCALIZATION 57
5.6 OBJECT 3D POSE ESTIMATION 59
5.7 PROVE OF CONCEPT: GENERATING A VIRTUAL ENVIRONMENT 60
6 CONCLUSION AND FUTURE WORKS 63
REFERENCES 65
APPENDIX A - References of Systematic Literature Review 69

1 INTRODUCTION

A robotic system needs to simultaneously extract information from the environment, build a map, and localize itself in this map when moving in an unknown environment autonomously. This computational challenge is known as SLAM, an abbreviation for Simultaneous Localization and Mapping. SLAM was initially proposed (1) to perform autonomous control of mobile robots. The SLAM solutions follow the technological advances in sensors and algorithms, and nowadays there are a diversity of solutions applied in a variety of uses. When a SLAM solution uses only a camera (monocular, RGB-D or stereo) as sensor device the problem is called visual SLAM or vSLAM, and are well summarized in (2, 3). vSLAM is broadly researched due to the simple configuration of hardware, but the technical challenges are higher than others because the only environment information is visual.

One of the steps of SLAM solutions is to map the environment and store it in some structure, for example, a point cloud. This map is a representation of the real world and only contains geometric information, such as point positions. It is essential to add a semantic meaning into this map to allow the computational systems to improve the interaction of the real world. This semantic assignment can be accomplished, for example, by identifying the real-world objects on the map. One of the applications of a semantic mapping of an environment is the possibility to reconstruct this environment in a virtual reality world. The virtual world would contain the objects of the recorded scene.

Object detection is the task of locating and labeling objects in an image, and it is an important and non-trivial task in the field of computer vision. There is a wide variety of object detection methods, mainly using machine learning approaches, especially convolutional neural networks. To identify real-world objects in the map, as mentioned before, it is necessary a technique to transform the 2D visual information into a 3D object representation.

The union of object detection and vSLAM techniques has great importance in the field of robotics, virtual and augmented reality, autonomous driven, among others. In literature, there are many visual SLAM solutions integrated with object detectors, but the majority of the studies use RGB-D cameras. The use of RGB-D cameras facilitates some steps of the SLAM solution due to additional pixel-depth information in every frame. However, RGB-D cameras are still expensive and limited to equipment with high value-added. On the other hand, monocular cameras are cheaper and more common than RGB-D cameras. Nevertheless, its use adds more complexity in the SLAM because of two frames are necessary to achieve the depth information.

Our main contribution in this work is to provide an integration of a 2D object detector, a 2D-to-3D object recover technique, and a monocular vSLAM solution. In short,

we record an environment, with the frames we identify the objects and, at the same time, we build a map and estimate the 3D pose of objects in the environment. To the best of our knowledge, we are the first to propose an object-aware monocular SLAM system that recognizes and reconstructs the 3D pose of objects using only RGB images and apply it to build a virtual world.

1.1 PROBLEM DEFINITION

The problem of identifying and determining the 3D positioning of objects in a static indoor environment is the focus of the present work, with the restriction that only RGB images should be used. The challenge in this work is to estimate the 3D position, scale, and rotation of the objects with only RGB images as input and integrates them in a SLAM map. In our hypothesis, it is possible to build a virtual environment from a real scene filmed by a monocular camera.

1.2 OBJECTIVES

The main objective of this work is the research and development of a method, coupled with the state of the art of SLAM techniques, to map a real environment using a monocular camera, identifying the objects in this environment according to a database, and determining their positions in real time. The secondary objective is to develop an application of this method in the area of virtual reality by building a virtual environment based on the mapped real environment.

1.3 OUTLINE

The remainder of this work is organized as follows. Chapter 2 presents a systematic literature review of researches related in the field. The main concepts needed to understand our work are the subject of Chapter 3. Our proposal and develop system are presented in Chapter 4. Chapter 5 shows our experimental results. Finally, the conclusion and suggestions for future works are the topics of Chapter 6.

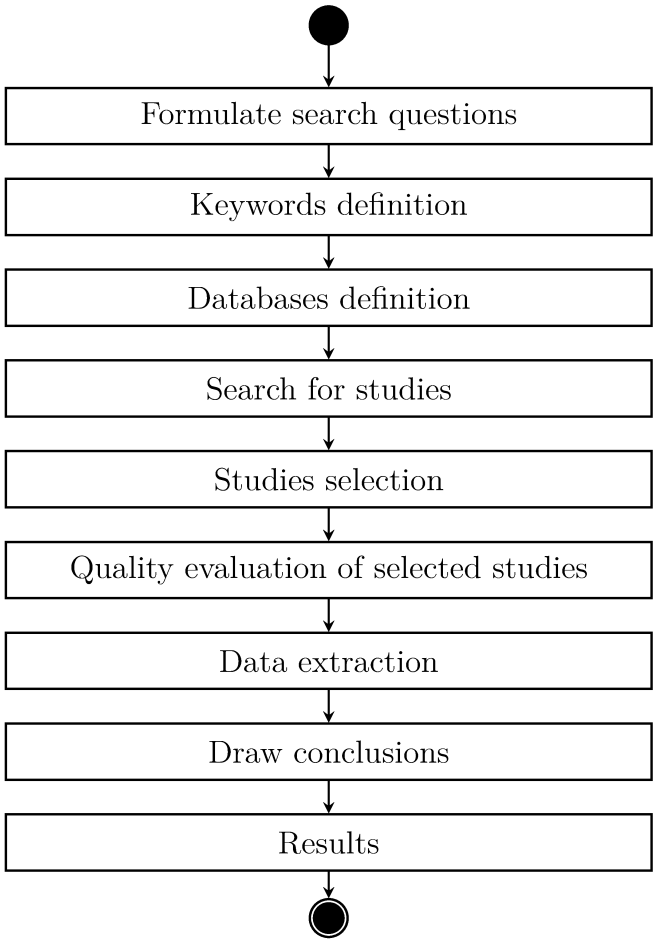
2 RELATED WORKS

As explained in Chapter 1, one of the objectives of this work is to develop a method to map a real scene with a monocular camera to determine the pose of objects in a scene. Thus, a Systematic Literature Review (SLR) was conducted to summarize the relevant works in the field where object detection techniques were combined with vSLAM solutions. At the end of the review, we evaluated what methods and techniques would be used to achieve the objective of this work.

2.1 SYSTEMATIC LITERATURE REVIEW

This SLR followed the steps on (4), which is an adaptation for research in Computer Science and related areas based on the work of (5). The steps of this SLR are presented in Figure 1.

Figure 1: Steps followed in this SLR.



Source: Elaborated by the author (2020).

2.1.1 Research questions

As the first step, we defined what we want to know about the subject, and we formulate questions about it. These research questions guided this SLR, and Table 1 listed the formulated research questions as well as their goals.

Table 1: Research questions and their aims.

Id	Research question	Aim
RQ1	What 3D object detection methods were applied with a visual SLAM approach?	Learning what methods were used with SLAM solutions and their specificities.
RQ2	How the methods affected the SLAM approach?	Understanding how the object detector 3D output was used to improve the SLAM.
RQ3	What techniques are used with these approaches?	Understanding what basic techniques were used to develop an object-SLAM approach in practice.
RQ4	What are the premisses of these approaches?	Understanding the approaches limitations and their applicability.

Source: Elaborated by the author (2020).

2.1.2 Keywords definition

The next step was to extract keywords from the questions to guide the search on databases. These keywords, combined with Boolean operators, synonyms, and related words, will form query strings that we use in databases to select studies. The keywords were visual, SLAM, object, and detection.

2.1.3 Databases definition

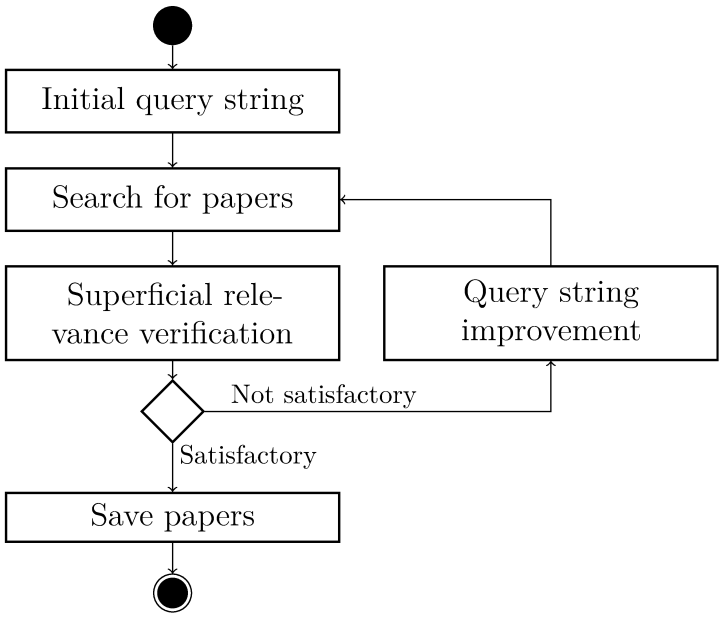
We used the main electronic databases in the field to retrieve the papers which were IEEE Xplore[1], Scopus[2], Science Direct[3], Web of Science[4] and ACM[5]. The advanced search page was used in all databases filtering the search, preferably by title, abstract, and keywords.

2.1.4 Search for studies

The search for studies consists of iterative steps, as shown in Figure 2 and explained in the next subsections.

[1] `https://ieeexplore.ieee.org`
[2] `https://www.scopus.com/`
[3] `https://www.sciencedirect.com/`
[4] `http://www.webofknowledge.com/`
[5] `https://dl.acm.org/`

Figure 2: Steps for search for studies.



Source: Elaborated by the author (2020).

The initial query string was a simple concatenation of chosen keywords: "visual AND SLAM AND Object AND detection" and 466 studies were retrieved. In this set, we noticed studies that use non-camera sensors like laser rangers, sonar, inertial measurement unit (IMU), combining or not with cameras, and these studies were out of our scope. Besides, some reference studies were left out of the set.

The query string was successively refined, and the final query string was "SLAM AND Object AND (Recognition OR Tracking) AND (Image OR Visual OR Camera)", and 1435 studies were retrieved. After that, the duplicated studies were removed, and we ended up with 867 candidate studies for the primary studies. Table 2 lists the search results on databases before and after removing duplicates.

Table 2: Search results before and after removing duplicates.

Database	Before	After
ACM	90	26
IEEE Xplore	473	274
Science Direct	23	4
Scopus	496	489
Web of Science	353	74
Total	1435	867

Source: Elaborated by the author (2020).

2.1.5 Studies selection

Inclusion and exclusion criteria were defined to align the candidate studies to the goal of the SLR. These criteria could be extracted from the questions and restrict other aspects, for example, field studies, surveys, article language, year of study, and others.

The inclusion criteria were:

a) Studies which used some rigid and static object detection technique combined with some SLAM solution;

b) Studies which used only monoculars, RGB-D or stereo cameras;

c) Studies which proposed approaches designed to indoor environments;

d) Papers written in English.

The exclusion criteria were:

a) Studies which did not use some rigid and static object detection technique combined with some SLAM solution;

b) Studies which did not use only monoculars, RGB-D or stereo cameras;

c) Studies which proposed approaches designed for outdoor environments;

d) Studies which the aim was not object detection;

e) Papers not written in English;

f) Short papers or papers with less than five pages.

The inclusion and exclusion criteria formulated previously were applied to the candidate studies in two phases. The objective of this step is to remove works that are not in line with the SLR. The criteria were applied in the title and abstract of all candidate studies in the first phase. In phase two, the criteria were applied in the full text of remaining candidate studies. The results are shown in Table 3. Since all studies in the ACM and Web of Science databases have been deleted, these databases will not appear in the results from now on.

2.1.6 Quality assessment (QA)

In this step, the quality criteria were defined to provide weight to the remaining studies to reach the goals described in (5) and the objective of this SLR. The QA was designed as a checklist with six questions to evaluate the quality of the study and the paper. The questions presented in the checklist, their possible answers, and the score scale

Table 3: Results of application of inclusion and exclusion criteria

Database	Before	Excluded		Remaining
		First step	Second step	
ACM	26	26	0	0
IEEE Xplore	274	257	9	8
Science Direct	4	2	1	1
Scopus	489	408	41	40
Web of Science	74	70	4	0
Total	867	763	55	49

Source: Elaborated by the author (2020).

are shown in Table 4. Each question was formulated with a purpose, and these purposes are presented in Table 5.

The final score of each study is the sum of points in each question. Consequently, the studies could reach between 0 to 12 points. For adherence with the review purpose, the studies which had 8 points or more were included in the final list of primary studies. After a full reading of the studies, the checklist was applied. The results are shown in Table 6. The scores of selected papers in each question are presented in Table 7.

2.2 ANALYSIS AND DISCUSSION OF SELECTED PAPERS

In this section, the results in this SLR are presented and analyzed. The 30 studies were discussed to answer the research questions. The next subsections will discuss the answers to the research questions formulated in Section 2.1.1.

Some questions in the quality checklist were designed to select relevant studies that had features that were only possible through recent accomplishments. For example, public large-scale image datasets and collaborative repository hosting services. This selection is visible in Table 8, and the selected papers were restricted to the last decade.

2.2.1 RQ1: What 3D object detection methods are applied with a visual SLAM approach?

The purpose of this question is to understand what are the features of the 3D object detection methods used with SLAM in the selected studies. The methods in the selected studies can be grouped under several aspects as follows.

The majority of the selected studies, 69.0% (20), used RGB-D cameras (A1, A2, A5, A7, A9, A10, A11, A13, A16, A17, A18, A19, A20, A23, A24, A25, A26, A27, A28, A29) and 31.0% (9) used monocular cameras (A3, A4, A6, A8, A12, A14, A15, A21, A22). This result is expected because the RGB-D cameras have the advantage of the pixel-wise depth information, and this facilitates the SLAM modeling. Also, the RGB-D sensors

Table 4: Quality assessment checklist.

Id	Question	Answer	Value
Qa	Is the methodology clearly described and justified?	The authors did not describe or justify properly	0
		The authors described and justified in a simple way	1
		The authors described and justified broadly	2
Qb	Are the tracked objects incorporated on the SLAM map points?	The objects are not incorporated	0
		A simple representation of the objects is incorporated	1
		A full representation of the objects is incorporated	2
Qc	Are there more than one object for tracking in the experiments?	Only one object is used in the experiments	0
		Two or three objects are used in the experiments	1
		More than three objects are used in the experiments	2
Qd	Are public datasets used for evaluation?	A public dataset is not used in the experiments	0
		A public dataset is used in the experiments	1
Qe	Are the results compared with other techniques?	The results are not compared with other techniques	0
		The results are compared with only one technique	1
		The results are compared with more than one technique	2
Qf	Are the results extensively discussed?	The results are described and not discussed	0
		The results are described and poorly discussed	1
		The results are described and discussed	2
		The results are described and broadly discussed	3

Source: Elaborated by the author (2020).

Table 5: Justification of questions in the quality assessment.

Id	Justify
Qa	The methods has to be comprehensively described and justified to answer the research questions
Qb	Having a satisfactory integration with SLAM, the objects detected should have a representation in the SLAM map
Qc	More than one object must be detected in the experiments to avoid bias
Qd	The use of public datasets or turning their own datasets public allows the comparisons of the results
Qe	The comparison of techniques allows us to evaluate the evolution in the field
Qf	The results analyzes allow us to evaluate the benefits and limitations of each technique

Source: Elaborated by the author (2020).

Table 6: Quality assessment result.

Database	Excluded	Selected
IEEE Xplore	2	6
Scopus	18	22
Science Direct	0	1
Total	20	29

Source: Elaborated by the author (2020).

became cheaper and accessible over the last years. However, the depth sensor is restricted to use in indoor environments, and the depth range is limited.

One way to classify 3D object detectors is how the method achieves the object detection. Based on the work of Cheng and Han (6), the object detection methods in the selected studies can be divided into three non-independent categories: template matching-based methods, knowledge-based methods, and machine learning-based methods. In template matching-based methods, there is a previous generation of templates for each object to be detected and a matching technique to measure the similarity among objects detected in the scene and the templates. The knowledge-based object detection methods test hypotheses by establishing heuristics through the previous knowledge of the problem. This knowledge is usually geometric and has context information. The machine learning-based methods utilize the recent advances in machine learning techniques reducing the object detection problem to a classification problem. These categories are not independent, and Figure 3 shows the division of the selected studies by their predominant classes according to the classification previously defined.

Among the select studies, those that use machine learning approaches stand out. The principal method of machine learning used in the selected studies was Deep Convolutional Neural Network as 2D object detector, 13 studies, (A2, A8, A9, A10, A12,

Table 7: Quality assessment of studies.

Reference	Qa	Qb	Qc	Qd	Qe	Qf	Total
(A1)	2	2	2	1	0	2	9
(A2)	2	1	2	1	2	3	11
(A3)	2	2	2	1	0	2	9
(A4)	2	2	2	0	0	2	8
(A5)	2	2	2	1	0	2	9
(A6)	2	2	2	1	2	3	12
(A7)	2	2	2	0	1	2	9
(A8)	2	2	2	1	2	1	10
(A9)	2	2	2	1	1	2	10
(A10)	2	2	2	1	0	1	8
(A11)	2	2	2	1	1	2	10
(A12)	2	2	2	0	1	1	8
(A13)	2	2	2	0	0	2	8
(A14)	2	2	2	0	0	3	9
(A15)	2	2	2	1	2	3	12
(A16)	2	2	1	0	0	3	8
(A17)	2	2	2	1	0	3	10
(A18)	2	2	2	1	2	3	12
(A19)	2	2	2	1	2	3	12
(A20)	2	2	2	1	2	3	12
(A21)	2	2	0	1	1	3	9
(A22)	2	2	2	1	2	3	12
(A23)	2	1	2	1	2	1	9
(A24)	2	2	1	1	2	2	10
(A25)	2	2	2	1	0	2	9
(A26)	2	2	2	1	2	2	11
(A27)	2	1	2	0	2	3	10
(A28)	2	2	2	1	2	2	11
(A29)	2	2	2	1	0	2	9

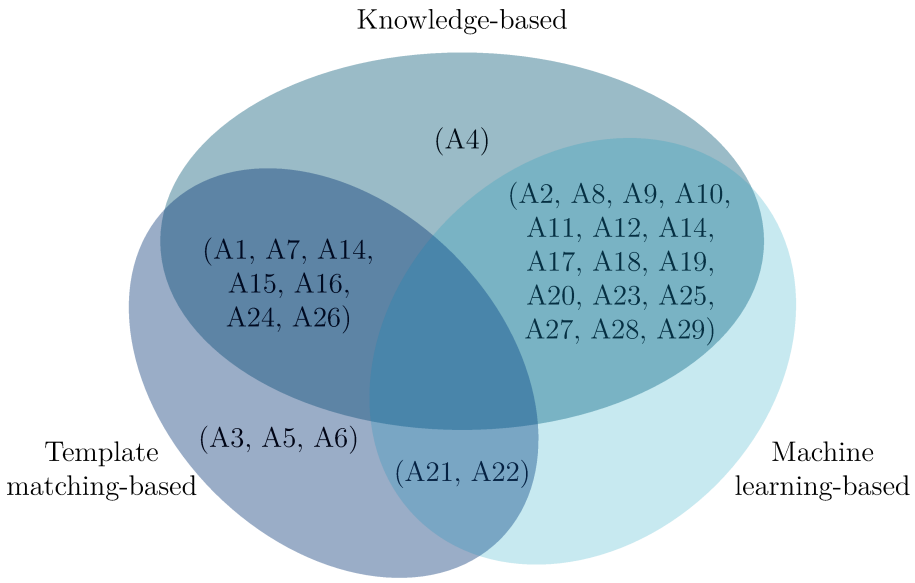
Source: Elaborated by the author (2020).

Table 8: Publication years in this SLR.

Year	Studies	References
2011	1	(A3)
2013	1	(A5)
2014	3	(A16, A24, A29)
2015	2	(A14, A22)
2016	5	(A1, A4, A6, A11, A26)
2017	1	(A25)
2018	8	(A7, A10, A15, A17, A18, A19, A21, A28)
2019	8	(A2, A8, A9, A12, A13, A20, A23, A27)

Source: Elaborated by the author (2020).

Figure 3: Selected studies divided by detection method classes.



Source: Elaborated by the author (2020).

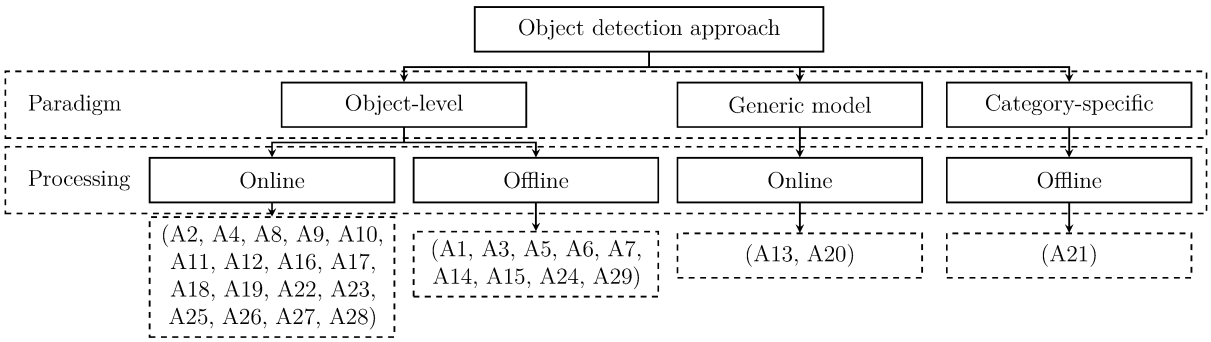
A13, A17, A18, A19, A20, A23, A25, A27, A28). It is interesting to note in Figure 3 that all methods dealing with machine learning in the selected studies are connected to other classes. This observation is expected since the output of these object detection methods is the estimation of the pose of the object in 2D image coordinates and a label, and other methods transform the amount of 2D data in 3D information. The majority of these methods used rules based on the knowledge of the application context to build the object model.

The object detection methods in the selected studies can also be classified according to three paradigms (7): object level, generic model, and category-specific. These paradigms are based on how the objects are characterized within SLAM. In the object-level paradigm, there are models of the objects previously constructed, or the models are constructed individually during the execution of the method. In the second paradigm, generic model, all objects detected fit in a representation regardless of their category. In the category-specific paradigm, an average object model is constructed for each object category to be detected. The vast majority of the selected studies, 26 studies (89.7%), uses object detection approaches with the object-level paradigm. Two studies (A13, A20) uses object detectors with the generic model paradigm. Only one study (A21) uses the category-specific paradigm.

The object models that will be incorporated in the SLAM solution can be generated by online or offline processing. In online processing, the models of the objects are built during the algorithm execution using some heuristic from information given by the object detector. Therefore, in selected studies, this processing is most used when the interest

is in a 3D scene reconstruction with individual object representation (8) with semantic understanding of the scene (A11, A18, A19, A22, A25, A28) or detecting duplicate objects in the scene without previous knowledge (A4). In offline processing, the object models are already built in a previous step, and they are matched using some strategies during algorithm execution. In many cases there must be a prior knowledge of the scene to build the models database (A1,A3,A5,A6,A7,A14,A15,A16,A24,A26). Figure 4 shows the paradigms and the processing type of selected studies.

Figure 4: Object detection paradigms and processing types in the selected studies.



Source: Elaborated by the author (2020).

2.2.2 RQ2: How the methods affected the SLAM approach?

The purpose of this question is to understand which aspects of the integration between 3D object detector and SLAM solution benefited each other.

As mentioned in Section 3.2, the SLAM approaches can be separated in frame-based or keyframe-based. The choice between frame-based and keyframe-based SLAM approaches influences how the object detection methods will be integrated with SLAM. The option to process all frames leads to choose computationally efficient object detectors to achieve real-time processing. On the other hand, object detection methods that use keyframes can be more elaborated because they have more time for processing available since a keyframe generation rate can be adaptable. In the 29 selected studies, 17 used object detection methods that process every frame (A1, A3, A6, A10, A11, A13, A16, A17, A19, A18, A20, A21, A22, A23, A24, A26, A27), but this does not mean that the SLAM associated approach has to be frame-based. Twelve studies used keyframes as object detection input (A2, A4, A5, A7, A8, A9, A12, A14, A15, A25, A28, A29).

All selected studies added some semantic information on the SLAM map. Only 9 (31.0%) studies did not relate any SLAM improvement. The main improvement given by 3D object models was in the localization SLAM task with 19 (65.5%) studies. This was achieved mainly by adding data to the bundle adjustment cost function (Section 3.3) (A1, A4, A5, A6, A14, A15) and SLAM cost function (A8, A9, A16, A17, A20, A21, A25).

2.2.3 RQ3: What techniques are used with these approaches?

The goal of this question is to know the most used techniques among the selected studies and their dependencies.

The main vSLAM system was ORB-SLAM2 (9) used in 7 studies (A2, A8, A9, A12, A23, A25, A28) and his previous version, ORB-SLAM (10) used in 2 studies (A21, A22). The ORB-SLAM2 is an open-source vSLAM for monocular, RGB-D, and stereo camera while ORB-SLAM is only monocular. Twelve studies developed their SLAM system, (A1, A4, A5, A7, A10, A11, A16, A17, A20, A24, A26, A29).

In the researches that use some machine learning techniques, 15 studies used DCNNs, (A2, A8, A9, A10, A12, A13, A17, A18, A19, A20, A21, A23, A25, A27, A28). The most used DCNNs were Mask R-CNN (11) and YOLOv2 (12), with three studies each (A2, A13, A17) and (A19, A21, A28) respectively. Mask R-CNN has three outputs for each object detected, a class label, a bounding box, and a pixel-level mask. YOLOv2 is a fast DCNN capable of real-time processing, and its output is a class label, and a bounding box for each object detected.

The structure most used to represent the object were 3D point clouds. Point cloud is a group of points in a coordinate system, and a point is a structure that storage its position in the space and other relevant data. The 3D point cloud is also the main SLAM map representation in the selected studies, and it is used as object representation in 9 studies (A2, A3, A7, A8, A19, A23, A25, A28, A29).

2.2.4 RQ4: What are the premisses of these approaches?

This question aims to understand the limitations of the methods. The first limitation of all approaches is the hardware. The computational cost of SLAM methods is high, and the hardware should be very robust to achieve real-time performance. In all selected studies, the experiments were made in some of the best CPU and GPU available at that time, when informed in the paper. This fact is most evident in the studies that use neural networks as part of the method.

The performance of vSLAM approaches are affected by image noise, dynamic lighting, motion blur, video defocus, uniform texture image, occlusions, and moving objects. Concerning moving objects, some works went beyond this limitation, developing strategies to remove the features of moving objects, improving the SLAM odometry (A23, A28).

In the template matching-based methods, the first limitation is the prior existence of the object models to be detected. Consequently, there must be prior knowledge of the scene, and it is a limitation in applications that deal with unknown environments. The methods in (A14, A15, A21), the object model was 3D textureless meshes previously

generated. Particularly in (A4), the scene must contain duplicate objects to be detected.

In the machine learning-based approaches, the possible object types to be detected are limited by the training dataset. For example, the MS COCO (13) dataset used to CNN training contains 80 distinct classes, and a small subset of 10 of them is applied in tested indoor scenes (A25). A fine-tuning operation can overcome this limitation, but it demands another image set of the interest class.

2.3 CONCLUSIONS

All the selected studies evidence the complexity of the integration of the two fields. Many methods and heuristics are necessary to achieve a full integration that benefits both approaches, and each study sought to use the best methods available at that time based on its purpose.

According to analyses performed, the most common method was who has the follows features: RGB-D image as input, machine learning-based, object-level output characterization with online processing type.

Among all studies, we assessed what would be the most suitable methods and techniques to achieve the objective of this work. The chosen method and techniques and the its main reasons are listed below:

- ORB-SLAM2 as vSLAM solution:
  ORB-SLAM2 is a state-of-art SLAM system for monocular, RGB-D, and stereo cameras with real-time processing. It has superior performance and accuracy compared to other vSLAM solutions, it has an open-source code available, and we have previous experiences using it.

- YOLOv2 (12) as object detector:
  YOLOv2 was chosen because it is a fast DCNN with the potential for real-time processing, and it has public source code in the same programming language as ORB-SLAM2, which allows us an easy integration. Besides, its recognition results are near to the state-of-art DCNNs in an indoor environment.

- The works of Rubino (14) and Rubino et al. (15) as object pose estimator:
  This pose estimator is based on the conversion of ellipses in different views to an ellipsoid without considering details of surface and shape of objects. This method has a clear description in the doctoral thesis (14), which allowed us a full understanding of the math involved, and we consider the ease of implementation and integration with other chosen solutions. This method is used in the work of Nicholson et. al. (A20) present in this SLR.

Researching the 3D object estimator, we evaluated the use of bounding boxes to estimate a cube that fits the object in (16). However, we chose to use ellipses considering the following aspects. First, the projection of the ellipsoid in an image, independently of perspective, is always an ellipse. There is a linear mathematics relation between ellipses and ellipsoids (Section 3.4), while to reconstruct a cube from bounding boxes, some heuristics are required. Secondly, in the (15) and (A20), the method using ellipses was broadly used in indoor environment, which is aligned with our objectives. While in (16), the method was more applied and discussed in outdoor environments. Finally, we judged, based in the papers, that the method in (15) deals better with a more variability of object shapes than the method in (16).

3 FUNDAMENTALS

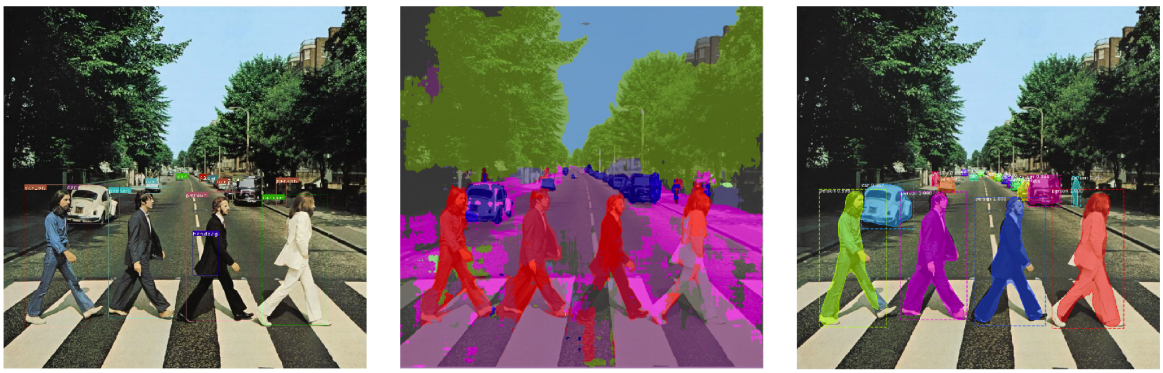
In this chapter it is presented the main fundamentals of this work.

3.1 OBJECT DETECTION

Object detection is the task of locating and labeling instances of objects in images. First, an object is any structure in real world with a classification, such as inanimate objects (e.g., bicycle, chair and traffic sign), buildings (e.g., walls, roads, and houses), living beings (e.g., people, dogs, and trees), and micro or macro body structures (e.g., cells, organelles, eyes, and face).

The task of locating is to determine the limits of object instances detected in the image. There are three main representations of these limits (17): (a) bounding boxes, (b) semantic segmentation, and (c) object instance segmentation (Figure 5). A bounding box is a rectangle aligned with the axes and it fits the object instance detected. In semantic segmentation, each pixel in the image is assigned to a semantic object class, and in object instance segmentation the pixels are grouped in instances.

Figure 5: Examples of object detection types.



(a) Bounding box (b) Semantic segmentation (c) Instance segmentation

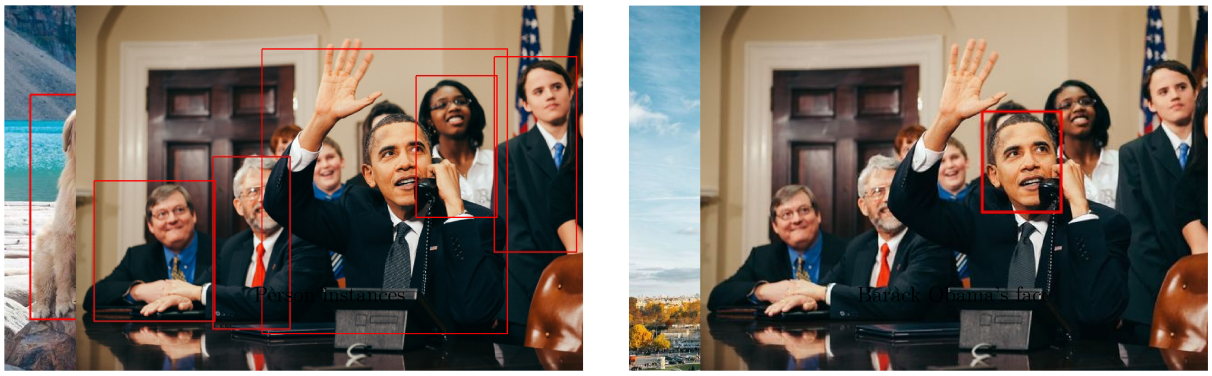
Source: Elaborated by the author (2020).

Labeling is the task to determine which object class the instance detected belongs. According to Zhang et al.(18), this task can be divided into two types: assigning the object instances detected according to a predefined set of object classes or detection of specific object instances which is basically a matching problem (17), as illustrated in Figure 6.

Until the early years of the 2010s, the object detection researches with better performance were based on handcrafted features detectors, local descriptors, and statistical classifiers (17). A broad survey of these researches is given by Zhang et al.(18).

Nowadays, the majority of researches in object class detection has deep convolutional neural networks (DCNNs) as the core method. The recent advances in object class detection with deep learning are well summarized in the following papers (17, 19, 20).

Figure 6: Example of object labeling. (a) labeling all object instances present in images by a predefined set of object classes. (b) labeling the object instances by a specific object label.



(a) Category labeling (b) Specific object labeling

Source: Adapted from (17).

3.2 VISUAL SIMULTANEOUS LOCALIZATION AND MAPPING

A basic SLAM framework involves the extraction of environment features, 3D map construction, localization estimation, and upgrade of 3D map. The features extraction method is dependent on the sensor devices to explore the environment, such as acoustic sensors, laser rangefinders, GPS, inertial measurement units, and cameras (21). A SLAM solution using only a camera is referred to as visual SLAM (vSLAM). vSLAM algorithms have been widely proposed using different camera types, such as monocular, RGB-D, and stereo in recent years.

The visual extraction information can be either direct, indirect, or a hybrid of both. Direct methods use a gradient-based approach to extract information of each pixel in the image generating dense environment information. On the other hand, indirect methods are handcrafted design to extract salient image locations. Indirect methods use feature extractors such as SIFT (22), SURF (23) and ORB (24). Taketomi et al.(3) discuss that direct methods are more robust than indirect methods because they exploit all information available in the image, but they are more sensitive to illumination changes. Besides, modern feature extractors are designed to minimize the sensibility of light conditions, viewpoint variation, and to be computationally efficient.

The SLAM approaches can be separated in frame-based or keyframe-based concerning which information is processed and stored. A keyframe is a frame that is slightly different from its predecessor through a set of rules to represent a new location (25). In the frame-based SLAM, all frames are used in all SLAM steps. The keyframe-based SLAM approaches use the keyframe to reduce information redundancy and to decrease the computational cost. Therefore, the choice between frame-based or keyframe-based SLAM approaches influences in how the object detection methods will be integrated with SLAM.

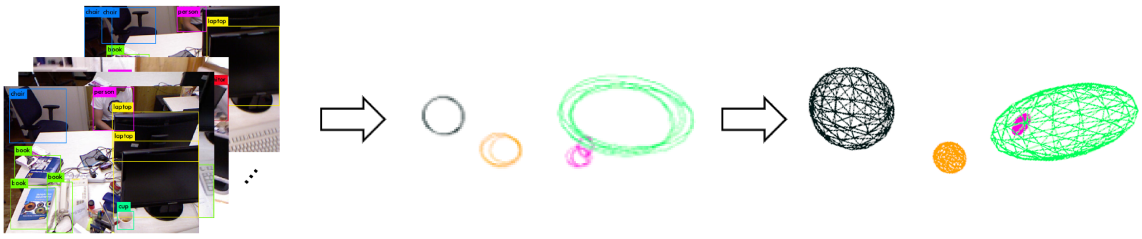
3.3 BUNDLE ADJUSTMENT

Bundle Adjustment (BA) is a batch optimization method to collectively estimate camera parameters and the 3D scene structure to minimize cost function (26). This method can be applied to minimize the local or global camera localization error, i.e. improving the odometry. Odometry is the use of sensor data for a mobile computer system to determine and update its position over time in an environment. The terms in this cost function are the geometric relationships among the SLAM elements. Typically, these terms use the camera pose parameters and the parameters involving the scene geometry domain changes like the re-projection error of the entities in the SLAM map.

3.4 CONIC SECTIONS AND QUADRIC SURFACES

As presented in conclusion of Chapter 2, the 3D object pose estimator chosen uses ellipses in different views to reconstruct ellipsoids. And the ellipses are obtained from object detector output (Figure 7). Therefore, it is essential to formalize the concepts of conics sections and quadrics surfaces. These concepts will be important to the method description in Chapter 4. The fundamentals presented in this section are based on the thesis of Rubino (14).

Figure 7: Basic steps of 3D object pose estimator. The object detector process the images and generates bounding boxes as output (a), the ellipses that fits the bounding boxes are calculated (b), and the ellipsoids are estimated (c).



(a) (b) (c)

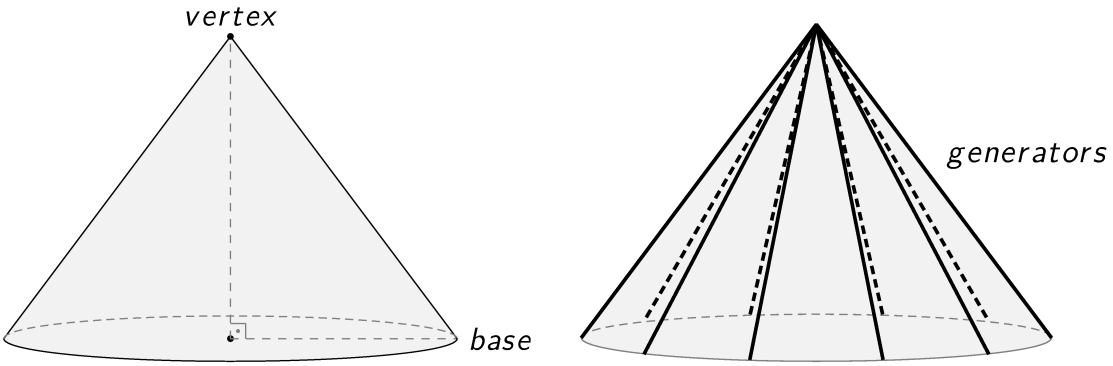
Source: Elaborated by the author (2020).

3.4.1 Conic sections

A right circular cone is a 3D surface that has a circle as a base and the vertex is right above the center of the circle (Figure 8a). Generators are the lines that intercept the vertex and the base (27) (Figure 8b).

The conic sections, or conics, are 2D surfaces in Euclidean spaces generated in the intersection between a double right circular cone and a plane in different orientations. The conics can be divided into degenerated and non-degenerated groups depending on how the plane intercepts the double cone. A degenerate conics arises when a plane intercepts a

Figure 8: The right circular cone and example of generators.



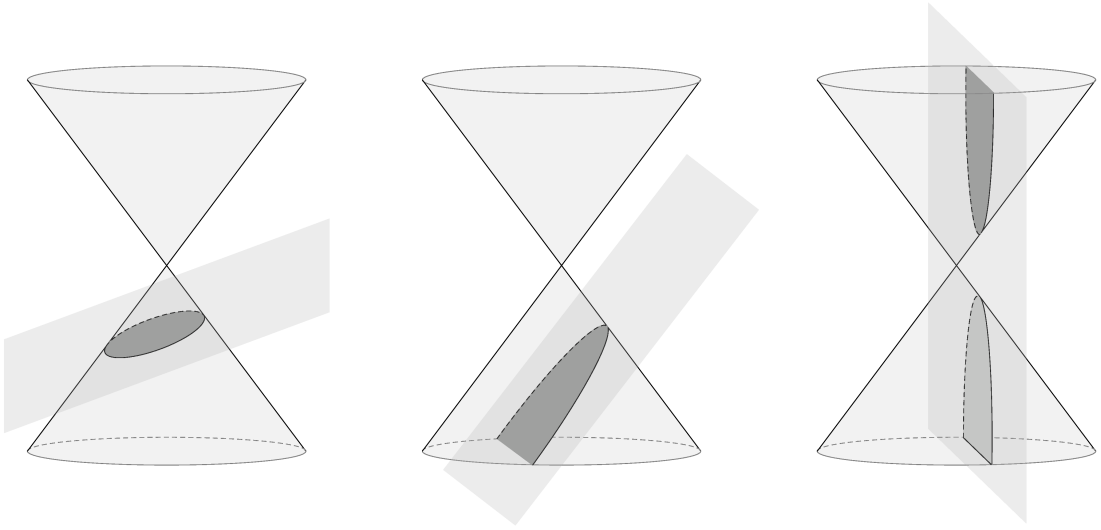
(a) Right circular cone (b) Generators in a right circular cone

Source: Elaborated by the author (2020).

double right circular cone in the vertex. The main types of degenerate conics are a point, line and cross line. Degenerate conics will not be discussed in this work.

The main types of non-degenerate conics are the ellipse, parabola and hyperbole. If a plane fully intercepts all generators of one of two right circular cones, the closed curve generated is an ellipse (Figure 9a). When a plane intercepts the double right circular cone and it is parallel to one generator, the curve in the interception is a parabola, Figure 9b. A hyperbole arises when a plane intercepts the dual right circular cone and it is parallel to two generators (Figure 9c).

Figure 9: The three non-degenerate conics in the intersection of a double right circular cone and plans.



(a) Ellipse (b) Parabola (c) Hyperbole

Source: Elaborated by the author (2020).

A conic curve can be represented analytically by the following second-order equation:

$$ax_1^2 + bx_1x_2 + cx_2^2 + dx_1 + ex_2 + f = 0. \tag{3.1}$$

where $\{a, b, c, d, e, f\} \in \mathbb{R}$ are the coefficients of the equation. Equation 3.1 also represents a set of points using homogeneous coordinates:

$$\mathcal{C} = \{\mathbf{x} \in \mathbb{R}^3 \,|\, \mathbf{x}^{\mathrm{T}}\mathtt{C}\mathbf{x} = 0\}, \tag{3.2}$$

where $\mathbf{x}$ is a homogeneous 2D point belonging to the conic and $\mathtt{C} \in \mathbb{R}^{3\times3}$ is the symmetric coefficient matrix of the conic in the form:

$$\mathtt{C} = \begin{bmatrix} a & b/2 & d/2 \\ b/2 & c & e/2 \\ d/2 & e/2 & f \end{bmatrix}. \tag{3.3}$$

The matrix $\mathtt{C}$ has five degrees of freedom: two for magnitude of semi-axes, two for position and one for orientation. The coefficient matrix $\mathtt{C}$ associated with the conic will represent the conic itself in the following parts of this work.

3.4.1.1 Ellipse

It is possible to determine the type of a conic from matrix $\mathtt{C}$. An ellipse, in particular, should fulfill the following requirements:

$$\begin{cases} \det \mathtt{C} \neq 0 \\ \det \mathtt{C}_{22} > 0 \\ \frac{\det \mathtt{C}}{\det \mathtt{C}_{22}} < 0 \end{cases}, \text{ where } \mathtt{C}_{22} = \begin{bmatrix} a & b/2 \\ b/2 & c \end{bmatrix}, \tag{3.4}$$

where $\det \mathtt{C}$ and $\det \mathtt{C}_{22}$ are the determinants of the matrices $\mathtt{C}$ and $\mathtt{C}_{22}$, respectively.

Some geometric elements of an ellipse (Figure 10) can be computed from the matrix $\mathtt{C}$. The center $\mathbf{x}_c = [\mathrm{x}_{1c}\ \mathrm{x}_{2c}]^{\mathrm{T}}$ of the conic is obtained as follows:

$$\mathbf{x}_c = -\mathtt{C}_{22}^{-1} \cdot \begin{bmatrix} d/2 \\ e/2 \end{bmatrix}, \tag{3.5}$$

The sizes of semi-axes $\boldsymbol{\xi} = [\xi_1\ \xi_2]^{\mathrm{T}}$ are calculated as follows:

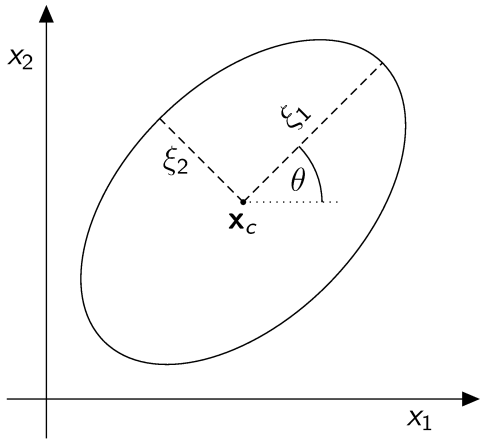
$$\boldsymbol{\xi} = \sqrt{-\frac{\det \mathtt{C}}{\det \mathtt{C}_{22}} \begin{bmatrix} \lambda_1^{-1} \\ \lambda_2^{-1} \end{bmatrix}}, \tag{3.6}$$

where λ_1 and λ_2 are the eigenvalues of the matrix $\mathtt{C}_{22}$. The orientation matrix $\mathtt{R} \in \mathbb{R}^{2\times2}$ is obtained by computing the normalized eigenvectors $\boldsymbol{\upsilon}_1$ and $\boldsymbol{\upsilon}_2$ of the matrix $\mathtt{C}_{22}$ as follows:

$$\mathtt{R} = \begin{bmatrix} \boldsymbol{\upsilon}_1 & \boldsymbol{\upsilon}_2 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \tag{3.7}$$

where $\boldsymbol{\upsilon}_i = [\upsilon_{i1}\ \upsilon_{i2}]^{\mathrm{T}}$ and θ is the angle relative to the abscissa axis about the origin of a two-dimensional Cartesian coordinate system.

Figure 10: Conic geometric elements exemplified in an ellipse. ξ_1 and ξ_2 are the semi-axes, θ is the angle relative to abscissa axis and $\mathbf{x}_c$ is the center vector.



Source: Elaborated by the author (2020).

3.4.2 Quadric Surfaces

A quadric is a 3D surface which can be defined by the following second-order analytical equation:

$$ax_1^2 + bx_1x_2 + cx_1x_3 + dx_2^2 + ex_2x_3 + fx_3^2 + gx_1 + hx_2 + ix_3 + j = 0. \tag{3.8}$$

where $\{a, b, c, d, e, f, g, h, i, j\} \in \mathbb{R}$ are the coefficients of the equation. Examples of quadrics are ellipsoid, elliptic paraboloid, hyperbolic paraboloid and hyperboloid of one or two sheets, and cylinders. Equation 3.11 can be rewritten as a set of points using homogeneous coordinates:

$$\mathcal{Q} = \{\mathbf{x} \in \mathbb{R}^4 \mid \mathbf{x}^{\mathrm{T}}\mathtt{Q}\mathbf{x} = 0\}, \tag{3.9}$$

where $\mathbf{x}$ is a homogeneous 3D point belonging to the quadric and $\mathtt{Q} \in \mathbb{R}^{4\times4}$ is the symmetric coefficient matrix of the quadric in the form:

$$\mathtt{Q} = \begin{bmatrix} a & b/2 & c/2 & g/2 \\ b/2 & d & e/2 & h/2 \\ c/2 & e/2 & f & i/2 \\ g/2 & h/2 & i/2 & j \end{bmatrix}. \tag{3.10}$$

The matrix $\mathtt{Q}$ has nine degrees of freedom: three for magnitude of the semi-axes, three for position and three for orientation. The coefficient matrix $\mathtt{Q}$ associated with the quadric will represent the quadric itself in the following parts of this work.

3.4.2.1 Ellipsoid

Likewise for conics, it is possible to determine the type of quadric from the matrix $\mathtt{Q}$. An ellipsoid, in particular, should fulfill the following requirements:

$$\begin{cases} \det \mathtt{Q} < 0 \\ \det \mathtt{Q}_{33} \neq 0 \\ \frac{\lambda_1}{|\lambda_1|} = \frac{\lambda_2}{|\lambda_2|} = \frac{\lambda_3}{|\lambda_3|} \end{cases}, \text{ where } \mathtt{Q}_{33} = \begin{bmatrix} a & b/2 & c/2 \\ b/2 & d & e/2 \\ c/2 & e/2 & f \end{bmatrix}, \tag{3.11}$$

$\det \mathtt{Q}$ and $\det \mathtt{Q}_{33}$ are the determinants of matrix $\mathtt{Q}$ and $\mathtt{Q}_{33}$, respectively, and λ_1, λ_2 and λ_3 are the eigenvalues associated with $\mathtt{Q}_{33}$.

As with the ellipse, some geometric elements of the ellipsoid can be computed from the matrix $\mathtt{Q}$. The center of the ellipsoid $\mathbf{x}_c = [\mathrm{x}_{1c}\ \mathrm{x}_{2c}\ \mathrm{x}_{3c}]^{\mathrm{T}}$ is obtained as follows:

$$\mathbf{x}_c = -\mathtt{Q}_{33}^{-1} \cdot \begin{bmatrix} g/2 \\ h/2 \\ i/2 \end{bmatrix}. \tag{3.12}$$

The size of semi-axes $\boldsymbol{\xi} = [\xi_1\ \xi_2\ \xi_3]^{\mathrm{T}}$ can be calculated as follows:

$$\boldsymbol{\xi} = \sqrt{-\frac{\det \mathtt{Q}}{\det \mathtt{Q}_{33}} \begin{bmatrix} \lambda_1^{-1} \\ \lambda_2^{-1} \\ \lambda_3^{-1} \end{bmatrix}}, \tag{3.13}$$

where λ_1, λ_2 and λ_3 are the eigenvalues of the matrix $\mathtt{Q}_{33}$. The orientation matrix $\mathtt{R} \in \mathbb{R}^{3\times3}$ is obtained by computing the normalized eigenvectors $\boldsymbol{v}_1, \boldsymbol{v}_2$ and $\boldsymbol{v}_3$ of the matrix $\mathtt{Q}_{33}$ as follows:

$$\mathtt{R} = \begin{bmatrix} \boldsymbol{v}_1 & \boldsymbol{v}_2 & \boldsymbol{v}_3 \end{bmatrix} = \begin{bmatrix} \cos\beta\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \cos\beta\sin\gamma & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma \\ -\sin\beta & \sin\alpha\cos\beta & \cos\alpha\cos\beta \end{bmatrix}, \tag{3.14}$$

where $\boldsymbol{v}_i = [v_{i1}\ v_{i2}\ v_{i3}]^{\mathrm{T}}$ and α, β and γ are the angles relative to the X-axis, Y-axis and Z-axis, respectively, about the origin of a three-dimensional Cartesian coordinate system.

3.4.3 Dual conics and dual quadrics

A conic can be represented by an equation of points, primal representation (Equation 3.2), or lines, dual representation (27). To achieve the line representation, each point $\mathbf{x}$ in the conic has a tangent line $\mathbf{l}$ given by:

$$\mathbf{l} = \mathtt{C}\mathbf{x}. \tag{3.15}$$

Using the inverse of 3.15, $\mathbf{x} = \mathtt{C}^{-1}\mathbf{l}$, in 3.2 we find the equation of a line conic as follows:

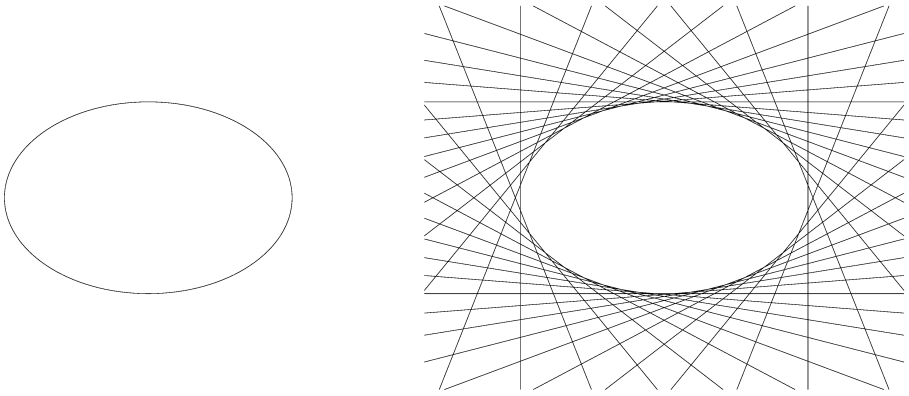
$$(\mathtt{C}^{-1}\mathbf{l})^{\mathrm{T}}\mathtt{C}(\mathtt{C}^{-1}\mathbf{l}) = \mathbf{l}^{\mathrm{T}}\mathtt{C}^{-^{\mathrm{T}}}\mathtt{C}\mathtt{C}^{-1}\mathbf{l} = \mathbf{l}^{\mathrm{T}}\mathtt{C}^{-^{\mathrm{T}}}\mathbf{l} = 0. \tag{3.16}$$

The matrix $\mathtt{C}$ is symmetric and therefore $\mathtt{C}^{-\mathrm{T}} = \mathtt{C}^{-1}$ and the set of tangent lines $\mathcal{L}$ can be defined as an equation of lines:

$$\mathcal{L} = \{\mathbf{l} \in \mathbb{R}^3 \,|\, \mathbf{l}^{\mathrm{T}}\, \mathtt{D}\, \mathbf{l} = 0\}, \tag{3.17}$$

where $\mathtt{D} = \mathtt{C}^{-1}$ and $\mathtt{D} \in \mathbb{R}^{3\times3}$ is the dual conic of $\mathtt{C}$. The adjoint matrix often represents the dual conic instead of its inverse because of $\mathtt{C}^{-1} = \mathrm{adjoint}(\mathtt{C}) \setminus \det(\mathtt{C})$. The advantage of the adjoint compared to the inverse is that it is also computable if $\mathtt{C}$ is not invertible (28).

Figure 11: (a) Points $\mathbf{x}$ satisfying $\mathbf{x}^{\mathrm{T}}\mathtt{C}\mathbf{x} = 0$ lie on a point conic. (b) Lines $\mathbf{l}$ satisfying $\mathbf{l}^{\mathrm{T}}\mathtt{D}\mathbf{l} = 0$ are tangent to the point conic $\mathtt{C}$. The conic $\mathtt{C}$ is the envelope of the lines $\mathbf{l}$.



(a) Point conic (b) Line conic d

Source: Hartley and Zisserman(27).

As with the dual conics, a quadric can be represented by an equation of points, primal representation (Equation 3.8), or an equation of planes, dual representation. To achieve the plane representation, each point $\mathbf{x}$ in homogeneous coordinates in the quadric has a tangent plane $\boldsymbol{\rho}$ given by:

$$\boldsymbol{\rho} = \mathtt{Q}\, \mathbf{x}. \tag{3.18}$$

The same mathematical manipulation seen in Equation 3.16, can be done using Equation 3.9 and we obtain the following set of tangent planes $\mathcal{P}$:

$$\mathcal{P} = \{\boldsymbol{\rho} \in \mathbb{R}^4 \,|\, \boldsymbol{\rho}^{\mathrm{T}}\, \mathtt{E}\, \boldsymbol{\rho} = 0\}, \tag{3.19}$$

where $\mathcal{P}$ is the set of planes that envelopes the quadric, $\mathtt{E} \in \mathbb{R}^{4\times4}$ is the dual quadric of $\mathtt{Q}$. The adjoint matrix can be used as dual quadric likewise dual conic.

The relation between conic and quadric is nonlinear in primal space. But between dual conic and dual quadric, the relation is linear (29). Demonstrating this relation, consider a dual quadric $\mathtt{Q}$ in four-dimensional space and a dual conic $\mathtt{D}$ in a three-dimensional space (Figure 12). For each line tangent $\mathbf{l}$ to $\mathtt{D}$, which satisfies Equation 3.17, can be back-projected to a plane $\boldsymbol{\rho}$, which satisfies Equation 3.19, throw a linear transformation:

$$\boldsymbol{\rho} = \mathtt{P}^{\mathrm{T}}\mathbf{l}, \tag{3.20}$$

where $\mathtt{P} \in \mathbb{R}^{3\times4}$ is a projection matrix of a four-dimensional subspace to three-dimensional subspace. Applying the last result in Equation 3.19, we obtain:

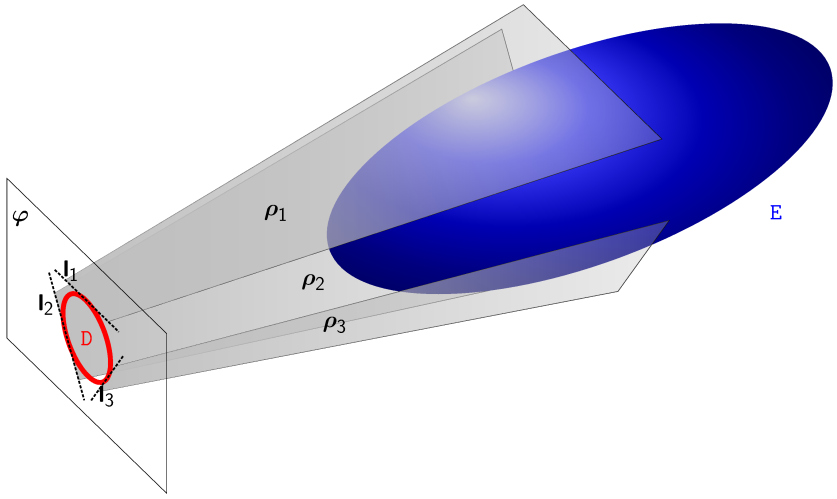
$$\boldsymbol{\rho}^{\mathrm{T}}\mathtt{E}\boldsymbol{\rho} = \mathbf{l}^{\mathrm{T}}\mathtt{P}\,\mathtt{E}\,\mathtt{P}^{\mathrm{T}}\mathbf{l} = \mathbf{l}^{\mathrm{T}}\,\mathtt{D}\,\mathbf{l} = 0, \tag{3.21}$$

and we derive the expression:

$$\mathtt{D} = \mathtt{P}\,\mathtt{E}\,\mathtt{P}^{\mathrm{T}}. \tag{3.22}$$

Equation 3.22 describes the linear relation between the dual space of conics and the dual space of quadrics.

Figure 12: Relation between dual conics and dual quadrics. The ellipse $\mathtt{D}$ is tangent to three generic lines $\mathbf{l}_1$, $\mathbf{l}_2$ and $\mathbf{l}_3$, which belong to three generic planes $\boldsymbol{\rho}_1$, $\boldsymbol{\rho}_2$ and $\boldsymbol{\rho}_3$ tangent to the ellipsoid $\mathtt{E}$.



Source: Rubino(14).

3.4.4 Rigid transformations

In dual space, to apply a rigid transformation in a line l, it is necessary to multiply with a roto-translation matrix H given by:

$$\mathbf{l}' = \mathtt{H}^{\mathrm{T}}\,\mathbf{l}, \tag{3.23}$$

where the matrix $\mathtt{H} \in \mathbb{R}^{3\times3}$ is in the form

$$\mathtt{H}(\phi, \mathbf{t}_2) = \begin{bmatrix} \mathtt{R}_2(\phi) & \mathbf{t}_2 \\ \mathbf{0}_2^{\mathrm{T}} & 1 \end{bmatrix}, \tag{3.24}$$

where $\mathbf{t}_2 = [t_x\, t_y]^{\mathrm{T}}$ is the translation vector and $\mathtt{R}_2(\phi) \in \mathbb{R}^{2\times2}$ is the rotation matrix.

Applying Equation 3.23 on Equation 3.16 we have:

$$\mathbf{l}'^{\mathrm{T}}\mathtt{D}\mathbf{l}' = (\mathtt{H}^{\mathrm{T}}\mathbf{l})^{\mathrm{T}}\mathtt{D}(\mathtt{H}^{\mathrm{T}}\mathbf{l}) = \mathbf{l}^{\mathrm{T}}\mathtt{H}\mathtt{D}\mathtt{H}^{\mathrm{T}}\mathbf{l} = 0, \tag{3.25}$$

then the dual conic roto-translated $\mathtt{D}'$ is given by:

$$\mathtt{D}' = \mathtt{H}\mathtt{D}\mathtt{H}^{\mathrm{T}}. \tag{3.26}$$

Using Equation 3.26, the function $\mathtt{D}(\phi, \mathbf{t}_2, \boldsymbol{\xi}_2)$ can represent an ellipse as follows:

$$\mathtt{D}(\phi, \mathbf{t}_2, \boldsymbol{\xi}_2) = \begin{bmatrix} \mathtt{R}_2(\phi) & \mathbf{t}_2 \\ \mathbf{0}_2^{\mathrm{T}} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{\xi}_2 & \mathbf{0}_2 \\ \mathbf{0}_2^{\mathrm{T}} & -1 \end{bmatrix} \begin{bmatrix} \mathtt{R}_2(\phi) & \mathbf{0}_2 \\ \mathbf{t}_2^{\mathrm{T}} & 1 \end{bmatrix}, \tag{3.27}$$

where $\boldsymbol{\xi}_2 = [\xi_1 \ \xi_2]^{\mathrm{T}}$ is the vector of semi-axes of $\mathtt{D}$.

The same logic can be applied on dual quadrics. In this case, the matrix associated with the rigid transformation is $\mathtt{Z} \in \mathbb{R}^{4\times4}$ as:

$$\mathtt{Z}(\boldsymbol{\theta}, \mathbf{t}_3) = \begin{bmatrix} \mathtt{R}_3(\boldsymbol{\theta}) & \mathbf{t}_3 \\ \mathbf{0}_3^{\mathrm{T}} & 1 \end{bmatrix}, \tag{3.28}$$

where $\mathbf{t}_3 = [t_x \ t_y \ t_z]$ is the translation vector and $\mathtt{R}_3(\boldsymbol{\theta}) \in \mathbb{R}^{3\times3}$ is the rotation matrix $\mathtt{R}(\boldsymbol{\theta}) = \mathtt{R}_z(\theta_3)\mathtt{R}_y(\theta_2)\mathtt{R}_x(\theta_1)$. The roto-translated dual quadric $\mathtt{E}' \in \mathbb{R}^{4\times4}$ is calculated by the equation:

$$\mathtt{E}' = \mathtt{Z}\,\mathtt{E}\,\mathtt{Z}^{\mathrm{T}}. \tag{3.29}$$

Using Equation 3.29, the function $\mathtt{E}(\boldsymbol{\theta}, \mathbf{t}_3, \boldsymbol{\xi}_3)$ can represents any ellipsoid as follows:

$$\mathtt{E}(\boldsymbol{\theta}, \mathbf{t}_3, \boldsymbol{\xi}) = \begin{bmatrix} \mathtt{R}_3(\boldsymbol{\theta}) & \mathbf{t}_3 \\ \mathbf{0}_3^{\mathrm{T}} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{\xi}_3 & \mathbf{0}_3 \\ \mathbf{0}_3^{\mathrm{T}} & -1 \end{bmatrix} \begin{bmatrix} \mathtt{R}_3(\boldsymbol{\theta}) & \mathbf{0}_3 \\ \mathbf{t}_3^{\mathrm{T}} & 1 \end{bmatrix}, \tag{3.30}$$

where $\boldsymbol{\xi}_3 = [\xi_1 \ \xi_2 \ \xi_3]^{\mathrm{T}}$ is the vector of semi-axes of $\mathtt{E}$.

4 PROPOSED METHOD

In this chapter, we describe our solution for the problem of monocular object SLAM. This solution was developed over the work of Rubino (14) and Rubino et al.(15) as object pose estimator and ORB-SLAM2 (9) as SLAM solution. The main reasons for using these methods are described in Section 2.3. The monocular ORB-SLAM2 will be referred as ORB-SLAM2 in the following parts of this work.

The ORB-SLAM2 has three main parallel threads: Tracking, Local Mapping, and Loop Closing. The Tracking thread is responsible for extracting the features from frames, matching the features to the local map, execute motion-only BA, localize the camera, and decide the new keyframe. The Local Mapping thread manages the points map and optimizes it, creating new points or culling it, performing the local BA and culling keyframes. Finally, the Loop Closing thread detects loop closing and performs a pose-graph optimization. This thread launches a fourth thread when a loop closed is detected to execute a full BA.

The proposed method adds two other threads to ORB-SLAM2. The first thread executes the 2D object detection on every keyframe, creates the ellipses associated with each region detected, and executes the full BA (Section 4.3). The second thread performs the object association strategy (Section 4.2), executes the 3D pose optimization for objects which new data was added on it (Section 4.1) and updates the objects map. Figure 13 shows an overview of the developed system. The red labels highlight our developments or modifications. The next sections describe the developments and modifications.

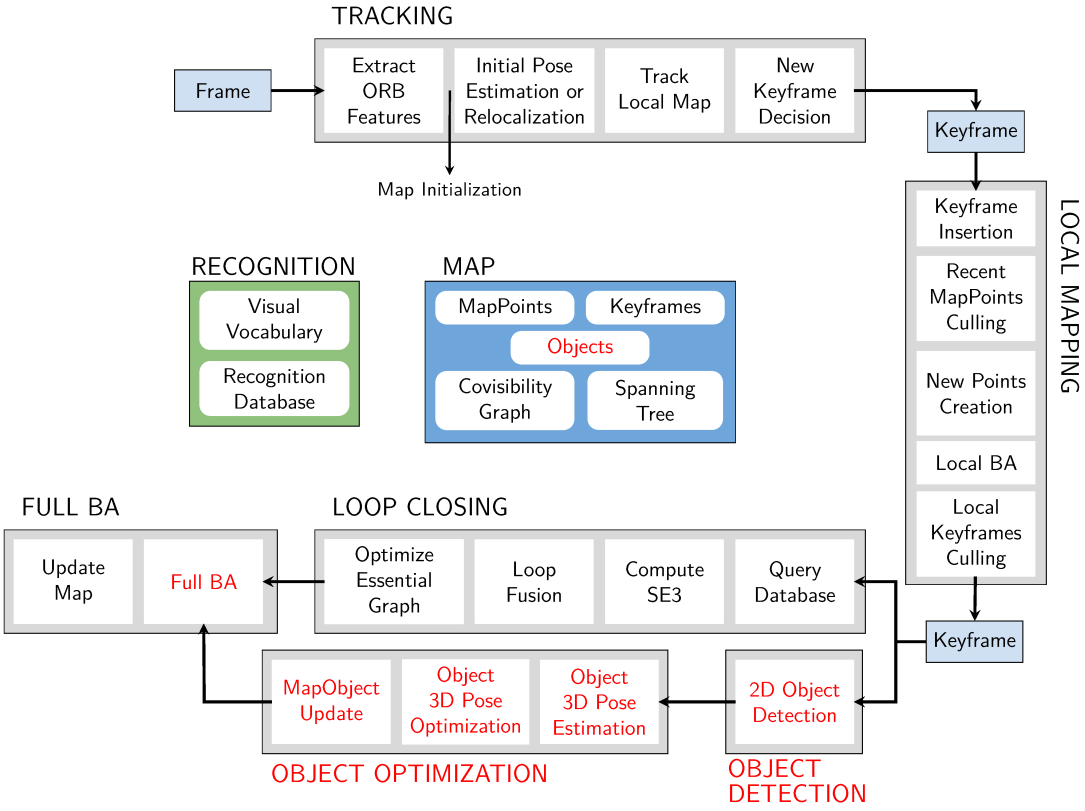
4.1 3D OBJECT POSE ESTIMATION AND OPTIMIZATION

The method described in this section, emphasizing once again, is entirely part of the work presented in Rubino (14, 15). We just added a better detail of the algebra involved.

Consider a static scene as a sequence $f = 1, \ldots, F$ of multiple image frames from different perspectives. After the processing by a generic object detector, each image frame contains a set of regions $i = 1, \ldots, N$. Each region is delimited by bounding boxes b_{fi} and has a set of classes with its respective probabilities. The bounding box is defined as $b_{fi} = \{w_{fi}, h_{fi}, \mathbf{c}_{if}\}$, where w_{fi} and h_{fi} are respectively the width and height, and the vector $\mathbf{c}_{fi}$ is the center. For each bounding box b_{fi}, there is an ellipse $\mathtt{C}_{fi}$ inscribed in the bounding box, as illustrated in Figure 14, where the centers are matched, and w_{fi} and h_{fi} are the axes of ellipsoid aligned with the image axes.

The core of the proposed object detection is to reconstruct an ellipsoid $\mathtt{Q}_i$ from a set of ellipses $\mathcal{C}_i = \{\mathtt{C}_{fi}\}$ of different views of the same object i. As already stated in Section 3.1, the relation between $\mathtt{Q}_i$ and $\mathtt{C}_{fi}$ is nonlinear in primal space. However,

Figure 13: Overview of the system developed. The red labels highlights our developments or modifications in ORB-SLAM2.



Source: Adapted from (10).

the relation between the dual quadric, represented by the matrix $\mathtt{E}_i$, and the dual conic, represented by the matrix $\mathtt{D}_{fi}$, is linear in dual space. Consider a set of frames $\mathcal{F} = 1, \ldots, f$ taken with a calibrated camera, i.e., there is a camera matrix for each frame. In the Equation 3.22, $\mathtt{P}_f \in \mathbb{R}^{3\times4}$ is the camera matrix and it can be written:

$$\mathtt{P}_f = \mathtt{K}[\mathtt{R}_f|\mathbf{t}_f], \tag{4.1}$$

where $\mathtt{K} \in \mathbb{R}^{3\times3}$ is the camera intrinsic parameters matrix, while $\mathtt{R}_f \in \mathbb{R}^{3\times3}$ and $\mathbf{t}_f \in \mathbb{R}^3$ are respectively the rotation matrix and translation vector for each frame f, which are known from the SLAM solution. Thus, the Equation 3.22 can be reformulated to:

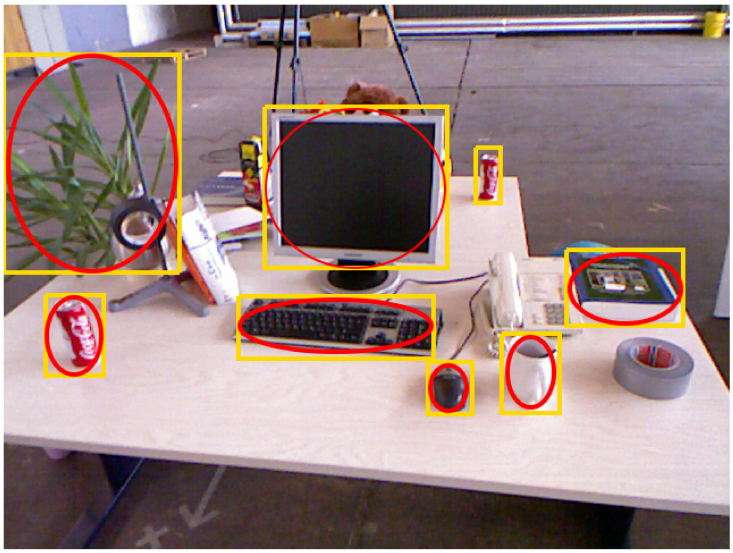
$$\beta_{fi}\mathtt{D}_{fi} = \mathtt{P}_f\mathtt{E}_i\mathtt{P}_f^{\mathrm{T}} \tag{4.2}$$

where $\beta_{fi} \in \mathbb{R}$ is a scale factor.

To achieve the linear relation between $\mathtt{D}_{fi}$ and $\mathtt{E}_i$, we used the following well-know vectorization operations over a symmetric matrix $\mathtt{X} \in \mathbb{R}^{m\times m}$:

$$\begin{aligned} vec(\mathtt{X}) &= [x_{1,1}, \ldots, x_{m,1}, x_{1,2}, \ldots, x_{m,2}, \ldots, x_{1,n}, \ldots, x_{m,m}]^{\mathrm{T}} \\ vech(\mathtt{X}) &= [x_{1,1}, x_{2,1}, x_{2,2}, x_{3,1}, x_{3,2}, x_{3,3}, \ldots, x_{m,1}, \ldots, x_{m,m}]^{\mathrm{T}}, \end{aligned} \tag{4.3}$$

Figure 14: Example of bounding boxes (yellow) and corresponding fitted ellipses (red) for a set of objects.



Source: Adapted from (14).

and the following property (30):

$$vec(\mathtt{ABC}) = (\mathtt{C}^{\mathrm{T}} \otimes \mathtt{A})vec(\mathtt{B}), \tag{4.4}$$

where $\mathtt{A}, \mathtt{B}$, and $\mathtt{C}$ are suitable dimensioned matrices, and $\otimes$ is the Kronecker product.

Applying the property 4.4 in Equation 4.2 gives us:

$$\begin{aligned}
vec(\beta_{fi}\,\mathtt{D}_{fi}) &= vec(\mathtt{P}_f\,\mathtt{E}_i\,\mathtt{P}_f^{\mathrm{T}}) \\
\beta_{fi}\,vec(\mathtt{D}_{fi}) &= (\mathtt{P}_f \otimes \mathtt{P}_f)\,vec(\mathtt{E}_i) \\
\beta_{fi}\,vec(\mathtt{D}_{fi}) &= (\mathtt{P}_f \otimes \mathtt{P}_f)\,\mathtt{U}\,vech(\mathtt{E}_i) \\
\beta_{fi}\,\mathtt{J}\,vec(\mathtt{D}_{fi}) &= \mathtt{J}\,(\mathtt{P}_f \otimes \mathtt{P}_f)\,\mathtt{U}\,vech(\mathtt{E}_i) \\
\beta_{fi}\,vech(\mathtt{D}_{fi}) &= \mathtt{J}\,(\mathtt{P}_f \otimes \mathtt{P}_f)\,\mathtt{U}\,vech(\mathtt{E}_i).
\end{aligned} \tag{4.5}$$

Simplifying the result in 4.5, we have:

$$\beta_{fi}\mathbf{c}_{fi} = \mathtt{G}_f\mathbf{v}_i, \tag{4.6}$$

where $\mathbf{c}_{fi} \in \mathbb{R}^6$ and $\mathbf{v}_i \in \mathbb{R}^{10}$ are respectively the vectorized forms of the dual symmetric matrices $\mathtt{D}_{fi}$ and $\mathtt{E}_i$ using $vech$ operation, $\mathtt{J} \in \mathbb{R}^{6\times9}$ and $\mathtt{U} \in \mathbb{R}^{16\times10}$ are two matrices such that $vech(\mathtt{X}) = \mathtt{J}vec(X)$ and $vec(\mathtt{Y}) = \mathtt{U}vech(\mathtt{Y})$ respectively, where $\mathtt{X} \in \mathbb{R}^{9\times9}$ and $\mathtt{Y} \in \mathbb{R}^{16\times16}$ are two symmetric matrices. The matrix $\mathtt{G} \in \mathbb{R}^{6\times10}$ where $\mathtt{G}_f = \mathtt{J}(\mathtt{P}_f \otimes \mathtt{P}_f)\mathtt{U}$ has a patterned structure and it is made explicit in Equation 4.7.

$$\mathtt{G} = \begin{bmatrix}
p_{11}^2 & 2p_{11}p_{12} & p_{12}^2 & 2p_{11}p_{13} & 2p_{12}p_{13} & p_{13}^2 & 2p_{11}p_{14} & 2p_{12}p_{14} & 2p_{13}p_{14} & p_{14}^2 \\
p_{21}p_{11} & p_{21}p_{12}+p_{22}p_{11} & p_{22}p_{12} & p_{21}p_{13}+p_{23}p_{11} & p_{22}p_{13}+p_{23}p_{12} & p_{23}p_{13} & p_{21}p_{14}+p_{24}p_{11} & p_{22}p_{14}+p_{24}p_{12} & p_{23}p_{14}+p_{24}p_{13} & p_{24}p_{14} \\
p_{21}^2 & 2p_{21}p_{22} & p_{22}^2 & 2p_{21}p_{23} & 2p_{22}p_{23} & p_{23}^2 & 2p_{21}p_{24} & 2p_{22}p_{24} & 2p_{23}p_{24} & p_{24}^2 \\
p_{31}p_{11} & p_{31}p_{12}+p_{32}p_{11} & p_{32}p_{12} & p_{31}p_{13}+p_{33}p_{11} & p_{32}p_{13}+p_{33}p_{12} & p_{33}p_{13} & p_{31}p_{14}+p_{34}p_{11} & p_{32}p_{14}+p_{34}p_{12} & p_{33}p_{14}+p_{34}p_{13} & p_{34}p_{14} \\
p_{31}p_{21} & p_{31}p_{22}+p_{32}p_{21} & p_{32}p_{22} & p_{31}p_{23}+p_{33}p_{21} & p_{32}p_{23}+p_{33}p_{22} & p_{33}p_{23} & p_{31}p_{24}+p_{34}p_{21} & p_{32}p_{24}+p_{34}p_{22} & p_{33}p_{24}+p_{34}p_{23} & p_{34}p_{24} \\
p_{31}^2 & 2p_{31}p_{32} & p_{32}^2 & 2p_{31}p_{33} & 2p_{32}p_{33} & p_{33}^2 & 2p_{31}p_{34} & 2p_{32}p_{34} & 2p_{33}p_{34} & p_{34}^2
\end{bmatrix}, \tag{4.7}$$

At least three images in different views of object i are necessary to obtain a unique solution for $\mathbf{v}_i$ (31). Thus, we can stacking the Equation 4.6 for $f = 1, \ldots, F$, with $F \geq 3$ to the following system of equations:

$$\begin{cases} \mathtt{G}_1\mathbf{v}_i - \beta_{1i}\mathbf{c}_{1i} = 0_6 \\ \mathtt{G}_2\mathbf{v}_i - \beta_{2i}\mathbf{c}_{2i} = 0_6 \\ \vdots \\ \mathtt{G}_F\mathbf{v}_i - \beta_{Fi}\mathbf{c}_{Fi} = 0_6 \end{cases}. \tag{4.8}$$

The system of equations can be rewritten as a matricial equation:

$$\mathtt{M}_i\mathbf{w}_i = 0_{6F}, \tag{4.9}$$

where 0_{6F} is a column vector of length $6F$, the matrix $\mathtt{M}_i \in \mathbb{R}^{6F\times(10+F)}$ and the vector $\mathbf{w} \in \mathbb{R}^{10+F}$ are defined as follows:

$$\mathtt{M}_i = \begin{bmatrix} \mathtt{G}_1 & -\mathbf{c}_{1i} & 0_6 & 0_6 & \cdots & 0_6 \\ \mathtt{G}_2 & 0_6 & -\mathbf{c}_{2i} & 0_6 & \cdots & 0_6 \\ \mathtt{G}_3 & 0_6 & 0_6 & -\mathbf{c}_{3i} & \cdots & 0_6 \\ \vdots & 0_6 & 0_6 & 0_6 & \ddots & 0_6 \\ \mathtt{G}_F & 0_6 & 0_6 & 0_6 & \cdots & -\mathbf{c}_{Fi} \end{bmatrix}, \quad \mathbf{w}_i = \begin{bmatrix} \mathbf{v}_i \\ \beta_i \end{bmatrix}, \tag{4.10}$$

where the vector $\beta_i = [\beta_{1i}, \ldots, \beta_{Fi}]^{\mathrm{T}}$.

Using the first ten elements of $\mathbf{w}_i$, we have the vector $\mathbf{v}_i$ which is used to generate the dual matrix $\mathtt{E}_i$:

$$\mathtt{E}'_i = vech^{-1}(\mathbf{v}_i), \tag{4.11}$$

where $vech^{-1}$ computes the vector and obtain a symmetric matrix. Then the matrix $\mathtt{Q}'_i$ can be computed by the inverse of $\mathtt{E}'_i$.

The approach presented so far is enough in the ideal scenario. However, many errors can lead to weak solutions in real scenarios, such as camera calibration errors, 2D object location errors caused by occlusions, for example. Thus, we have a minimization problem to find the near solution to the ideal of Equation 4.9:

$$\mathbf{w}'_i = \arg\min_{\mathbf{w}} \|\mathtt{M}_i\mathbf{w}_i\|^2. \tag{4.12}$$

Computing the Singular Value Decomposition (SVD) of the $\mathtt{M}_i$, the solution of Equation 4.12 is the singular vector related to the smallest singular value. Thus, $\mathtt{E}'_i = vech^{-1}(\mathbf{v}'_i)$ is the first value of the quadric.

4.1.1 Normalization

As mentioned, measurement errors embed the vector $\mathbf{w}'_i$. Even small errors can propagate and may lead to an inaccurate solution due to the variety of the magnitude of

the entries of $\mathtt{M}_i$. The solution proposed by Rubino (14) to attenuate the influence of those errors is to extract the translation matrix $\mathtt{H}_{fi}$, which applied to dual conic normalized and centralized in the origin $\mathtt{D}^c_{fi}$, gives the dual conic $\mathtt{D}'_{fi}$ as follows:

$$\mathtt{D}'_{fi} = \mathtt{H}_{fi}\,\mathtt{D}^c_{fi}\,\mathtt{H}^{\mathrm{T}}_{fi}, \tag{4.13}$$

where:

$$\mathtt{H}_{fi} = \begin{bmatrix} h & 0 & t^c_x \\ 0 & h & t^c_y \\ 0 & 0 & 1 \end{bmatrix} \text{ and } \mathtt{D}^c_{fi} = \begin{bmatrix} d^c_{11} & d^c_{12} & 0 \\ d^c_{21} & d^c_{22} & 0 \\ 0 & 0 & -1 \end{bmatrix}, \tag{4.14}$$

where t^c_x and t^c_y are the coordinates of the ellipse center, $h = \sqrt{\xi_1^2 + \xi_2^2}$, and ξ_1 and ξ_2 are the semi-axes. The values of $\mathtt{D}^c_{fi}$ may vary depending on the operation to obtain it, either inverse or adjoint.

After that, applying the inverse of the $\mathtt{H}_{fi}$ and $\mathtt{H}^{\mathrm{T}}_{fi}$ on both sides of Equation 4.2:

$$\beta_{fi}\,\breve{\mathtt{D}}_{fi} = \mathtt{H}^{-1}_{fi}\,\mathtt{P}_f\,\mathtt{E}'_i\,\mathtt{P}^T_{fi}\,\mathtt{H}^{-T}_{fi}. \tag{4.15}$$

The same transformation of Equation 4.13 can be performed in the dual quadric:

$$\mathtt{E}'_i = \mathtt{T}_i\,\mathtt{E}'^c_i\,\mathtt{T}^{\mathrm{T}}, \tag{4.16}$$

and the transformation matrix $\mathtt{T}_i \in \mathbb{R}^{4\times4}$ is applied in the Equation 4.15 as follows:

$$\beta_{fi}\,\breve{\mathtt{D}}_{fi} = \mathtt{H}^{-1}_{fi}\,\mathtt{P}_f\,\mathtt{T}_i\,\mathtt{T}^{-1}_i\,\mathtt{E}'_i\,\mathtt{T}^{-\mathrm{T}}_i\,\mathtt{T}^{\mathrm{T}}_i\,\mathtt{P}^T_{fi}\,\mathtt{H}^{-T}_{fi}, \tag{4.17}$$

which leads to:

$$\beta_{fi}\,\breve{\mathtt{D}}_{fi} = \mathtt{P}'_{fi}\,\mathtt{E}'^c_i\,\mathtt{P}'^T_{fi}, \tag{4.18}$$

where $\mathtt{P}'_{fi} = \mathtt{H}^{-1}_{fi}\,\mathtt{P}_f\,\mathtt{T}_i$. The value of matrix $\mathtt{T}_i$ is calculated from $\mathtt{E}'_i$ (Equation 4.11). Then, the Equation 4.18 is used to solve a new linear optimization problem:

$$\breve{\mathbf{w}}^c_i = \arg\min_{\breve{\mathbf{w}}} \|\mathtt{M}'^c_i\breve{\mathbf{w}}_i\|^2, \tag{4.19}$$

where $\breve{\mathbf{w}}^c_i$ is the vectorized form of the dual quadric $\breve{\mathtt{E}}^c_i$. The conversion of $\breve{w}^c_i$ to $\breve{\mathtt{E}}^c_i$ is achieved using first 10 elements of $\breve{w}^c_i$ and Equation 4.11. After, the following equation is applied:

$$\breve{\mathtt{E}}_i = \mathtt{T}_i\,\breve{\mathtt{E}}^c_i\,\mathtt{T}^{\mathrm{T}}_i. \tag{4.20}$$

4.1.2 Nonlinear optimization

Errors on bounding box estimation can lead to unsatisfactory solutions, such as degenerate ellipsoids and different quadrics. A non-linear cost function was designed to force the dual quadric matrix $\mathtt{E}_i$ to be a correct estimation of an ellipsoid and satisfy the conditions on Section 3.4.2.1.

As described in Section 3.4.4, the function $\mathtt{E}(\boldsymbol{\theta}, \mathbf{t}, \boldsymbol{\xi})$ can represent the dual quadric matrix $\mathtt{E}$. Using the *vech* operation (Equation 4.3) on $\mathtt{E}(\boldsymbol{\theta}, \mathbf{t}, \boldsymbol{\xi})$, we obtain the vector $\mathbf{v}(\boldsymbol{\theta}, \mathbf{t}, \boldsymbol{\xi})$:

$$\mathbf{v}(\boldsymbol{\theta}, \mathbf{t}, \boldsymbol{\xi}) = \begin{bmatrix} r_{11}^2(\boldsymbol{\theta})\xi_1^2 + r_{12}^2(\boldsymbol{\theta})\xi_2^2 + r_{13}^2(\boldsymbol{\theta})\xi_3^2 - t_1^2 \\ r_{11}(\boldsymbol{\theta})r_{21}(\boldsymbol{\theta})\xi_1^2 + r_{12}(\boldsymbol{\theta})r_{22}(\boldsymbol{\theta})\xi_2^2 + r_{13}(\boldsymbol{\theta})r_{23}(\boldsymbol{\theta})\xi_3^2 - t_1t_2 \\ r_{21}^2(\boldsymbol{\theta})\xi_1^2 + r_{22}^2(\boldsymbol{\theta})\xi_2^2 + r_{23}^2(\boldsymbol{\theta})\xi_3^2 - t_2^2 \\ r_{11}(\boldsymbol{\theta})r_{31}(\boldsymbol{\theta})\xi_1^2 + r_{12}(\boldsymbol{\theta})r_{32}(\boldsymbol{\theta})\xi_2^2 + r_{13}(\boldsymbol{\theta})r_{33}(\boldsymbol{\theta})\xi_3^2 - t_1t_3 \\ r_{21}(\boldsymbol{\theta})r_{31}(\boldsymbol{\theta})\xi_1^2 + r_{22}(\boldsymbol{\theta})r_{32}(\boldsymbol{\theta})\xi_2^2 + r_{23}(\boldsymbol{\theta})r_{33}(\boldsymbol{\theta})\xi_3^2 - t_2t_3 \\ r_{31}^2(\boldsymbol{\theta})\xi_1^2 + r_{32}^2(\boldsymbol{\theta})\xi_2^2 + r_{33}^2(\boldsymbol{\theta})\xi_3^2 - t_3^2 \\ -t_1 \\ -t_2 \\ -t_3 \\ -1 \end{bmatrix}, \tag{4.21}$$

where $\boldsymbol{\theta} = [\theta_1\ \theta_2\ \theta_3]^{\mathrm{T}}$ is the vector of angles, $\mathbf{t} = [t_1\ t_2\ t_3]^{\mathrm{T}}$ is the vector of translated terms, $\boldsymbol{\xi} = [\xi_1\ \xi_2\ \xi_3]^{\mathrm{T}}$ is the vector of semi-axes, and $r_{mn} \mid m, n = \{1, 2, 3\}$ are the elements of the rotation matrix $\mathtt{R}(\boldsymbol{\theta})$. Consequently, the vector $\mathbf{v}$ is a element of the vector $\mathbf{w}$ (Equation 4.9) and the inputs of vector $\mathbf{w}$ can be expressed by a vector $\mathbf{e} \in \mathbb{R}^{9+F}$ as follows:

$$\mathbf{e} = [\theta_1\ \theta_2\ \theta_3\ t_1\ t_2\ t_3\ \xi_1\ \xi_2\ \xi_3\ \beta_1\ \cdots\ \beta_F]^{\mathrm{T}}, \tag{4.22}$$

and the functional form of the vector $\mathbf{w}(\mathbf{e})$ is:

$$\mathbf{w}(\mathbf{e}) = \begin{bmatrix} r_{11}^2(\boldsymbol{\theta})\xi_1^2 + r_{12}^2(\boldsymbol{\theta})\xi_2^2 + r_{13}^2(\boldsymbol{\theta})\xi_3^2 - t_1^2 \\ r_{11}(\boldsymbol{\theta})r_{21}(\boldsymbol{\theta})\xi_1^2 + r_{12}(\boldsymbol{\theta})r_{22}(\boldsymbol{\theta})\xi_2^2 + r_{13}(\boldsymbol{\theta})r_{23}(\boldsymbol{\theta})\xi_3^2 - t_1t_2 \\ r_{21}^2(\boldsymbol{\theta})\xi_1^2 + r_{22}^2(\boldsymbol{\theta})\xi_2^2 + r_{23}^2(\boldsymbol{\theta})\xi_3^2 - t_2^2 \\ r_{11}(\boldsymbol{\theta})r_{31}(\boldsymbol{\theta})\xi_1^2 + r_{12}(\boldsymbol{\theta})r_{32}(\boldsymbol{\theta})\xi_2^2 + r_{13}(\boldsymbol{\theta})r_{33}(\boldsymbol{\theta})\xi_3^2 - t_1t_3 \\ r_{21}(\boldsymbol{\theta})r_{31}(\boldsymbol{\theta})\xi_1^2 + r_{22}(\boldsymbol{\theta})r_{32}(\boldsymbol{\theta})\xi_2^2 + r_{23}(\boldsymbol{\theta})r_{33}(\boldsymbol{\theta})\xi_3^2 - t_2t_3 \\ r_{31}^2(\boldsymbol{\theta})\xi_1^2 + r_{32}^2(\boldsymbol{\theta})\xi_2^2 + r_{33}^2(\boldsymbol{\theta})\xi_3^2 - t_3^2 \\ -t_1 \\ -t_2 \\ -t_3 \\ -1 \\ \beta_1 \\ \vdots \\ \beta_F \end{bmatrix}. \tag{4.23}$$

Thus, a new optimization problem can be formulated and the solution forced to be a ellipsoid:

$$\tilde{\mathbf{e}}_i = \arg\min_{\mathbf{e}_i} \left\| \mathtt{M}_i'^{c} \mathbf{w}(\mathbf{e}_i) \right\|^2. \tag{4.24}$$

The vector $\mathbf{e}$ needs to be initialized to achieve a good solution. The initial values for $\mathbf{e}$ are from the result of Equation 4.20. Using the Equations 3.12, 3.13 and 3.15 in Section 3.4.2.1, we recover the values of the rotation vector $\boldsymbol{\theta}_i^0$, translation vector $\mathbf{t}_i^0$ and semi-axes vector $\boldsymbol{\xi}_i^0$. The scale factors $\breve{\boldsymbol{\beta}}_i$ are from $\breve{\mathbb{w}}_i^c$ (Equation 4.19). Thus, the initial $\mathbf{e}_i^0$ is:

$$\mathbf{e}^0 = [\boldsymbol{\theta}_i^0 \ \mathbf{t}_i^0 \ \boldsymbol{\xi}_i^0 \ \breve{\boldsymbol{\beta}}_i]. \tag{4.25}$$

The final dual quadric $\mathtt{E}$ is calculated from the result of the non-linear optimization $\tilde{\mathbf{e}}_i$ using Equation 3.30.

The non-linear optimization can be constrained by the prior geometric knowledge of the elements in the scene. We can add these constraints in the scale ξ and position $\mathbf{t}$. The lower and upper bounds on the semi-axis ξ are the smaller and larger semi-axis size of the conic that originate the quadric. The bound of position is a threshold distance of the initial position $\mathbf{t}_i^0$.

4.2 OBJECT ASSOCIATION

The method proposed in this work supports any object detection method whose outputs are the classes and the bounding boxes that fit the perspective of the objects for each frame. Moreover, the method described in Section 4.1 uses ellipses already grouped to estimates an ellipsoid. However, so far, we do not have a method to associate the bounding boxes relative to the same real object to produces an ellipsoid. Thus, we developed a method to group the ellipses using geometric features representing an object.

Consider a static scene as a sequence $f = 1, \ldots, F$ of multiple frames from different perspectives with a set of detected objects $\mathcal{O} = \{o_j \mid j \in \mathbb{N} \text{ and } o_j = \{c_j, \mathtt{Q}_j\}\}$ in the scene, where c_j and $\mathtt{Q}_j$ are the class and the quadric of o_j, respectively. Also, consider each frame has a set of detected regions $i = 1, \ldots, N$, and each region is associated with an object $o_{fi} = \{c_{fi}, k_j, \mathtt{C}_{fi}\}$, where c_{fi} and $\mathtt{C}_{fi}$ are respectively the class and the conic that fits the bounding box of object o_{fi}. The object o_j can be defined as a set of objects o_{fi} as follows:

$$o_j = \{o_{fi} \mid c_j = c_{fi} \text{ and } f(o_{fi}) > \delta\}, \tag{4.26}$$

where $f(o_{fi})$ is a function which evaluate if a observable object o_{fi} in an image i is a view of object o_j in the map through a set of rules based on our knowledge of the problem.

We first considered creating a set of rules based on the common projected map points inside the detected conics of all objects o_{fi} with the same class, similar to (8). However, δ would be dependent on the scene due to factors that interfere in the number of features detected by ORB, such as textureless environment and poor illumination conditions, among others.

The second set of rules was formulated based on geometric relationships among the conics of o_j and their related keyframes. This strategy takes into account that an object

in the scene is detected in sequential frames, and the respective conics are close together in 3D world coordinates. These rules are described as follows:

Step 1: With the detected object o'_{fi}, containing only one conic $\mathtt{C}'_{fi}$, the subset of detected objects $\mathcal{O}''_j \subseteq \mathcal{O}_j$ and $\mathcal{O}''_j = \{o''_j \mid c''_j = c'_{fi}\}$ is obtained.

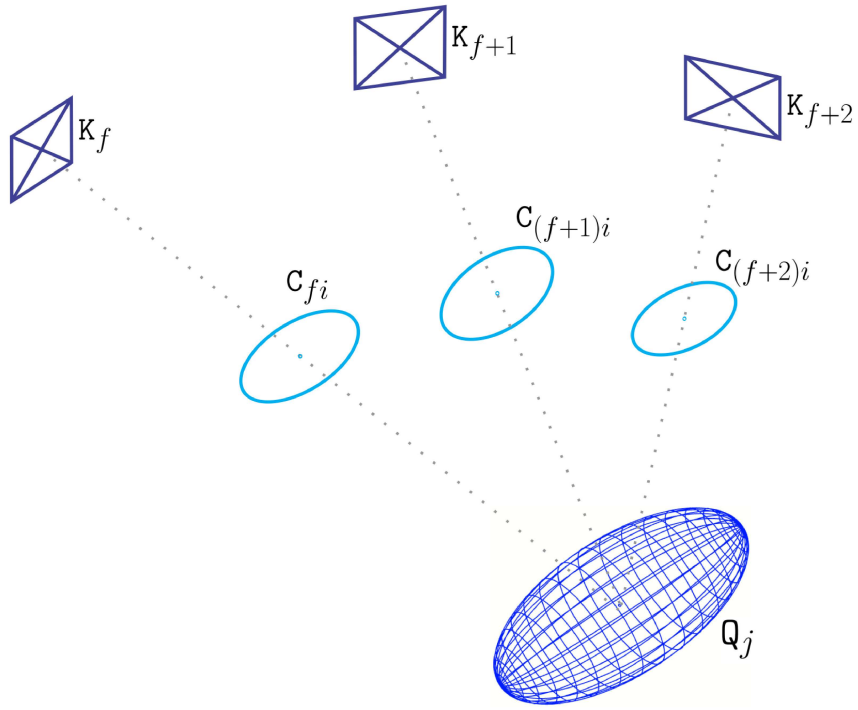
Step 2: The midpoint $\mathbf{x}''_j$ of all conics center of each object in $\mathcal{O}''_j$ is calculated.

Step 3: If the distance between $\mathbf{x}'_{fi}$ and the nearest $\mathbf{x}''_j$ is less than a threshold, the objects o'_{fi} and o''_j are merged into an object $\check{o}_j$; otherwise, the object o'_{fi} is added to $\mathcal{O}_j$.

Step 4: The center of the object $\check{o}_j$ is calculated as the nearest point of lines passing through each center of the conic $\check{\mathtt{C}}_j$ and the center of the associated keyframe (Figure 15). This center is used as initial position in nonlinear optimization (Equation 4.25).

Step 5: The quadric $\check{\mathtt{Q}}_j$ is calculated as described in Section 4.1.

Figure 15: Example of geometric elements in a scene. $\mathtt{K}_f$, $\mathtt{K}_{f+1}$ and $\mathtt{K}_{f+2}$ are sequential keyframes, $\mathtt{C}_{fi}$, $\mathtt{C}_{(f+1)i}$ and $\mathtt{C}_{(f+2)i}$ are the conics related with each keyframe from a generic region i in each keyframe, and $\mathtt{Q}_j$ is the quadric related to object o_j.



Source: Adapted from (14).

4.3 FULL BUNDLE ADJUSTMENT

As explained in Section 3.3, Bundle adjustment (BA) is a batch optimization method for collectively estimating camera extrinsic parameters and the structure of the 3D scene while minimizing a cost function.

In the monocular ORB-SLAM2, BA is performed in three points of the system. The first point, called motion-only, optimizes only the camera pose for each frame and it is executed in tracking thread. The second, called *local BA*, iteratively optimizes the poses of currently processed keyframe and map points connected within and it is executed in the local mapping thread. The third, called global BA, iteratively optimizes the poses of all keyframes and map points after closing the trajectory and it is executed after a loop closure detection in the loop closing thread.

The addition of new geometric entities can benefit BA (26). Thus, we choose to use the ellipses due to the high number of them compared with ellipsoids to add to BA performed by ORB-SLAM2. The object optimization thread creates the ellipses after keyframe optimization, i.e. after local BA. Therefore, the choice was to add the ellipses to the global BA and execute it after processing of a new keyframe by optimizing the position of all ellipses and improving the ellipsoids position.

The cost function of full BA performed by our method is defined as follows:

$$\{\mathbf{X}, \mathbf{R}, \mathbf{t}\} = \underset{\mathbf{X}_i, \mathbf{R}_k, \mathbf{t}_k}{\operatorname{argmin}} \sum_{i \in \mathcal{X}} \sum_{k \in \mathcal{K}} \rho \left(\| \mathbf{x}_i - \pi \left(\mathbf{R}_k \mathbf{X}_i + \mathbf{t}_k \right) \|^2_{\Sigma} \right), \tag{4.27}$$

where $\mathbf{X} \in \mathbb{R}^3$ is the position of points in world coordinates, $\mathbf{R}$ and $\mathbf{t}$ are the rotation matrices and position vector of keyframes, $\mathbf{x} \in \mathbb{R}^2$ is the positions of the corresponding points to $\mathbf{X}$ in image coordinates, $\mathcal{X}$ is the set of map points and ellipses center, $\mathcal{K}$ is the set of keyframes, ρ is the robust Huber loss function:

$$\rho(x) = \begin{cases} x^2 & \text{if } |x| \leq \delta^2 \\ 2\delta\sqrt{|x|} - \delta^2 & \text{otherwise} \end{cases}, \tag{4.28}$$

Σ is the covariance matrix associated with the scale of the point and π is the monocular projection function:

$$\pi \left(\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \right) = \begin{bmatrix} f_x \frac{x}{z} + c_x \\ f_y \frac{y}{z} + c_y \end{bmatrix}, \tag{4.29}$$

where (f_x, f_y) is the focal distance and (c_x, c_y) is the principal point.

The full BA is executed in our system after all ellipses of a keyframe were processed, and respective ellipsoids calculated.

5 EXPERIMENTAL RESULTS

In this chapter, we present the results obtained with our system. We compare the results of the camera trajectory estimation of our system over the original ORB-SLAM2. We did not find in the literature a dataset for object pose estimation with SLAM support. Thus, we evaluate the object pose estimation qualitatively. Finally, we present an application of our system in the construction of a virtual environment populated with the objects of a static indoor scene.

5.1 IMPLEMENTATION DETAILS

The implementation of our system, called VEM-SLAM, was in C++ programming language, with the ORB-SLAM2 implementation available on GitHub[1], the open source neural network framework Darknet, also available on GitHub[2], and the DCNN Yolov2 (12) as object detector with the weights distributed on the official webpage[3], which was trained on the MS COCO dataset (13).

5.2 YOLOV2 AND ORB-SLAM2 INTEGRATION

The object detection could be made in images on frames or keyframes. We evaluate the advantages and disadvantages of each option, and we chose to process the images in keyframes for the following reasons. The computational cost of processing images on frames could compromise the real-time processing due to the number of posterior calculations to estimate the ellipsoids. Moreover, the matrix of the camera associated with the frame is not optimized by local bundle adjustment, only the keyframes matrices. Thus, the matrix $\mathtt{P}_f$ in Equation 4.1 is already optimized on ellipsoid estimation. Another reason is the keyframes has minimum difference of perspective to each other, which minimizes the redundancy of ellipses with a near pose and decrease the errors associated with the camera matrix (Equation 4.2).

Yolov2 bounding boxes output with edges closer than a threshold to the image boundaries was discarded to minimize the errors of occlusions related to limits of the image. This threshold was empirically defined as 10 pixels.

5.3 NONLINEAR OPTIMIZATION THROUGH GENETIC ALGORITHM

A nonlinear optimizer is needed in the last ellipsoid optimization (Section 4.1.2). This nonlinear optimizer is processed in every addition of ellipse in an object and with more

[1] `https://github.com/raulmur/ORB_SLAM2`
[2] `https://github.com/pjreddie/darknet`
[3] `https://pjreddie.com/darknet/yolov2/`

than three ellipses. Among the many nonlinear methods available on literature (32, 33), we chose the genetic algorithm due to be suitable for our problem, easy to implement, and flexible to constraints. We implemented the classic genetic algorithm with GPU processing using the library Hemi[4].

One of the disadvantages of the genetic algorithm is the number of parameters to configure. In our implementation, there were only three parameters for determining their optimum values, population size, crossover and mutation rate. Finding the optimum values with a greedy strategy has a high computational cost. Thus, we use the Irace (34) to determine the optimal values for all the eight parameters. The Irace package implements the iterated racing procedure, which is an extension of the Iterated F-race procedure, for the statistical environment R. The inputs of Irace were instances of our problem, the list of parameters to be optimized and their search ranges, and also the programming to process the instances who generates a value to be minimized. The instances of the problem were made using the sequences in TUM RGB-D Dataset (35). The programming was the nonlinear genetic optimizer implemented. The output was the optimal value of each parameter, and other equivalent values for them, (Figure 16). Table 9 presents the values of inputs and outputs of Irace optimization.

Figure 16: The output of Irace execution on optimization of nonlinear genetic algorithm parameters.

```
# Elite configurations (first number is the configuration ID; listed from best to
  worst according to the sum of ranks):
    maxgeneration population mutation
126           200        400      0.1
115           350        440      0.1
73            250        410      0.2
119           260        460      0.1
```

Source: Elaborated by the author (2020).

Table 9: Search range and optimum values of each parameters on Irace optimization

Parameter	Irace search range	Optimum Value
Population	50 - 500	400
Max generation	10 - 1000	200
Mutation ratio	0.10 - 0.90	0.10

Source: Elaborated by the author (2020).

The optimized values obtained in Irace execution were influenced by the number of ellipses used to estimate the ellipsoids. In the instances inputs used in the test presented above, the maximum number of ellipses used to ellipsoid estimation was nine. Thus, the optimized values obtained could not e suitable for large scenes.

[4] `http://harrism.github.io/hemi/`

5.4 THREADS TIMING EVALUATION

In order to evaluate the computational cost of each step in the threads of the proposed system and compare it to the original ORB-SLAM2, we executed each system ten times using the freiburg3_long_office_household TUM RGB-D dataset sequence. Both systems processed 2504 frames in each execution, the frame rate was set in 30 FPS, and the ORB features extraction was remaining default in 1000 features.

Table 10 shows the mean and standard deviation execution time and the frequency (per frame) of execution of each step for both systems. The mean FPS achieved was 43.02 $\pm$ 1.17 for ORB-SLAM2 and 32.10 $\pm$ 2.21 for VEM-SLAM. Comparing the results, the inclusion of two threads in VEM-SLAM and the execution of Full BA at the end of every new keyframe processing, Local Mapping thread, raises the time execution comparing with the original ORB-SLAM2 threads. However, this raising did not impact enough to lose the real-time processing. The concurrent execution of 2D object detection and 3D object optimization steps in GPU impacts time execution in respective threads. In VEM-SLAM, the mean processing frequency of keyframe by Local Mapping thread was 4.10 FPS while the processing in Object Detection and Object Optimization threads was, respectively, 1.67 and 5.50 FPS. Therefore, there is an accumulation of keyframes in the processing queue of the last two cited threads, and this accumulation is the bottleneck of our processing. Another point is the large standard deviation of times, and this was due to the increasing of elements number to process in each thread. This observation leads us to believe that with longer sequences in number of frames, the real-time processing capacity will be lost.

5.5 CAMERA LOCALIZATION

In order to evaluate the impact on camera localization accuracy due to our modification in the global bundle adjustment function cost (Section 4.3), we executed our system and two versions of ORB-SLAM2 in same conditions. The systems was executed on a machine with Intel® Core™ i7-6700 CPU 3.40GHz, Nvidia Titan XP GPU, and 16 GB of RAM. We used eight sequences of the TUM RGB-D dataset (35) for SLAM systems evaluation, and the Root-Mean-Square Error (RMSE) of Absolute Trajectory Error (ATE) calculated using the tool available on-line[5] as evaluation criteria.

ATE is the main evaluation criteria for SLAM solutions to evaluate the camera trajectory estimation, and it directly measures the difference of the camera positions between ground truth and estimated. The camera positions are associated using the timestamps of each data and aligned using the method of Horn (36). However, the ATE is an absolute measure, and not consider the length of the trajectories. The ATE calculation

[5] `https://github.com/raulmur/evaluate_ate_scale`

Table 10: ORB-SLAM2 and VEM-SLAM timing results of each stage in proposed pipeline in milliseconds (mean ± std deviations) and frequency of execution (per frame). Bold figures denote the lower mean time processing for each stage.

Thread	Step	ORB-SLAM2		VEM-SLAM	
		Frequency	Time	Frequency	Time
Tracking	Extract ORB Features	1.00	12.79 ± 1.42	1.00	15.46 ± 7.76
	Map Initialization	0.01	20.16 ± 4.58	0.01	28.78 ± 15.10
	Initial Pose Estimation	0.99	2.33 ± 0.71	0.99	2.99 ± 3.15
	Track Local Map	0.99	8.20 ± 3.32	0.99	10.07 ± 6.80
	New Keyframe Decision	0.99	0.07 ± 0.13	0.99	0.11 ± 0.67
	Total	1.00	**23.5 ± 4.42**	1.00	28.81 ± 12.28
L. Mapping	Keyframe Insertion	0.12	9.41 ± 3.04	0.11	12.60 ± 6.87
	Recent MapPoints Culling	0.12	0.07 ± 0.02	0.11	0.12 ± 0.79
	New Points Creation	0.12	12.55 ± 2.87	0.11	19.00 ± 9.00
	Local BA	0.12	149.06 ± 102.98	0.11	18.78 ± 9.07
	Local Keyframes Culling	0.12	7.96 ± 6.18	0.11	10.13 ± 9.16
	Total	0.12	**177.7 ± 113.15**	0.11	243.83 ± 155.72
L. Closing	Query Database	0.12	3.05 ± 2.03	0.11	3.45 ± 3.24
	Compute SE3	0.06	0.45 ± 0.99	0.06	0.90 ± 4.00
	Loop Fusion	0.0004	362.02 ± 40.01	0.0004	508.63 ± 180.30
	Optimize Essential Graph	0.0004	195.65 ± 22.92	0.0004	302.99 ± 63.76
	Total	0.12	**5.16 ± 33.42**	0.11	7.48 ± 54.75
Full BA	Full BA	0.0004	1411.26 ± 268.41	0.11	378.64 ± 347.94
	Map Update	0.0004	118.62 ± 38.99	0.11	82.40 ± 58.79
	Total	0.0004	1509.84 ± 266.87	0.11	**460.89 ± 395.81**
Ob. Detection	2D Object Detection			0.11	598.02 ± 767.00
	Total			0.11	598.02 ± 767.00
Ob. Optimizer	Object 3D Pose Estimation			0.05	0.16 ± 0.60
	Object3D Pose Optimization			0.03	488.43 ± 121.22
	Map Object Update			0.08	0.04 ± 0.04
	Total			0.08	181.78 ± 247.61

Source: Elaborated by the author (2020).

ignores the camera positions without association and, therefore, not always representing an evaluation for the whole trajectory. Figure 17 shows differences in the estimated trajectory between two executions of ORB-SLAM2 for the same TUM frames sequence, showing differences at the beginning of trajectory estimation (orange rectangles) and a loss of localization (green rectangle). A simple comparison of ATE results could lead to wrong interpretations in this case. Thus, for a better evaluation, we calculate a distance ratio being the percentage of the total distance covered by the estimated trajectory over the total distance of the ground truth trajectory for each sequence. In all tests, the value of the frequency of frames processing was fixed in 30 FPS setting a input parameter of ORB-SLAM2. This was necessary because this value of FPS was the same condition that the sequences in TUM RGB-D dataset were recorded. But, the values of FPS in the experiments varied between 41.2 and 48.6 FPS achieving real-time processing.

To isolate the influence of ellipses addition in full BA execution in VEM-SLAM, we implemented a variation of ORB-SLAM2, called ORB-SLAM2BA, which executed the original full BA at the same point that VEM-SLAM performs. We executed the

ORB-SLAM2, ORB-SLAM2BA and VEM-SLAM two hundred times for each sequence in the dataset and we took the one hundred results with higher distance ration to the final calculation of ATE. This selection was necessary due to the non-deterministic character of the monocular vSLAM solutions involved, varying the map initialization point and the distance of estimated trajectory.

Table 11 shows the average of RMSE ATE and the distance ratio for each sequence executed using the original ORB-SLAM2, the ORB-SLAM2BA and our system. The differences in distance ratio results among the systems are due to some aspects. The map initialization step depends on a significant parallax between two frames, and its processing is executed independently of the frame capture, varying the beginning of camera trajectory estimation between executions. Furthermore, some sequences of the dataset have an abrupt change of camera position, which creates motion blur and leads to the SLAM system to lose the localization. Comparing the RMSE ATE results in Table 11, the number of frames in the sequence influenced the improvement with the addition of ellipses in BA. For sequences with few number of frames, equal or lower than 2359, there was no evidence to support the improvement of camera trajectory estimation with the addition of new elements in the BA. For sequences with more frames, equal or higher than 2504, the difference between the results of ORB-SLAM2BA and VEM-SLAM showed that there was an improvement of camera trajectory estimation with the addition on new elements in BA. With more frames, the number of keyframes generated increase, and this leads to an increasing number of elements in BA calculation, improving the optimization.

Table 11: Average localization errors and distance ratios on TUM sequences. The improvement of camera trajectory estimation was significant for sequences with a high number of frames, below the dotted line. Bold figures denote the lower value of RMSE ATE of each sequence.

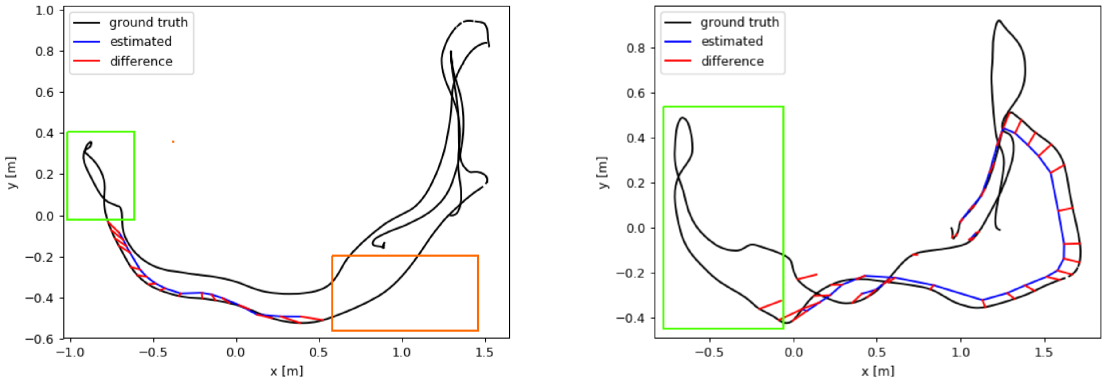
Sequence	Number of frames processed	ORB-SLAM2		ORB-SLAM2 BA		VEM-SLAM	
		RMSE ATE (m)	Distance ratio (%)	RMSE ATE (m)	Distance ratio (%)	RMSE ATE (m)	Distance ratio (%)
fr1_360	755	0.0914 ± 0.0213	14.3	0.0926 ± 0.0231	13.5	**0.0729 ± 0.0252**	13.7
fr1_desk	595	0.0176 ± 0.0026	47.4	0.0159 ± 0.0017	46.7	**0.0144 ± 0.0017**	68.9
fr1_desk2	639	0.0236 ± 0.0018	22.0	**0.0208 ± 0.0015**	34.0	0.0232 ± 0.0035	53.9
fr1_room	1360	0.0276 ± 0.0058	40.9	**0.0207 ± 0.0018**	38.9	0.0270 ± 0.0055	36.3
fr1_xyz	797	**0.0072 ± 0.0005**	52.4	0.0074 ± 0.0005	52.8	0.0076 ± 0.0004	54.2
fr2_desk	2359	0.0114 ± 0.0006	40.5	0.0091 ± 0.0006	38.7	**0.0081 ± 0.0006**	43.3
fr2_xyz	3665	0.0054 ± 0.0002	60.1	0.0055 ± 0.0002	60.6	**0.0029 ± 0.0003**	57.2
fr3_long	2504	0.0225 ± 0.0022	98.2	0.0199 ± 0.0013	97.7	**0.0156 ± 0.0009**	98.2

Source: Elaborated by the author (2020).

5.6 OBJECT 3D POSE ESTIMATION

We did not find any dataset in the literature for the evaluation of 3D pose of indoor objects compatible with our object-SLAM system. The majority of works cited in Chapter 2 produced their datasets and only for qualitative evaluation. Nicholson et al.

Figure 17: Example of errors (red) on estimated trajectory (blue) and ground truth (black) in same TUM sequence. There are differences in camera trajectory initialization and loss of localization. The green rectangles emphasizes the initialization differences between the executions and the orange rectangle emphasis the localization loss.



Source: Elaborated by the author (2020).

(37) used UnrealCV plugin (38) to build a virtual world with the ground truth of objects 3D pose on it. We tried to use this plugin, and the DCNN Yolov2 detected objects in the tested scene, but the ORB-SLAM2 did not recognize enough features to initialize a map, even changing the predefined parameters relative to the initialization map. Thus, we were not able to perform a quantitative evaluation of 3D object pose estimation.

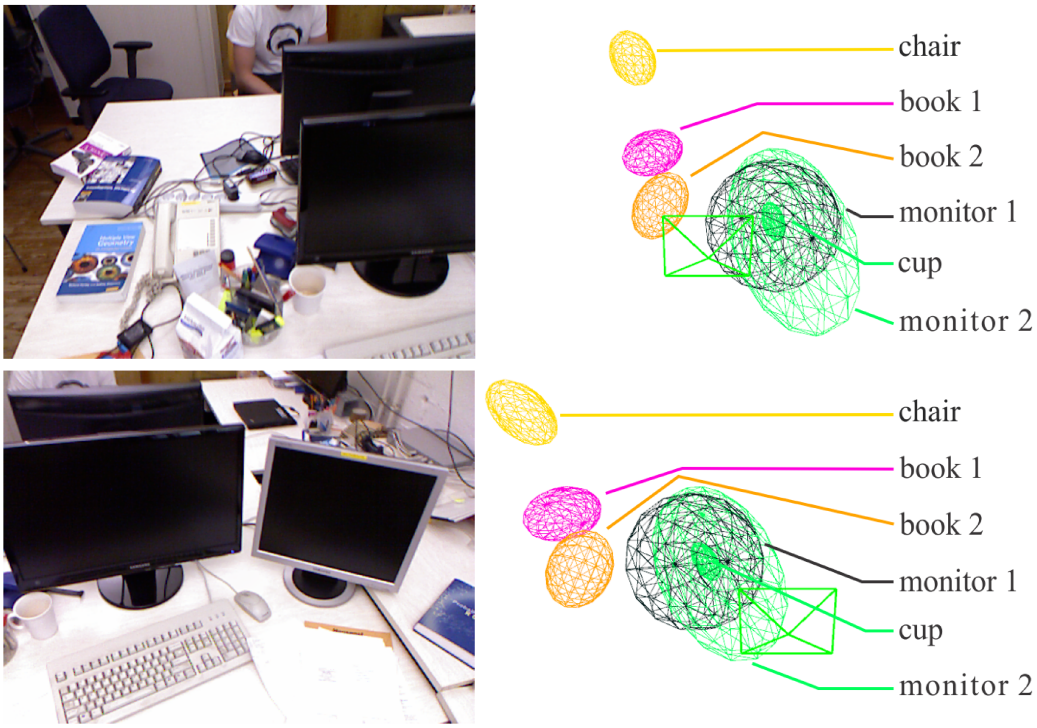
We demonstrate a qualitative evaluation for TUM RGB-D dataset sequences (Figure 18). The 3D object map has been put in the same perspective has the keyframe image (green rectangle). These images show how accurately the ellipsoids represent the geometry of some objects in the scene. The object pose is directly dependent on the quality of map initialization, since the the camera pose estimation is used to reproject the quadric from the conic (Equations 4.2). The position estimation of the objects has a more stable behavior than rotation and scale. We believe that this feature is due to the constraining in nonlinear optimization.

5.7 PROVE OF CONCEPT: GENERATING A VIRTUAL ENVIRONMENT

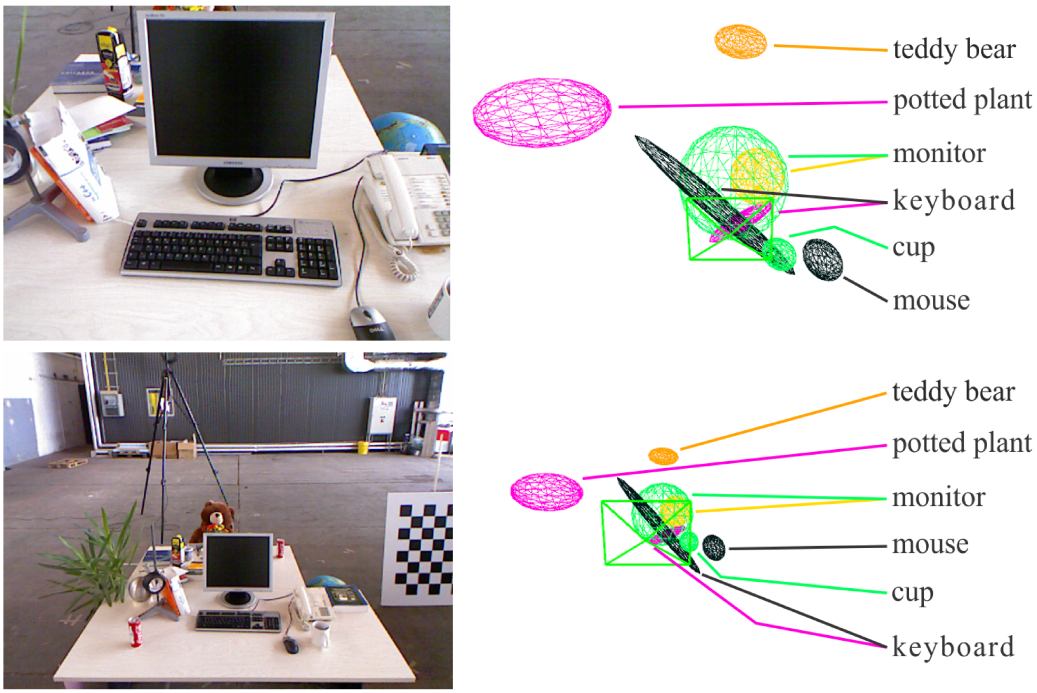
With the purpose to demonstrate the applicability of our system, we developed a software to reconstruct a virtual environment of a scene filmed by a monocular camera. The software was built using the open-source javascript framework for 3D computer graphics Three.js[6] in a web browser. Figure 19 shows images of a scene virtually reconstructed by the application. We used the VEM-SLAM estimated 3D objects position of a scene as input to create the virtual environment. The software has an object model library, so the detected objects must have their respective virtual models.

[6] `https://threejs.org/`

Figure 18: Objects pose and shape estimation of objects in an sequences of TUM dataset. The pose of the keyframes are represented by the green rectangle in the objects map.



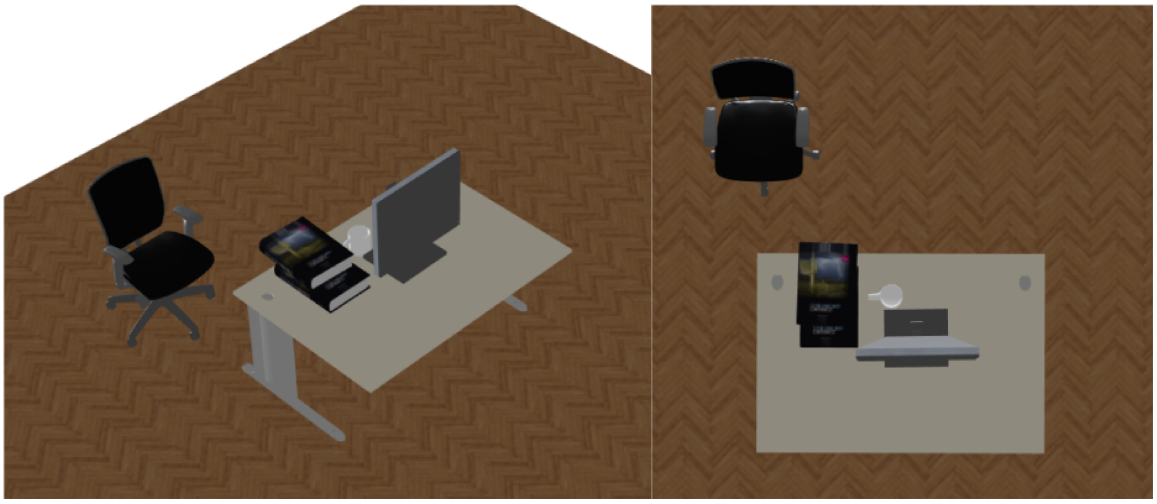
(a) Sequence fb1_xyz. In this particular processing, the cup position was wrong estimated. In the second image, the third monitor to the right, did not have enough detections to be represented.



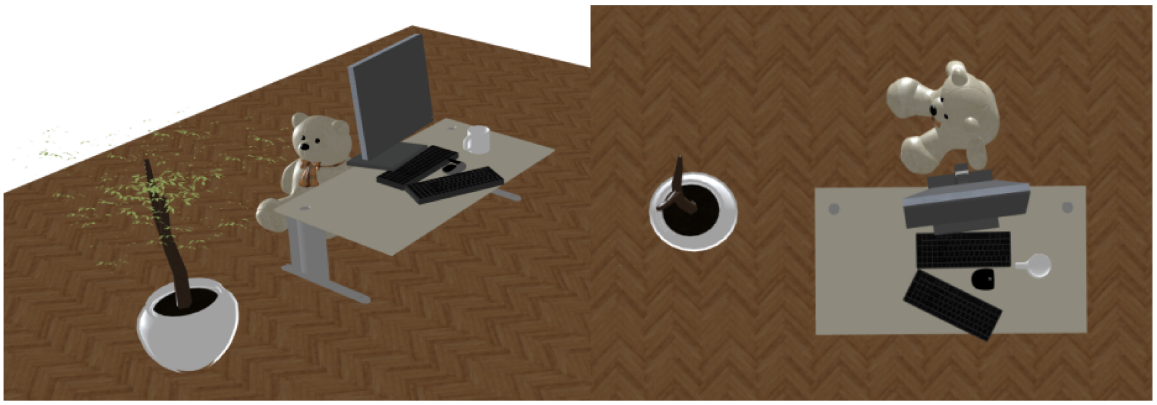
(b) Sequence fb2_desk. In this particular processing, the keyboard and monitor were wrong estimated by two ellipsoids.

Source: Elaborated by the author (2020).

Figure 19: Application developed to generating a virtual environment from TUM dataset sequences. The desk was fixed in both environments.



(a) Virtual environment for sequence fb1_xyz



(b) Virtual environment for sequence fb2_desk.

Source: Elaborated by the author (2020).

Some real scenes have objects that 2D object detector does not be able to find. Thus it needs to add manually in the virtual environment, per example, the table in Figure 19.

We used only the position of 2D objects detected because there was no correlation between the orientation and scale of the ellipsoids and the models in the library. The rotation and scale ellipsoid parameters are only to fit the ellipsoid onto the volume of objects. Thus, the size and rotation of objects in the virtual environment have to be defined in advance.

Another limitation is the hierarchical structure of the objects. As the virtual environment has collision and gravity, once the objects are inserted in the environment, their positions can vary. Objects can be on top of each other, as the size of the model may not reflect the size of the real object.

6 CONCLUSION AND FUTURE WORKS

This work presented a new approach to deal with 3D object pose estimation from images of an indoor scene. We propose a new integration of a 2D object detector and a keyframe-based monocular vSLAM solution. We hypothesized that is possible to build a virtual environment from a real scene filmed by a monocular camera. To verify our hypothesis, we develop an integration of state-of-the-art monocular vSLAM solution (9) with an object detector based on a deep convolutional neural network (12), using a 3D object pose recover from 2D images (15). The system developed uses conics fitted in the bounding boxes extracted by the object detector to reconstruct quadrics with the information of position, rotation, and scale of the objects. To verify the quality of pose estimation, we used the public TUM RGB-D dataset (35).

A prove of concept of our system was achieved with the construction of a virtual environment from a real scene with accurate positioning of objects. Besides, the addition of the conics in the full bundle adjustment (BA) benefited the camera localization and led to a better object pose estimation.

Our system has limitations inherent to the methods used. Regarding the 3D object pose estimation, since a full observation of the object in most of the scene is necessary for a good 3D reconstruction, our system fails on partially occluded objects in the scene. Also, scenes with few variations of perspective throughout the camera trajectory lead to a few keyframes on SLAM and, consequently, few objects data. Another aspect is the modeling the 2D perspective of objects in the image by an ellipse fitted in the bounding box aligned with the image axes. This modeling is not a good representation of all possible object perspectives in an image, and it impacts the quality of quadric pose estimation.

As future works, we propose the use of an instance semantic object detector instead of an object localization detector. The information about the contour of the object can be used to estimate the orientation of conics and benefits the quadric optimization. As far as we know, there is no public dataset for 3D object pose estimation evaluation. The UnrealCV plugin cited in Section 5 is an attempt to fill this void, but it is not working with all SLAM systems. Thus, relevant future work for the field is the construction of a dataset for 3D object pose estimation compatible with SLAM, contained camera and objects ground truth. Furthermore, other proposals for future work are our system generalization for stereo and RGB-D SLAM and the enhancement of our system to dynamic indoor and outdoor scenes.

REFERENCES

[1] CHATILA, R.; LAUMOND, J. Position referencing and consistent world modeling for mobile robots. **Proceedings. 1985 IEEE International Conference on Robotics and Automation**, v. 2, p. 138–145, 1985.

[2] SAPUTRA, M. R. U.; MARKHAM, A.; TRIGONI, N. Visual SLAM and Structure from Motion in Dynamic Environments: A Survey. **ACM COMPUTING SURVEYS**, v. 51, n. 2, jun. 2018.

[3] TAKETOMI, T.; UCHIYAMA, H.; IKEDA, S. Visual SLAM algorithms: a survey from 2010 to 2016. **IPSJ Transactions on Computer Vision and Applications**, v. 9, n. 1, p. 16, 2017.

[4] NEIVA, F. W.; SILVA, R. L. S. Revisão Sistemática em Ciência da Computaçao - Guia Prático. **Universidade Federal de Juiz de Fora**, 2016.

[5] KITCHENHAM, B. **Procedures for Performing Systematic Reviews**. Keele, UK, Keele Univ., v. 33, n. TR/SE-0401, p. 28, 2004.

[6] CHENG, G.; HAN, J.. A survey on object detection in optical remote sensing images. **ISPRS Journal of Photogrammetry and Remote Sensing**, v. 117, p. 11–28, 2016.

[7] PARKHIYA, P. et al. Constructing Category-Specific Models for Monocular Object-SLAM. In: **2018 IEEE International Conference on Robotics and Automation (ICRA)**. 2018.

[8] MCCORMAC, J. et al. Fusion++: Volumetric Object-Level SLAM. In: **2018 International Conference on 3D Vision (3DV)**. 2018.

[9] MUR-ARTAL, R.; TARDOS, J. D. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. **IEEE Transactions on Robotics**, v. 33, n. 5, p. 1255–1262, out. 2017.

[10] MUR-ARTAL, R.; MONTIEL, J. M. M.; TARDOS, J. D. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. **IEEE Transactions on Robotics**, v. 31, n. 5, p. 1147–1163, 7 out. 2015.

[11] HE, K. et al. Mask R-CNN. **Proceedings of the IEEE International Conference on Computer Vision**, v. 2017- Octob, p. 2980–2988, 2017.

[12] REDMON, J.; FARHADI, A. YOLO9000: Better, Faster, Stronger. In: **IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, 2017. p. 6517–6525.

[13] LIN, T. Y. et al. Microsoft COCO: Common objects in context. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 8693 LNCS, n. PART 5, p. 740–755, 2014.

[14] RUBINO, Cosimo. **Scene Understanding with Multi-view Geometry and Semantics.** Thesis (Doctoral) — Universit‘a di Genova, 2017.

[15] RUBINO, C.; CROCCO, M.; DEL BUE, A. 3D Object Localisation from Multi-view Image Detections. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 40, n. 6, p. 1–1, 2017.

[16] YANG, Shichao; SCHERER, Sebastian. CubeSLAM: Monocular 3-D Object SLAM. **IEEE Transactions on Robotics**, v. 35, n. 4, p. 925–938, 2019.

[17] LIU, L et al. Deep Learning for Generic Object Detection: A Survey. **International Journal of Computer Vision**, 2019.

[18] ZHANG, X. et al. Object Class Detection: A Survey. **ACM Computing Surveys**, v. 46, n. 1, p. 1–53, 1 out. 2013.

[19] JIAO, L.; ZHAO, J. A Survey on the New Generation of Deep Learning in Image Processing. **IEEE Access**, v. 7, p. 172231–172263, 2019.

[20] ZHAO, Z. et al. Object Detection With Deep Learning: A Review. **IEEE Transactions on Neural Networks and Learning Systems**, v. PP, p. 1–21, 2019.

[21] CHONG, T.J. et al. Sensor Technologies and Simultaneous Localization and Mapping (SLAM). **Procedia Computer Science**, v. 76, n. Iris, p. 174–179, 2015.

[22] LOWE, D.G. Object recognition from local scale-invariant features. In: **Proceedings of the Seventh IEEE International Conference on Computer Vision**, 1999. p. 1150–1157 vol.2.

[23] BAY, H.; TUYTELAARS, T.; VAN GOOL, L. SURF: Speeded Up Robust Features. In: **Computer Vision – ECCV 2006. ECCV 2006.**, Springer Berlin Heidelberg, 2006. p. 404–417.

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

[25] FUENTES-PACHECO, J.; RUIZ-ASCENCIO, J.; RENDÓN-MANCHA, J. M. Visual simultaneous localization and mapping: a survey. **Artificial Intelligence Review**, v. 43, n. 1, p. 55–81, 13 jan. 2015.

[26] TRIGGS, Bill et al. Bundle Adjustment - A Modern Synthesis. In: **Vision Algorithms: Theory and Practice**, 2000. v. 1883, p. 298–372. ISBN 3-540-67973-1.

[27] HARTLEY, R. I.; ZISSERMAN, A. **Multiple View Geometry in Computer Vision.** 2nd. ed. New York: Cambridge University Press, 2004.

[28] RICHTER-GEBERT, Jürgen. **Perspectives on Projective Geometry**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[29] KARL, W. C.; VERGHESE, G. C.; WILLSKY, A.S. Reconstructing Ellipsoids from Projections. **CVGIP: Graphical Models and Image Processing**, v. 56, n. 2, p. 124–139, mar. 1994.

[30] HENDERSON, H. V.; SEARLE, S R. Vec and vech operators for matrices, with some uses in jacobians and multivariate statistics. **Canadian Journal of Statistics**, v. 7, n. 1, p. 65–81, 1979.

[31] MA, S.; LI, L. Ellipsoid reconstruction from three perspective views. In: **Proceedings of 13th International Conference on Pattern Recognition**, 1996. p. 344–348 vol.1.

[32] EISELT, H. A.; SANDBLOM, C. **Nonlinear Optimization.** Cham: Springer International Publishing, 2019. v. 282.

[33] MICHALEWICZ, Z. Evolutionary computation techniques for nonlinear programming problems. **International Transactions in Operational Research**, v. 1, n. 2, p. 223–240, apr. 1994.

[34] LÓPEZ-IBÁÑEZ, M. et al. The irace package: Iterated racing for automatic algorithm configuration. **Operations Research Perspectives**, v. 3, p. 43–58, 2016.

[35] STURM, J. et al. A benchmark for the evaluation of RGB-D SLAM systems. In: **IEEE/RSJ International Conference on Intelligent Robots and Systems**, 2012. p. 573–580. ISBN 978-1-4673-1736-8. ISSN 21530858.

[36] HORN, B. K. P. Closed-form solution of absolute orientation using unit quaternions. **Journal of the Optical Society of America A**, v. 4, n. 4, p. 629, 1987.

[37] NICHOLSON, L.; MILFORD, M.; SUNDERHAUF, N. QuadricSLAM: Dual Quadrics From Object Detections as Landmarks in Object-Oriented SLAM. **IEEE Robotics and Automation Letters**, v. 4, n. 1, p. 1–8, jan. 2019.

[38] QIU, W. et al. UnrealCV: Virtual Worlds for Computer Vision. In: **Proceedings of the 25th ACM International Conference on Multimedia**, 2017, New York, USA: ACM Press, 2017. p. 1221–1224.

APPENDIX A - References of Systematic Literature Review

[A1] ATAER-CANSIZOGLU, E.; TAGUCHI, Y. Object detection and tracking in RGB-D SLAM via hierarchical feature grouping. In: IEEE; RSJ. **2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. [S.l.]: IEEE, 2016. p. 4164–4171. ISBN 978-1-5090-3762-9.

[A2] CHI, J.; WU, H.; TIAN, G. Object-Oriented 3D Semantic Mapping Based on Instance Segmentation. **Journal of Advanced Computational Intelligence and Intelligent Informatics**, v. 23, n. 4, p. 695–704, jul 2019. ISSN 1883-8014.

[A3] CIVERA, J. et al. Towards semantic SLAM using a monocular camera. In: **2011 IEEE/RSJ International Conference on Intelligent Robots and Systems**. [S.l.]: IEEE, 2011. p. 1277–1284. ISBN 978-1-61284-454-1. ISSN 2153-0858.

[A4] DHARMASIRI, T.; LUI, V.; DRUMMOND, T. MO-SLAM: Multi object SLAM with run-time object discovery through duplicates. In: **2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. [S.l.]: IEEE, 2016. v. 2016-Novem, p. 1214–1221. ISBN 978-1-5090-3762-9. ISSN 21530866.

[A5] FIORAIO, N.; CERRI, G.; Di Stefano, L. Towards Semantic KinectFusion. In: PETROSINO, A. (Ed.). **International Conference on Image Analysis and Processing**. [S.l.], 2013, (Lecture Notes in Computer Science, v. 8157). p. 299–308. ISBN 978-3-642-41184-7; 978-3-642-41183-0.

[A6] GÁLVEZ-LÓPEZ, D. et al. Real-time monocular object SLAM. **Robotics and Autonomous Systems**, Elsevier B.V., v. 75, p. 435–449, jan 2016. ISSN 09218890.

[A7] GAO, Q. H. et al. Object registration in semi-cluttered and partial-occluded scenes for augmented reality. **Multimedia Tools and Applications**, Multimedia Tools and Applications, nov 2018. ISSN 1380-7501.

[A8] HOSSEINZADEH, M. et al. Real-Time Monocular Object-Model Aware Sparse SLAM. In: **International Conference on Robotics and Automation (ICRA)**. [S.l.]: IEEE, 2019. v. 2019-May, p. 7123–7129. ISBN 978-1-5386-6027-0. ISSN 10504729.

[A9] HOSSEINZADEH, M. et al. Structure Aware SLAM Using Quadrics and Planes. In: JAWAHAR, C. V. et al. (Ed.). **Computer Vision – ACCV 2018**. Cham: Springer International Publishing, 2019. (Lecture Notes in Computer Science, i), p. 410–426. ISBN 978-3-030-20892-9. ISSN 16113349.

[A10] IQBAL, A.; GANS, N. R. Localization of Classified Objects in SLAM using Nonparametric Statistics and Clustering. In: **2018 IEEE/RSJ International Conference**

on Intelligent Robots and Systems (IROS). [S.l.]: IEEE, 2018. p. 161–168. ISBN 978-1-5386-8094-0. ISSN 21530866.

[A11] LI, C. et al. Incremental scene understanding on dense SLAM. In: IEEE; RSJ. **2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. [S.l.]: IEEE, 2016. p. 574–581. ISBN 978-1-5090-3762-9.

[A12] LI, J.; MEGER, D.; DUDEK, G. Semantic Mapping for View-Invariant Relocalization. In: **International Conference on Robotics and Automation (ICRA)**. [S.l.]: IEEE, 2019. v. 2019-May, p. 7108–7115. ISBN 978-1-5386-6027-0. ISSN 10504729.

[A13] LIU, K. et al. Object-aware Semantic Mapping of Indoor Scenes using Octomap. In: **Chinese Control Conference (CCC)**. [S.l.]: IEEE, 2019. v. 2019-July, p. 8671–8676. ISBN 978-9-8815-6397-2. ISSN 21612927.

[A14] LOESCH, A. et al. Generic edgelet-based tracking of 3D objects in real-time. In: **2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. [S.l.]: IEEE, 2015. v. 2015-Decem, p. 6059–6066. ISBN 978-1-4799-9994-1. ISSN 21530866.

[A15] LOESCH, A. et al. Localization of 3D objects using model-constrained SLAM. **Machine Vision and Applications**, Springer Berlin Heidelberg, v. 29, n. 7, p. 1041–1068, oct 2018. ISSN 0932-8092.

[A16] MA, L.; SIBLEY, G. Unsupervised Dense Object Discovery, Detection, Tracking and Reconstruction. In: Fleet, D and Pajdla, T and Schiele, B and Tuytelaars, T. (Ed.). **European Conference on Computer Vision**. [S.l.: s.n.], 2014. (Lecture Notes in Computer Science, v. 8690), p. 80–95. ISBN 978-3-319-10605-2; 978-3-319-10604-5. ISSN 0302-9743.

[A17] MCCORMAC, J. et al. Fusion++: Volumetric Object-Level SLAM. In: **2018 International Conference on 3D Vision (3DV)**. [S.l.]: IEEE, 2018. p. 32–41. ISBN 978-1-5386-8425-2.

[A18] NAKAJIMA, Y.; SAITO, H. Simultaneous Object Segmentation and Recognition by Merging CNN Outputs from Uniformly Distributed Multiple Viewpoints. **IEICE Transactions on Information and Systems**, E101.D, n. 5, p. 1308–1316, may 2018. ISSN 0916-8532.

[A19] NAKAJIMA, Y.; SAITO, H. Efficient Object-oriented Semantic Mapping with Object Detector. **IEEE Access**, IEEE, v. 7, p. 1–1, 2018. ISSN 2169-3536.

[A20] NICHOLSON, L.; MILFORD, M.; SUNDERHAUF, N. QuadricSLAM: Dual Quadrics From Object Detections as Landmarks in Object-Oriented SLAM. **IEEE Robotics and Automation Letters**, v. 4, n. 1, p. 1–8, jan 2019. ISSN 2377-3766.

[A21] PARKHIYA, P. et al. Constructing Category-Specific Models for Monocular Object-SLAM. In: **2018 IEEE International Conference on Robotics and Automation (ICRA)**. [S.l.]: IEEE, 2018. p. 1–9. ISBN 978-1-5386-3081-5.

[A22] PILLAI, S.; LEONARD, J. Monocular SLAM Supported Object Recognition. In: **Robotics: Science and Systems XI**. [S.l.]: Robotics: Science and Systems Foundation, 2015. ISBN 9780992374716. ISSN 2330765X.

[A23] QU, X.; LI, W. LLN-SLAM: A Lightweight Learning Network Semantic SLAM. In: CUI, Z. et al. (Ed.). **Intelligence Science and Big Data Engineering. Big Data and Machine Learning**. [S.l.]: Springer International Publishing, 2019. v. 11936 LNCS, p. 253–265. ISBN 9783030362034. ISSN 16113349.

[A24] STÜCKLER, J.; BEHNKE, S. Multi-resolution surfel maps for efficient dense 3D modeling and tracking. **Journal of Visual Communication and Image Representation**, v. 25, n. 1, p. 137–147, jan 2014. ISSN 10473203.

[A25] SUNDERHAUF, N. et al. Meaningful maps with object-oriented semantic mapping. In: **2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. [S.l.]: IEEE, 2017. v. 69, n. 8, p. 5079–5085. ISBN 978-1-5386-2682-5. ISSN 15792242.

[A26] TATENO, K.; TOMBARI, F.; NAVAB, N. When 2.5D is not enough: Simultaneous reconstruction, segmentation and recognition on dense SLAM. In: **2016 IEEE International Conference on Robotics and Automation (ICRA)**. [S.l.]: IEEE, 2016. v. 2016-June, p. 2295–2302. ISBN 978-1-4673-8026-3. ISSN 10504729.

[A27] TIAN, G. et al. ObjectFusion: An object detection and segmentation framework with RGB-D SLAM and convolutional neural networks. **Neurocomputing**, Elsevier B.V., v. 345, p. 3–14, 2019. ISSN 18728286.

[A28] ZHANG, L. et al. Semantic SLAM Based on Object Detection and Improved Octomap. **IEEE Access**, IEEE, v. 6, p. 75545–75559, 2018. ISSN 2169-3536.

[A29] ZHAO, Z.; CHEN, X. Semantic mapping for object category and structural class. In: **2014 IEEE/RSJ International Conference on Intelligent Robots and Systems**. [S.l.]: IEEE, 2014. p. 724–729. ISBN 978-1-4799-6934-0. ISSN 21530866.