Pontifícia Universidade Católica
do Rio de Janeiro

João Carlos Virgolino Soares

Real-Time Metric-Semantic Visual SLAM for Dynamic and Changing Environments

Tese de Doutorado

Thesis presented to the Programa de Pós–graduação em Engenharia Mecânica, do Departamento de Engenharia Mecânica da PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Engenharia Mecânica.

Advisor : Prof. Marco Antonio Meggiolaro
Co-advisor: Prof. Marcelo Gattass

Rio de Janeiro
May 2022

Pontifícia Universidade Católica
do Rio de Janeiro

João Carlos Virgolino Soares

Real-Time Metric-Semantic Visual SLAM for Dynamic and Changing Environments

Thesis presented to the Programa de Pós–graduação em Engenharia Mecânica da PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Engenharia Mecânica. Approved by the Examination Committee:

Prof. Marco Antonio Meggiolaro
Advisor
Departamento de Engenharia Mecânica – PUC-Rio

Prof. Marcelo Gattass
Co-advisor
Departamento de Informática – PUC-Rio

Prof. Luiz Carlos Pacheco Rodrigues Velho
Instituto de Matemática Pura e Aplicada

Prof. Marcelo Becker
Universidade de São Paulo

Prof. Helon Vicente Hultmann Ayala
Departamento de Engenharia Mecânica – PUC-Rio

Prof. Wouter Caarls
Departamento de Engenharia Elétrica – PUC-Rio

Rio de Janeiro, May the 11th, 2022

João Carlos Virgolino Soares

Holds a master's degree (2018) and a bachelor's degree (2015) in Mechanical Engineering, both from the Pontifical Catholic University of Rio de Janeiro (PUC-Rio). He is currently an assistant professor of Control and Automation Engineering at PUC-Rio.

Bibliographic data

Soares, João Carlos Virgolino

Real-Time Metric-Semantic Visual SLAM for Dynamic and Changing Environments / João Carlos Virgolino Soares; advisor: Marco Antonio Meggiolaro; co-advisor: Marcelo Gattass. – 2022.

133 f: il. color. ; 30 cm

Tese (doutorado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Mecânica, 2022.

Inclui bibliografia

1. Engenharia Mecânica – Teses. 2. SLAM Visual. 3. Mapeamento Métrico-Semântico. 4. Ambientes Dinâmicos. 5. Ambientes em Mudança. I. Meggiolaro, Marco Antonio. II. Gattass, Marcelo. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Mecânica. IV. Título.

CDD: 621

In memory of
João Carlos Pinheiro Soares
Isaías de Matos Balthazar
Vilson Freitas Teixeira
and Rose Mari Virgolino Teixeira

Acknowledgments

To God, for being the reason and the purpose of everything. Also, I would like to thank my family. To my wife Ana, thank you for being by my side during these 14 years of love, dreams, and partnership. To my mother Imar, my first and best teacher. To Rosemary, Lucas, Carlos Eduardo, Ignacio, Jaqueline, Pedrinho, Nadir, Norma, Vera, and Letícia.

To my advisor, prof. Marco Antonio Meggiolaro, for all these 10 years of guidance, friendship, and for the valuable lessons that I will carry for my life. To my co-advisor, prof. Marcelo Gattass, for believing in my work, and for guiding me through the first steps of computer vision.

I also would like to thank all thesis committee members. To Prof. Marcelo Becker for the welcoming reception in his laboratory, and for allowing this productive and fruitful partnership between our Universities. My sincere thanks to prof. Luiz Velho for accepting this invitation and for his interest in my work. To Prof. Wouter Caarls and Prof. Helon Ayala for helping me building my theoretical foundations during the first semesters of my thesis.

To all my colleagues from LabRob (Robotics Lab) at PUC-Rio, especially Anna Rafaela, Diego Rosa, Felipe, Arnaldo, Gustavo Duarte, Júnior and Paulo. It was always a delight to work and drink coffee with you. Special thanks to prof. Vivian Suzano and Gabriel Fischer, who helped me so much. I also would like to thank everyone from the WellRobot project at Ouro Nova. It was a incredible and rewarding experience working with you.

To all my friends from PUC-Rio, especially Rodrigo Bianchi, Gabriel Barsi, Juliana Leão, Ivan, Thiago Almeida, Bruno Calasaes, André, Marisa, Igor Girsas, Eduardo Cota, Matheus Cosenza, José Benatti, Pedro Froner, Felipe Salles, and Antônio. To all my friends from Nova Iguaçu, especially Fernando, Douglas, Fernanda, Marco Antonio Sales, and Jacqueline. To José Renato and everyone from the NachoMods' group, thank you for bringing me so much joy in these tough times.

To PUC-Rio, for being my second home for all these years. Also, to all the staff from the Mechanical Engineering Department of PUC-Rio, especially Simone and Carina.

Finally, to the Brazilian National Council for Scientific and Technological Development (CNPq) for the scholarship and financial support to this research. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Abstract

Soares, João Carlos Virgolino; Meggiolaro, Marco Antonio (Advisor); Gattass, Marcelo (Co-Advisor). **Real-Time Metric-Semantic Visual SLAM for Dynamic and Changing Environments**. Rio de Janeiro, 2022. 133p. Tese de Doutorado – Departamento de Engenharia Mecânica, Pontifícia Universidade Católica do Rio de Janeiro.

Mobile robots have become increasingly important in modern society, as they can perform tasks that are tedious or too repetitive for humans, such as cleaning and patrolling. Most of these tasks require a certain level of autonomy of the robot. To be fully autonomous and perform navigation, the robot needs a map of the environment and its pose within this map. The Simultaneous Localization and Mapping (SLAM) problem is the task of estimating both map and localization, simultaneously, only using sensor measurements. The visual SLAM problem is the task of performing SLAM only using cameras for sensing. The main advantage of using cameras is the possibility of solving computer vision problems that provide high-level information about the scene, such as object detection. However, most visual SLAM systems assume a static environment, which imposes a limitation on their applicability in real-world scenarios. This thesis presents solutions to the visual SLAM problem in dynamic and changing environments. A custom deep learning-based people detector allows our solution to deal with crowded environments. Also, a combination of a robust object tracker and a filtering algorithm enables our visual SLAM system to perform well in highly dynamic environments containing moving objects. Furthermore, this thesis proposes a visual SLAM method for changing environments, i.e., in scenes where the objects are moved after the robot has already mapped them. All proposed methods are tested in datasets and experiments and compared with several state-of-the-art methods, achieving high accuracy in real time.

Keywords

Visual SLAM; Metric-Semantic Mapping; Dynamic Environments; Changing Environment.

Resumo

Soares, João Carlos Virgolino; Meggiolaro, Marco Antonio; Gattass, Marcelo. **SLAM Visual Métrico-Semântico em Tempo Real para Ambientes em Mudança e Dinâmicos**. Rio de Janeiro, 2022. 133p. Tese de Doutorado – Departamento de Engenharia Mecânica, Pontifícia Universidade Católica do Rio de Janeiro.

Robôs móveis são cada dia mais importantes na sociedade moderna, realizando tarefas consideradas tediosas ou muito repetitivas para humanos, como limpeza ou patrulhamento. A maioria dessas tarefas requer um certo nível de autonomia do robô. Para que o robô seja considerado autônomo, ele precisa de um mapa do ambiente, e de sua posição e orientação nesse mapa. O problema de localização e mapeamento simultâneos (SLAM) é a tarefa de estimar tanto o mapa quanto a posição e orientação simultaneamente, usando somente informações dos sensores, sem ajuda externa. O problema de SLAM visual consiste na tarefa de realizar SLAM usando somente câmeras para o sensoriamento. A maior vantagem de usar câmeras é a possibilidade de resolver problemas de visão computacional que provêm informações de alto nível sobre a cena, como detecção de objetos. Porém a maioria dos sistemas de SLAM visual assume um ambiente estático, o que impõe limitações para a sua aplicabilidade em cenários reais. Esta tese apresenta soluções para o problema de SLAM visual em ambientes dinâmicos e em mudança. Especificamente, a tese propõe um método para ambientes com multidões, junto com um detector de pessoas customizado baseado em aprendizado profundo. Além disso, também é proposto um método de SLAM visual para ambientes altamente dinâmicos contendo objetos em movimento, combinando um rastreador de objetos robusto com um algoritmo de filtragem de pontos. Além disso, esta tese propõe um método de SLAM visual para ambientes em mudança, isto é, em cenas onde os objetos podem mudar de lugar após o robô já os ter mapeado. Todos os métodos propostos são testados com dados públicos e experimentos, e comparados com diversos métodos da literatura, alcançando um bom desempenho em tempo real.

Palavras-chave

SLAM Visual; Mapeamento Métrico-Semântico; Ambientes Dinâmicos; Ambientes em Mudança.

Table of contents

List of figures

List of tables

List of Abreviations

ATE – Absolute Trajectory Error

CNN – Convolutional Neural Network

EKF – Extended Kalman Filter

FPS – Frames per second

GICP – Generalized Iterative Closest Point

GPS – Global Positioning System

ICP – Iterative Closest Point

IMU – Inertial Measurement Unit

RANSAC – Random Sampling Consensus

RMSE – Root Mean Square Error

ROS – Robot Operating System

RPE – Relative Pose Error

SLAM – Simultaneous Localization and Mapping

SSD – Single Shot Detector

TSDF – Truncated Signed Distance Field

Não sou nada
Nunca serei nada
Não posso querer ser nada
À parte isso, tenho em mim todos os sonhos
do mundo

Fernando Pessoa, *Tabacaria*.

1
Introduction

Mobile robots are becoming more important each day in modern society, as they can perform tasks that are tedious or too repetitive for humans, such as cleaning and patrolling. They are also used to increase productivity in factories and farms, and to increase reliability of inspection activities. Furthermore, mobile robots can perform tasks that are too dangerous or even impossible for a human being, such as underwater or space exploration. Most of the previous tasks required a certain level of autonomy of the robot.

To be fully autonomous and perform navigation, the robot needs two main elements. First, it needs a map, i.e., some model or representation of the environment in which the robot is operating. This map can be a geometric model, position of landmarks, or even a high-level representation with semantic information. Second, it needs its pose (position and orientation) in this map.

Usually the robot does not have any prior information of the environment, neither an external measurement system or GPS available, as in indoor scenarios. Also, GPS usually does not work properly in narrow streets [12]. In these situations, the robot needs to rely only on on-board sensors. The Simultaneous Localization and Mapping (SLAM) problem is the task of creating a map of the environment using only sensor measurements without external aid, and concurrently estimating the pose of the robot in the created map. SLAM is considered a hard problem because the knowledge of the pose of the robot is required to create a consistent map, but a map is required to perform localization.

One aspect that impacts directly the formulation of the problem, the algorithms used, as well as the output, is the choice of sensors used in the robot. Laser range finders are the default sensors for several SLAM systems. However, cameras have several advantages, including lower cost and richness of information, extracting data that range sensors cannot provide, such as color and texture. RGB-D cameras, such as the Kinect, have the extra advantage of a direct depth measurement. This thesis is focused on SLAM systems with only a camera as sensor.

Another advantage of using cameras is the possibility of solving hard computer vision problems that provide high-level information of the scene, such

as object detection. These tasks improved considerably in both precision and performance in the recent years with the advances in deep learning techniques.

There are several Visual SLAM systems in the literature, with high precision and efficiency. Figure 1.1 shows the point cloud map of a static environment generated by ORB-SLAM2 [1], a state-of-the-art method for SLAM using cameras. Both map and trajectory have good accuracy.

Figure 1.1: Output of a typical visual SLAM in a static scenario. A map of a static environment generated by ORB-SLAM 2 [1] with the respective camera trajectory. The blue markers represent the poses of the camera during the mapping session.

Due to the uncertainties inherent to sensor measurements and unstructured environments, the SLAM problem is formulated using probability theory. There are three main probabilistic formulations for SLAM: extended Kalman filters, particle filters, and graph-based approaches [13]. The graph-based approach is currently the standard formulation for SLAM, and will be detailed in the following sections.

A visual SLAM system with a graph-based approach has three main steps: motion estimation, loop closure and graph optimization. Motion estimation is usually performed with a visual odometry algorithm. Loop closure is the detection of a previously visited place to estimate the errors accumulated by visual odometry. These errors are corrected during graph optimization.

The majority of visual SLAM systems assume a static environment, which imposes a limitation of their applicability in real-world scenarios. In a dynamic environment, not only the camera tracking is compromised by the presence of moving objects in the scene, but also loop detection. Figure 1.2 shows the point cloud map generated by ORB-SLAM2 of an environment containing people moving in the scene. The map was corrupted by their movement.

Figure 1.2: Map of a dynamic scene generated by ORB-SLAM2. Image from Soares et al. [2]. The people in the scene act as outliers, corrupting the camera pose estimation and populating the map.

The static assumption is very restrictive, as it precludes the SLAM system to operate in real-world scenarios, such as offices, factories, and any other human-populated environment. This thesis is focused in solving the SLAM problem in dynamic environments, assuring robustness in all three main steps of SLAM: pose estimation, data association (place recognition and loop closure) and graph optimization, using only a camera as sensor.

In addition to the previously mentioned challenges, robots in real-world scenarios need to meet a hard requirement. They need to perform expensive computations in real time, and usually they can only rely on on-board computers. Therefore, efficiency is also an important variable in this work.

1.1 Related Work

The probabilistic formulation for SLAM began in 1986 [14], but the term "SLAM" was coined in 1995 by Durrant-Whyte et al. [15]. Initially, the robots were equipped with laser scanners or sonars. The first works using cameras were proposed in the early 2000s. When cameras started to be more affordable, different problems were proposed in even more complex scenarios and applications. The problem of performing SLAM using only cameras is called "visual SLAM". This section presents the state-of-the-art visual SLAM methods in the literature and the current unsolved problems in the subject.

1.1.1 Visual SLAM

There are two main different classifications for Visual SLAM systems [16]. The first one is related to how information is extracted from the images. The methods can be either direct or indirect. Indirect methods first extract an intermediate representation from the image, such as a sparse set of keypoints to be matched against other frames. This information is used to estimate camera motion and the geometry of the scene. Direct methods, on the other hand, do not use an intermediate representation. They use pixel intensity values and optimize a photometric error. Indirect methods are, in general, faster and deal better with geometric noise in the system. On the other hand, direct methods deal better with environments with a low quantity of distinctive features.

In the second type of classification, the methods can be either sparse or dense. Sparse methods use only a set of independent points to reconstruct the scene. Dense methods, on the other hand, use all pixels.

A common misconception is that all direct methods are dense, and all indirect methods are sparse. There are dense-indirect methods, such as [17], that uses a dense optical flow. Another dense-indirect method is VOLDOR-SLAM [18], which takes a dense optical flow as input and uses a depth map alignment framework. There are also sparse-direct methods such as DSO [16], that combines photometric error minimization with a joint geometric and camera motion optimization to obtain visual odometry. However, dense-direct and sparse-indirect methods are more usual.

Examples of dense-direct methods include KinectFusion [19] and LSD-SLAM [20]. KinectFusion uses only depth data to create a dense volumetric model, and ICP to track the camera pose. LSD-SLAM is a direct SLAM method for monocular cameras which uses a tracking method that explicitly detects scale-drifts, allowing the reconstruction of large-scale maps.

The sparse-indirect formulation is the most broadly used for visual SLAM. Most sparse-indirect methods are called feature-based, because they use visual features for camera tracking and mapping. This thesis is focused on feature-based methods. MonoSLAM [21], developed by Davison et al., was one of the first Visual SLAM systems for monocular cameras. Its main drawback is the need to process every frame for map and camera pose estimations, which costs a high computational effort in exchange of little new information. The keyframe-based approaches, on the other hand, use only selected relevant frames for bundle adjustment. PTAM [22], developed in 2007 by Klien et al., is an example of a keyframe-based system. It was the first work to divide the tasks of camera tracking and mapping in two parallel threads, which

considerably reduced the computational cost. After PTAM was published, most visual SLAM algorithms used this multi-threading approach [23].

In 2015, Mur-Artal et al. [24] proposed an open-source system called ORB-SLAM. It works with three parallel threads: tracking, local mapping and loop closure. The tracking thread is responsible for localizing the camera in every frame using ORB features [25], and performing relocalization in case of a lost track. The local mapping thread performs local bundle adjustment for every new keyframe. The loop closing thread searches for loops using a robust place recognition system each time a new keyframe is added. The relocalization system is in real time with high invariance to viewpoint and illumination. ORB-SLAM is also able to work in real time in large environments. It became one of the most referenced visual SLAM systems in the literature, due to its high accuracy, robustness and scalability.

RGB-D SLAM systems have the advantage of having a direct measure of the depth. One of the first RGB-D SLAM systems was proposed by Henry et al. [26], using visual features, GICP and pose-graph optimization. Other feature-based RGB-D SLAM systems include RGBDSLAM [27] and RTAB-Map [28].

In 2017, the ORB-SLAM2 [1] was presented by Mur-Artal and Tardós as an extension of their previous work. It has the same structure and components of ORB-SLAM, but with a hybrid formulation that can be used in systems with monocular cameras, stereo cameras and RGB-D sensors. Due to its robustness and the possibility of using different camera types, ORB-SLAM2 became the basis for several SLAM systems that expanded its implementation to work in different scenarios [29][30][31][32].

In 2021, Mur-Artal and Tardós presented ORB-SLAM3 [3], an improved version of ORB-SLAM2, with a more robust place recognition system, an integration with IMU [33], and a multi-mapping system called Atlas [34], that considerably improves recovery from lost tracks.

1.1.2 Visual SLAM in Highly Dynamic Environments

Past visual SLAM systems were designed with a static environment assumption. Therefore, they are not able to handle dynamic scenarios. The ones that deal with dynamic content in the scene usually treat it as noise, and filter it using direct or feature-based methods.

StaticFusion [35] and ReFusion [36], for instance, are two direct methods for RGB-D cameras. ReFusion combines the TSFD model representation of KinectFusion with a purely geometric approach to filter the dynamic content.

StaticFusion also uses a geometric approach, but with a surfel representation.

Dib and Charpillet [37] proposed a dense visual odometry system for RGB-D cameras in dynamic environments using RANSAC. Alcantarila et al. [38] proposed a dense scene flow representation to detect moving objects using stereo cameras. Sun et al. [39] combined image differencing and a Maximum-a-posteriori estimator to perform motion removal. In another work [40], Sun et al. proposed another method for motion removal using dense optical flow. Other direct approaches include the works of Wang and Huang [41], and Kim et al. [42]. Feature-based methods include the work of Cheng et al. [43], who proposed a system based on ORB-SLAM that uses optical flow to distinguish dynamic feature points.

The previously cited approaches, however, are unable to detect *a priori* dynamic objects in the scene, such as people or cars, when they remain static. The DS-SLAM [31] system deals with dynamic objects combining optical flow with a semantic segmentation network, which allows the detection of people. SOF-SLAM [44] is a feature-based method, built on ORB-SLAM2, that combines semantic segmentation and epipolar geometry to filter dynamic features. Sun et al. [45] proposed a weakly-supervised semantic segmentation network to provide a binary mask to ORB-SLAM2 indicating movable objects, without the need for expensive annotations in the training process. However, their semantic segmentation step required 1.27 seconds to process a single image.

DynaSLAM [30] uses the Mask R-CNN [9] instance segmentation framework to obtain the pixel-wise information of people in the scene, using it to filter *a priori* dynamic features. Despite its high accuracy and robustness, DynaSLAM cannot perform in real time due to the high computational requirement of the Mask R-CNN framework.

Deep learning-based object detection has been widely applied in SLAM systems to filter dynamic features. In Detect-SLAM [46], Zhong et al. used SSD object detection only on keyframes to overcome the slow inference time of 0.31 s. Xiao et al. proposed the Dynamic SLAM [47] system, which also uses SSD object detection to filter dynamic features. They proposed a semantic correction module to create a mask with the same size of the image, to map static and dynamic points, and a selective tracking algorithm to eliminate the dynamic objects. However, the mask creation can be demanding in images with high resolution.

Liu et al. [48] uses the YOLOv3 [49] object detector combined with optical flow for dynamic feature point removal. However, their method does not remove *a priori* dynamic objects, potentially causing wrong loop closures

and odometry drifts in a scene with initially static people.

The main problem with object detection for dynamic keypoint filtering is that there are conditions that could lead to feature depletion and lost tracks. For instance, in case of a person is too close to the camera, or in a scene with many people. This happens because when the keypoints inside the bounding box are filtered, some keypoints that belong to the background are also filtered, as occurs in Dynamic-SLAM [47]. Using an instance segmentation framework to differentiate objects from the background would solve this problem, but the inference time of instance segmentation networks is very high. SGC-VSLAM [50] handled this problem by using optical flow and computing the fundamental matrix between two frames to decide which keypoints belong to objects, which is computationally demanding. A solution to this problem is proposed in Chapter 5, with a robust and fast feature repopulation algorithm for object detectors.

Besides DynaSLAM [30], several other works use instance segmentation to detect and filter dynamic objects. DP-SLAM [51] combines the semantic information of Mask R-CNN with a geometric approach based on epipolar geometry and probability propagation to classify dynamic keypoints.

SaD-SLAM [32], proposed by Yuan and Chen, combines depth information and Mask R-CNN [9] instance segmentation to find dynamic features in the image. Each feature point is individually classified as static, dynamic, or static and movable. SaD-SLAM has a high accuracy, higher than DynaSLAM [30] in some scenarios. Its main drawback is that the semantic segmentation is processed offline.

DOTMask [52], proposed by Vincent *et al.*, uses instance segmentation to obtain the pixel-wise information of the objects in the image, and an Extended Kalman Filter to track these objects. Their aim was to provide a faster SLAM system in exchange of a lower accuracy, in comparison with DynaSLAM, for example. The main problem with this approach is that the use of instance segmentation makes it still too slow, and the accuracy is considerably lower than SaD-SLAM [32] or DynaSLAM [30].

Ji et al. [53] proposed a faster Semantic RGB-D SLAM method for dynamic environments extracting semantic information only from keyframes. Also, they combined K-Means with depth reprojection to identify unknown moving objects in the other frames. Despite achieving an accuracy comparable with DynaSLAM with less computational effort, their tracking thread runs at approximately 13 FPS.

Some works consider people as *a priori* dynamic objects, such as [30] and [29]. The assumption of considering people as dynamic *a priori* may seem

strong, but in reality people are dynamic by nature and they eventually move. Mapping a scenario where most people are static for a long period is unrealistic. Furthermore, even if people in a scene are static, mapping them would lead to a future wrong loop closure when revisiting that scene after they have moved. One way to overcome this issue is to use features from static people only for tracking purposes, as done in SaD-SLAM [32]. However, this approach can be computationally expensive, especially in a crowded scenario.

Finally, none of the mentioned methods are designed to deal with crowded scenarios. Working in a crowded scene is very challenging for several reasons. First, the object detector must be prepared to deal with a high number of instances in each frame. This can be very time consuming, depending on the framework used for object detection. Also, there is the problem of feature depletion, which can be caused by filtering multiple bounding boxes.

1.1.3 Semantic SLAM

The main concept of modern semantic SLAM systems is to use high-level information provided by deep learning-based semantic framework to improve the metric localization accuracy of SLAM. Besides improving localization, the resulted semantic map can be useful for other tasks such as navigation and augmented reality. The semantic framework can perform different tasks, depending on the type of information needed. For instance, it can perform object detection, providing bounding boxes with the size and location of the objects in the scene, as well as the class of each object. Instead of object detection, the semantic framework can perform semantic segmentation, which assigns a label to every pixel in the image, as done in Kimera [54], for example. Another example is SemanticFusion [55], which is a dense slam method that combines the approach of ElasticFusion [56] with semantic segmentation.

This thesis is focused on object-based semantic SLAM, which is also called Object-SLAM. There are several SLAM systems in the literature capable of performing object mapping, such as SLAM++ [57]. However, they need to register all types of 3D objects in a database before the online process.

Other systems use 3D bounding boxes to represent objects in the map. CubeSLAM [58] is a method for object detection and SLAM using only a monocular camera. They create 3D bounding boxes using a single image and vanishing points sampling. The main advantage of CubeSLAM is to not require any prior object models.

MOLTR [59] is another example that represents objects using 3D bounding boxes. It localizes, tracks and reconstructs multiple objects in the scene.

It uses a 3D object detector that receives a single RGB image and outputs a set of object attributes, such as object class, 2D bounding box, translation, rotation with respect to the camera, and 3D dimensions. A Bayesian filter is used to track the state of the objects in the map. Its main drawbacks are the assumption of known camera pose and the run time. Using GPU, it can run at a maximum of 4 Hz in a scene with only 5 objects.

Some systems use quadrics to represent objects. For instance, consider the works of Qian et al. [60] and QuadricSLAM [61]. Sharma et al. [62] proposed an Object-SLAM method using instance segmentation to create masked keyframes, which are used for an object-level data association and map update.

Other systems use more recent deep learning-based techniques. S3LAM [63] is a monocular SLAM system based on ORB-SLAM2 that uses panoptic segmentation to segment objects from the background, detecting clusters of 3D points. Panoptic segmentation is a combination between instance and semantic segmentations. S3LAM runs at 20 FPS, but is designed for static environments. More details about the differences between object detection, semantic segmentation, instance segmentation and panoptic segmentation are presented in Chapter 2.

1.1.4 Visual SLAM in Changing Environments

Despite still being an open problem, methods for visual SLAM in highly dynamic environments have received increased attention from the Robotics community in the recent years, especially with the development of new deep learning techniques that help solving the problem. However, none of the previous approaches deal with other types of dynamic factors that happen in a real environment.

One of the situations usually not considered in methods for dynamic environments such as DynaSLAM [30], DS-SLAM [31] or SaD-SLAM [32], is when a change happens after the robot has already mapped the scene. When it revisits the scene, some objects are in different locations, some are missing, and new objects may have appeared. This is often referred in the literature as SLAM in low dynamic environments [64][65], semi-static environments [66], changing environments [67], or simply long-term mapping [68].

The term "changing environments" was chosen as the more appropriate for the task, as "low dynamic" or "semi-static" can be used in the context of a scene with objects moving slowly in front of the camera, and "long-term mapping" emphasizes the scalability issue.

An early solution to this problem was proposed by Walcott-Bryant et al. [65] in 2012. They proposed a method for planar indoor environments with robots using laser scanners. They proposed a dynamic pose-graph that could be edited, removing poses according to scan matching results.

Lee and Myung [69] showed through experiments that the wrong loop closures caused by a moved object could not be solved by pose-graph optimization techniques robust to outliers, such as Switchable Constraints [70], Max Mixtures [71] or Dynamic Covariance Scaling [72].

Rosen et al. [66] proposed a method to model environmental change of features over time, called feature persistence, using a recursive Bayesian estimator. Hashemifar and Dantu [73] extended Rosen's formulation, incorporating the persistence filter to ORB-SLAM and testing in a real environment.

Gomez et al. [64] developed a method for dealing with changing environments on an object level. To create a 3D bounding box of an object, they use 2D object detection and point cloud to estimate the centroid position and object dimensions. They use a floodfill algorithm and the median of the 2% smallest depths within the 2D bounding box to extrapolate the maximum and minimum depths of the object. Also, they create an object-based pose graph, connecting the robot poses and objects. The graph is updated computing the probability of finding the object in that location based on new measurements. The main drawback of their formulation is that the robot always revisits the same locations to update the object-graph. Thus, they do not perform SLAM, but mapping with known poses.

Zhao et al. [67] proposed a framework for lifelong localization and 2D mapping, tracking the changes in the scene and maintaining an updated map accordingly through a technique called pose-graph refinement. Their method uses IMU, wheel encoders and LiDAR measurements. Lazaro et al. [74] also proposed a method for changing environments using laser scans.

Derner et al. [75] proposed a method for visual localization in changing environments. Their method uses a previously built visual database, used to perform matching against query images to determine the pose of the robot.

Schmid et al.[76] proposed a method for mapping in changing environments using panoptic segmentation to build and maintain volumetric maps during operation, receiving robot poses from an external estimator.

A survey of robust SLAM systems from 2021 [77] mentions DX-SLAM [78] as the only example of a visual SLAM system designed for lifelong operations. DX-SLAM is a visual SLAM method that uses features from a deep convolutional neural network. Despite considerably improving robustness in changing environments, deep features alone did not improve robustness in

dynamic environments.

None of the mentioned methods deal with both dynamic and changing environments using cameras. The ones that deal with changing environments are either not robust to highly dynamic environments, or assume a known camera pose or a known map, i.e., do not perform SLAM.

Besides moving objects, other possible problems in changing environments are changes due to illumination or weather, and deformable objects. However, these problems are not in the scope of this work.

1.1.5 Datasets for Visual SLAM

Datasets and benchmarks are very important for the advances of SLAM research, as they provide an accessible way for comparing multiple methodologies and evaluate them with clear criteria. There are several datasets for visual SLAM in the literature, each one focused on a different problem, with different types of raw data and ground-truth.

The KITTI dataset [79] is used for the evaluation of several outdoor problems, including visual odometry, visual SLAM, multi-object tracking, segmentation, among others. It contains monocular, stereo and RGB-D data.

The TUM RGB-D dataset [80] is one of the most used for the evaluation of visual SLAM systems. It has 39 sequences of static scenarios, scenes with dynamic objects, with low texture, among others. It uses two evaluation metrics: the absolute trajectory error (ATE), which is suited to evaluate SLAM systems, and the relative pose error (RPE), which is suited to evaluate visual odometry drift or loop closure accuracy. The ground-truth was made using a motion capture system. Similar to the TUM dataset is the Bonn RGB-D [36]. It uses same evaluation metrics from the TUM dataset, but with the focus on highly dynamic scenarios.

The OpenLORIS-Scene Dataset [81] was developed for real environments with several challenges that were not embraced by past datasets, such as changing environments, changing view point, and illumination.

There are also datasets designed for evaluating long-term operations, such as KAIST Day/Night Dataset [82], the Oxford Robotcar Dataset [83], and others made for extreme environmental conditions, such as the CityScapes Dataset [84], which containts foggy scenes, or the Multi-Sensor Perception (Marulan) Dataset [85], with smoke, dust, and rain.

However, these datasets are focused on outdoor environments, dealing with changes in illumination or weather. None of the previously mentioned

datasets are specifically designed for the evaluation of visual SLAM systems in indoor changing environments.

1.2 Original Contributions

The main goal of this thesis is to perform visual SLAM in real time with robustness to both dynamic and changing environments, with the aid of high-level object detection and mapping.

This thesis presents several contributions to the field of visual SLAM. First, it presents Crowd-SLAM, the only-to-date visual SLAM method specifically designed to operate in crowded environments. This work also includes a newly trained network called CYTi, optimized for crowded environments. Both Crowd-SLAM and CYTi are open-source.

Also, a method is presented for highly dynamic environments that can operate in real time with high accuracy. The proposed method has a robust keypoint classification algorithm that filters *a priori* dynamic objects and uses an Extended Kalman Filter to track movable objects in the scene. This resulted in a visual SLAM system for highly dynamic environments that runs faster than DOT-Mask[52] and the method of Ji et al. [53], with an accuracy similar to DynaSLAM [30] and SaD-SLAM [32]. Furthermore, the problem of feature depletion caused by filtering features from the background in the bounding boxes is solved with a fast and reliable method, using statistical data of the depth in each bounding box.

This thesis also presents the first dataset especially designed for the evaluation of the robustness of visual SLAM methods in changing environments. The data is collected using an RGB-D camera, while a robust motion capture system is used to generate a ground-truth.

Finally, to the best of our knowledge, this thesis proposes the first methodology for Semantic SLAM in both dynamic and changing environments. The system uses the MapPoints derived from feature detection, combined with the output of an object detector to determine the 3D centroid of the objects in the scene, and create an object-level semantic map that maintains a belief about the pose of each mapped object. This results in a real-time 3D object detection using a semantic point clustering approach, without the need for instance or panoptic segmentation or an off-the-shelf 3D object detector. A robust long-term data association is also proposed, using the object centroids. The state of the objects in the map is updated using a Bayesian filter. Different from the approaches found in the literature, the proposed method does not assume a known camera pose, nor a known map *a priori*.

All proposed SLAM systems are evaluated using benchmark datasets such as the TUM and Bonn datasets, and compared with several previously mentioned state-of-the-art methods, including ORB-SLAM2, ORB-SLAM3, DynaSLAM, SaD-SLAM, and DOT-Mask.

1.3 Thesis Outline

This thesis is divided into 8 chapters that are structured as follows. Chapters 2 contains the theoretical background, Chapters 3-7 present the proposed contributions, and Chapter 8 shows the conclusions and suggestions for future work.

Chapter 1 is the introductory section of the work, with motivation, literature review, objectives, and achieved contributions.
Chapter 2 presents the fundamental concepts and theoretical background of feature-based visual SLAM with a graph-based formulation, and explains the main techniques based on deep learning for computer vision problems.
Chapter 3 proposes a methodology for the SLAM problem in highly dynamic environments with *a priori* dynamic objects.
Chapter 4 proposes a methodology for the SLAM problem in crowded environments, including comparisons with state-of-the-art methods using datasets.
Chapter 5 proposes a methodology for the SLAM problem in highly dynamic environments with movable and dynamic objects.
Chapter 6 shows the details about the PUC/USP dataset for changing environments.
Chapter 7 proposes a real-time Semantic SLAM system for both dynamic and changing environments.
Chapter 8 presents the conclusions, propositions for future works and the publications made during the development of this thesis.

2
Theoretical Background

2.1
Introduction

The aim of this chapter is to formulate the basic concepts regarding feature-based visual SLAM in static scenarios, using a graph-based probabilistic formulation, and present the main techniques based on deep learning used to solve computer vision problems related to SLAM. Section 2.2 explains the pinhole camera model and intrinsic camera parameters. Section 2.3 is dedicated to present the probabilistic formulation of the SLAM problem using the graph-based approach. Section 2.4 presents details about camera tracking and loop closure, and Section 2.5 lists frameworks of graph optimization. Section 2.6 presents the output of a feature-based SLAM system. Finally, Sections 2.7 and 2.8 present the main differences between object detection, instance segmentation and panoptic segmentation.

2.2
Camera Model

The pinhole camera model, shown in Fig. 2.1, is used in this thesis to establish a relationship between a point in the image and a point in 3D space.

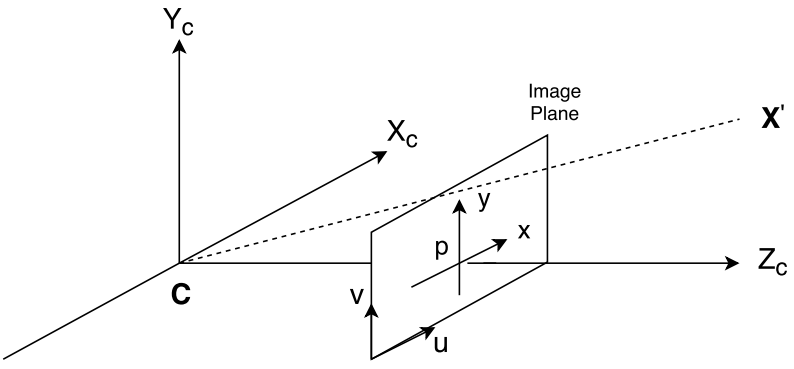


Figure 2.1: Pinhole Camera Model

Using the pinhole camera model, the relationship between a 3D point in the environment with X', Y' and Z' coordinates and a point in the image plane is given by:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & c_x & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} \tag{2-1}$$

where $[u, v, 1]$ are the coordinates of the mapped point in the image plane, written in homogeneous coordinates, f is the focal length of the camera, and c_x and c_y are the coordinates of the optical center. These parameters are called the camera intrinsic parameters, and are obtained through calibration.

With an RGB-D camera, it is possible to extract 3D coordinates of a point in the image by

$$x = \frac{(u - c_x)}{f_x} z \tag{2-2}$$

$$y = \frac{(v - c_y)}{f_y} z \tag{2-3}$$

$$z = depth(v, u) \tag{2-4}$$

where z is the depth directly measured by the sensor.

2.3 Probabilistic Formulation

The Graph-SLAM is currently the most used approach for SLAM, due to its accuracy and efficiency. The approach is divided into two main steps: front-end and back-end. The front-end is responsible for the graph construction, and comprises two main tasks: pose estimation and loop closure. Pose estimation is the process of locally estimating the pose of the robot, while loop closure is the long-term data association. The back-end is responsible for the graph optimization, to estimate an optimal trajectory of the robot and an accurate map of the environment. The general system is shown in Fig. 2.2.

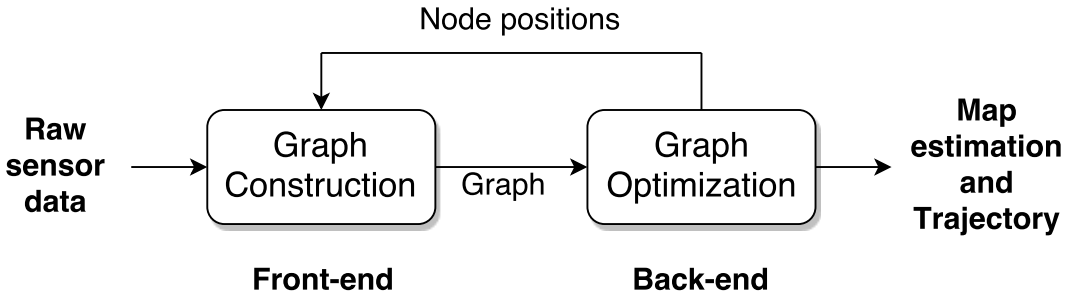
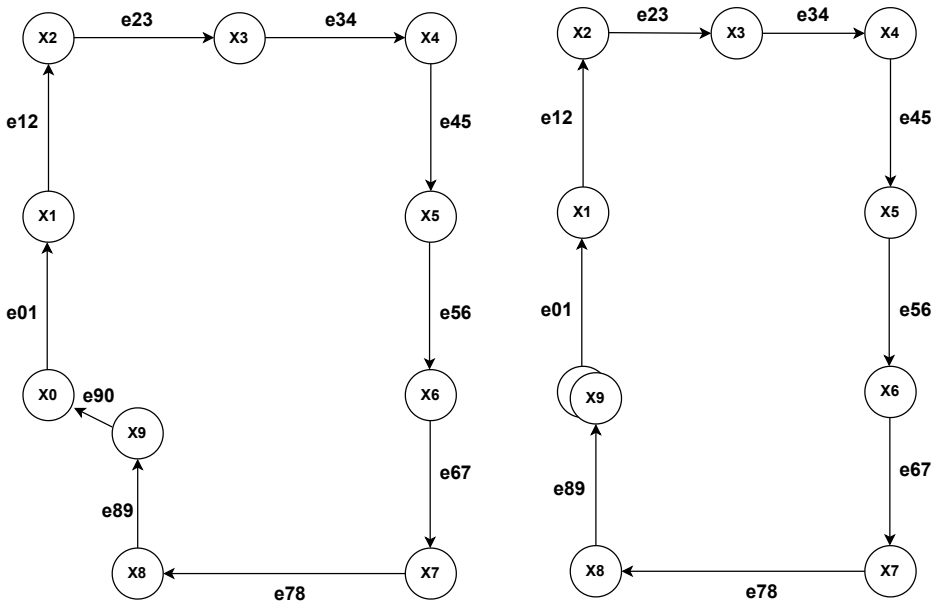


Figure 2.2: Graph-SLAM system

The problem is represented by a graph, where the nodes are the poses of the robot and the edges are measurement constraints between the poses. The

objective is to optimize the error constraints caused by odometry drift. There are two types of edges in visual SLAM: the ones created with visual odometry and the loop closure edges. Fig 2.3(a) shows the graph before the optimization. The edges e01 to e89 represent visual odometry estimations between poses. The edge e90 represents a loop closure. In visual SLAM methods, the loop is detected with a type of place recognition algorithm that evaluates if the robot is in a previously visited place, and then establishes a relation between the current and the old pose. After the loop closure, the graph is optimized and the drifts caused by visual odometry are corrected, as shown in the schematic example of Fig. 2.3(b).



2.3(a): Pose-graph with loop closure 2.3(b): Pose-graph after optimization

Figure 2.3: Schematic of a pose-graph being optimized after loop closure

The mathematical formulation of the graph-based SLAM approach is based on Maximum-a-posteriori estimation problems [86], where the objective is to find the state X^* that maximizes the belief of the state X given the measurements Z:

$$X^* = \underset{X}{\operatorname{argmax}}\, p(X|Z) = \underset{X}{\operatorname{argmax}}\, p(Z|X)\, p(X) \tag{2-5}$$

where $p(Z|X)$ is the likelihood of the measurements Z given the state X, and p(X) is the prior probability. Assuming the measurements Z are independent, Eq. (2-5) leads to

$$X^* = \underset{X}{\operatorname{argmax}}\, p(X) \prod_{k=1}^{m} p(z_k|X) = \underset{X}{\operatorname{argmax}}\, p(X) \prod_{k=1}^{m} p(z_k|X_k) \tag{2-6}$$

Moreover, assuming the measurement noise is a zero-mean Gaussian, then

$$p(z_k|X_k) \propto exp\left(-\frac{1}{2}\|h_k(X_k) - z_k\|^2_{\Omega_k} \right) \tag{2-7}$$

where Ω_k is the information matrix associated with the measurement. Finally, the MAP estimation can be written as

$$X^* = \underset{X}{\text{argmin}} \; -log\left(p(X) \prod_{k=1}^{m} p(z_k|X_k) \right) = \underset{X}{\text{argmin}} \sum_{k=0}^{m} \|h_k(X_k) - z_k\|^2_{\Omega_k} \tag{2-8}$$

which is a nonlinear least squares problem, solved using the frameworks mentioned in Section 2.5.

2.4 Front-end

2.4.1 Camera tracking

The tracking system is responsible for finding feature matches between frames to estimate the motion of the camera. There are several robust feature detectors, the most used are SIFT [87], SURF [88], and ORB [25]. Figure 2.4 shows ORB feature detection performed by ORB-SLAM3 [3].

Figure 2.4: Feature detection performed by ORB-SLAM3 [3]

2.4.2 Loop Closure

Loop closure is one of the most important steps in SLAM, as it is a vital step to achieve a consistent map and trajectory, correcting the drift caused by odometry. Visual SLAM systems usually detect loops using a technique called place recognition.

The bag-of-words is a popular method for place recognition systems that represents images by visual words from a vocabulary. These words are obtained from local descriptors such as SIFT [87], or FAST+BRIEF. The bag-of-words method allows high-speed comparisons between images over large datasets, instead of a slow direct comparison between features. It quantizes features into a vocabulary, increasing efficiency. Several place recognition systems use this technique, such as FAB-MAP [89], FAB-MAP2 [90], and DBoW2 [91].

RTAB-Map [28], for instance, uses the place recognition system from Labbe and Michaud [92], and ORB-SLAM2 [1] uses DBoW2 [91]. Both are based on the bag-of-words methodology.

2.5 Back-end

There are several graph-optimization frameworks in the literature, for instance $\sqrt{SAM}$ (*square root SAM*) developed by Dellaert and Kaess [93] in 2006, and iSAM, proposed by Kaess et al. [94] in 2008.

The most broadly used is the g^2o framework [95], presented in 2011 by Kümmerle et al. It is an open-source C++ framework that has been used as a back-end in several monocular, stereo and RGB-D SLAM implementations, such as ORB-SLAM [24], ORB-SLAM2 [1], ORB-SLAM3 [3] and RGBDSLAM [27]. It can use different solvers such as Cholesky, Preconditioned Conjugate Gradient and Levenberg-Marquardt.

2.6 Sparse Map

Sparse maps are usually composed of a set of sparse 3D landmarks. Figure 2.5 shows a feature-based map generated by ORB-SLAM3 [3]. The camera poses are represented in blue, and the points are landmarks. The green lines represent the edges of the graph. This map can be used later for relocalization. Even though sparse maps such as the ones generated by ORB-SLAM3 cannot be used directly for navigation, there are techniques, such as the one proposed by Cheng and Liu [96], that can reconstruct the map as a navigable environment.

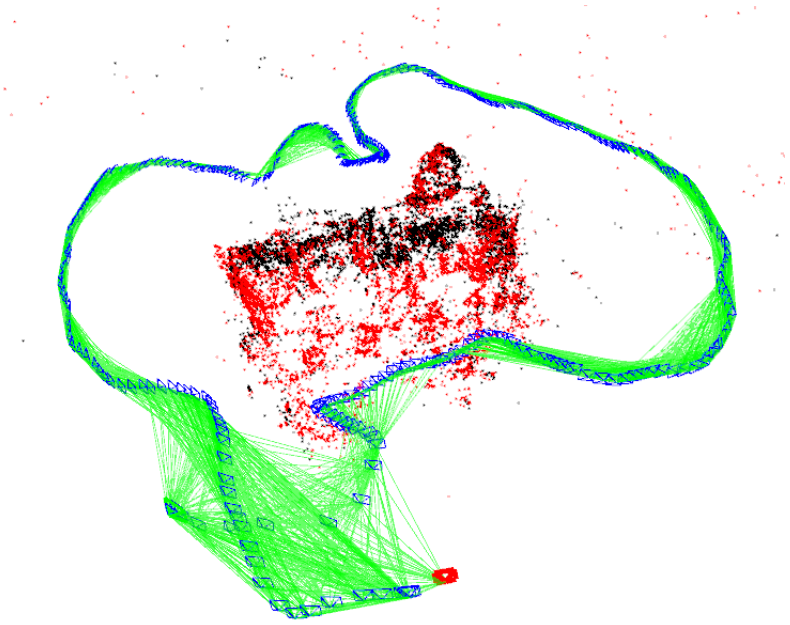


Figure 2.5: Sparse map built with ORB-SLAM3

2.7 Deep Learning

Despite being robust and accurate, classical computer vision methods are being surpassed by Deep Learning techniques. Figure 2.6, adapted from Goodfellow et al. [97], shows the main differences among AI disciplines. The green boxes represent components that are able to learn from data. Classical approaches use hand-designed programs, and modern Machine Learning systems rely only on data. The main difference between Deep Learning and Machine Learning is the increased number of intermediate layers. This is only possible due to a higher amount of available data and modern GPUs.

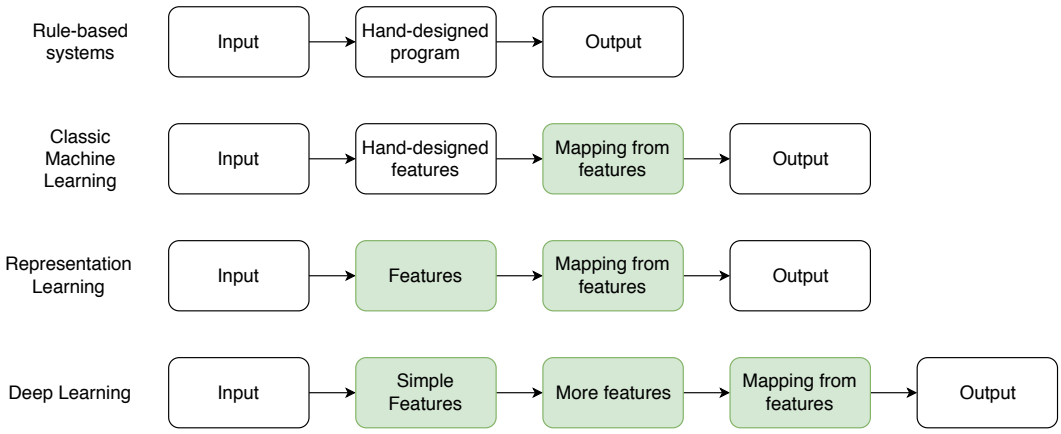


Figure 2.6: Computer Vision Tasks

2.8 Computer Vision Tasks

Figure 2.7, from [4], shows four different types of computer vision tasks that are important for Visual SLAM systems. Image classification is a simple task, in which just the class of objects is obtained. Object detection is the task of determining the class, general shape and location of objects within an image. Semantic segmentation associates each pixel of an image to a class label. Instance segmentation is a combination of object detection and semantic segmentation. The next sections will further discuss the object detection and instance segmentation tasks.

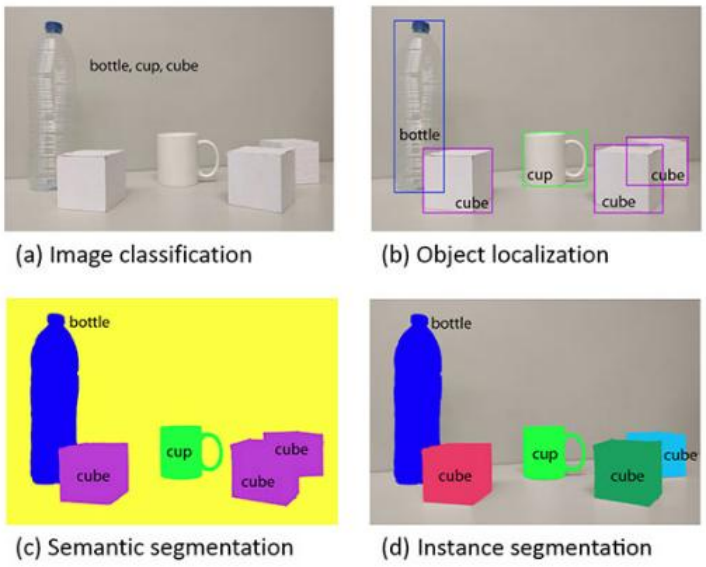


Figure 2.7: Computer Vision Tasks, from [4]

2.8.1 Object Detection

Object detection is the task of determining the location of objects in an image, and also the class of each detected object. There are different types of deep learning-based object detection algorithms. They all provide the classes of the detected objects, the 2D bounding boxes with their corresponding positions, and a confidence number.

The R-CNN detectors use convolutional neural networks (CNN) to extract features from images. The first R-CNN approach [5] generates approximately 2000 regions in the image, called region proposals. A CNN extracts features from these regions and a classification algorithm would determine the presence of the object. Figure 2.8 shows the model of the R-CNN. The main

problem with this implementation is the high inference time of about 47 seconds for each image [6].

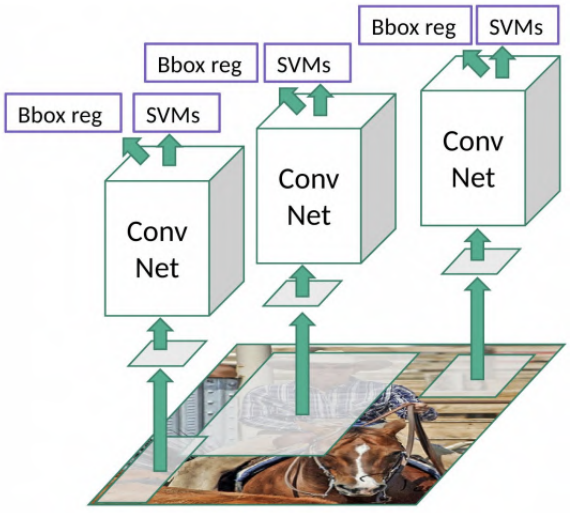


Figure 2.8: R-CNN Model from Girshick et al. [5]

The Fast R-CNN algorithm [6] considerably reduced the inference time of its predecessor, by feeding the entire image to the CNN instead of 2000 region proposals, as shown in the model of Fig. 2.9. The output of the CNN is a convolutional feature map. The region proposals are identified from the feature map, and then classified.

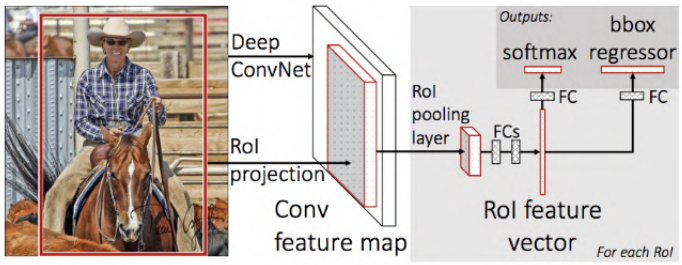


Figure 2.9: Fast R-CNN Model from Girshick [6]

A new version was proposed in 2015, called Faster R-CNN [7], introducing a technique called Region Proposed Network. Figure 2.10 shows the Faster R-CNN model.

Despite being accurate, these algorithms are not able to work in real time due to their complex pipelines. "One shot detectors", on the other hand, are much more efficient, as they do not use region proposals to localize objects in the image.

The YOLO (You Only Look Once) technique, from Redmon et al. [8], works as a single regression problem, using a convolutional network to predict the bounding boxes and their class probabilities. Figure 2.11 shows the YOLO

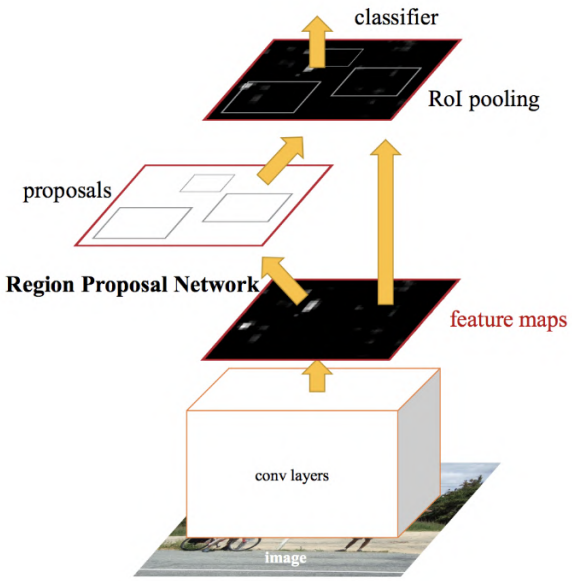


Figure 2.10: Faster R-CNN Model from Ren et al. [7]

model. The input image is divided in a grid, in which each cell predicts one set of class probabilities and a number of bounding boxes. Each bounding box consists of 5 predictions: the x and y coordinates of the center of each box relative to the border of the cell, its width, height and the confidence. The bounding boxes whose class probabilities are above a threshold are selected.

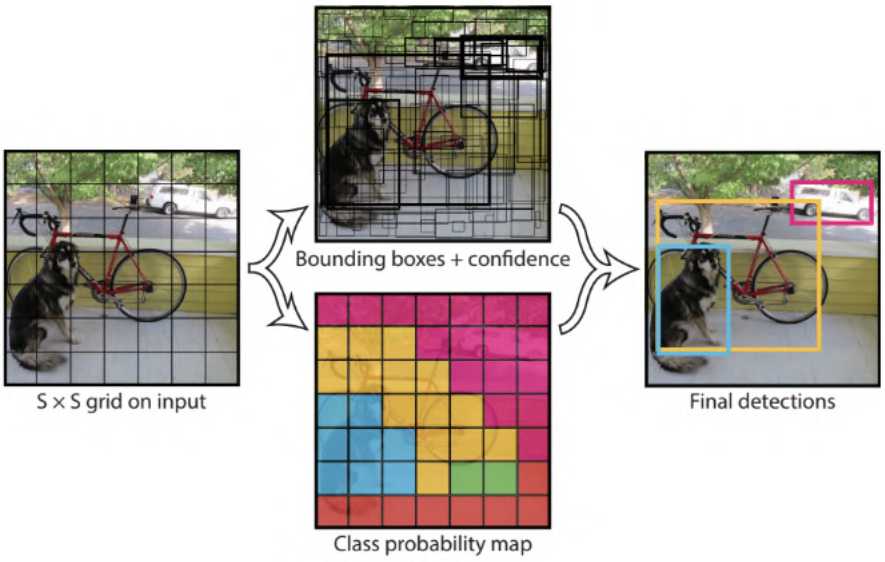


Figure 2.11: YOLO Model from Redmon et al. [8]

Other example of "one shot" object detector is the Single Shot Detector (SSD) [98], which is used by several SLAM systems [99][47].

2.8.2 Instance Segmentation

Instance segmentation is a combination of object detection and semantic segmentation. It gives the pixel-wise information of the detected classes and the bounding boxes with the respective location of the objects.

Mask R-CNN, develop by He et al. [9], is one example of an instance segmentation framework. Figure 2.12, from [9], shows the framework of Mask R-CNN. It consists of an extension of the Faster R-CNN object detection framework, with an extra neural network for each region of interest to predict the object mask. Figure 2.13 shows a comparison between the outputs of YOLO and Mask R-CNN. Other instance segmentation frameworks include YOLOACT [100] and YOLACT++ [101].

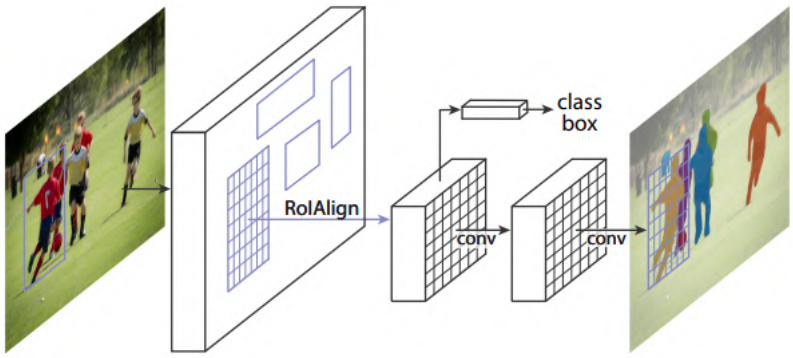
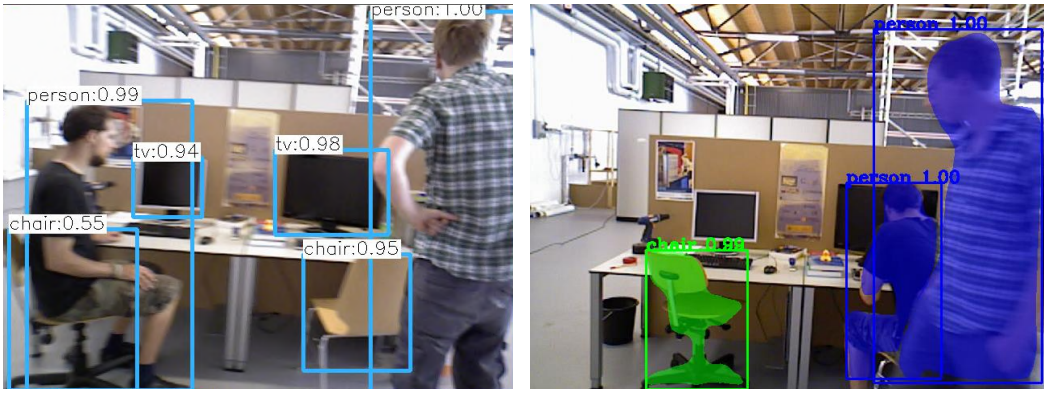


Figure 2.12: Mask R-CNN Model from He et al. [9]



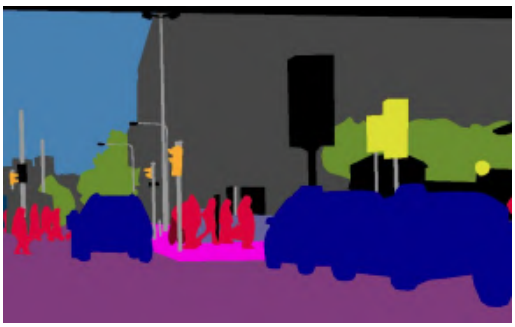
2.13(a): YOLO Output 2.13(b): Mask R-CNN Output

Figure 2.13: Comparison between the outputs of Object Detection (YOLO) and Instance Segmentation (Mask R-CNN)

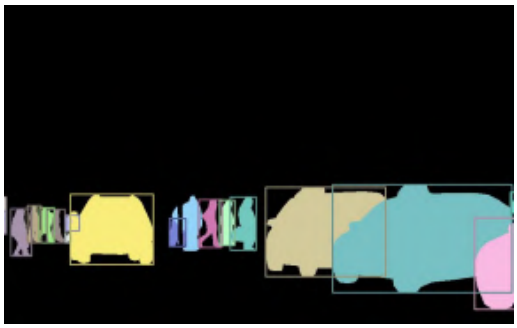
2.8.3 Panoptic Segmentation

Panoptic segmentation was proposed by [10] as a combination of instance and semantic segmentation. Figures 2.14(a) through 2.14(d), from [10], show the differences among the outputs of semantic, instance and panoptic segmentation. Figure 2.14(a) shows the original image. Figure 2.14(b) shows the output of semantic segmentation, where there is a clear distinction among which pixels belong to the sky, building, floor, people and cars. However, there is no way to distinguish one car or one person from another. On the other hand, instance segmentation can distinguish each car and each person, but it cannot label different parts of the background, as shown in Fig. 2.14(c). Panoptic segmentation can identify the class of all pixels in the image and all instances. Recent SLAM methods that use panoptic segmentation include [63] and [102].

2.14(a): Original Image



2.14(b): Semantic Segmentation



2.14(c): Instance Segmentation

2.14(d): Panoptic Segmentation

Figure 2.14: Output differences among semantic, instance, and panoptic segmentation, from [10]

In the next chapter, a first approach to visual SLAM in human populated scenarios is proposed using the concepts presented in the previous sections.

3
Mapping in Human Populated Environments

3.1
Introduction

This chapter proposes a simple solution to visual SLAM in human environments using a deep learning-based object detection technique combined with an activation filter, considering only *a priori* dynamic objects [103].

RTAB-Map (Real-Time Appearance-Based Mapping) [28][92] is a SLAM system with a robust place recognition methodology and an efficient memory management system. However, it does not work properly when there are moving objects in the scene. Figure 3.1 shows the point cloud map generated by the RTAB-Map system using data from a sequence of an RGB-D dataset. This sequence is used to evaluate the capability of the SLAM system to deal with dynamic environments. The map is corrupted by the movement of people.

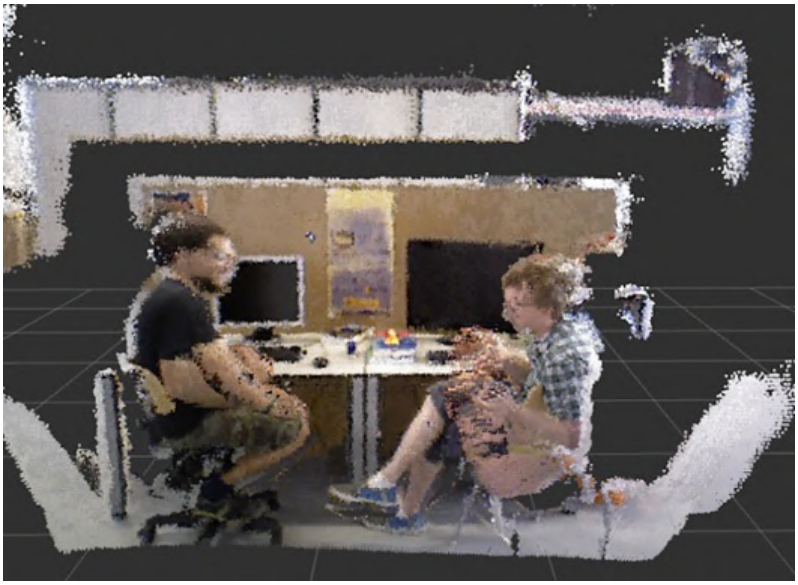


Figure 3.1: Point cloud Map of a dynamic scene using RTAB-Map

This chapter proposes an extension of the RTAB-Map system to allow its use in dynamic scenarios, using a deep learning-based object detection system to filter the frames containing people, which are *a priori* dynamic objects. The TUM RGB-D SLAM dataset [80] is used to evaluate the proposed system and

a mobile robot is used to perform experiments. The system is built as a Robot Operating System (ROS) package.

Section 3.2 explains the methodology used to solve the problem. Section 3.3 presents the evaluation of the system using an RGB-D dataset, the robot used for indoor experiments and the results obtained. Section 3.4 presents the conclusions and suggestions for future work.

3.2 Methodology

The proposed methodology is composed of three main parts: SLAM, object detection, and an activation filter, which are detailed in the following sections. The SLAM system receives sensor data and outputs the point cloud map of the environment, along with the optimized trajectory of the robot. The object detection system and the filter are responsible for restricting the data sent to the SLAM framework. Figure 3.2 shows the block diagram of the proposed methodology.

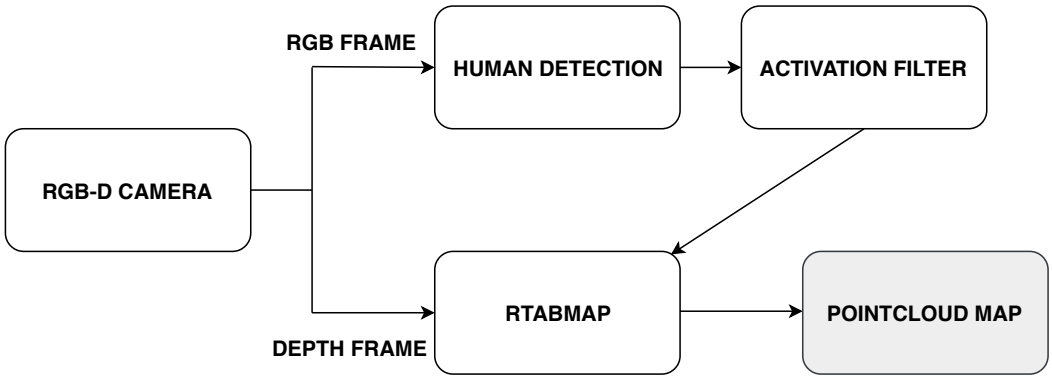


Figure 3.2: Diagram of the proposed methodology

3.2.1 Object Detection

This work uses the OpenCV implementation of YOLO, trained with the COCO dataset [104]. Figure 3.3 shows an image with detected objects. It shows different categories being classified, such as people, and other static objects, with their respective bounding boxes and confidence numbers.

3.2.2 Filter

In the RGB-D mode, the RTAB-Map framework is only activated when it receives both depth and RGB images. Instead of sending both images to the

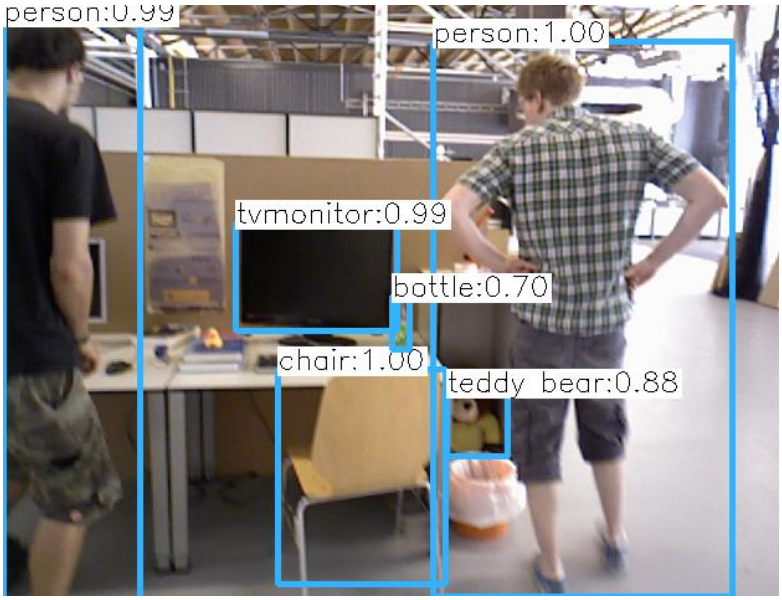


Figure 3.3: Final Object Detection

RTAB-Map, the RGB images pass through the object detector and a filter. If there are no people detected in the frame, then it is sent to RTAB-Map. In the other case, the system waits for the next frame. With this methodology, no person is added to the final map.

3.3 Results

The system was tested using a Dataset with a dynamic sequence and with a robot in a real environment.

3.3.1 TUM Dataset

The RGB-D dataset from the Technical University of Munich [80] was used to evaluate the system. It provides several sequences of color and depth images obtained from a Kinect sensor. The sequence "fr3_walking_xyz" was chosen for the evaluation, with 28 seconds of duration and $5.791m$ of trajectory length. This sequence is suitable to evaluate the robustness of SLAM algorithms to people moving quickly in the scene. The camera is moved along three directions whilst two persons are moving around. Figure 3.4 shows the final map using only the RTAB-Map system. Figure 3.5 shows the final map using the proposed methodology.

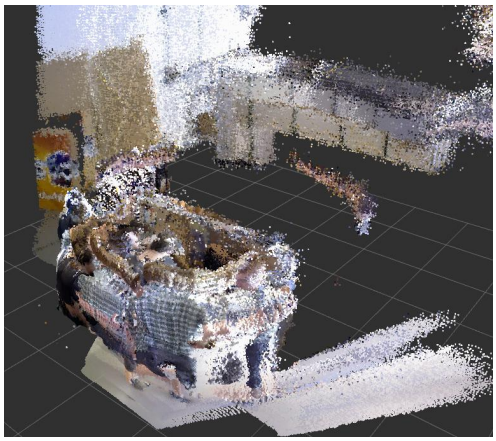


Figure 3.4: Mapping without object detection filter

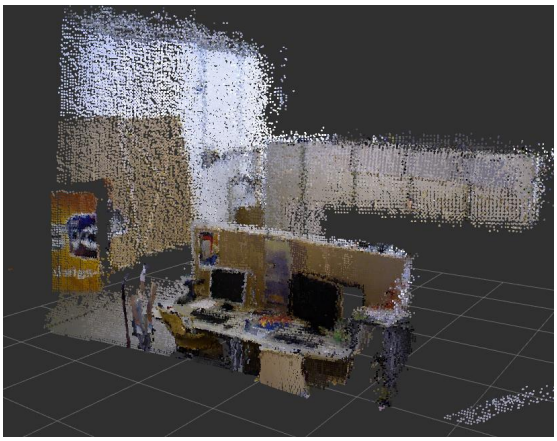


Figure 3.5: Mapping with object detection filter

3.3.2 Experimental Evaluation

Indoor experiments were conducted in the Robotics Laboratory from the Pontifical Catholic University of Rio de Janeiro. Figure 3.6 shows the mobile robot used in the experiments, an iRobot Create, equipped with a ZED Stereo Camera and a Jetson TX2 board. The iRobot Create is a commercial differential drive platform equipped with an encoder in each wheel.

Figure 3.6: Experimental Setup

The ZED camera is a stereo camera developed by Stereolabs. It provides

visual odometry and depth information, besides the stereo images. Despite not having a direct depth measurement, it has several advantages. The ZED camera is lighter than other similar cameras, has a single USB connection for both power and signal, has a high field of view, and a high depth range, as shown in Table 3.1.

Table 3.1: ZED Camera Specifications

Max. resolution	4416 x 1242
Field of View	90º x 60º
Max. frame rate	100 Hz
Mass	159 g
Depth range	0.5 - 20 m

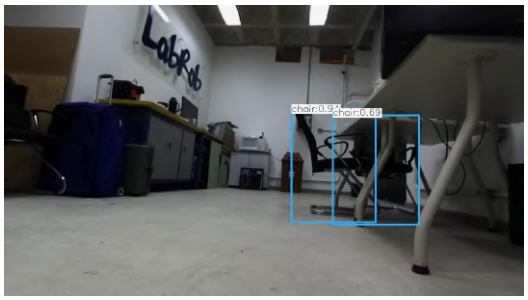
Besides the object detector/filter developed and RTAB-Map, the following third-party ROS nodes were used in this work:

- Rviz: A visualization tool used to show the point cloud map
- ZED-ros-wrapper: The camera driver
- create_autonomy: iRobot driver

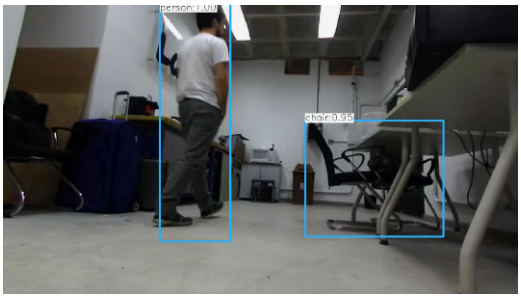
The ZED driver provides the camera RGB and depth images, sent to the object detector and filter. The robot driver receives velocity commands that are sent to the wheels and sends wheel odometry readings to RTAB-Map, which can also be used with visual odometry.

In the experiments, the robot performed SLAM with a person walking in the scene. Figures 3.7(a) and 3.7(b) show RGB images received by the ZED camera with the object detection system. A person is detected in Fig. 3.7(b). Figure 3.8(a) shows the point cloud map without the filter. The person corrupted the final map. Figure 3.8(b) shows that the person was filtered from the map using the proposed approach.

The experiments show that the proposed system was able to prevent people from corrupting the point cloud map. One drawback of this approach is not being able to map the environment while people are in front of the camera.



3.7(a): Object Detection



3.7(b): Person detected

Figure 3.7: Object detection in human populated environments

3.8(a): Without object detection filter

3.8(b): With object detection filter

Figure 3.8: Mapping in human populated environments

3.4 Conclusions and Discussion

This work presented an extension to the RTAB-Map system to perform SLAM in environments with people. Dataset tests showed that the proposed approach was successful. The system also worked on a mobile robot performing SLAM in an indoor environment with one person walking through the front of the robot. Despite effective, the proposed methodology has limitations, such as the need for having the scene clear of people to enable the SLAM problem to be solved. In the next chapter, a method for visual SLAM in human environments that addresses this limitation is presented.

4
Visual SLAM in Crowded Environments

4.1
Introduction

The main challenges of performing SLAM in dynamic environments are: to detect dynamic objects in the scene, to prevent those objects from being tracked, and to exclude them from the map. Some SLAM systems that work in dynamic environments rely on purely geometric approaches to detect moving objects. However, they usually fail to detect the presence of *a priori* dynamic objects, e.g., people, when they are initially static, which can lead to odometry drifts or long-term wrong loop closures. Computer vision tasks, such as object detection and instance segmentation, provide semantic information of the scene that allows the recognition of such objects.

There is an increasing number of Visual SLAM systems relying on deep learning object detectors to filter the dynamic content of images. However, they are neither efficient nor suitable for working with crowds. The main goal of this work is to create a SLAM system capable of working in crowded environments in real time, by using a custom YOLO Tiny network specialized in people detection in crowds, and an algorithm for keypoint filtering.

Thus, this chapter introduces Crowd-SLAM [29], a new open-source[1] Visual SLAM system for crowded environments based on ORB-SLAM2. The YOLO object detection is used to filter people in the scene. The YOLO network is trained using a dataset for crowded environments, achieving a real-time performance with high precision. The proposed methodology is evaluated using multiple datasets and compared with state-of-the-art systems. As ORB-SLAM2, Crowd-SLAM works with monocular, stereo, and RGB-D cameras.

This chapter is organized as follows. Section 4.2 details the proposed methodology, Section 4.3 shows the results using three different datasets, and finally the conclusions and suggestions for future work are presented in Section 4.4.

[1] https://github.com/virgolinosoares/Crowd-SLAM

4.2 Methodology

Figure 4.1 shows an overview of the proposed methodology, composed of four threads: Object Detection, Tracking, Local Mapping, and Loop Closing. The four threads run in parallel. The RGB images are processed in the object detection and tracking threads simultaneously. The tracking thread extracts ORB features [25] and waits for the bounding boxes provided by the object detection thread. The feature points and bounding boxes are sent to the dynamic keypoint filter and feature number update system inside the tracking thread. The dynamic keypoint filter removes the keypoints inside the bounding boxes, and the feature update system changes the feature number proportionally to the total filtered area, in order to prevent lost tracks.

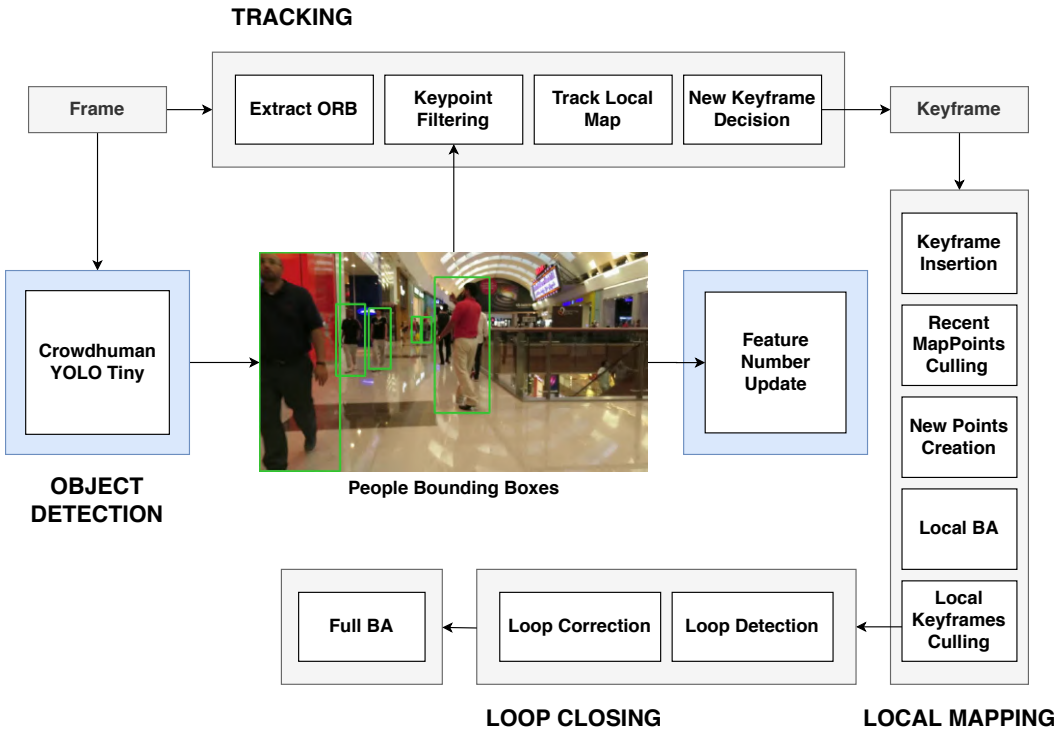


Figure 4.1: Framework of Crowd-SLAM

4.2.1 SLAM

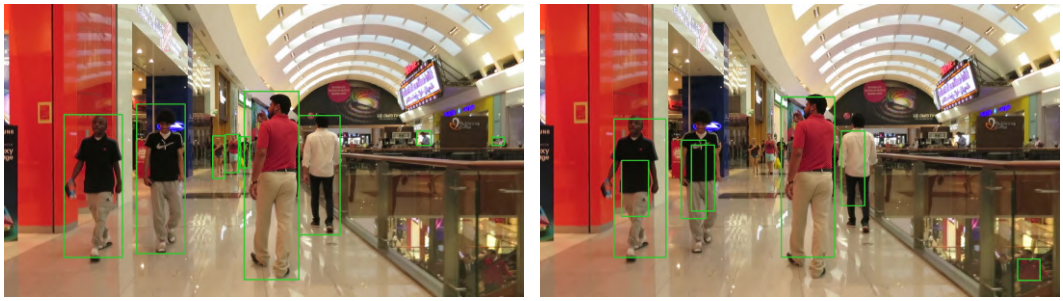
This work uses ORB-SLAM2 as the global SLAM solution. ORB-SLAM2 has three main threads: tracking, loop closing, and local mapping. Crowd-SLAM uses the same loop closing and local mapping threads of ORB-SLAM2. However, the tracking thread was modified to include the outlier removal

algorithm and the feature number update system. Also, a new thread was added for Object Detection.

ORB-SLAM2 works using a keyframe-based methodology. The tracking thread decides whether every new frame is a new keyframe. The loop closing system compares the information of every new keyframe with past keyframes, searching for new closed loops using a bag-of-words place recognition module based on DBoW2 [91]. Once a loop is detected, the graph is optimized with the g2o framework [95] to assure a consistent trajectory. ORB-SLAM2 outputs a sparse point cloud map, and the optimized trajectory of the camera.

4.2.2 People Detection

YOLOv3 provides the classes of the detected objects, 2D bounding boxes with their corresponding positions, and a confidence number for each box. Figure 4.2(a) shows a detection in an image from the MOT Challenge 2020 [105], using a YOLOv3 framework trained with the COCO dataset [104]. YOLOv3 Tiny is a version of YOLO with fewer layers and filters, which has 10 times higher inference speed. However, it has a lower accuracy. Figure 4.2(b) shows an example of a YOLOv3 Tiny detection, also trained with the COCO dataset.



4.2(a): YOLOv3 4.2(b): YOLOv3 Tiny

Figure 4.2: Comparison between YOLOv3 and YOLOv3 Tiny

The MS COCO [104], used in YOLOv3, has 80 different object classes. However, only the people class is needed in this work. We propose the Crowdhuman YOLO Tiny (CYTi), a specialization of YOLO-Tiny that is more accurate in crowded environments with people.

CYTi is trained with the Crowdhuman dataset [106], composed of more than 20000 pictures of people in crowded environments, with 470000 humans and an average density of 23 people per image, much more than other people datasets. Figure 4.3 shows one image of the used training data. 15000 images were used for training and 4370 for validation. The training and validation were

performed on an NVIDIA Quadro P2000 GPU using the Darknet framework [107]. The batch size was set to 24 with a learning rate of 0.001.

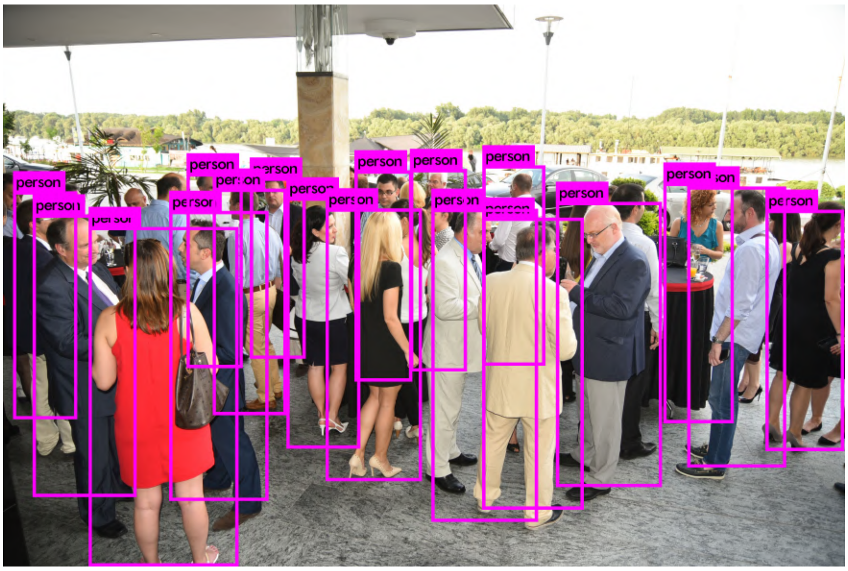


Figure 4.3: YOLO detection in an image from the Crowdhuman Dataset

The Multiple Object Tracking (MOT) Challenge dataset [108, 105] is used to evaluate the improvements of the newly trained network in crowded environments. Several metrics are used to measure the detection capabilities. The *precision*, stated in Eq. (4-1), is the rate between true positives (TP) and the total number of detections, which is the sum of true positives and false positives (FP).

$$Precision = \frac{TP}{(TP + FP)} \tag{4-1}$$

The *recall*, stated in Eq. (4-2), is the rate between true positives and the sum of true positives and false negatives (FN).

$$Recall = \frac{TP}{(TP + FN)} \tag{4-2}$$

The Multiple Object Detection Precision (MODP) is stated by Eq. (4-3)

$$MODP = \frac{\sum_{t=1}^{N_{frames}} \frac{OverlapRatio}{N_{mapped}(t)}}{N_{frames}} \tag{4-3}$$

where N_{frames} is the total number of frames, $N_{mapped}(t)$ is the number of mapped objects in the frame t, and the overlap ratio is the sum of the intersection over union of every object for every frame.

According to Stiefelhagen et al. [109], the Multiple Object Detection Accuracy (MODA) is defined by Eq. (4-4)

$$MODA = 1 - \frac{\sum_{i=1}^{N_{frames}} (m_i + fp_i)}{\sum_{i=1}^{N_{frames}} N_G^i} \tag{4-4}$$

where m_i and fp_i are, respectively, the missed detections and false positives in the frame i, and N_G^i is the number of objects in the frame i.

Two sequences of the MOT17 and MOT20 challenges were selected: MOT20-01, MOT20-02, MOT17-09, and MOT17-11. The first two are crowded scenes in an indoor train station with a static camera. The MOT17-11 is a sequence in a crowded shopping mall with a forward-moving camera. The MOT17-09 is an outdoor crowded scene, with a static camera close to the people.

Table 4.1 shows the detection results using the YOLO Tiny, YOLOv3, and CYTi for the MOT challenge sequences. Besides the previous metrics, they also show the total number of true positives, false negatives, and false positives. The down and up arrow symbols next to the metric names mean that the lower or higher the number, the better it is for the overall detection performance, respectively.

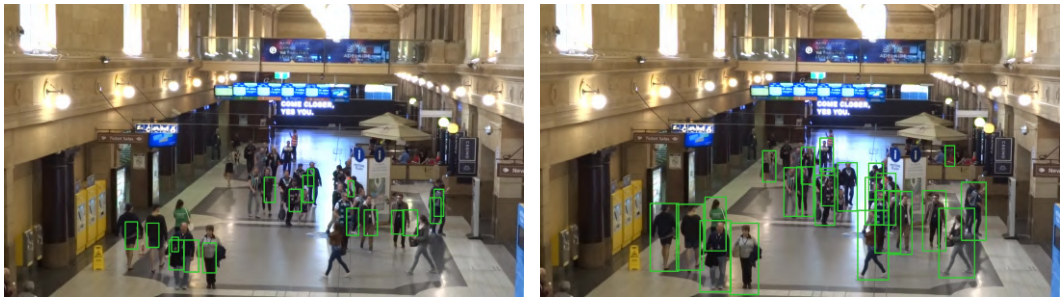
CYTi outperforms the YOLO Tiny network in every metric of every sequence. YOLO Tiny has a poor performance in these sequences, especially in MOT20-01, not finding a single true positive. The results of YOLOv3 are also compared as reference. CYTi maintained the inference speed of YOLO Tiny, being more than ten times faster than YOLOv3, while increasing the detection accuracy.

Table 4.1: Detection results for YOLO Tiny, YOLOv3 and CYTi

Sequences	Method	MODA ↑	MODP ↑	TP ↑	FN ↓	FP ↓	Prec. ↑	Recall ↑
	Tiny	-69.4	0.0	0	8924	6192	0.0	0.0
MOT20-01	YOLOv3	4.6	70.0	**6851**	**2144**	6437	51.6	**76.2**
	CYTi	**11.4**	**73.2**	6609	2441	**5575**	**54.2**	73.0
	Tiny	-67.4	64.2	78	20001	13612	0.4	0.6
MOT20-02	YOLOv3	8.1	69.7	**15971**	**4279**	14324	**78.9**	52.7
	CYTi	**14.0**	**73.4**	14872	5356	**12037**	73.5	**55.3**
	Tiny	-71.8	75.4	856	2184	3040	22.0	28.2
MOT17-09	YOLOv3	1.8	**77.4**	**2722**	**398**	2665	50.5	**87.2**
	CYTi	**60.6**	77.2	2528	559	**657**	**79.4**	81.9
	Tiny	-35.6	74.9	2064	3666	4104	33.5	36.0
MOT17-11	YOLOv3	29.5	**80.3**	**4917**	**1062**	3156	60.9	**82.2**
	CYTi	**52.4**	78.4	4786	1169	**1665**	**74.2**	80.4

Figures 4.4(a) and 4.4(b) show, respectively, the object detection output in a crowded scene from the MOT Challenge 2020 [105] using the YOLO

Tiny network and CYTi. The improvement in both precision and accuracy is noticeable.



4.4(a): YOLOv3 Tiny

4.4(b): CYTi

Figure 4.4: Comparison between YOLOv3 Tiny and CYTi

4.2.3 Outlier Removal

Once the images pass through the people detector, the keypoints that belong to people are removed from the image. The Dynamic keypoint filtering algorithm is as follows. The point (x, y) of D^{F_k} corresponds to the coordinates of the top left corner of the bounding box. w and h are the width and height of the box, respectively.

Unlike other systems, this algorithm does not need a mask of the frame with information about static and dynamic regions. It uses directly the bounding boxes to perform the filtering, therefore it does not depend on the image size.

Figures 4.5(a) and 4.5(b) show the keypoint detection of ORB-SLAM2 and with the proposed object detection filter, respectively. The filters successfully erased all keypoints in the regions with people. The keypoints appearing in the left chair are also erased, due to becoming merged with the person bounding box, resulting in an indirect filtering of potential dynamic objects.

4.2.4 Feature Number Update

If a large number of keypoints are filtered from a single frame, the information available for the SLAM system may not be enough to perform tracking. Two main problems can occur simultaneously or independently. First, there can be too many people in the scene. Secondly, one person can be too close to the camera, occupying most of the image. In both situations, the problem is not the number of people, but the total filtered area of the image.

Algorithm 1: Dynamic keypoint filtering algorithm

Data: Frame F_k, bounding box list D^{F_k}(x,y,w,h), keypoints p^{F_k}

```
new_keypoints
for p_i in Frame F_k do
    bool_key = false;
    for people in D^{F_k} size do
        box = D[people];
        if p_i inside box then
            bool_key is true;
            break;
        end
    end
    if bool_key is false then
        new_keypoints append p_i;
    end
end
p^{F_k} = new_keypoints;
```

4.5(a): ORB-SLAM 4.5(b): Crowd-SLAM

Figure 4.5: Comparison of feature detection between ORB-SLAM and the proposed approach

Even with the robust relocalization system of ORB-SLAM2, specifically designed to recover from a lost track, a crowded scene can prohibit the SLAM process. To overcome this issue, we propose a module to check the filtered area and update the number of detected ORB features, instead of setting a static high number. The number of feature points starts with a given initial value, and increases 300 for 30% of filtered area, 500 for 60%, 700 for 90%, and 1200 for more than 95%. This method benefits the performance, because more features extracted implies more computational effort, and simply defining a high static value would slow down the tracking without need, in the case of no people in the scene.

Figures (4.6(a))-(4.6(c)) show three scenes with (a) no people, (b) one

person, and (c) two persons. The number of detected keypoints is increased in the third image due to the increased filtered area.

4.6(a): With no people in the scene

4.6(b): With 1 people in the scene

4.6(c): With 2 people in the scene

Figure 4.6: Feature number update

4.3 Results

4.3.1 TUM Dataset

Crowd-SLAM was numerically evaluated using the TUM RGB-D dataset [80]. It contains sequences of RGB and depth images obtained from a Microsoft Kinect camera, with their corresponding ground truth trajectories. The data was recorded at 30Hz with a 640 x 480 resolution.

Two types of sequences were used in this evaluation. In the fr3_w sequences, two people are walking in the room, moving behind a desk, passing in front of the camera, and sitting on chairs. These sequences are, therefore, highly dynamic. The fr3_s sequences can be considered low-dynamic, as people are sitting, making movements mainly with their hands. Both types are used in this evaluation.

There are four types of camera motion considered: xyz, rpy, half, and static. For the motion xyz, the camera is moved along the three axes, keeping the same orientation. In the rpy sequence, the camera is rotated over roll, pitch, and yaw axes. In the half sequence, the camera follows the trajectory of a half-sphere. In the static sequence, the camera is manually kept at the same position and orientation. Table 4.2 shows the duration, trajectory length, average translational velocity, and average rotational velocity of the camera for every sequence.

The Absolute Trajectory Error (ATE) [80] is used to evaluate the global consistency of the estimated trajectory, comparing the absolute distances between the translational components of the estimated and ground truth

Table 4.2: Details of each TUM dataset sequence

Sequence	Duration [s]	Length [m]	Vel. [m/s]	Vel. [deg/s]
fr3_s_static	23.63	0.259	0.011	1.699
fr3_s_xyz	42.50	5.496	0.132	3.562
fr3_s_rpy	27.48	1.110	0.042	23.841
fr3_s_half	37.15	6.503	0.180	19.094
fr3_w_static	24.83	0.282	0.012	1.388
fr3_w_xyz	28.83	5.791	0.208	5.490
fr3_w_rpy	30.61	2.698	0.091	20.903
fr3_w_half	35.81	7.686	0.221	18.267

trajectories. Equation (4-5) shows the computation of the ATE at a time step i:

$$ATE_i = E_i^{-1} T G_i \tag{4-5}$$

where E is the estimated trajectory, G represents the ground truth, and T is the transformation that aligns the two trajectories. For a sequence of N poses, the RMSE of ATE is given by Eq. (4-6).

$$RMSE(ATE_{1:N}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \|trans(ATE_i)\|^2} \tag{4-6}$$

The Relative Pose Error (RPE) is used to evaluate the translational and rotational drifts of the trajectory over a fixed interval Δ. The RPE at a time step i is shown in Eq. (4-7). The RMSE of RPE is given by Eq. (4-8).

$$RPE_i = (G_i^{-1} G_{i+\Delta})^{-1} (E_i^{-1} E_{i+\Delta}) \tag{4-7}$$

$$RMSE(RPE_{1:N}, \Delta) = \sqrt{\frac{1}{m}\sum_{i=1}^{m} \|trans(RPE_i)\|^2} \tag{4-8}$$

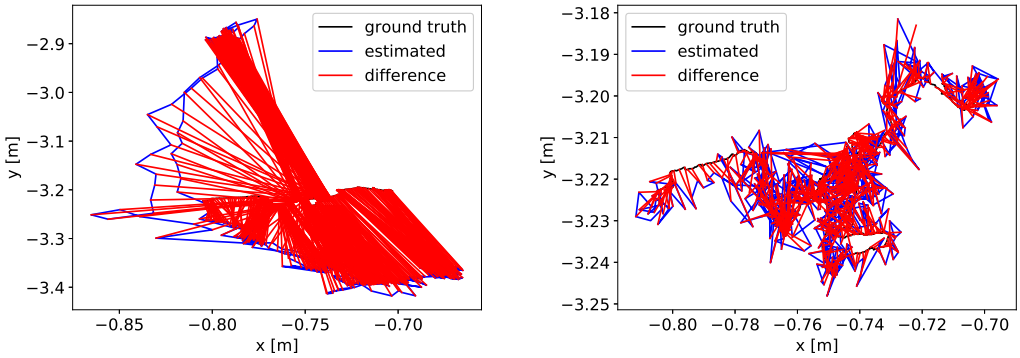
where $m = N - \Delta$.

All tests were performed five times and the median results were used for the evaluation, as proposed by Mur-Artal and Tardós [1], to consider the non-deterministic nature of the system.

Figures 4.7(a) through 4.11(b) show the ATE plots from ORB-SLAM2 and Crowd-SLAM for the fr3_w_static, xyz, rpy, halfsphere, and fr3_s_xyz sequences.

In the fr3_sitting_xyz sequence, all three trajectories are close to the ground truth. The low-dynamic nature of this sequence allows ORB-SLAM2 to eliminate the few dynamic features through its outlier detection methods,

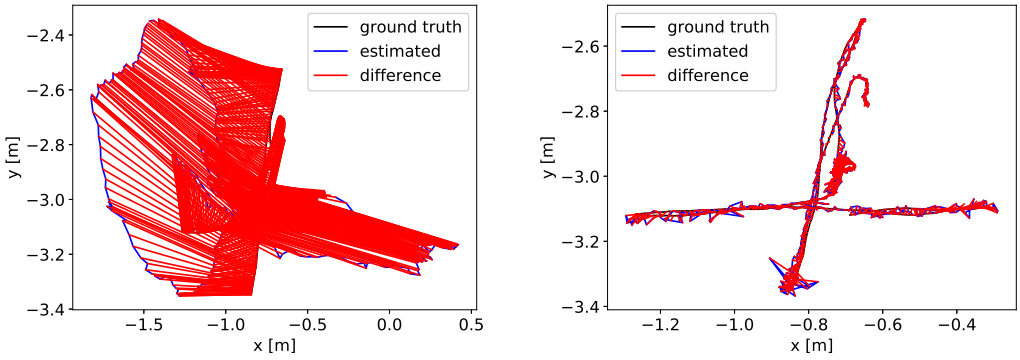
such as RANSAC. In the walking sequences, on the other hand, ORB-SLAM2 is not able to detect the highly dynamic features and the estimated trajectories deviate from the ground truth.



4.7(a): ORB-SLAM2

4.7(b): Crowd-SLAM

Figure 4.7: Ground truth and estimated trajectory in the sequence fr3_walking_static



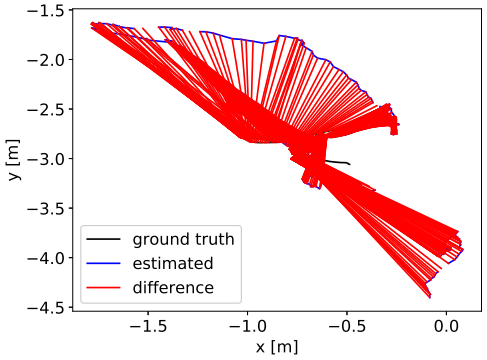
4.8(a): ORB-SLAM2

4.8(b): Crowd-SLAM

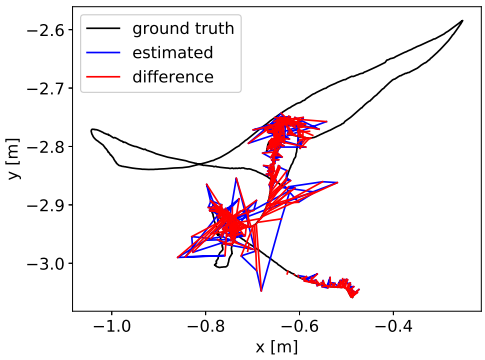
Figure 4.8: Ground truth and estimated trajectory in the sequence fr3_walking_xyz

Table 4.3 shows the Root Mean Square (RMSE) of the ATE comparison between Crowd-SLAM and other direct methods for dynamic environments: ReFusion [36], StaticFusion [35], and the works of Sun et al. [39, 40]. ReFusion and StaticFusion results were obtained in the work of Palazzolo et al. [36]. Our system outperformed the direct methods in all evaluated sequences.

Crowd-SLAM was also compared with ORB-SLAM2 and three feature-based methods for dynamic environments based on ORB-SLAM2: DS-SLAM [31], DynaSLAM [30], and SOF-SLAM [44]. The results are shown in Table 4.4. DynaSLAM has the best results in four sequences. However, the difference between their results and Crowd-SLAM is between 1mm and 9mm, depending

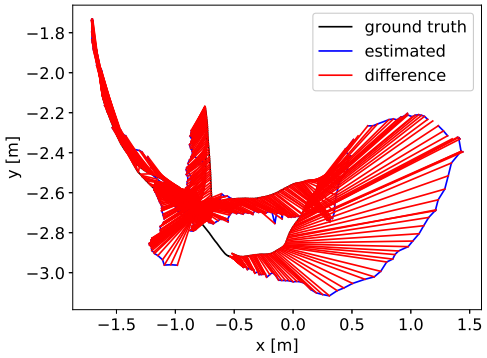


4.9(a): ORB-SLAM2

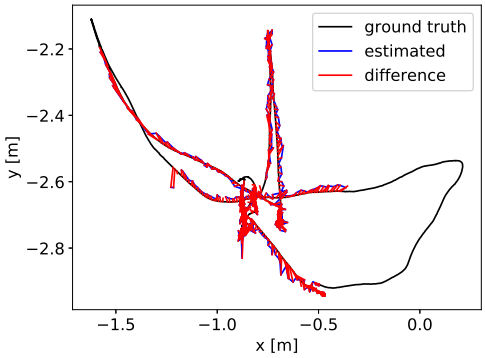


4.9(b): Crowd-SLAM

Figure 4.9: Ground truth and estimated trajectory in the sequence fr3_walking_rpy



4.10(a): ORB-SLAM2



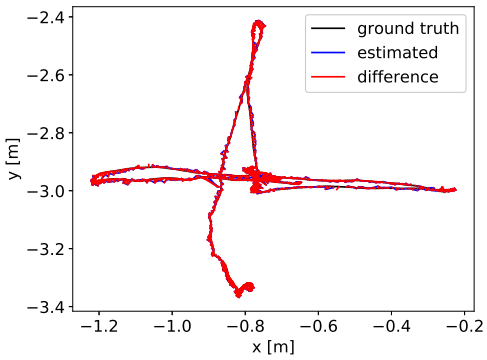
4.10(b): Crowd-SLAM

Figure 4.10: Ground truth and estimated trajectory in the sequence fr3_walking_halfsphere

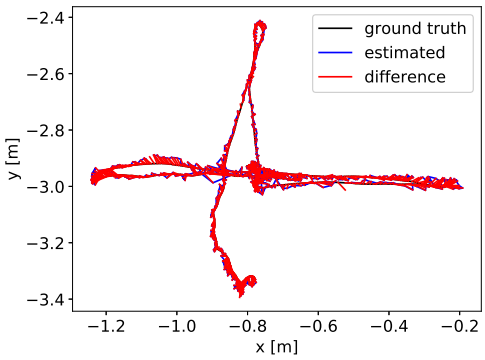
on the sequence. As DynaSLAM is an offline method, our results are considered satisfactory.

Crowd-SLAM was also compared with three Visual SLAM systems that use object detection to filter dynamic content: Liu et al. [48], Detect SLAM [46], and Dynamic SLAM [47]. The results are shown in Table 4.5. Our system achieved better results in two sequences, Dynamic SLAM achieved better results in four sequences, and Liu et al. in two sequences. Overall, the results of the four methods are similar, except in the fr3_w_rpy result of Detect-SLAM, which had a higher error.

Tables 4.6 and 4.7 show the RMSE of translational and rotational drifts (RPE), respectively, of Crowd-SLAM against ORB-SLAM2, DS-SLAM, and two works of Sun et al. [39, 40], in m/s and deg/s. In the translational drift analysis, Crowd-SLAM outperformed the other works in five sequences, achieving values similar to the best results on the other three sequences. In



4.11(a): ORB-SLAM2



4.11(b): Crowd-SLAM

Figure 4.11: Ground truth and estimated trajectory in the sequence fr3_sitting_xyz

Table 4.3: Comparison of the RMSE of ATE [m] of Crowd-SLAM against Sun et al., StaticFusion and ReFusion using the TUM dataset

Sequence	Sun et al.[39]	Sun et al.[40]	StaticFusion	ReFusion	Crowd-SLAM
fr3_s_static	—	—	0.014	0.009	**0.008**
fr3_s_xyz	0.048	0.051	0.039	0.040	**0.018**
fr3_s_half	0.047	0.066	0.041	0.110	**0.020**
fr3_w_static	0.065	0.033	0.015	0.017	**0.007**
fr3_w_xyz	0.093	0.066	0.093	0.099	**0.020**
fr3_w_rpy	0.133	0.073	—	—	**0.044**
fr3_w_half	0.125	0.067	0.681	0.104	**0.026**

the rotational drift analysis, Crowd-SLAM achieved the best results in six sequences. For instance, in the fr3_w_rpy sequence, Crowd-SLAM achieved nearly half the error of DS-SLAM.

4.3.2 Bonn RGB-D Dynamic Dataset

Another evaluation was made using the Bonn RGB-D Dynamic Dataset [36]. It is a dataset with highly dynamic sequences, with people walking and performing different tasks. It was recorded with an Asus Xtion Pro Live Sensor and an Optitrack Prime 13 motion capture system for the ground truth. It also has the same evaluation metrics of the TUM dataset.

Five sequences of the dataset were chosen for the evaluation: crowd1, crowd2, crowd3, synchronous1, and synchronous2. Despite having less abrupt

Table 4.4: Comparison of the RMSE of ATE [m] of Crowd-SLAM against ORB-SLAM2, DS-SLAM, DynaSLAM, and SOF-SLAM using the TUM dataset

Sequence	ORB-SLAM2	DS-SLAM	DynaSLAM	SOF-SLAM	Crowd-SLAM
fr3_s_static	0.008	**0.006**	—	0.010	0.008
fr3_s_xyz	**0.009**	—	0.015	—	0.018
fr3_s_rpy	0.019	—	—	—	**0.015**
fr3_s_half	0.021	—	**0.017**	—	0.020
fr3_w_static	0.409	0.008	**0.006**	0.007	0.007
fr3_w_xyz	0.724	0.024	**0.015**	0.018	0.020
fr3_w_rpy	0.781	0.444	0.035	**0.027**	0.044
fr3_w_half	0.374	0.030	**0.025**	0.029	0.026

Table 4.5: Comparison of the RMSE of ATE [m] of Crowd-SLAM against Liu et al., Detect SLAM, and Dynamic SLAM using the TUM dataset

Sequence	Liu et al.	Detect-SLAM	Dynamic SLAM	Crowd-SLAM
fr3_s_static	**0.006**	—	—	0.008
fr3_s_xyz	—	0.020	**0.006**	0.018
fr3_s_rpy	—	—	0.034	**0.015**
fr3_s_half	—	0.023	**0.015**	0.020
fr3_w_static	0.010	—	—	**0.007**
fr3_w_xyz	0.016	0.024	**0.013**	0.020
fr3_w_rpy	**0.042**	0.296	0.060	0.044
fr3_w_half	0.031	0.051	**0.021**	0.026

camera movements in comparison with the TUM sequences, for example roll spin, the sequences of the Bonn dataset have more challenging scenarios. Figure 4.12 shows a frame where most of the keypoints are filtered by the presence of 3 people close to the camera. Even so, the system is able to perform tracking.

As done in the TUM sequences, all tests were performed five times and the median results were used for the evaluation. Figs. 4.13(a) to 4.16(b) show the ATE plots from ORB-SLAM2 and Crowd-SLAM for the crowd and synchronous sequences. All ORB-SLAM2 trajectories deviate from the ground truth. Our system, on the other hand, was able to achieve low errors.

Table 4.8 shows the ATE comparison between Crowd-SLAM and DynaSLAM, ReFusion [36], and StaticFusion [35]. Their results were obtained in the work of Palazzolo et al. [36]. Our system outperformed all methods in three

Table 4.6: RMSE values of the Translational Drift (RPE) in m/s of Crowd-SLAM against ORB-SLAM2, DS-SLAM, and Sun et al. using the TUM dataset

Sequence	ORB-SLAM2	DS-SLAM	Sun et al.[39]	Sun et al.[40]	Crowd-SLAM
fr3_s_static	0.009	**0.008**	—	—	0.009
fr3_s_xyz	**0.011**	—	0.033	0.036	0.020
fr3_s_rpy	0.025	—	—	—	**0.021**
fr3_s_half	0.024	—	0.046	0.055	**0.022**
fr3_w_static	0.234	**0.010**	0.084	0.031	**0.010**
fr3_w_xyz	0.384	0.033	0.121	0.067	**0.025**
fr3_w_rpy	0.373	0.150	0.175	0.097	**0.065**
fr3_w_half	0.323	**0.030**	0.167	0.061	0.037

Table 4.7: RMSE values of the Rotational Drift (RPE) in deg/s of Crowd-SLAM against ORB-SLAM2, DS-SLAM, and Sun et al. using the TUM dataset

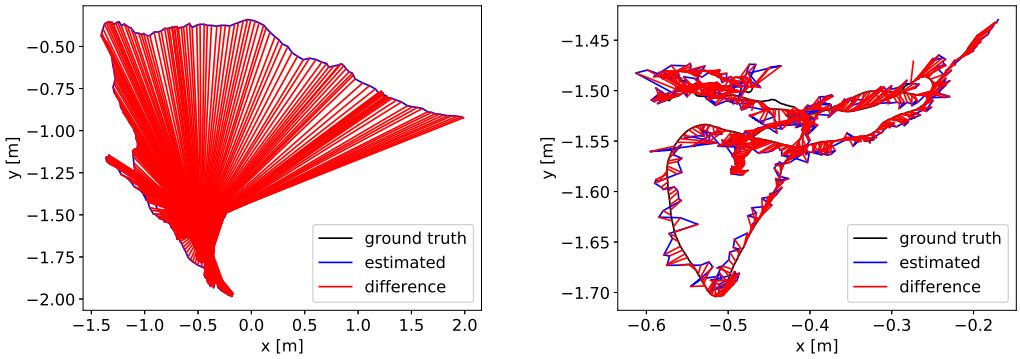
Sequence	ORB-SLAM2	DS-SLAM	Sun et al.[39]	Sun et al.[40]	Crowd-SLAM
fr3_s_static	0.289	0.273	—	—	**0.261**
fr3_s_xyz	0.483	—	0.983	1.036	**0.478**
fr3_s_rpy	0.784	—	—	—	**0.508**
fr3_s_half	**0.598**	—	2.375	2.268	0.653
fr3_w_static	4.207	0.269	2.049	0.900	**0.265**
fr3_w_xyz	7.302	0.826	3.235	1.595	**0.658**
fr3_w_rpy	7.229	3.004	4.375	2.593	**1.519**
fr3_w_half	5.960	**0.814**	5.010	1.900	0.820

sequences, including DynaSLAM. In all three *crowd* sequences, Crowd-SLAM outperformed ORB-SLAM2, StaticFusion, and ReFusion by a high margin.

4.3.3 ETH Stereo Dataset

Despite being broadly used as a benchmark for Visual SLAM systems, the TUM Dataset is not ideal for a good evaluation of crowded environments, as their dynamic sequences only contain at maximum of two people in the scene. Also, the Bonn RGB-D Dynamic Dataset sequences contain a maximum of three people. The ETH Loewenplatz sequence [110] was used to test the system in more crowded scenarios. It consists of an autonomous robot, SmartTer [111],

Figure 4.12: Keypoint filtering in Bonn Crowd scene



4.13(a): ORB-SLAM2

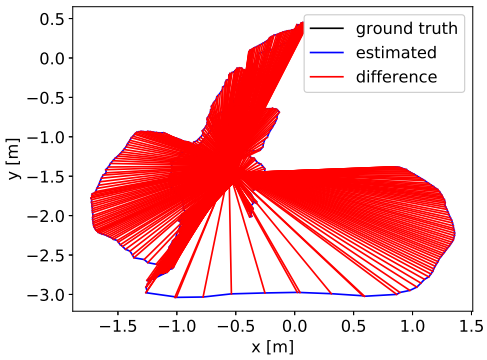
4.13(b): Crowd-SLAM

Figure 4.13: Ground truth and estimated trajectory in the sequence crowd1

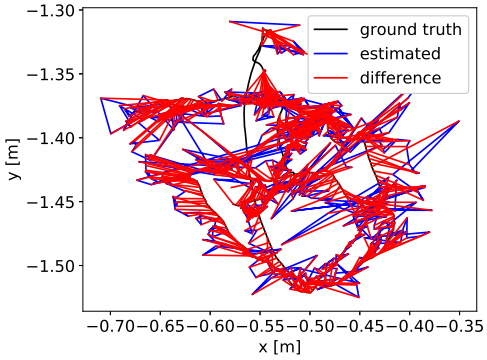
Table 4.8: Comparison of the RMSE of ATE [m] of Crowd-SLAM against ORB-SLAM2, StaticFusion, ReFusion, and DynaSLAM using the Bonn Dataset

Sequence	ORB-SLAM2	StaticFusion	ReFusion	DynaSLAM	Crowd-SLAM
crowd1	0.963	3.586	0.204	**0.016**	0.018
crowd2	1.372	0.215	0.155	0.031	**0.030**
crowd3	1.262	0.168	0.137	0.038	**0.034**
synchronous1	1.121	0.446	0.441	0.015	**0.009**
synchronous2	1.507	0.027	0.022	**0.009**	0.012

traveling through a road with people along the sidewalks and crossing the streets, with a stereo camera pair recording images at 13 FPS. It provides the cameras calibration and odometry. Figure 4.17(a) shows an image of the sequence, together with the people detection in Fig. 4.17(b), and the

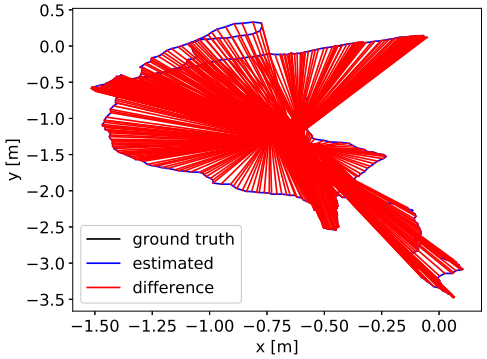


4.14(a): ORB-SLAM2

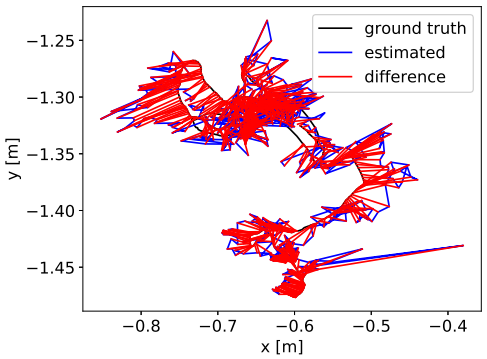


4.14(b): Crowd-SLAM

Figure 4.14: Ground truth and estimated trajectory in the sequence crowd2



4.15(a): ORB-SLAM2

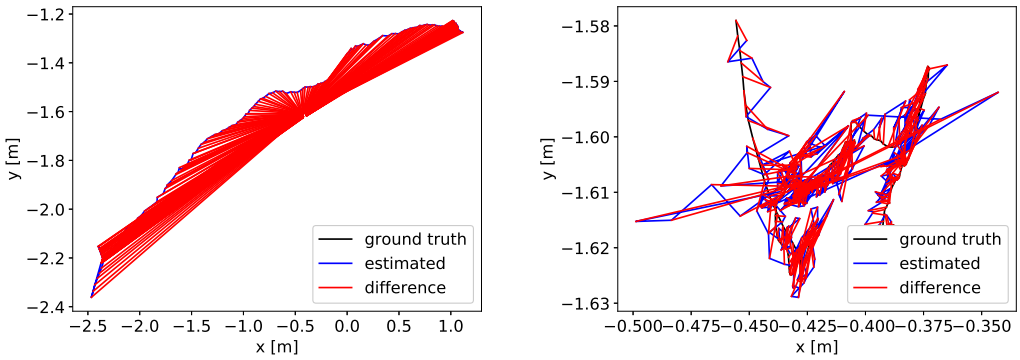


4.15(b): Crowd-SLAM

Figure 4.15: Ground truth and estimated trajectory in the sequence crowd3

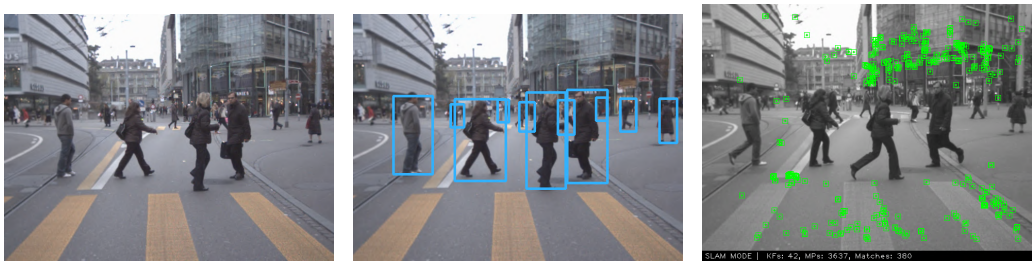
corresponding Dynamic Feature Removal in Fig. 4.17(c).

The provided odometry was made with visual feature tracking using the static background and offline bundle adjustment. An ATE comparison was made between Crowd-SLAM and the provided odometry, and between ORB-SLAM2 and the provided odometry. Crowd-SLAM achieved an RMSE of 33.90 m, and ORB-SLAM2 achieved an RMSE of 41.35 m. As there were no major loop closures in the trajectory, this improvement is due to the tracking system of Crowd-SLAM. Figure 4.18 shows the trajectories estimated by Crowd-SLAM and ORB-SLAM2 compared to the provided odometry. With an absence of loop closures, an eventual gradual deviation was expected. However, the trajectory of ORB-SLAM2 immediately deviates from the odometry due to the presence of people.



4.16(a): ORB-SLAM2 4.16(b): Crowd-SLAM

Figure 4.16: Ground truth and estimated trajectory in the sequence synchronous2



4.17(a): RGB Left Image 4.17(b): Object Detection 4.17(c): Outlier Removal

Figure 4.17: Ground truth and estimated trajectory in the sequence synchronous2

4.3.4 Feature Number Update Improvements

The objective of the feature number update (FNU) step is to prevent lost track due to keypoint filtering. Table 4.9 shows the percentage of successfully tracked frames in six sequences, with and without the update, using the standard ORB-SLAM2 feature number.

Using the standard number of features without update, the system is not able to recover from lost track in the fr3_s_half sequence. Using the proposed approach, the system is able to track more frames. In other sequences, the FNU also provided a considerable improvement. The low percentage of tracked frames in the $fr3\_s\_half$ sequence is caused by two main reasons: (i) this sequence has several frames with the camera close to one person, causing depletion of features; and (ii) it has several large roll camera movements. However, the system was able to correctly relocalize in all situations.

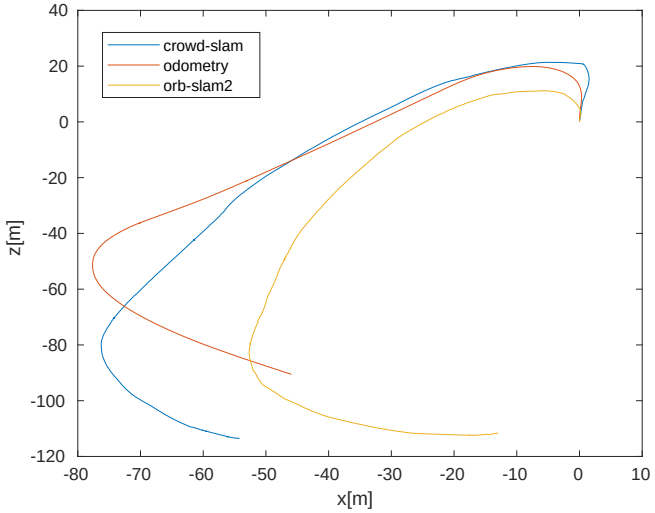


Figure 4.18: Trajectory results from Crowd-SLAM and ORB-SLAM2 compared with the provided odometry for the Loewenplatz sequence

Table 4.9: Percentage of successfully tracked frames

Sequence	Without FNU	With FNU
fr3_s_static	73.09	99.41
fr3_s_xyz	73.83	90.97
fr3_w_static	71.27	95.67
fr3_w_xyz	83.31	99.88
crowd2	82.46	85.47
fr3_s_half	0.0	23.09

4.3.5 Implementation and Run-time Analysis

All tests were performed on a notebook with an Intel Core i7 6700 HQ 2.60 GHz and 16 GB of RAM running Ubuntu Linux 18.04 LTS. The system is implemented in C++, and the object detection is performed with OpenCV, using only CPU. Table 4.10 shows the mean frame rate in FPS of ORB-SLAM2 and Crowd-SLAM for every sequence of Bonn and TUM datasets used for evaluation. Crowd-SLAM achieved an average frame rate of 26.22 FPS, while ORB-SLAM2 achieved 21.50 FPS. For comparison, Detect-SLAM, which also uses object detection, spends 0.34 seconds per frame just for moving-object removal. Dynamic-SLAM achieved a mean performance of 22.2 FPS on $fr3\_w\_xyz$ with GPU. Also, the achieved frame rate of Crowd-SLAM is higher than those of other systems that use GPU, for instance, DynaSLAM (1.35 FPS on $fr3\_w\_halfsphere$), and DS-SLAM (13.08 FPS).

Table 4.10: Mean tracking time [FPS] of ORB-SLAM2 and Crowd-SLAM in the TUM and Bonn sequences

Sequence	ORB-SLAM2	Crowd-SLAM
fr3_s_static	24.260	28.765
fr3_s_xyz	27.894	25.953
fr3_s_rpy	24.343	30.049
fr3_s_half	21.886	30.206
fr3_w_static	17.361	25.893
fr3_w_xyz	18.416	23.960
fr3_w_rpy	21.218	28.432
fr3_w_half	18.487	28.211
crowd1	18.702	28.216
crowd2	18.318	26.492
crowd3	17.721	28.293
synchronous1	25.189	35.534
synchronous2	25.773	29.330
Average	21.505	**26.223**

4.3.6 Limitations

Our approach suffers from two main drawbacks. Although the FNU was able to prevent lost track in many scenarios, there is a limitation. Figure 4.19 shows a frame of the $fr3\_s\_half$ sequence where the bounding box of a person occupies a large portion of the scene, due to the pose of the person and the proximity of the camera, which may cause the depletion of features, even with the FNU system. To overcome this issue, it would be necessary to use the static features detected inside the bounding box. A possible approach to this problem is to use epipolar geometry, matching the current frame with past frames, and selecting features inside the bounding box using a geometric constraint.

Moreover, our approach does not filter other moving objects besides people. However, in the considered crowded environment scenarios, the main source of movement indeed came from people.

Figure 4.19: Bounding box occupying a large portion of the image

4.3.7 Applications

One example of application of the proposed method in a real-world scenario is a people-following system [112] that used CYTi for object detection and Crowd-SLAM to generate the trajectory of the robot. Figure 4.20 shows a person detected using CYTi, which is later tracked and followed. Figure 4.21 shows the final camera trajectory generated by Crowd-SLAM for the evaluation of the people-following system.

4.4 Conclusion

This chapter presented Crowd-SLAM, a new open-source Visual SLAM system designed to perform in crowded human environments. The system is based on ORB-SLAM2, with four main threads: tracking, object detection, local mapping, and loop closing. An efficient dynamic keypoint filtering algorithm was proposed, together with a newly trained network for object detection, and a feature number update system.

The effectiveness of Crowd-SLAM was evaluated on challenging dynamic sequences of the TUM, Bonn, and Loewenplatz datasets. The results indicate that the proposed methodology was successful, with a lower computational time and a better accuracy compared to state-of-the-art methods. To our knowledge, the proposed system has the best results in the crowd2, crowd3, and synchronous1 sequences of the Bonn dataset.

Figure 4.20: People detection with CYTi used in a people-following task

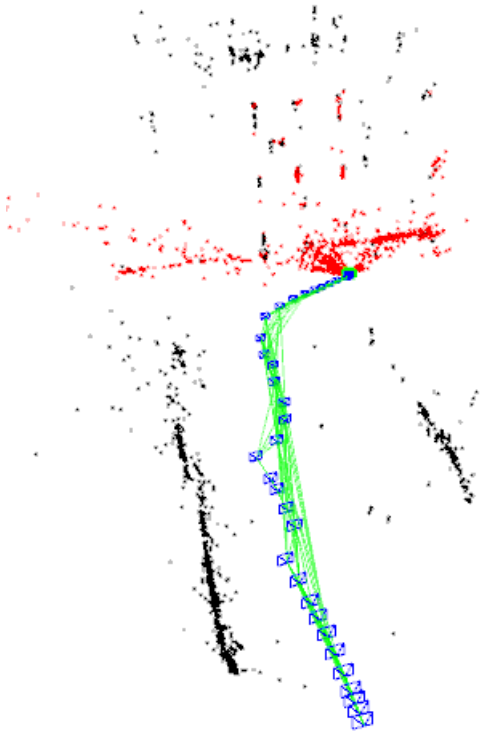


Figure 4.21: Camera trajectory generated by Crowd-SLAM during a people-following task

There are several open problems to explore for future works. For instance, works to allow filtering other moving objects without jeopardizing the performance, and to use the static features inside the bounding boxes in order to

prevent more lost track cases. Other promising works include the integration of tracking and loop closing modules. In the next chapter, some limitations of the proposed Crowd-SLAM approach are discussed and addressed.

5
Visual SLAM in Dynamic Environments

5.1
Introduction

Chapter 4 presented an accurate and efficient approach for visual SLAM in crowded environments. Its main limitations included feature depletion in regions with many objects, or in scenes with an object occupying a large portion of the image, and the inability of detecting and filtering other classes of dynamic objects. The aim of this chapter is to propose a methodology to overcome these limitations while maintaining the real-time performance of Crowd-SLAM.

Thus, this chapter presents a new method for Visual SLAM in Dynamic Environments that uses semantic information over time to predict the poses of dynamic objects appearing in front of the camera, and then filter their features. The proposed method is tested with datasets and compared with other methods from the literature, achieving high accuracy in real time.

Section 5.2 details the proposed methodology, Section 5.3 presents the results and comparisons with the state of the art. Finally, Section 5.4 shows the conclusions and suggestions for future works.

5.2
Methodology

Figure 5.1 shows the framework of the proposed methodology. The system is built on ORB-SLAM3, and is composed of four threads running in parallel: Object Detection, Tracking, Local Mapping, and Loop Closing. ORB features are extracted from the RGB image in the tracking thread, and each associated keypoint is initially classified as dynamic, movable or static, according to the semantic information provided by the object detection thread. A feature repopulation algorithm is proposed to differentiate object features from background ones, and features from people are filtered *a priori*.

The local mapping thread adds new keyframes and points to the map. The MapPoints, created from the detected and classified features, generate MapObjects with a semantic class and an unique ID. An extended Kalman

filter is used to track the objects and predict their state, based on their ID. If an object has a velocity above a threshold, its keypoints are filtered from the image.

The loop closing thread and graph-optimization remain the same of ORB-SLAM3. The output is the pose of the camera frame by frame, processed with filtered sensor information, and the sparse map clear of outliers.

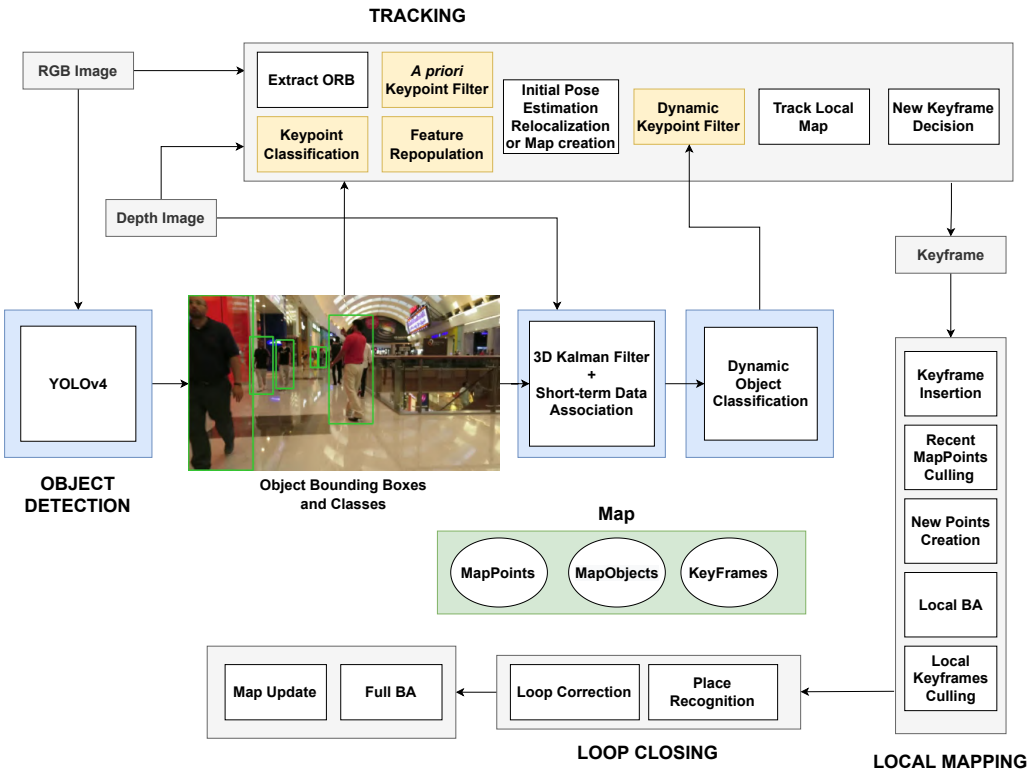


Figure 5.1: Framework of the proposed methodology

5.2.1 Semantic Detection

Crowd-SLAM [29] presented CYTi as an alternative for YOLOv3 that could achieve the same inference time of YOLO-Tiny, but with a similar precision to YOLOv3. CYTi works really well in a crowded environment. However, in an indoor environment there can be objects moving alongside with people that could be relevant for estimation. Thus, this methodology needs a framework that can handle multiple object classes with a low inference time.

Recently, YOLOv4 [11] was developed by Bochkovskiy *et al.* as an evolution of YOLOv3. Figure 5.2 from [11] shows the comparison of YOLOv4 and other object detection frameworks, such as YOLOv3, in terms of average precision (AP) and speed in FPS. For its high precision and inference time, YOLOv4 was chosen for object detection in the proposed methodology.

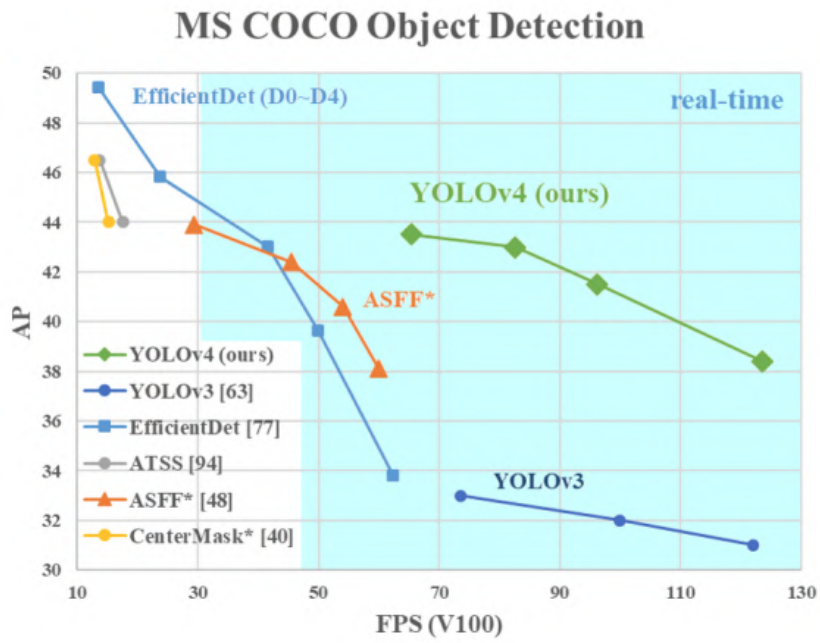


Figure 5.2: Comparison between YOLOv4 and other object detection frameworks in terms of average precision and speed, from Bochkovskiy *et al.* [11].

One important aspect to be considered about the semantic detection thread is that the proposed methodology needs the class, size and position of each object in the scene, and to differentiate what the background is and what is not. As explained in Chapter 4, one of the disadvantages of Crowd-SLAM is the loss of information due to unnecessary filtering of the keypoints in the background inside a bounding box, as the box does not give information of which keypoints actually belong to the object itself and which ones belong to the background.

Therefore, ideally, the proposed methodology would use panoptic segmentation. However, this process is still very computationally expensive to be performed in real time. Section 5.2.2 explains the proposed approach to overcome this problem when using object detection in this framework, to avoid the need for instance or panoptic segmentation. It is a reliable and fast method that achieves a high accuracy in real time.

Thus, despite this methodology being proposed and explained with an object detection framework, it can be easily adapted for other computer vision tasks when they become computationally feasible.

5.2.2 Keypoint Classification

The keypoints detected in the tracking thread are initially classified into three categories: dynamic, movable or static. All keypoints belonging to people

are classified as *a priori* dynamic, keypoints that belong to objects are classified as movable, and the keypoints from the background are static.

Different from SaD-SLAM [32], two keypoints that belong to the same object cannot have different classifications. This improves the speed of the process, because evaluating the dynamics of each individual keypoint is unfeasible in real time. Figure 5.3 shows an example of the initial keypoint classification being performed in a frame with two people and one chair.

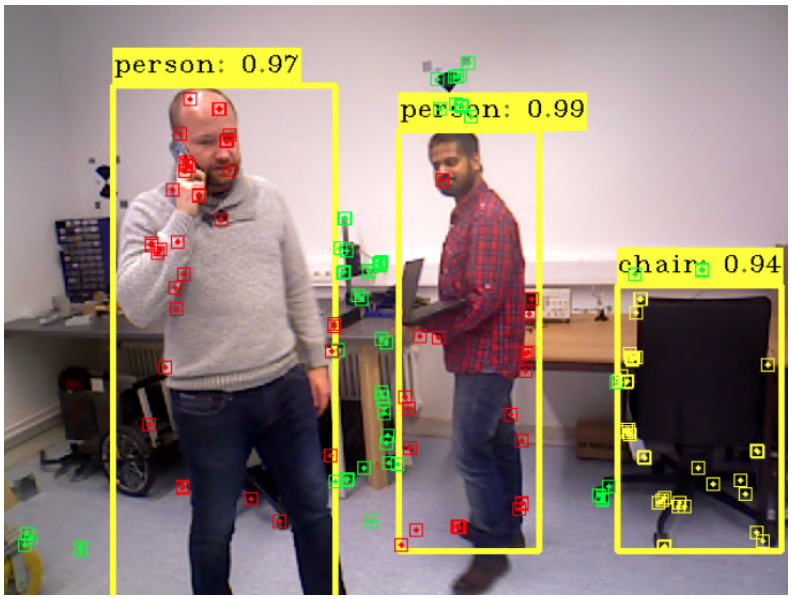


Figure 5.3: Initial Keypoint Classification. Dynamic keypoints are red, movable keypoints are yellow, and static keypoints are green.

Using object detection for feature removal can lead to a depletion of features, especially when there are many dynamic objects in the scene, or when a dynamic object occupies a large portion of the image. Instead of using computationally expensive methods based on epipolar geometry and RANSAC, this thesis presents an efficient method to correctly classify the features that belong to the bounding box but are not dynamic, called feature repopulation.

In each bounding box the median, mean, maximum and minimum pixel depth values are extracted, as well as the standard deviation. With this information, together with the IoU matrix, it is possible to evaluate whether the detected object is being occluded, and to differentiate the object from the background.

To correctly classify the keypoints, it is necessary to consider a few possible conditions. If the object is not being occluded, then the classification is straightforward. Keypoints with a depth greater than the minimum depth plus a threshold value are considered belonging to the background. The threshold

depends on the class of the object. If the object is being partially occluded by another known detected object, the depths inside the overlapping area are not considered. This occlusion is evaluated calculating the IoU between the bounding boxes of the frame. The main problem arises when a non-labeled or non-detectable object is occluding the target object. This problem is identified when the standard deviation of the depths is too high, or if the median depth is higher than the depth of the center of the bounding box. If this happens, it means that the center of the bounding box is being occluded. In this last scenario, the keypoints of this object are not considered. Figures 5.4(a) and 5.4(b) show the result of the feature repopulation technique. In Fig. 5.4(a), all keypoints inside the bounding box are filtered out. However, in Fig. 5.4(b), just the keypoints that belong to people are filtered out, while the keypoints from the background are kept.

5.4(a): Without feature repopulation

5.4(b): With feature repopulation

Figure 5.4: *A priori* people keypoint filtering proposed in this methodology. The figure shows the effect of the feature repopulation technique that keeps keypoints that belong to the background but are located inside the bounding box.

Algorithm 2 details the process of keypoint classification, feature repopulation, and keypoint filtering. The number of the class associated with each keypoint is the same number of that class in the COCO dataset [104]. For example, "person" is 0, "tvmonitor" is 62, and "chair" is 56. When a keypoint has its class set as -1, it means that the keypoint belongs to the background.

Algorithm 2 shows the process for people, but the same procedure is done with movable objects. However, instead of being filtered *a priori*, keypoints that belong to movable objects receive a class number corresponding to the object class.

5.2.3 MapPoints and MapObjects

MapPoints are one of the key elements in ORB-SLAM3 [3]. They are created from the ORB features detected in each frame, and used for all tasks, including camera tracking, local optimization, loop closure. Finally, they constitute the map representation.

A MapObject is an element created to represent a group of MapPoints that belong to same entity, with a common body and class. In the proposed methodology, besides all other pieces of information needed for the ORB-SLAM3 framework, MapPoints store a 3D position $X_{w,i}$ in the world coordinates, the class, and the ID of the MapObject associated with the MapPoint. When the MapPoint is created, it receives the class and ID of the MapObject associated with the bounding box where the keypoint was located. Each MapObject stores:

- First detected bounding box
- Current detected bounding box
- Class
- List of associated MapPoints
- 3D position in world coordinates
- Unique global ID

The 3D position of the MapObject is obtained by computing the centroid of the associated MapPoints.

5.2.4 Short-term Data Association

The short-term data association evaluates if the new bounding boxes detected in the current frame correspond to the bounding boxes of the MapObjects present in the last frame. This is done using the intersection over union (IoU).

In each new frame, for every bounding box the IoU is computed with all detections of the same class in the previous frame. If no matches are found, a new MapObject instance is created. This process is detailed in Algorithm

Algorithm 2: Keypoint classification and filtering

Data: Current frame F_k, Last frame F_{k-1}, bounding box list D^{F_k}(x,y,w,h, class), keypoint list p^{F_k}, DOC

```
1  firstloop = true;
2  nbox = 0;
3  new_keypoints;
4  if D^{F_k} size > 0 then
5      for p_i in p^{F_k} do
6          bool_key = false;
7          while nbox < D^{F_k} size do
8              if is the first loop then
9                  Get bounding box depth statistics
10             end
11             for bb in D^{F_k} do
12                 Check occlusion between D^{F_k}[nbox] and D^{F_k}[bb]
13             end
14             if D^{F_k}(class) is person then
15                 if p_i is inside D^{F_k}[nbox] then
16                     bool_key = true;
17                     if no occlusion by a labeled object then
18                         if no occlusion by unlabeled object then
19                             if p_i depth > mindepth + depth threshold
                                then
20                                 bool_key = false;
21                                 keypoint class = -1;
22                             end
23                             else
24                                 keypoint class = D^{F_k}(class);
25                             end
26                         end
27                         else
28                             bool_key = false;
29                             keypoint class = -1;
30                         end
31                     end
32                     else
33                         disregard the occluded part and perform the
                            steps of lines 18 to 30;
34                     end
35                 end
36             end
37             nbox++;
38         end
39         firstloop = false;
40         if bool_key is false then
41             new_keypoints receive p_i
42         end
43     end
44     p^{F_k} = new_keypoints;
45 end
```

3. When a new KeyFrame is created, the system checks whether a tracked MapObject has new associated MapPoints. If it has, then its pose is updated.

Algorithm 3: Short-term Data Association

```
Data: Frame F_k, bounding box list D^{F_k}(x,y,w,h, class), list of last
      frame objects lastFrameObjs
for det in D^{F_k} do
    associationfound = false;
    for lfo in lastFrameObjs do
        if lastFrameObjs not NULL then
            iou = GetIOU(det, bounding box of lastFrameObjs(lfo));
            IOU_Matrix(det)(lfo) = iou;
            if current_class == class of lastFrameObjs(lfo) then
                if IOU_Matrix(det)(lfo) > IOU_Threshold then
                    associationfound = true;
                    lastObjID = lfo;
                    break;
                end
            end
        end
    end
    if associationfound then
        currentMapObjects receive lastFrameObjs(lastObjID);
    end
    else
        Create new MapObject;
        currentMapObjects receive new MapObject;
    end
end
```


5.2.5 Object Tracking

The position of a MapObject is given by the 3D position of its centroid. The position of the centroid is obtained through the position of each associated MapPoint. With the short-term data association algorithm, presented in Section 5.2.4, each new detection is matched with all MapObjects from the last frame, which allows the possibility to track objects over time. In order to filter dynamic keypoints, it is necessary to infer if the tracked objects are moving or not.

The Extended Kalman Filter (EKF) is a non-linear state estimator that considers motion and observations corrupted by a zero mean Gaussian noise, as shown by Eqs. (5-1) and (5-2), where **f** is a non-linear state transition function, **h** is a non-linear measurement function, and ϵ and δ are Gaussian noises, as

stated in Eqs. (5-3) and (5-4), with zero mean and covariance matrices $\mathbf{Q}$ and $\mathbf{R}$, respectively.

$$\mathbf{x}_{k|k-1} = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_k) + \epsilon_k \tag{5-1}$$

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k) + \delta_k \tag{5-2}$$

$$\epsilon \sim N(0, Q) \tag{5-3}$$

$$\delta \sim N(0, R) \tag{5-4}$$

The objective of the EKF is to obtain the best estimate of $\mathbf{x}$ given the measurements $\mathbf{z}$. An EKF is used to track each MapObject, in order to predict which one is moving. An EKF is initialized for each new MapObject. The state of the object is defined as its 3D position and velocity, as stated by

$$\mathbf{x} = [x \ y \ z \ \dot{x} \ \dot{y} \ \dot{z}]^T \tag{5-5}$$

The prediction step at frame k is given by

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}\hat{\mathbf{x}}_{k-1|k-1} \tag{5-6}$$

where k is the current frame, $k-1$ is the last frame, and $\mathbf{F}$ is given by

$$\mathbf{F} = \begin{bmatrix} \mathbf{I}_3 & \Delta t \mathbf{I}_3 \\ \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix} \tag{5-7}$$

where Δt is the time between predictions, $\mathbf{I_3}$ is a 3x3 identity matrix, and $\mathbf{0_3}$ is a 3x3 matrix with zeros. The state covariance is estimated as stated by

$$\mathbf{P}_{k|k-1} = \mathbf{F}\mathbf{P}_{k-1|k-1}\mathbf{F}^T + \mathbf{Q} \tag{5-8}$$

where $\mathbf{P}_0 = \mathbf{I}$. The measurement update is given by Eq. (5-9), and the innovation covariance by Eq. (5-10).

$$\tilde{\mathbf{y}}_k = \mathbf{z}_k - \hat{\mathbf{h}}(\hat{\mathbf{x}}_{k|k-1}) \tag{5-9}$$

$$\mathbf{S}_k = \mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^T + \mathbf{R} \tag{5-10}$$

where $\mathbf{z}_k$ correspond to the measured coordinates of the centroid. The $\hat{\mathbf{h}}$ function transforms the predicted state from the world frame to the camera frame. Finally, the Kalman gain is stated by

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}_k^T\mathbf{S}_k^{-1} \tag{5-11}$$

The updated state estimate and covariance are given by Eqs. (5-12) and (5-13).

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\tilde{\mathbf{y}}_k \tag{5-12}$$

$$\mathbf{P}_{k|k} = (\mathbf{I}_6 - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k|k-1} \tag{5-13}$$

5.2.6 Dynamic Object Classification

The output of the object tracking is the state of each tracked MapObject. The dynamic object classification module outputs a Boolean result, establishing if the object is moving or not in that particular frame, based on a threshold, called "DOC threshold".

Keypoints belonging to new objects are filtered *a priori*. If the object tracker establishes that the object is static, the keypoints are used for feature tracking. Keypoints belonging to moving objects are classified with the same procedure as the one described in Algorithm 2.

5.3 Results

5.3.1 TUM Dataset

Five dynamic sequences from the TUM Dataset were used to evaluate the robustness of the proposed method: the four *walking* sequences and the *fr3_sitting_xyz* sequence, to be used as a baseline. The system was compared to several others discussed in Chapter 1. The comparison includes feature-based methods designed for static environments, such as ORB-SLAM3 [3], direct methods designed for dynamic environments, such as StaticFusion [35] and ReFusion [36], and feature-based methods designed for dynamic environments, for instance DynaSLAM [30], DS-SLAM [31], SaD-SLAM [32], and DOT-Mask [52].

The results from Liu *et al.*, DynaSLAM, DS-SLAM, SOF-SLAM, Detect-SLAM, SaD-SLAM, DOT-Mask, Ji *et al.* were obtained in their respective publications. The results from ORB-SLAM2 and ORB-SLAM3 were obtained running their codes using 1500 keypoints in each frame. Finally, the results from StaticFusion and ReFusion were obtained in [36]. For the results from the proposed work, the tests were performed 5 times and the median results were used for the evaluation, as proposed by Mur-Artal and Tardós [24].

Table 5.1 show the parameters used in the simulations. Table 5.2 shows the RMSE of the ATE comparison between the proposed method and 13 methods. The bold values represent the best results in each sequence, and the underlined values represent the second best ones. The proposed method

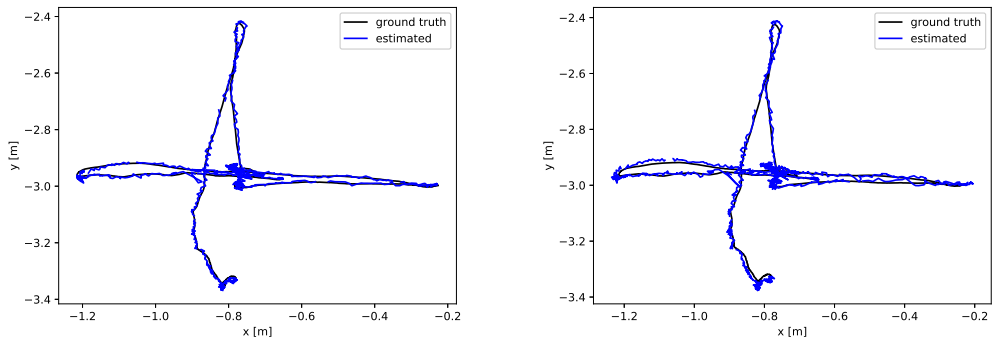
achieved a high accuracy, with the best result in the challenging sequence $fr3\_w\_xyz$.

Table 5.1: Parameters used in the simulations

Description	Value
Number of features	2000
IoU threshold	0.15
Depth threshold [m]	0.1
DOC threshold [m/s]	0.1

Tables 5.3 and 5.4 show the RMSE of translational and rotational drifts (RPE), respectively, of the proposed method against ORB-SLAM3, DS-SLAM, Liu et al., Sun et al., and Ji et al. The papers of DynaSLAM, ReFusion, StaticFusion, SaD-SLAM, DOT-Mask, Detect-SLAM, SOF-SLAM do not contain RPE results. Thus, they were left from Tables. The proposed method achieved the best result in three sequences.

Figures 5.5(a) through 5.9(b) show, respectively, the comparison between the ground-truth and estimated trajectories of ORB-SLAM3 and the proposed method in the TUM sequences. The trajectory estimated by ORB-SLAM3 completely deviates from the ground-truth in the *walking* sequences, as expected. The proposed work, on the other hand, is able to match the ground-truth.



5.5(a): ORB-SLAM3 5.5(b): Ours

Figure 5.5: Ground-truth and estimated trajectory in the sequence fr3_s_xyz

Table 5.2: Comparison of the RMSE of ATE [m] of the proposed method against ORB-SLAM3, StaticFusion, ReFusion, DynaSLAM, DS-SLAM, SOF-SLAM, Detect-SLAM, Liu *et al.*, Sun *et al.*, Sun *et al.*, SaD-SLAM, DOT-Mask, and Ji *et al.* using the TUM dataset

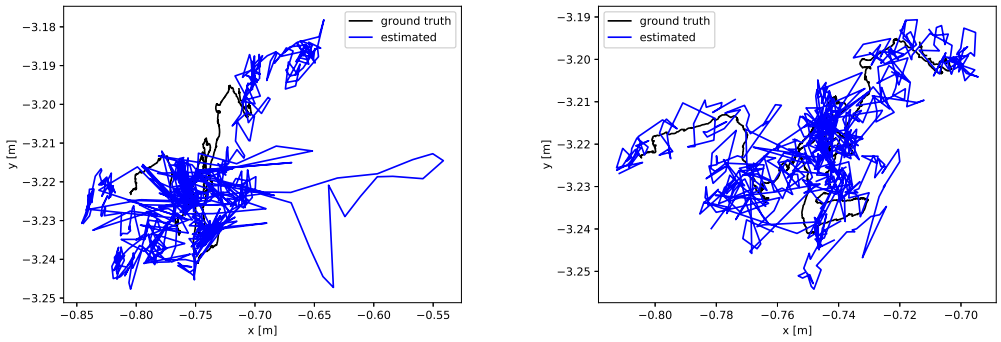
Sequence	*fr3_s_xyz*	*fr3_w_static*	*fr3_w_xyz*	*fr3_w_rpy*	*fr3_w_half*
Ours	0.013	0.009	**0.015**	<u>0.031</u>	0.028
ORB-SLAM3	**0.009**	0.038	0.819	0.957	0.315
StaticFusion	0.039	0.015	0.093	—	0.681
ReFusion	0.040	0.017	0.099	—	0.104
DynaSLAM	0.015	**0.006**	**0.015**	0.035	**0.025**
DS-SLAM	—	0.008	0.024	0.444	0.030
SOF-SLAM	—	<u>0.007</u>	0.018	**0.027**	0.029
Detect-SLAM	0.020	—	0.024	0.296	0.051
Liu *et al.* [48]	—	0.011	<u>0.016</u>	0.042	0.031
Sun *et al.* [39]	0.048	0.065	0.093	0.133	0.125
Sun *et al.* [40]	0.051	0.033	0.066	0.073	0.067
SaD-SLAM	<u>0.012</u>	0.017	0.017	0.032	<u>0.026</u>
DOT-Mask	0.018	0.008	0.021	0.053	0.040
Ji *et al.* [53]	<u>0.012</u>	0.011	0.020	0.037	0.029

5.3.2 Run-time Analysis

All tests were performed on a notebook with an Intel Core i7-10750H and 16 GB of RAM running Ubuntu Linux 18.04 LTS. The system is implemented in C++, and the object detection is performed with OpenCV 4.5, using an Nvidia GeForce RTX 2060 GPU. Table 5.5 shows the mean frame rate in ms of the proposed method for every sequence of the TUM datasets used in this evaluation. The proposed system achieved an average speed of 27.8 FPS, which is higher than all compared methods. For instance, SaD-SLAM [32] is offline, and DynaSLAM [30] runs at less than 1 FPS. DOT-Mask [52], even using GPU, only achieves 8 FPS.

Table 5.3: Comparison of the RMSE of translational RPE [m/s] of the proposed method against ORB-SLAM3, DS-SLAM, Liu *et al.*, Sun *et al.*, Sun *et al.*, and Ji *et al.* using the TUM dataset

Sequence	*fr3_s_xyz*	*fr3_w_static*	*fr3_w_xyz*	*fr3_w_rpy*	*fr3_w_half*
Ours	0.019	0.013	<u>0.022</u>	**0.044**	0.039
ORB-SLAM3	**0.013**	0.055	1.203	1.414	0.467
DS-SLAM	—	**0.010**	0.033	0.150	**0.030**
Liu *et al.* [48]	—	0.015	**0.021**	0.064	<u>0.033</u>
Sun *et al.* [39]	0.033	0.084	0.121	0.175	0.167
Sun *et al.* [40]	0.036	0.031	0.067	0.097	0.061
Ji *et al.* [53]	<u>0.017</u>	<u>0.012</u>	0.023	<u>0.047</u>	0.042



5.6(a): ORB-SLAM3 5.6(b): Ours

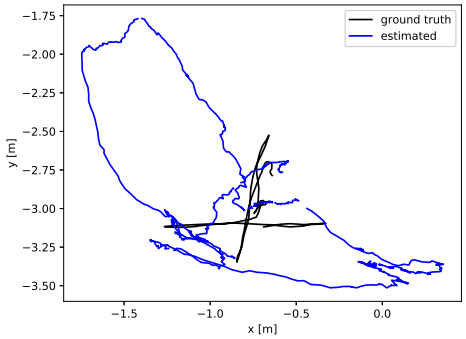
Figure 5.6: Ground truth and estimated trajectory in the sequence fr3_w_static

5.4 Discussions and Conclusions

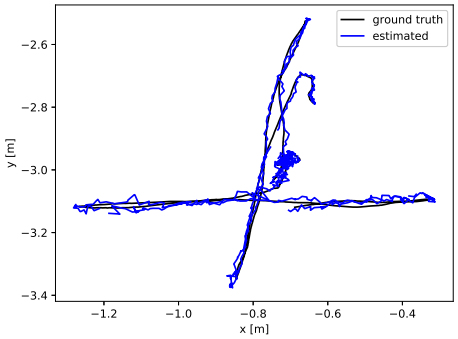
This chapter presented a new methodology for visual SLAM in highly dynamic environments. The system is based on ORB-SLAM3, with four main threads: tracking, object detection, local mapping, and loop closing. The proposed method expands the one presented in Chapter 4 to include the ability to perceive and filter the movement of objects, besides people. The effectiveness of this method was evaluated on challenging dynamic sequences of the TUM dataset, achieving a high accuracy in real time, running in 27.8 FPS in average. Also, the proposed method was compared with 13 methods from the literature, some of them offline, and achieved similar results to the ones with the best accuracy but significantly faster.

Table 5.4: Comparison of the RMSE of rotational RPE [deg/s] of the proposed method against ORB-SLAM3, DS-SLAM, Liu *et al.*, Sun *et al.*, Sun *et al.*, and Ji *et al.* using the TUM dataset

Sequence	*fr3_s_xyz*	*fr3_w_static*	*fr3_w_xyz*	*fr3_w_rpy*	*fr3_w_half*
Ours	0.599	0.318	**0.621**	**0.936**	0.901
ORB-SLAM3	**0.577**	1.055	23.179	27.449	10.078
DS-SLAM	—	0.269	0.826	3.004	<u>0.814</u>
Liu *et al.* [48]	—	<u>0.311</u>	<u>0.630</u>	1.407	**0.787**
Sun *et al.* [39]	0.983	2.049	3.235	4.375	5.010
Sun *et al.* [40]	1.036	0.900	1.595	2.593	1.900
Ji *et al.* [53]	<u>0.597</u>	**0.287**	0.637	<u>1.059</u>	0.965

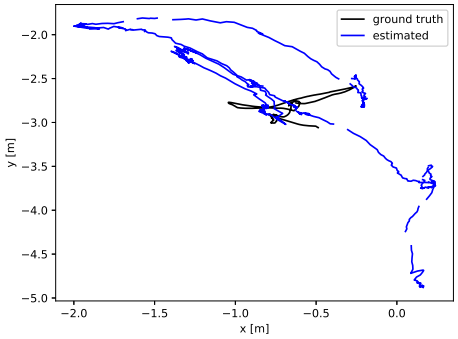


5.7(a): ORB-SLAM3

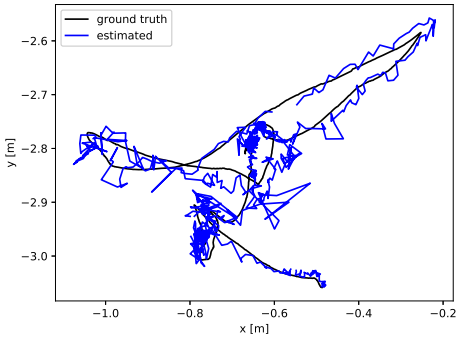


5.7(b): Ours

Figure 5.7: Ground truth and estimated trajectory in the sequence fr3_w_xyz

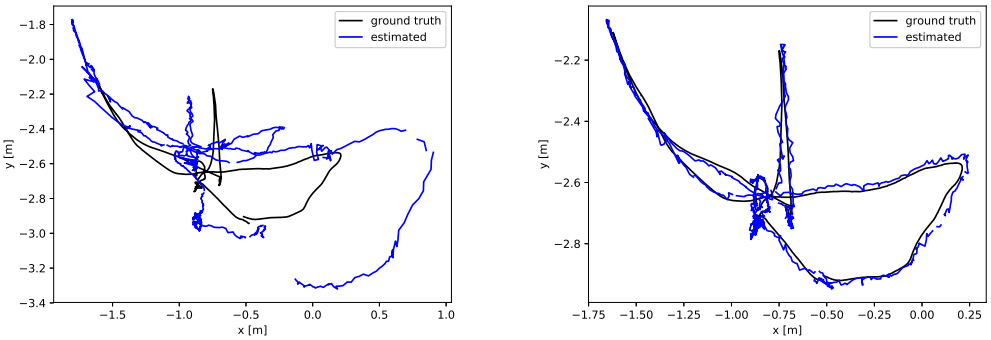


5.8(a): ORB-SLAM3



5.8(b): Ours

Figure 5.8: Ground truth and estimated trajectory in the sequence fr3_w_rpy



5.9(a): ORB-SLAM3 5.9(b): Ours

Figure 5.9: Ground truth and estimated trajectory in the sequence fr3_w_halfsphere

Table 5.5: Mean tracking time [ms] of the proposed method in the TUM sequences

Sequence	Ours
fr3_s_xyz	36.5
fr3_w_static	35.8
fr3_w_xyz	37.4
fr3_w_rpy	36.8
fr3_w_half	35.9
Average	36.48

6
Dataset for Visual SLAM in Changing Environments

6.1
Introduction

Public datasets are fundamental elements of the evolution of SLAM systems. They are key players for the rapid advance of the state-of-the-art in the SLAM community.

However, even SLAM in highly dynamic environments - a problem that has been broadly studied in the past years with an increasing number of methods being proposed - relies on a small number of available datasets [80][36]. Furthermore, there is a clear absence of datasets for visual SLAM systems focused on changing environments.

This chapter presents the PUC/USP Changing Environments dataset, a new dataset for the evaluation of visual SLAM systems in indoor changing environments. To our knowledge, this is the first one focused on the evaluation of this problem.

The dataset has 6 sequences recorded at the Aeronautics Department of the University of São Paulo, São Carlos campus, shown in Figs. 6.1(a) and 6.1(b). Each sequence contains color and depth images captured by an Intel RealSense camera, and a ground-truth trajectory obtained using a motion capture system. The sequences were designed to capture different scenarios that could lead to a tracking failure or a map corruption. All data is publicly available[1].

Section 6.2 shows the experimental setup composed of a mobile robot, a camera for data collection, and a motion capture system for ground-truth generation. Section 6.3 discusses the objects present in the scenes, details each sequence of the dataset, and presents the ground-truth trajectories with the corresponding numerical data. Finally, Section 6.4 presents the conclusions and suggestions for future works.

[1]https://github.com/virgolinosoares/changing-dataset

6.1(a): External vision of the test site

6.1(b): Test site

Figure 6.1: Aeronautics Department of the University of São Paulo, São Carlos campus

6.2 Experimental Setup

The experimental setup is composed of a motion capture system and a mobile robot equipped with an RGB-D camera, which are presented in the following sections.

6.2.1 Motion Capture System

The motion capture system consists of seven OptiTrack Flex13 cameras [113], shown in Figs. 6.2(a) and 6.2(b). All cameras are positioned toward the workspace where the experiments took place and distributed in a way that maximizes the field of view without compromising the calibration quality. The OptiTrack cameras have an image resolution of 1920x1024 pixels and 120 FPS for data acquisition. According to the manufacturer, the root-mean-square localization accuracy of a reflective marker is claimed to be sub-millimeter and 0.3 mm at optimal conditions.

6.2.2 Robot

Figure 6.3 shows the robot used in the experiments. The TerraSentia robot, developed by EarthSense [114], is a mobile robot with a size of 0.32x0.54x0.4 meters and equipped with four active wheels for skid-steer locomotion. All data acquisition, processing, and locomotion control are performed using an Intel i7 NUC in combination with a RaspberryPi 3B board.

6.2(a): Motion capture system

6.2(b): OptiTrack Camera

Figure 6.2: External motion capture system from OptiTrack used to track the pose of the camera in the robot

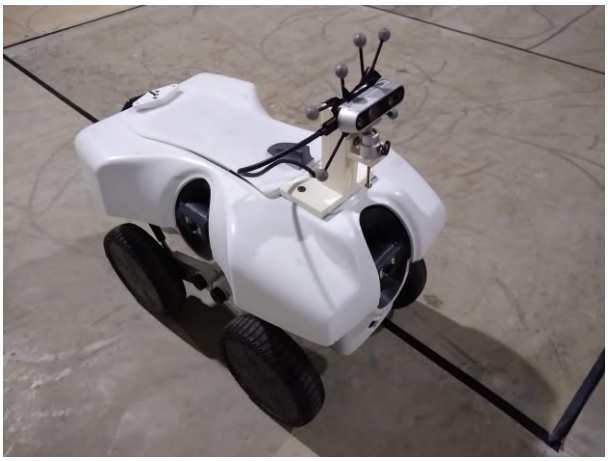


Figure 6.3: TerraSentia Robot

6.2.3 Tracked Camera

The TerraSentia robot was originally designed for autonomously collecting crop data. But modifications were made to the robot in order to install an Intel RealSense D435i camera. The D435i camera uses a stereo depth module and an RGB sensor with color image signal processing to produce a depth frame from the scene. A camera resolution of 848x480 pixels and 30 FPS was configured to collect the images for the dataset. Figure 6.4 shows the camera fixed in the front of the robot, with the reflexive markers used by the OptiTrack system for computing the ground-truth trajectory.

Figure 6.4: Intel RealSense D435i Camera with reflective markers

6.2.4 Data Acquisition and File Formats

Figures 6.5(a) and 6.5(b) show an example of data collected by the Intel RealSense camera at each timestamp. The depth image was normalized to allow a clear visualization. Both images have a time-synchronized pose saved by the OptiTrack system. The Network Time Protocol (NTP) was used to assure timestamp synchronization between the motion capture system and the camera. The rgb and depth images are stored with the same format of the TUM Dataset [80], with one folder "rgb" containing all color images, one folder "depth" containing all depth images, an "rgb.txt" file containing a list with all color image names, following their respective timestamp, a "depth.txt" file containing a list with all depth image names, following their respective timestamp, and a "groundtruth.txt" file. The ground-truth file also has the same format used in the TUM Dataset. Each line in the text file represents a pose with a unique timestamp. The pose is written as a pose-quaternion.

6.5(a): RGB Image

6.5(b): Depth Image

Figure 6.5: Example of RGB and depth images collected by the Intel RealSense camera

6.2.5 Motion Capture System Calibration

The motion capture system was calibrated using the software provided by OptiTrack. The procedure consisted of waving a calibration stick, shown in Fig. 6.6, while the calibration software registered the movement, receiving information from each camera. Figure 6.8 shows the calibration procedure being performed, and Fig. 6.10 shows the calibration software informing that the procedure was successful, with a high quality. The next step is to set a ground reference, using the calibration tool shown in Fig. 6.7. Figure 6.9 shows the Intel RealSense camera being localized after calibration. A global precision of $4mm$ was achieved after measuring the length of a metal rod with two reflective markers, following a process similar to the one suggested by [80].

Figure 6.6: Calibration wand

Figure 6.7: Ground reference

Figure 6.8: Calibration procedure

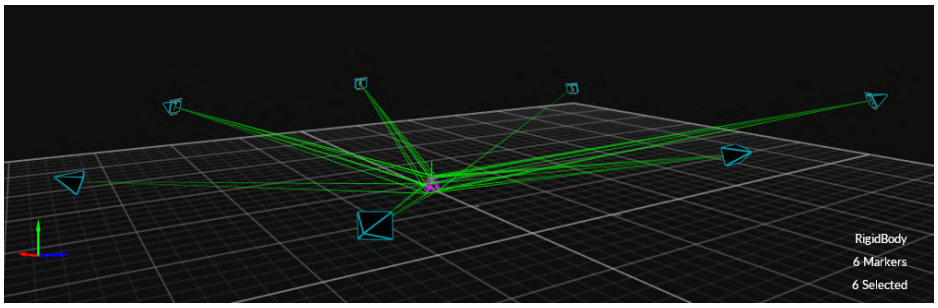


Figure 6.9: OptiTrack Software showing the tracked camera

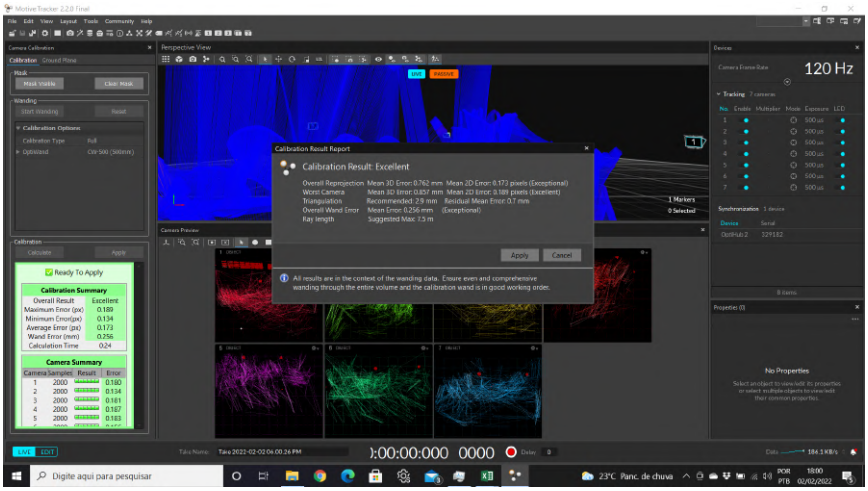


Figure 6.10: OptiTrack Software notifying the result of the calibration

6.3 Dataset

6.3.1 Objects

Several objects from the COCO Dataset [104] were used in the sequences, such as a teddy bear, umbrellas, chairs and monitors, as shown in Figs. 6.11(a) and 6.11(b). These objects were chosen because several semantic detectors use COCO for training, such as YOLOv3 [49], YOLOv4 [11], and Mask R-CNN [9].

6.3.2 Sequences

Six sequences were recorded in this dataset: *Static*, *OneChair*, *Vanishing*, *Changing*, *Shift*, and *Changing2*. All sequences were recorded with the camera fixed on the robot, with the robot moving in the plane shown in Fig. 6.2(a). Despite being fixed and located on a wheeled mobile robot, the motion of the Intel RealSense camera is not entirely restricted to a plane due to irregularities

6.11(a): Teddy bear in a chair

6.11(b): Umbrella

Figure 6.11: Objects used in the sequences

on the floor as well as robot vibration, as shown in Figs. 6.13(a), 6.16(a), 6.19(a), 6.22(a), 6.25(a) and 6.28(a).

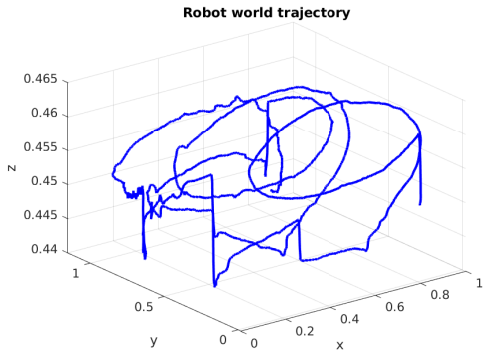
Table 6.1 shows the statistics over the six recorded sequences, containing the duration, total length, average translational velocity and average rotational velocity. The actual velocity of the robot was higher, because the robot eventually needed to stand still for some moments so the objects could be moved in the scene out of its field of vision.

Table 6.1: Information about each sequence

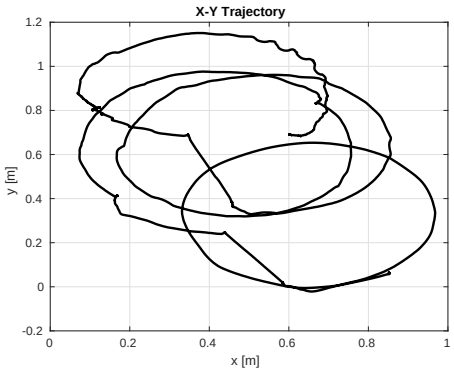
Sequence	Duration [s]	Length [m]	Avg. T. Vel. [m/s]	Avg. R. Vel. [deg/s]
Static	109.73	9.09	0.0828	27.08
OneChair	294.83	18.78	0.0637	14.07
Vanishing	277.53	23.17	0.0833	24.44
Changing	195.46	13.89	0.071	25.36
Shift	366.36	25.19	0.0687	16.26
Changing2	174.53	15.49	0.0887	46.55

The *Static* sequence is a baseline for the evaluations. No objects were moved in this sequence. The robot just wanders within a static scene. Figures 6.12(a) and 6.12(b) show the ground-truth trajectory of the *OneChair* sequence.

The *OneChair* sequence is suitable to evaluate the ability of SLAM systems to avoid wrong loop closures caused by moved objects. It starts with the robot facing a chair, as shown in Fig. 6.14(a). As the robot moves, the chair is also moved outside of the robot field of view. When the robot sees the chair again, as shown in Fig. 6.14(c), the chair is at a different position, which can cause a wrong loop detection. Figures 6.15(a) and 6.15(b) show the ground-truth trajectory of the *OneChair* sequence.

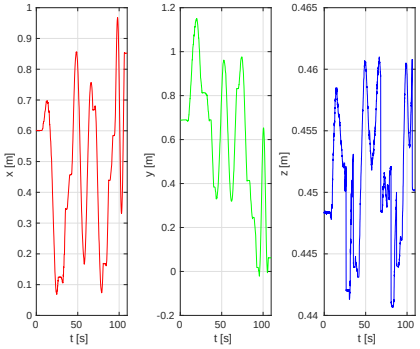


6.12(a): 3D ground-truth trajectory

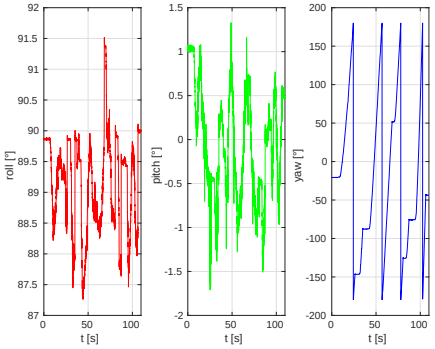


6.12(b): 2D ground-truth trajectory

Figure 6.12: Ground-truth trajectory of the *Static* sequence



6.13(a): X-Y-Z coordinates over time



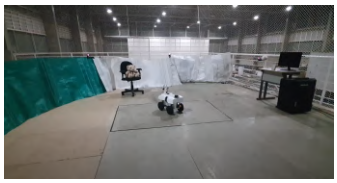
6.13(b): Roll-Pitch-Yaw over time

Figure 6.13: Metrics from the *Static* sequence

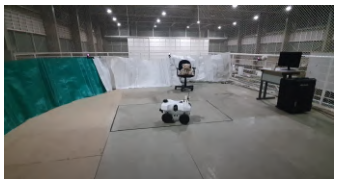
The *Vanishing* sequence is suitable to evaluate the ability of SLAM systems to eliminate vanished objects in the map. It starts with the robot facing a chair with a teddy bear, as shown in Fig. 6.17(a). As the robot wanders, some objects are moved, as shown in Fig. 6.17(c), and others are removed from the scene, as shown in Figs. 6.17(b) and 6.17(d), until there are no more objects in the scene, as shown in Fig. 6.17(f). Even though the missing objects would not cause a track failure or a wrong loop closure in a SLAM system not robust to changing environments, they would be present in the final map. This would interfere on a path planning algorithm that uses this map, for example.

The *Changing* sequence initially contains two umbrellas and two chairs. Also, there is one teddy bear on one of the chairs. It starts with the robot facing the chair with the teddy bear, as shown in Fig. 6.20(a). As the robot wanders within the scene, the teddy bear is moved to the other chair, as shown in Fig. 6.20(b). This can potentially trigger a wrong loop closure.

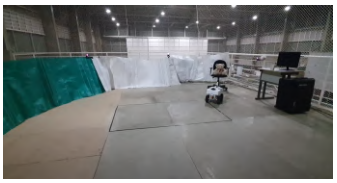
The *Shift* sequence initially contains an umbrella on the floor, a teddy



6.14(a): Scene 1

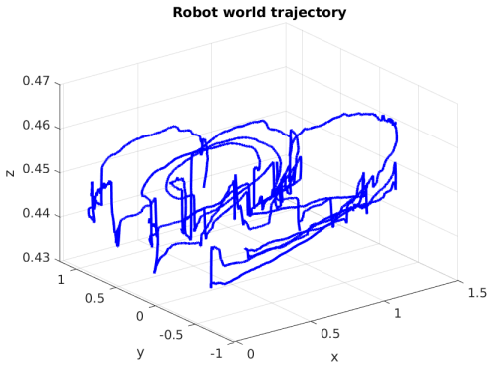


6.14(b): Scene 2

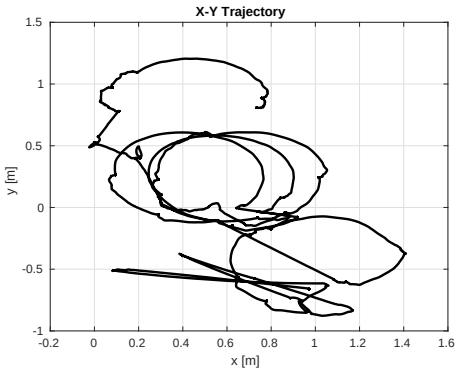


6.14(c): Scene 3

Figure 6.14: Selected scenes from the *OneChair* sequence



6.15(a): 3D ground-truth trajectory

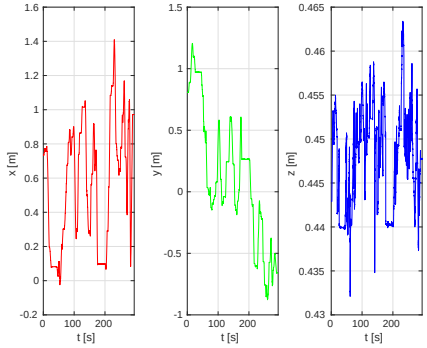


6.15(b): 2D ground-truth trajectory

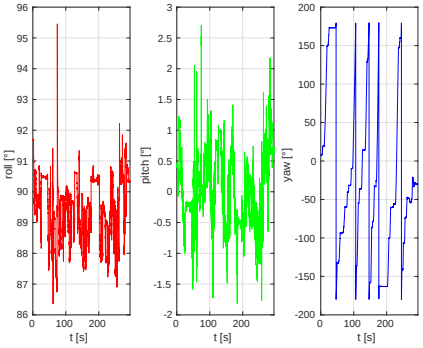
Figure 6.15: Ground-truth trajectory of the *OneChair* sequence

bear on a chair, and a suitcase with a monitor and a mug. The robot starts facing the chair with the teddy bear, as shown in Fig. 6.23(a). As the robot wanders within the scene, the umbrella and chair are moved to the positions shown in Fig. 6.23(b). After several turns, the suitcase together with the monitor and mug are also moved, as shown in Fig. 6.23(e). This sequence is suitable to evaluate the ability of SLAM systems to detect changes of multiple objects. Figures 6.24(a) and 6.24(b) show the ground-truth trajectory of the *Shift* sequence generated by the motion capture system.

The *Changing2* sequence initially contains a chair with books, another chair with a teddy bear on it, and a suitcase with a monitor and a mug. The robot starts facing the chair with the teddy bear, as shown in Fig. 6.26(a). As the robot wanders within the scene, the books are removed from the other chair, as shown in Fig. 6.26(c). After a while, the chair with the teddy bear is replaced by an umbrella, as shown in Fig. 6.26(d). Finally, the other chair is removed, as shown in Fig. 6.26(e). Figures 6.27(a) and 6.27(b) show the ground-truth trajectory of the *Changing2* sequence generated by the motion capture system.



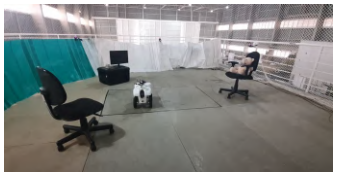
6.16(a): X-Y-Z coordinates over time



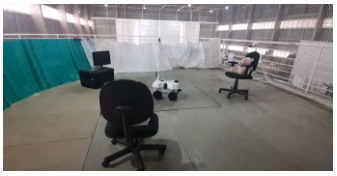
6.16(b): Roll-Pitch-Yaw over time

Figure 6.16: Metrics from the *OneChair* ground-truth sequence

6.17(a): Scene 1



6.17(b): Scene 2



6.17(c): Scene 3

6.17(d): Scene 4

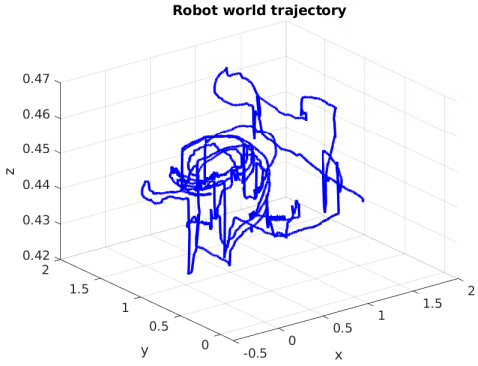
6.17(e): Scene 5

6.17(f): Scene 6

Figure 6.17: Selected scenes from the *Vanishing* sequence

6.4 Conclusions

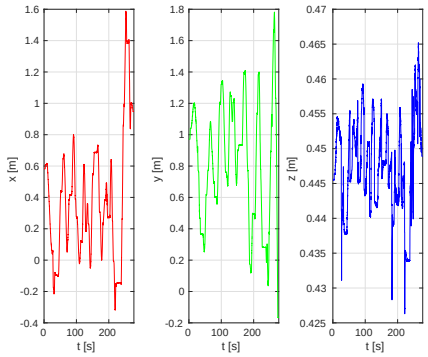
This chapter presented a dataset for the evaluation of visual SLAM systems operating in changing environments. To our knowledge, this is the first dataset focused on this problem. It contains six sequences with color and depth images, and associated ground-truth trajectories created using a reliable motion capture system. The motion capture cameras were carefully calibrated to assure high quality tracking. Future versions of this dataset could include data from other sensors, such as IMU, use different objects, and test other scene configurations.



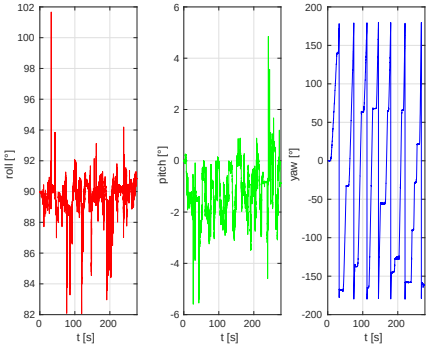
6.18(a): 3D ground-truth trajectory

6.18(b): 2D ground-truth trajectory

Figure 6.18: Ground-truth trajectory of the *Vanishing* sequence



6.19(a): X-Y-Z coordinates over time



6.19(b): Roll-Pitch-Yaw over time

Figure 6.19: Metrics from the *Vanishing* ground-truth sequence

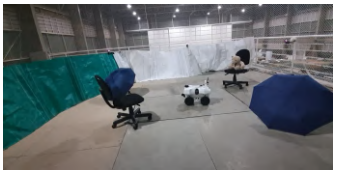
6.20(a): Scene 1

6.20(b): Scene 2

6.20(c): Scene 3

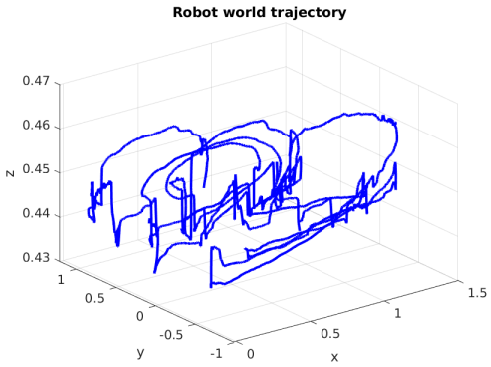
6.20(d): Scene 4

6.20(e): Scene 5



6.20(f): Scene 6

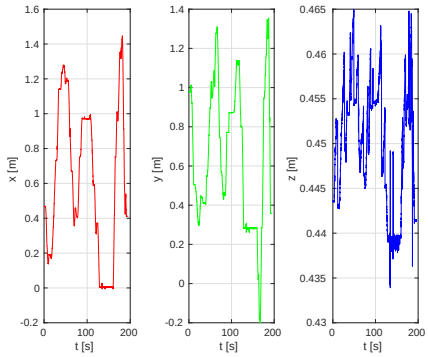
Figure 6.20: Selected scenes from the *Changing* sequence



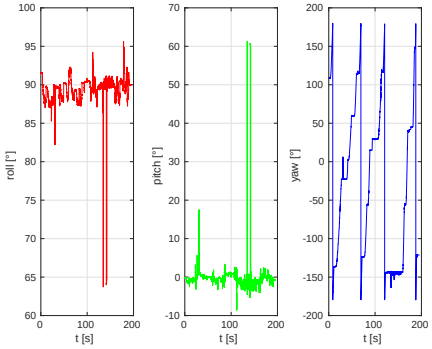
6.21(a): 3D ground-truth trajectory

6.21(b): 2D ground-truth trajectory

Figure 6.21: Ground-truth trajectory of the *Changing* trajectory



6.22(a): X-Y-Z coordinates over time

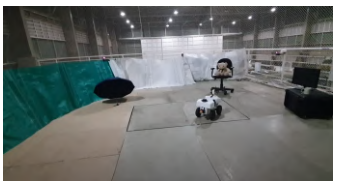


6.22(b): Roll-Pitch-Yaw over time

Figure 6.22: Metrics from the *Changing* sequence

6.23(a): Scene 1

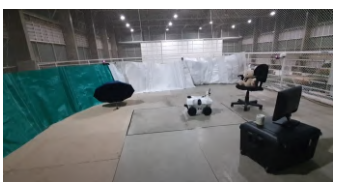
6.23(b): Scene 2



6.23(c): Scene 3

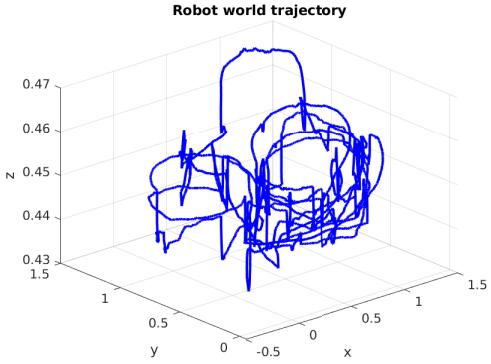
6.23(d): Scene 4

6.23(e): Scene 5

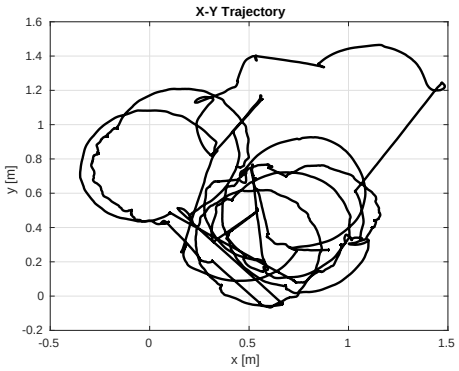


6.23(f): Scene 6

Figure 6.23: Selected scenes from the *Shift* sequence

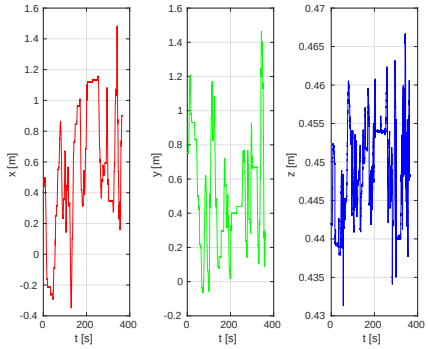


6.24(a): 3D ground-truth trajectory

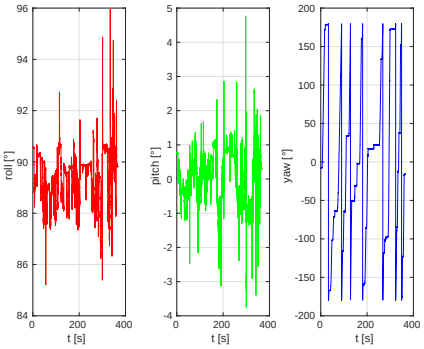


6.24(b): 2D ground-truth trajectory

Figure 6.24: Ground-truth trajectory of the *Shift* sequence



6.25(a): X-Y-Z coordinates over time

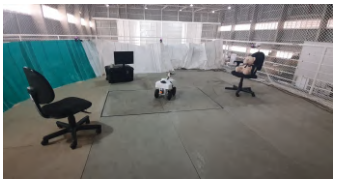


6.25(b): Roll-Pitch-Yaw over time

Figure 6.25: Metrics from the *Shift* ground-truth sequence

6.26(a): Scene 1

6.26(b): Scene 2



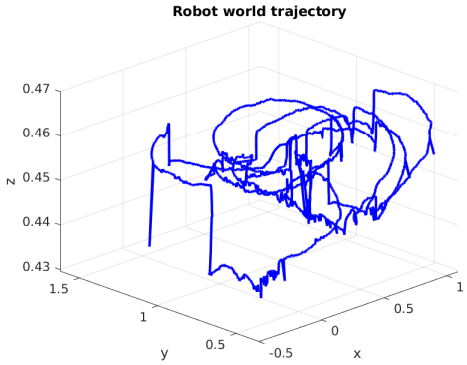
6.26(c): Scene 3

6.26(d): Scene 4

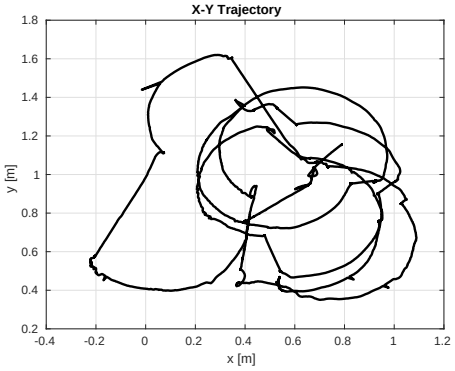
6.26(e): Scene 5

6.26(f): Scene 6

Figure 6.26: Selected scenes from the *Changing2* sequence

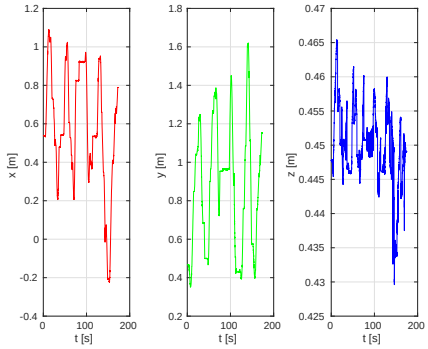


6.27(a): 3D ground-truth trajectory

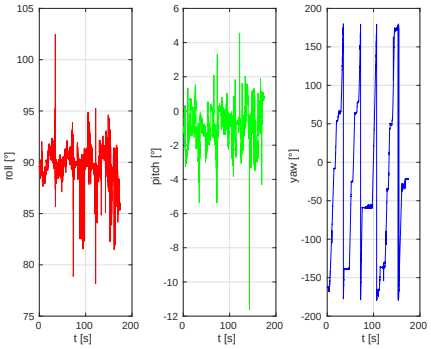


6.27(b): 2D ground-truth trajectory

Figure 6.27: Ground-truth trajectory of the *Changing2* sequence



6.28(a): X-Y-Z coordinates over time



6.28(b): Roll-Pitch-Yaw over time

Figure 6.28: Metrics from the *Changing2* ground-truth sequence

7
Semantic SLAM in Dynamic and Changing Environments

7.1
Introduction

The methods proposed in Chapters 3, 4 and 5 could deal with highly dynamic environments, but are not robust to changing environments, i.e., when aspects of the the environment are modified after the robot has already mapped it.

As stated in Chapter 1, there are methods that deal with changing environments that consider the pose of the camera known [64], i.e., which do not perform SLAM, but mapping with known poses. Other methods [115][75] perform localization in changing environments with a known map. Also, DXSLAM [78] has an increased robustness to changing environments, but fails in regular dynamic environments. The SLAM methods found in the literature that are robust to both dynamic and changing environments only perform 2D SLAM using LiDAR fused with other sensors such as IMU and odometry [67].

This chapter presents Changing-SLAM, a method for robust Visual SLAM in dynamic and changing environments. It maintains a semantic map of the environment, updating the belief about the poses of objects in time. To the best of our knowledge, this is the first Visual SLAM system that is robust to both dynamic and changing environments, not assuming a given camera pose nor a known map. The proposed method is tested using datasets and experiments, and compared with other state-of-the-art methods.

Section 7.2 presents the problem formulation, Section 7.3 details the proposed methodology, Section 7.4 presents the results and comparisons with the state-of-the-art. Finally, Section 7.5 shows the conclusions and suggestions for future works.

7.2
Problem Formulation

The semantic SLAM problem can be formulated as stated in Eq. (7-1), where $\hat{X}$ is the set of estimated robot poses, $\hat{L}$ is a set of estimated landmark

positions and semantic classes, and Z is a set of sensor measurements [116].

$$\hat{X}, \hat{L} = \underset{X,L}{\operatorname{argmax}}\, p(X, L|Z) \tag{7-1}$$

In the proposed approach, however, the semantic SLAM formulation is decoupled from the geometric one. An object belief map is created and updated acting as a pre-filter for the standard feature-based SLAM procedure, making the SLAM process occur in two levels. Within the high level (objects), a Bayesian filter is used to create a belief map about the poses of objects in the map. This belief map decides which features are used for the low-level step. Within the low level (points), the SLAM problem is solved using the feature-based methods proposed by ORB-SLAM3, adapted for dynamic environments, as proposed in Chapter 5. This approach results in a reliable tracking system, robust to changes in the map, and a semantic map in an object-level that can be used for other problems such as autonomous navigation.

7.3 Methodology

Figure 7.1 shows the framework of the proposed methodology. The system is built on ORB-SLAM3, and is composed of four threads running in parallel: Object Detection, Tracking, Local Mapping, and Loop Closing. Changing-SLAM modifies all three threads of ORB-SLAM3 and adds a new thread for object detection.

The same semantic detection approach from Chapter 5 is used in Changing-SLAM, using YOLOv4 [11] for object detection. The keypoint classification pipeline, the feature repopulation technique, the *a priori* people filtering, and the short-term data association are also performed using the same approach presented in Chapter 5.

Changing-SLAM explores one of the main novelties of ORB-SLAM3, the Atlas framework [34]. It is a multi-map system that can store a set of disconnected maps, and merge them when a loop is detected. This considerably improves the SLAM solution in scenarios with lost tracks. The proposed approach modifies Atlas to include storage and operations with MapObjects.

A long-term data association is also proposed to decide whether an object detected in the current frame is already in the map, or if it is a new object. The long-term data association is also responsible for updating the belief about the persistence of the objects in the environment. Finally, MapPoints associated to MapObjects with a low belief are filtered from the loop closing and graph-optimization steps.

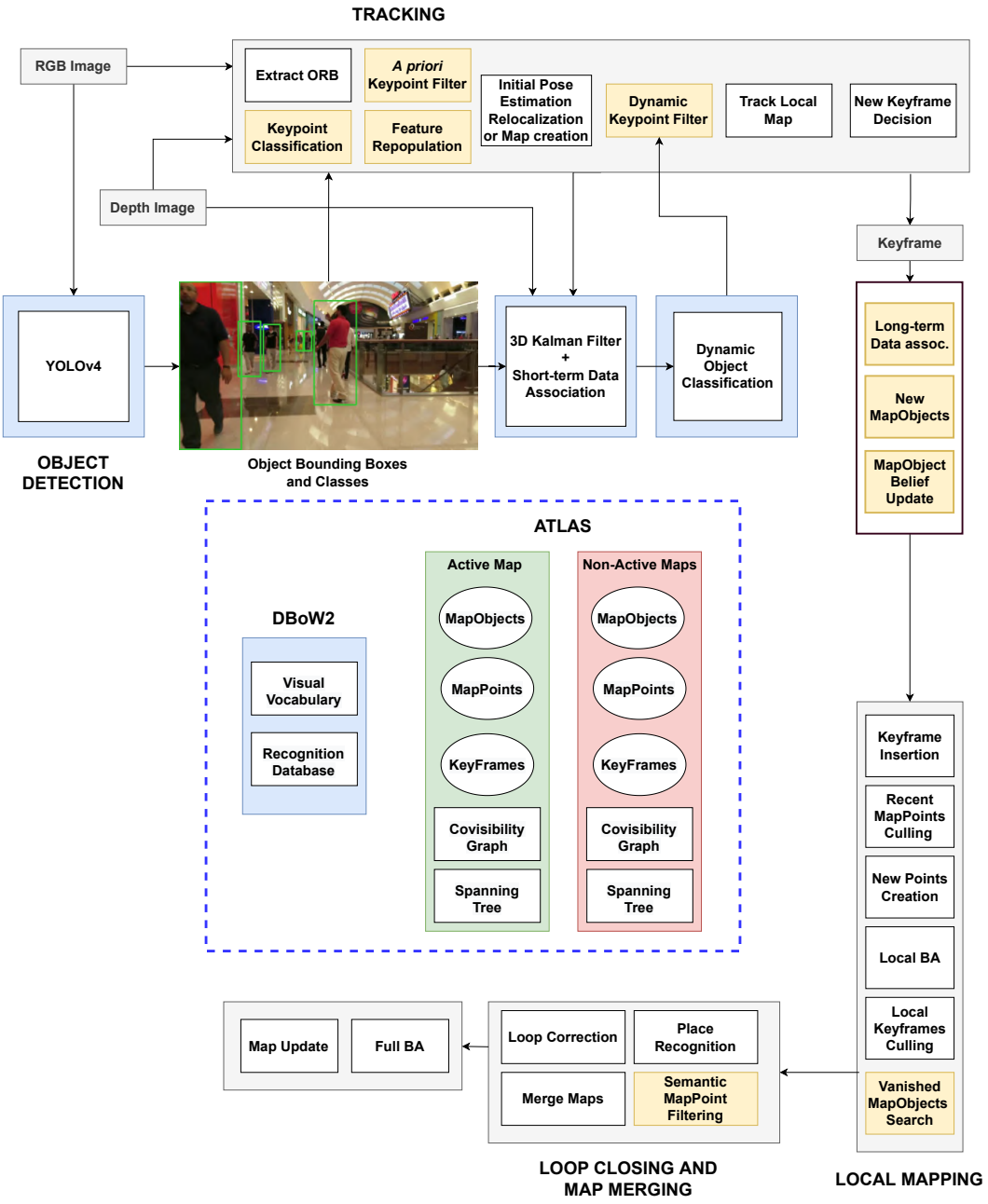


Figure 7.1: Main components of Changing-SLAM

7.3.1 MapObjects and MapPoints

The MapObject is the key element in this methodology, being responsible for creating the connection between semantic and geometric formulations. This is an improved version of the MapObject class proposed in Chapter 5. Each MapObject stores:

- First bounding box
- Current bounding box

- Class
- List of associated MapPoints
- 3D position in world coordinates
- Unique global ID
- Belief of persistence
- 3D bounding box dimensions

The belief of persistence is the numerical value that will determine whether the MapObject, and its associated MapPoints, will remain active in the map. Details of its initialization and update are given in Sections 7.3.2 and 7.3.3.

The 3D position is obtained by computing the centroid of the associated MapPoints. The maximum object dimensions are obtained by computing the median of the 5% maximum and minimum x, y and z coordinates of the associated MapPoints. The MapPoints are also an updated version of the one presented in Chapter 5. Each MapPoint has a key that allows it to be used or not for mapping and loop closure.

The proposed MapObject formulation for object representation has advantages in comparison with other methods in the literature. As stated in Chapter 1, Gomez et al. [64] proposed a 3D cuboid generation using point clouds and a 2D bounding box to calculate the centroid of the object. They used the 2% smallest depths inside the bounding box to define the minimum object depth. Figure 7.2 shows a frame from the fr3_office sequence of the TUM Dataset [80]. The chair on the right is occluded by an undetected object. Using the approach from Gomez et al. would lead to a wrong estimation. The proposed approach, on the other hand, can deal with occlusions by known or unknown objects.

7.3.2 MapObject Persistence

This work proposes a recursive Bayes' filter [13] to estimate the belief about the MapObjects' persistence in the map. When a MapObject is created, the belief is set to 0.5.

H is a discrete random variable that represents the persistence of a given MapObject initialized at a certain 3D position. It can assume the values 0 or 1, i.e., either the object is not there or it is there. The belief about the persistence of a given MapObject is stated by

$$bel(H) = \eta\, p(Y|H)p(H) \tag{7-2}$$

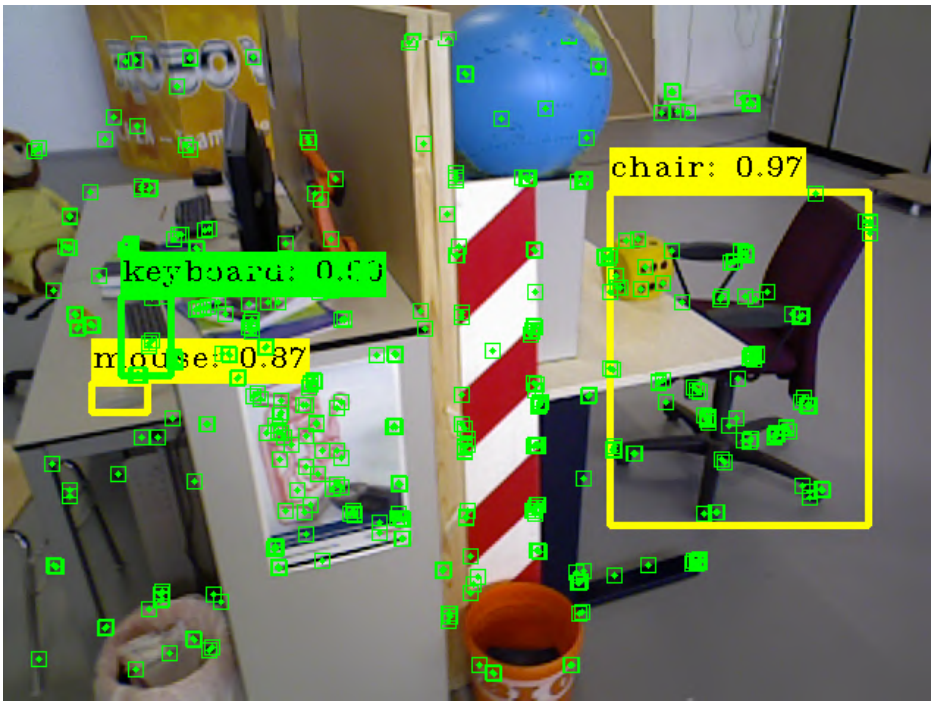


Figure 7.2: RGB frame from the fr3_office sequence of the TUM Dataset. The chair is being partially occluded by an undetected object.

where η is a normalization factor given by

$$\eta = \frac{1}{bel(H=1) + bel(H=0)} \tag{7-3}$$

$$bel(H=1) = \eta\, p(Y|H=1)p(H=1) \tag{7-4}$$

$$bel(H=0) = \eta\, p(Y|H=0)p(H=0) \tag{7-5}$$

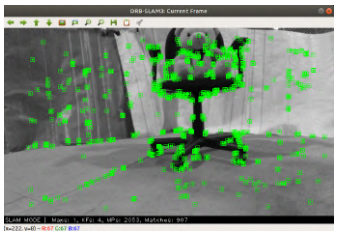
For each iteration, the prior $p(H)$ is the last belief. The likelihood $p(Y|H)$ is measured using the distance between the centroids, the measured and the one in the map. If an object is not detected at the place it was previously seen, its belief is lowered. The belief update of each MapObject is performed during the long-term data association, which is explained in the next section.

7.3.3 Long-term Data Association

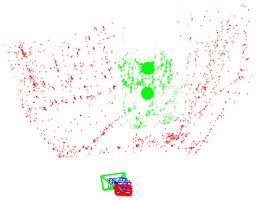
The long-term data association evaluates if the MapObjects created in the current frame correspond to existing objects in the map. This process is detailed in Algorithm 4. First, the centroid of a MapObject candidate is computed using its associated MapPoints. Then, the Euclidean norm between the candidate's pose and close MapObjects with the same class is computed. If they are close enough, they are merged and their MapPoints are combined. Also, the belief of the object in the Map is updated accordingly.

If the belief of a MapObject is below a threshold (BeliefThreshold), then all its associated MapPoints are marked as inactive and cannot be used for tracking, mapping or loop closure. As all MapObjects start with a 0.5 belief, initially all MapPoints that belong to MapObjects are inactive. With this approach, objects that are moved or vanish from the map cannot interfere in the mapping process. As an object continues to be seen, its belief grows and, eventually, it is added to the map and its MapPoints become active.

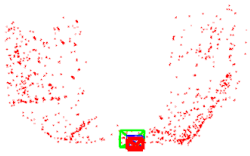
Figures 7.3(a)-7.3(c) show an example of the proposed method in the OneChair sequence of the PUC/USP dataset. There are two objects in the scene: a chair and a teddy bear. First, the features are detected in the RGB image, as shown in Fig. 7.3(a). The keypoints are classified as belonging to the objects or to the background. Figure 7.3(b) show the MapPoints in green classified as belonging to the objects, and the MapPoints in red as from the background. The larger green dots represent the centroids of the MapObjects. The green MapPoints are initially removed from the map, as shown in Fig. 7.3(c), as they belong to MapObjects which beliefs are 0.5. However, both objects are detected, estimated and stored in memory. If the objects are observed again later, their belief would increase and they would be inserted again in the map together with their MapPoints.



7.3(a): Feature detection



7.3(b): Initial classification



7.3(c): Filtered map

Figure 7.3: MapPoint filtering process performed by Changing-SLAM in the *OneChair* sequence. The two larger green dots represent two MapObjects in the map. The small green dots in the map represent the MapPoints associated to the MapObjects.

7.3.4 Semantic Map

The final output of Changing-SLAM is the complete camera trajectory and a metric-semantic map. The metric map is composed of active MapPoints. The semantic map is composed of MapObjects with their respective centroids and 3D bounding boxes, as shown in Fig. 7.4. Only objects with a high belief

Algorithm 4: Long-term Data Association

Data: Current frame F_k, last frame F_{k-1}, list of MapObject candidates $newMO$, current Map, current keyframe KF_k, last keyframe KF_{k-1}, list of MapObjects in the current Map $ObjInMap$

```
1  for Obj in newMO do
2      if number of MapPoints in Obj > 0 then
3          if Obj is new then
4              Compute centroid of Obj;
5              merged = false;
6              Get Obj pose ObjPose;
7              for OiM in ObjInMap do
8                  Get OiM pose;
9                  if class of Obj == class of OiM then
10                     dist = euclidean norm (Obj pose, OiM pose);
11                     if dist < ltdaThreshold then
12                         merge objects (Obj, OiM);
13                         merged = true;
14                         if OiM was not saw in the past N Keyframes
                            then
15                             Update Belief of OiM;
16                             OiM last saw in KF_k;
17                             OiMMP = Get all MapPoints from OiM;
18                             if Belief < BeliefThreshold then
19                                 OiMMP inactive;
20                             end
21                             else
22                                 OiMMP active;
23                             end
24                         end
25                         else
26                             OiM last saw in KF_k;
27                         end
28                     end
29                 end
30             end
31             if merged is false then
32                 Add Obj in the Map;
33             end
34         end
35         else if Obj is not in ObjInMap then
36             perform steps of lines 4 to 33;
37         end
38         else
39             Compute centroid of Obj;
40             Update Obj in map;
41         end
42     end
43 end
```

are active in the map. This information can be very useful for autonomous navigation.

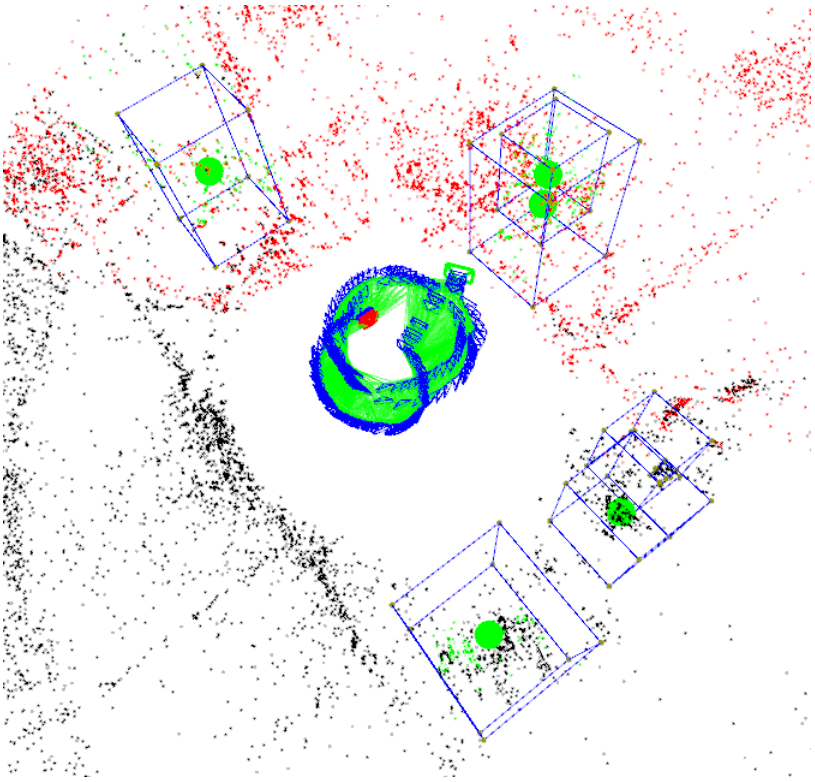


Figure 7.4: Semantic Map generated by Changing-SLAM in the *Static* sequence of the PUC/USP Dataset

7.4 Results

7.4.1 TUM Dataset

Five dynamic sequences from the TUM Dataset were used to evaluate the robustness of Changing-SLAM against dynamic environments: the four *walking* sequences and one *sitting* sequence to be used as a baseline. Table 7.1 show the parameters used in the TUM simulations.

The system was compared to ORB-SLAM3 [3] and DXSLAM. The DXSLAM results were obtained in the publication from Li et al. [78]. Table 7.2 shows the RMSE of the ATE comparison between Changing-SLAM, ORB-SLAM3 and DXSLAM. Changing-SLAM outperformed the other systems in all evaluated sequences. The results show that both ORB-SLAM3 and DXSLAM cannot cope with dynamic environments. Their results are satisfactory only in the *sitting* sequence, were the people in the scene are sitting, just moving their hands. The results also show that Changing-SLAM is robust in

dynamic environments. Figures 7.5(a) and 7.5(b) show, respectively, the comparison between the ground-truth and estimated trajectories of ORB-SLAM3 and Changing-SLAM. The trajectory estimated by ORB-SLAM3 completely deviates from the ground-truth.

Table 7.1: Parameters used in the TUM simulations

Description	Value
Number of features	1500
IoU threshold	0.15
Depth threshold [m]	0.1
DOC threshold [m/s]	0.1
ltda threshold [m]	0.3
Belief threshold	0.8

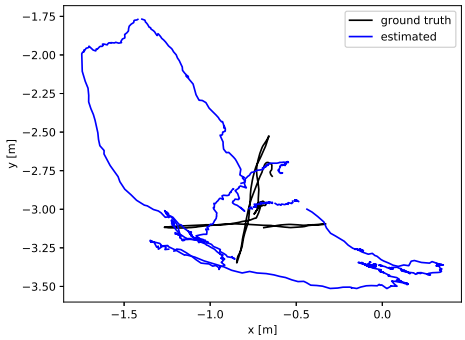
Table 7.2: Comparison of the RMSE of ATE [m] of Changing-SLAM against ORB-SLAM3 and DX-SLAM using the TUM dataset

Sequence	ORB-SLAM3	DXSLAM	Changing-SLAM
fr3_s_xyz	**0.009**	—	0.018
fr3_w_static	0.038	0.0167	**0.008**
fr3_w_xyz	0.819	0.3088	**0.016**
fr3_w_rpy	0.957	—	**0.067**
fr3_w_half	0.315	—	**0.039**

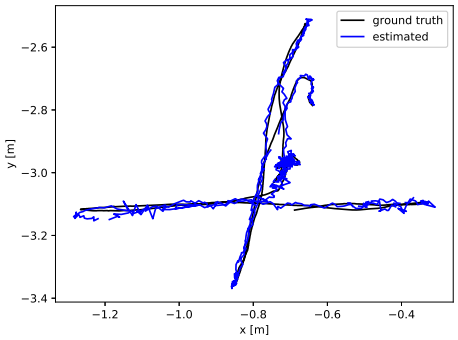
7.4.2 PUC/USP Dataset

The PUC/USP Dataset, presented in Chapter 6, was used to evaluate the robustness of the proposed methodology in changing environments. The same metrics proposed by Sturm et al. [80] are used: ATE and RPE. Table 7.3 shows the parameters used in the simulations.

The *Static* sequence is the first one chosen for evaluation, to be used as a baseline. Figures. 7.6(a) and 7.6(b) show the comparison between the ground-truth and estimated trajectories of ORB-SLAM3 and Changing-SLAM, respectively. As expected, both systems achieved good results. Figures 7.7(a) through 7.11(b) show the same comparison for the other sequences of the



7.5(a): ORB-SLAM3



7.5(b): Changing-SLAM

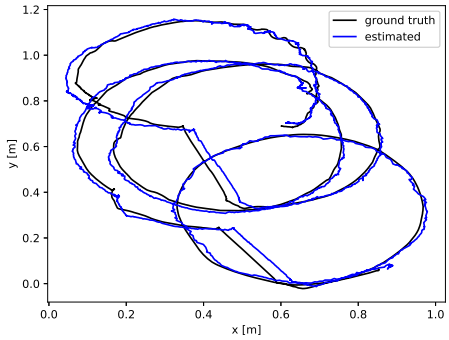
Figure 7.5: Ground truth and estimated trajectory in the sequence fr3_w_xyz

Table 7.3: Values of parameters used in the PUC/USP simulations

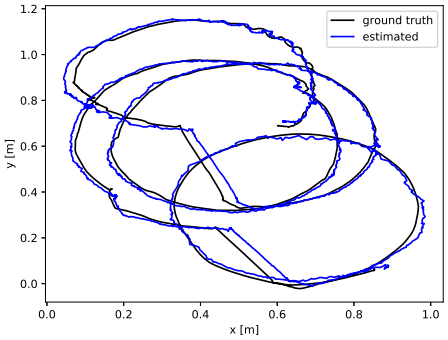
	Sequence					
Description	Static	OneChair	Vanishing	Changing	Shift	Changing2
Number of features	2000	2650	2000	1200	2650	2650
IoU threshold	0.15	0.15	0.15	0.15	0.15	0.15
Depth threshold [m]	0.1	0.1	0.1	0.1	0.1	0.1
DOC threshold [m/s]	0.1	0.1	0.1	0.1	0.1	0.1
ltda threshold [m]	0.3	0.3	0.3	0.3	0.3	0.3
Belief threshold	0.8	0.8	0.8	0.8	0.8	0.8

dataset. Table 7.4 shows the ATE comparison between ORB-SLAM3, DXSLAM and Changing-SLAM. Despite the fact that Changing-SLAM improved every result of ORB-SLAM3, it is noticeable that the effect of changing environments is not always critical for the localization accuracy, as it happens in dynamic environments. The *Vanishing* sequence was not expected to cause a major error, because there are no new objects added in the scene, which could have caused a wrong loop closure. The *Changing* sequence caused an increase in the localization error of ORB-SLAM3, which was corrected by Changing-SLAM.

The sequence *OneChair*, on the other hand, triggered a wrong loop closure in ORB-SLAM3, which caused a considerable increase in the RMSE. Table 7.5 shows the comparison of the ATE between Changing-SLAM, DXSLAM and ORB-SLAM3 for the *OneChair* sequence. Bold numbers indicate the best results in the metric. The error of ORB-SLAM3 is one order of magnitude higher

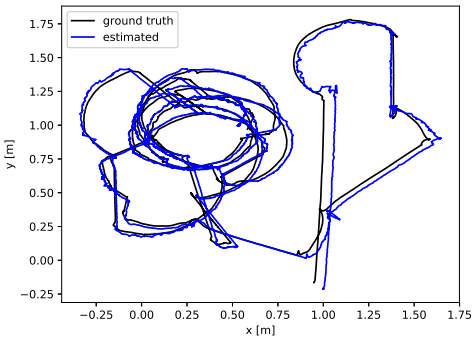


7.6(a): ORB-SLAM3

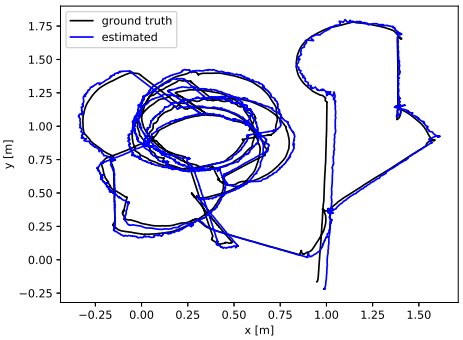


7.6(b): Changing-SLAM

Figure 7.6: Comparison of ground truth and estimated trajectory in the sequence *Static*



7.7(a): ORB-SLAM3



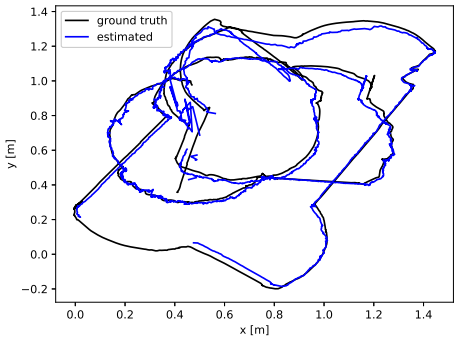
7.7(b): Changing-SLAM

Figure 7.7: Comparison of ground truth and estimated trajectory in the sequence *Vanishing*

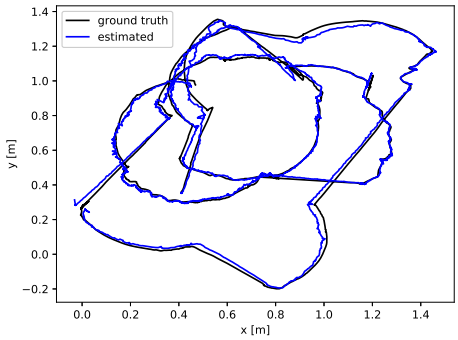
Table 7.4: Comparison of the RMSE of ATE [m] of Changing-SLAM against ORB-SLAM3 and DX-SLAM using the PUC/USP dataset

Sequence	ORB-SLAM3	DX-SLAM	Changing-SLAM
Static	**0.033**	0.036	**0.033**
OneChair	0.407	0.097	**0.089**
Vanishing	0.052	0.062	**0.049**
Changing	0.071	0.044	**0.029**
Shift	**0.075**	0.077	**0.075**
Changing2	0.049	0.055	**0.047**

than DX-SLAM and Changing-SLAM.

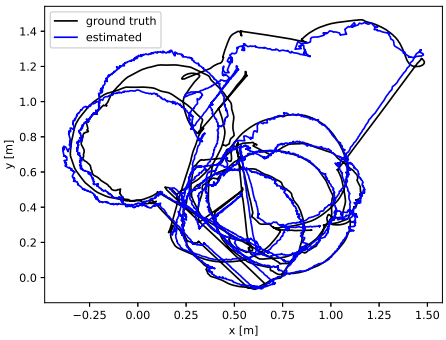


7.8(a): ORB-SLAM3

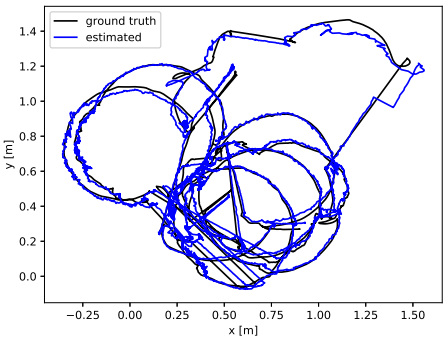


7.8(b): Changing-SLAM

Figure 7.8: Comparison of ground truth and estimated trajectory in the sequence *Changing*

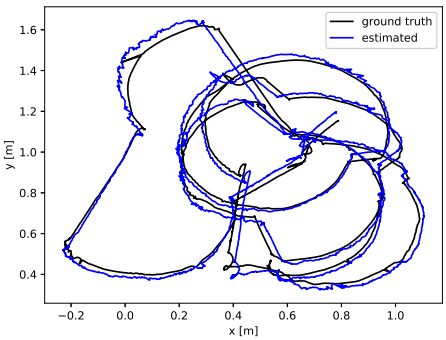


7.9(a): ORB-SLAM3

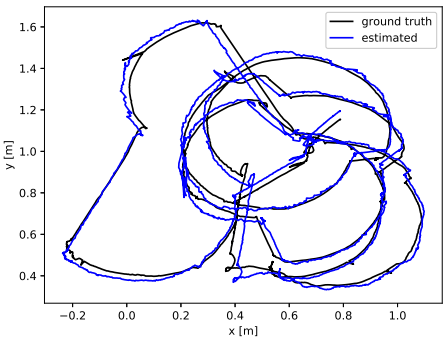


7.9(b): Changing-SLAM

Figure 7.9: Comparison of ground truth and estimated trajectory in the sequence *Shift*

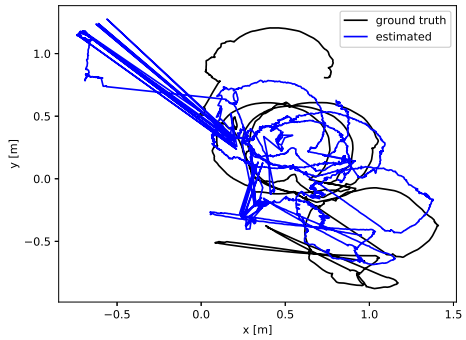


7.10(a): ORB-SLAM3

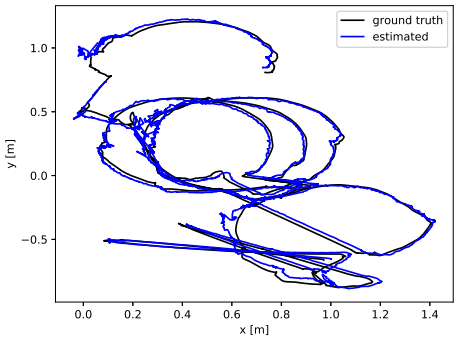


7.10(b): Changing-SLAM

Figure 7.10: Comparison of ground truth and estimated trajectory in the sequence *Changing2*



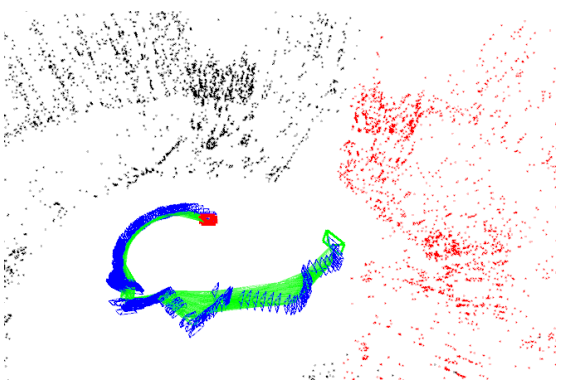
7.11(a): ORB-SLAM3



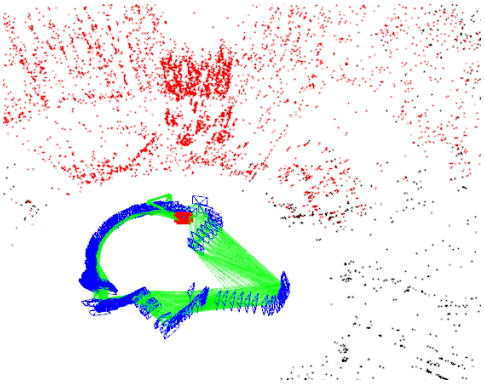
7.11(b): Changing-SLAM

Figure 7.11: Comparison of ground truth and estimated trajectory in the sequence *OneChair*

Figures 7.12(a) and 7.12(b) show the wrong loop closure made by ORB-SLAM 3 in the OneChair sequence. The shapes of two chairs are noticeable in Fig. 7.12(a), merged into one chair in Fig. 7.12(b). The wrong loop occurred due to the change in the chair position and the inability of ORB-SLAM3 to detect this change. After this wrong loop closure, the system is no longer able to recover from the error, even with the robust multi-mapping and re-localization systems of ORB-SLAM3. Figures 7.13(a) and 7.13(b) show, respectively, the pose-graph of ORB-SLAM3 and Changing-SLAM in the *OneChair* sequence.



7.12(a): Before loop closure



7.12(b): After loop closure

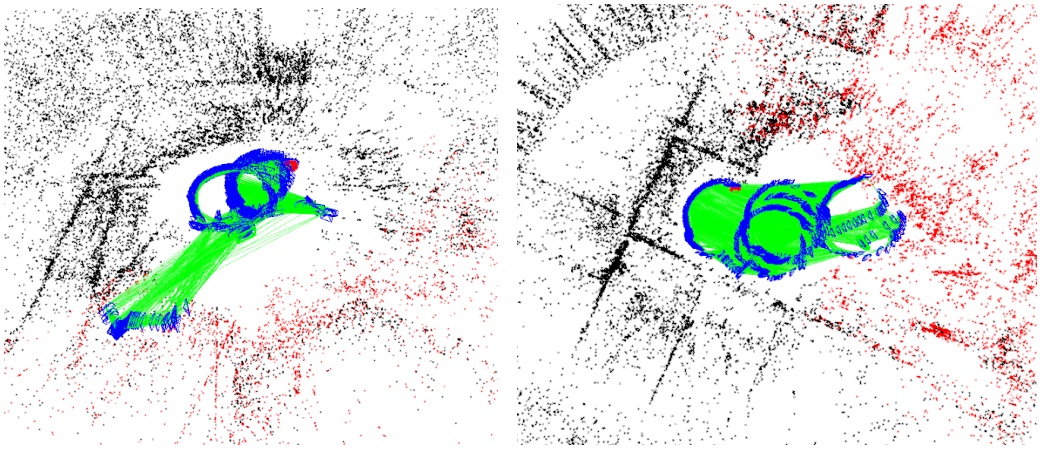
Figure 7.12: Wrong loop detection by ORB-SLAM3 in the OneChair sequence

7.4.3 Run-time Analysis

All tests were performed on a notebook with an Intel Core i7-10750H and 16 GB of RAM running Ubuntu Linux 18.04 LTS. The system is implemented in C++, and the object detection is performed with OpenCV 4.5, using a

Table 7.5: Comparison of the ATE [m] of Changing-SLAM against ORB-SLAM3 and DynaSLAM using the OneChair sequence

Metric	ORB-SLAM3	DX-SLAM	Changing-SLAM
RSME	0.407	0.097	**0.089**
mean	0.323	0.077	**0.067**
median	0.213	0.060	**0.043**
std	0.247	0.059	**0.058**



7.13(a): ORB-SLAM3 7.13(b): Changing-SLAM

Figure 7.13: Comparison of pose-graphs between ORB-SLAM3 and Changing-SLAM in the OneChair sequence

Nvidia GeForce RTX 2060 GPU. Table 7.6 shows the mean frame rate in ms of the proposed method for every sequence of the PUC/USP dataset. Changing-SLAM achieved an average tracking speed of 23.8 FPS, which can be categorized as real time.

7.4.4 Limitations

Changing-SLAM does not deal with deformable objects such as blankets, rigid objects that can change shape such as cabinets with closed and open doors, and objects not labeled in the object detector training process. The latter problem can be solved re-training the network to include more objects that are common to the environment.

The COCO Dataset [104] has several classes that are not suitable for indoor environments, which is the scope of this work, such as cars, and even other classes that are not common for regular outdoor situations, such as giraffes and other wild animals. Therefore, it would be beneficial to train the

Table 7.6: Mean tracking time [ms] of the proposed method in the PUC/USP sequences

Sequence	Changing-SLAM
Static	38.0
OneChair	47.3
Vanishing	42.4
Changing	41.4
Shift	42.0
Changing2	41.3
Average	42.1

network with more common indoor objects. However, this approach still cannot deal with unexpected new objects, as performed in the approach proposed by Ji et al. [53].

The other issues are harder to solve, especially the one with deformable objects, which is still an open problem, with recent new solutions [117].

7.5 Conclusions

This chapter presented Changing-SLAM, our proposed method to perform visual SLAM in both dynamic and changing scenarios in real time. To our knowledge, this is the first system able to perform these tasks simultaneously in real time only using a camera.

The proposed method was tested with a dataset especially designed for the evaluation of visual SLAM systems in changing scenarios, achieving a high accuracy in comparison with ORB-SLAM3 and DXSLAM. Changing-SLAM is very robust in such scenarios, preventing the detection of a wrong loop closure that would ruin the SLAM process.

Besides correcting localization, the semantic map generated by Changing-SLAM can be useful for a wide variety of applications. One example would be within the work from Chen and Liu [96], which generates navigable paths from the maps generated by ORB-SLAM2 and ORB-SLAM3. These maps would be corrupted if objects were moved in the scene.

Finally, as explained in Chapter 5, the use of object detection and feature repopulation to differentiate object features from the background ones is a method to decrease the computational effort of the system. However, the semantic mapping, dynamic object filtering and belief update methods are not restricted to that. The proposed method can be performed using other

types of semantic detection such as instance or panoptic segmentation, when they become computationally feasible. Therefore, for future work, panoptic segmentation could be used in the methodology to evaluate if there is an improvement in accuracy.

8
Conclusions and Future Work

This thesis has presented four different formulations for visual SLAM in dynamic and changing scenarios. The proposed formulations ranged from a simple filter combined with a deep learning framework, to advanced techniques for more complex scenarios.

First, a simple solution to the problem of visual SLAM in human environments was proposed. The system was tested using a differential drive robot and a stereo camera within an indoor environment, with people walking in front of the camera as the robot wandered through the scene. The solution consisted of a filter that prevented the main SLAM system to work while there were people in the scene, using a deep learning-based object detection framework to identify their presence. The system was tested with RTAB-Map [28][92], and achieved good qualitative results.

Moreover, an open source method designed for crowded environments was proposed. It is the first visual SLAM system designed for this purpose. The methodology included a keypoint filtering algorithm, a method to avoid feature depletion, and CYTi, an object detection framework built on YOLOv3-tiny especially designed for the operation in crowded scenarios, with the accuracy of YOLOv3, but with the inference speed of YOLOv3-tiny. The system was tested with three different datasets for dynamic environments and compared with several visual SLAM systems, outperforming them in multiple sequences with a real-time performance. Also, Crowd-SLAM achieved, to our knowledge, the best results in the literature in 3 sequences of the Bonn dataset.

Furthermore, this thesis presented the development of a visual SLAM system robust to dynamic environments, including moving objects, rather than just people. The methodology consists of detecting and tracking objects in the scene, using object detection and an extended Kalman filter. A method was proposed to effectively classify and represent objects using only object detection and depth information. The proposed system was evaluated using datasets and compared with 13 other systems from the literature, achieving an accuracy similar to several highly accurate offline methods, however with the advantage of working in real time.

Finally, this thesis proposed contributions to the problem of visual SLAM

in changing environments, where objects are moved outside of the field of view of the robot, after the scene had already been mapped. First, a new dataset has been developed with RGB-D data especially designed for the evaluation of changing environments on an object level, called PUC/USP dataset. To our knowledge, this is the first dataset focused on this problem. Six sequences were created using a mobile robot, an RGB-D camera and a motion capture system.

The PUC/USP dataset was then used to evaluate the robustness of an additional method proposed in this thesis, called Changing-SLAM. This formulation extends the previous ones providing the ability to work in changing environments. This is a very challenging problem, with solutions usually relying on assumed known poses, or focusing on localization with *a priori* made maps. Also, the existing solutions for 3D environments are not simultaneously robust to dynamic objects moving in front of the camera. To our knowledge, Changing-SLAM is the first system that is robust in both situations, being also able to operate in real time. This was achieved by using a Bayesian filter combined with a long-term data association algorithm. The proposed approach was tested and compared with other systems in the literature, proving it was the only one robust to both dynamic and changing scenarios.

8.1 Future Works

This thesis opens several possibilities for future works. Firstly, all of the proposed approaches obtained its semantic classification through an object detection framework with a limited number of class labels. Using proper training, the detection system can cover most of the relevant objects that can be found in typical indoor environments such as residences or offices. If the robot will operate in more specific environments, such as a hospital, for example, the detector could be trained for this purpose. However, it would be desirable to have a system with a certain level of robustness to unpredictable classes. Furthermore, the proposed methods cannot deal with deformable objects, or rigid objects that change its shape, such as cabinets or doors.

Regarding the PUC/USP dataset, future versions could include data from other sensors, such as IMU, using different objects, and testing other scene configurations.

Furthermore, despite being effective, the object persistence model, proposed in Chapter 7, only updates the belief about the states of the objects in the scene when the robot revisits the location where the objects had been mapped. However, it could be argued that an object eventually changes its position regardless of the state of the robot. An interesting approach would be

to include a temporal factor in the object persistence model, as done in the approach proposed for features by [73] and [66].

Another interesting evaluation would be to propose a method for coupled Multi-object Tracking and SLAM, such as the one in CubeSLAM [58], DynaSLAM II [118], or [119], adding the methods to handle changing environments, and comparing with the approach proposed in Chapter 7.

Finally, it would be interesting to study the use of recent new technologies such as event cameras [120], to evaluate their advantages in changing scenarios, and the possibility of their integration with the proposed algorithms.

8.2 Publications

Several articles were published during the development of this thesis. The following journal paper was published in the Journal of Intelligent & Robotic Systems, regarding the Crowd-SLAM formulation:

– J. C. V. Soares, M. Gattass and M. A. Meggiolaro, **Crowd-SLAM: Visual SLAM Towards Crowded Environments using Object Detection**. Journal of Intelligent & Robotic Systems 102, 50 (2021).

The following papers were published in international conferences:

– J. C. V. Soares, M. Gattass and M. A. Meggiolaro, **Visual SLAM in Human Populated Environments: Exploring the Trade-off between Accuracy and Speed of YOLO and Mask R-CNN**, 2019 19th International Conference on Advanced Robotics (ICAR), 2019, pp. 135-140.

– J. C. V. Soares, M. Gattass and M. A. Meggiolaro, **Mapping in Dynamic Environments using Deep Learning-based Human Detection**, 25th ABCM International Congress of Mechanical Engineering (COBEM 2019), Uberlândia, MG, Brazil, 2019.

– J. C. V. Soares and M. A. Meggiolaro, **Keyframe-Based RGB-D SLAM for Mobile Robots with Visual Odometry in Indoor Environments Using Graph Optimization**, 2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE), 2018, pp. 94-99

Moreover, the following two papers are in the final stages of production for future submission:

– J. C. V. Soares, V. S. Medeiros, G. F. Abati, M. Gattass, and M. A. Meggiolaro, **Semantic Visual SLAM in Dynamic and Changing Environments**. Robotics and Automation Letters, 2022.

– J. C. V. Soares, V. S. Medeiros, G. F. Abati, M. Becker, G. Caurin, M. Gattass, and M. A. Meggiolaro, **A Dataset for RGB-D SLAM in Changing Environments**, International Conference on 3D Vision, 2022.

Besides the thesis-related works, other publications related to Mobile Robotics were published during the period of this thesis:

– G. F. Abati, J. C. V. Soares, M. Gattass, and M. A. Meggiolaro, **People Following System for Holonomic Robots Using an RGB-D Sensor**. 26th ABCM International Congress of Mechanical Engineering (COBEM), 22-26 de novembro de 2021.

– J. C. V. Soares, G. F. Abati, G. H. D. Lima and M. A. Meggiolaro, **Autonomous Navigation System for a Wall-painting Robot based on Map Corners**. 2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE), 2020, pp. 1-6

– J. C. V. Soares, G. F. Abati, G. H. D. Lima, C. L. M. de Souza and M. A. Meggiolaro, **Project and Development of a Mecanum-wheeled Robot for Autonomous Navigation Tasks**. Proceedings of the XVIII International Symposium on Dynamic Problems of Mechanics, 2019.

Moreover, the following article involving an autonomous mobile robot has been submitted and awaits for final approval.

– J. Q. M. Guedes, J. C. V. Soares, T. S. B. da Cruz, M. A. Meggiolaro, J. H. G. Batista, **Autonomous modular robotic platform for rigless operations**. Rio Oil & Gas Expo and Conference 2022.

Bibliography

[1] R. Mur-Artal and J. D. Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[2] J. C. V. Soares, M. Gattass, and M. A. Meggiolaro. Visual slam in human populated environments: Exploring the trade-off between accuracy and speed of yolo and mask r-cnn. In *2019 19th International Conference on Advanced Robotics (ICAR)*, pages 135–140, 2019.

[3] C. Campos, R. Elvira, J. J. Gómez Rodríguez, J. M. M. Montiel, and J. D. Tardós. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021.

[4] A. Garcia-Garcia, S. Orts, S. Oprea, V. Villena-Martinez, and J. G. Rodríguez. A review on deep learning techniques applied to semantic segmentation. *ArXiv*, abs/1704.06857, 2017.

[5] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[6] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[7] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[9] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[10] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár. Panoptic segmentation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9396–9405, 2019.

[11] A. Bochkovskiy, C. Wang, and H. M. Liao. Yolov4: Optimal speed and accuracy of object detection. *CoRR*, 2020.

[12] J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J. M. Rendón-Mancha. Visual simultaneous localization and mapping: a survey. *Artificial Intelligence Review*, 43(1):55–81, 2015.

[13] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[14] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics Automation Magazine*, 13:99 – 110, 2006.

[15] H. F. Durrant-Whyte, D. C. Rye, and E. M. Nebot. Localization of autonomous guided vehicles. 1996.

[16] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(3):611–625, 2018.

[17] L. Valgaerts, A. Bruhn, M. Mainberger, and J. Weickert. Dense versus sparse approaches for estimating the fundamental matrix. *International Journal of Computer Vision*, 96:212–234, 2011.

[18] Z. Min and E. Dunn. Voldor+slam: For the times when feature-based or direct methods are not good enough. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13813–13819, 2021.

[19] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136, 2011.

[20] J. Engel, T. Schöps, and D. Cremers. Lsd-slam: Large-scale direct monocular slam. In *Computer Vision – ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part II*, pages 834–849, 2014.

[21] A. Davison, I. Reid, N. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1052–1067, 2007.

[22] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1–10. IEEE Computer Society, 2007.

[23] T. Taketomi, H. Uchiyama, and S. Ikeda. Visual slam algorithms: a survey from 2010 to 2016. *IPSJ T Comput Vis Appl*, 9(16), 2017.

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

[25] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011.

[26] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *The International Journal of Robotics Research*, 31:647–663, 2012.

[27] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard. 3-d mapping with an rgb-d camera. *IEEE Transactions on Robotics*, 30:177 – 187, 2014.

[28] M. Labbé and F. Michaud. Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics*, 36(2):416–446, 2019.

[29] J. C. V. Soares, M. Gattass, and M. A. Meggiolaro. Crowd-slam: Visual slam towards crowded environments using object detection. *Journal of Intelligent & Robotic Systems*, 102(2):50, May 2021.

[30] B. Bescos, J. M. Fácil, J. Civera, and J. Neira. Dynaslam: Tracking, mapping, and inpainting in dynamic scenes. *IEEE Robotics and Automation Letters*, 3(4):4076–4083, 2018.

[31] C. Yu, Z. Liu, X. Liu, F. Xie, Y. Yang, Q. Wei, and Q. Fei. Ds-slam: A semantic visual slam towards dynamic environments. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1168–1174. IEEE, 2018.

[32] X. Yuan and S. Chen. Sad-slam: A visual slam based on semantic and depth information. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020.

[33] R. Mur-Artal and J. D. Tardós. Visual-inertial monocular slam with map reuse. *IEEE Robotics and Automation Letters*, 2(2):796–803, 2017.

[34] R. Elvira, J. D. Tardós, and J. M. M. Montiel. Orbslam-atlas: a robust and accurate multi-map system. *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6253–6259, 2019.

[35] R. Scona, M. Jaimez, Y. R. Petillot, M. Fallon, and D. Cremers. Staticfusion: Background reconstruction for dense rgb-d slam in dynamic environments. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3849–3856, 2018.

[36] E. Palazzolo, J. Behley, P. Lottes, P. Giguère, and C. Stachniss. Refusion: 3d reconstruction in dynamic environments for rgb-d cameras exploiting residuals. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7855–7862, 2019.

[37] A. Dib and F. Charpillet. Robust dense visual odometry for rgb-d cameras in a dynamic environment. In *2015 International Conference on Advanced Robotics (ICAR)*, pages 1–7. IEEE, 2015.

[38] P. F. Alcantarilla, J. J. Yebes, J. Almazán, and L. M. Bergasa. On combining visual slam and dense scene flow to increase the robustness of localization and mapping in dynamic environments. In *2012 IEEE International Conference on Robotics and Automation*, pages 1290–1297. IEEE, 2012.

[39] Y. Sun, M. Liu, and M. Meng. Improving rgb-d slam in dynamic environments: A motion removal approach. *Robotics and Autonomous Systems*, 89, 11 2016.

[40] Y. Sun, M. Liu, and M. Q.-H. Meng. Motion removal for reliable rgb-d slam in dynamic environments. *Robotics and Autonomous Systems*, 108:115–128, 2018.

[41] Y. Wang and S. Huang. Towards dense moving object segmentation based robust dense rgb-d slam in dynamic scenarios. In *2014 13th International Conference on Control Automation Robotics & Vision (ICARCV)*, pages 1841–1846. IEEE, 2014.

[42] D. Kim and J. Kim. Effective background model-based rgb-d dense visual odometry in a dynamic environment. *IEEE Transactions on Robotics*, 32(6):1565–1573, 2016.

[43] J. Cheng, Y. Sun, W. Chi, C. Wang, H. Cheng, and M. Q.-H. Meng. An accurate localization scheme for mobile robots using optical flow in dynamic environments. In *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 723–728, 2018.

[44] L. Cui and C. Ma. Sof-slam: A semantic visual slam for dynamic environments. *IEEE Access*, 7:166528–166539, 2019.

[45] T. Sun, Y. Sun, M. Liu, and D. Yeung. Movable-object-aware visual slam via weakly supervised semantic segmentation. *ArXiv*, abs/1906.03629, 2019.

[46] F. Zhong, S. Wang, Z. Zhang, C. Chen, and Y. Wang. Detect-slam: Making object detection and slam mutually beneficial. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1001–1010, 2018.

[47] L. Xiao, J. Wang, X. Qiu, Z. Rong, and X. Zou. Dynamic-slam: Semantic monocular visual localization and mapping based on deep learning in dynamic environment. *Robotics and Autonomous Systems*, 117:1–16, 2019.

[48] H. Liu, G. Liu, G. Tian, S. Xin, and Z. Ji. Visual slam based on dynamic object removal. In *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 596–601, 2019.

[49] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[50] S. Yang, G. Fan, L. Bai, C. Zhao, and D. Li. Sgc-vslam: A semantic and geometric constraints vslam for dynamic indoor environments. *Sensors*, 20(8), 2020.

[51] A. Li, J. Wang, M. Xu, and Z. Chen. Dp-slam: A visual slam with moving probability towards dynamic environments. *Information Sciences*, 556:128–142, 2021.

[52] J. Vincent, M. Labb'e, J. Lauzon, F. Grondin, P. Comtois-Rivet, and F. Michaud. Dynamic object tracking and masking for visual slam. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4974–4979, 2020.

[53] T. Ji, C. Wang, and L. Xie. Towards real-time semantic rgb-d slam in dynamic environments. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021.

[54] A. Rosinol, A. Violette, M. Abate, N. Hughes, Y. Chang, J. Shi, A. Gupta, and L. Carlone. Kimera: From slam to spatial perception with 3d dynamic scene graphs. *The International Journal of Robotics Research*, 40(12-14):1510–1546, 2021.

[55] J. McCormac, A. Handa, A. J. Davison, and S. Leutenegger. Semanticfusion: Dense 3d semantic mapping with convolutional neural networks. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4628–4635, 2017.

[56] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison. Elasticfusion: Dense slam without a pose graph. In *Robotics: Science and Systems*, 2015.

[57] R. F Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison. Slam++: Simultaneous localisation and mapping at the level of objects. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1352–1359, 2013.

[58] S. Yang and S. A. Scherer. Cubeslam: Monocular 3-d object slam. *IEEE Transactions on Robotics*, 35:925–938, 2019.

[59] K. Li, H. Rezatofighi, and I. D. Reid. Moltr: Multiple object localization, tracking and reconstruction from monocular rgb videos. *IEEE Robotics and Automation Letters*, 6:3341–3348, 2021.

[60] Z. Qian, K. Patath, J. Fu, and J. Xiao. Semantic slam with autonomous object-level data association. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11203–11209, 2021.

[61] L. Nicholson, M. Milford, and N. Sünderhauf. Quadricslam: Dual quadrics from object detections as landmarks in object-oriented slam. *IEEE Robotics and Automation Letters*, 4(1):1–8, Jan 2019.

[62] A. Sharma, W. Dong, and M. Kaess. Compositional and scalable object slam. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11626–11632, 05 2021.

[63] M. Gonzalez, E. Marchand, A. Kacete, and J. Royan. S3lam: Structured scene slam. *ArXiv*, abs/2109.07339, 2021.

[64] C. Gomez, A. Hernandez, and E. Derner. Object-based pose graph for dynamic indoor environments. *IEEE Robotics and Automation Letters*, 5(4):5401 – 5408, 2020.

[65] A. Walcott-Bryant, M. Kaess, H. Johannsson, and J. J. Leonard. Dynamic pose graph slam: Long-term mapping in low dynamic environments. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1871–1878. IEEE, 2012.

[66] D. M. Rosen, J. Mason, and J. J. Leonard. Towards lifelong feature-based mapping in semi-static environments. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1063–1070, 2016.

[67] M. Zhao, X. Guo, L. Song, B. Qin, X. Shi, G. Lee, and G. Sun. A general framework for lifelong localization and mapping in changing environment. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2021.

[68] N. Bore, J. Ekekrantz, P. Jensfelt, and J. Folkesson. Detection and tracking of general movable objects in large three-dimensional maps. *IEEE Transactions on Robotics*, 35(1):231–247, 2019.

[69] D. Lee and H. Myung. Solution to the slam problem in low dynamic environments using a pose graph and an rgb-d sensor. *Sensors*, 14(7):12467–12496, 2014.

[70] N. Sünderhauf and P. Protzel. Switchable constraints for robust pose graph slam. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1879–1884. IEEE, 2012.

[71] E. Olson and P. Agarwal. Inference on networks of mixtures for robust robot mapping. *The International Journal of Robotics Research*, 32(7):826–840, 2013.

[72] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard. Robust map optimization using dynamic covariance scaling. In *2013 IEEE International Conference on Robotics and Automation*, pages 62–69. Citeseer, 2013.

[73] Z. Hashemifar and K. Dantu. Practical persistence reasoning in visual slam. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7307–7313, 2020.

[74] M. T. Lázaro, R. Capobianco, and G. Grisetti. Efficient long-term mapping in dynamic environments. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 153–160, 2018.

[75] E. Derner, C. Gomez, A. C. Hernandez, R. Barber, and R. Babuska. Change detection using weighted features for image-based localization. *Robotics and Autonomous Systems*, 135:103676, 2021.

[76] L. M. Schmid, J. A. Delmerico, J. L. Schönberger, J. I. Nieto, M. Pollefeys, R. Siegwart, and C. Cadena. Panoptic multi-tsdfs: a flexible representation for online multi-resolution volumetric mapping and long-term dynamic scene consistency. *CoRR*, abs/2109.10165, 2021.

[77] H. Bujanca, X. Shi, M. Spear, P. Zhao, B. Lennox, and M. Luján. Robust slam systems: Are we there yet? In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[78] D. Li, X. Shi, Q. Long, S. Liu, W. Yang, F. Wang, Q. Wei, and F. Qiao. Dxslam: A robust and efficient visual slam system with deep features. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4958–4965, 2020.

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

[80] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580, 2012.

[81] X. Shi, D. Li, P. Zhao, Q. Tian, Y. Tian, Q. Long, C. Zhu, J. Song, F. Qiao, L. Song, Y. Guo, Z. Wang, Y. Zhang, B. Qin, W. Yang, F. Wang, R. H. M. Chan, and Q. She. Are we ready for service robots? the OpenLORIS-Scene datasets for lifelong SLAM. In *2020 International Conference on Robotics and Automation (ICRA)*, pages 3139–3145, 2020.

[82] Y. Choi, N. Kim, S. Hwang, K. Park, J. S. Yoon, K. An, and I. S. Kweon. Kaist multi-spectral day/night dataset for autonomous and assisted driving. *IEEE Transactions on Intelligent Transportation Systems*, 2018.

[83] W. Maddern, G. Pascoe, C. Linegar, and P. Newman. 1 Year, 1000km: The Oxford RobotCar Dataset. *The International Journal of Robotics Research (IJRR)*, 36(1):3–15, 2017.

[84] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[85] T. Peynot, S. Scheding, and S. Terho. The Marulan Data Sets: Multi-Sensor Perception in Natural Environment with Challenging Conditions.

International Journal of Robotics Research, 29(13):1602–1607, November 2010.

[86] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. Leonard. Past, present, and future of simultaneous localization and mapping: Towards the robust-perception age. *IEEE Transactions on Robotics*, 32:1309–1332, 2016.

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

[88] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *Computer Vision – ECCV 2006*, pages 404–417, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

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

[90] M. J. Cummins and P. Newman. Highly scalable appearance-only slam - fab-map 2.0. In *Robotics: Science and Systems*, 2009.

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

[92] M. Labbé and F. Michaud. Appearance-based loop closure detection for online large-scale and long-term operation. *IEEE Transactions on Robotics*, 29(3):734–745, 2013.

[93] F. Dellaert and M. Kaess. Square root sam: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, 25(12):1181–1203, 2006.

[94] M. Kaess, A. Ranganathan, and F. Dellaert. isam: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378, 2008.

[95] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[96] Z. Chen and L. Liu. Navigable space construction from sparse noisy point clouds. *IEEE Robotics and Automation Letters*, 6(3):4720–4727, 2021.

[97] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[98] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, and A. Berg. Ssd: single shot multibox detector. In *Proceedings of the European conference on computer vision*, pages 21–37, 2016.

[99] N. Sünderhauf, T. T. Pham, Y. Latif, M. Milford, and I. D. Reid. Meaningful maps with object-oriented semantic mapping. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5079–5085, 2017.

[100] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee. Yolact: Real-time instance segmentation. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9156–9165, 2019.

[101] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee. Yolact++: Better real-time instance segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2020.

[102] D. de Geus, P. Meletis, and G. Dubbelman. Fast panoptic segmentation network. *IEEE Robotics and Automation Letters*, 5(2):1742–1749, 2020.

[103] J. C. V. Soares, M. Gattass, and M. A. Meggiolaro. Mapping in dynamic environments using deep learning-based human detection. In *25th International Congress of Mechanical Engineering - COBEM*, 2019.

[104] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[105] P. Dendorfer, H. Rezatofighi, A. Milan, J. Shi, D. Cremers, I. D. Reid, S. Roth, K. Schindler, and L. Leal-Taix'e. Mot20: A benchmark for multi object tracking in crowded scenes. *ArXiv*, abs/2003.09003, 2020.

[106] S. Shao, Z. Zhao, B. Li, T. Xiao, G. Yu, X. Zhang, and J. Sun. Crowdhuman: A benchmark for detecting human in a crowd. *arXiv preprint arXiv:1805.00123*, 2018.

[107] J. Redmon. Darknet: Open source neural networks in c, 2016.

[108] A. Milan, L. Leal-Taixé, I. D. Reid, S. Roth, and K. Schindler. Mot16: A benchmark for multi-object tracking. *ArXiv*, abs/1603.00831, 2016.

[109] R. Stiefelhagen, K. Bernardin, R. Bowers, J. Garofolo, D. Mostefa, and P. Soundararajan. The clear 2006 evaluation. In *Multimodal Technologies for Perception of Humans*, pages 1–44, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[110] A. Ess, B. Leibe, K. Schindler, and L. Van Gool. A mobile vision system for robust multi-person tracking. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[111] D. Scaramuzza, L. Spinello, R. Triebel, and R. Siegwart. Key technologies for intelligent and safer cars - from motion estimation to predictive collision avoidance. In *2010 IEEE International Symposium on Industrial Electronics*, pages 2803–2808, 2010.

[112] G. F. Abati, J. C. V. Soares, M. Gattass, and M. A. Meggiolaro. People following system for holonomic robots using an rgb-d sensor. In *26th International Congress of Mechanical Engineering - COBEM 2021*, 11 2021.

[113] Optitrack flex13. `http://https://optitrack.com/cameras/flex-13/`. Accessed: 2022-04-22.

[114] Earthsense. `https://www.earthsense.co/home`. Accessed: 2022-04-22.

[115] X. Li, Y. Du, Z. Zeng, and O. C. Jenkins. Seannet: Semantic understanding network for localization under object dynamics. *ArXiv*, abs/2110.02276, 2021.

[116] K. Doherty, D. Fourie, and J. Leonard. Multimodal semantic slam with probabilistic data association. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2419–2425, 2019.

[117] J. Lamarca, S. Parashar, A. Bartoli, and J. M. M. Montiel. Defslam: Tracking and mapping of deforming scenes from monocular sequences. *arXiv preprint arXiv:1908.08918*, 2019.

[118] B. Bescós, C. Campos, J. D. Tardós, and J. Neira. Dynaslam ii: Tightly-coupled multi-object tracking and slam. *IEEE Robotics and Automation Letters*, 6:5191–5198, 2021.

[119] M. Henein, J. Zhang, R. Mahony, and V. Ila. Dynamic slam: The need for speed. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2123–2129, 2020.

[120] G. Gallego, T. Delbruck, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza. Event-based vision: A survey. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 44(01):154–180, jan 2022.