Pontifícia Universidade Católica
do Rio de Janeiro

Gabriel Fischer Abati

Visual SLAM in Dynamic Environments using Panoptic Segmentation

Dissertação de Mestrado

Dissertation 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 Mestre em Engenharia Mecânica.

Advisor : Prof. Marco Antonio Meggiolaro
Co-advisor: Prof. João Carlos Virgolino Soares

Rio de Janeiro
Maio 2023

Pontifícia Universidade Católica
do Rio de Janeiro

Gabriel Fischer Abati

Visual SLAM in Dynamic Environments using Panoptic Segmentation

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

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

Prof. João Carlos Virgolino Soares
Co-Advisor
Departamento de ciências mecânicas e engenharia - Universidade de Ilinois em Urbana-Champaign

Prof. Alberto Barbosa Raposo
Departamento de Informática - PUC-Rio

Prof. Luiz Carlos Pacheco Rodrigues Velho
IMPA

Prof. Vivian Suzano Medeiros
Departamento de Engenharia Mecânica - PUC-Rio

Rio de Janeiro, Maio 17th, 2023

Bibliographic data

Abati, Gabriel Fischer

Visual SLAM in dynamic environments using panoptic segmentation / Gabriel Fischer Abati ; advisor: Marcos Antonio Meggiolaro ; co-advisor: João Carlos Virgolino Soares. – 2023.
91 f. : il. color. ; 30 cm

Dissertação (mestrado)–Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Mecânica, 2023.
Inclui bibliografia

1. Engenharia Mecânica – Teses. 2. SLAM visual. 3. Segmentação panoptica. 4. Ambientes dinâmicos. I. Meggiolaro, Marcos Antonio. II. Soares, João Carlos Virgolino. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Mecânica. IV. Título.

CDD: 621

Aos meus pais e família
pelo apoio e encorajamento.

Acknowledgments

To my parents, Maria Cristina Fischer Matos, Ismael João de Oliveira Abati and all many family members for all the love, support and guidance.

To my advisor, Professor Marco Antonio Meggiolaro, for the trust, motivation and oportunities for almost a decade. To my co-advisor and friend, Dr João Carlos Virgolino, for the years of mentoring, all competitions and prizes we won at Riobotz and introducing me into academic community.

I also would like to thank all committe members. To Prof. Luiz Carlos Pachecho Rodrigues Velho, Prof. Alberto Barbosa Raposo and Prof. Vivian Suzano Medeiros for acceppting this invitation and interest in the work.

To all my colleagues from LabRob Robotics Lab at PUC-Rio University, specially Anna Ferreira, Paulo Teixeira, Gustavo Duarte, Hilton Santana, Lucas Simões and Emily Alves. Thank you for helping to make LabRob a great academic environment, as well as all the challenging robotic projects and awesome competitions we participated.

To my long time friends, Marco Azeredo, Lucas Miranda, Rafael Canário, Caio Serra, Gabriela Serra, Carolina Mello, Guilherme Couto, João Azeredo, Ana Carla, Luiz Azeredo, Caio Cordeiro, Thales Cavaliere and Clara Malizia. Thank you for bringing me joy through all the tough times.

To all my colleagues from Fu2re Smart Solutions company. Specially to Thiago Cazes, Andre Sih, Rodrigo Ferreira, Carla Brito, Julia Ataide, Raul Sobral, Pedro Asad, Marcellus Alkemin, João Wieland, Carlos Henrique, Daniel Porto, Ricardo Rei, Thiago Matheus Bruno and Helena. Thank you for the opportunity to work with professional deep learning projects and the flexibility to do my research at the same time.

To PUC-Rio and all the staff from the Mechanical Engineering Departament, specially Carina and Simone.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001

Abstract

Fischer Abati, Gabriel; Meggiolaro, Marco Antonio (Advisor); Soares, João Carlos Virgolino (Co-Advisor). **Visual SLAM in Dynamic Environments using Panoptic Segmentation**. Rio de Janeiro, 2023. 91p. Dissertação de Mestrado – Departamento de Engenharia Mecânica, Pontifícia Universidade Católica do Rio de Janeiro.

Mobile robots have become popular in recent years due to their ability to operate autonomously and accomplish tasks that would otherwise be too dangerous, repetitive, or tedious for humans. The robot must have a map of its surroundings and an estimate of its location within this map to achieve full autonomy in navigation. The Simultaneous Localization and Mapping (SLAM) problem is concerned with determining both the map and localization concurrently using sensor measurements. Visual SLAM involves estimating the location and map of a mobile robot using only visual information captured by cameras. Utilizing cameras for sensing provides a significant advantage, as they enable solving computer vision tasks that offer high-level information about the scene, including object detection, segmentation, and recognition. There are several visual SLAM systems in the literature with high accuracy and performance, but the majority of them are not robust in dynamic scenarios. The ones that deal with dynamic content in the scenes usually rely on deep learning-based methods to detect and filter dynamic objects. However, these methods cannot deal with unknown objects. This work presents a new visual SLAM system robust to dynamic environments, even in the presence of unknown moving objects. It uses Panoptic Segmentation to filter dynamic objects from the scene during the state estimation process. The proposed methodology is based on ORB-SLAM3, a state-of-the-art SLAM system for static environments. The implementation was tested using real-world datasets and compared with several systems from the literature, including DynaSLAM, DS-SLAM and SaD-SLAM. Also, the proposed system surpasses ORB-SLAM3 results in a custom dataset composed of dynamic environments with unknown moving objects.

Keywords

Visual SLAM; Panoptic Segmentation; Dynamic Environments.

Resumo

Fischer Abati, Gabriel; Meggiolaro, Marco Antonio; Soares, João Carlos Virgolino. **SLAM visual em ambientes dinâmicos utilizando Segmentação Panóptica**. Rio de Janeiro, 2023. 91p. Dissertação de Mestrado – Departamento de Engenharia Mecânica, Pontifícia Universidade Católica do Rio de Janeiro.

Robôs moveis se tornaram populares nos últimos anos devido a sua capacidade de operar de forma autônoma e performar tarefas que são perigosas, repetitivas ou tediosas para seres humanos. O robô necessita ter um mapa de seus arredores e uma estimativa de sua localização dentro desse mapa para alcançar navegação autônoma. O problema de Localização e Mapeamento Simultâneos (SLAM) está relacionado com a determinação simultânea do mapa e da localização usando medidas de sensores. SLAM visual diz respeito a estimar a localização e o mapa de um robô móvel usando apenas informações visuais capturadas por câmeras. O uso de câmeras para o sensoriamento proporciona uma vantagem significativa, pois permite resolver tarefas de visão computacional que fornecem informações de alto nível sobre a cena, incluindo detecção, segmentação e reconhecimento de objetos. A maioria dos sistemas de SLAM visuais não são robustos a ambientes dinâmicos. Os sistemas que lidam com conteúdo dinâmico normalmente contem com métodos de aprendizado profundo para detectar e filtrar objetos dinâmicos. Existem vários sistemas de SLAM visual na literatura com alta acurácia e desempenho, porem a maioria desses métodos não englobam objetos desconhecidos. Este trabalho apresenta um novo sistema de SLAM visual robusto a ambientes dinâmicos, mesmo na presença de objetos desconhecidos. Este método utiliza segmentação panóptica para filtrar objetos dinâmicos de uma cena durante o processo de estimação de estado. A metodologia proposta é baseada em ORB-SLAM3, um sistema de SLAM estado-da-arte em ambientes estáticos. A implementação foi testada usando dados reais e comparado com diversos sistemas da literatura, incluindo DynaSLAM, DS-SLAM e SaD-SLAM. Além disso, o sistema proposto supera os resultados do ORB-SLAM3 em um conjunto de dados personalizado composto por ambientes dinâmicos e objetos desconhecidos em movimento.

Palavras-chave

SLAM Visual; Segmentação Panoptica; Ambientes dinâmicos.

Table of contents

List of figures

List of tables

List of algorithms

List of Abreviations

SLAM – Simultaneous Localization and Mapping

MOT – Multi Object Tracking

RANSAC – Random Sample Consensus

COCO – Common Objects in Context

DL – Deep Learning

EKF – Extended Kalman Filter

TSDF – truncated signed distance function

DNN – Deep Neural Network

SSD – Single Shot Multibox Object Detector

RCNN – Recurrent Convolution Neural Network

IoU – Intersection Over Union

YOLO – You Only Look Once

RPN – Region Proposal Network

SVD – Singular Value Decomposition

RGB – Red Green Blue

FPN – feature Pyramid Network

GPU – Graphical Processor Unit

ATE –Absolute Trajectory Error

Eu não procuro saber as respostas, procuro compreender as perguntas.

Confúcio, .

1
Introduction

Robotics and automation are topics that have been widely studied in the last 30 years. These topics are revolutionizing how machines can interpret and interact with the world and having huge impact in applications in the field of medicine, aerospace, manufacturing industry, telecommunications and others. Most industrial robots have movement constrains and can only work in a limited area. On the other hand, mobile robots are capable to move freely inside an environment. They can be applied to tasks that are too risky or impossible for humans to safely perform. For instance, self-driving, house-cleaning, food delivery, nursery, surveillance and tourism are some complex applications that can benefit from the mobile robots' versatility.

There are four main problems that allow robots to achieve full autonomy: perception, localization and mapping, path planning and motion control. Figure 1.1 shows how these problems are related to each other.

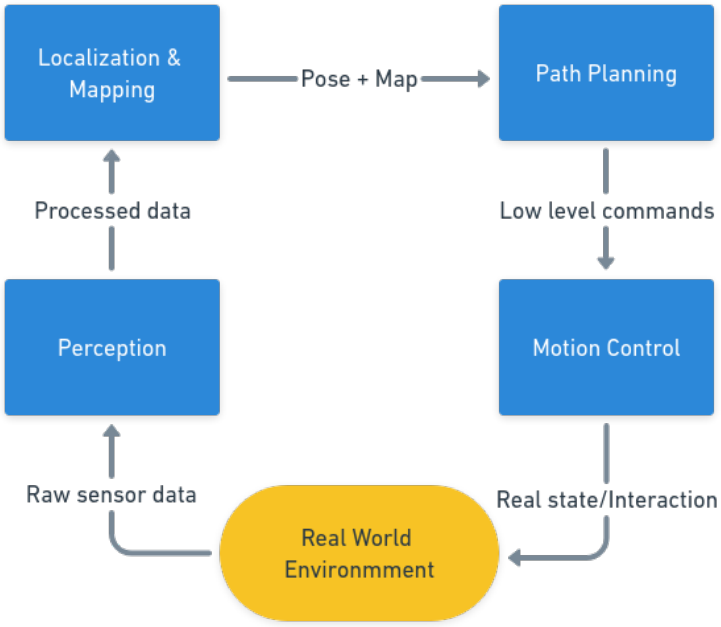


Figure 1.1: Diagram of the tasks performed by an autonomous mobile robot

The perception task is responsible for the initial raw data acquisition and processing. In practice, dedicated sensors are equipped to a robot to produce data about its surroundings that will be later processed to interpret it with high level algorithms. For instance, the creation of a 3D point cloud from RGB-D images. For indoor applications, normally, the robot does not have any information about the environment nor GPS data available.

In order to perform autonomous navigation, a robot needs two main elements: a map and an estimation of its pose. The map is a representation of the environment that can be created by sensor measurements or supplied to the robot *a priori*, and the pose (position and orientation) of the robot is estimated with respect to the global coordinate system. However, usually neither of the elements are provided to the robot *a priori*, and both need to be estimated using only sensor data. This is a fundamental problem in robotics, and it is known as Simultaneous Localization and Mapping (SLAM). By performing localization and mapping simultaneously, errors in both tasks can be reduced. The map can be updated based on new measurements, which can then be used to improve the accuracy of the localization estimate. Similarly, accurate localization can be used to correct errors in the map. After the map construction and pose estimation, the robot can autonomously navigate a path inside the environment and reach its final destination. This is one of the most challenging tasks in robotics due to unpredictable events, noisy sensor measurements and movement uncertainty. Finally, the motion control analyses the robot's kinematics and low-level characteristics.

The choice of sensors has a high influence on the problem formulation, as well as on its outcome. For SLAM systems, the laser range finder, inertial measurement units and wheel encoders are commonly used sensors, according to the literature. Nonetheless, cameras can be a good choice on account of its low cost and abundance of information. RGB-D cameras or depth sensors are a special kind of camera that can provide both geometric and semantic data about the environment.

Visual SLAM systems solve the SLAM problem using only cameras. They are receiving increasing attention in the robotics community due to the low cost and richness of information provided by cameras. There are several visual SLAM systems in the literature with high precision using monocular [12][13], stereo [14] and RGB-D cameras [15]. However, these SLAM systems are not prepared to work in scenarios with moving objects, which results in inaccurate localization and inconsistent mapping. To address this, there are several approaches to incorporate dynamic elements into Visual SLAM. Recently, deep learning-based techniques have been explored for SLAM in dynamic environments for providing high-level information about the scenes. Some methods [16][17] use object detection, such as YOLO [18] to detect and track labeled objects, filtering the keypoints of the dynamic ones. Others [19][20] use semantic segmentation combined with epipolar geometry to filter dynamic keypoints. Other methods [21][22][23] rely on instance segmentation techniques, such as Mask R-CNN [7] or YOLACT++ [24].

The main problem with all previous approaches is the necessity to have a pre-determined number of classes that can be detected, and consequently, filtered. In other words, if a non-labeled object is moving in the scene, it would not be filtered and its features would cause a drift in the localization and outliers in the map. The main objective of this work is to explore the richness of information provided by RGB data, using deep learning techniques to allow robust SLAM in dynamic environments in the presence of unknown and unlabeled moving objects.

1.1 Related Work

1.1.1 SLAM

The challenge in SLAM is to place a mobile robot in an unknown environment with an unknown pose and have it gradually built a map of the environment and simultaneously determines where the robot lies within the created map. Several methods have been proposed to solve SLAM. Bailey *et al.* [25] made one of the earliest surveys about this problem. SLAM methods can be characterized in terms of map type (topological or metric), and in terms of data use (feature-based or raw). All methods can be traced back to a probabilistic interpretation, due to the uncertainties inherent to sensor measurements and unstructured environments. There are three main probabilistic formulations for SLAM: extended Kalman filters, particle filters, and graph-based approaches [26].

EKF-SLAM algorithms such as [27] are non-linear SLAM systems predominantly implemented as an extended Kalman Filter. EKF is a non-linear state estimator that considers system noise as a zero mean Gaussian distribution for motion and observations. Several publications highlight the EKF-SLAM problems [28][29][30]: the model linearization does not match the true model, the true system noise is non-Gaussian, the estimated uncertainty is too small compared to the real error and eventual inconsistencies for large-scale maps due to the necessity of state covariance matrix inversion.

To achieve better performance, Stentz *et al.* presented FastSLAM [31]. This Baysian algorithm uses a Rao-Blackwellized particle filter [32] to solve the SLAM problem. The particle filter is an extension of the Monte Carlo Localization algorithm. It uses a set of weighted particles, sampled by a motion model distribution, to represent several hypotheses about the state of the robot. Each particle is associated with a landmark location in the

map [33][34]. The use of the particle filter makes it possible to make all the landmarks independent for a given particle (this is the factorization of the SLAM conditionally to the trajectory). Thus, the FastSLAM algorithm guarantees a much lower execution time than EKF-SLAM. On the other hand, this algorithm is not consistent due to the phenomenon of particle degeneration [35], which refers to gradual loss of particle diversity and resulting in accuracy loss.

Unlike EKF-SLAM or FastSLAM, which process the information at time t only using the measurements at time t and the state at time $t-1$, graph-based approaches solve the SLAM problem on the whole trajectory and are currently the standard formulation for SLAM. GraphSLAM [36] extracts from all measurements a set of soft constrains, represented by a sparse graph. This graph leads to a sum of nonlinear quadratic constraints. Optimizing these constraints yields a maximum likelihood map and a corresponding set of robot poses. Due to the graph-optimization, GraphSLAM can handle a large number of features and incorporate landmark information in large scale mapping processes.

1.1.2 Visual SLAM

A visual SLAM system with a graph-based approach has three main steps: motion estimation, loop closure and graph optimization. Motion estimation is a process of determining the spacial transformation from one 2D image to another that describes the robot's movement. Loop closure detection is used to identify revisited places in order to estimate the accumulated errors during the motion estimation task. Finally, these errors are then rectified by the graph optimization algorithm.

These systems can be classified by how they reconstruct the environment. The methods can be either sparse or dense. Sparse methods use a set of individual points to reconstruct the scene, as the dense methods uses all pixels available.

Another common classification of visual SLAM systems refers to how information is extracted from input images. Visual SLAM methods can be either direct or indirect. Direct methods use pixel intensity data to estimate the photometric error and minimize it directly to estimate the camera pose and the 3D scene. For this reason, they deal better with environments with low amounts of unique features. Indirect methods, on the other hand, utilize an intermediate representation to have a faster performance. They first extract a sparse set of keypoints to match with other frames in order to estimate the

camera pose and reconstruct the scene. They deal better with geometric noise from the system.

VOLDOR-SLAM [37] is an example of a dense-indirect method that uses a dense Optical Flow [38] technique as input and utilizes a depth map alignment framework. DSO [15] is a sparse-indirect method that fuses photometric error minimization and joint optimization of geometric and camera motion to acquire visual odometry. LSD-SLAM [13] is a dense-direct method for monocular cameras that uses visual tracking algorithms that can reconstruct large-scale maps through scale-drifts detection. Moreover, KinectFusion [39] uses Iterative Closest Point algorithm [40] to track camera pose and depth information to create dense volumetric representations.

In visual SLAM, the sparse-indirect methods are more usual than the others presented. Most methods use visual features to create the map and perform camera tracking. Mono-SLAM's [41] key concept is a probabilistic feature-based map. This map represents the current estimates, including their uncertainties, of the camera's state and the features of interest. The disadvantage of this technique is the need to process every frame in order to map and estimate camera pose. PTAM [42], on the other hand, is a keyframe-based approach, which replace incremental mapping with a frame batch method, i.e, bundle adjustment, to obtain globally consistent estimates over a frame sequence. One of the main contributions of PTAM was the division of camera tracking and mapping processes in parallel threads to reduce the computational cost.

There are also SLAM systems that utilize RGB-D sensors and have the advantage to directly measure depth. Examples of RGB-D SLAM systems include RGB-D mapping [43], RGBDSLAM [44] and RTAB-Map [45].

ORB-SLAM [12] is an open-source system that works with three parallel threads: tracking, loop closure and local mapping. The tracking thread uses ORB visual features [46] to track the camera frame by frame, as well as, to perform camera re-localization in the event of a lost track. The loop closure process searchers for previously visited places using a robust place recognition algorithm for every new keyframe. The local mapping thread performs bundle adjustment for each new keyframe added. Due to its high accuracy, robustness, scalability and real time performance in large environments, ORB-SLAM is one of the most referenced papers about visual SLAM in the literature.

ORB-SLAM2 [14] is a continuation of ORB-SLAM. It has the same three-thread structure from its predecessor, but works using different camera types, such as monocular, stereo or RGB-D. In 2021, ORB-SLAM3 [9] was presented, with IMU integration and improving recovery from lost track

by applying Atlas, a visual-inertial multi-mapping system. Atlas was first introduced in [47] and received improvements in ORB-SLAM3 in order to be able to effectively represent a set of disconnected maps and applying mapping operations smoothly across all of them. Since all three versions of ORB-SLAM are open-source, they are used as framework to other SLAM systems. Although ORB-SLAM is a state-of-the-art SLAM method, it still has poor accuracy when it is used in dynamic environments.

Most visual SLAM systems are designed with the assumption of a static environment, which limits their applicability in real-world scenarios. When a robot is performing SLAM in a dynamic environment, the presence of moving objects in the scene not only compromises camera tracking but also loop detection and mapping. Figure 1.2 shows an example of a point cloud map generated from the output of ORB-SLAM2, representing an environment with moving people in the scene. The movement of these objects has resulted in a corrupted map.

Figure 1.2: Point cloud map of a dynamic scene [1] generated from the output of ORB-SLAM2. The people populating the map with outliers, corrupting the camera pose estimation.

1.1.3 Visual SLAM in Dynamic Environments

Traditional SLAM system struggle in dynamic environments, where objects can move, change sizes and occlude each other. These moving objects lead to inaccuracies in the map and pose estimates. Most researchers treat dynamic objects as outliers, and several visual SLAM systems have been proposed to address these outliers. These solutions can be broadly categorized into two main groups: geometry-based and learning-based methods. Geometry-based methods rely on classic computer vision techniques to detect and filter dynamic content. In general, they have inferior accuracy compared to learning-

based methods. Their main advantage is to not require prior knowledge about the objects in the scene.

In the work from Dib et al. [48], the authors propose a dense visual SLAM system based on the Lucas-Kanade [49] image alignment algorithm for RGB-D sensors and RANSAC [50] optimization for outlier rejection. In the same context, Alcantarilla *et al.* [51] presented a dense scene representation that combines stereo vision, optical flow and motion likelihoods to detect moving objects. Another example of a dense method is from Sun *et al.* [52]. Their system performs motion removal by analyzing consecutive images with ego-motion, compensated frame differencing, particle filter-based tracking and Maximum-a-posteriori estimation.

ReFusion [53] and StaticFusion [54] are direct methods for SLAM in dynamic environments using RGB-D cameras. ReFusion apply direct tracking on a truncated signed distance function (TSDF) in a voxel hashing representation, along with encoded color information, to estimate the camera pose. StaticFusion focuses on background segmentation and dynamic objects filtering by segmenting every RGB and depth pair images into geometric clusters. Since direct methods use all pixels from the camera frame, they are less robust in dynamic environments due to its high sensitivity to moving objects, in comparison with feature-based methods.

1.1.4 Visual SLAM in Dynamic Environments using Deep learning

Previous methods rely only on classical computer vision algorithms to detect and filter dynamic content. They estimate movement using mainly geometric approach. Deep learning techniques, on the other hand, are able to learn and identify complex objects in the scene such as people, tables, chairs by their geometric and semantic information.

Detect-SLAM [55] was the first work to combine an ORB-SLAM based system with a DNN detector such as SSD [56] only on keyframes in order to perform faster and to improve robustness of the system in dynamic environments. The Dynamic SLAM [57] system also uses SSD object detection to filter dynamic features and includes a semantic correction module that generates a mask with the same size as the image to distinguish between static and dynamic points. The system also has a selective tracking algorithm to eliminate dynamic objects.

Crowd-SLAM [16], based on ORB-SLAM2, uses a custom YOLOV3-Tiny [58] object detection model to locate people in the image and remove dynamic features inside the predicted bounding boxes. Furthermore, the

authors developed a feature repopulation algorithm to reduce lost tracks caused by feature depletion.

DS-SLAM [19], combines the SegNet [59] semantic segmentation network with a moving consistency check algorithm based on optical flow to detect and reduce the impact of dynamic objects, and generates a dense semantic octo-tree map from the environment.

RDS-SLAM [20] is a method based on ORB-SLAM3 that includes a SegNet segmentation parallel thread. This enables the tracking process to operate continuously without waiting for new semantic information. Furthermore, the authors introduced a keyframe selection strategy for semantic segmentation that aims to acquire the most recent semantic information possible, regardless of the segmentation method's speed. To update and propagate semantic information, they utilized the moving probability to detect and eliminate tracking outliers through a data association algorithm. Following a similar approach, SOF-SLAM [60] is based on the ORB-SLAM2 system, that combines SegNet semantic segmentation with optical flow and epipolar geometry constrains to detect dynamic features.

None of the previously cited methods can deal with unknown moving objects. Ji *et al.* [10] presented a Semantic RGB-D SLAM approach that operates in dynamic environments by extracting semantic information solely from keyframes. Despite being able to deal with unknown objects using k-means and depth reprojection, their accuracy is lower than other methods, such as DynaSLAM, in environments with people. Moreover, the authors have only tested their system with unknown moving objects in a single experiment, without using publicly available datasets.

DynaSLAM [21] is one of the first works to employs the Mask R-CNN [7] instance segmentation framework to identify people in the scene at a pixel-level, which it then uses to filter dynamic features. It is one of the visual SLAM systems with the best accuracy in the TUM benchmark dataset [61].

Yuan and Chen proposed SaD-SLAM [22], which integrates depth information and Mask R-CNN instance segmentation to identify dynamic features in images. The algorithm classifies each feature point as static, dynamic, or static and movable. SaD-SLAM exhibits high accuracy, surpassing DynaSLAM in certain scenarios. However, its main limitation is that the semantic segmentation is conducted offline. DOTMask [23], introduced by Vincent *et al.*, employs instance segmentation to obtain pixel-wise information about objects in an image, and an Extended Kalman Filter to track them. The authors aimed to develop a faster SLAM system at the expense of lower accuracy when compared to other approaches. The main problem of instance segmentation-based

approaches for visual pose estimation is the lack of background information, which decreases the possibility to infer about the existence of unknown moving objects.

1.1.5 Visual SLAM with Panoptic Segmentation

Panoptic segmentation [62] is a computer vision task that combines both instance segmentation and semantic segmentation. In semantic segmentation, each pixel in the image is labeled with a semantic category, without distinction to the number of objects in the same class. In instance segmentation, only distinct objects in the image are identified and segmented at a pixel level. The goal of panoptic segmentation is to unify these two tasks into a single one, where all pixels in the image are labeled with either an instance label, indicating which object it belongs to, or a semantic label, indicating which category it belongs to. This technique allows SLAM systems to have a better scene understanding on which pixels belongs to background or objects, resulting in a more precise information about moving objects.

Recently, there has been an increasing use of Panoptic Segmentation in visual state estimation systems. PVO [63] is a monocular SLAM framework that introduces a video panoptic segmentation module to enhance visual odometry capabilities. This module leverages depth, camera pose, and optical flow to improve segmentation accuracy by linking the segmentation results of the current frame with adjacent frames. SVG-LOOP [64] presents a loop-closure detection algorithm that uses a combination of a semantic bag-of-words model processed with panoptic segmentation to minimize the impact of dynamic features, and a semantic landmark vector model to encode the geometric connections within the semantic graph.

Zhu *et al.* [11] proposed a method based on ORB-SLAM2 that incorporates panoptic segmentation and geometry information. To minimize the impact of unknown moving objects, they propose a dynamic object classification approach based on epipolar geometry. Despite claiming robustness against unknown moving objects, this was not numerically evaluated in their paper. Also, they eliminate *a priori* all keypoints belonging to what they consider highly dynamic, without considering known movable objects that can be static (vehicles, for instance).

Finally, none of the mentioned methods could deal with unknown moving objects. The exceptions are Zhu *et al.* [11], that did not provide a quantitative analysis of the claim, and Ji *et al.* [10], that only tested their system in a single experiment with a proprietary custom dataset. Unknown objects are pixel

regions that the model could not predict any label. This happens due to objects that were not trained by the model, motion blur or illumination conditions. Since these regions can contain large information about the frame, they must be addressed to improve pose estimation and overall mapping results.

1.2 Contributions

This master thesis presents a visual SLAM for dynamic environments that uses Panoptic Segmentation to detect and filter moving objects. The contributions of the paper can be summarized as follows.

- The first open-source visual SLAM system that uses Panoptic Segmentation
- A new and robust method to deal with unknown moving objects
- Tests using datasets that explicitly have unknown dynamic objects in the scene, obtaining the best results amongst other systems in several sequences
- The method can be used with monocular, stereo or RGB-D cameras

1.3 Outline

This thesis is divided into 6 chapters that are structured as follows. Chapter 2 explains the computer vision algorithms and techniques relevant to the work, Chapter 3 details the structure of a feature-based SLAM system with graph optimization, Chapter 4 presents the proposed methodology, Chapter 5 displays the results of the proposed methodology using real world datasets from the literature, and Chapter 6 shows the conclusions and suggestions for future work.

2
Computer Vision Techniques

2.1
Introduction

This Chapter explains the basic computer vision techniques to understand how the camera sensor extract information from the environment. The pinhole camera model and intrinsic camera parameters are described in Section 2.2. The Section 2.3 show the methods used to estimate the camera parameters, while Sections 2.4 to 2.7 present the mathematical formulation and techniques to analyze image content and associate multiple 2D camera frames from 3D scenes. Moreover, Section 2.8 describes the most common tasks in digital image analysis and the recent panoptic segmentation technique.

2.2
Camera Model formulation

The pinhole camera model, shown in Fig. 2.1, is a mathematical model that describes the relationship between the three-dimensional (3D) world and the two-dimensional (2D) image captured by a camera. The model assumes that light rays from each point in the 3D world pass through a single point in space called the camera center, and then pass through a small aperture or pinhole, forming an inverted image on the opposite side of the camera. The image is then captured on a flat image sensor or film plane located at a fixed distance from the pinhole.

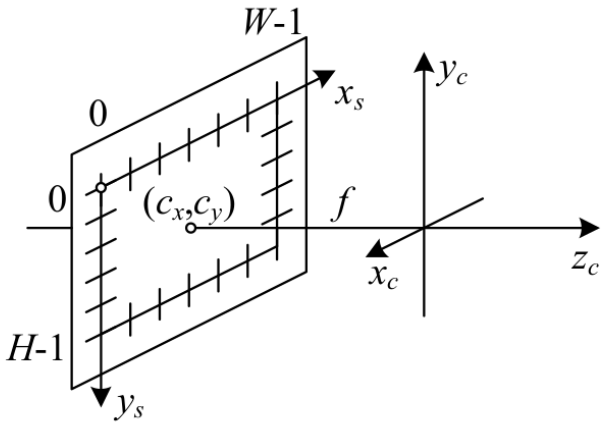


Figure 2.1: Pinhole Camera Model [2].

The model can be represented mathematically using homogeneous coordinates. The camera matrix, also known as the projection matrix, maps 3D points in space to their corresponding 2D image points. It can be computed using the intrinsic camera parameters, which include the focal length, the principal point, and the skew coefficient, as well as the extrinsic camera parameters, which describe the position and orientation of the camera in the world coordinate system. The image point coordinates can by computed as

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

where $[X, Y, Z]^T$ are the coordinates of the point in space, $[u, v, 1]^T$ are the coordinates of the mapped point in the image plane, f is the camera's focal length, and c_x and c_y are the optical center coordinates.

RGB-D or depth cameras, are sensors that capture both color (RGB) and depth information of a scene. They work by projecting a pattern of infrared light onto the scene and then using the time-of-flight or structured light method to calculate the distance between the camera and each point in the scene. This produces a 3D point cloud representation of the scene, where each point has X,Y and Z coordinates

$$X = \frac{u - c_x}{f_x} z \tag{2-2}$$

$$Y = \frac{v - c_y}{f_y} z \tag{2-3}$$

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

where Z is the depth measured by the camera for each pixel position. In general the camera lens produces a circular image, and the magnification and angle of view are the same in all directions, i.e., the focal length is assumed to be the same in both x and y directions $f_x = f_y = f$. Figure 2.2 shows an example of RGB-D camera output with the RGB image and its associated depth map.

This work uses the pinhole camera model for the camera measurements in the SLAM formulation. This model provides SLAM systems with a simplified and accurate representation of how a camera captures image data. Also, the pinhole model is relatively simple to calibrate, since it only requires knowledge of a few intrinsic camera parameters.

(a) RGB Image (b) Depth Map

Figure 2.2: RGB-D Camera Output

2.3 Camera Calibration

Camera calibration involves estimating the intrinsic and extrinsic parameters of a camera to obtain accurate measurements from images. The intrinsic parameters include the focal length, principal point, and distortion coefficients as explained in Sec. 2.2, while the extrinsic parameters define the position and orientation of the camera relative to the world coordinate system. There are different methods for camera calibration, including the calibration plate method, which uses a planar target with known geometry and known positions in 3D space, and the Zhang method [65], which uses multiple images of a planar target (checkerboard) with different orientations. The checkerboard have a known pattern that is used to detect feature points across all images. Then, the parameters are estimated using the correspondence between the 2D image points and their corresponding 3D points on the calibration pattern. Once the intrinsic and extrinsic parameters are estimated, they can be used to correct distortions in the image, perform accurate measurements, and reconstruct 3D scenes from multiple images. Fig 2.3 shows an example of checkerboard image with known size used to estimate camera parameters.

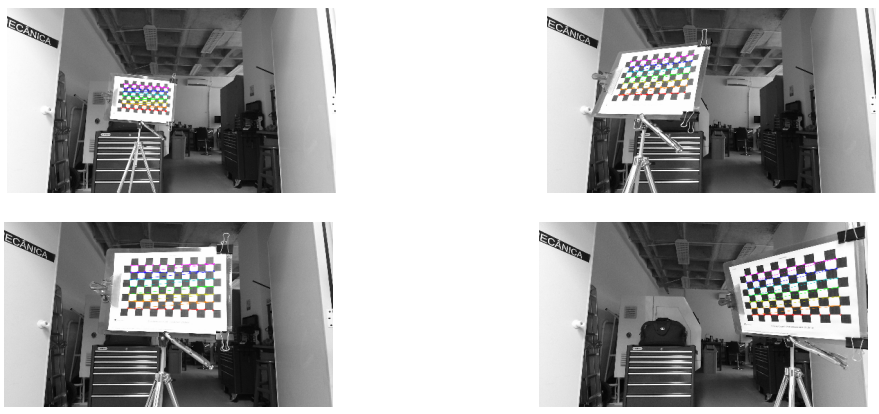


Figure 2.3: Zhang camera calibration method

2.4 Visual Features

Feature detection refers to the process of identifying keypoints or regions in an image that can be used for further analysis or processing. These features are typically defined as local regions of an image that have distinct visual characteristics, such as edges, corners, or blobs. There are various algorithms and techniques for feature detection in computer vision, including Harris corner detection [66], SIFT [67], SURF [68], FAST [69] and ORB [46]. Feature detection algorithms are typically used in conjunction with feature descriptors and matching techniques to enable computer vision systems to recognize and track objects in real-world scenarios.

The Harris corner algorithm computes a corner response function at every pixel location in an image and detects corners based on their response values. Corners are regions where there are significant variations in image intensity in multiple directions. SIFT (Scale-invariant feature transform) detects feature points that are invariant to scale, rotation, and affine transformations. It uses a difference-of-Gaussian (DoG) filter to detect potential interest points and then applies a series of operations to generate descriptors that are used for matching. SURF (Speeded-up robust feature) is a faster alternative to SIFT and detects feature points using a similar approach, but with the use of box filters and an approximation of the Laplacian of Gaussian (LoG) filter. FAST (Features from Accelerated Segment Test) is designed for real-time applications and detects feature points based on the presence of contiguous pixel values that exceed a threshold in a circular pattern.

ORB is a fast and efficient algorithm that uses a combination of the FAST keypoint detector and the BRIEF (Binary Robust Independent Elementary Features) descriptor. The keypoint detector identifies points in an image that are distinct and repeatable, while the descriptor is used to represent these points in a compact and efficient manner. One of the main advantages of ORB is its ability to detect and describe features with high repeatability, even under significant variations in scale, orientation, and lighting conditions. It is also robust to occlusion and clutter in the scene. Fig 2.4 shows ORB feature detection performed by ORB-SLAM3.

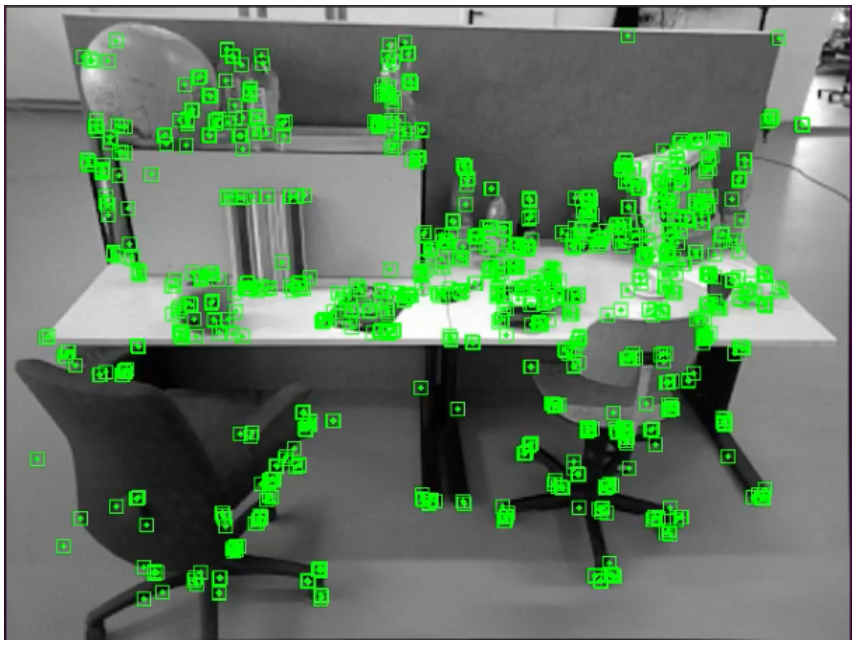


Figure 2.4: Feature detection performed by ORB-SLAM3

2.5 Feature Matching

After the keypoint detection, it is necessary to determine which features correspond to certain locations in different images. A feature descriptor is designed to capture local information about an image, such as color, texture, and shape, that can be used to distinguish one object from another. Feature descriptors are typically computed from regions of interest, such as corners or interest points. Once the regions of interest have been identified, feature descriptors are computed by applying mathematical operations, such as convolution or gradient calculations, to the pixel values within the region.

ORB uses the BRIEF descriptor to generate a binary code that represents the intensity comparison between pairs of pixels in a small patch around a feature point. The patch is typically a square with a fixed size and centered on the feature point. The pixel pairs used for the comparison are randomly selected from within the patch. In order to generate the binary code, each pair of pixels is compared, and the result of the comparison is recorded as a binary value of 0 or 1. The comparison is based on whether the intensity of the first pixel is greater than that of the second pixel. The resulting binary string is used to represent the feature point and can be compared with other feature points using a simple Hamming distance metric.

Once features and their descriptors are extracted from two or more images, the next step is to establish some preliminary feature matches between these images. The goal of feature matching is to find a set of corresponding

features that can be used to align and merge the images. One popular algorithm for feature matching is the k-nearest neighbors (KNN) algorithm.

In the KNN algorithm, each feature in one image is compared with all the features in the other image to find the K closest matches. The distance between the feature vectors is computed using a distance metric, such as Euclidean distance or Hamming distance. The K closest matches are then selected based on their distances.

Once the K nearest matches have been found, a threshold is applied to reject matches that are not sufficiently close. This threshold is usually set based on the ratio of the distance to the first and second nearest neighbors. If this ratio is above a certain threshold, the match is rejected as ambiguous. Fig 2.5 show an example of two consecutive camera frames with their respective keypoints and their association.

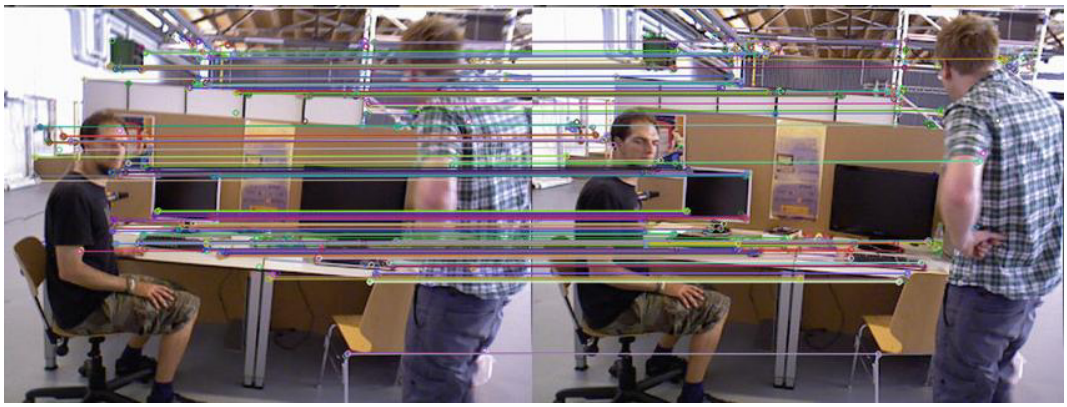


Figure 2.5: Feature Matching

2.6 Epipolar Geometry

Epipolar geometry is the relationship between two views of a scene captured by two cameras. It describes how the geometry of the cameras, the 3D structure of the scene, and the 2D projection of the scene into the cameras are related. Specifically, epipolar geometry is concerned with the relationship between corresponding points in the two views, known as epipolar points. An epipolar line is the line connecting the epipole (the intersection of the line connecting the camera centers with the image plane) to an epipolar point.

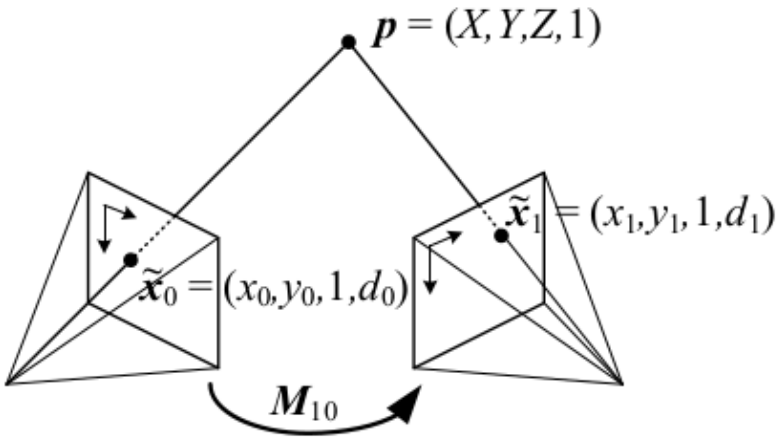


Figure 2.6: Epipolar geometry [2] schematic.

The fundamental matrix is a mathematical representation of the epipolar geometry. There are several methods to calculate the fundamental matrix from a set of corresponding points. One common method is the 8-point algorithm. This algorithm first normalizes the corresponding points to reduce the effects of scaling and distortion. Then it requires at least 8 corresponding points in the two views and finds the fundamental matrix by solving a set of linear equations. However, the resulting matrix may not satisfy the constraints of the epipolar geometry, so additional steps are necessary to enforce these constraints.

2.7 RANSAC

A more advanced method is RANSAC [50]. The basic idea behind RANSAC is to randomly sample a subset of data points from the dataset and fit a model to this subset. The model is then tested on the remaining data points, and the number of inliers (data points that fit the model) is counted. The inliers are obtained using Least-Median of Squares method [70]. If the number of inliers exceeds a certain threshold, the subset is considered a good fit and the model is re-estimated using all the inliers. This process is repeated multiple times, and the best model (with the highest number of inliers) is selected as the final model.

RANSAC is particularly useful for estimating the fundamental matrix because it is robust to outliers in the corresponding point pairs. Outliers can occur due to noise or errors in the feature detection and matching process. By randomly selecting subsets of corresponding point pairs and iteratively refining the fundamental matrix, RANSAC can effectively filter out the outliers and provide a robust estimate of the fundamental matrix.

In order to increase the probability of finding a true set of inliers through random sampling, a sufficient number of trials (S) must be conducted. Let p denote the probability of any given correspondence being valid and let P represent the total probability of success after S trials. The likelihood of all k random samples being inliers in one trial is given by p raised to the power of k. Therefore, the likelihood of all S trials failing is given by the complement of the probability that at least one trial succeeds, which is 1-P.

$$1 - P = (1 - p^k)^S \tag{2-5}$$

The required minimum number of trials is

$$S = \frac{log(1 - P)}{log(1 - p^k)} \tag{2-6}$$

Stewart [70] provides examples of the number of trials required to achieve a 99% success probability in the fundamental matrix problem. The table 2.1 shows that the required number of trials increases rapidly with the number of sample points used. This fact motivates practitioners to use the smallest possible number of sample points, k, for each trial when using RANSAC in practice.

Table 2.1: Number of trials S to attain a 99% success probability

k	p	S
3	0.5	35
6	0.6	97
6	0.5	293

2.7.1 Fundamental Matrix with RANSAC

Here, we demonstrate how to calculate the fundamental matrix with an example from two consecutive images. First, we apply the ORB feature extractor in both images to obtain the keypoints and descriptors. Then, the fundamental matrix is used to estimate the mapping of points in one image to lines in another. It can be defined as $(x')^T = Fx$ or

$$\begin{bmatrix} u' & v' & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = 0 \tag{2-7}$$

To solve this system of homogeneous linear equations, a Singular Value Decomposition (SVD) can be applied to extract the solution F by taking the

row of V corresponding to the smallest singular value. One equation is required to constrain the solution for each unknown variable. Following the 8-point algorithm, eight keypoint pairs are randomly sampled to solve the system. The estimated F has full rank, while the fundamental matrix is a rank 2 matrix, requiring to reduce its rank. One way to achieve this is by decomposing F using singular value decomposition into matrices.

$$F = U\Sigma V' \tag{2-8}$$

The fundamental matrix can be improved by normalizing the keypoints coordinates before computation. The normalization consists in a linear transformation to make the keypoints follow a standard Gaussian distribution.

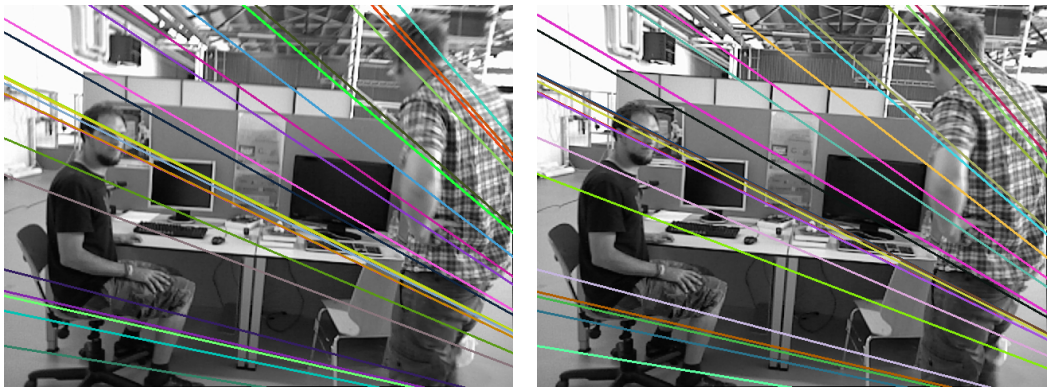
$$Point_{norm} = (T_{scale}) * (T_{offset}) * Point \tag{2-9}$$

$$\begin{bmatrix} u_{norm} \\ v_{norm} \\ 1 \end{bmatrix} = \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -c_u \\ 0 & 1 & -c_v \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{2-10}$$

Where c_u and c_v are the mean coordinates and s is the reciprocal of the standard deviation. The normalized fundamental matrix can be calculated with $T_a = T_{scale}T_{offset}$ from a set of 2D keypoints and $T_b = T_{scale}T_{offset}$ from another set of points.

$$F_{orig} = T_b^T F_{norm} T_a \tag{2-11}$$

Algorithm 1 summarizes all steps to calculate the fundamental matrix using RANSAC. Furthermore, Fig. 2.7 show the resulting epipolar lines from two consecutive images and its fundamental matrix. Since the time interval between the two images is very small, the epipolar lines are similar.



(a) Image Inliners from Image at time t-1 (b) Image Inliners from Image at time t

Figure 2.7: Epipolar lines generate by RANSAC algorithm

Algorithm 1: Fundamental Matrix with RANSAC

Data: Keypoints Kp_t, from current frame
Data: Keypoints Kp_{t-1}, from last frame
Data: N_{it} number of iterations

Kp_t^* , Kp_{t-1}^* , M = FeatureMatching(Kp_t,Kp_{t-1});
T_a , T_b = normalization(Kp_t^* , Kp_{t-1}^*);
bestInliners = 0;
bestModel = NULL;
for *i=1:*N_{it} **do**
    Kp_t^* , Kp_{t-1}^* = Sample_8_pairs(Kp_t^* , Kp_{t-1}^*);
    F = SVD(Kp_t^* , Kp_{t-1}^*);
    F_n=norm(F , T_a , T_b);
    nInliers=0;
    for *j=1:M* **do**
        if $/x_{2j}^T \; F_n \; x_{1j}/ < \epsilon$ **then**
            nInliers = nInliers + 1;
        end
    end
    if *nInliers* > *bestInliers* **then**
        bestInliers = nInliers;
        bestModel = F_n;
    end
end

$$F = \begin{bmatrix} 3.74 * 10^{-7} & 2.55 * 10^{-4} & -1.55 * 10^{-1} \\ -2.53 * 10^{-4} & 3.07 * 10^{-6} & 2.51 * 10^{-1} \\ 1.49 * 10^{-1} & -2.54 * 10^{-1} & 1 \end{bmatrix} \tag{2-12}$$

2.8 Deep Learning

Classical programming, also known as rule-based programming, involves writing a set of instructions (rules) that a computer follows to solve a problem. These rules are based on a programmer's understanding of the problem and are often hand-coded. Classical programming requires a deep understanding of the problem domain, and the rules must be updated manually if the problem changes or if new data becomes available.

Machine learning, on the other hand, is an approach to problem-solving that involves training a computer to learn patterns from data and make predictions based on that data. Machine learning techniques are designed to

automatically improve their accuracy over time as they are fed with more data.

Another key difference between classical programming and machine learning is their performance in dealing with complex and dynamic problems. Classical programming may struggle when faced with highly complex problems or those that involve a large amount of data. Machine learning, on the other hand, can handle such problems with ease, as it can automatically identify patterns that would be difficult or impossible for a human to detect. Fig. 2.8 shows a comparison between classical programming and machine learning.

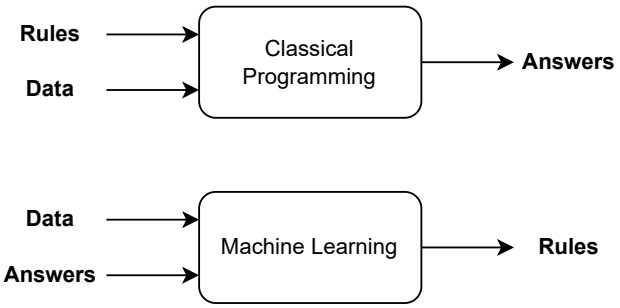


Figure 2.8: Comparison between classical programming and machine learning

Deep learning is a subfield of machine learning that uses deep neural networks to learn and extract features from data. One of the key advantages of deep learning is its ability to learn representations directly from unstructured data, such as images, audio and text. This has led to significant improvements in accuracy and robustness in many computer vision tasks. Deep Learning differs from Machine Learning primarily due to its ability to use a significantly larger number of intermediate layers in its architecture. This is made possible by the availability of large amounts of data and the advances in GPU technology.

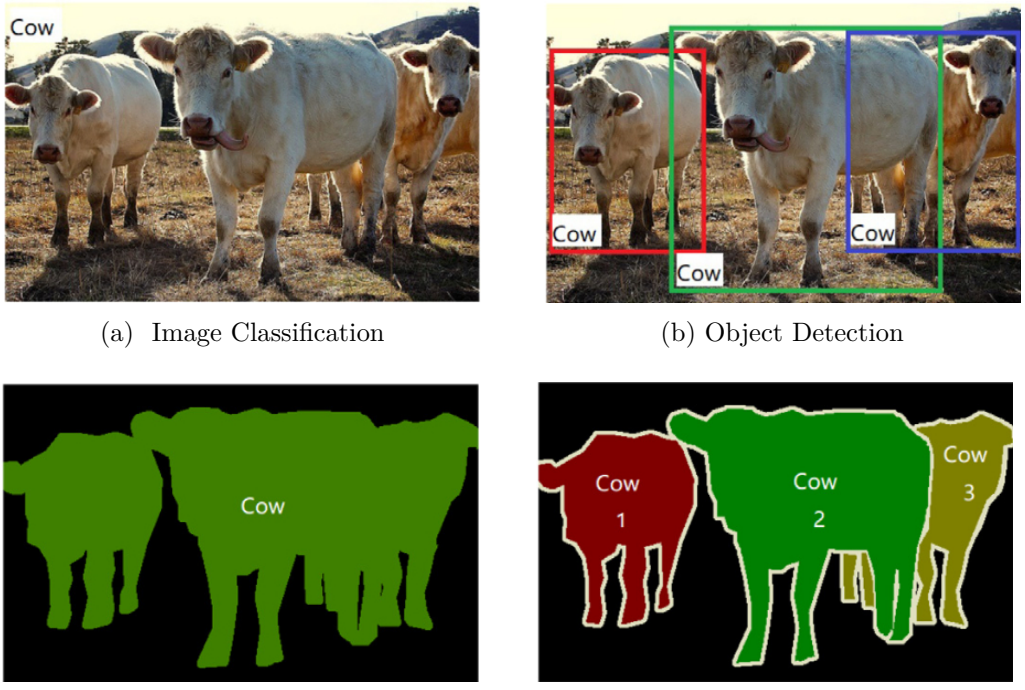
2.8.1 Datasets

Datasets are crucial for deep learning, as they provide the required data needed to train deep neural networks. In supervised learning, deep neural networks learn to recognize patterns in the input data by minimizing the difference between the predicted output and the ground truth label. The quality and quantity of training data have a significant impact on the performance of deep learning models. Large, diverse datasets such as the COCO Dataset [71] provide deep learning models with a broad range of examples to learn from, helping them to generalize better to new inputs and perform image analysis tasks.

Besides training data, the COCO dataset enables the comparison between different models on a common set of benchmarks. COCO contains over 330,000 images annotations for 80 different classes. Most of these classes may appear in an indoor environment, such as TV monitors, tables and cabinets. Trained models can be a valuable asset for SLAM systems, as they can aid in detecting objects within the environment and tracking their movements. By measuring the dynamics of each object, a more comprehensive understanding of the scene can be obtained, ultimately enhancing the performance and accuracy of a SLAM system.

2.8.2 Image analysis tasks

In computer vision, deep learning has been used for a wide range of applications, including image classification, object detection, semantic segmentation, and instance segmentation, as shown in Fig. 2.9. Image classification is the task of assigning a label or category to an image based on its content. The next sections will further discuss the object detection as well as the semantic, instance and panoptic segmentation tasks.



(a) Image Classification (b) Object Detection

(c) Semantic Segmentation (d) Instance Segmentation

Figure 2.9: Deep learning tasks [3]

2.8.3 Object detection

Object detection is a computer vision task that involves detecting and localizing objects within an image or video, and draw 2D bounding boxes around them to indicate their location, along with their confidence score. There are several popular deep learning approaches for object detection, including Faster R-CNN, SSD, and YOLO.

Faster R-CNN [4] is a region-based convolutional neural network. This approach involves two stages: region proposal and object detection. In the region proposal stage, a set of candidate regions are generated using a region proposal network. In the object detection stage, each candidate region is classified and refined using a detection network. Fig. 2.10 shows the Faster R-CNN model.

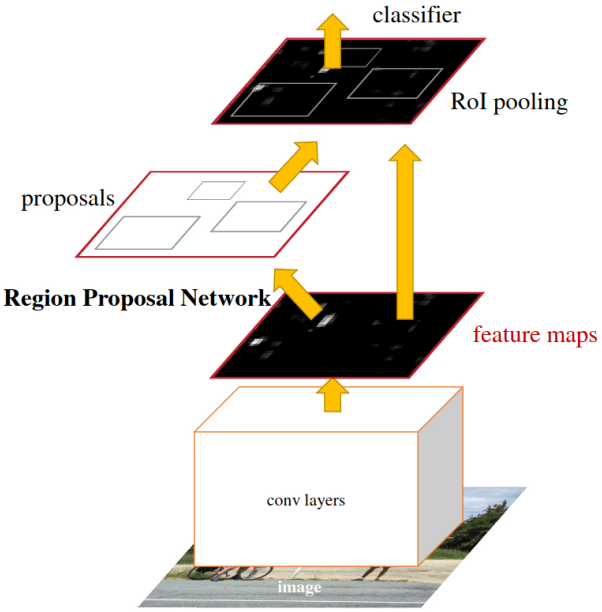


Figure 2.10: Faster R-CNN Model, image from [4]

While accurate, traditional object detection algorithms have complex pipelines and are not suitable for real-time applications. In contrast, "one-shot detectors" are more efficient as they do not rely on region proposals to locate objects within an image.

SSD, proposed by Liu *et al.* [56], is a deep learning approach that performs object detection in a single shot. It involves predicting both the class and location of objects directly from the input image. SSD is designed to be faster and more accurate than traditional two-stage methods like Faster R-CNN. SSD were used in SLAM systems by Sunderhauf *et al.* [72] and Xiao *et al.*[73].

Another single shot technique is YOLO, from Liu *et al.* [5]. YOLO is designed for real-time object detection. It involves dividing the input image into a grid and predicting the class and location of objects within each grid cell. Fig 2.11 shows the YOLO architecture. The image input is partitioned into a grid, where each cell predicts a group of class probabilities and a specific number of bounding boxes. These bounding boxes comprise five predictions, which are the x and y coordinates of the box's center relative to the cell's border, its width, height, and confidence. The boxes with class probabilities that surpass a certain threshold are then chosen. This approach allows for efficient object detection without the need for region proposals.

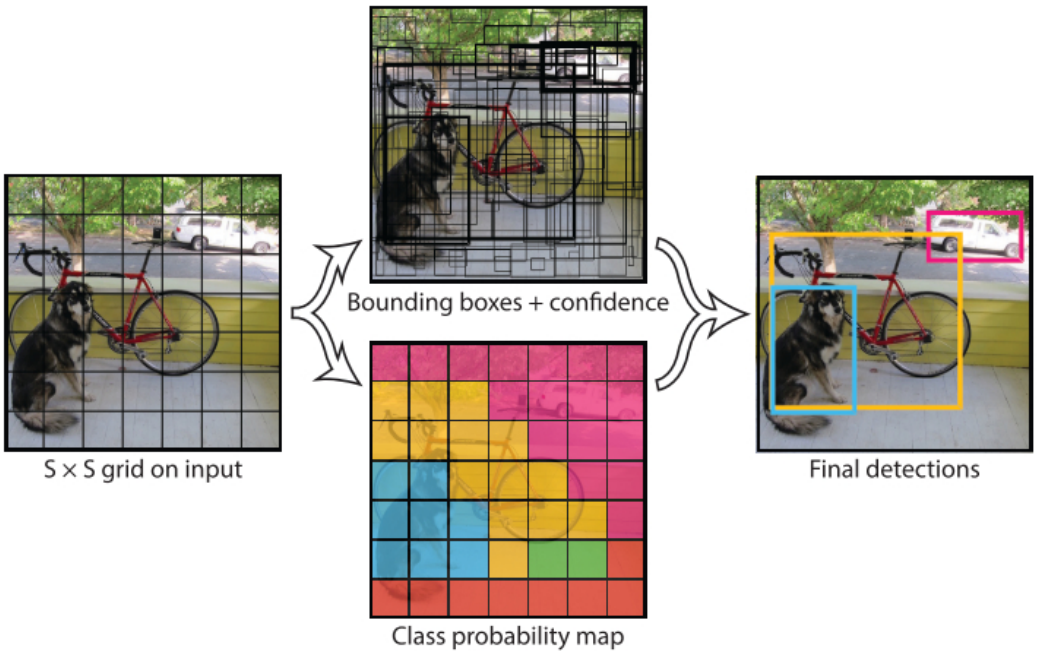


Figure 2.11: YOLO Model, image from [5]

YOLO is constantly in development and already has eight versions. Every new released version improved YOLO in its accuracy and time performance, making it fit to run in a SLAM system such as Crowd-SLAM [16].

2.8.4 Semantic Segmentation

Semantic segmentation assign a semantic label to every pixel in an image. The goal is to partition an image into meaningful segments and classify each segment according to its semantic meaning. Deep learning semantic segmentation models can learn high-level representations of images by extracting features at different levels of abstraction. These features can be used to predict the class labels of each pixel in an image.

U-Net [6] is a pioneer work in semantic segmentation. It produces dense output maps that preserve the spatial resolution of the input image. This architecture, shown in Fig 2.12, is a type of encoder-decoder network, which can transform an input sequence of data into an abstract latent representation, and then decode the representation vector to generate an output sequence of data.

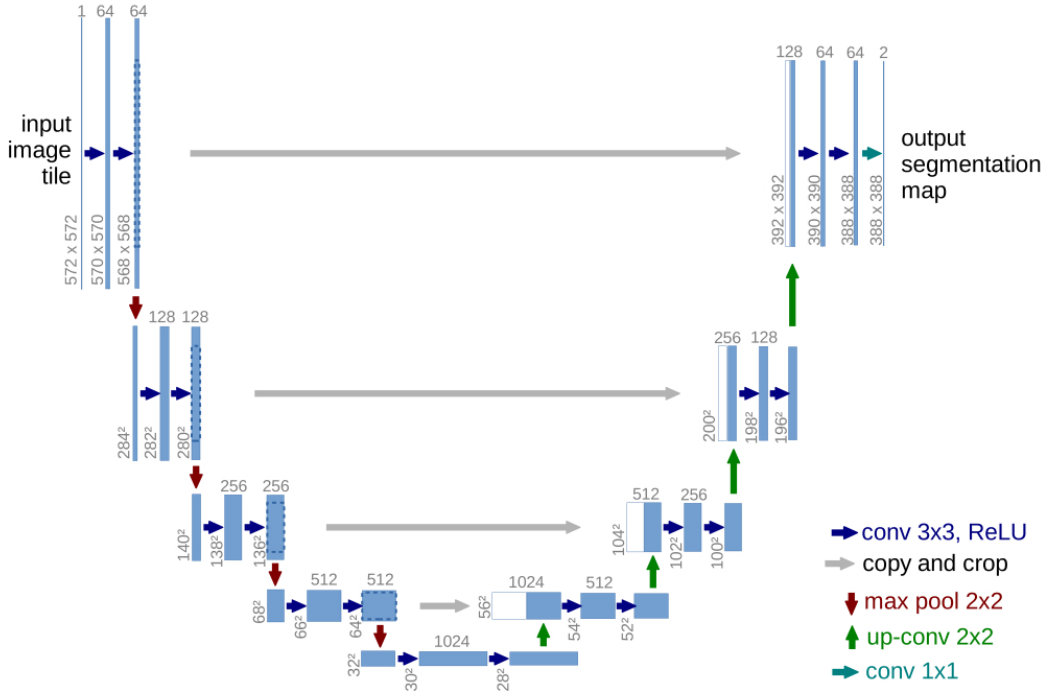


Figure 2.12: U-Net encoder-decoder network, image from [6]

For instance, DS-SLAM [19] and Ji *et al.* [10] are SLAM systems that utilize SegNet [59], an encoder-decoder semantic segmentation network similar to U-Net, to identify moving objects in the scene. However, semantic segmentation has some limitations. It may not perform as well on complex or cluttered scenes with multiple overlapping objects.

2.8.5 Instance Segmentation

Instance segmentation involves not only detecting objects in an image, but also identifying each instance of an object and differentiating it from other instances and other objects in the same image. In other words, instance segmentation aims to locate and segment each object instance in an image, providing a precise pixel-level segmentation mask for each object.

Instance segmentation is a more challenging task than semantic segmentation, which involves only classifying each pixel in an image into a predefined set of classes. Instance segmentation requires both object detection and

pixel-level segmentation, which makes it more computationally demanding and requires more data and specialized algorithms.

Mask R-CNN [7], is an extension of the faster R-CNN object detection model. Mask R-CNN model, shown in Fig 2.13, adds a parallel branch to the faster R-CNN model that generates a binary mask for each object instance, in addition to predicting the class and bounding box coordinates of each object.

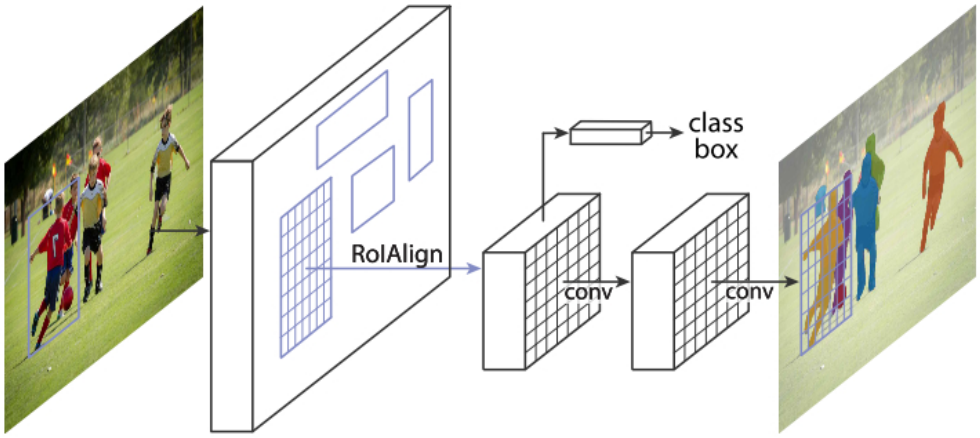


Figure 2.13: Mask R-CNN model architecture [7]

2.8.6 Panoptic Segmentation

Panoptic segmentation aims to provide a comprehensive understanding of visual scenes by simultaneously detecting and segmenting all objects in the scene, including both "Stuff" (e.g. sky, sand, sea) and "Thing" (e.g. people, kites) classes, as shown in Fig 2.14. Unlike traditional semantic segmentation, which only labels pixels with class labels, panoptic segmentation assigns unique IDs to each individual instance of an object, allowing for more fine-grained analysis of the scene. Panoptic segmentation is a challenging task due to the large number of object classes and the high variability in object appearance and shape.

The paper from Kirillov *et al.* [62], was the first work to talk about panoptic segmentation as a unified view of image segmentation. The authors proposed a panoptic quality (PQ) metric to measure the accuracy of *things* and *stuff* objects in datasets. PQ calculates the intersection over union (IoU), which will be explained with more details in Chapter 4, between predicted segments and ground-truth for each trained class and average over all trained classes. PQ is defined as

$$PQ = \frac{\sum_{(p,q)\in TP} IoU(p,q)}{|TP| + 0.5|FP| + 0.5|FN|} \tag{2-13}$$

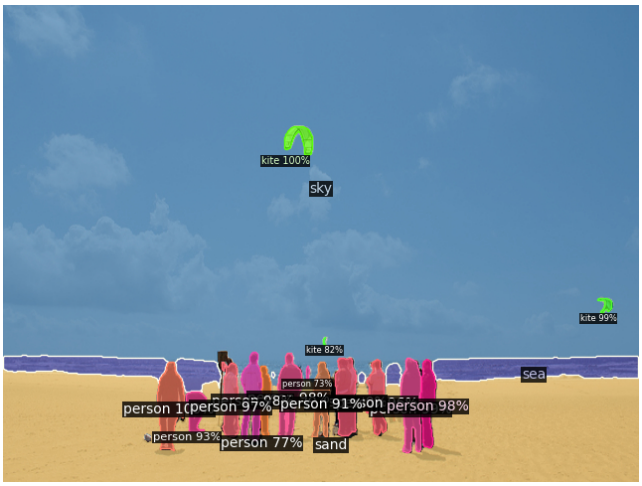


Figure 2.14: Panoptic Segmentation output

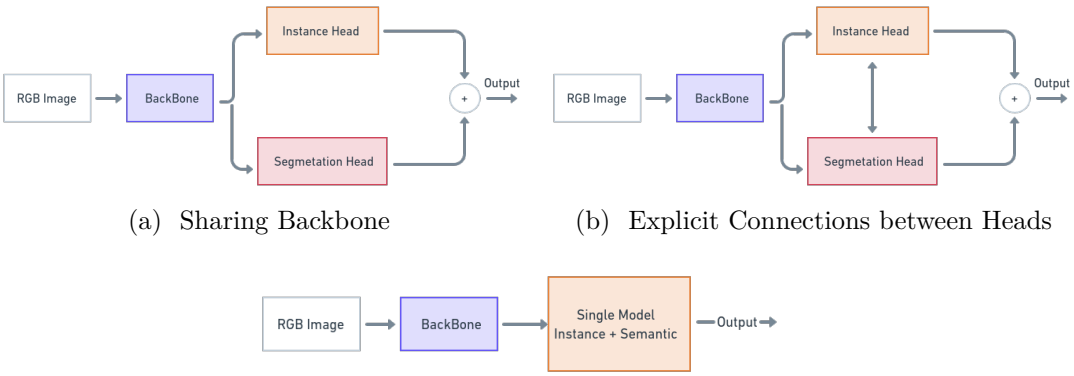
Where TP means true positives, FP means false positives and FN means false negatives. The PQ metric does not evaluate predictions for unknown pixels, i.e, pixels that the model could not predict a class label to. The PQ metric is important to evaluate panoptic segmentation models from the literature with respect to the COCO dataset, and selecting the most suitable model for the employment in the proposed methodology.

Most of the proposed panoptic segmentation use RGB images or LiDAR measurements as inputs. In the rest of this subsection, it will be discussed existing RGB image-based methods.

Several techniques for panoptic segmentation utilize instance segmentation as a separate step before combining or aggregating the results to obtain the final panoptic segmentation output. In these approaches, the shared backbone of the network is used to extract features, which are then employed in other parts of the network, as shown in Fig. 2.15a. Alternatively, some frameworks have employed explicit connections between instance and semantic networks to achieve similar results, portrayed in Fig. 2.15b. Furthermore, some approaches combine the two modules as a single model, shown in Fig. 2.15c, to avoid merging the outputs from individual heads.

JSIS-Net [74] utilizes a ResNet-50 backbone to extract features in a CNN-based model. This is followed by the implementation of two branches for instance and semantic segmentation. The authors merge these two branches through a heuristic method to produce the final output of panoptic segmentation.

In their work, Li et al. [8] introduce the attention-guided unifed network (AUNet). The attention mechanism is used to focus on specific parts of the input data that are more relevant to a particular task. It allows the network to selectively weigh the importance of different features. AUNet employs three parallel branches: RPN, background and foreground segmentation. The proposed attention modules are added to each branch and the results merged to provide the panoptic output. Moreover, this paper used Feature Pyramid Network (FPN) [75] as a backbone to perform multiscale feature representations.



(a) Sharing Backbone (b) Explicit Connections between Heads

(c) Singel Shot

Figure 2.15: Network arquiteture types for panoptic segmentation models

FPN is a common backbone used in several panoptic segmentation models. The basic idea behind FPN, shown in Fig 2.16, is to construct a pyramid of feature maps with increasing spatial resolutions and decreasing semantic information. This is achieved by adding a top-down pathway to CNN backbone, such as ResNet, and merging it with the bottom-up pathway to create a feature pyramid.

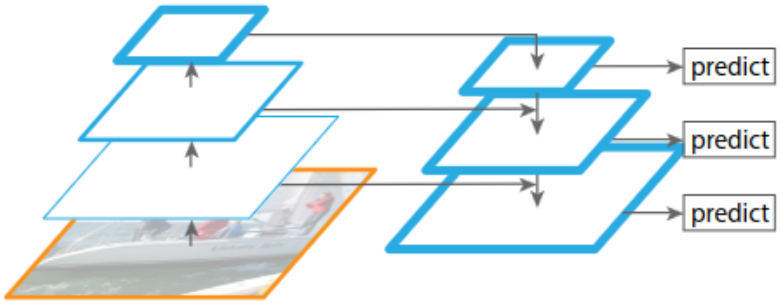


Figure 2.16: Feature pyramid network [8]

UPSNet [76] is another method that utilize FPN as backbone. The authors designed an instance segmentation head based on Mask-RCNN and proposed a deformable convolution based on semantic convolution for pixel-wise

classification. The same authors from [62] proposed PanopticFPN [77], a single model with two subtasks based on FPN, which demonstrated greater precision in relation to the concatenation from instance and semantic segmentation. Li et al. [78] introduced the *things* and *stuff* consistency network with explicit connections (TASCNet), which generates a binary mask for each pixel predicting *things* versus *stuff*. Additionally, an extra loss is integrated to ensure coherence between the predictions of *things* and *stuff*.

Occlusion Aware Network (OANet) [79] also proposes a single network for two subtasks. To fuse the instance with semantic subtasks, the authors developed a spatial ranking module based on a CNN architecture to address the issue of occlusion between the predicted instances.

SOGNet [80] utilizes a Scene Overlap Graph (SOG) to represent the interactions between instances and a spatial-semantic hierarchy to capture the scale and context information. This work uses FPN followed by an overlap-aware graph construction and propagation module that captures instance interactions.

The paper from Lazarow *et al.* [81] proposes OCFusion, an instance occlusion module, to improve the panoptic segmentation task. The module is based on Mask-RCNN to generate occlusion maps. These maps modify the RoIAlign operation to better handle with occluded instances.

Wang *et al.* [82] proposed a method called pixel consensus voting (PCV). The objective of PCV is to elevate pixels to a first-class task, where each pixel is expected to give evidence for the existence and position of the objects it potentially affiliates to.

DETR [83] is a new approach to object detection that uses transformer, which is an encoder-decoder neural network architecture commonly used in natural language processing tasks. In this approach, the input image is first divided into a set of non-overlapping image patches, and a transformer network is applied to each patch to extract features. The resulting set of patch features is then processed by a series of transformer layers to generate a set of object detections, including their class labels, bounding boxes, and confidence scores. The authors added a FPN-style CNN to generate binary masks for each bounding box detected and merge using pixel-wise argmax to perform panoptic segmentation tasks. DETR was later used as reference for MaskFormer [84], which introducted six additioanl transformers encoder layers after the CNN backbone from DETR and resulted in direct binary masks prediction with better accuracy.

In the work from Hou *et al.* [85]. The authors propose a single-shot panoptic network using a dense object detector and segmenting each instance

using an efficient segmentation network. The method refines the masks by propagating information across nearby instances using a graph-based approach to produce the panoptic output.

Panoptic-Deeplab [86] uses a different approach than the previous methods. It adopts a CNN architecture with a dual-decoder based on the author's previous work called DeepLabV3+ [87], and dual spatial pyramid pooling (ASPP) modules for semantic and instance segmentation. In the same context, the authors from [88], [89] and [90] developed methods based on the transformer's encoder-decoder architecture with custom self-attention layers to achieve more accurate and consistent segmentation results.

Table 2.2 summarize all previously discussed panoptic segmentation methods in a chronological order. PanopticFPN was selected to be incorporated into the proposed SLAM system because of its open-source nature, straight forward documentation and strong accuracy using the PQ metric in the COCO test-dev dataset.

Table 2.2: Panoptic Segmentation Models

Model	*Year*	*Open-Source*	*COCO test-dev PQ*	*FPN backbone*
UPSNet	2019	Yes	42.5	Yes
TascNet	2019	No	40.7	Yes
JSIS-Net	2019	No	27.2	No
PanopticFPN	2019	Yes	46.8	Yes
OANet	2019	No	41.3	Yes
AUNet	2019	No	46.5	Yes
SOGNet	2019	No	47.8	Yes
Dense-Det	2020	Yes	37.1	Yes
PCV	2020	Yes	37.1	Yes
DETR	2020	Yes	45.1	Yes
Panoptic-DeepLab	2020	No	41.4	No
Axial-Deeplab	2020	Yes	43.9	No
MaskFormer	2021	Yes	52.7	No
Max-Deeplab	2021	Yes	51.3	No
CMT-Deeplab	2022	No	55.7	No

This chapter described the basic concepts from classical computer vision techniques and deep learning approaches. In the next chapter, it is presented the basic ideas of a visual SLAM system.

3
Feature-based Visual SLAM

3.1
Introduction

The objective of this chapter is to introduce the fundamental concepts of feature-based visual SLAM in static scenarios, utilizing a graph-based probabilistic formulation. ORB-SLAM3 will be used as a baseline, since it is adopted as common framework by many SLAM systems, and it is used as a major example to explain the SLAM subsystems and how they interact with each other. Section 3.2 is presenting the probabilistic formulation of the SLAM problem using a graph-based approach. The SLAM frond-end, which includes camera tracking, map points and loop closure, is discussed in detail in Section 3.3, while Section 3.4 provides an overview of graph optimization frameworks. The output of a feature-based SLAM system is presented in Section 3.5.

3.2
Graph-based formulation

The Graph-SLAM approach is a common choice for SLAM due to its high accuracy and efficiency. It involves two main steps: front-end and back-end. The front-end is responsible for constructing the graph and comprises two main tasks: pose estimation and loop closure. Pose estimation involves locally estimating the robot's pose, while loop closure is responsible for long-term data association. The back-end is responsible for graph optimization, which estimates an optimal trajectory of the robot and a precise map of the environment. The general system is shown in Fig 3.1.

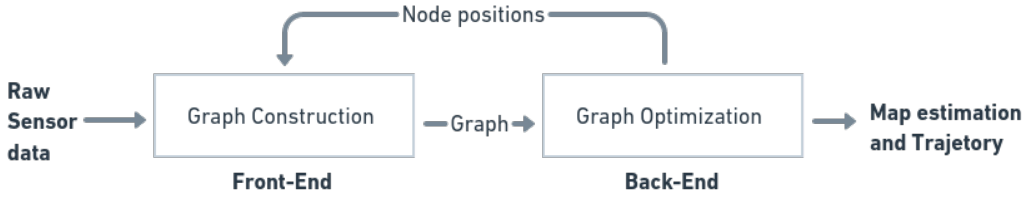


Figure 3.1: Graph-SLAM system

The problem is modeled as a graph with nodes representing the robot's poses and edges representing measurement constraints between these poses. The aim is to optimize the error constraints caused by odometry drift. Two

types of edges exist in visual SLAM: those created using visual odometry and those created for loop closure.

The graph is initially constructed with nodes representing robot poses and edges representing constraints between poses based on measurements obtained by visual odometry. Fig 3.2 shows a representation of the graph.

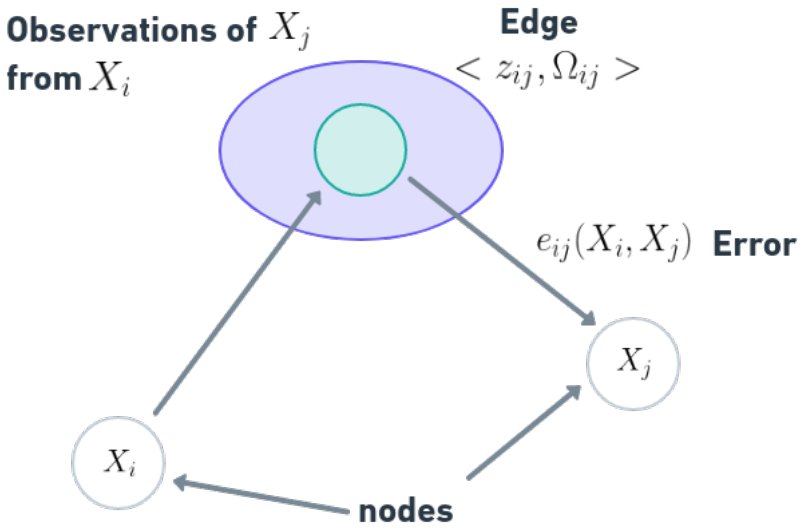
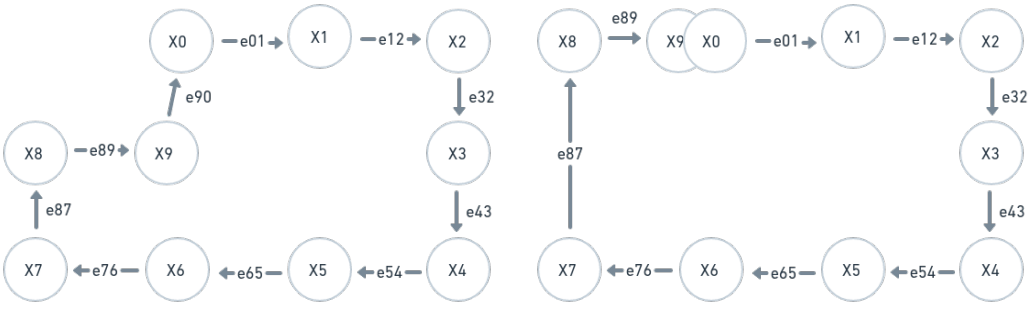


Figure 3.2: Pose-Graph representation

The initial graph structure is illustrated in Fig. 3.3a. The edges between poses are denoted by e01 to e89, representing visual odometry estimations. Additionally, there is a loop closure edge represented by e90, which is detected by a place recognition algorithm that evaluates if the robot has revisited a previously explored location and establishes a connection between the current and the old poses. Following loop closure, the graph is optimized to correct the visual odometry drifts, as demonstrated in the diagram OF Fig. 3.3b.



(a) Pose-graph withloop closure (b) Pose-graph after optimization

Figure 3.3: Example of a pose-graph being optmized after loop closure

The mathematical formulation of the graph-based SLAM approach is based on Maximum-a-posteriori estimation (MAP). MAP estimation is a sta-

tistical approach used to estimate an unknown parameter or set of parameters in a probabilistic model. In this method, a prior probability distribution is combined with the likelihood function to estimate the posterior probability distribution of the parameters. MAP estimation aims to find the state that maximize the posterior probability distribution given measurements

$$X^* = \underset{X}{argmax}\ p(X|Z) = \underset{X}{argmax}\ p(Z|X)\ p(Z) \tag{3-1}$$

Where the X^* is the state, X is the posterior probability distribution, Z correspond to measurements for a given state and p(X) is the prior probability. Assuming independent measurements, Eq. 3-1 is modified to

$$X^* = \underset{X}{argmax}\ p(X) \prod_{k=1}^{m} p(z_k|X) = \underset{X}{argmax}\ p(X) \prod_{k=1}^{m} p(z_k|X_k) \tag{3-2}$$

The MAP approach is particularly useful when dealing with problems that involve uncertainty or noisy data. Assuming that the measurement noise is a zero-mean Gaussian distribution, then

$$p(z_k|X_k) \propto exp(-0.5||k_k(X_k) - z_k||^2_{\Omega_k}) \tag{3-3}$$

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

$$X^* = \underset{X}{argmin} - log(p(X) \prod_{k=1}^{m} p(z_k|X_k)) = \underset{X}{argmin} \sum_{k=0}^{m} ||k_k(X_k) - z_k||^2_{\Omega_k} \tag{3-4}$$

In the context of SLAM, MAP estimation is often used to estimate the robot's trajectory and the map of the environment. The measurements collected by the robot are used to construct a probabilistic model, and the prior information about the robot's motion and the environment is incorporated to estimate the posterior distribution of the robot's pose and the map.

3.3 SLAM Front-End

The pose-graph front-end is responsible for constructing the graph of robot poses and landmark positions. The graph is a data structure that represents the history of robot motion and observations. The graph construction process involves two main tasks: camera tracking and loop closure detection.

3.3.1 Map Points

The sparse map is a set of 3D points in space that is estimated from 2D image feature correspondences in multiple camera frames. These 3D points are called map points, and each map point stores: (a) Its position in world coordinate system; (b) The viewing directions, i.e., the rays connecting a point with the optical center of the keyframes that observe it; (c) The ORB descriptor vector that describes the appearance of the map point based on its surrounding image patches in the keyframes where it was observed; (d) Various flags that indicate the status of the map point, such as whether it is currently visible, whether it has been recently updated or deleted, and whether it is being used for loop closure detection; (e) Information about the scale of the map point relative to the camera's focal length and image resolution, which can be used to estimate the depth of the point in the scene. Map point coordinates (X, Y, Z) are calculated using equations 2-2, 2-3 and 2-4 respectively. Their purposed is to track the motion of the camera between frames, to verify the consistency of the estimated camera poses, as well as to detect and correct loop closures.

3.3.2 Camera Tracking

The tracking system is responsible for finding feature matches between consecutive frames to estimate the motion of the camera relative to a global reference frame, while building a map representation of the environment. Motion blur, occlusions, and changes in lighting are challenges that the camera system must surpass.

ORB-SLAM3's camera tracking subsystem can be described as follows. First, ORB feature detector is used to extract keypoints and its descriptors. Then, the initial pose estimation is predicted using a constant velocity model and a guided search of the map points from the last frame. The camera motion between consecutive frames is estimated using matched featured from the feature matching algorithm. In case of lost track, the current frame is convert into a bag-of-words and stored as global relocalization candidate in order to find the camera pose. Once the camera pose is estimated and a set of matched features are obtained, ORB-SLAM3 optimizes the camera pose with all the map point correspondences found in the image frame using the local map tracking algorithm. After that, the current frame passes through a decision processes to be spawned as a keyframe or not. Keyframes are selected based on their motion with respect to the previous keyframe, as well as the number of features and the map coverage. After that, ORB-SLAM3 performs

a local bundle adjustment to optimize the camera poses and the 3D points of the selected keyframes. Finally, the system detects loop closures and performs global bundle adjustment to optimize camera poses and map points of all the keyframes.

3.3.3 Loop Closure

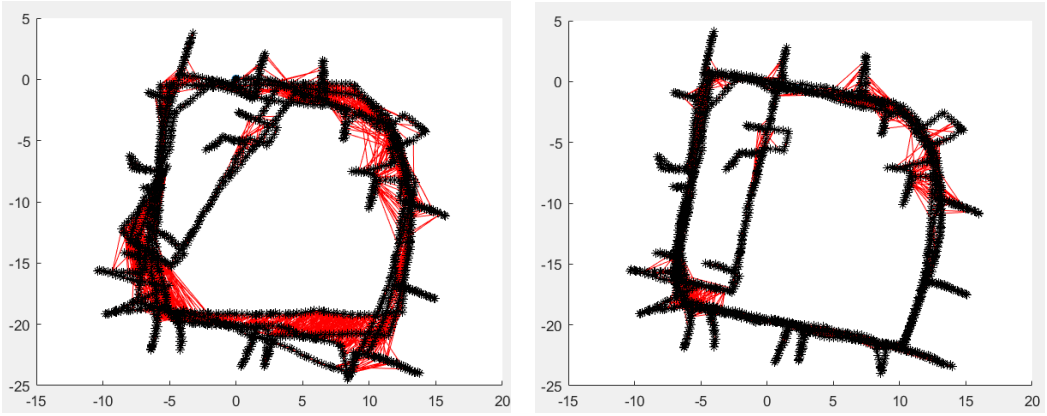
Loop closure detection is the process of identifying when the robot revisits a previously visited location. This is done by comparing the current sensor readings to the previously recorded data and checking if they match well enough. If a loop closure is detected, a new edge is added to the graph that links the current pose to the previously visited pose. This edge helps to correct any drift that may have occurred due to errors in pose estimation. Also, this helps to manage the size of the graph by discarding nodes and edges that are no longer useful. This is necessary to avoid memory overload and to ensure that the graph can be efficiently optimized.

The bag-of-words approach is a technique used in place recognition systems for representing images through visual words selected from a vocabulary. Local descriptors such as SIFT [67] or ORB [46] are used to obtain these words. This approach enables fast comparisons between images across large datasets, instead of slow direct comparisons between individual features. By quantizing features into a vocabulary, it increases computational efficiency. A number of place recognition systems, including FAB-MAP [91], FAB-MAP2 [92], and DBoW2 [93], employ the bag-of-words technique.

The loop closure algorithm from ORB-SLAM3 has seven main steps. First, the algorithm checks for significant motion before searching for a loop candidate. Then, it searches for a viable loop candidate by calculating bag-of-words-based matching scores between the current keyframe and its connected frames. If a loop candidate is found, the algorithm performs a temporal consistency test to ensure consistency between the current keyframe and its loop candidate. If the consistency test is passed, the algorithm finds ORB matches between the current keyframe and its loop candidate and performs RANSAC iterations to find a similarity transformation. The transformation is then optimized using a pose-graph optimization tool. Finally, the algorithm projects map-points of the loop candidate onto the current keyframe using the optimized transformation and checks if the number of matches is greater than a threshold. If any of these steps fail, the current keyframe is added to the central keyframe database, aborting the loop closure routine.

3.4 SLAM Back-End

While the front-end is responsible for constructing the graph, the back-end is responsible for graph optimization. The optimization involves minimizing the error of MAP estimation in Eq. 3-4 between the measurements and the expected values represented by the graph. The objective function to be minimized is defined as the sum of the squared error of the residuals of the measurements. This nonlinear optimization problem is typically solved using iterative techniques such as Gauss-Newton or Levenberg-Marquardt. Fig 3.4 shows an example of a 2D graph without optimization and the resulting map using Gauss-Newton optimization.



(a) Graph without optimization (b) Graph optimized

Figure 3.4: Example of a graph optimization

Various graph optimization frameworks for SLAM have been proposed in the literature. For instance, $\sqrt{SAM}$ (square root SAM), which was developed by Dellaert *et al.* [94], and iSAM, proposed by Kaess *et al.* [95]. However, the most widely used framework is g2o [96]. This open-source C++ framework has been employed as a back-end in many monocular, stereo, and RGB-D SLAM implementations, including ORB-SLAM [12], ORB-SLAM2 [14], ORB-SLAM3 [9], and RGBDSLAM [44]. The g2o framework allows the use of different solvers, such as Cholesky, Preconditioned Conjugate Gradient, and Levenberg-Marquardt, to achieve optimal results.

3.5 Sparse Map

Sparse maps provide a compact representation of the environment that can be used to generate the constraints between poses, while minimizing the amount of redundant information in the graph. This reduces the computational

complexity of the optimization process and makes it easier to obtain accurate estimates of the robot's trajectory and the map of the environment.

The visual representation in Figure 3.5 illustrates a feature-based map created by the ORB-SLAM3 algorithm [9]. The camera poses are depicted in blue, while the red and black points are registered map points, and the green lines indicate the edges of the graph. While sparse maps generated by ORB-SLAM3 cannot be directly utilized for navigation, certain methods, such as the approach proposed by Chen *et al.* [97], can transform the map into a navigable representation.

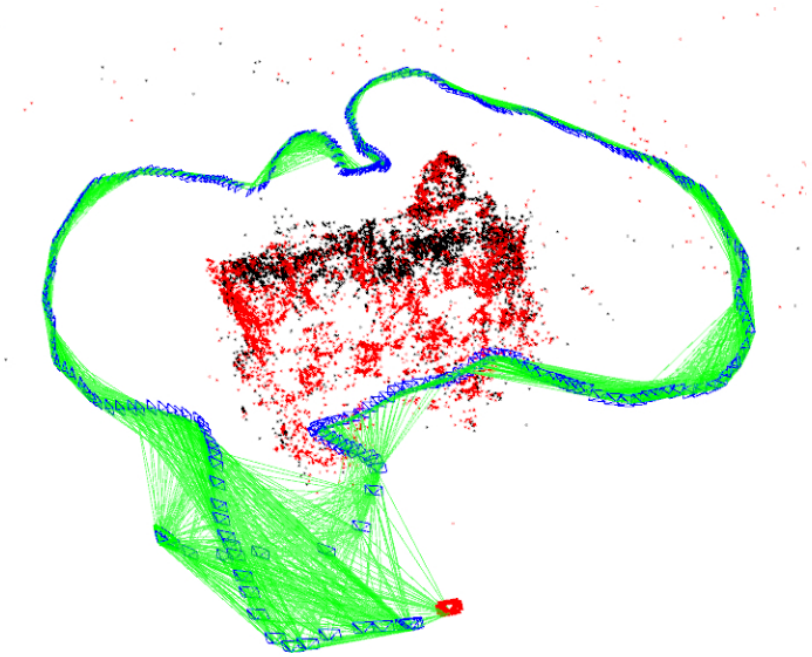


Figure 3.5: Sparse map obtained with ORB-SLAM3 [9].

Chapter 3 showed the theoretical formulation of a visual SLAM system. Chapter 4 uses the concepts presented in chapter 2 and 3 to describe the proposed methodology.

4
Methodology

4.1
Introduction

This chapter present Panoptic-SLAM, a visual SLAM for dynamic environments that uses panoptic segmentation to detect and filter moving objects. To the best of our knowledge, this is the first visual SLAM system based on panoptic segmentation that can perform robust localization in dynamic environments in the presence of unknown and unlabeled moving objects.

4.2
Overview

Figure 4.1 shows a diagram of the proposed approach. The SLAM system is based on ORB-SLAM3 [9], and it is composed of four threads that run in parallel: panoptic segmentation, tracking, local mapping and loop closing. First, the image frames are process in both tracking and panoptic segmentation threads. ORB features are extracted in the tracking thread, and the image is segmented into objects, background and unknown information in the panoptic segmentation thread. The known objects are sent to a short-term data association algorithm to determine if they are new or were present in the last frame. The features associated with the background are matched with the ones of the last frame and used to compute a fundamental matrix. Using this fundamental matrix, the keypoints associated to known and unknown objects are classified as dynamic or static. Only static features are used for mapping and loop closing.

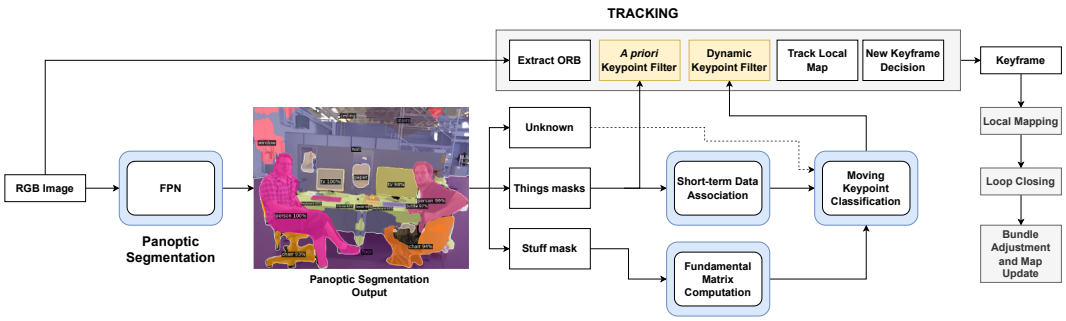


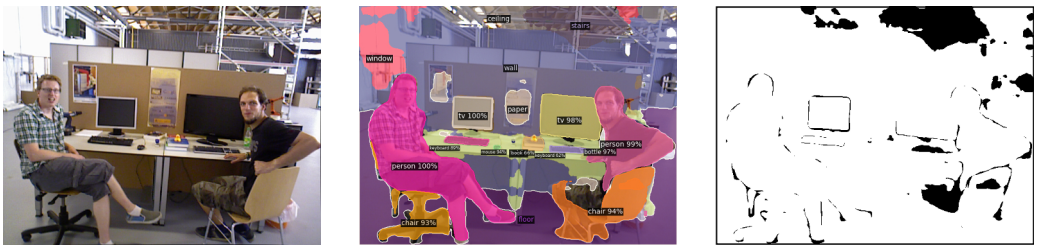
Figure 4.1: Panoptic-SLAM System Framework

4.3 Panoptic Segmentation

As mentioned in Chapter 2, in panoptic segmentation, pixels are classified either as "Things" or "Stuff". Things are countable objects with well-defined boundaries and are potentially movable, such as people, animals, or vehicles. Stuff, on the other hand, refers to uncountable amorphous regions of the image, mostly "unmovable" such as sky, floor, or walls. The instance segmentation identifies individual objects in the "Things" category, while the semantic segmentation labels all pixels in the image with their corresponding category. With high accuracy and efficiency, this segmentation method surpasses previous box-based or box-free models.

Also, some parts of the image are unknown, i.e. regions that the panoptic model could not predict any label. The unknown masks can happen due to motion blur or unlabeled objects in the scene. There can also be wrong detections, i.e., objects with wrong classification. This can happen either if an object in the scene is labeled, but there is another class with similar characteristics (e.g. tv and monitor), or if an unlabeled object is similar to a labeled class.

This master thesis uses PanopticFPN [77] for inference, trained with the COCO dataset [71], which can segment up to 80 different labels. Fig.4.2a shows an example of an input rgb image, Fig. 4.2b shows the associated panoptic masks and Fig. 4.2c unknown pixel regions in black.



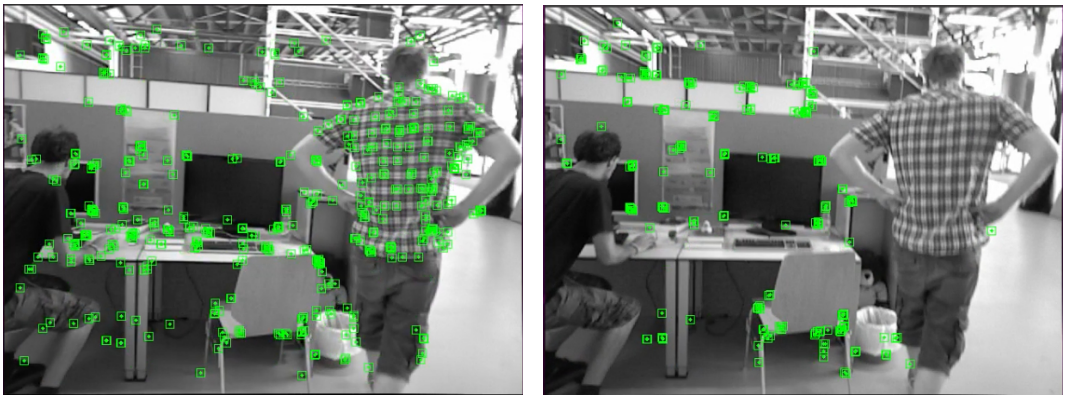
(a) RGB image (b) Panoptic image (c) Unknown pixels image

Figure 4.2: Panoptic image outputs

4.4 Dynamic Keypoint Filtering

The proposed method for dynamic keypoint detection and filtering is divided into four processes: *a priori* keypoint filtering, fundamental matrix computation, short-term data association, and moving keypoint classification of Things and Unknown.

First, the panoptic model generates all masks predictions. People keypoints are filtered *a priori*, as humans can be considered as highly dynamic, and just in rare situations they remain completely still for a long period. Figure 4.3 shows a comparison between people keypoint filter and ORB-SLAM3. The keypoints belonging to the "Stuff" mask are used for computing the fundamental matrix with RANSAC [50]. In order to calculate the fundamental matrix, corresponding features must be identified in both views. Without an accurate feature matching, the fundamental matrix cannot be accurately calculated, which can result in errors in subsequent tasks. The feature matching algorithm matches the ORB descriptors of the keypoints in both the reference and the query images using a nearest neighbor approach. Specifically, for each keypoint in the reference image, the algorithm searches for the closest keypoint in the query image based on the similarity of their ORB descriptors.



(a) ORB-SLAM3 (b) Panoptic-SLAM

Figure 4.3: People keypoint filter

The fundamental matrix is used to map feature points from the previous frame to their corresponding search domain in the current frame, namely the epipolar line. Assuming that the matched points in the current and last frames are p1 and p2, respectively, their homogeneous coordinate form can be represented as P1 and P2.

$$P1_j = [u_1, v_1, 1] \tag{4-1}$$

$$P2_j = [u_2, v_2, 1] \tag{4-2}$$

where "u" and "v" represent the pixel coordinates in the image frame, and "j" symbolizes the keypoint category ("Thing" or "Unknown"). Using these values, we can compute the epipolar line, denoted as L

$$L \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = FP1_j \tag{4-3}$$

Given that X, Y, Z represent line vectors and F represents the fundamental matrix, the distance between a matched point and its corresponding epipolar line can be determined as

$$D_j = \frac{P2_j^T FP1_j}{\sqrt{||X||^2 + ||Y||^2}} \tag{4-4}$$

The idea is to have the fundamental matrix calculated with static "Stuff" keypoints and use it with "Things" and "Unknown" feature matches in order to determine whether the distance from a matched point to its corresponding epipolar line is less than a certain threshold. If the distance is lower than the threshold, the matched keypoint is considered to be static. For the case of insufficient "Stuff" keypoints in the scene, the system will use the last calculated fundamental matrix.

4.5 Short-term Data Association

The moving keypoint classification of "Things" uses a short-term data association algorithm to address the issue of multiple objects with the same label in a frame. The short-term data association evaluates, using the intersection over union (IoU) metric, shown in Eq. 4-5, if a new "Thing" mask detected in the current frame correspond to a "Thing" mask of the last frame. Every instance mask is predicted with its respective bounding box. For every new pair of frames at time k and k-1, the algorithm checks if any two bounding boxes from the same label overlap. Once an overlap is detected, the algorithm will extract the object contours C^k and C^{k-1} to compute the IoU and determine if both contours have an association.

$$IoU = \frac{|C^k \cap C^{k-1}|}{|C^k \cup C^{k-1}|} \tag{4-5}$$

This process is detailed in Algorithm 2. For every new pair of consecutive images, the system checks whether a tracked "Thing" object has a new association and if objects have been added or removed from the scene.

Algorithm 2: Short-term Data Association

Data: Frame F_k, Panoptic object list P^{F_k} of current frame objects CurrentFrameObjs

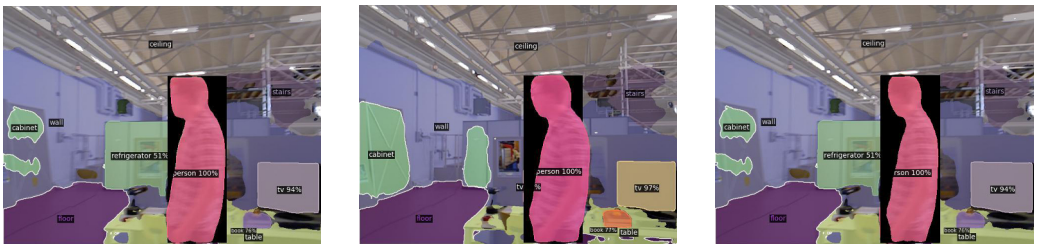
Data: Frame F_{k-1}, Panoptic object list $P^{F_{k-1}}$ of last frame objects lastFrameObjs

```
for det in P^{F_k} do
    associationfound = false; for lfo in P^{F_{k-1}} do
        if P^{F_k-1} not NULL then
            if det.label == lfo.label then
                iou =
                 GetIOU(det.bounding_box,det.mask,lfo.bounding_box,lfo.mask);
                IOU_Matrix(det)(lfo) = iou;
                if IOU_Matrix(det)(lfo) > IOU_Threshold then
                    associationfound = true;
                    lastObjID = lfo.tracking_id;
                    break;
                end
            end
        end
    end
    if associationfound then
        det.tracking_id = LastObjID;
    end
    else
        Create new tracking_id;
        det.tracking_id = new tracking_id;
    end
end
```

To filter the keypoints of objects and determine which ones are in motion, it is essential to differentiate between multiple objects that share the same label. If an object in the current frame is not matched with any object in the previous frame, it is considered a new one and the features associated with its mask are filtered.

Figures 4.4a and 4.4b show the short-term data association algorithm tracking a person during the instant T_0 and T_1, respectively. Both masks are extracted from the area of the respective bounding boxes. In Fig. 4.4c both masks are overlapped, the IoU metric is calculated, and the tracking ID is

defined for that particular object. This process is repeated for every object pair with the same label.



(a) tracking person in time T_0

(b) tracking person in time T_1

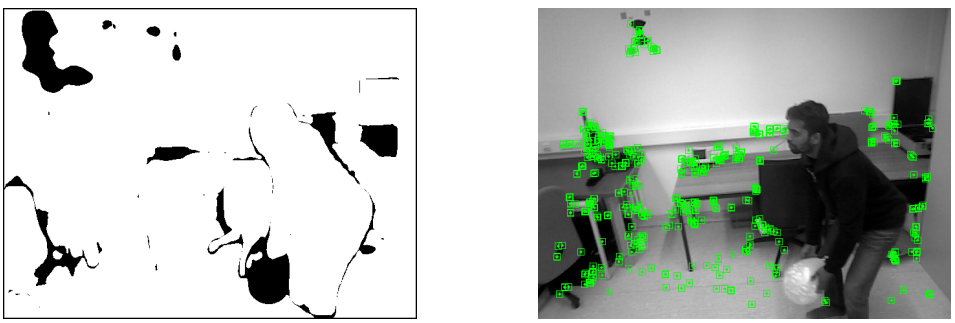
(c) Overlapped of the two masks

Figure 4.4: Example of the short-term data association algorithm tracking a person

4.6 Dynamic Unkonwn and Thing KeyPoints Classification

After the data association, the current and last frame keypoints belonging to "Things" from the same label and same tracking ID are matched. D_{thing} is calculated using the matched features and the "Stuff" fundamental matrix to determine which points are dynamic and, consequently, filtered.

Features belonging to unknown pixels are located by adding all known masks into a single image and analyzing the black areas in it, as shown in Fig. 4.5a. The process is similar to the moving keypoint classification of "Things". After matched keypoints from current and last frame are found in the unknown mask, these points are checked using the "Stuff" fundamental matrix to calculate the epipolar distance D_{unk} and to determine if they are static or dynamic. The unmatched points are filtered. In Fig. 4.5b is shown a man moving an "Unknown" object (floating balloon) and the keypoints belonging to this object been filtered.



(a) Unknwon mask

(b) Keypoint filtering

Figure 4.5: Example of an unknown moving object being filtered in a sequence of the Bonn Dynamic Dataset

In Algorithm 3 contains all steps from the Panoptic-SLAM framework. Every set of keypoints filtered by each filtering processed is stored in a list to be removed from the SLAM system.

Algorithm 3: Panoptic-SLAM algorithm

Data: Current RGB Image Img_k
Data: Current Image Keypoints KP_k
KP_Filtered = list();
$Panoptic\_objs_k$ = PanopticNet(Img_k);
KP_Filtered_apriori = Apriori_Filtering($Panoptic\_objs_k$);
matched_KP = Feature_Machting(KP_k,KP_{k-1});
$Thing\_KP_k$, $Stuff\_KP_k$, $Unknown\_KP_k$ = Panoptic_KP_Classification($Panoptic\_objs_k$, matched_KP);
F = Fundamental_Matrix($Stuff\_KP_k$, $Stuff\_KP_{k-1}$);
$Thing\_KP_k$ = Short-term_DataAssociation($Thing\_KP_k$,$Thing\_KP_{k-1}$);
Thing_Dynamic_KP = Dynamic_Keypoint_Filtering($Thing\_KP_k$, F);
Unknown_Dynamic_KP = Dynamic_Keypoint_Filtering($Unknown\_KP_k$, F);
// Append all dynamic keypoints
KP_Filtered.append(KP_Filtered_apriori);
KP_Filtered.append(Thing_Dynamic_KP);
KP_Filtered.append(Unknown_Dynamic_KP);

4.7 Implementation Details

The SLAM system is implemented in C++. Most of the tools for inferencing deep learning models are written in Python and the compatibility with C++ are still in development. As a workaround, the panoptic segmentation model is implemented in Python using Pytorch and Detectron2 frameworks [98] and sends its results to SLAM system via Python-C-API. This API works as an interface for both languages to share functions and information online.

4.8 Loop closure

The loop closure algorithm benefits from the proposed dynamic keypoint filter. Since the filter removes outliers keypoints, that results in a reliable loop closure detection.

4.9 Final output

The final output from Panoptic-SLAM is a sparse map representation and the optimized camera trajectory. In Fig. 4.6 show an example of final output, where the red dot are map points, the blue squares are the camera poses and the green lines represent pose associations.

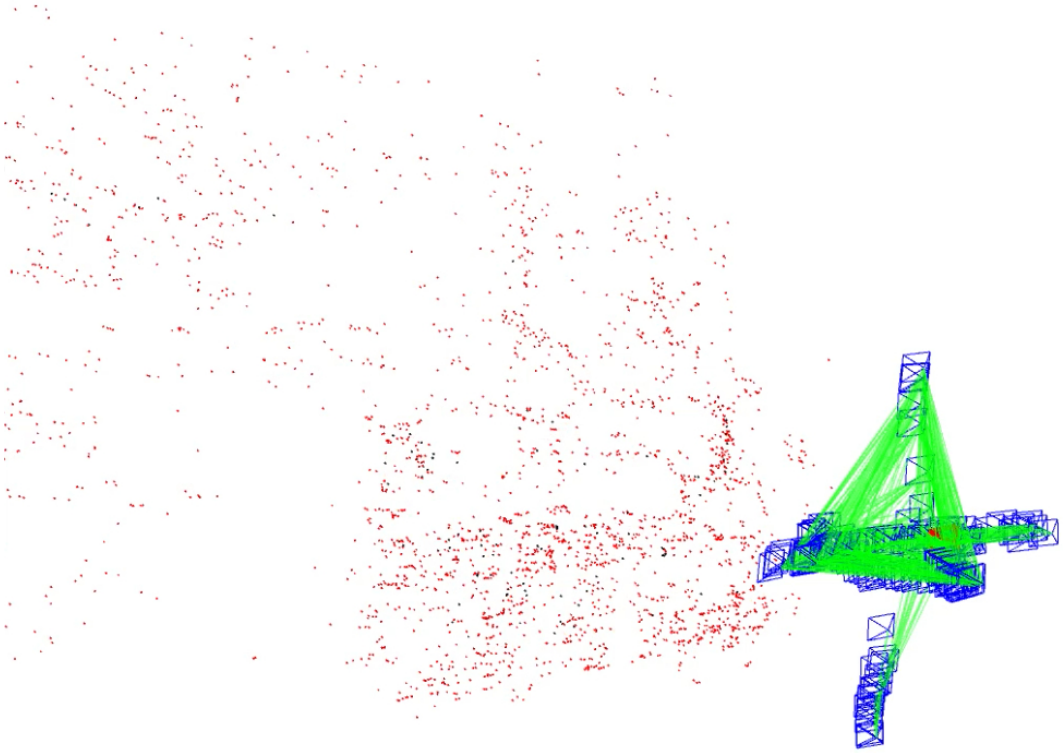


Figure 4.6: Panoptic-SLAM final output

In the following chapter, the results from Panoptic-SLAM and other state-of-the-art methods in real-world datasets are displayed and compared.

5
Results

5.1
Introduction

This chapter presents the results of the proposed methodology using datasets and experiments. Panoptic-SLAM was compared with state-of-the-art methods from the literature in challenging dynamic environments. Section 5.2 describes the evaluation metric used to compare the methods. Section 5.3 displays the parameter selection to run the simulation with the datasets. Section 5.4 and 5.5 explains the content of each evaluation dataset, along with the numerical results. Section 5.7 and 5.8 shows how the overall results are affected by each system configurations and run time analysis, respectively. Finally, Section 5.9 demonstrated the proposed methodology in a custom dataset.

5.2
Evaluation metric

The global consistency of the estimated trajectory is analyzed using the Absolute Trajectory Error (ATE). This metric compares the absolute distances between the translational components of ground truth and estimated trajectories. The Horn method [99] is necessary to align both trajectories, since they can be specified in arbitrary coordinate frames. The Horn method determines the rigid-body transformation S. Let us define in Eq. 5-1 the absolute trajectory error matrix at time i as

$$E_i = Q_i^{-1} S P_i \tag{5-1}$$

where P_i is the spatial pose of the estimated trajectory at time i and Q_i is the ground truth trajectory at time i in an arbitrary world coordinate system. In Eq. 5-2, the ATE metric is defined as the root-mean-square error of the error matrices:

$$ATE_{rmse} = \left(\frac{1}{n} \sum_{n=1}^{n} ||trans(E_i)||^2 \right)^{\frac{1}{2}} \tag{5-2}$$

5.3 Choice of parameters

Table 5.1 show the parameters used in Panoptic-SLAM. All sequences from TUM, BONN and custom datasets were performed using the presented parameters. For comparison, ORB-SLAM3 used the same number of keypoints as Panoptic-SLAM.

Table 5.1: Parameters used in the simulations and experiments

Description	Value
Number of Keypoints	2000
Thing obj. movement threshold	0.01
Unknown obj. movement threshold	0.01
IoU threshold	0.15

5.4 TUM Dataset

The TUM [61] dataset is used to evaluate the robustness of the system in dynamic environments. It includes both RGB and depth image sequences captured by a Microsoft Kinect camera, along with the corresponding ground-truth trajectories. The data was recorded at a resolution of 640 x 480 and a frequency of 30 Hz. Four sequences were chosen for the evaluation: fr3_w_static, fr3_w_xyz, fr3_w_rpy and fr3_w_halfsphere. These sequences depict two persons walking in a room, moving behind a desk, passing in front of the camera, and sitting on chairs.

The main difference between each sequence is the camera motion. In the xyz sequence, shown in 5.1, the camera is moved along the three axis while maintaining a fixed orientation. In the rpy sequence the camera is rotated around roll, pitch, and yaw axis, maintaining a fixed position, shown in Fig 5.2. In the halfsphere sequence, shown in Fig. 5.3, the camera follows a trajectory along a half-sphere. In the static sequence, the camera remains still as shown in Fig 5.4.

Figure 5.1: Images from the TUM fr3_w_xyz sequence

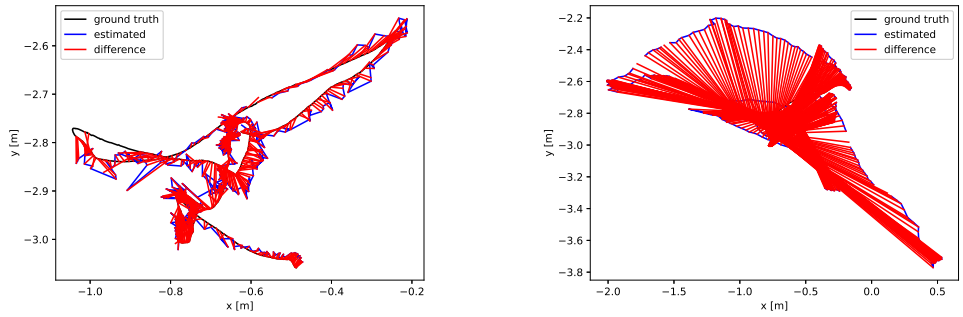
Figure 5.2: Images from the TUM fr3_w_rpy sequence

Figure 5.3: Images from the TUM fr3_w_halfsphere sequence

Figure 5.4: Images from the TUM fr3_sitting_static sequence

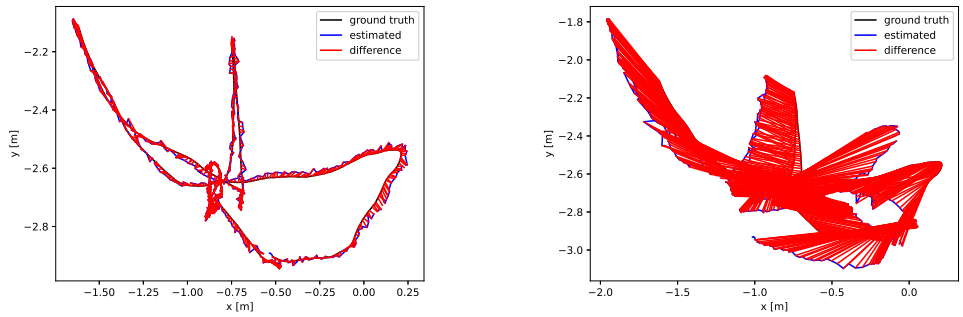
Figures 5.5 and 5.6 and 5.7 show the comparison between the ground-truth and the estimated trajectories of the proposed methodology and ORB-SLAM3, respectively, in the fr3_w_rpy and fr3_w_halfsphere and fr3_w_xyz sequences. The scenes are very challenging, with fast and complex camera motions, especially in the halfsphere sequence. The proposed system was able to perform an accurate estimation in all sequences. ORB-SLAM3, on the order

hand, resulted in a wrong camera pose estimation in all sequences due to the dynamic contents in the scenes.



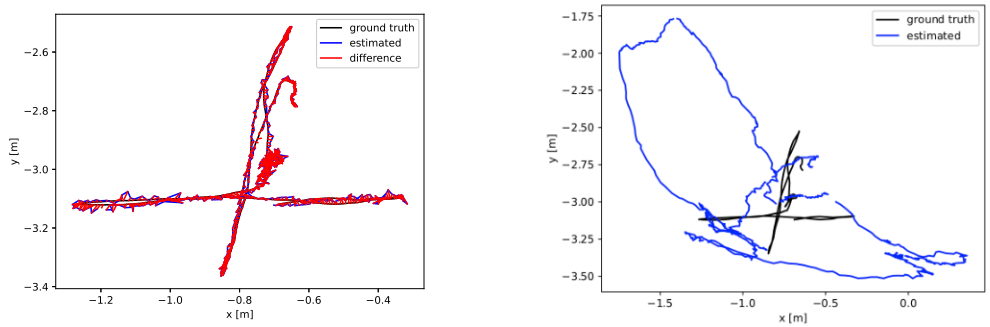
(a) Panotic-SLAM (b) ORB-SLAM3

Figure 5.5: Comparison between the ground-truth and the trajectory estimated by Panoptic-SLAM and ORB-SLAM3 in fr3_w_rpy sequence



(a) Panoptic-SLAM (b) ORB-SLAM3

Figure 5.6: Comparison between the ground-truth and the trajectory estimated by Panoptic-SLAM and ORB-SLAM3 in fr3_w_halfsphere sequences



(a) Panoptic-SLAM (b) ORB-SLAM3

Figure 5.7: Comparison between the ground-truth and the trajectory estimated by Panoptic-SLAM and ORB-SLAM3 in fr3_w_xyz sequence

Table 5.2 shows the ATE comparison between our system and several methods from the literature. The best and second-best results are highlighted in bold and underlined, respectively. Based on the comparison results, our system achieved a similar accuracy of DynaSLAM. This is likely due to the fact that DynaSLAM filters people in advance, whereas the TUM fr3 walking dataset primarily has people moving. Ji *et al.* [10] has an inferior accuracy than ours, despite also being robust to unknown labels in the environments. Our method also outperforms Zhu *et al.* [11], that also uses panoptic segmentation. Overall, Panoptic-SLAM achieved the best results in the most challenging sequences.

Table 5.2: Comparison of the RMSE of ATE [m] of the proposed method against ORB-SLAM3, ReFusion, DynaSLAM, DS-SLAM, SaD-SLAM, DOT-Mask, Ji *et al.* [10], and Zhu *et al.* [11] using the TUM dataset

Sequence	*fr3_w_static*	*fr3_w_xyz*	*fr3_w_rpy*	*fr3_w_half*
Ours	0.009	**0.014**	**0.032**	**0.025**
ORB-SLAM3	0.038	0.819	0.957	0.315
ReFusion	0.017	0.099	—	0.104
DynaSLAM	**0.006**	0.015	0.035	**0.025**
DS-SLAM	0.008	0.024	0.444	0.030
SaD-SLAM	0.017	0.017	**0.032**	0.026
DOTMask	0.008	0.021	0.053	0.040
Ji *et al.* [10]	0.011	0.020	0.037	0.029
Zhu *et al.* [11]	0.013	0.018	0.039	0.030

5.5 Bonn Dataset

The Bonn Dynamic Dataset [100] is used to evaluate the robustness against moving objects, people and non-labeled moving objects. It also uses the same evaluation metrics from the TUM dataset. Six sequences, filmed inside an indoor environment, were chosen from the dataset for evaluation: balloon, ballon2, non-obstructing box (non-obst box) 1 and 2, placing non-obstructing box (placing_no_box) 1 and 2. The balloon sequences show a person walking and interacting with a floating balloon, as shown in Fig 5.8. In the non-obst box sequence, a person moves a cardboard box from one place to another, as shown in Fig. 5.9. In the placing_no_box sequences, shown in Fig 5.10, a person appears and places a card box on the floor. The card box

was not previously in the scene. It is important to state that both balloon and cardboard box classes are not present in the COCO dataset and, consequently, cannot be explicitly detected by our segmentation model.

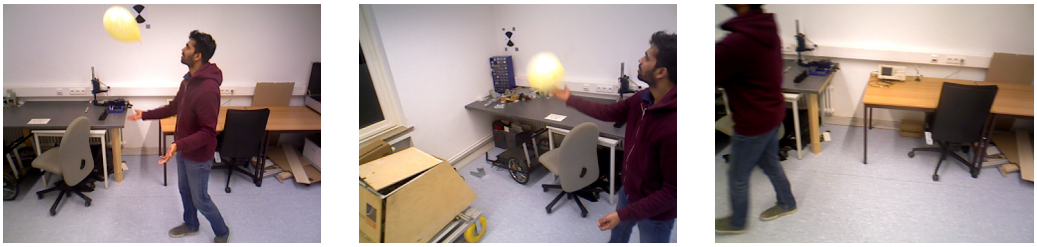


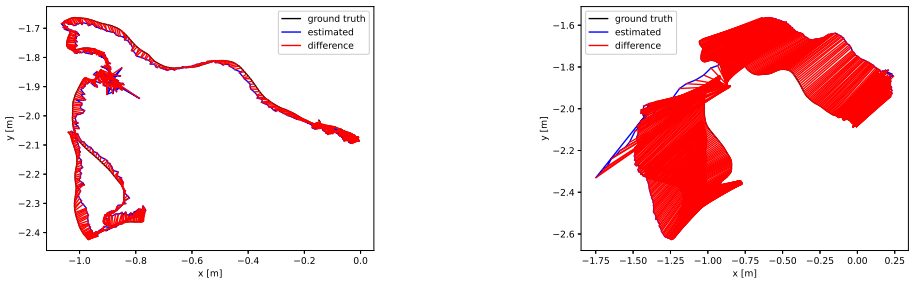
Figure 5.8: Images from the BONN ballon sequence

Figure 5.9: Images from the BONN non-obstructing box sequence

Figure 5.10: Images from the BONN placing_no_box sequence

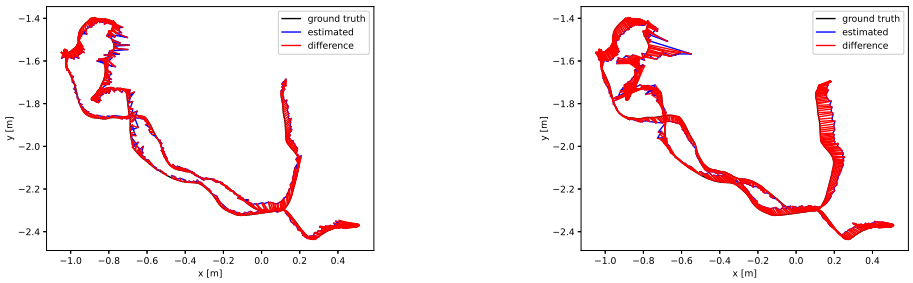
Figures 5.11 and 5.12 show the comparison between the ground-truth and estimated trajectories of ORB-SLAM3 and our system, respectively, in both non-obst box sequences. In these two sequences, a person moves a static box that was previously filmed by the camera from one place to another. ORB-SLAM3 completely deviates from the ground-truth, while Panoptic-SLAM was could correctly estimate the camera pose. Figures 5.13 and 5.14 show the results in both balloons sequences. The man and the floating balloon appear and disappear multiple times in the sequence. Panoptic-SLAM is able to perform a correct estimation, opposed to ORB-SLAM3. Finally, in Figs. 5.15 and 5.16 is presented the results from the placing_no_box sequences. A

man appears and places a box on the floor. Again, Panoptic-SLAM was able to perform correct estimation and ORB-SLAM3 was not successful.



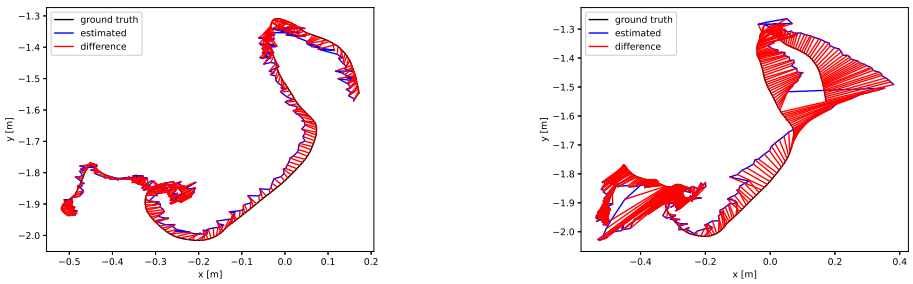
(a) Panoptic-SLAM (b) ORB-SLAM 3

Figure 5.11: Comparison between the ground-truth and the trajectory estimated by ORB-SLAM3 and our system in the non-obstructing box sequence of the Bonn dynamic dataset



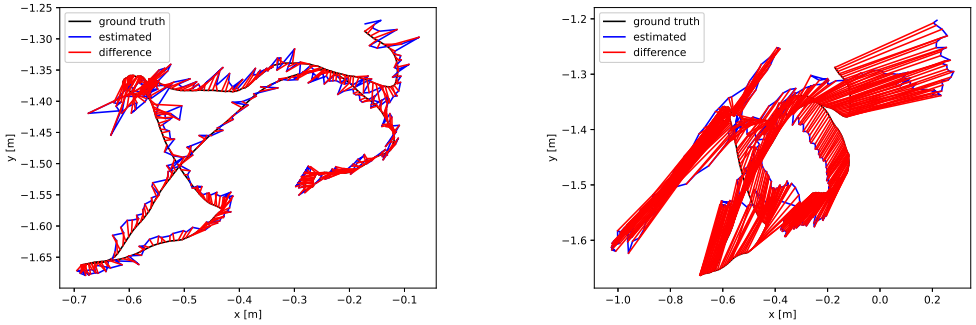
(a) Panoptic-SLAM (b) ORB-SLAM 3

Figure 5.12: Comparison between the ground-truth and the trajectory estimated by ORB-SLAM3 and Panoptic-SLAM in the non-obstructing box 2 sequence of the Bonn dynamic dataset



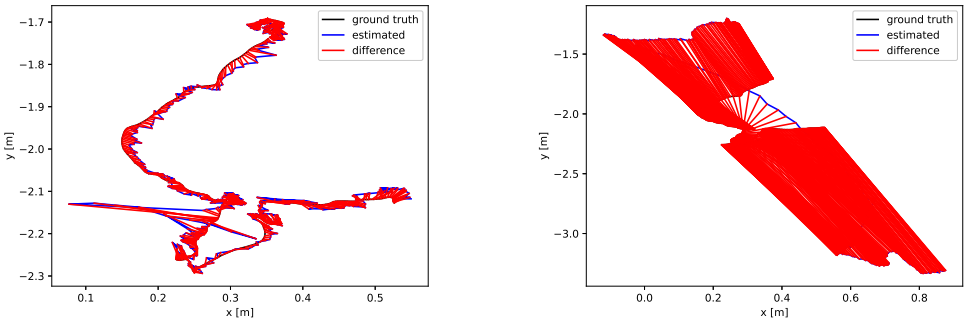
(a) Panoptic-SLAM (b) ORB-SLAM 3

Figure 5.13: Comparison between the ground-truth and the trajectory estimated by ORB-SLAM3 and Panoptic-SLAM in the balloon 1 sequence of the Bonn dynamic dataset



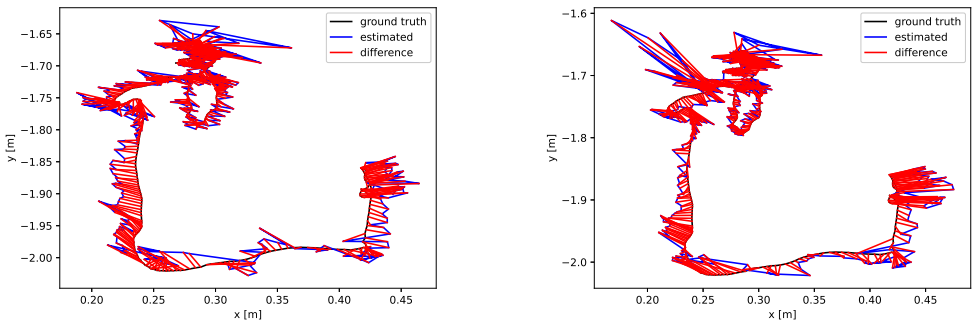
(a) Panoptic-SLAM (b) ORB-SLAM 3

Figure 5.14: Comparison between the ground-truth and the trajectory estimated by ORB-SLAM3 and Panoptic-SLAM in the balloon 2 sequence of the Bonn dynamic dataset



(a) Panoptic-SLAM (b) ORB-SLAM 3

Figure 5.15: Comparison between the ground-truth and the trajectory estimated by ORB-SLAM3 and Panoptic-SLAM in the Placing non obstruction box (placing_no_box) sequence of the Bonn dynamic dataset



(a) Panoptic-SLAM (b) ORB-SLAM 3

Figure 5.16: Comparison between the ground-truth and the trajectory estimated by ORB-SLAM3 and Panoptic-SLAM in the placing non obstruction box 2(placing_no_box2) sequence of the Bonn dynamic dataset

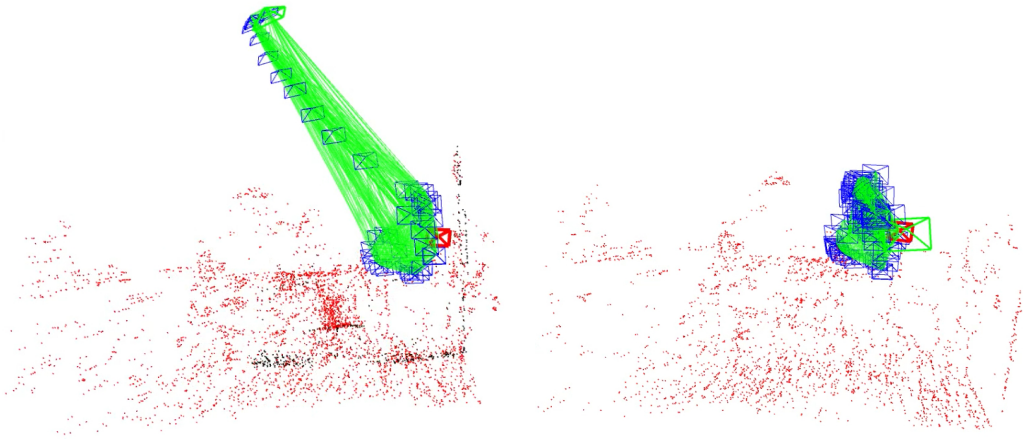
Table 5.3 shows the RMSE of the ATE comparison between our system and ORB-SLAM3, DynaSLAM [21] and ReFusion [53]. The results of ReFusion and DynaSLAM were obtained in [53]. Our system outperformed ORB-SLAM3 in every sequence. Our system also outperformed the other systems in every sequence, except for the last one, where it achieved similar results to DynaSLAM, with a difference of 1 mm. Despite the fact that DynaSLAM achieved similar results to ours in the TUM dataset, the same does not occur in the Bonn dataset due to the presence of unknown moving objects. This is evident in the non-obst box and placing-no-box results, where the error of DynaSLAM is ten times larger than ours in orders of magnitude.

Table 5.3: Comparison of the RMSE of ATE [m] of the proposed method against ORB-SLAM3, DynaSLAM, and ReFusion using the Bonn Dynamic dataset

Sequence	Ours	ORB-SLAM3	DynaSLAM	ReFusion
non-obst box	**0.027**	0.347	0.232	0.071
non-obst box2	**0.033**	0.043	0.039	0.179
balloon	**0.029**	0.092	0.030	0.175
balloon2	**0.027**	0.215	0.029	0.254
placing_no_box	**0.044**	0.842	0.575	0.106
placing_no_box2	0.022	0.023	**0.021**	0.141

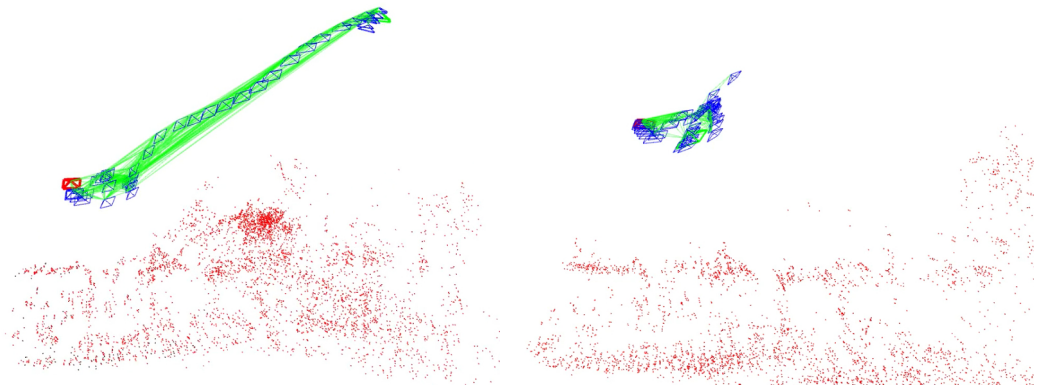
5.6 Final output comparison

Besides the ATE results, we can also notice qualitative improvements of our system by analyzing the SLAM final output. In Fig. 5.17, it is compared the final outputs of Panoptic-SLAM and ORB-SLAM3 in the non-obst box sequence after the person left the scene. ORB-SLAM3 had a big drift due to the dynamic content, while the Panoptic-SLAM maintained the camera pose correctly. Moreover, the sparse map also suffers from ORB-SLAM3 drift, resulting in a wrong output map. Similar results in relation to drift and sparse map creation can be seen in Fig. 5.18.



(a) ORB-SLAM 3 (b) Panoptic-SLAM

Figure 5.17: Comparison between the estimated camera trajectory by ORB-SLAM3 and Panoptic-SLAM in the non-obstructing box sequence of the Bonn dynamic dataset



(a) ORB-SLAM 3 (b) Panoptic-SLAM

Figure 5.18: Comparison between the estimated camera trajectory by ORB-SLAM3 and Panoptic-SLAM in the placing non obstruction box sequence of the Bonn dynamic dataset

5.7 Evaluation of different System configuration

To show the importance of each step of our methodology, experiments were made using three different configurations of the system: using only the people filter, the people filter together with the known moving object filter, and the people filter together with the unknown moving object filter. Two challenging sequences of the Bonn dataset that contain unknown moving objects were chosen for the evaluation: non-obst box and placing-non-box. Table

III shows the RMSE of the ATE of the different configurations, compared to the combined system. Due to the presence of both known and unknown moving objects in these scenes, every configuration fails except for the combined system.

Table 5.4: Evaluation of the ATE on the Bonn dynamic dataset using Panoptic-SLAM with different configurations [m]

Sequence	non-obst box	placing_no_box
People filter	0.481	0.707
Moving object filter	0.029	0.765
Unknown object filter	0.302	0.721
Panoptic-SLAM	**0.027**	**0.044**

5.8 Run-time Analysis

All tests were performed on a notebook with Intel i7 CPU with 16 GB of RAM and NVIDIA RTX3060 GPU with 8 GB of VRAM running Ubuntu 20.04 LTS Linux. Table 5.5 shows the mean tracking time of the system in four sequences of the TUM dataset. The average of the panoptic segmentation inference time was 0.2 second for every sequence.

Table 5.5: Mean tracking time [s]

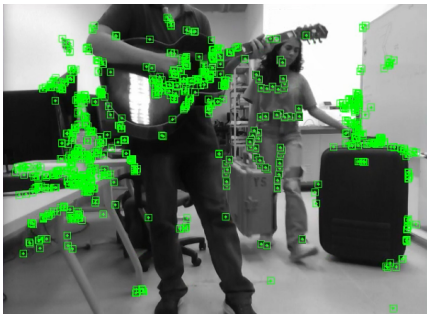
Sequence	Mean tracking time [s]
fr3_w_static	0.344
fr3_w_xyz	0.344
fr3_w_rpy	0.319
fr3_w_half	0.323

5.9 Experiments

Besides the Bonn and TUM datasets, Panoptic-SLAM was also tested in two custom datasets. Both datasets were filmed at the Robotics Lab (LabRob) at PUC-Rio, using an Intel RealSense D435i RGB-D camera [101] attached to a tripod, shown in Fig. 5.19.

Figure 5.19: Intel RealSense D435i camera

The idea is to create a dynamic sequence where people move around the environment and interact with multiple objects, including unlabeled objects (e.g., a guitar). Due to the absence of a motion capture system, the camera was maintained static. In Fig. 5.20 shows the results of ORB-SLAM3 in two frames from the first sequence. In Fig. 5.20a, a man is moving and holding a guitar while a woman is walking and carrying a suitcase. In Fig. 5.20b the man continues his behavior, while the woman had the suitcase placed on the ground and was moving with the luggage. From the images, we notice that ORB-SLAM3 used the features from static and dynamic contents in the scene to estimate the camera pose.



(a) ORB-SLAM3 custom dataset result 1

(b) ORB-SLAM3 custom dataset result 2

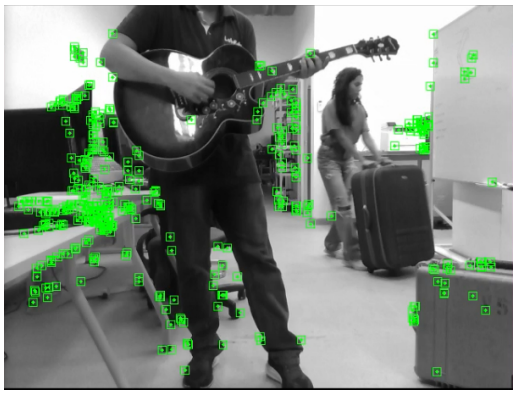
Figure 5.20: Results from ORB-SLAM3 in first experiment dataset sequence

In Fig. 5.21 is shown the results from Panoptic-SLAM in the same

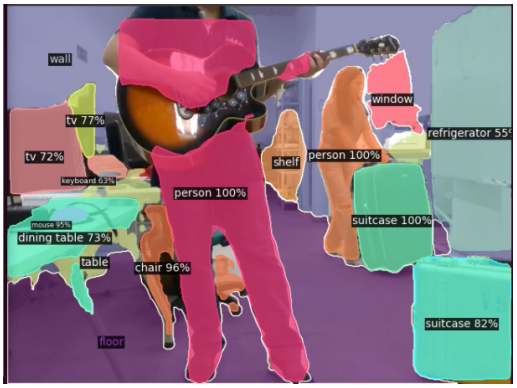
sequence. Figures 5.21a and 5.21c represent the same event as previously described, and Figs. 5.21b and 5.21d are the associated panoptic masks. From the segmentation masks, we notice that the guitar is not recognized by the panoptic model and even was erroneously labeled as a pair of scissors. However, despite the lack of the information about the object or wrong inference label, Panoptic-SLAM was capable of detecting it as a moving object and filter its keypoints. The suitcase that the woman is carrying shows another interesting behavior from our system. When it was moving with the woman, the keypoints were filtered due to the dynamic content. Once it became static, the keypoints were no more filtered and used for the pose estimation.

(a) Panoptic-SLAM custom dataset result 1

(b) Panoptic-SLAM masks result 1



(c) Panoptic-SLAM custom dataset result 2

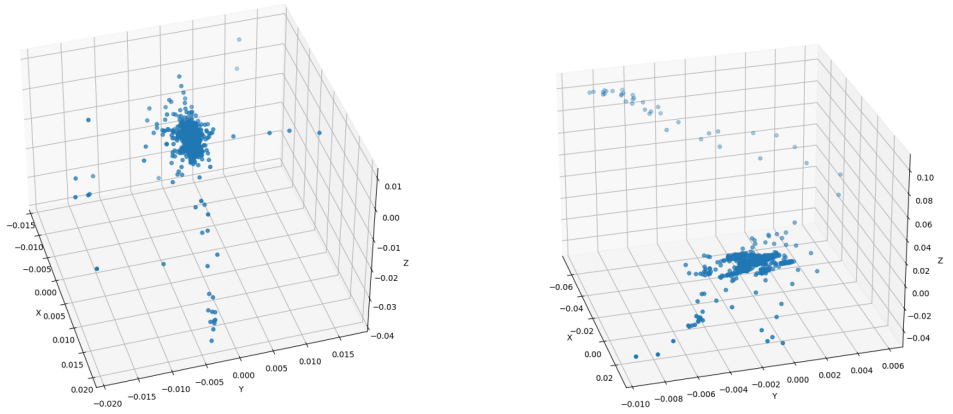


(d) Panoptic-SLAM masks result 2

Figure 5.21: Results from Panoptic-SLAM in custom dataset

Finally, we can compare the camera pose estimation from Panoptic-SLAM and ORB-SLAM3 in Fig. 5.22. Panoptic-SLAM could correctly estimate the camera pose with a few ouliers, while ORB-SLAM3 resulted in a more scattered estimation with larger errors and more outliers with respect to Panoptic-SLAM. Even though the results from Panoptic-SLAM were more accurate than ORB-SLAM3, both system could estimate the camera pose near

the origin for this dataset.



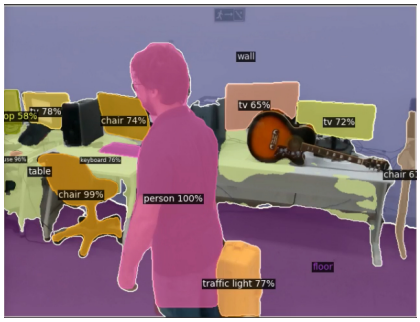
(a) Panoptic-SLAM (b) ORB-SLAM3

Figure 5.22: Camera pose estimation from Panoptic-SLAM and ORB-SLAM3 in the first experiment

The second experiment consists of multiple people walking in random directions carrying a suitcase and a guitar as unlabeled objects. This dataset, shown in Fig. 5.23, simulates a highly dynamic environment with people appearing/disappearing from the scene, occlusions and object interactions.

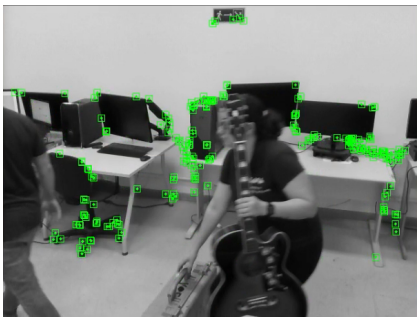
Figure 5.23: Second experiment dataset sequence

This experiment aims to show the ability of the proposed method to deal with unknown moving objects. The sequence starts with the guitar static on the table. Figure 5.24a shows the panoptic inference where the guitar does not have a mask, indicating an unknown object. Figure 5.24b shows the same image with the associate keypoints. Since the guitar is not moving, the Panoptic-SLAM system is using the keypoints to estimate the camera pose. In another frame, the guitar is being carried by a woman. The guitar is detected as a moving object and the keypoints are filtered even though this object is unknown by the panoptic model, as shown in Fig. 5.24c. In comparison, Fig. 5.24d presents the result from ORB-SLAM3 where both the woman and the guitar have detected keypoints.

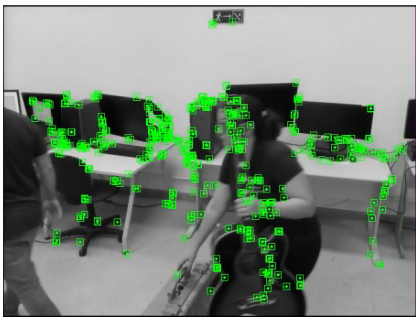


(a) Frame with static guitar-Panoptic-SLAM panoptic masks

(b) Frame with static guitar-Panoptic-SLAM keypoints



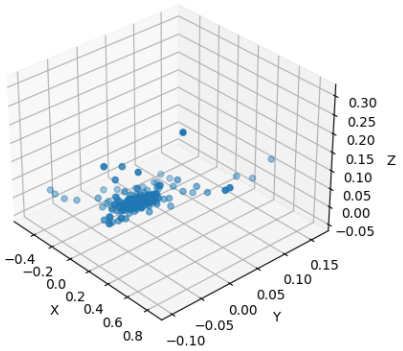
(c) Frame with moving guitar-Panoptic-SLAM keypoints



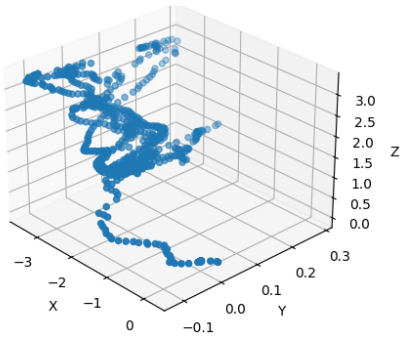
(d) Frame with moving guitar-ORB-SLAM3 keypoints

Figure 5.24: Unknown moving object filter in second experiment

Figure 5.25 displays the camera pose estimation from Panoptic-SLAM and ORB-SLAM3 in the second experiment. The Tab. 5.6 presents the statistics (average and standard deviation values) from the same experiment. The bold values indicate the best results. The resulting data shows that the Panoptic-SLAM correctly estimated the camera pose as static with small errors for this challenging dataset. ORB-SLAM3, on the other hand, drifted meters away from the origin due to the dynamic content.



(a) Panoptic-SLAM



(b) ORB-SLAM3

Figure 5.25: Camera pose estimation from Panoptic-SLAM and ORB-SLAM3 in the second experiment

Table 5.6: Evaluation of pose camera statistics [mm]

Statistic	Panoptic-SLAM	ORB-SLAM3
X mean	**6.236**	-2000.421
Y mean	**0.4181**	45.356
Z mean	**2.274**	1659.054
X std	**76.842**	981.023
Y std	**16.271**	73.520
Z std	**19.338**	826.440

The moving filters from Panoptic-SLAM are not flawless. There are some situations when the moving keypoints are not filtered. For instance, when an object appears as unknown at first, but becomes a known object in the next frame, only to become unknown again in the following frame. In order to deal with such situations, a buffer can be implemented to detect the consistency of the objects in the frame.

6
Conclusion and Future work

6.1
Conclusion

This master thesis presented a visual SLAM system built on ORB-SLAM3 that operates online and is robust in dynamic environments, even in the presence of unknown and unlabeled moving objects. Panoptic-SLAM is a pioneer work in SLAM with panoptic segmentation and the first one to be open-source. The proposed methodology included the ORB-SLAM3 framework, a PanopticFPN panoptic segmentation model and dynamic keypoint filtering based on panoptic content with epipolar constrains. Since our system uses only an RGB image as input for keypoint filtering, Panoptic-SLAM can be used for monocular, stereo and RGB-D SLAM systems. We show the effectiveness of our method by comparing it with several systems from the literature that are considered to have state-of-the-art results in dynamic environments, including DynaSLAM, DS-SLAM and SaD-SLAM on challenging dynamic sequences from TUM and BONN evaluation datasets. The results indicate that our system not only achieves the same levels of accuracy than DynaSLAM in highly dynamic scenes, but also surpasses it by a high margin in scenarios with unknown moving objects. Furthermore, Panoptic-SLAM obtained better results in all evaluated TUM sequences in relation to Zhu *et al.* [11], a system that also utilize panoptic segmentation for dynamic keypoint filtering and is robust against unknown objects. Last, Panoptic-SLAM was compared to ORB-SLAM3 in two sequences from a custom dataset. Both sequences contain highly dynamic scenes with an unknown moving object. From the experiments, our system significantly outperformed ORB-SLAM3.

6.2
Future Works

This thesis has many paths for improvement. First, we will test the method with other panoptic segmentation architectures, especially the ones based on transformers, to improve the inference performance and accuracy. Moreover, we plan to convert the panoptic segmentation model to be executed by a C++ application. The idea is to implement the whole system using only C++, dismissing the Python-C-API and improving in execution time. Furthermore, the methods proposed are not prepared to handle objects that

are deformable or rigid objects that can change their shape, such as doors or closets.

Another future work is to test the proposed methodology in more challenging sequences, such as crowded environments with multiple unknown objects. Also, we plan to deal with moving objects that occupy a large portion of the image in order to avoid lost track. In addition, we aim to include in the method a semantic map building system that can adapt over time, exploring even further capabilities of the panoptic segmentation.

Finally, investigating the application of modern technologies such as event cameras would be a compelling research direction. It is worthwhile to examine their benefits in dynamic environments and explore the potential of integrating them with the suggested algorithms.

6.3 Publications

The work from this thesis generated the following paper submitted to the 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2023):

– Abati G.F., Soares J.C.V., Meggiolaro M.A., "Panoptic-SLAM: Visual SLAM in Dynamic Environments using Panoptic Segmentation", in 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems 2023

The following paper related to SLAM in changing environments was submitted and is currently under review:

– Soares J.C.V., Medeiros V.S., Abati G.F., Becker M., Caurin G., Gattass M., Meggiolaro M., "Visual Localization and Mapping in Dynamic and Changing Environments"

Other publications related to mobile robotics were also published in international conferences:

– 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, Au- tonomous 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.

7
Bibliography

[1] 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)*, pp. 135–140, 2019.

[2] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer Science & Business Media, 2010.

[3] X. Wu, D. Sahoo, and S. C. Hoi, "Recent advances in deep learning for object detection," *Neurocomputing*, vol. 396, no. d, pp. 39–64, 2020.

[4] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 1137–1149, 2015.

[5] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2015.

[6] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *ArXiv*, vol. abs/1505.04597, 2015.

[7] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask r-cnn," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988, 2017.

[8] X. Li, T. Lai, S. Wang, Q. Chen, C. Yang, R. Chen, J. Lin, and F. Zheng, "Weighted feature pyramid networks for object detection," *2019 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, pp. 1500–1504, 2019.

[9] C. Campos, R. Elvira, J. J. G. Rodr'iguez, 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*, vol. 37, pp. 1874–1890, 2020.

[10] 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)*, pp. 11175–11181, 2021.

[11] H. Zhu, C. Yao, Z. Zhu, Z. Liu, and Z. Jia, "Fusing panoptic segmentation and geometry information for robust visual slam in dynamic environments," *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*, pp. 1648–1653, 2022.

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

[13] J. J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European Conference on Computer Vision*, 2014.

[14] 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*, vol. 33, pp. 1255–1262, 2016.

[15] J. J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, pp. 611–625, 2016.

[16] 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*, vol. 102, 2021.

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

[18] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *ArXiv*, vol. abs/1804.02767, 2018.

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

[20] Y. Liu and J. Miura, "Rds-slam: Real-time dynamic slam using semantic segmentation methods," *IEEE Access*, vol. 9, pp. 23772–23785, 2021.

[21] B. Bescós, J. M. Fácil, J. Civera, and J. Neira, "Dynaslam: Tracking, mapping, and inpainting in dynamic scenes," *IEEE Robotics and Automation Letters*, vol. 3, pp. 4076–4083, 2018.

[22] X. Yuan and S. Chen, "Sad-slam: A visual slam based on semantic and depth information," *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4930–4935, 2020.

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

[24] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "Yolact++ better real-time instance segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, pp. 1108–1121, 2019.

[25] T. Bailey and H. Durrant-whyte, "Simultaneous Localization and Mapping: Part 1," *Robotics and Automation magazine*, vol. 13(2), no. 3, pp. 99–110, 2006.

[26] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. Cambridge, Mass.: MIT Press, 2005.

[27] T. Bailey, J. I. Nieto, J. E. Guivant, M. Stevens, and E. M. Nebot, "Consistency of the ekf-slam algorithm," *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3562–3568, 2006.

[28] S. Huang and G. Dissanayake, "Convergence analysis for extended kalman filter based slam," *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pp. 412–417, 2006.

[29] J. A. Castellanos, J. Neira, and J. D. Tardós, "Limits to the consistency of ekf-based slam," *IFAC Proceedings Volumes*, vol. 37, pp. 716–721, 2004.

[30] S. J. Julier and J. K. Uhlmann, "A counter example to the theory of simultaneous localization and map building," *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, vol. 4, pp. 4238–4243 vol.4, 2001.

[31] S. Thrun, M. Montemerlo, D. Koller, B. Wegbreit, J. I. Nieto, and E. M. Nebot, "Fastslam: An efficient solution to the simultaneous localization and mapping problem with unknown data," 2004.

[32] K. P. Murphy, "Bayesian map learning in dynamic environments," in *NIPS*, 1999.

[33] G. Grisetti, G. D. Tipaldi, C. Stachniss, W. Burgard, and D. Nardi, "Fast and accurate slam with rao-blackwellized particle filters," *Robotics Auton. Syst.*, vol. 55, pp. 30–38, 2007.

[34] M. Montemerlo and S. Thrun, "Simultaneous localization and mapping with unknown data association using fastslam," *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 2, pp. 1985–1991 vol.2, 2003.

[35] T. Bailey, J. Nieto, and E. Nebot, "Consistency of the FastSLAM algorithm," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2006, no. 1, pp. 424–429, 2006.

[36] S. Thrun and M. Montemerlo, "The graph slam algorithm with applications to large-scale mapping of urban structures," *The International Journal of Robotics Research*, vol. 25, pp. 403 – 429, 2006.

[37] 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)*, pp. 13813–13819, 2021.

[38] B. K. P. Horn and B. G. Schunck, "Determining optical flow," in *Other Conferences*, 1981.

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

[40] Z. Zhang, *Iterative Closest Point (ICP)*, pp. 433–434. Boston, MA: Springer US, 2014.

[41] A. J. Davison, I. D. Reid, N. Molton, and O. Stasse, "Monoslam: Real-time single camera slam.," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 1052–1067, 2007.

[42] G. S. W. Klein and D. W. Murray, "Parallel tracking and mapping for small ar workspaces," *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 225–234, 2007.

[43] P. Henry, M. Krainin, E. V. 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*, vol. 31, pp. 647 – 663, 2012.

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

[45] 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*, vol. 36, pp. 416 – 446, 2018.

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

[47] 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)*, pp. 6253–6259, 2019.

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

[49] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence-Volume 2*, pp. 674–679, William Kaufmann Publishers, 1981.

[50] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, pp. 381–395, 1981.

[51] P. F. Alcantarilla, J. J. Y. Torres, 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," *2012 IEEE International Conference on Robotics and Automation*, pp. 1290–1297, 2012.

[52] Y. Sun, M. Liu, and M. Q. Meng, "Improving rgb-d slam in dynamic environments: A motion removal approach," *Robotics Auton. Syst.*, vol. 89, pp. 110–122, 2017.

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

[54] R. Scona, M. Jaimez, Y. R. Pétillot, M. F. Fallon, and D. Cremers, "Staticfusion: Background reconstruction for dense rgb-d slam in dynamic environments," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–9, 2018.

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

[56] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European Conference on Computer Vision*, 2015.

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

[58] I. Khokhlov, E. Davydenko, I. Osokin, I. Ryakin, A. Babaev, V. Litvinenko, and R. Gorbachev, "Tiny-yolo object detection supplemented with geometrical data," *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, pp. 1–5, 2020.

[59] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 2481–2495, 2015.

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

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

[62] A. Kirillov, K. He, R. B. Girshick, C. Rother, and P. Dollár, "Panoptic segmentation," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9396–9405, 2018.

[63] W. Ye, X. Lan, S. Chen, Y. Ming, X. rong Yu, H. Bao, Z. Cui, and G. Zhang, "Pvo: Panoptic visual odometry," *ArXiv*, vol. abs/2207.01610, 2022.

[64] Z. Yuan, K. Xu, X. Zhou, B. Deng, and Y. Ma, "Svg-loop: Semantic-visual-geometric information-based loop closure detection," *Remote. Sens.*, vol. 13, p. 3520, 2021.

[65] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, pp. 1330–1334, 2000.

[66] C. G. Harris and M. J. Stephens, "A combined corner and edge detector," in *Alvey Vision Conference*, 1988.

[67] G. LoweDavid, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, 2004.

[68] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," in *European Conference on Computer Vision*, 2006.

[69] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European Conference on Computer Vision*, 2006.

[70] C. V. Stewart, "Robust parameter estimation in computer vision," *SIAM Reviews*, 1999.

[71] T.-Y. Lin, M. Maire, S. J. 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*, 2014.

[72] 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)*, pp. 5079–5085, 2016.

[73] 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 Auton. Syst.*, vol. 117, pp. 1–16, 2019.

[74] D. de Geus, P. Meletis, and G. Dubbelman, "Panoptic segmentation with a joint semantic and instance segmentation network," *ArXiv*, vol. abs/1809.02110, 2018.

[75] Y. Li, X. Chen, Z. Zhu, L. Xie, G. Huang, D. Du, and X. Wang, "Attention-guided unified network for panoptic segmentation," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7019–7028, 2018.

[76] Y. Xiong, R. Liao, H. Zhao, R. Hu, M. Bai, E. Yumer, and R. Urtasun, "Upsnet: A unified panoptic segmentation network," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8810–8818, 2019.

[77] A. Kirillov, R. B. Girshick, K. He, and P. Dollár, "Panoptic feature pyramid networks," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6392–6401, 2019.

[78] J. Li, A. Raventos, A. Bhargava, T. Tagawa, and A. Gaidon, "Learning to fuse things and stuff," *ArXiv*, vol. abs/1812.01192, 2018.

[79] H. Liu, C. Peng, C. Yu, J. Wang, X. Liu, G. Yu, and W. Jiang, "An end-to-end network for panoptic segmentation," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6165–6174, 2019.

[80] Y. Yang, H. Li, X. Li, Q. Zhao, J. Wu, and Z. Lin, "Sognet: Scene overlap graph network for panoptic segmentation," in *AAAI Conference on Artificial Intelligence*, 2019.

[81] J. Lazarow, K. Lee, and Z. Tu, "Learning instance occlusion for panoptic segmentation," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10717–10726, 2019.

[82] H. Wang, R. Luo, M. Maire, and G. Shakhnarovich, "Pixel consensus voting for panoptic segmentation," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9461–9470, 2020.

[83] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," *ArXiv*, vol. abs/2005.12872, 2020.

[84] B. Cheng, A. G. Schwing, and A. Kirillov, "Per-pixel classification is not all you need for semantic segmentation," in *Neural Information Processing Systems*, 2021.

[85] R. Hou, J. Li, A. Bhargava, A. Raventos, V. C. Guizilini, C. Fang, J. P. Lynch, and A. Gaidon, "Real-time panoptic segmentation from dense detections," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8520–8529, 2019.

[86] B. Cheng, M. D. Collins, Y. Zhu, T. Liu, T. S. Huang, H. Adam, and L.-C. Chen, "Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12472–12482, 2019.

[87] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *European Conference on Computer Vision*, 2018.

[88] H. Wang, Y. Zhu, B. Green, H. Adam, A. L. Yuille, and L.-C. Chen, "Axial-deeplab: Stand-alone axial-attention for panoptic segmentation," in *European Conference on Computer Vision*, 2020.

[89] H. Wang, Y. Zhu, H. Adam, A. L. Yuille, and L.-C. Chen, "Max-deeplab: End-to-end panoptic segmentation with mask transformers," *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5459–5470, 2020.

[90] Q. Yu, H. Wang, D. Kim, S. Qiao, M. D. Collins, Y. Zhu, H. Adam, A. L. Yuille, and L.-C. Chen, "Cmt-deeplab: Clustering mask transformers for panoptic segmentation," *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2550–2560, 2022.

[91] M. J. Cummins and P. Newman, "Fab-map: Probabilistic localization and mapping in the space of appearance," *The International Journal of Robotics Research*, vol. 27, pp. 647 – 665, 2008.

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

[93] D. Gálvez-López and J. D. Tardós, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, pp. 1188–1197, 2012.

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

[95] M. Kaess, A. Ranganathan, and F. Dellaert, "isam: Incremental smoothing and mapping," *IEEE Transactions on Robotics*, vol. 24, pp. 1365–1378, 2008.

[96] R. Kümmerle, G. Grisetti, H. M. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," *2011 IEEE International Conference on Robotics and Automation*, pp. 3607–3613, 2011.

[97] Z. Chen and L. Liu, "Creating navigable space from sparse noisy map points," *ArXiv*, vol. abs/1903.01503, 2019.

[98] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2." `https://github.com/facebookresearch/detectron2`, 2019.

[99] M. Salas, Y. Latif, I. D. Reid, and J. Montiel, "Trajectory alignment and evaluation in slam: Horn's method vs alignment on the manifold," 2015.

[100] E. Palazzolo, J. Behley, P. Lottes, P. Giguère, and C. Stachniss, "ReFusion: 3D Reconstruction in Dynamic Environments for RGB-D Cameras Exploiting Residuals," in *iros*, 2019.

[101] A. Zabatani, V. Surazhsky, E. Sperling, S. B. Moshe, O. Menashe, D. H. Silver, Z. Karni, A. M. Bronstein, M. M. Bronstein, and R. Kimmel, "Intel® realsense™ sr300 coded light depth camera," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, pp. 2333–2345, 2020.