KTH ROYAL INSTITUTE
OF TECHNOLOGY

Degree Project in Computer Science

Second cycle, 30 credits

Analyzing different approaches to Visual SLAM in dynamic environments

A comparative study with focus on strengths and weaknesses

KRISTÍN SÓL ÓLAFSDÓTTIR

Stockholm, Sweden, 2023

Analyzing different approaches to Visual SLAM in dynamic environments

A comparative study with focus on strengths and weaknesses

KRISTÍN SÓL ÓLAFSDÓTTIR

Master's Programme, Systems, Control and Robotics, 120 credits
Date: July 12, 2023

Supervisor: Patric Jensfelt
Examiner: Jana Tumova
School of Electrical Engineering and Computer Science
Swedish title: Analys av olika metoder för Visual SLAM i dynamisk miljö
Swedish subtitle: En jämförande studie med fokus på styrkor och svagheter

Abstract

Simultaneous Localization and Mapping (SLAM) is the crucial ability for many autonomous systems to operate in unknown environments. In recent years SLAM development has focused on achieving robustness regarding the challenges the field still faces e.g. dynamic environments.

During this thesis work different existing approaches to tackle dynamics with Visual SLAM systems were analyzed by surveying the recent literature within the field. The goal was to define the advantages and drawbacks of the approaches to provide further insight into the field of dynamic SLAM. Furthermore, two methods of different approaches were chosen for experiments and their implementation was documented.

Key conclusions from the literature survey and experiments are the following. The exclusion of dynamic objects with regard to camera pose estimation presents promising results. Tracking of dynamic objects provides valuable information when combining SLAM with other tasks e.g. path planning. Moreover, dynamic reconstruction with SLAM offers better scene understanding and analysis of objects' behavior within an environment. Many solutions rely on pre-processing and heavy hardware requirements due to the nature of the object detection methods. Methods of motion confirmation of objects lack consideration of camera movement, resulting in static objects being excluded from feature extraction.

Considerations for future work within the field include accounting for camera movement for motion confirmation and producing available benchmarks that offer evaluation of the SLAM result as well as the dynamic object detection i.e. ground truth for both camera and objects within the scene.

Keywords

Visual SLAM, RGB-D Vision, Dynamic Objects, Object Detection, Multi-Object Tracking, Image Segmentation, Optical Flow

Sammanfattning

Simultaneous Localization and Mapping (SLAM) är för många autonoma system avgörande för deras förmåga att kunna verka i tidigare outforskade miljöer. Under de senaste åren har SLAM-utvecklingen fokuserat på att uppnå robusthet när det gäller de utmaningar som fältet fortfarande står inför, t.ex. dynamiska miljöer.

I detta examensarbete analyserades befintliga metoder för att hantera dynamik med visuella SLAM-system genom att kartlägga den senaste litteraturen inom området. Målet var att definiera för- och nackdelar hos de olika tillvägagångssätten för att bidra med insikter till området dynamisk SLAM. Dessutom valdes två metoder från olika tillvägagångssätt ut för experiment och deras implementering dokumenterades.

De viktigaste slutsatserna från litteraturstudien och experimenten är följande. Uteslutningen av dynamiska objekt vid uppskattning av kamerans position ger lovande resultat. Spårning av dynamiska objekt ger värdefull information när SLAM kombineras med andra uppgifter, t.ex. path planning. Dessutom ger dynamisk rekonstruktion med SLAM bättre förståelse om omgivningen och analys av objekts beteende i den kringliggande miljön. Många lösningar är beroende av förbehandling samt ställer höga hårdvarumässiga krav till följd av objektdetekteringsmetodernas natur. Metoder för rörelsebekräftelse av objekt tar inte hänsyn till kamerarörelser, vilket leder till att statiska objekt utesluts från funktionsextraktion.

Uppmaningar för framtida studier inom området inkluderar att ta hänsyn till kamerarörelser under rörelsebekräftelse samt att ta ändamålsenliga riktmärken för att möjliggöra tydligare utvärdering av SLAM-resultat såväl som för dynamisk objektdetektion, dvs. referensvärden för både kamerans position såväl som för objekt i scenen.

Nyckelord

Visual SLAM, RGB-D Syn, Dynamiska objekt, Objektdetektering, Multi-Objekt Spårning, Bildsegmentation, Optiskt Flöde

Acknowledgments

I want to express my deepest gratitude to my supervisor Patric Jensfelt for his constant encouragement and support throughout the entire thesis process.

Additionally, I wish to extend my heartfelt thanks to the members of my thesis group Andreas Eriksson, Elliði Ívarsson, Simon Edström, my friends at KTH and my family for their tireless support and encouragement.

Finally, I would like to convey my profound appreciation to my greatest inspirations and role models, my grandmothers Kristín Halla Jónsdóttir and Ásrún Ólafsdóttir.

Stockholm, July 2023
Kristín Sól Ólafsdóttir

Contents

List of Figures

List of Tables

List of acronyms and abbreviations

ATE	Absolute Trajectory Error
DOS	Dynamic Object State
EKF	Extended Kalman Filter
FOV	Field of View
GPU	Graphics Processing Unit
RMSE	Root Mean Square Error
RPE	Relative Pose Error
SfM	Structure from Motion
SLAM	Simultaneous Localization and Mapping
VO	Visual Odometry
VSLAM	Visual Simultaneous Localization and Mapping

Chapter 1

Introduction

Simultaneous Localization and Mapping (SLAM) refers to the process of determining the location of a system with respect to its surroundings while mapping its environment. SLAM is a crucial ability for many autonomous systems operating in environments without external sources of information and can be implemented with various sensors, each with its advantages and drawbacks. Visual Simultaneous Localization and Mapping (VSLAM), using vision-based sensory input, has gained popularity with improved camera technology and computer vision. Moreover, cameras are cheaper, compared to other sensors, and capable of providing rich information.

The focus of the development of VSLAM in the past years has been on achieving robustness as presented in [1]. The corresponding goal is for autonomous systems to operate accurately in challenging environments. However, VSLAM systems still face challenges in various settings as previously the development had mainly focused on static, structured environments, limited in scale. As a result, traditional VSLAM methods quickly fail in challenging environments, such as high-speed pose estimation and dynamic environments.

1.1 Problem

Methods for handling dynamics in VSLAM systems have been explored in recent times and are further discussed in Chapter 2. These methods leverage state-of-the-art methods for dynamic object detection and motion estimation to enhance VSLAM performance in dynamic environments. Many of the methods regard dynamic objects as outliers in the feature extraction process, while others focus on tracking the objects. Lastly, there are methods that aim

to reconstruct the dynamic objects within the environment. The technologies being used to treat dynamic objects are fast developing and it is not obvious what approach to use given the context and purpose. Depending on the application one approach might be more applicable than another. Moreover, a comparative analysis of VSLAM systems aiming for achieved performance in dynamic environments has yet to be explored in the literature.

1.1.1 Research question

Considering the problem and the recent development of dynamic VSLAM the following research question is defined as

What are the advantages and limitations of the different approaches and methods of handling dynamic objects with VSLAM?

Answering the research question should allow researchers in the domain to use the identified advantages and limitations to assist in decision-making when choosing an approach, depending on their application.

1.2 Purpose & Goals

The challenge of dealing with dynamics with VSLAM has been partly researched in different domains and with no easy way to see which approach is best to use depending on the application. The main focus is to analyze the different existing approaches to tackle dynamics with VSLAM systems that aim to achieve improved performance in dynamic environments. The desired outcome is to identify the advantages and drawbacks of the different approaches with the aim of providing valuable guidance for further research in the domain. The goal of this project is to provide a comprehensive analysis of the relevant solutions from the literature. This has been divided into the following three sub-goals:

1. Survey existing solutions and define a set of parameters to evaluate on.
2. Identify the advantages and limitations of the methods.
3. Identify gaps for improvement and practical hints for future research in the field.

Achieving the defined goals will involve performing and presenting a thorough literature review and background study within the field. The literature

review should conclude with the advantages and limitations of the different approaches. Based on the insights obtained from the background study, evaluation parameters will be established. Additionally, appropriate systems will be selected from the review to be used for experimental evaluation. The results from the evaluation will be presented alongside observations made during employment and experiments. Taking into account practical aspects of the chosen systems used for experiments. The conclusions will reflect on the experimental results and the identified advantages and limitations of the approaches in the field.

1.3 Ethical, Societal and Sustainability

The work is within the field of autonomous systems focusing on how SLAM algorithms can improve performance in dynamic environments. Dynamic environments include most often (if not always) living beings i.e. people and animals. One should therefore promote development with ethical and societal aspects in mind as deploying SLAM into systems in an environment undeniably raises ethical and societal concerns. However, autonomous systems can also be used for good; conducting rescue missions and work in dangerous environments. Society has benefited through the deployment of autonomous systems. Slowly monotonous and mundane jobs can be automated while other jobs are created within different industries. When looking at the development of Visual SLAM algorithms using heavy Neural Network architectures, as well as training, one could question the effect on our planet with heavy computational requirements and energy usage. However, aiming the field into lightweight solutions to be deployed on systems would lessen this negative effect.

1.4 Delimitations

The goal of the thesis is to provide guidance for further research, in the field of VSLAM in dynamic environments, as a comparison between the existing systems is lacking. Therefore, developing an implementation of a new method for dynamic VSLAM will not be explored. Rather, chosen representatives for the existing methods will be used for experiments. Methods aiming for improvements with other challenges VSLAM faces will not be explored. Evaluation will be performed on real-world datasets but experiments will not be performed in real life. However, the focus will be on how heavy the

systems are and taken into consideration as an advantage/limitation for real-time application.

1.5 Outline

The thesis is divided into four main parts. Chapter 2 presents the relevant background information regarding VSLAM and computer vision tasks, followed by Chapter 3 which presents a literature review in the field of VSLAM in dynamic environments. Chapter 4 provides detailed descriptions of the systems that will be used for experiments and Chapter 5 the description of the evaluation method. Chapter 6 presents the results from the evaluation and discussion. Finally, Chapter 7 provides conclusions and discussion regarding possible future research.

Chapter 2

Background

This chapter provides the related background information used throughout the thesis.

2.1 Visual SLAM

VSLAM can be grouped based on the sensor; Monocular (one camera), Stereo (two cameras), and RGB-D (camera with depth information) and approached in two different ways. Firstly, using a feature-based approach where important features in the image are detected and tracked. Secondly, using a direct approach where the focus is on the whole image instead of its features.

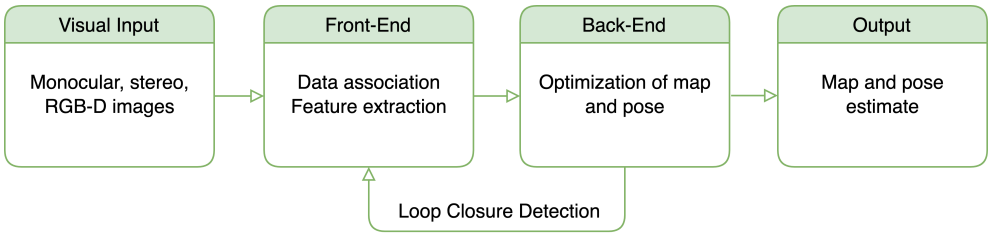


Figure 2.1: A simplified structure of a traditional Visual SLAM pipeline.

Feature-based VSLAM pipelines are traditionally structured of two main parts: the front-end and back-end, seen in Figure 2.1. The front-end typically contains data association, feature extraction, and loop detection whereas the back-end contains state optimization as summarized in [1]. The front-end, referred to as Visual Odometry (VO), receives the measurements (images) and extracts relevant features from the image view. Next, VO performs

data association to match the measurements. The data association involves estimating the association between the features and previously observed landmarks in the map [2]. From the association, VO provides a preliminary pose estimate. The preliminary estimate is then used by the back-end to optimize the estimates of the pose and landmarks. The loop closure detects scene similarity by comparing the detected features with the previously visited locations in the map. Using information from locations recognized by the loop closure, the estimated pose is corrected. The optimization of map and pose involves the recovery of a model of the environment and a sequence of the system's locations from VO. The optimization process is generally approached as a maximum a posteriori estimation problem [1] and solved in two possible ways. Firstly, using an Extended Kalman Filter (EKF). Secondly, with graphical representations using nonlinear optimization methods. The EKF provides an initial state estimate when receiving the first sensor measurement and predicts the following state based on a process model. The update step of the EKF combines the predicted and measured states and provides an updated state estimate. Here, weights are distributed on prediction and measurement depending on their corresponding uncertainty. The graphical representation represents landmarks in the map and the agent's location as nodes in a graph. The agent's location at time step t is connected in the graph with the preceding location at $t-1$, from the information obtained by the VO reading. Connections are formed from each landmark reading from the corresponding location of the agent. Finally, mapping creates and updates the produced map of the environment and allows for the system to reset the location uncertainty with loop closure by revisiting known areas. Typically, maps produced by a SLAM system are grouped as sparse or dense. Sparse maps represent a limited number of landmarks in the environment. Sparse SLAM methods use distinctive features such as corners or texture-rich areas as landmarks and estimate the system's pose and the landmarks' positions from those features. Dense SLAM tracks all pixels from given input images to estimate the system's pose and its surrounding map. The depth of each pixel is estimated to structure the resulting map. The sparse mapping methods are typically more lightweight and computationally efficient. Dense methods on the other hand are more robust to sensor noise and occlusions while demanding more computational complexity than the sparse methods.

2.2 Semantic Information

Autonomous systems employing VSLAM rely on image sensors to perceive their environment. However, the traditional VSLAM systems do not make use of understanding their environment as people do. Humans rely on information such as traffic signs and objects' movement when going about their day. Providing autonomous systems with the ability to perceive and understand their environment generally involves multi-class object detection [3], [4], or image segmentation. Object detection provides valuable information for semantic understanding by estimating the locations of objects in an image and the category they belong to [5]. This involves estimating bounding boxes for each object with a label to predict the location of the objects as seen in Figure 2.2a. Image segmentation is a related task to object detection and scene understanding. The process involves simultaneous detection and boundary segmentation of objects. Image segmentation can be divided into two groups; semantic segmentation, Figure 2.2b and instance segmentation, Figure 2.2c.

(a) Object Detection (b) Semantic Segmentation (c) Instance Segmentation

Figure 2.2: The difference between object detection and image segmentation.

2.2.1 Semantic Segmentation

Unlike object detection, where objects are categorized and located, semantic segmentation segments images into regions at a pixel level. Each pixel of an image is provided an object class label (*e.g.,* car, human, dog), classifying pixels of the same label in the same category [6]. The literature presents a substantial amount of development of deep learning networks for semantic segmentation such as DeepLab [7]. Segmentation networks transform the task of recognizing the boundaries of objects into a classification problem. Knowing the class of every pixel in the image results in the boundary of objects. However, semantic segmentation does not distinguish different

objects with the same label. Objects of the same class are not separated as the segmentation depends solely on the category of each pixel. In Figure 2.2b the two dogs are of the same category, therefore segmented with the same region and could be considered the same object.

2.2.2 Instance Segmentation

Instance segmentation allows for a more unique understanding of each object as it has the characteristics of semantic segmentation together with multi-class object detection. It tackles the problem of classifying the image at pixel level while segmenting different object instances regardless of their label. Therefore, providing the ability to detect each object of interest in the image as seen in Figure 2.2c. Examples of deep learning networks for instance segmentation include Mask-RCNN [8], YOLACT [9] and Mask DINO [10]. YOLACT was chosen for this thesis work as it is presented as a real-time solution for the task of instance segmentation with faster inference time compared to other state-of-the-art solutions.

2.3 Optical Flow

Motion estimation is another fundamental task in computer vision with decades of research. The most general version of motion estimation is the independent estimation of velocity for each pixel, known as optical flow [11]. The optical flow estimation represents the motion of objects between consecutive image frames, caused by the relative movement between the object and the camera. This involves estimating the velocity for each pixel in an image and determining the location of the pixels in a consecutive image [12]. The problem is visualized in Figure 2.3, where the pixels of the first frame have moved a certain distance (dx, dy) over a fixed time interval between the two consecutive frames. The fixed period is represented as t and depends on the frame rate of the image input. Many optical flow methods make two assumptions: brightness is constant and time is continuous *i.e.,* movement is small. An object moving between consecutive frames will have negligible change in brightness (constant brightness) and change in time should not cause drastic changes in objects' locations (small movements). The two main groups to solve the problem of optical flow are sparse and dense methods, represented in Figure 2.4.

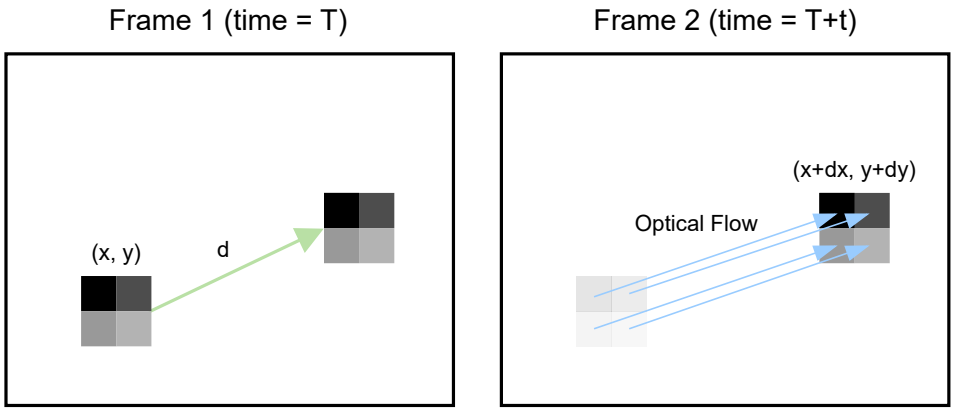


Figure 2.3: Simplified visualization of Optical Flow.

(a) Sparse Optical Flow (b) Dense Optical Flow

Figure 2.4: Sparse Optical Flow example on the left using the OpenCV implementation of Lucas-Kanade method[1]. Dense Optical Flow example on the right, using OpenCV implementation of Farneback method[2].

Lucas-Kanade [13], visualized in Figure 2.4a, is an example of a traditional sparse optical flow method. The sparse approach involves detecting regions of interest and computing the velocity vectors of their corresponding features. This could involve selecting a sparse feature set with feature detectors, such as Shi-Tomasi [14] or ORB [15]. The feature set is tracked between frames with optical flow, ensuring the tracking of points. Dense optical flow estimates the optical flow vector for every pixel of each frame. An example of a traditional dense optical flow method is Farneback [16] which is visualized in Figure 2.4b.

[1] Available at https://docs.opencv.org/3.0-beta/modules/video/doc/motion_analysis_and_object_tracking.html#calcopticalflowpyrlk

[2] Available at https://docs.opencv.org/3.0-beta/modules/video/doc/motion_analysis_and_object_tracking.html#calcopticalflowfarneback

Sparse optical flow is less computationally expensive as the image as a whole is not used for the computations. However, the sparse flow does not necessarily obtain the flow vectors of all moving objects since it depends on the detected regions of interest. Whereas, the dense flow computes the flow vectors of all pixels and therefore, all moving objects.

In 2015, A. Dosovitskiy *et al.,* [17] proposed FlowNet, a learning-based approach for dense optical flow computation. Since FlowNet, learning-based methods have surfaced in great quantity addressing challenges optical flow methods face *e.g.,* large displacement of objects [18], occlusion [19] and brightness [20]. RAFT [18] estimates a dense displacement field by mapping pixel coordinates between consecutive images and was chosen for the work in this project. RAFT was chosen as it achieved state-of-the-art results, is presented as cross-functional with regard to datasets and is a real-time solution. Both traditional and learning-based methods have their advantages and limitations. Learning-based methods have presented sped-up computations with the use of Graphics Processing Units (GPUs) and achieved a top place on various benchmarks. However, suitable training data is required with learning-based methods. The quality of this data can affect the performance and could also lead to overfitting of the data. Learning-based methods could also suffer from a lack of adaptability from their training nature.

2.4 Dynamic Objects

Dynamic objects in real-life environments take many forms with regard to movement. Some move relatively slowly (*e.g.,* walking people), others much faster (*e.g.,* cars), the latter often propelled by mechanical means. Most dynamic objects encountered in our environments furthermore move at an irregular pace, i.e. with varying speeds. But in addition to such dynamic objects, there are also objects that remain static for most of the time but are occasionally moved from one location to another by an external force, such as wind or the human hand. To further complicate the matter, there are also examples of objects that move while also remaining static *e.g.,* Ferris wheels and escalators.

Chapter 3

Literature Review

The performance of feature-based VSLAM systems can be impaired by features extracted from dynamic objects, increasing the need for research of VSLAM for dynamic environments. This chapter presents a literature review of the relevant work in the field of dynamic VSLAM. A substantial amount of solutions have been presented to solve the challenge of dynamic environments. Motivated by different goals, the existing solutions can be divided into three categories. The first involves treating data associated with dynamic objects as outliers and removing them from the feature extraction and estimation process, explained in Section 3.1. The second method involves disregarding the features for estimation while tracking the objects, explained in Section 3.2. The third is to reconstruct a map including both static and dynamic objects, Section 3.3.

3.1 Dynamic Feature Culling

The methods in this category aim to improve the accuracy and robustness of feature-based VSLAM by disregarding dynamic features. Dynamic feature points are detected so that only static feature points are fed into a feature-based VSLAM pipeline as seen in Figure 3.1.

Semantic information plays an important role in scene understanding in computer vision and has been exploited in VSLAM to improve performance in dynamic environments. Early examples of integrating semantic information into VSLAM pipelines are Mask-SLAM [21] and DS-SLAM [22]. Both methods segment RGB-D images into potentially dynamic and static regions. Mask-SLAM removes the classified dynamic regions from the feature extraction process. Directly removing points on movable objects without

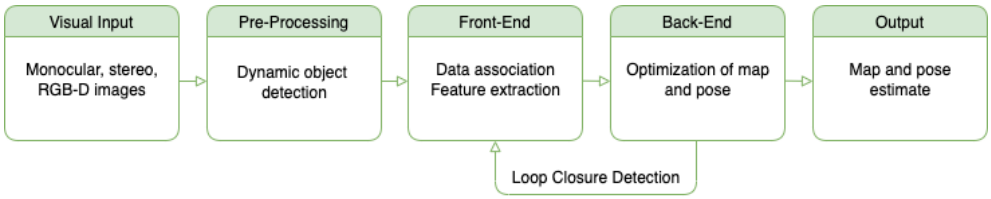


Figure 3.1: The structure of a Visual SLAM pipeline, disregarding dynamic features.

checking whether they are actually moving may result in lack of feature points to track. The dynamic regions could take up a large space in the image and the static background might not provide enough reliable points. This could result in the system breaking down. Addressing the problem of misclassification, DS-SLAM utilizes a moving consistency check in the form of geometric constraints on extracted features. Segmentation results are corrected before removing dynamic regions by comparing the results with a pre-defined geometric threshold. The paper presents robust performance with regard to walking people.

Other methods combining segmentation with geometrical constraints to determine dynamic features are presented in [23, 24, 25, 26]. The dynamic SLAM framework presented in [25] and DDL-SLAM [24] utilize a background inpainting strategy to restore features occluded by dynamic objects. The strategies involve tracking static information from previous frames to project into regions occluded by dynamic objects in the current frame. Moreover, [25] accounts for the motion of objects driven by people. These objects are not masked by the segmentation module due to their static nature *e.g.,* cups, laptops and bags. This involves utilizing Farneback [16] to compute dense optical flow between two consecutive frames and detect the motion of objects classified as static by the segmentation module. A threshold is computed from the pixels of the static object and the dynamic object triggering the motion. The threshold is defined as the average flow of all pixels within these objects. The pixels with flow exceeding the threshold are then considered dynamic and combined into the masked region. Other methods, [27, 28, 29, 30, 31, 32], correct segmentation results using optical flow and present different forms of thresholds to address the classification of dynamics to be excluded. The application of thresholds determines which dynamics should be excluded. The main difference between the methods is the choice of a segmentation network and the method for computing optical flow as

many solutions for both exist. Most of the above-mentioned solutions wait for semantic results in the tracking thread, resulting in increased computational time and unrealistic real-time usage. RDMO-SLAM [28] addresses this problem by exploiting optical flow for both motion estimation of objects and label prediction. Segmentation is performed on keyframes only and parallel to tracking. Utilizing label prediction offers more semantic information while ensuring real-time usage. Moreover, RDMO-SLAM highlights the challenge of determining which objects should be considered dynamic to achieve improved performance. OFM-SLAM [30] maps semantic labels of static parts of the scene, including the semantic information in the map reconstruction, and addressed the problem of camera motion compensation in the optical flow computations. For dynamic object classification OFM-SLAM utilizes a threshold in the form of an epipolar constraint.

The frameworks presented in [27], [31], and [32] utilize optical flow for motion information, feature matching and tracking. SDF-SLAM [27] uses semantic information with sparse optical flow for image feature tracking and fundamental matrix calculation to filter out truly dynamic features. The method combines the information with a depth filter to establish the dynamics of observations. YO-SLAM [31] uses segmentation to mask out people before the feature extraction. Next, optical flow with an epipolar constraint is utilized to perform feature matching. Combining optical flow tracking with feature point matching results in improved matching of point pairs. Similarly, the method presented in [32] tracks and matches feature points using optical flow after segmentation. The method utilizes a dynamic feature point removal to a VSLAM pipeline.

These solutions achieve robust and accurate estimation by discarding the dynamic information. However, the dynamic information has been shown to have potential benefits for SLAM with proper modeling so disregarding it has some disadvantages. In particular, understanding dynamic scenes as an addition to SLAM systems is crucial for other robotics tasks *e.g.,* planning, control, and obstacle avoidance. Moreover, determining how and when dynamics should be classified to be excluded seems to be a challenge the field faces. Thresholds for objects' motion have been introduced within the literature which seem to be defined from intuition. Objects are thereby classified as dynamic or static depending on whether or not their movement surpasses a defined threshold. It was observed that most solutions validate their results within one type of environment, containing a little variation of dynamics *e.g.,* traffic scenarios or walking people. Table 3.1 summarizes noteworthy methods from the review. The table includes the methods' sensor

input, map output, the hardware used, the year of publication and the datasets used for experiments. The table highlights the systems' dependencies on powerful hardware and how the majority of the solutions are validated through only one dataset.

Table 3.1: VSLAM methods for disregarding dynamic features.

Algorithm	Sensors	Map	Hardware	Year	Dataset
DDL-SLAM [24]	RGB-D	Sparse	i7 CPU 12GB RAM Unknown GPU	2020	TUM [33]
[25]	Monocular, Stereo, RGB-D	Sparse	i7-9700K CPU 48GB RAM RTX 2080 GPU	2021	KITTI TUM
RDMO-SLAM [28]	RGB-D	Sparse	RTX 2080 GPU	2021	TUM
OFM-SLAM [30]	RGB-D	Sparse	-	2021	TUM
YO-SLAM [31]	RGB-D	Sparse	i7 CPU 16GB RAM GTX 1060 GPU	2021	TUM
[32]	RGB-D	Sparse	i7 10750H CPU 16GB RAM RTX 2060 GPU	2021	TUM
[26]	RGB-D, IMU	Sparse	i7 CPU 16GB RAM RTX 2080 GPU	2022	TUM

3.2 Dynamic Object Tracking

Instead of directly removing dynamic information, other approaches consider the information valuable if properly used and tracked. An example pipeline of this approach is presented in Figure 3.2 Accurate motion estimation and tracking of objects can be relevant in many applications, such as collision avoidance, surveillance, and augmented reality. The existing solutions aim to improve the accuracy and robustness of VSLAM by reconstructing only the static part of a scene while simultaneously estimating and tracking the poses of the camera and multiple dynamic objects.

Examples of dynamic tracking are presented in [34, 35, 36]. Cluster-SLAM [34] is presented as a general SLAM backend for estimating the motion of objects in the environment while simultaneously tracking those objects.

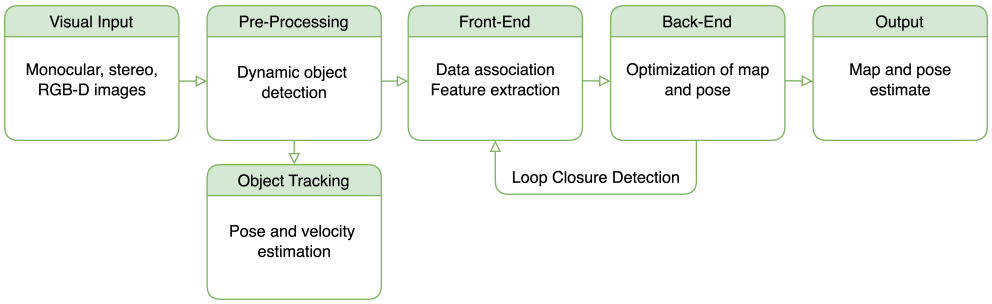


Figure 3.2: The structure of a Visual SLAM pipeline, tracking dynamic features.

The paper presents a solution aimed at autonomous driving with its object motion detection focusing on rigid bodies. Vincent et al. present a pipeline in [35] for improving both localization and mapping in dynamic environments. Instance segmentation is utilized with prior knowledge of dynamic objects from RGB-D images. The detected dynamic objects and depth images are used to estimate the positions and velocities of the objects while disregarding them in the feature extraction process. A static map is then reconstructed without dynamic objects.

Ballester *et al.,* [36] propose a front-end that can be added to an existing VSLAM pipeline to improve robustness and accuracy in highly dynamic environments. To determine which objects are actually moving, instance segmentation and geometric criteria are used with the estimated camera pose. The paper addresses the challenge of defining a threshold for objects' motion to exclude but proved good results for traffic scenarios. Motion results are used to update the segmented masks and feed the VSLAM system inputs. The updated masks are also tracked between frames making it unnecessary to segment every image frame. Truly dynamic objects are then tracked using the estimated camera motion and by minimizing the photometric reprojection error.

Zhang *et al.,* propose a visual odometry system [37], leveraging instance segmentation and optical flow estimation to improve the quality of tracked points and motion estimation accuracy. The method separates the scene into a static background and dynamic objects by leveraging instance segmentation and dense optical flow to ensure object tracking and object motion segmentation. The method contains image pre-processing, ego-motion estimation, and object motion tracking. The image pre-processing generates the image mask, the depth, and the dense flow for both the static and dynamic

parts of the scene. For ego-motion estimation, optical flow is employed to match features from a sparse feature set across frames. Finally, object motion tracking is performed in three steps. Firstly, classifying objects into dynamic and static classes followed by associating the dynamic objects across the two frames. An object is classified as dynamic if a proportion of its pixels is above a certain threshold, otherwise, the object is considered static. Lastly, the pipeline estimates individual object motion.

The presented solutions ignore dynamic objects for camera pose tracking and map estimation while tracking the dynamic objects in the environment. Advantages presented in the literature, compared to the methods in Chapter 3.1, are increased scene understanding, reduced rate of segmentation and less computational cost. All of which is favorable in real-time applications where decision-making (*e.g.,* path planning) is important. Regarding the challenge of determining how and when dynamics should be classified the solutions present intuitive thresholds for dynamic object classification. The thresholds seem to present strong results from the datasets (either containing moving cars or walking people) used for the evaluation of these solutions. However, there is a lack of validation for these solutions, which hinders their generalizability to different types of dynamics. Table 3.2 presents the noteworthy methods from the review.

Table 3.2: VSLAM methods for dynamic object tracking.

Algorithm	**Sensors**	**Map**	**Hardware**	**Year**	**Dataset**
ClusterSLAM [34]	Stereo	Sparse	Intel i7 CPU 32GB RAM GTX 1080 GPU	2019	CARLA, SUNCG, KITTI
[37]	Stereo RGB-D	Sparse	-	2020	KITTI, Virtual KITTI
DOTMask [35]	RGB-D	Dense Point Cloud	I5-8600k CPU GTX 1080 GPU	2020	TUM
DOT [36]	Stereo RGB-D	Sparse	Intel i5 CPU 8GB RAM	2021	KITTI

3.3 Dynamic Scene Reconstruction

Going beyond tracking dynamic objects and disregarding their features has been proposed in the literature with dynamic scene reconstruction, represented in Figure 3.3. Here the aim is to reconstruct the dynamic objects in the

mapping process.

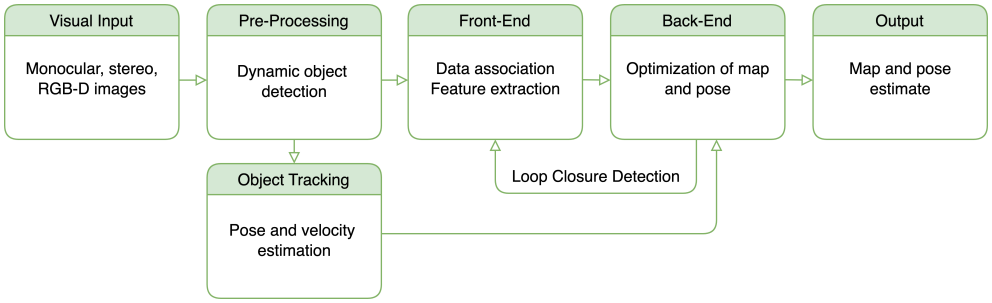


Figure 3.3: The structure of a Visual SLAM pipeline, containing dynamic reconstruction.

An early example of dynamic reconstruction is presented in [38] as a multibody visual SLAM system. The system utilizes an adaption of Structure from Motion (SfM) with segmentation and geometric constraints to achieve multibody tracking and reconstruction. Similarly, Dynamic SLAM in [39] utilizes segmentation and motion estimation of rigid objects by associating points from the dynamic objects at different time steps. A rigid body motion model shared by the points is then exploited to perform SLAM and dynamic object tracking. The presented solution is stated to perform well in autonomous driving scenarios. Reddy et al. [40] propose a stereo VSLAM pipeline capable of dynamic object tracking and reconstruction. Optical flow and depth are utilized to compute semantic motion segmentation. Semantic constraints are then employed to reconstruct static and dynamic objects independently. A dense mapping system was presented in [41] capable of reconstructing both a static background map and dynamic objects from a scene. The system consists of segmentation and tracking dynamic objects based on either motion or semantic cues. Dynamic objects are reconstructed separately using a surfel-based representation. Barsan et al. propose DynSLAM in [42] that utilizes the estimated camera poses and a fused depth map to reconstruct the static background and moving objects in the scene. Sparse scene flow is exploited to improve the reconstruction of dynamic objects by estimating their motion. Additionally, the authors propose a map pruning technique capable of improving the map accuracy, while substantially reducing memory consumption.

Recent solutions include VDO-SLAM [43] and DynaSLAM II [44]. VDO-SLAM is an expansion to the work done in [37] and [39]. The proposed approach utilizes semantic information with no prior knowledge about objects

to track and estimate the motion of dynamic objects. Prior to dynamic object tracking, the pixel flow of the detected objects is compared to a threshold for classification as dynamic or static. Factor graph optimization is applied to refine camera poses and object motions to reconstruct both static and dynamic objects in the environment. DynaSLAM II [44] integrates the multi-object tracking capability by utilizing instance semantic segmentation and ORB features to track dynamic objects. The structures of the static scene and the dynamic objects are reconstructed and optimized jointly from the trajectories of both the camera and the moving objects. The paper demonstrates that tracking dynamic objects can be beneficial for camera tracking.

The presented solutions combine the dynamic objects in the reconstruction. The identified advantages are related to spatial awareness and long-term analysis. Including dynamic objects in the reconstruction entails a complete representation of the scene. Moreover, the approach produces information that can be used to analyze dynamic objects' patterns or interactions with the scene. The solutions for dynamic reconstruction, as discussed within the previously presented approaches, primarily focus either on driving scenarios or environments that involve walking people. These cases seem to serve as the basis for addressing and validating the obtained results. Here, the lack of validation for variations in dynamics is important to note. Such as different types of movements or scenarios beyond driving and environments involving walking people. Table 3.3 summarizes noteworthy methods from the review. It should be noted that the hardware used seems to be less demanding than from Tables 3.1 and 3.2. This is due to a lack of hardware statements in the papers and excluding pre-processing of data from their experimental setup.

Table 3.3: VSLAM methods for dynamic reconstruction.

Algorithm	Sensors	Map	Hardware	Year	Dataset
Multibody SLAM [38]	Monocular	Sparse	Intel i7	2011	Camvid Versailles Road Moving Box
DynSLAM [42]	Stereo	Dense	GTX 1080 GPU	2018	KITTI
Dynamic SLAM [39]	Stereo RGB-D	Sparse	-	2020	KITTI, Virtual KITTI
VDO-SLAM [43]	Stereo RGB-D	Sparse	Intel i7, 16GB RAM	2020	KITTI Oxford-Multimotion
Dyna-SLAM II [44]	Stereo RGB-D	Sparse	-	2021	KITTI Oxford-Multimotion

3.4 Summary

This chapter summarized the recent literature and its recent advances. Considering the multiple solutions out there, a lack of availability was noted. Two systems, DOTMask [35] and VDO-SLAM [43] were chosen for further analysis due to their compatibility and promising performance. Both systems are relatively new and presented an improved performance in comparison to their dynamic SLAM counterparts. DOTMask belongs to the dynamic object tracking group and achieves improved performance compared to current solutions on dynamic data. VDO-SLAM belongs to the dynamic mapping group and also presents improved performance to other solutions. Both systems are open-source and provide good documentation in order to reproduce results.

Chapter 4

Systems & Datasets

This chapter provides an overview of the systems evaluated through experiments and the datasets used for evaluation.

4.1 Systems

As mentioned in Chapter 3.4, DOTMask [35] and VDO-SLAM [43] were chosen for further analysis and used for experiments. The following sections provide descriptions of the systems' functionalities.

4.1.1 DOTMask

DOTMask [35], Figure 4.1, is presented as a simple and fast front-end pipeline for a VSLAM system for improving both localization and mapping in dynamic environments. The pipeline consists of an instance segmentation network and a tracking module to track and estimate the velocity and position of dynamic objects. The resulting map is static, free of the tracked dynamic objects. As seen in Figure 4.1 RGB images are fed into an instance segmentation network, providing the Dynamic Object State (DOS). The DOS contains bounding boxes, class types and binary masks for each detected dynamic object. The masks from DOS are applied to the depth image, and the tracking module, where estimation of the velocity and position of the dynamic objects takes place. The tracking module of DOTMask uses the updated camera pose from VO for the velocity and position estimation. The predictions are fed into a dynamic object classifier which provides information on whether an object is actually moving or not. Truly dynamic objects are then removed from the original depth image and fed into the VO of the VSLAM pipeline, where

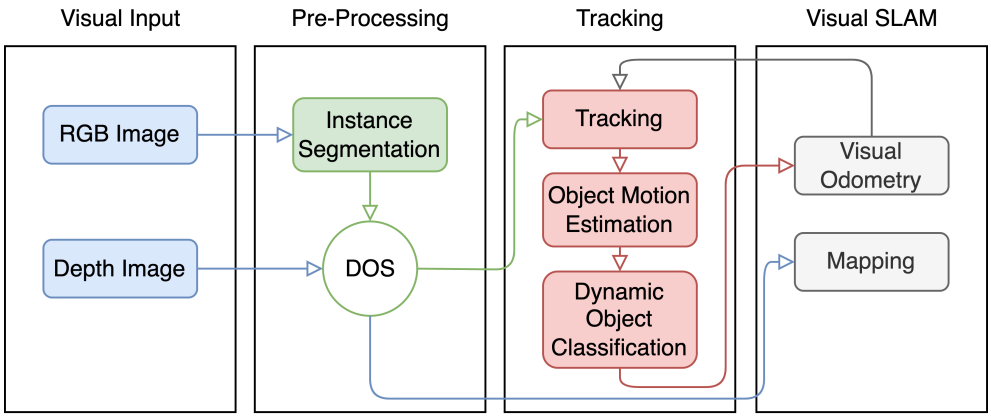


Figure 4.1: The structure of DOTMask pipeline.

features of the masked dynamic objects are ignored. The masks provided by the segmentation are fed to the mapping module of the VSLAM pipeline to reconstruct a static map.

The tracking module consists of an Extended Kalman Filter (EKF). The DOS from the segmentation and the original depth image are used by the EKF to estimate the objects' velocities and positions. Tracking is initialized for every newly detected object where priori knowledge about the object's class defines some of the EKF's parameters. The output of the tracking module is the object's state which contains its position and velocity. A priori estimate of the state is predicted from the previous state and the dynamic model of an instance being tracked depends on its class. When an object is not observed for a couple of frames its tracking is disregarded. The object's state is then used for dynamic classification, if an object is above a pre-defined threshold the object is considered dynamic and excluded in the camera pose estimation in VO.

4.1.2 VDO SLAM

VDO-SLAM [43], unlike DOTMask [35], is a whole VSLAM system, not only a front-end pipeline. The VDO-SLAM pipeline consists of four main modules; pre-processing, tracking, VO and mapping as seen in Figure 4.2. The pre-processing module detects and separates objects from the background and enables object tracking within the system. The detection is comprised of an instance segmentation network and an optical flow network. The segmentation process identifies all objects in the field of view with their corresponding

semantic information *e.g.,* people, animals and cars. Moreover, boundary masks are produced for each object which provides the boundary for tracking feature points on the objects. The optical flow network has the purpose of maximizing the number of tracked points on the detected moving objects from the segmentation process. The justification for a dense optical flow method is due to objects only occupying small portions of the image, and sparse optical flow would not ensure long-term feature tracking given such an assumption. The dense approach offers an increased number of features to be tracked by sampling all pixels within the object segmentation mask. This enables multiple object tracking with unique identification on every pixel within each mask. The approach also enables the recovery of masks when the segmentation fails.

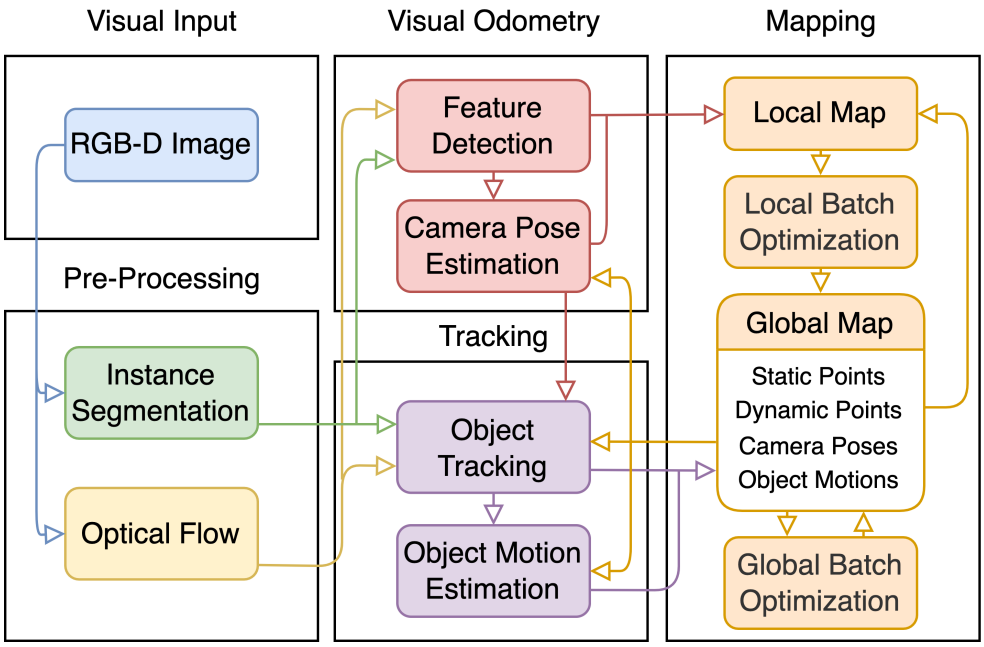


Figure 4.2: The structure of VDO-SLAM pipeline.

The VO module contains feature detection and camera pose estimation. The camera pose is computed from feature points detected on the static background. For every frame, outliers are excluded from the map while inliers, fitting the camera motion estimate, are saved to the map and used for correspondence between frames. If inlier points fall below a certain level, new detections are added. The static points are projected onto the image plane and the camera pose is computed from a cost function minimizing the re-projection

error. The tracking module tracks both the objects and their movement. The process involves classifying objects as static or dynamic and matching the dynamic objects across consecutive frames. For classification, scene flow is computed from all sampled feature points between consecutive frames. An object is considered static if the scene flow of 70% of its sampled points is below a pre-defined threshold and dynamic otherwise. The optical flow input is then utilized to associate and track points between frames and dynamic object motion estimation follows a similar method as used for the camera pose estimation with a cost function minimizing the re-projection error between an object point and its corresponding point on the image plane.

The mapping module constructs a local map of the last 20 processed frames and is optimized with regard to the camera pose and static structure. The module also maintains a global map that is updated for every processed frame and receives camera pose estimates from the local map and object motion from the tracking module.

4.2 Datasets

Various datasets exist for VSLAM evaluation. As the systems at hand are designed for an RGB-D camera the scope was focused on RGB-D datasets. The following chapters provide some details about the datasets used for experiments.

4.2.1 TUM

The TUM dataset [33] contains RGB-D sequences and ground truth data from several categories. The dataset is popular for the evaluation of object reconstruction, SLAM, and VO methods. The *Walking* sequences were chosen as the work focuses on dynamic environments. These sequences expose systems to dynamic objects in the form of moving people in an office environment. Moreover the sequences contain different types of camera movement and covering of the camera's Field of View (FOV).

The *Walking* sequences include four types of camera motion, indicated in their corresponding name. Firstly it includes the *Walking-Static* sequence, seen in Figure 4.3. Here the camera is kept in place throughout the recording. Secondly, it includes keeping the camera at a fixed orientation while moving the camera along the translational axes x, y and z, very slowly for the *Walking-XYZ* sequence. These sequences expose systems to slightly and quickly moving dynamic objects in large parts of the field of view. Thirdly, the

Figure 4.3: The Walking-*Static* sequence in TUM RGB-D.

camera is moved on a half sphere approximately 1 meter in diameter for the *Walking-Halfsphere* sequences. Lastly, The camera was kept at a fixed position and rotated along the rotational axes roll, pitch and yaw for *Walking-RPY* sequences. Rotating while keeping a fixed position can be good to evaluate the performance of orientation estimation of the Visual SLAM system at hand.

4.2.2 OpenLoris

OpenLORIS [45] contains RGB-D indoor sequences and ground truth data from several environments. The dataset contains a total of 22 sequences from different environments. Compared to other VSLAM evaluation datasets the environment is not tightly controlled but rather features realistic scenarios with a wide variety of challenges; occlusions, dynamic motion, textureless environment, and changes in lighting. However, as the focus is on dealing with dynamics, the sequences *Cafe*, *Corridor*, and *Market* were chosen for experiments.

Figure 4.4: *Corridor 5* sequence in OpenLoris.

Corridor, Figure 4.4, is one of the dynamic sequences found in the dataset. The camera follows a person walking a corridor while being exposed to light and can be used to evaluate the performance of SLAM algorithms to slow movement and light exposure. Challenges within the sequence include lack of features, contrast due to light exposure from windows, and completely dark rooms without light.

Figure 4.5: *Cafe 2* sequence in OpenLoris.

The *Cafe* sequences, Figure 4.5, are dynamic with respect to both camera handling and environment as the camera is manually moved around a cafe filled with people. The environment also contains difficult lighting conditions with light exposure from windows. The third group of dynamic sequences within the OpenLoris dataset are the *Market* sequences. The *Market* environment, seen in Figure 4.6 is a busy supermarket filled with people and products stuffed on shelves. Challenges include people covering the field of view, moving and stopping repeatedly. Other challenges include light reflection on the floor and some exposure from windows.

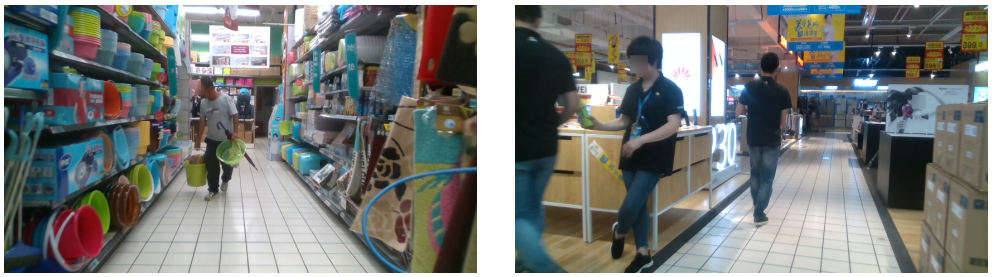


Figure 4.6: *Market 3* sequence in OpenLoris.

4.2.3 Summary

The data sequences used for experiments are summarized in Table 4.1. The TUM Walking sequences were captured at 30 Hz and 640x480 resolution with a Microsoft Kinect sensor. The OpenLoris sequences were captured at 30 Hz and 848x480 resolution with a RealSense D435i camera along its corresponding ground truth trajectory. Both datasets include corresponding ground truth trajectories with the sequences for evaluation purposes.

Table 4.1: Data sequences used for experiments.

Dataset	Sequence	Description	Challenges
TUM *Walking*	Halfsphere	Camera orbits along a halfsphere	Moving people, parallel to camera motion
	RPY	Camera moves along RPY axes	Moving people, erratic camera motion
	Static	Static camera	Moving people, FOV blocked
	XYZ	Camera moves along XYZ axes	Moving people, parallel to camera motion
OpenLoris	Cafe	Camera moves around a cafe	Moving & static people, difficult lighting, FOV blocked
	Corridor	Camera moves around hallways	Moving people, difficult lighting, lack of features
	Market	Camera moves around a market	Moving & static people, crowded, FOV blocked

Chapter 5

Evaluation Design

Developing the evaluation approach for the field of Visual SLAM systems dealing with dynamics in their environment involved exploring the existing evaluation methods for Visual SLAM systems and their variations. The main objective of the development of an evaluation method was to define a set of properties that could provide a wide range of comparisons. This chapter describes the metrics used to evaluate the overall performance and usage. Visual SLAM systems vary in design depending on the application. Therefore, looking at only one metric/benchmark and achieving a certain threshold would not be considered a valid approach due to the lack of modularity.

5.1 Qualitative Observations

During the experiments, qualitative observations will be made. This will mainly consist of observing the tracked features during runs to reflect on the quantitative results. During the experiments, qualitative observations will be made to provide additional information and insights. The observations will primarily involve closely monitoring the extracted features in the image frames throughout the runs. By doing so, we can further enhance our understanding of the quantitative results and conclude explanations for their causes. Qualitative observations will provide a more comprehensive perspective on the performance and behavior of the systems. Combining qualitative and quantitative evaluation will contribute to a holistic evaluation of the experiment outcomes.

5.2 Accuracy of Trajectory Estimation

The output of a traditional VSLAM system is the estimated camera trajectory and the reconstructed map. To evaluate the resulting estimates, accurate ground truth values must be available. Sturm *et al.,* [33] advocated the simple evaluation method of considering the estimated camera trajectory instead of maps, as accurate ground truth maps are difficult to obtain. The proposed evaluation compares an estimated camera trajectory from an input sequence of RGB-D images to a ground truth trajectory.

5.2.1 Relative Pose Error

The Relative Pose Error (RPE) is the local error between the estimated and ground truth trajectories over a fixed period, Δ. The length of the period can be specified in terms of the number of frames/timestamps, seconds or meters. As the local error can represent the drift in trajectory, the period is commonly defined as the interval between two consecutive frames. Consequently, the resulting RPE represents the drift per frame.

The RPE computations involve dividing both the estimated trajectory and the ground truth trajectory into sub-trajectories over the fixed period, Δ, using sequences of poses $P_1, P_2, ..., P_N \in SE(3)$ and $G_1, G_2, ..., G_N \in SE(3)$ respectively. These sub-trajectories are then aligned, considering their different reference frames, and the RPE matrix over the time period of $[t : t + \Delta]$ is then defined as:

$$RPE_{t,t+\Delta} = (G_t^{-1} G_{t+\Delta})^{-1} T P_t^{-1} P_{t+\Delta} \in SE(3) \tag{5.1}$$

Where t runs over all timestamps $[1 : N - \Delta]$, Δ represents the interval between frames/timestamps used to segment the trajectories, and T denotes the transformation for alignment of reference frames for the two trajectories. The translational part, $(trans(RPE_{t,t+\Delta}) = [dx, dy, dz]^{-1})$, is then extracted from the $RPE_{t,t+\Delta}$ matrix in equation 5.1 and its magnitude denoted by:

$$||trans(RPE_{t,t+\Delta})|| = \sqrt{dx^2 + dy^2 + dz^2} \tag{5.2}$$

The rotational part of the RPE metric could also be evaluated with similar computations. However, as Sturm *et al.,* advocate, the translational part will be considered sufficient since rotational errors typically lead to inaccurate translational estimation of the trajectory. Finally, the Root Mean Square Error (RMSE) is computed from the translational components of the RPEs from all

the sub-trajectories using the following equation.

$$RMSE(RPE_{1:N}, \Delta) = \sqrt{\frac{1}{n}\sum_{t=1}^{n} ||trans(RPE_{t,t+\Delta})||^2} \tag{5.3}$$

Where N denotes the absolute number of timestamps (corresponding to each pose), n indicates the number of sub-trajectories and RPEs. If considering the PRE between consecutive frames, Δ is defined as 1 and $n = N - \Delta = N - 1$.

5.2.2 Absolute Trajectory Error

The global error of trajectory is measured using the Absolute Trajectory Error (ATE) and provides insights into the global accuracy of a VSLAM system. The ATE is an important metric to evaluate VSLAM systems, by comparing the absolute distance between their estimated trajectories and a reference ground truth trajectory.

The computations for ATE resemble those of the previous chapter's RPE. The key difference lies in the absence of trajectory division for ATE, *i.e.,* the entire trajectory is considered as a whole for alignment instead of being divided into sub-trajectories. Firstly, the ATE matrix for a pose pair (between the estimated P and ground truth trajectories G) at timestamp t is defined as:

$$ATE_t = G_t^{-1} T P_t \in SE(3) \tag{5.4}$$

Where t runs over all timestamps $[1 : N]$ and T denotes the transformation between all the poses of the two trajectories, $G_{1:N}$ and $P_{1:N}$, for alignment of reference frames. The translational part, $(trans(ATE_t) = [dx, dy, dz]^{-1})$, is then extracted from the ATE_t matrix in equation 5.4 and its magnitude denoted by:

$$||trans(ATE_t)|| = \sqrt{dx^2 + dy^2 + dz^2} \tag{5.5}$$

Lastly, the RMSE is computed of the translational components, 5.5, using the following equation.

$$RMSE(ATE_{1:N}) = \sqrt{\frac{1}{N}\sum_{t=1}^{N} ||trans(ATE_t)||^2} \tag{5.6}$$

Where N represents the total number of timestamps (corresponding to

each pose) in the estimated trajectory.

5.3 Robustness

As stated in [1], SLAM entered the *Robust-Perception Age* some years ago. It is therefore important that the robustness of existing solutions is confirmed through experiments. Lifelong VSLAM is the VSLAM problem applied for long-term deployment [45]. This entails maintaining a persistent map that can be reused to ensure accurate localization in the environment. Similar to [46], we define the robustness metric as the VSLAM system's ability to avoid failing by detecting and recovering from environmental disturbances in the form of dynamic objects. Achieving long-term performance requires the system to be robust when disturbances are encountered. Operation in previously untested environments is key, and therefore important to vary the datasets used for experiments to prove the system's robustness.

5.4 Practicality

Evaluating practicality involves taking into consideration the portability of the system, its setup, and its usage. This involves assessing the dependency on the original hardware, as presented in the papers, and the ability to transfer the system to other specifications, *e.g.,* lower computing power. Assessing the setup considers the modularity to change, *e.g.,* the dependency on packages. The usage is evaluated by assessing the compatibility and the running process of the system. It should be important to note that conclusions made about the practicality of the systems can not be considered completely objective. However, the aim is to consider practical aspects with regard to the real world.

Chapter 6

Results & Discussion

This chapter presents the experimental results of DOTMask [35] and VDO-SLAM [43]. The experiments consisted of executing the VSLAM systems on the chosen data. Details for implementation and execution are found in Appendix A. This chapter starts off by presenting noteworthy qualitative observations made throughout the experiments. The results presented are from individual experiments without averaging. Following the observations, quantitative results are presented. For evaluation purposes, 10 runs were performed on each data sequence and the average error was analyzed. Lastly, a general comparison is presented between the two systems with a reflection on robustness and practicality.

6.1 Qualitative Results

The qualitative analysis involved observing the systems' behavior during experiments on the chosen datasets. The following sections present the noteworthy observations made throughout the runs and the chapter concludes with a summary of the qualitative results.

TUM, *Freiburg 3 - Walking*

As mentioned in Chapter 4.2.1 the *Walking* sequences present an office environment with differing types of camera movement; *Halfsphere*, *RPY*, *Static* and *XYZ*. Noteworthy differences were observed in the extracted features between sequences. Since the same method was used for mask generation, the difference in dynamic object detection between the systems lies in the method of determining if objects are dynamic or static. Both systems properly

(a) Example from DOTMask (b) Example from VDO-SLAM

Figure 6.1: Extracted features on *Static* sequence.

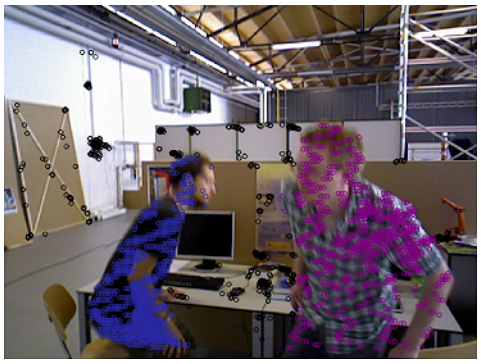
detected the dynamic objects in the static sequence. However, they also appeared to exclude people when they were not moving. Figure 6.1a shows how DOTMask regarded the person sitting on the right as dynamic, with no features visible. VDO-SLAM also excluded the people when static as seen in Figure 6.1b. DOTMask seemed to manage to detect the person to the left as static, whereas VDO-SLAM failed. Moreover, DOTMask was observed to extract more features throughout the sequence.

Both systems were observed to perform worse feature detection in the *Halfsphere*, *RPY* and *XYZ* sequences. It is worth noting that neither system handle ”difficult” camera movement which might have affected the feature extraction during the runs. The camera movement was quite erratic during these sequences and many frames precluded any extracted features. The effect of the camera movement on VDO-SLAM performance was most apparent in the *Halfsphere* sequence. As the camera turned parallel to the moving people, features appeared on the people, indicating the dynamic object detection was not working properly. An example of this can be seen in Figure 6.2b. For comparison, 6.2a shows the same for DOTMask and here neither person contains extracted features.

What could have affected VDO-SLAM in this instance is the system’s use of optical flow. The optical flow might not have been detected correctly as the people move parallel to the camera during some frames. When reviewing the literature it was quite apparent that there is a lack of focus on camera motion. In fact, all focus is on the dynamic objects the cameras observe and consideration of camera movement is little to none.

Regarding the *RPY* and *XYZ* sequences, more similarity was noted. Here the camera moved faster compared to the other two sequences. Both sequences

(a) Example from DOTMask

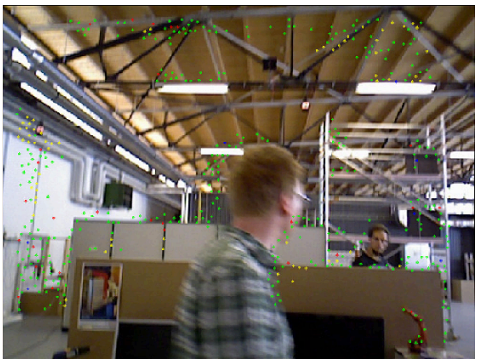


(b) Example from VDO-SLAM

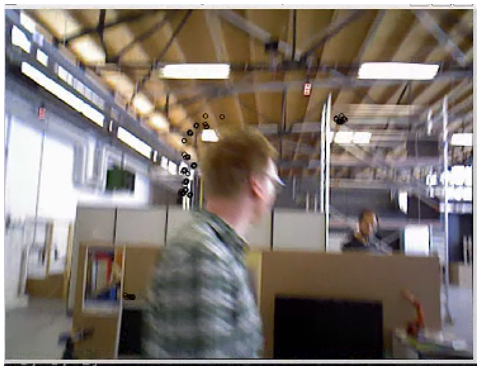
Figure 6.2: Extracted features on *Halfsphere* sequence.

included frames where a large part of the camera's view was blocked by one of the persons present. However, DOTMask was observed to extract more features throughout both sequences compared to VDO-SLAM. Figures 6.3 and 6.4 show an example of fewer features being extracted in VDO-SLAM.

Overall VDO-SLAM feature extraction performance appeared to be worse than DOTMask for the *TUM Walking* sequences.

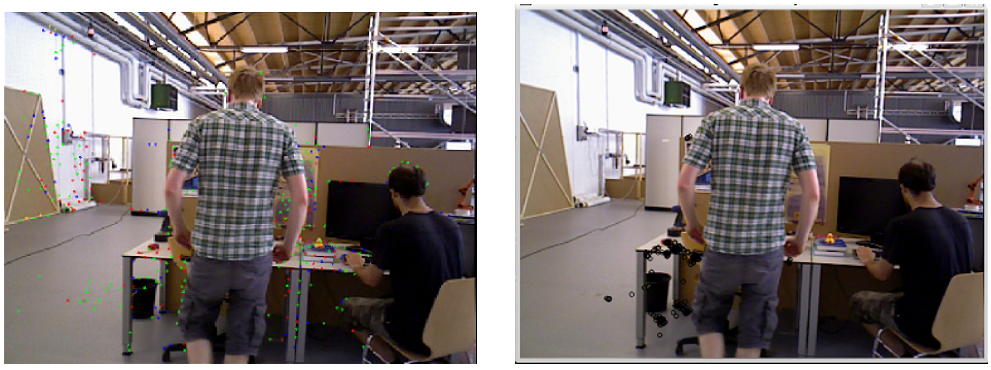


(a) Example from DOTMask



(b) Example from VDO-SLAM

Figure 6.3: Extracted features on *RPY* sequence.

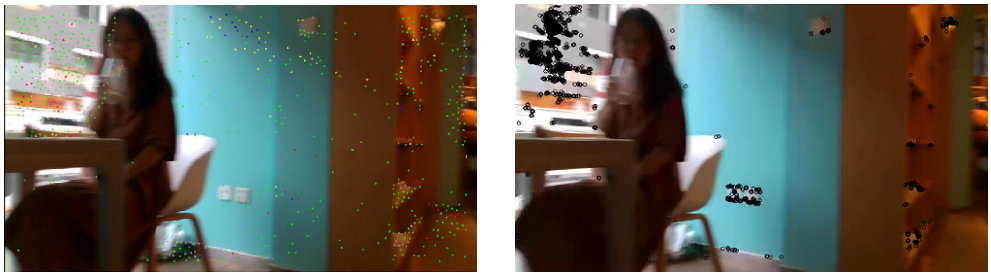


(a) Example from DOTMask (b) Example from VDO-SLAM

Figure 6.4: Extracted features on *XYZ* sequence.

OpenLoris, *Cafe*

Cafe 1 can be considered a static environment since throughout the sequence people sit at tables without moving. The challenge of this sequence for a dynamic SLAM system is to correct the dynamic object classification with motion detection. DOTMask was observed to extract more features than VDO-SLAM. Both from sitting people and the cafe's surroundings. However, DOTMask did fail in some cases as did VDO-SLAM in extracting features from still people as seen in Figure 6.5.



(a) Example from DOTMask (b) Example from VDO-SLAM

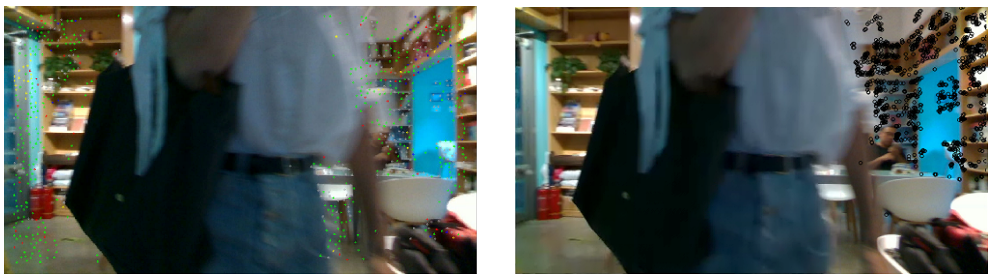
Figure 6.5: Extracted features on *Cafe 1* sequence.

Cafe 2 contains more movement with people walking around and blocking the field of view. Similar observations were made during runs of *Cafe 2* as were made for *Cafe 1*. DOTMask extracted more features and both systems failed to properly detect people as static and use their area for feature extraction. Instead, features were not extracted from still people. Both systems did exclude feature extraction on moving people within the sequence. A couple of frames

are largely covered by a walking person as seen in Figure 6.6. The authors of VDO-SLAM claim objects do not cover large parts of the frame. Which can in fact happen and result in few extracted features over these frames.

As the nature of these systems is to detect and exclude dynamic objects it is no surprise that there is a difference in extracted features. However, the only potentially dynamic objects within these sequences are people. When dynamic objects are removed from the pipeline of a SLAM system it results in their area being excluded from pose estimation. These objects might include rich information and if the system does not properly detect a potentially dynamic object to be static, it results in loss of information to the pipeline. This could be problematic if the system does not manage to extract enough features from the environment or if the object covers a large part of the field of view. Dynamic SLAM systems that do not check for the true movement of a potentially dynamic object could perform badly in a static environment with many potentially dynamic objects that are in a static state. Prior to confirming an object's motion, an object, rich with informative features, might be excluded from the feature extraction process.

Throughout both sequences, there are big windows causing strong light exposure. The performance with regard to difficult lighting seems to affect the feature extraction for both systems. However, VDO-SLAM was observed to suffer more from this circumstance. With regard to loop closure, problems might arise in the *Cafe 1* sequence due to DOTMask not allowing idle objects into the loop closure detection. Both sequences should have allowed for loop closure detection for both systems. DOTMask did succeed in detecting loops within the sequences but VDO-SLAM failed.

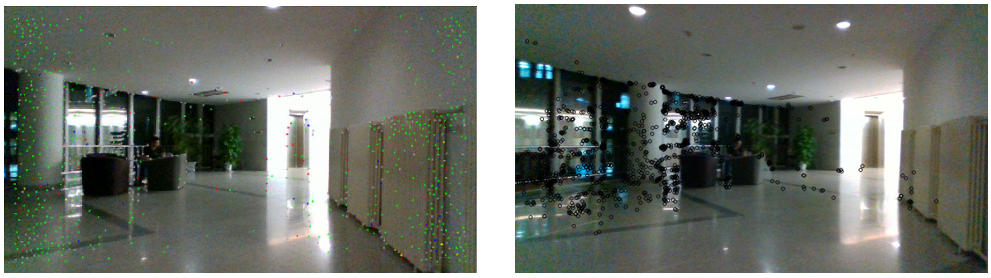


(a) Example from DOTMask (b) Example from VDO-SLAM

Figure 6.6: Extracted features on *Cafe 2* sequence.

OpenLoris, *Corridor*

The *Corridor* sequences consist of an environment lacking features, light exposure and reflection. Both DOTMask and VDO-SLAM were observed to suffer from these challenges as presented in Figure 6.7, from *Corridor 4*. Both systems struggled to extract features from the floor due to the lack of features and the reflection caused by light exposure at the end of the hallway. For a feature-based SLAM to function properly within that type of environment it is important to grab the available features. Additionally, dynamic SLAM could encounter a problem if not properly detecting dynamic objects as static and excluding their features for computations.



(a) Example from DOTMask (b) Example from VDO-SLAM

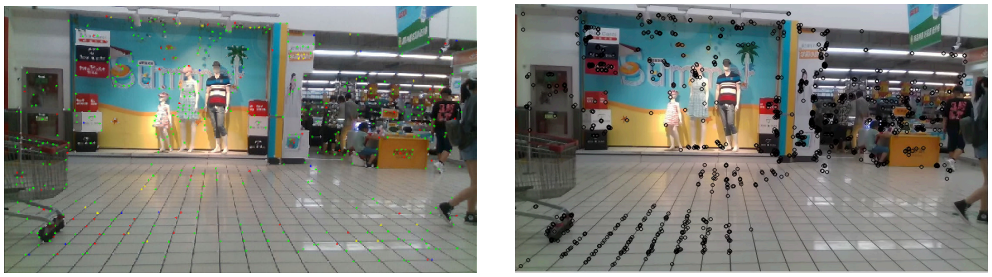
Figure 6.7: Extracted features on *Corridor 4* sequence.

In *Corridor 3*, for part of the sequence the hallway is completely dark. Both systems did not manage to track its pose during the blackout but continued running until a lit-up room was entered. The sequence contained a loop closure which DOTMask detected whereas VDO-SLAM failed. This could be due to VDO-SLAM extracting fewer features throughout the sequence compared to DOTMask. Depending on the ability In general, the application results within dark environments depend on the ability of a system to get back on track after encountering a period where no reliable estimation values are computed.

OpenLoris, *Market*

The *Market* sequences showed promising results. As a market environment is rich in features there were not many frames where both systems extracted few features. However, as was observed in the TUM and *Cafe* sequences, both systems excluded static objects from feature extraction. Although there were people moving within the sequences, they did sometimes stop for a few

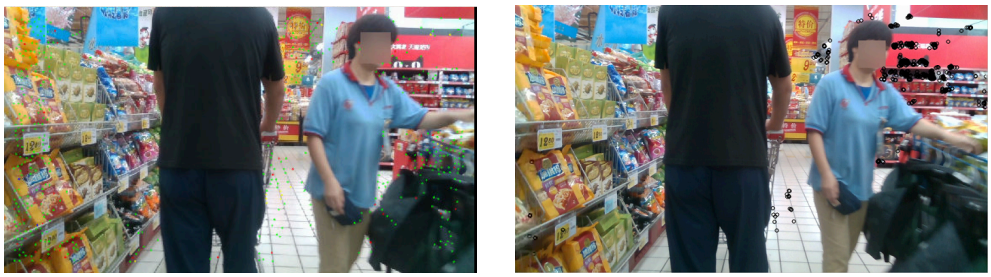
frames. The systems struggled to detect the absence of motion when people stopped, resulting in the exclusion of their features from the feature extraction. Mannequins, featured throughout the *Market* sequences, were also wrongfully classified as dynamic in both systems. DOTMask was observed to extract features from mannequins in some frames as seen in Figure 6.8a. VDO-SLAM, on the other hand, was observed to exclude mannequins in most cases, as seen in Figure 6.8b. The probable culprit is the instance segmentation network which classifies mannequins as people. However, both DOTMask and VDO-SLAM should have detected the absence of motion in these cases according to claims made in their paper. After classifying the static people and mannequins as potentially dynamic, they should have been corrected as static instead of being considered dynamic. Again, as with the *Cafe* and *Corridor* sequences, the floor proved difficult due to reflection, resulting in few features being extracted from the floor. Figure 6.8 shows the problems encountered by the systems in the *Market* sequence due to the reflection of the floor. However, this problem was more noticeable for DOTMask.



(a) Example from DOTMask (b) Example from VDO-SLAM

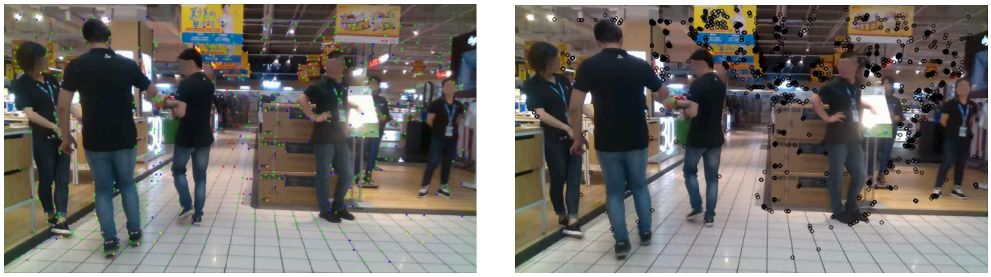
Figure 6.8: Extracted features on *Market 1* sequence.

VDO-SLAM seemed to struggle with feature extraction when facing fully stuffed shelves, seen in Figure 6.9. DOTMask, however, demonstrated the ability to extract features from those difficult areas. Again, some artifacts of the segmentation masks were present throughout the market sequences. However, neither system performed feature extraction on the moving people in the environment.



(a) Example from DOTMask (b) Example from VDO-SLAM

Figure 6.9: Extracted features on *Market 2* sequence.



(a) Example from DOTMask (b) Example from VDO-SLAM

Figure 6.10: Extracted features on *Market 3* sequence.

6.1.1 Summary of qualitative evaluation

Both systems successfully disregarded dynamic objects most of the time but the quality of the dynamic object detection was not perfect. The culprit could be remnants of erroneous computations of pixels from segmentation masks and optical flow. The limitation of the systems' behavior was the handling of the camera movement which resulted in errors of dynamic object detection. VDO-SLAM was observed to misclassify objects as dynamic when in fact they were static. DOTMask, on the other hand, was observed to extract features of static people. The same method was used for mask generation in both systems which indicates the culprit of VDO-SLAM misclassifications to relate to its method of confirming motion of objects. DOTMask initializes the estimated velocity for a new object with the use of masks, depth images and priori information in the form of the object's class *e.g.,* person or dog. Conversely, VDO-SLAM uses scene flow estimation to detect the motion of objects. Camera movement in the experimented sequences may have proven problematic for scene flow computations. This would especially apply to the TUM Walking sequences when encountering difficult camera movement.

Both systems utilize manual thresholds for computations. DOTMask employs a threshold of movement set at 0.01 m/sec for people and 0.1 m/sec for other dynamic objects. The estimated object velocities from the object motion estimation module of DOTMask are compared to these pre-defined movement thresholds. If an object's velocity is greater than the threshold it is classified as dynamic and static if the velocity is below the threshold. DOTMask was observed to extract features from dynamic objects at frames where motion started. This behavior could be the result of the threshold as the motion was not detected to be over the threshold at the start. VDO-SLAM employs a threshold for scene flow magnitude, set at 0.12 to account for noise and error. Scene flow is computed for all pixels within a masked object and any object with 30% of pixels with scene flow greater than the threshold is regarded as dynamic. The authors justify the choice of a low threshold to avoid incorrectly identifying dynamic objects as static and inadvertently extracting features from them, which is not the system's intended purpose. The systems' thresholds could be fine-tuned for specific datasets to improve performance.

6.2 Trajectory Results

The following section presents the errors from the generated trajectories from both systems on all the data sequences used for experiments. As mentioned in Chapter 5, the quality of the trajectories can be measured via RPE and ATE. Firstly, we look at the produced RPE which represents the local accuracy of the systems' trajectories. Secondly, we look at the produced ATE which represents the global accuracy of the trajectories. The section concludes by summarizing the results. Both metrics were computed using evo[1], an open-source package that provides tools for trajectory evaluation for SLAM systems.

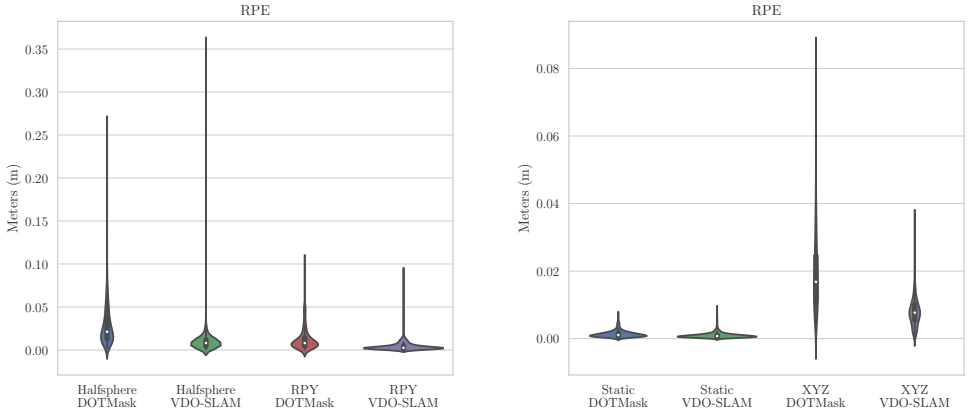
6.2.1 RPE results

As mentioned in Chapter 5.2.1 RPE presents a periodical evaluation where the estimated and ground truth trajectories are divided into sub-trajectories. The length of these sub-trajectories for the evaluation was determined by two consecutive frames, corresponding to the error between the two frames. The average RPE error over 10 runs on each sequence for both systems was computed. The results are presented in Figures 6.11, 6.12 and 6.13. The figures visualize the distribution of the RPE errors. The median is a white

[1] Available at https://github.com/MichaelGrupp/evo

dot placed on the black bar representing the interquartile range. The outliers appear at the ends of the ”violin”.

TUM



(a) *Halfsphere* and *RPY* sequences

(b) *Static* and *XYZ* sequences

Figure 6.11: The RPE results of the TUM sequences represented on a violin plot.

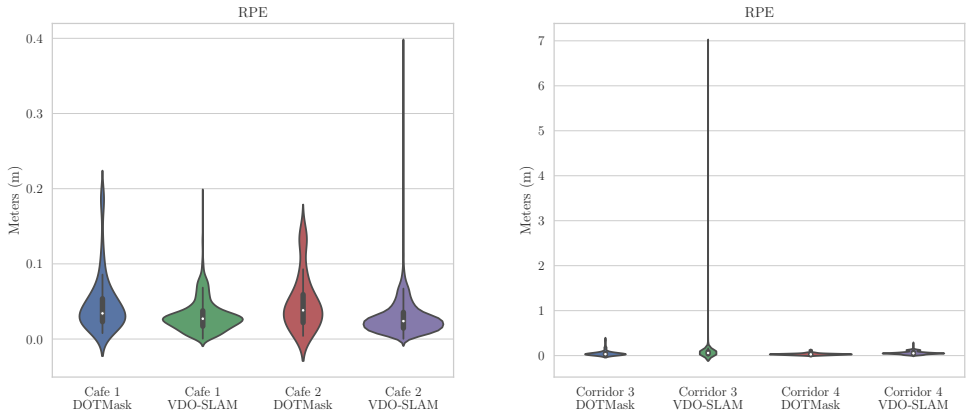
Figure 6.11 presents the RPE results from both systems on the TUM sequences. The performance was overall quite similar between the systems with VDO-SLAM proving to have slightly better performance. As mentioned in Chapter 6.1, both systems were observed to be affected by the camera movement in terms of dynamic object detection. VDO-SLAM was observed to misclassify static people within the sequences as dynamic. This did not seem to have affected the local pose estimation between frames severely. However, DOTMask was observed to extract features when the people went from static to dynamic which could explain the RPE results and inferior performance compared to VDO-SLAM. Moreover, frames within the sequence allowed for few extracted features due to large occlusions.

The effects of the camera movement and large occlusions are most prominent in the *Halfsphere*, Figure 6.11a, and *XYZ*, Figure 6.11b, sequences. As seen from the figures, the *Halfsphere* sequence presents extreme outliers for both systems. Observations from individual runs on this sequence showed that outliers were always present for this sequence. Moreover, the *XYZ* sequence includes large occlusions of a moving person which the systems detected and excluded for feature extraction. Excluding large areas for feature extraction

seemed to negatively affect the estimation computation. The difference between the systems' performance on the *RPY* and *Static* sequences was nominal as seen in Figures 6.11a and 6.11b. Both systems had the best performance on the *Static* sequence with regard to RPE. This indicates a lack of ability to account for the camera movement while detecting dynamic objects within the environment, whilst highlighting the performance of handling dynamic objects.

Cafe

The RPE results for the *Cafe* sequences are presented in Figure 6.12a. VDO-SLAM provides slightly better relative accuracy than DOTMask in both sequences. No notable difference can be observed between the sequences from each system. These results are interesting since observations during the runs highlighted that VDO-SLAM excluded features on static people throughout the sequence. Whereas DOTMask managed to extract features from static people.



(a) Cafe sequences

(b) Corridor sequences

Figure 6.12: The RPE results of the *Cafe* and *Corridor* sequences represented on a violin plot.

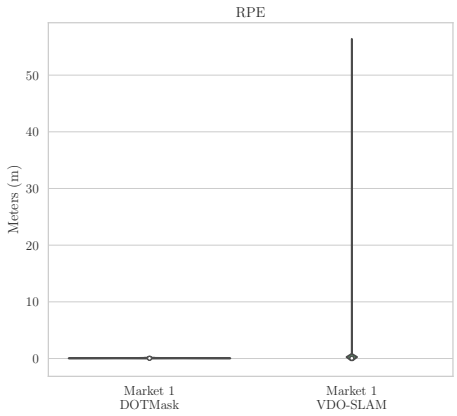
Corridor

Both systems performed similarly in the *Corridor* sequences. The results from sequences 3 and 4 are presented in Figure 6.12b. As mentioned in Chapter 6.1, these two sequences contained difficult lighting in the form of darkness. DOTMask had slightly better results but the difference was nominal.

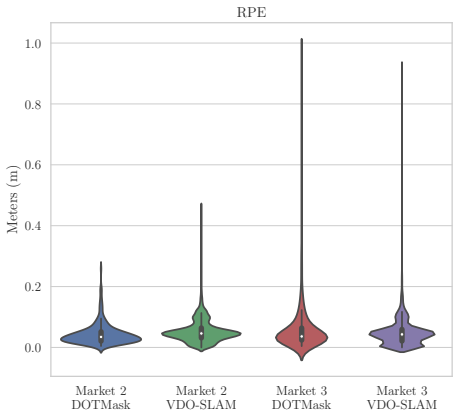
VDO-SLAM presented more extreme outliers in *Corridor 3* which could be a result of the blackout that was present for a few frames.

Market

The *Market* results were again similar with regard to RPE. DOTMask did perform slightly better than VDO-SLAM in all sequences except *Market 3*. VDO-SLAM did better in this sequence. Additionally, extreme outliers were present in the results from VDO-SLAM on *Market 1*.



(a) Market 1



(b) Market 2 & 3

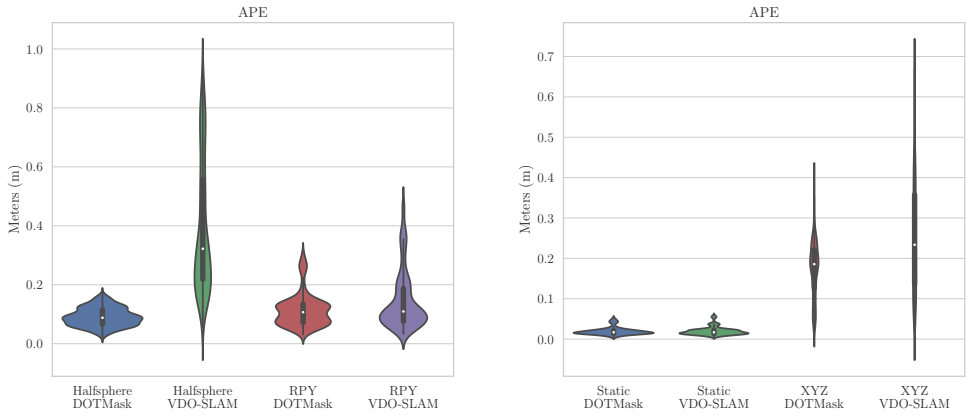
Figure 6.13: The RPE results of the *Market* sequences represented on a violin plot.

6.2.2 ATE results

As mentioned in Chapter 5.2.2, ATE exemplifies the ability of the global accuracy of a system. In other words, the accuracy over a sequence as a whole is evaluated, instead of relative poses as is the case for RPE. The average ATE error over 10 runs on each sequence for both systems was computed. The results are presented in Figures 6.14, 6.15 and 6.16. The figures visualize the distribution of the ATE errors. The median is a white dot placed on the black bar representing the interquartile range. The outliers appear at the ends of the ”violin”.

TUM

The results from the TUM sequences, Figure 6.14, were overall quite similar between the systems. Comparing the results to the RPE results it becomes apparent that DOTMask achieved better global performance than VDO-SLAM even though DOTMask's local estimation proved worse.



(a) *Halfsphere* and *RPY* sequences

(b) *Static* and *XYZ* sequences

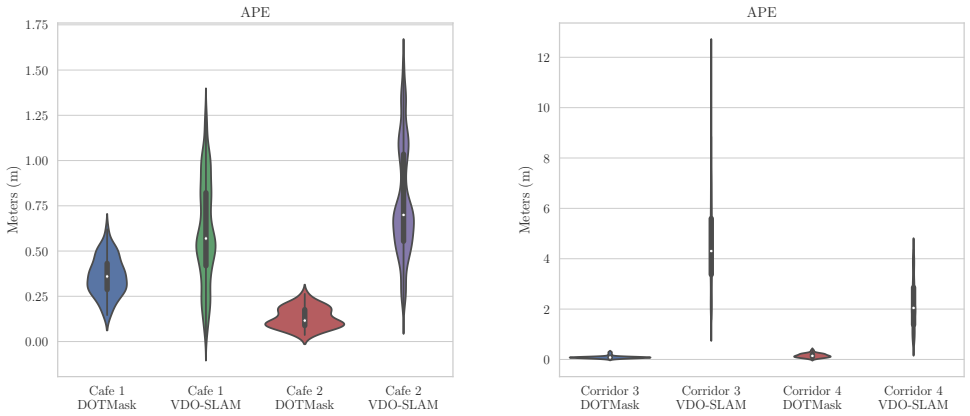
Figure 6.14: The ATE results of the TUM Walking sequences represented on a violin plot.

The performance of VDO-SLAM on the *Halfsphere* sequence was notably worse than the performance of DOTMask with regard to ATE. As Figure 6.14a shows, the distribution entails many outliers within the results. The results were quite similar for the ATE error of both systems on the *RPY* sequence. The distribution of VDO-SLAM was again more spread out but not as extreme as in the case of the *Halfsphere* sequence. The *RPY* results highlight the orientation performance of the systems. As seen in Figure 6.14b, the ATE error was worse for both systems for the *XYZ* movement compared to both the *Halfsphere* and *RPY* movement. The distribution is spread which indicates the systems' global accuracy was affected by the camera movement and the lack of ability to effectively handle the movement while dealing with dynamic objects in the environment. Both systems performed noticeably better on the *Static* sequence where the camera was static as seen in Figure 6.14b.

Looking back at the RPE error from Figure 6.11 it is noteworthy to see that the difference between the two systems is greater for the ATE compared to the RPE. DOTMask performed better in terms of global accuracy in all sequences whereas VDO-SLAM presented better local performance.

Cafe

The ATE results from the *Cafe* sequences are presented in Figure 6.15a. Considering the lack of truly dynamic objects in the *Cafe 1* sequence it is interesting to note that the ATE results for DOTMask were indeed better than VDO-SLAM. DOTMask managed to detect a loop closure within the sequences whereas VDO-SLAM failed. This could also be the reason for the difference in the performance between the systems.



(a) *Cafe* sequences (b) *Corridor* sequences

Figure 6.15: The ATE results of the *Cafe* and *Corridor* sequences represented on a violin plot.

The *Cafe 2* sequence introduces moving people and it is noteworthy to see that the difference in results increased. DOTMask achieves better results in comparison to the *Cafe 1* results. This could be due to the light exposure from the windows and reflection on the floor in *Cafe 1*. However, this sequence also includes static people and could be a further indication of VDO-SLAM not properly detecting immobility. Instead, it regard them as dynamic and hence to be excluded from the feature extraction process, resulting in worse ATE results.

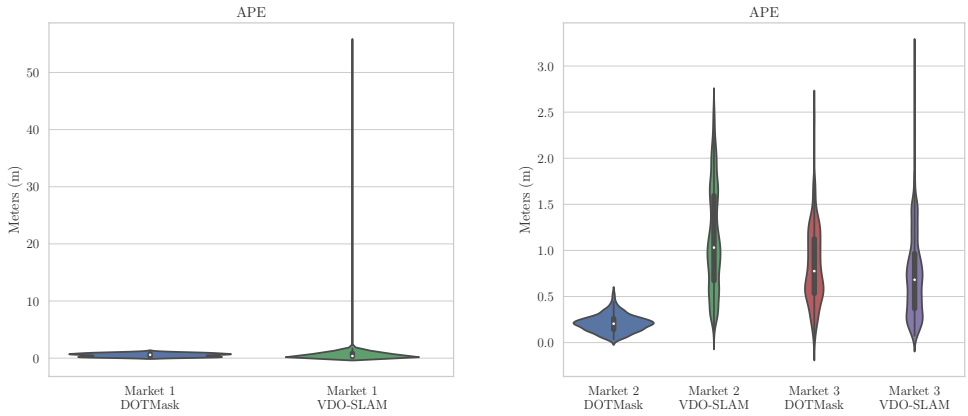
Corridor

The ATE results can be seen in Figure 6.15b. As has been pointed out, neither system managed to extract and track features during the blackout. However, DOTMask managed to recover from the blackout and successfully detect a loop closure. Whereas VDO-SLAM did not recover and failed to detect the loop within the sequence. The results show DOTMask performing much better

than VDO-SLAM on the *Corridor* sequences in terms of global estimation, having fewer outliers and lower average ATE error compared to VDO-SLAM. VDO-SLAM performed better on *Corridor 4* than on *Corridor 3*. This further highlights the effect of the blackout in *Corridor 3* on the system's performance and the lack of recovering via loop closure.

Market

The ATE results from the *Market* sequences are represented in Figure 6.16. Here, DOTMask and VDO-SLAM show similar results. This environment could be classified as highly dynamic due to the amount of moving people within the sequences. The ATE results are more similar between the systems compared to the other sequences. These results indicate that in a highly dynamic environment, it could be enough to classify and exclude potentially dynamic objects without checking their motion. Considering VDO-SLAM's lack of ability to identify potentially dynamic objects as static. What could contribute to the error is the fact that people often stopped for a brief moment while blocking a large area of the camera view. For these couple of frames, the systems sometimes failed to identify the people as static, resulting in no features being extracted from that area.



(a) Market 1

(b) Market 2 & 3

Figure 6.16: The ATE results of the *Market* sequences represented on a violin plot.

As mentioned in Chapter 6.1, VDO-SLAM was observed to be challenged by the feature-rich shelves as it failed to identify features while DOTMask was successful in identifying them. Even though the shelves contained multiple

feature-rich objects they looked quite similar which could have affected the feature extraction process.

6.2.3 Summary of quantitative evaluation

Reflecting on the trajectory error results from both systems, DOTMask proved noticeably stronger global accuracy on the experimented sequences whereas VDO-SLAM proved slightly better performance with regard to local estimation, mainly noticeable in the TUM results. The results highlight the effectiveness of loop closure on the performance as DOTMask correctly detected loop closures where VDO-SLAM failed to do so.

6.3 General comparison

This section presents other observations made from running the systems on the chosen datasets. This involves reflecting on the consistency between the experimental environments and setup.

6.3.1 Robustness

The robustness metric as defined in Chapter 5.3 is the system's ability to avoid crashing by detecting and recovering from environmental disturbances in the form of dynamic objects. The systems showed robustness with regard to dynamic objects, whereas those objects were excluded from the feature extraction process. The TUM sequences showed the dynamic object detection of both systems was affected when facing different types of camera movement. Parallel movement of objects with regard to the camera was difficult as the *Halfsphere* and *XYZ* sequences showed. Robustness in highly dynamic environments such as the *Market* sequences was good considering that it is a difficult environment. Since these systems are optimized for dynamic environments, it is interesting to see the error is slightly higher there compared to the other sequences.

DOTMask proved to be more robust with regard to its absolute trajectory results. The system was observed to track features from static objects and correctly detect loop closures. As discussed in Chapter 6.1, what could have contributed to the degraded performance of VDO-SLAM is the threshold for scene flow. VDO-SLAM was observed to fail in correcting the classification of static objects as dynamic *e.g.,* sitting people. The threshold's value could possibly be fine-tuned for the different types of environments to yield better

results. However, if the performance depends on fine-tuning of a parameter it would make the system brittle since one environment could contain different types of motion.

Consistency between the sequences varied with regard to the quantitative results and feature extraction. The performance was affected when facing perturbations such as light exposure, dark environments and camera movement. However, these perturbations still pose challenges to the field of Visual SLAM although the systems were able to recover to some sense while facing them. DOTMask was less brittle, running on long sequences went smoothly and it showed the ability to get back on track when encountering frames without any feature extraction as mentioned in Chapter 6.1. VDO-SLAM was brittle between datasets, resulting in hard crashes with long data sequences due to bad memory allocation. It managed to recover from frames without feature extraction but consistency between changing environments was not proven successful.

6.3.2 Practicality

As mentioned in Chapter 5.4, evaluating the systems with regard to practicality involved assessing their setup and usage. The following sections present the observations from the implementation and experiments conducted. The observations highlight the practical aspects with regard to real-world application. However, these observations are from a personal perspective and can not be considered completely objective.

DOTMask

DOTMask was easy to set up and execute on the chosen datasets. The system's dependencies were not highly sensitive to library and package versions, and the authors provided a well-documented setup guide. This is important for reproducibility and comparison with other systems on unforeseen data. The system is implemented with ROS which may be a drawback for those unfamiliar with this framework. An advantage of using ROS is the ability to visualize the system's modules using tools like RViz and Rqt, which aid in debugging and integrating new sensors. Visualization is crucial for optimizing algorithms and understanding dynamic objects. The pipeline is simple and modular, allowing easy addition of functionality. This open-source solution facilitates further development in the field. Overall, the implementation of DOTMask was smooth, thanks to well-documented guidance.

VDO-SLAM

VDO-SLAM was not as user-friendly as DOTMask and was sensitive to dependencies' versions. VDO-SLAM was provided with documentation and a Dockerfile for smooth implementation. However, the documentation lacked details on pre-processing. The system did also encounter memory problems when running long data sequences. Dynamic reconstruction methods seem to use more memory compared to the other two approaches as they store more information about the map. Configuring processing steps like image segmentation for SLAM often involves pre-processing, which DOTMask avoids. In contrast, VDO-SLAM required instance segmentation and optical flow computation to match its format. This pre-processing requirement is counterproductive for real-world applications and goes against the aim of improving algorithms to work in unknown environments.

6.4 Further discussions

Dynamic tracking and mapping have advantages for SLAM in terms of performance in dynamic environments and providing additional information to other tasks such as path-planning, obstacle avoidance and object mapping. With regard to application, it should be noted that tracking methods typically require lower computational complexity and less memory since reconstruction methods maintain a more detailed map containing the dynamic objects. The tracking approach provides a simplified reconstruction of the environment which could be advantageous for applications where a detailed reconstruction is not necessary. The dynamic mapping approach provides a more detailed reconstruction of the environment and captures objects' behavior which could be considered advantageous for a comprehensive understanding of the environment and its objects.

To summarize the observations and evaluation within the literature, it is important to acknowledge the limitations imposed by the object detection method capabilities and the method of confirming movement in dynamic environments. Moreover, the field lacks solutions handling a variety of dynamics and consideration for complex examples such as a Ferris wheel, which is an object that should be considered in the reconstruction of an environment.

The theme of pre-processing data is apparent in the literature. However, claiming a system can handle real-time scenarios by handling dynamic objects through pre-segmented inputs is not feasible. Additionally, an intriguing

aspect to consider is the handling of dynamic objects with regard to camera movement.

The scope of application of the existing solution is limited by their performance when facing other perturbations such as difficult lighting. Both DOTMask and VDO-SLAM were challenged throughout the experiments with regard to light exposure and dark frames.

Chapter 7

Conclusions & Future Work

This thesis has presented a survey of the recent work done to solve the challenge of dynamic conditions when deploying VSLAM. It has set forth a systematic evaluation of two open-source state-of-the-art VSLAM algorithms under such conditions. Moreover, metrics were defined in order to evaluate the systems to provide better insight into the state of dynamic solutions within the literature. This chapter will present the main conclusions from the thesis work and the possible future work in the field.

7.1 Conclusions

Developing dynamic SLAM is of great importance due to the forever-changing (non-static) nature of the real world. Assuming a static environment for mapping leads to bad positioning and fraudulent reconstruction.

Advantages

Reflecting on the research question formed in Chapter 1.1.1, the advantages as discussed in Chapter 3 and Chapter 6 are the fact that these systems disregard the assumption of a static world and exclude dynamic features in the extraction process.

Moreover, the tracking and reconstruction approaches provide valuable information if combining SLAM with decision-making tasks such as path planning. Adding object awareness to SLAM offers the ability to predict the movement of an object and take it into account when planning ahead.

Disregarding dynamic objects for reconstruction should offer a simplified map of an environment excluding these dynamic objects that should not be

part of the reconstruction. Meanwhile, including dynamic objects within the reconstruction offers improved scene understanding. The observations and results from experiments show excluding features of dynamic objects in camera pose estimation can provide good results.

Limitations

The main limitations of the existing methods, as pointed out throughout this thesis, is the dependency on heavy-weight software (*e.g.,* image segmentation neural networks implemented with GPU-dependent code) to handle applied object detection methods, pre-processing of input data and lack of consideration for camera movement or difference in dynamics.

It is worth noting that the dependencies are not practical in real-world scenarios. Many solutions are limited to pre-processing of input data prior to execution of the system, powerful hardware and/or low processing rate of data. This behavior is a contradiction to the aim of handling real-world scenarios, considering the real world requires fast processing time.

With regard to camera movement, the field seems to lack consideration of this aspect and the effects it can have on dynamic object detection. A related issue is the confirmation of object movement and the challenge of classifying objects as dynamic or static. The exclusion of static objects due to dynamic misclassification is of concern. Many solutions claim dynamic objects do not cover large areas in images which can in fact be the case. It should be especially important to confirm the motion of objects if a static object is covering a large area and being excluded when in fact it is static. This was observed throughout the experiments, large chunks of areas being wrongfully excluded that could have provided useful information for the estimation process.

Many solutions within the literature apply thresholds for objects' motions for classification depending on whether or not their movement surpasses the threshold. These thresholds are often set low to ensure that non-static objects are considered dynamic. The combination of low motion thresholds and lack of consideration for camera movement was observed throughout the experiments which resulted in dynamic objects being wrongfully classified as static. Hence, I believe the field requires discussion regarding the variability of dynamic objects encountered in real-life and how the motion (*e.g.,* fast, slow, erratic, disappearing, moving yet static) of different objects can affect the outcome.

Even though the literature presents the ability of handling dynamic objects with some accuracy, parameters like light exposure, lack of features and dark

environments are still an issue.

7.2 Future Work

With regard to future work within the field, the results of the review and experiments bring to light the need for consideration regarding how dynamic object detection methods are constructed. This applies to camera movement, the computational requirements of the method and the classification of dynamic objects. As presented in the thesis, camera movement can have a negative effect on dynamic object detection. Accounting for the movement of the camera should improve dynamic object detection by offering information to be extracted from the environment when static objects are misclassified as dynamic. Objects moving parallel with regard to the camera could then also be considered dynamic. With regard to the weight, this might rely on the field of object detection since the literature shows solutions picking the newest state-of-the-art at each time to operate with their SLAM. Moreover, for the continued development within this field, I believe a better understanding of the variations between the dynamic objects encountered in the real world is required.

With the continuous development of dynamic SLAM, it is clear that more evaluation datasets will need to contain ground truth for the poses of objects as well as the camera in order to be able to evaluate the dynamic object detection in parallel to the estimated camera trajectory. This should offer optimization in dynamic object detection. The limited availability of scenes to test systems is also a hindrance when it comes to testing systems' recognition of dynamic objects and how they handle them. For example, it would be interesting to evaluate these approaches in environments containing complex objects (*e.g.,* escalators). It would also be of interest to feed a system map of a previously explored scene where static objects have been moved or removed.

References

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016. doi: 10.1109/TRO.2016.2624754. [Online]. Available: https://ieeexplore.ieee.org/document/7747236/ [Pages 1, 5, 6, and 32.]

[2] B. Siciliano, O. Khatib, and T. Kröger, *Springer Handbook of Robotics*. Springer, 2008, vol. 200. [Page 6.]

[3] C. Rui, Y. Liu, J. Shen, Z. Li, and Z. Xie, "A Multi-Sensory Blind Guidance System Based on YOLO and ORB-SLAM," in *2021 IEEE International Conference on Progress in Informatics and Computing (PIC)*, 2021. doi: 10.1109/PIC53636.2021.9687018 pp. 409–414. [Page 7.]

[4] Y. Wang and S. Huang, "Towards dense moving object segmentation based robust dense RGB-D SLAM in dynamic scenarios," in *2014 13th International Conference on Control Automation Robotics & Vision (ICARCV)*. Singapore: IEEE, Dec. 2014. doi: 10.1109/ICARCV.2014.7064596. ISBN 978-1-4799-5199-4 pp. 1841–1846. [Online]. Available: http://ieeexplore.ieee.org/document/7064596/ [Page 7.]

[5] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object Detection With Deep Learning: A Review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3212–3232, 2019. doi: 10.1109/TNNLS.2018.2876865 [Page 7.]

[6] S. Minaee, Y. Y. Boykov, F. Porikli, A. J. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image Segmentation Using Deep Learning: A Survey,"

IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 1–1, 2021. doi: 10.1109/TPAMI.2021.3059968 [Page 7.]

[7] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, Apr. 2018. doi: 10.1109/TPAMI.2017.2699184 [Page 7.]

[8] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," 2017, pp. 2961–2969. [Online]. Available: https://openaccess.thecvf.com/content_iccv_2017/html/He_Mask_R-CNN_ICCV_2017_paper.html [Page 8.]

[9] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "YOLACT: Real-Time Instance Segmentation," 2019, pp. 9157–9166. [Online]. Available: https://openaccess.thecvf.com/content_ICCV_2019/html/Bolya_YOLACT_Real-Time_Instance_Segmentation_ICCV_2019_paper.html [Page 8.]

[10] F. Li, H. Zhang, H. Xu, S. Liu, L. Zhang, L. M. Ni, and H.-Y. Shum, "Mask dino: Towards a unified transformer-based framework for object detection and segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 3041–3050. [Page 8.]

[11] R. Szeliski, *Computer Vision - Algorithms and Applications, Second Edition*, ser. Texts in Computer Science. Springer, 2022. [Page 8.]

[12] S. D. Roy and M. K. Bhowmik, "A Comprehensive Survey on Computer Vision Based Approaches for Moving Object Detection," in *2020 IEEE Region 10 Symposium (TENSYMP)*, 2020. doi: 10.1109/TENSYMP50017.2020.9230869 pp. 1531–1534. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9230869 [Page 8.]

[13] B. D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 674–679, 1981. [Page 9.]

[14] Jianbo Shi and Tomasi, "Good features to track," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition CVPR-94*. Seattle, WA, USA: IEEE Comput. Soc. Press, 1994. doi: 10.1109/CVPR.1994.323794. ISBN 978-0-8186-5825-9 pp. 593–600. [Online]. Available: http://ieeexplore.ieee.org/document/323794/ [Page 9.]

[15] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *2011 International Conference on Computer Vision*. Barcelona, Spain: IEEE, Nov. 2011. doi: 10.1109/ICCV.2011.6126544. ISBN 978-1-4577-1102-2 978-1-4577-1101-5 978-1-4577-1100-8 pp. 2564–2571. [Online]. Available: http://ieeexplore.ieee.org/document/6126544/ [Page 9.]

[16] G. Farnebäck, "Two-Frame Motion Estimation Based on Polynomial Expansion," in *Image Analysis*, J. Bigun and T. Gustavsson, Eds. Berlin, Heidelberg: Springer, 2003. doi: 10.1007/3-540-45103-X_50. ISBN 9783540451037 pp. 363–370. [Pages 9 and 12.]

[17] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, "FlowNet: Learning Optical Flow With Convolutional Networks," 2015, pp. 2758–2766. [Online]. Available: https://openaccess.thecvf.com/content_iccv_2015/html/Dosovitskiy_FlowNet_Learning_Optical_ICCV_2015_paper.html [Page 10.]

[18] Z. Teed and J. Deng, "Raft: Recurrent all-pairs field transforms for optical flow," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*. Springer, 2020, pp. 402–419. [Page 10.]

[19] C. Bailer, B. Taetz, and D. Stricker, "Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015. [Page 10.]

[20] M. A. Mohamed, H. A. Rashwan, B. Mertsching, M. A. García, and D. Puig, "Illumination-robust optical flow using a local directional pattern," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 9, pp. 1499–1508, 2014. doi: 10.1109/TCSVT.2014.2308628 [Page 10.]

[21] M. Kaneko, K. Iwami, T. Ogawa, T. Yamasaki, and K. Aizawa, "Mask-SLAM: Robust feature-based monocular SLAM by masking using semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2018, pp. 258–266. [Page 11.]

[22] C. Yu, Z. Liu, X.-J. Liu, F. Xie, Y. Yang, Q. Wei, and Q. Fei, "DS-SLAM: A Semantic Visual SLAM towards Dynamic Environments," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid: IEEE, Oct. 2018. doi: 10.1109/IROS.2018.8593691. ISBN 978-1-5386-8094-0 pp. 1168–1174. [Online]. Available: https://ieeexplore.ieee.org/document/8593691/ [Page 11.]

[23] X. Long, W. Zhang, and B. Zhao, "PSPNet-SLAM: A Semantic SLAM Detect Dynamic Object by Pyramid Scene Parsing Network," *IEEE Access*, vol. 8, pp. 214 685–214 695, 2020. doi: 10.1109/ACCESS.2020.3041038 [Page 12.]

[24] Y. Ai, T. Rui, M. Lu, L. Fu, S. Liu, and S. Wang, "DDL-SLAM: A Robust RGB-D SLAM in Dynamic Environments Combined With Deep Learning," *IEEE Access*, vol. 8, pp. 162 335–162 342, 2020. doi: 10.1109/ACCESS.2020.2991441 [Pages 12 and 14.]

[25] J. Chang, N. Dong, and D. Li, "A Real-Time Dynamic Object Segmentation Framework for SLAM System in Dynamic Scenes," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–9, 2021. doi: 10.1109/TIM.2021.3109718. [Online]. Available: https://ieeexplore.ieee.org/document/9527213/ [Pages 12 and 14.]

[26] Z. Hu, J. Zhao, Y. Luo, and J. Ou, "Semantic SLAM Based on Improved DeepLabv3 in Dynamic Scenarios," *IEEE Access*, vol. 10, pp. 21 160–21 168, 2022. doi: 10.1109/ACCESS.2022.3154086. [Online]. Available: https://ieeexplore.ieee.org/document/9721010/ [Pages 12 and 14.]

[27] L. Cui and C. Ma, "SDF-SLAM: Semantic Depth Filter SLAM for Dynamic Environments," *IEEE Access*, vol. 8, pp. 95 301–95 311, 2020. doi: 10.1109/ACCESS.2020.2994348. [Online]. Available: https://ieeexplore.ieee.org/document/9093003/ [Pages 12 and 13.]

[28] Y. Liu and J. Miura, "RDMO-SLAM: Real-Time Visual SLAM for Dynamic Environments Using Semantic Label Prediction With

Optical Flow," *IEEE Access*, vol. 9, pp. 106 981–106 997, 2021. doi: 10.1109/ACCESS.2021.3100426. [Online]. Available: https://ieeexplore.ieee.org/document/9497091/ [Pages 12, 13, and 14.]

[29] S. Miao, X. Liu, D. Wei, and C. Li, "A visual SLAM robust against dynamic objects based on hybrid semantic-geometry information," *ISPRS International Journal of Geo-Information*, vol. 10, no. 10, p. 673, 2021. [Page 12.]

[30] X. Zhao, T. Zuo, and X. Hu, "OFM-SLAM: A Visual Semantic SLAM for Dynamic Indoor Environments," *Mathematical Problems in Engineering*, vol. 2021, 2021. [Pages 12, 13, and 14.]

[31] D. Lai, C. Li, and B. He, "YO-SLAM: A Robust Visual SLAM towards Dynamic Environments," in *2021 International Conference on Communications, Information System and Computer Engineering (CISCE)*. Beijing, China: IEEE, May 2021. doi: 10.1109/CISCE52179.2021.9445920. ISBN 978-1-66540-352-8 pp. 720–725. [Online]. Available: https://ieeexplore.ieee.org/document/9445920/ [Pages 12, 13, and 14.]

[32] Y. Wang, X. Duan, Y. Sun, and J. Wang, "A Visual SLAM Algorithm Based on Image Semantic Segmentation in Dynamic Environment," in *2021 4th International Conference on Pattern Recognition and Artificial Intelligence (PRAI)*. Yibin, China: IEEE, Aug. 2021. doi: 10.1109/PRAI53619.2021.9550800. ISBN 978-1-66541-322-0 pp. 401–405. [Online]. Available: https://ieeexplore.ieee.org/document/9550800/ [Pages 12, 13, and 14.]

[33] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vilamoura-Algarve, Portugal: IEEE, Oct. 2012. doi: 10.1109/IROS.2012.6385773. ISBN 978-1-4673-1736-8 978-1-4673-1737-5 978-1-4673-1735-1 pp. 573–580. [Online]. Available: http://ieeexplore.ieee.org/document/6385773/ [Pages 14, 24, and 30.]

[34] J. Huang, S. Yang, Z. Zhao, Y.-K. Lai, and S. Hu, "ClusterSLAM: A SLAM Backend for Simultaneous Rigid Body Clustering and Motion Estimation," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. doi: 10.1109/ICCV.2019.00597 pp. 5874–5883. [Pages 14 and 16.]

[35] J. Vincent, M. Labbe, J.-S. Lauzon, F. Grondin, P.-M. Comtois-Rivet, and F. Michaud, "Dynamic Object Tracking and Masking for Visual SLAM," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Las Vegas, NV, USA: IEEE, Oct. 2020. doi: 10.1109/IROS45743.2020.9340958. ISBN 978-1-72816-212-6 pp. 4974–4979. [Online]. Available: https://ieeexplore.ieee.org/document/9340958/ [Pages 14, 15, 16, 19, 21, 22, 33, and 65.]

[36] I. Ballester, A. Fontan, J. Civera, K. H. Strobl, and R. Triebel, "DOT: Dynamic Object Tracking for Visual SLAM," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. Xi'an, China: IEEE, May 2021. doi: 10.1109/ICRA48506.2021.9561452. ISBN 978-1-72819-077-8 pp. 11 705–11 711. [Online]. Available: https://ieeexplore.ieee.org/document/9561452/ [Pages 14, 15, and 16.]

[37] J. Zhang, M. Henein, R. Mahony, and V. Ila, "Robust Ego and Object 6-DoF Motion Estimation and Tracking," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Las Vegas, NV, USA: IEEE, Oct. 2020. doi: 10.1109/IROS45743.2020.9341552. ISBN 978-1-72816-212-6 pp. 5017–5023. [Online]. Available: https://ieeexplore.ieee.org/document/9341552/ [Pages 15, 16, and 17.]

[38] A. Kundu, K. M. Krishna, and C. V. Jawahar, "Realtime multibody visual SLAM with a smoothly moving monocular camera," in *2011 International Conference on Computer Vision*. Barcelona, Spain: IEEE, Nov. 2011. doi: 10.1109/ICCV.2011.6126482. ISBN 978-1-4577-1102-2 978-1-4577-1101-5 978-1-4577-1100-8 pp. 2080–2087. [Online]. Available: http://ieeexplore.ieee.org/document/6126482/ [Pages 17 and 19.]

[39] M. Henein, J. Zhang, R. Mahony, and V. Ila, "Dynamic SLAM: The Need For Speed," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020. doi: 10.1109/ICRA40945.2020.9196895 pp. 2123–2129. [Pages 17 and 19.]

[40] N. D. Reddy, P. Singhal, V. Chari, and K. M. Krishna, "Dynamic body VSLAM with semantic constraints," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Hamburg, Germany: IEEE, Sep. 2015. doi: 10.1109/IROS.2015.7353626. ISBN 978-1-4799-9994-1 pp. 1897–1904. [Online]. Available: http://ieeexplore.ieee.org/document/7353626/ [Page 17.]

[41] M. Rünz and L. Agapito, "Co-fusion: Real-time segmentation, tracking and fusion of multiple objects," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. Singapore, Singapore: IEEE, May 2017. doi: 10.1109/ICRA.2017.7989518. ISBN 978-1-5090-4633-1 pp. 4471–4478. [Online]. Available: http://ieeexplore.ieee.org/document/7989518/ [Page 17.]

[42] I. A. Barsan, P. Liu, M. Pollefeys, and A. Geiger, "Robust Dense Mapping for Large-Scale Dynamic Environments," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. Brisbane, QLD: IEEE, May 2018. doi: 10.1109/ICRA.2018.8462974. ISBN 978-1-5386-3081-5 pp. 7510–7517. [Online]. Available: https://ieeexplore.ieee.org/document/8462974/ [Pages 17 and 19.]

[43] J. Zhang, M. Henein, R. Mahony, and V. Ila, "VDO-SLAM: A Visual Dynamic Object-Aware SLAM System," *arXiv preprint arXiv:2005.11052*, 2020. [Pages 17, 19, 21, 22, and 33.]

[44] B. Bescos, C. Campos, J. D. Tardós, and J. Neira, "DynaSLAM II: Tightly-Coupled Multi-Object Tracking and SLAM," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5191–5198, 2021. doi: 10.1109/LRA.2021.3068640 [Pages 17, 18, and 19.]

[45] X. Shi, D. Li, P. Zhao, Q. Tian, Y. Tian, Q. Long, C. Zhu, J. Song, F. Qiao, L. Song, Y. Guo, Z. Wang, Y. Zhang, B. Qin, W. Yang, F. Wang, R. H. M. Chan, and Q. She, "Are We Ready for Service Robots? The OpenLORIS-Scene Datasets for Lifelong SLAM," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. Paris, France: IEEE, May 2020. doi: 10.1109/ICRA40945.2020.9196638. ISBN 978-1-72817-395-5 pp. 3139–3145. [Online]. Available: https://ieeexplore.ieee.org/document/9196638/ [Pages 25 and 32.]

[46] M. Bujanca, X. Shi, M. Spear, P. Zhao, B. Lennox, and M. Lujan, "Robust SLAM Systems: Are We There Yet?" in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Prague, Czech Republic: IEEE, Sep. 2021. doi: 10.1109/IROS51168.2021.9636814. ISBN 978-1-66541-714-3 pp. 5320–5327. [Online]. Available: https://ieeexplore.ieee.org/document/9636814/ [Page 32.]

[47] M. Labbé and F. Michaud, "Multi-session visual slam for illumination invariant re-localization in indoor environments," *Frontiers in Robotics and AI*, p. 115, 2022. [Page 65.]

Appendix A

Implementation

This chapter explains the steps taken to get the two systems running and insights into the present solutions within the field. The section also provides the details regarding necessary pre-processing steps required for the systems to process video sequence data. The specifications are summarized in Table A.1.

Table A.1: The platform environment used for experiments.

Environment	
Operating System	Ubuntu 18.04
Central Processing Unit	AMD Ryzen 5 5600H
Graphics Processing Unit	Nvidia GeForce RTX 3060, 6 GB
Memory	32 GB RAM

A.1 DOTMask

DOTMask [35] is a front-end system based on RTAB-Map SLAM [47]. The system has well-documented software and package dependencies, although the specific versions used for experiments are not stated. Table A.2 lists the dependencies used for the thesis work. The system is implemented with Python 3 and ROS melodic, which can be set up to work together with the provided guide. The comprehensive documentation allows for a smooth implementation.

It should be noted that one of the system's dependencies is RTAB-Map SLAM. As DOTMask is a front-end system it requires a SLAM system to

Table A.2: Dependencies for DOTMask.

Library and Packages	Version
Catkin_pkg	0.5.2
Cuda	11.3.1
Cython	0.29.33
Matplotlib	3.3.4
OpenCV	3.4.0
Pillow	9.4.0
Pycocotools	2.0.6
Python	3.6.9
PyTorch	1.12.1
PyQt	5.15.7
Rospkg	1.4.0
ROS	Melodic
Rtabmap-ros-melodic	0.20.22
Scikit-Learn	0.24.2
Torchvision	0.13.1

operate. Therefore, DOTMask could be set up with another Visual SLAM system. RTAB-Map is a SLAM approach that has a ROS-supported version as well as a standalone application. Both come with concrete installation guides. The authors do not specify which version they use but the ROS version[1] of RTAB-Map is used for the experiments in this thesis. DOTMask was built and tested with three different segmentation Neural Networks; YOLACT, YOLACT++ and MaskRCNN. YOLACT was chosen due to its lightweight nature compared to YOLACT++ and MaskRCNN. The different methods are documented in DOTMask's installation guide.

To run the system with a new dataset or camera sensor, the launch file needs to be configured to publish the relevant sensor data and frames. DOTMask requires input data in rosbag format, but image data can be recorded into a rosbag file. The launch file configuration involves subscribing to the correct topics for depth images, RGB images, and camera information. The image topics should contain the appropriate tf frame information, and the camera info topic should be published continuously. Modifications to the launch file may be necessary to ensure continuous camera info publishing.

[1] Available at https://wiki.ros.org/rtabmap_ros

TUM

The TUM sequences are provided as zip files in the ”known” TUM format and as rosbags. The rosbags were used for DOTMask as the system is implemented with ROS as mentioned above. The system was tested with sequences from the TUM dataset and therefore minimal modifications, defining the camera calibration parameters, were required for the system to process the Walking sequences. The estimated trajectory output depends on the SLAM system used with DOTMask, in this case RTAB-Map. RTAB-Map output trajectory is in the TUM format.

OpenLoris

The official website of OpenLoris offers the data sequences both as rosbags and as zip files. The rosbags were used for DOTMask as the system is implemented with ROS. The OpenLoris sequences were recorded with a different camera module and modifications were required such that DOTMask could process the topics from the rosbag. The OpenLoris rosbags were filtered to the topics *d400/color/image_raw* and *d400/aligned_depth_to_color/image_raw*. The camera intrinsics were published through the systems launch file with the extrinsic connections between rgb and depth frames.

A.2 VDO-SLAM

VDO-SLAM is a complete SLAM system and its dependencies are listed in Table A.3. The dependencies vary between the choices of instance segmentation and optical flow methods as VDO-SLAM requires mask and flow files as input.

Table A.3: Dependencies for VDO-SLAM.

Software	Version
Cuda	11.3.1
Cython	0.29.33
Eigen	3.3
GCC	9.2.1
G2O	Custom
Matplotlib	3.5.3
OpenCV	3.4.0
Pillow	9.4.0
Pycocotools	2.0.6
PyTorch	1.12.1
Python	3.6.9
PyQt	5.15.7
Scipy	1.10.0
Tensorboard	2.10.0
Torchvision	0.13.1

The implementation of VDO-SLAM relies on specific versions of dependencies but the authors provide a Dockerfile for straightforward setup. The system was implemented on Ubuntu 18.04. The authors mention it should be easy to compile VDO-SLAM to other platforms. However, the pre-processing methods may be limited to GPU hardware due to computational requirements and dependencies. Otherwise, it could affect the execution time or require modifications to the methods if they are dependent on a certain hardware *e.g.,* cudatoolkit with cuda. The OpenCV library is used for image and feature processing in both DOTMask and VDO-SLAM. It was proven unsuccessful to implement the system with versions other than those stated in the documentation from the authors. The authors modify the g2o library in order to perform optimization in their pipeline. The details of the modifications are not explicitly described but VDO-SLAM was incompatible with the original g2o library was incompatible. However, the custom version is included in their repository and was used for the experiments. YOLACT and RAFT were used for data pre-processing, sharing common dependencies of the same versions as DOTMask.

The data input should be synchronized i.e. every rgb frame requires a corresponding depth frame. Fortunately, there are open-source tools that can

associate rgb frames to their corresponding depth frames[1].

As with other visual SLAM systems, the system requires information regarding camera calibration. The scene flow threshold was defined with the camera calibration. Throughout the experiments the threshold was set to 0.12 for both the TUM and OpenLoris sequences. The system outputs the trajectory in a custom file format and can process the ground truth of the camera and object poses as input to perform an internal evaluation. However, the internal evaluation tool was not used for experiments. The system's output was converted to the known TUM format such that it could be evaluated with the open-source evaluation tool, *evo*[2].

TUM

As mentioned above, VDO-SLAM provides an internal evaluation for each run. Moreover, the system requires ground truth data to run. Since TUM data does not contain the ground truth of object poses, empty ground truth files of object poses were used such that VDO-SLAM would run. The camera poses' ground truth was changed to the custom format the system could process. The internal evaluation was not used due to the absence of object ground truth and the preference for coherent evaluation across both systems. Color and depth images were synchronized with their corresponding timestamps; such that every rgb image had a corresponding depth image. Optical flow files were generated using RAFT on each data sequence, stored and used as input to VDO-SLAM. Segmentation masks were generated using YOLACT on each data sequence and used for input to VDO-SLAM.

OpenLoris

The integration with OpenLoris data was similar to the TUM data. Empty ground truth object pose files were used and modification was made to ground truth camera poses. The dataset is synchronized with every rgb image having a corresponding depth image. Therefore, synchronization was not required. Optical flow files were generated using RAFT on each data sequence, stored and used as input to VDO-SLAM. Segmentation masks were generated using YOLACT on each data sequence, stored and used for input to VDO-SLAM. The OpenLoris sequences are large (at least compared to TUM) and VDO-

[1] Available at https://svncvpr.in.tum.de/cvpr-rospkg/trunk/rgbd_benchmark/rgbd_benchmark_tools/src/rgbd_benchmark_tools/associate.py [2] Available at https://github.com/MichaelGrupp/evo

SLAM did not crash due to detecting a memory leak. This should depend on the hardware at hand and was discussed in Chapter 6.

TRITA – EECS-EX-2023:572