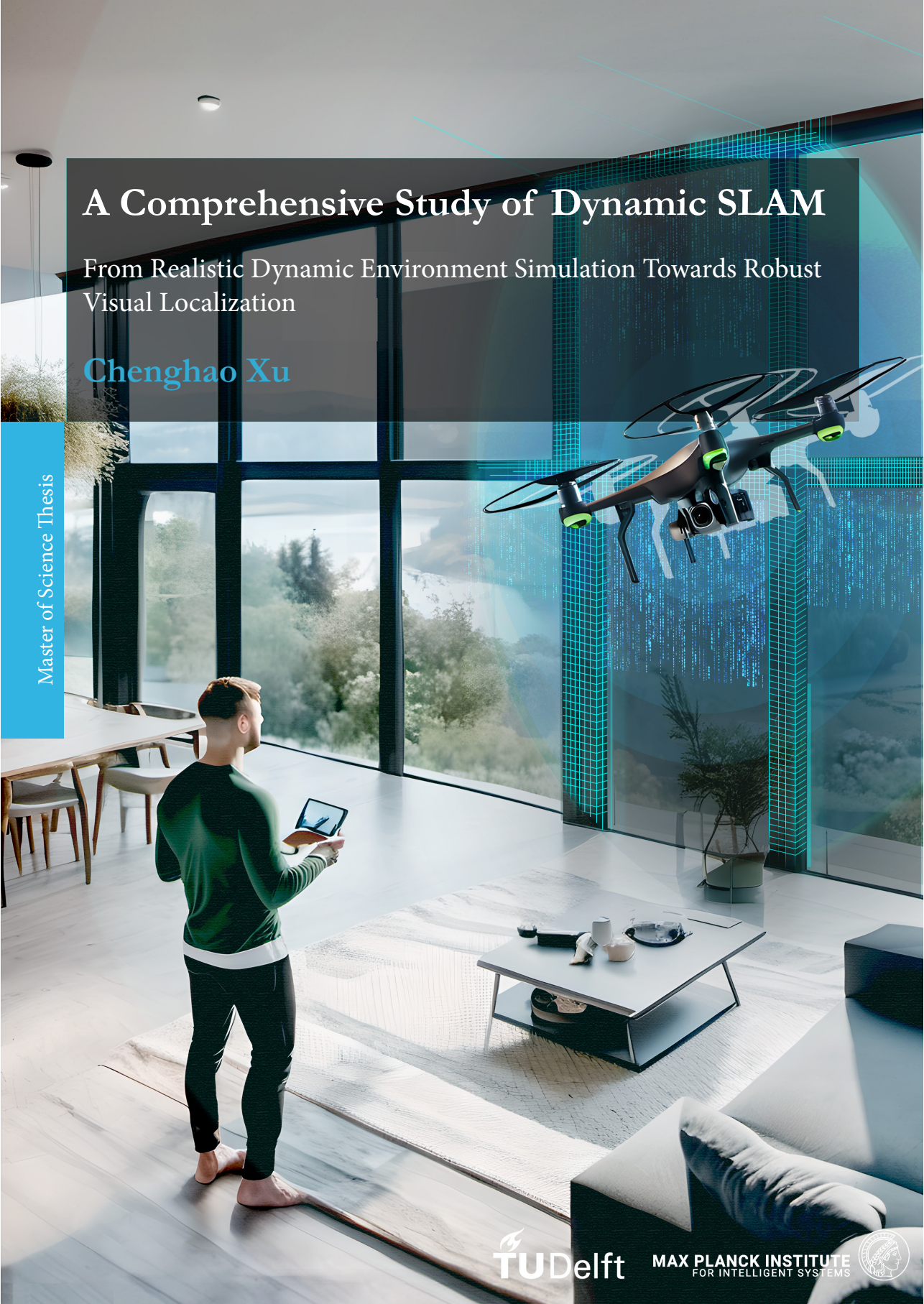
A Comprehensive Study of Dynamic SLAM

From Realistic Dynamic Environment Simulation Towards Robust Visual Localization

Chenghao Xu

Master of Science Thesis



TU Delft

MAX PLANCK INSTITUTE FOR INTELLIGENT SYSTEMS

A Comprehensive Study of Dynamic SLAM

From Realistic Dynamic Environment Simulation Towards Robust Visual Localization

Master of Science Thesis

For the degree of Master of Science in Robotics at Delft University of Technology

Chenghao Xu

November 8, 2023

Thesis Committee:	Dr. Javier Alonso-Mora	TU Delft
	Prof. Dr. Ir. Martijn Wisse	TU Delft
	Dr. Manon Kok	TU Delft
	Jun.-Prof. Dr.-Ing. Aamir Ahmad	University of Stuttgart
	M.Sc. Elia Bonetto	MPI for Intelligent Systems

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

This thesis work was supported by the Erasmus+ Traineeship Programme and Max Planck Institute for Intelligent Systems. Their cooperation is hereby gratefully acknowledged.

An electronic version of this dissertation is available at
http://repository.tudelft.nl/.

ABSTRACT

In recent years, visual Simultaneous Localization and Mapping (SLAM) have gained significant attention and found wide-ranging applications in diverse scenarios. Recent advances in computer vision and deep learning also enrich visual SLAM capabilities in scene understanding and large-scale operation. However, despite remarkable performance in these fields, most visual SLAM frameworks are designed with the static world assumption. Thus, they often confront challenges in dynamic environments, manifesting reduced localization accuracy, tracking failures, and restricted generalization.

To investigate the impact of moving objects in dynamic indoor environments, we first benchmark representative visual (dynamic) SLAM approaches, complemented by robustness assessments for preliminary insights. During this process, we adopt challenging sequences from GRADE, an ideal platform for simulating dynamic indoor scenes. Notably, the mainstream of dynamic SLAM methods employs detection or segmentation techniques as solutions. To explore the correlation between detector accuracy and overall SLAM performance, we integrate a series of trained YOLOv5 and Mask R-CNN models, each with varying accuracy levels, into dynamic SLAM systems. Subsequently, we evaluate these configurations on the TUM RGB-D sequences. Contrary to common intuition, the experiments indicate that more accurate object detectors do not necessarily lead to improved visual SLAM performance. This benchmarking process also illuminates several inherent limitations of current dynamic SLAM techniques, underscoring the imperative for further advancements.

Building upon these insights, we introduce DynaPix SLAM, an innovative visual SLAM system for dynamic indoor environments, where participation of visual cues (e.g., features) is weighted based on per-pixel motion probability values. Our approach consists of a semantic-free pixel-wise motion estimation module and an improved pose optimization process. In the first stage, our motion probability estimator employs a novel static background differencing method on both images and optical flows to identify moving regions. These probabilities are then incorporated into the map point selection and weighted bundle adjustment for backend optimization. We evaluate our DynaPix SLAM and its variant, DynaPix-D, in comparison with ORB-SLAM2 and DynaSLAM. These assessments are performed on both TUM RGB-D and GRADE sequences, with additional tests on the static versions of the GRADE ones. The results demonstrate that DynaPix SLAM consistently outperforms the other methods, showcasing reduced localization errors and longer tracking durations across various scenarios.

ACKNOWLEDGEMENTS

This work, focusing on visual Simultaneous Localization and Mapping (SLAM) for indoor dynamic environments, was developed during my internship at the Flight Robotics and Perception Group of Max-Planck Institute for Intelligent Systems, Tübingen, under the joint supervision of Dr. Javier Alonso-Mora, Associate Professor at the Delft University of Technology; Dr.-Ing. Aamir Ahmad, Junior Professor at the University of Stuttgart; and Ph.D. student Elia Bonetto.

As this remarkable journey reaches its final chapter, I cherish every joy, challenge, and effort of the past three years. Although this path may have spanned longer than peers, I take pride in my growth and transformation, from a fresher struggling with coding to an independent researcher as an incoming PhD student. I am deeply grateful to every person along this journey, and for their support, recognition, and encouragement.

I would like to thank my supervisors, Elia Bonetto and Dr.-Ing. Aamir Ahmad, for their unwavering support and mentorship throughout the year. They grant me this fantastic opportunity to delve into the research atmosphere at the Max Planck Institute for Intelligent Systems. I also express my heartfelt gratitude to Dr. Javier Alonso-Mora, for his continuous assistance and understanding during the entire process. Their guidance has been instrumental in shaping my aspirations to embark on this academic journey and in refining my approach to scientific research.

Meanwhile, I am grateful to all my friends in both Delft and Tübingen, for their companionship and assistance. The bonds forged during this time are invaluable and stand as one of my cherished experiences. Their insights, passions, and achievements continually inspire me to pursue impactful research and enjoy the discovery process like them.

Lastly, I wish to convey my sincere gratitude to my family and Ms. Yizhuo Zhang. Indeed, this journey hasn't always been smooth and has presented several challenges. Your unwavering love and encouragement have always been my anchor, consistently helping me reach this significant milestone of graduation.

Delft University of Technology — Chenghao Xu

November 8, 2023

The main contents of this thesis work have been formulated into several scientific papers for submission and possible publication. The author acknowledges the possibility of overlap with these papers and apologizes for any inconvenience caused by such repetition. The applied data and benchmarking results are available on our project page: https://grade.is.tue.mpg.de/.

The visual SLAM benchmarking (Chapter 4) is correlated to the following papers:

- Elia Bonetto, Chenghao Xu, and Aamir Ahmad. *GRADE: Generating Realistic Animated Dynamic Environments for Robotics Research.* 2023. DOI: 10.48550/ARXIV.2303.04466. URL: https://arxiv.org/abs/2303.04466.
- Elia Bonetto, Chenghao Xu, and Aamir Ahmad. "Simulation of Dynamic Environments for SLAM". In: *ICRA 2023 Workshop on Active Methods in Autonomous Navigation.* June 2023. URL: https://arxiv.org/abs/2305.04286.
- Elia Bonetto, Chenghao Xu, and Aamir Ahmad. "Learning From Synthetic Data Generated with GRADE". In: *ICRA 2023 Workshop on Pretraining for Robotics (PT4R).* June 2023. URL: https://openreview.net/forum?id=SUIOuV2y-Ce.

The relevant code and implementation details for this chapter are open-source at: https://github.com/robot-perception-group/GRADE_tools.git.

The work presented in DynaPix SLAM (Chapter 5) corresponds to:

- Chenghao Xu*, Elia Bonetto*, and Aamir Ahmad. *DynaPix SLAM: A Pixel-Based Dynamic SLAM Approach.* 2023. arXiv: 2309.09879. URL: https://arxiv.org/abs/2309.09879.

The source code and results related to this chapter will be made available soon [1].

* Chenghao Xu and Elia Bonetto contributed equally to this work as first authors

[1] The repository will be available at: https://github.com/robot-perception-group/DynaPix.git

CONTENTS

List of Figures

LIST OF TABLES

1 INTRODUCTION

1.1 BACKGROUND

Simultaneous Localization and Mapping (SLAM) technology has undergone significant development over the past three decades since the concept was initially proposed in 1986 [1]. The SLAM process involves simultaneous estimation of robot states and map construction of operation environments using onboard sensors, where the constructed maps can inform path planning for robot navigation and support higher-level representations to enable human-robot interaction [2]. Thus, SLAM algorithms currently find wide-ranging applications in various scenarios, including indoor service robots [3], autonomous vehicles [4], aerial robot navigation [5], and augmented reality devices [6].

To perceive surrounding environments, a wide range of sensors are utilized in SLAM systems, including Laser Rangefinder (LRF), Light Detection and Ranging (LiDAR), Radio Detection and Ranging (Radar), cameras, and more [7]. Among these, cameras provide rich visual information about the environment at a relatively low cost compared to other sensor modalities. As a result, visual SLAM [8] has received tremendous attention in recent years, mainly in the form of monocular [9], RGB-D [10], or stereo-based [11] methods. Furthermore, recent advances in deep learning empower visual SLAM systems [12, 13, 14] to integrate cutting-edge computer vision techniques, such as recognition, detection, and segmentation. This enhances scene comprehension and incorporates semantic information into the SLAM process, thereby enabling more advanced applications.

(a) Point cloud map (b) Densely reconstructed map (c) Semantically annotated map

Figure 1.1: Illustration of SLAM applications in the real world. (a) exhibits the outdoor point cloud map generated by 3D LiDAR SLAM [15]. (b) shows the reconstructed office from the dense visual SLAM system Elastic Fusion [16]. (c) displays the modern semantic map generated by Kimera [14]

With the growing demand for applications in large-scale scenes, numerous visual SLAM frameworks adopt the graph-based formulation [17] to achieve higher accuracy and efficiency, as opposed to the filter-based approaches [18, 19]. To facilitate this real-

world deployment, techniques such as bundle adjustment (BA) [20], pose graph optimization [21], and place recognition (for loop closure) [22] have been introduced to reduce increasing computational complexity and accumulated localization errors. Thus, current visual SLAM frameworks typically comprise the following core modules: i) *Front End Visual Odometry*, ii) *Back End Optimization*, iii) *Loop Closure Detection*, and iv) *Mapping*, as shown in Figure 1.2. These modules are designed with the assumption of an ideally static world [23, 24], where changes between adjacent frames are only due to robot/camera movement. In other words, the observed landmarks (3D map points registered from image features) should remain static and unchanged.

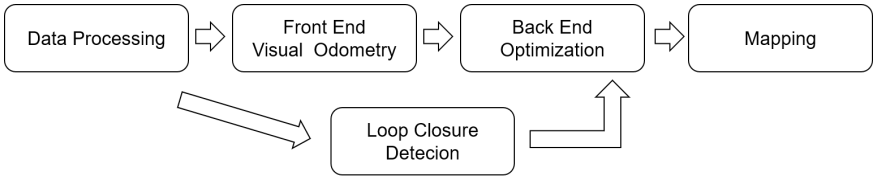


Figure 1.2: Architecture of Classic Visual SLAM Framework [8].

However, real-world scenarios inevitably involve many short-term dynamic entities (e.g., pedestrians or vehicles) and variations over longer time scales (e.g., seasonal changes or altered placements). The presence of these variations directly induces misleading visual information, including false associations between visual features and landmarks, incorrect place recognition, and sensor occlusions. Such disruptions severely impact, and in some cases, even terminate the operation of core modules like state estimation and loop closure detection. Consequently, dynamically changing scenes can cause a significant **degradation in both estimation accuracy and system robustness** of these visual SLAM systems.

Moreover, although recent approaches have demonstrated remarkable accuracy in well-established benchmark datasets like TUM RGB-D [25] and KITTI [26], most visual SLAM algorithms demand extensive tuning to adapt to diverse environments [2]. Even after such fine-tuning, these well-designed systems can underperform or fail in challenging scenarios, such as highly dynamic/deformable scenes, textureless regions, harsh environments, or extreme weather/lighting conditions. The **limited generalization capability** can be attributed to the scarce availability of real-world data and the considerable simulation-to-reality gap. Specifically, this gap is primarily characterized by issues concerning visual realism, controllable customization, and the absence of featured entities in these scenarios (e.g., moving humans, non-rigid objects, fog, snow).

1.2 MOTIVATION

Despite several studies utilizing advanced detection [13, 27, 28] or learning [29, 30] techniques to address the aforementioned challenges, such as dynamic entities, long-term variations, and generalized performance, these issues have not been entirely resolved, hindering their widespread real-world application. Beyond these, Cadena *et al.* in [2] underscored further open problems in current SLAM research, ranging from fail-recovery self-tuning SLAM systems to optimal map representation. Within this thesis, our focus

narrows to **visual SLAM systems in dynamic indoor environments (dynamic SLAM)**, with an emphasis on the effects of short-term moving entities, where the prominent research problems can be categorized into three main aspects:

- **Accuracy Degradation**: Moving objects can directly induce noticeable drift and a decline in localization accuracy. Although detection modules effectively reduce the impact of moving objects in dynamic scenes, further investigations are required into the relationships between **detection accuracy and visual SLAM performance**. Additionally, this also propels us to examine **variations of localization accuracy** within identical dynamic/static environments.

- **Tracking Failure**: System robustness here primarily concerns the duration of valid tracking within a single run. Environmental perturbations often lead to tracking loss without estimation outputs. However, many studies allocate less attention to this tracking failure issue, leading to **inadequate evaluations of overall system performance**, especially in long-term operations.

- **Generalized Performance**: The adaptability of SLAM systems across diverse scenarios still demands extensive tuning efforts, particularly in challenging environments. Notably, while many dynamic SLAM approaches excel in dynamic scenes due to specialized strategies for moving objects, they often underperform in static scenarios. This limited generalization can be further constrained by a **dependency on underlying detection/segmentation modules**.

To address these, we will first benchmark various state-of-the-art visual (dynamic) SLAM approaches to draw preliminary insights into their overall performance. Furthermore, we aim to investigate how different detection and segmentation performances influence both the accuracy and robustness capabilities of such methods. By employing challenging sequences from the GRADE dataset [31], this benchmarking process is a fundamental step to understand the existing limitations and identify how we can improve these systems. Following this, we will propose an innovative dynamic SLAM approach designed to resolve these issues, characterized by improved tracking duration and generalized performance across diverse scenes. Ultimately, this work strives to provide significant contributions to the SLAM community through insightful suggestions and innovative solutions, addressing current limitations and pushing the boundaries of visual SLAM for dynamic indoor environments.

1.3 RESEARCH QUESTION

Along with exploring key challenges in dynamic SLAM, this thesis articulates a series of specific research questions to direct each segment of the study, formulated as follows:

- Are current testing and evaluation techniques adequate for advancing dynamic SLAM research, particularly regarding robustness assessment?

- What limitations persist in current dynamic SLAM systems, and how can we propose an approach to effectively overcome these issues?

- How do different visual (dynamic) SLAM algorithms perform when confronted with dynamic entities within the challenging GRADE sequences?
- What is the impact of detection and segmentation accuracy on dynamic SLAM capabilities? Specifically, does higher detector accuracy lead to improved visual SLAM performance in dynamic scenes?
- How does the localization accuracy vary between dynamic scenes and their corresponding static ones? That is, do these dynamic SLAM strategies maintain comparable performance without compromise in static environments?

1.4 CONTRIBUTION

The main contributions of this thesis can be summarized as follows:

- Toolbox development for the GRADE framework, including data post-processing, image set formulation, and visual SLAM implementations. Contribute to the construction and annotation of a real-world image set for instance segmentation, comprising 3579 images, 5362 human instances, and 130 background samples. All relevant code, data, and experimental results are open-source and available.
- Benchmarking various representative visual (dynamic) SLAM approaches on the challenging GRADE sequences, complemented by robustness assessments and analysis of failure cases. The customized experiments enable a detailed investigation into the effect of moving entities on visual SLAM systems.
- Investigating the relationship between detector accuracy and its effect on SLAM performance. This benchmarking process integrates a series of trained model weights into dynamic SLAM systems for testing on the TUM RGB-D sequences.
- Introducing a novel visual SLAM system for dynamic indoor environments based on per-pixel motion probabilities, achieving a better trade-off between localization accuracy and operational robustness. The proposed pixel-wise motion estimation module can identify moving regions without relying on semantic information and overcome the inaccuracies inherent in optical flow calculations.

1.5 THESIS OUTLINE

The remainder of this thesis is structured as: Chapter 2 provides a comprehensive review of the prevailing testing and evaluation techniques in visual SLAM research. Chapter 3 delves into the recent advancements in dynamic SLAM across various fields. In Chapter 4, we detail our SLAM benchmarking experiments and investigate the relationship between detector accuracy and dynamic SLAM performance, complemented by thorough analyses. Chapter 5 introduces our innovative dynamic SLAM system, DynaPix SLAM, and evaluates its performance through extensive experiments on both the GRADE and TUM RGB-D sequences. Lastly, Chapter 6 concludes this thesis, summarizing our principal findings and outlining directions for future work.

2
TESTING & EVALUATION

Beyond the core modules of SLAM frameworks, the testing and evaluation processes play a crucial role in verifying the effectiveness of a visual SLAM system. Generally, SLAM algorithms are tested through targeted simulations or by using off-the-shelf datasets, which can effectively replicate practical scenarios with less danger and financial burdens. The use of appropriate metrics ensures a fair assessment of SLAM performance. Within this chapter, our focus revolves around the argument "Are current testing and evaluation methods adequate for advancing visual SLAM research?".

2.1 SIMULATION

The robotics community has historically employed simulation software for a variety of tasks, including perception, navigation, and motion control. Indeed, they are all tasks that are dangerous to be performed directly in the real world due to the inherent danger of damaging objects, hurting people, and destroying equipment due to unexpected or unaccounted errors, noise, or malfunctions of the robotic platforms. Thus, they usually require verification in simulation prior to their deployment. Clearly, since SLAM comprises both navigation, perception and motion control, especially in its active (autonomous) modality [32], simulation is an essential building block to develop such a system. This is especially true within the context of dynamic SLAM research. For that, an ideal simulation, aimed at bridging the simulation-to-reality gap, should exhibit the following critical characteristics [31]: i) physical realism, ii) photorealism, iii) full controllability, and particularly, iv) the inclusion of dynamic entities. However, the majority of existing simulation frameworks fail to comprise all these aspects.

Gazebo [33], presently the most popular framework for robot simulations, is tightly integrated with the Robot Operating System (ROS) framework [34] for offline programming and fast prototyping. Similarly, simulators like CARLA [35], WeBots [36], CoppeliaSim [37], and RobotStudio [38] have also been widely adopted due to their comprehensive infrastructure and reliable physics engines, which support a wide range of sensors, actuators, and robot configurations. However, these platforms primarily emphasize the physical modeling and control of robots to achieve realistic motion and behavior, placing less attention on the fidelity of their working environments. As a result, the visual realism in these simulators is not entirely satisfactory, and constructing expected dynamic scenarios (e.g., indoor environments with naturally moving humans) often demands significant effort.

Thanks to recent advancements in graphics and embodied AI research, more simulators with impressive visual realism have emerged [39]. For instance, Gibson [40] and AI-Habitat [41] are constructed through full 3D scans, specializing in indoor visual nav-

igation. Meanwhile, iGibson [40] and AI2Thor [42] are distinguished for their highly interactive capabilities and extended physical representations of 3D assets. Nevertheless, these platforms either contain unrealistic artifacts stemming from the scanning process or display insufficient visual realism due to a simplified rendering process. On the other hand, simulators including AirSim [43], Sim4CV [44], and BenchBot [45], employing advanced rendering engines like Unreal Engine [1] and NVIDIA Omniverse [2], provide highly detailed and visually appealing simulation environments. However, all these simulators encounter common restrictions, particularly in the absence of dynamic assets and constrained flexibility in simulation customization (e.g., indoor placement, robot/sensor configuration, illumination/material settings).

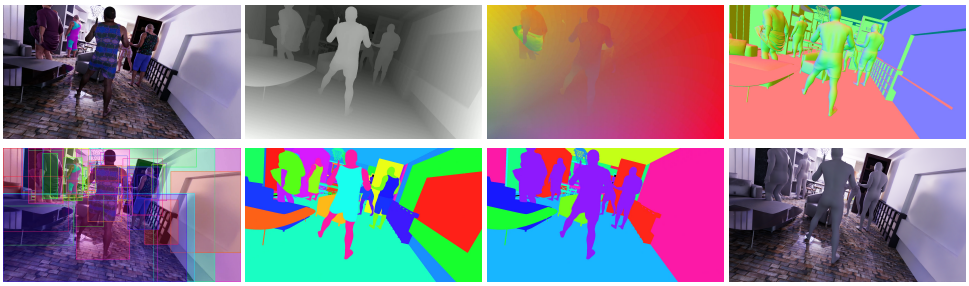


Figure 2.1: An example of generated data from GRADE [31], assets from Cloth3D humans [46] and one of the environments from 3D-Front [47]. Top row, left to right: Rendered RGB image, corresponding depth map, optical flow, and surface normals. Bottom row, left to right: 2D bounding boxes, semantic instances, semantic segmentation, and SMPL [48] shapes.

Recently, GRADE [31], developed on NVIDIA Issac Sim, steps further by incorporating dynamic assets (e.g., clothed 3D humans [49, 46, 48], flying objects [50, 51]) into the simulation, along with highly controllable customization and ROS integration. Despite imperfections in human-scene interactions and human appearance modeling, GRADE currently stands as a more suitable platform for dynamic SLAM research compared to other simulators. Apart from its unique realistic dynamic entities, GRADE also provides specialized challenging scenarios (i.e., with occlusions) to reveal the weakness of current visual SLAM approaches. In addition, GRADE contains identical scenes in dynamic environments and their replicated static environments, fulfilling our need to explore the performance variations in both scenarios.

2.2 DATASETS

Alternatively, existing datasets offer a cost-effective solution for verification, which ensures visual realism and includes dynamic entities by directly using real-world data or synthesized data from rendering engines. Several widely recognized datasets, such as TUM RGB-D [25], KITTI [26], and EuRoC [52], serve as public benchmarks for performance comparison, which has gradually become a fundamental basis of today's visual SLAM research.

[1] https://www.unrealengine.com/
[2] https://www.nvidia.com/en-us/omniverse/

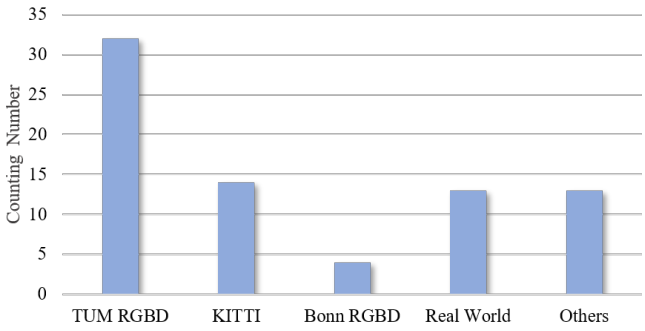


Figure 2.2: Statistics of Evaluated Datasets in Dynamic SLAM Research

However, the usage of existing SLAM datasets is quite biased, as shown in Figure 2.2. We conducted a survey of adopted datasets from 45 latest dynamic SLAM literature, which demonstrates that TUM RGB-D appears the most in the dynamic SLAM research. This may result in over-fitting since each dataset can easily fall into a similar pattern, thereby remarkable scores on a certain benchmark cannot ensure equal performance in generalized scenarios. Specifically, TUM RGB-D consists of several short-term indoor sequences but only covers limited views with slight movements. Therefore, good scores on certain benchmarks cannot ensure equal performance in the real world.

The reliance on datasets testing pose also other issues. First, prerecorded datasets have limitations in their adaptability and extension due to the differences in the robot configuration (e.g., placement of the sensors) and the sensor settings (e.g., sensor type, focal length). Second, they are an offline processing tool, in which it is not possible to test active methods that need to interact with the environment or make informative decisions, e.g. active SLAM. Third, they are difficult and expensive to collect because of, for example, the limitations of ground-truth recording systems such as VICON halls and the time taken to envision, carry out, and label the experiments. Finally, it is almost impossible to expand the already collected dataset by adding another LiDAR or modifying camera parameters. Synthesized datasets are also used for SLAM evaluation these days. Recent progress in random scene generation and photorealistic rendering makes it theoretically possible to synthesize scene changes for dynamic SLAM, but it would be difficult to model realistic changes as in natural lives. Moreover, the testing is still limited to offline methods.

2.3 EVALUATION

Given the testing platforms. evaluation metrics/criteria play important roles in quantifying and comparing the SLAM performance. The mainstream of adopted metrics includes absolute trajectory error (ATE), relative trajectory error, time-consuming (running time), CPU usage, and more [7]. We also conducted statistics on evaluation metrics adopted in the literature, as shown in Figure 2.3. ATE is the most frequent metric appearing almost in each dynamic SLAM paper, which demonstrates that the importance of localization accuracy of SLAM system is widely addressed in current research. RPE

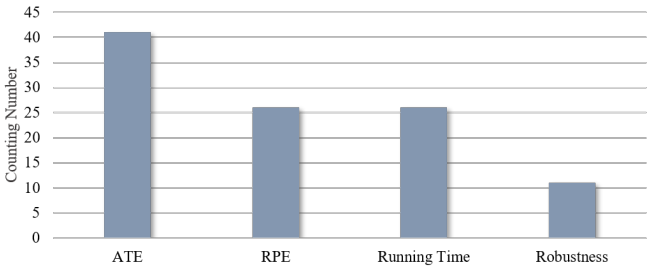


Figure 2.3: Statistics of Evaluation Metrics in Dynamic SLAM Research

and running time quite similarly appear in the literature, illustrating an aided function in verifying SLAM performance. Robustness evaluation only appears 10 times during this survey, which is less addressed, requiring more attention on the aspect of robust operation.

The absolute trajectory error (ATE) measures the localization accuracy by comparing the absolute distances between the estimated and ground truth trajectories. As both trajectories may lie in different coordinate frames, an external alignment is required by using rigid-body transformation $\mathbf{S} \in SE(3)$ to map estimated poses $\mathbf{P}_i \in SE(3)$ to the ground truth poses $\mathbf{Q}_i \in SE(3)$, then the ATE at time instant i could be computed as:

$$\mathbf{E}_i := \mathbf{Q}_i^{-1}\mathbf{S}\mathbf{P}_i \tag{2.1}$$

Similarly, the relative pose error (RPE) measures the local accuracy of the trajectory over a fixed time interval Δ, which can be formulated as:

$$\mathbf{F}_i^{\Delta} = \mathbf{Q}_i^{-1}\mathbf{Q}_{i+\Delta} \cdot \mathbf{P}_i^{-1}\mathbf{P}_{i+\Delta} \tag{2.2}$$

The majority of SLAM algorithms adopt the root mean square error (RMSE) of these metrics over the entire sequence for general evaluation, which can be defined respectively:

$$\begin{aligned} \mathrm{RMSE}(\mathbf{E}_{1:n}) &= \left(\frac{1}{n}\sum_{i=1}^{n} ||\mathrm{trans}(\mathbf{E}_i)||^2\right)^{1/2} \\ \mathrm{RMSE}(\mathbf{F}_{1:n}, \Delta) &= \left(\frac{1}{m}\sum_{i=1}^{m} ||\mathrm{trans}(\mathbf{F}_i)||^2\right)^{1/2} \end{aligned} \tag{2.3}$$

In addition, more recent studies [53, 54] address tracking rate (TR) for evaluating the system robustness. This metric, which is often overlooked, represents the ratio between the actually tracked time over the entire sequence duration.

3
DYNAMIC SLAM

In static environments, visual SLAM methods can achieve remarkable performance. However, moving objects severely affect core modules of such systems as state estimation and loop closure detection, limiting their deployment in general real-world scenarios. Since dynamic SLAM approaches are typically built on popular visual SLAM frameworks (e.g., ORB-SLAM2 [55]), we first introduce some underlying classic algorithms, and then follow with specific dynamic SLAM approaches.

3.1 VISUAL SLAM

Visual SLAM follows a similar formulation as the general SLAM process [56] as

$$\begin{cases} P(\mathbf{x}_k|\mathbf{x}_{k-1},\mathbf{u}_k) & \Leftrightarrow \quad \mathbf{x}_k = f(\mathbf{x}_{k-1},\mathbf{u}_k,\mathbf{w}_k), \quad k = 1,...,K \\ P(\mathbf{z}_{k,j}|\mathbf{x}_k,\mathbf{y}_j) & \Leftrightarrow \quad \mathbf{z}_{k,j} = h(\mathbf{y}_j,\mathbf{x}_k,\mathbf{v}_{k.j}), \quad (k,j) \in \mathcal{O} \end{cases} \tag{3.1}$$

where the first line conveys the motion model and the second line represents the observation model. At a time instant k, $\mathbf{x}_k$ is the state vector describing the location and orientation of robot/camera, $\mathbf{u}_k$ indicates the control input vector applied at time $k-1$, $\mathbf{y}_j$ is the vector describing the location of the ith landmark, and $\mathbf{z}_{k,j}$ represents the observation of the ith landmark from the location $\mathbf{x}_k$ at time k. $\mathbf{w}_k$ and $\mathbf{v}_{k,j}$ indicate the Gaussian motion disturbances and Gaussian observation errors, respectively.

Differently, in dynamic SLAM formulation, we have to assume that $\mathbf{y}_j$ can vary during each observation. To simplify the problem formulation without considering long-term variations, we assume that $\mathbf{y}_j$ is constant if it belongs to the facility structure in indoor environments (e.g., wall, floor, furniture), as opposed to humans and other movable objects (e.g., chairs, books, cups).

Most visual SLAM frameworks adopt the bundle adjustment (BA) process for camera pose optimization. Given matched feature points and initialized camera pose, the camera pose can be further optimized using bundle adjustment (BA), which aims to minimize the sum of reprojection errors for the matched feature points:

$$\{R^*, t^*\} = \arg\min_{R,t} \frac{1}{2} \sum_{i=1}^{n} \|x_i - \pi(RX_i + t)\|_2^2 \tag{3.2}$$

where R and t represent the camera orientation and position, respectively. X_i and x_i denote the 3D points in the world frame and their corresponding keypoints in the image frame. The function $\pi_{(\cdot)}$ represents the projection transformation from 3D camera coordinates to 2D plane coordinates.

Notably, well-known approaches like the ORB-SLAM series [55, 57], Semi-Direct Visual Odometry (SVO) [58], and VINS series [59, 60] have gained significant popularity in recent years and can achieve impressive accuracy (<10 cm) when assuming the observed environments are static during the SLAM process. ORB-SLAM2 [55] is a relatively robust *feature-based* method, which extracts and tracks ORB features from images to build the sparse map. Moreover, ORB-SLAM3 [57] proposed an improved framework with a visual-inertial and multi-map SLAM system. Compared with the traditional feature-based methods, *direct methods* (e.g., SVO [58], DSO [61]) track and triangulate pixels that are characterized by image gradients to estimate the camera motion, instead of relying on repeatable feature extractor and correct feature matching. Besides, VINS-Mono [59] is a typical *visual-inertial SLAM* framework, which proposed a tightly coupled and optimization-based visual-inertial odometry to achieve higher accuracy and robustness. Although the visual-inertial SLAM framework can overcome invalid visual tracking due to occlusion or dynamic objects, this framework typically disables its loop closure or mapping modules.

3.2 DYNAMIC SLAM

To specify why SLAM degrades in dynamic environments, we first briefly introduce the pose estimation mechanism. In terms of two-view geometry shown in Fig. 3.1, all observed landmarks (or map points) are taken into the calculation of camera transformation $\mathbf{T} \in \mathbf{SE}(3)$, where the presence of dynamic factors violates the epipolar constraint and adversely affect the estimation process [62]. To further illustrate this issue within the context of multi-view pose optimization, we introduce the pose graph framework depicted in Fig. 3.2, where the edges connecting camera poses constrain the motion model, and the edges linking camera poses and landmarks constrain the measurement model [63]. Notably, dynamic landmarks with associated false edges (denoted as red elements) can generate negative effects that even impact neighboring pose estimation.

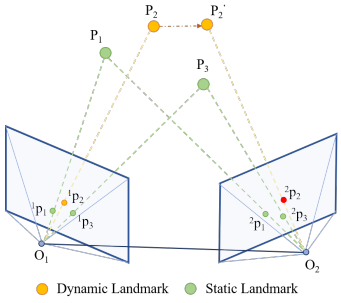
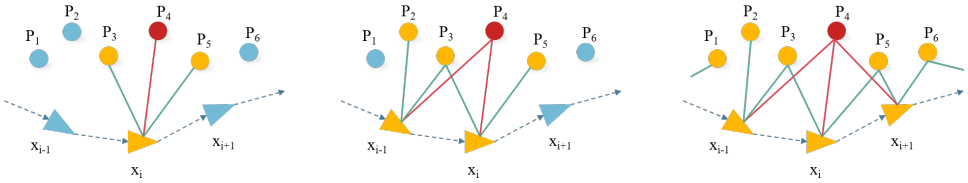


Figure 3.1: Two-view geometry for camera pose estimation with static and dynamic landmarks.

Therefore, built upon classic SLAM frameworks, dynamic SLAM algorithms typically integrate an external module to discard dynamic factors from the estimation process, thereby reducing the influence of dynamic factors and improving overall performance. These modules are typically based on advanced detection (e.g., YOLO [64]) or segmentation (e.g., Mask R-CNN [65] modules. However, these integrations also pose several

challenges including limited generalization and more frequent tracking failures.



(a) Pose Optimization (b) Local BA Optimization (c) Global BA Optimization

Figure 3.2: Example of pose graph applied in ORB-SLAM2 [55], where circles represent the landmarks (map points) and triangles denote the camera poses. The yellow highlights the elements that contribute to the optimization. while the red indicates the dynamic factor.

3.2.1. SEMANTIC-BASED DYNAMIC SLAM

Thanks to recent advancements in computer vision techniques, most SLAM algorithms are expanded with object detection or semantic segmentation modules to adapt to dynamic environments. The dynamic regions are then separately tracked [66, 67] or discarded as outliers [13, 27, 28, 68, 69, 70] to reduce their negative effects in pose estimation. Most of these methods assume that dynamic landmarks are highly correlated to their semantic attributes. For example, persons and vehicles are regarded as dynamic objects, while objects like chairs and tables are assumed as static. The features belonging to the dynamic objects will be discarded, and only the supposedly static features can be retrained for any given frame. DynaSLAM [13] combines Mask R-CNN [65] and multi-view geometry to process moving objects, while DS-SLAM [27] applies a lightweight SegNet [71] to obtain segmented masks. Similarly, Detect-SLAM [28] uses SSD [72] for object detection in keyframes, whereas YOLO-SLAM [73] employs YOLOv3 [64] as its underlying detection module to identify dynamic regions.

However, relying on an a-priori categorization heavily restricts the generalization of SLAM systems [74, 29] due to their dependency on both the quality of underlying networks and the pre-selection of movable object classes. Thus, these methods pay less attention to the actual motion states of the detected objects. While *actually* moving objects not belonging to those classes continue to degrade performance, the ones belonging to them are blindly masked. This easily causes few features retained and degradation in static scenes with frequent wrong detections, thus further degrading the system performance and causing possible failures [31]. For instance, the features belonging to parked cars and temporally static humans will be directly discarded from the estimation process.

3.2.2. GEOMETRY-BASED DYNAMIC SLAM

On the other hand, instead of extracting semantic information, DymSLAM [66] applies multi-motion fitting to segment different moving objects to estimate the motion of both camera and moving objects. Lu et al. [75] combine dense optical flow with depth information to generate scene flow masks to capture dynamic regions, while Flowfusion [76] uses optical flow residuals to highlight dynamic regions in RGB-D point clouds. Dy-

tanVO [29] proposes an iterative framework to jointly refine both ego-motion estimation and motion segmentation. While effectively overcoming the reliance on semantic information, these methods are heavily restricted by noise sensitivity in the presence of imperfect sensors or indoor and highly dynamic environments. This leads to possible misclassifications, especially when adopting thresholds to identify dynamic objects.

3.2.3. PROBABILITY-BASED DYNAMIC SLAM

Methods that integrate motion probability normally fuse the semantic attributes with the motion attributes. Detect-SLAM [28] introduces the propagation of motion probability derived from object detection, while DP-SLAM [68] and Cheng et al. [69] put forward dynamic region removal techniques within the Bayesian framework to enhance motion probability updates. All of these methods, however, eliminate the dynamic features with a threshold-based filter and do not retain motion probabilities.

More recently, a different strategy has been adopted to preserve more semi-static feature points by utilizing feature weights in the bundle adjustment (BA) process. WF-SLAM [77] employs tightly coupled semantic and geometric constraints to evaluate the weight of each feature point, while OVD-SLAM [78] combines YOLOv5 and sparse optical flow with a chi-square test to assign feature weights. Both methods incorporate these feature weights into the BA process to jointly optimize camera poses and feature weights, ensuring less contribution from dynamic features during each iteration.

4
BENCHMARK

Within this chapter, we aim to address the third and fourth research questions via benchmarking experiments. GRADE provides a comparably ideal platform to simulate dynamic indoor environments for visual SLAM operations. Thus, to delve deeper into the overall performance and potential limitations of current SLAM methods, we benchmark the representative approaches using challenging sequences from the GRADE, complemented by extended robustness assessments in Section 4.1.

Moreover, to investigate the relationship between detection accuracy and SLAM performance, we apply a series of trained YOLOv5 and Mask R-CNN models with varying accuracy levels. These models can be seamlessly integrated into the Dynamic-VINS and DynaSLAM systems. We then test all these configurations on the well-recognized TUM RGB-D dataset, as detailed in Section 4.2.

4.1 BENCHMARKING DYNAMIC SLAM

We have already introduced the advancements of GRADE [31] in Section 2.1. In order to make it more convenient for SLAM implementations and benchmarking, we apply preconfigured sequences from the GRADE dataset in the following sections. Notably, compared to other widely adopted benchmarking datasets, GRADE provides a wide range of specialized indoor scenarios. These cover most challenges posed by dynamic environments that could affect visual SLAM systems, including realistic humans with a variety of movements, deformable (non-rigid) clothing, flying items, and textureless areas.

Figure 4.1: Examples of applied GRADE sequences [31]. From left to right: D sequence with humans only, F sequence with flying objects, WO sequence with occlusions, and S sequences with no dynamic entities.

The GRADE dataset comprises 342 sequences, each with 1800 frames, amounting to 342 minutes of video. Each sequence lasts 60 seconds and provides RGB (30 fps), depth (30 fps) and IMU (240 Hz) data for visual (inertial) SLAM inputs. Among the available sequences, we select 8 specialized sequences (referred to as **GRADE sequences**) for our benchmarking process, as illustrated in Figure 4.1. In four of these sequences, the simulated quadrotor maintains a horizontal flight (H) without movements on the pitch and

roll axes. The remaining four sequences allow free movements along all axes. Static sequences (S) exclude any dynamic entities. The dynamic ones can be further categorized as: only with humans (D); with humans and random flying objects (F); and with temporal camera occlusions (WO). It is necessary to point out that dynamic entities might be static during observations. In other words, these dynamic entities can vary their motion states at different moments, which is consistent with real-world scenarios.

4.1.1. EXPERIMENTAL RESULTS

From the SLAM methods outlined in Chapter 3, we initially assess two popular visual SLAM methods, ORB-SLAM2 [55] and RTAB-Map [79], to demonstrate the usability of visual inputs based on their performance on static sequences. Tartan VO [30] is also selected to represent learning-based visual odometry systems. For dynamic SLAM methods, we incorporate StaticFusion [80], DynaSLAM [13], and Dynamic-VINS [74] (in both VO and VIO variations) into our benchmarking process. To ensure fairness, all methods are implemented without parameter tuning, except that the number of features in DynaSLAM and ORB-SLAM2 systems is increased to 3000. Our primary evaluation metrics are the absolute trajectory error (ATE) and the tracking rate (TR). The results within a 3.5 m depth range are reported in Table 4.1, while the ones within a 5.0 m depth range are presented in Table 4.2. Further detailed implementation and evaluation process can be found in our public repository [1].

Table 4.1: ATE RMSE [m] and Tracking Rate (TR) of the GRADE sequences in both their ground-truth and noisy versions. Each experiment is 60 seconds long and the depth is limited to 3.5 m.

Sequence		Dynamic-VINS (VIO)		Dynamic-VINS (VO)		Tartan VO		StaticFusion		DynaSLAM		ORB-SLAM2		RTAB-Map	
		ATE [m]	TR	ATE [m]	TR	ATE [m]	TR	ATE [m]	TR	ATE [m]	TR	ATE [m]	TR	ATE [m]	TR
Ground Truth	FH	0.069	0.99	0.201	0.99	0.551	1.00	0.085	1.00	0.241	0.94	0.149	1.00	0.126	1.00
	F	0.647	0.98	1.337	0.95	4.132	1.00	2.866	1.00	0.147	0.18	0.431	0.35	0.115	0.22
	DH	8.103	0.99	1.178	0.86	1.259	1.00	1.664	1.00	0.008	0.05	0.005	0.19	0.094	0.64
	D	0.188	0.99	1.304	0.99	1.264	1.00	1.212	1.00	0.057	0.89	0.459	1.00	0.492	0.88
	WOH	0.239	0.98	1.272	0.98	2.361	1.00	1.98	1.00	0.015	0.54	0.012	0.54	0.042	0.57
	WO	0.501	0.96	0.985	0.94	2.38	1.00	2.807	1.00	0.083	0.08	0.163	0.20	0.053	0.16
	SH	0.109	0.99	0.023	0.99	2.395	1.00	0.594	1.00	0.016	1.00	0.012	1.00	0.039	1.00
	S	0.205	0.99	0.039	0.99	1.205	1.00	7.919	1.00	0.01	1.00	0.011	1.00	0.043	1.00
Noisy	FH	0.155	0.98	0.367	0.99	0.582	1.00	0.854	1.00	0.309	0.98	0.386	1.00	0.097	0.97
	F	0.886	0.98	1.857	0.98	4.223	1.00	3.992	1.00	0.179	0.18	0.167	0.26	0.125	0.22
	DH	1.681	0.99	1.183	0.95	1.234	1.00	1.091	1.00	0.002	0.04	0.005	0.05	0.013	0.18
	D	0.707	0.99	1.598	0.97	1.356	1.00	2.278	1.00	0.043	0.55	0.7	0.82	0.405	0.52
	WOH	0.491	0.98	0.871	0.98	2.399	1.00	1.826	1.00	0.023	0.52	0.022	0.54	0.101	0.53
	WO	1.086	0.96	1.163	0.95	2.473	1.00	2.213	1.00	0.119	0.08	0.171	0.20	0.075	0.16
	SH	0.419	0.99	0.069	0.99	2.517	1.00	4.184	1.00	0.016	1.00	0.018	1.00	0.072	1.00
	S	0.177	0.99	0.137	0.99	1.306	1.00	3.538	1.00	0.029	1.00	0.026	1.00	0.133	1.00

As can be inferred from the benchmarking results, most methods perform admirably in static sequences, achieving remarkable localization accuracy (≤ 10 cm) as expected. This also demonstrates that visual inputs from the GRADE sequences are effective for SLAM testing. However, Tartan VO and StaticFusion are notable exceptions across all testing sequences. Since TartanVO is a learning-based system, a possible reason lies in that the synthesized GRADE data falls outside the distribution of its original training set. StaticFusion, on the other hand, exhibits significant drifts when faced with textureless areas and shows limited performance over a long period of operation. Despite its

[1] https://github.com/robot-perception-group/GRADE_tools/blob/main/SLAM_evaluation

Table 4.2: ATE RMSE [m] and Tracking Rate (TR) of the GRADE sequences in both their ground-truth and noisy versions. Each experiment is 60 seconds long and the depth is limited to 5.0 m.

Sequence		Dynamic-VINS (VIO)		Dynamic-VINS (VO)		Tartan VO		StaticFusion		DynaSLAM		ORB-SLAM2		RTAB-Map	
		ATE [m]	TR	ATE [m]	TR	ATE [m]	TR	ATE [m]	TR	ATE [m]	TR	ATE [m]	TR	ATE [m]	TR
Ground Truth	FH	0.073	0.99	0.259	0.99	0.551	1.00	0.059	1.00	0.221	1.00	0.199	1.00	0.229	0.87
	F	1.814	0.98	1.362	0.97	4.132	1.00	2.781	1.00	0.228	0.25	0.512	0.38	0.129	0.19
	DH	7.492	0.99	1.811	0.88	1.259	1.00	>10.0	1.00	0.008	0.09	0.009	0.17	0.108	0.19
	D	0.201	0.99	0.738	0.99	1.264	1.00	>10.0	1.00	0.038	0.93	0.275	0.99	0.154	0.61
	WOH	0.228	0.98	1.256	0.96	2.361	1.00	4.926	1.00	0.012	0.54	0.015	0.54	0.053	0.49
	WO	0.679	0.96	1.031	0.95	2.473	1.00	1.418	1.00	0.107	0.08	0.138	0.20	0.025	0.14
	SH	0.121	0.99	0.023	0.99	2.395	1.00	2.721	1.00	0.012	1.00	0.011	1.00	0.019	0.82
	S	0.226	0.99	0.035	0.99	1.205	1.00	>10.0	1.00	0.011	1.00	0.014	1.00	0.018	0.79
Noisy	FH	0.179	0.99	0.349	0.99	0.568	1.00	2.379	1.00	0.200	0.93	0.291	1.00	0.086	0.75
	F	0.904	0.97	2.315	0.96	4.192	1.00	2.661	1.00	0.189	0.24	0.129	0.28	0.125	0.16
	DH	1.749	0.99	2.047	0.94	1.214	1.00	>10.0	1.00	0.002	0.04	0.005	0.05	0.030	0.17
	D	0.611	0.99	1.616	0.98	1.350	1.00	>10.0	1.00	0.110	0.90	0.652	0.91	0.171	0.30
	WOH	0.561	0.98	1.550	0.99	2.389	1.00	2.691	1.00	0.024	0.54	0.022	0.54	0.047	0.45
	WO	0.962	0.96	1.429	0.96	2.399	1.00	1.724	1.00	0.112	0.08	0.142	0.20	0.041	0.12
	SH	0.404	0.99	0.063	0.99	2.537	1.00	5.602	1.00	0.017	1.00	0.017	1.00	0.061	0.71
	S	0.199	0.99	0.066	0.99	1.259	1.00	>10.0	1.00	0.029	1.00	0.027	1.00	0.220	0.81

semantic-free nature, this approach fails to correctly classify dynamic objects, resulting in noisy pose estimations, consistent with the findings in [81, 82].

For dynamic sequences, although DynaSLAM, ORB-SLAM2, and RTAB-Map show promising ATE results, the reduced tracking rates within these methods require more attention for improvements. Many of these methods frequently face tracking failures and cannot stably track the entire sequence, lacking efficient failure-recovery mechanisms. Conversely, Tartan VO and StaticFusion consistently track without failures (TR=100%), but at the expense of much larger localization errors. Dynamic-VINS addresses a better trade-off between localization accuracy and operation robustness. Notably, its VIO variation outperforms when considering both ATE and TR results, indicating the beneficial role of IMU sensors in the estimation process.

Moreover, performance tends to be more degraded in scenarios with flying objects (F/FH) and camera occlusions (WO/WOH). Firstly, these sequences contain unknown dynamic objects that YOLOv5 (in Dynamic-VIN) and Mask R-CNN (in DynaSLAM) cannot accurately detect without a priori information. This heavy reliance on the quality of underlying networks constrains the dynamic SLAM performance. Secondly, most current visual-based methods struggle to overcome camera occlusions, thereby easily falling into tracking failures.

In general, we can observe that the experiments on noisy sequences are marginally inferior to the ground truth ones, which aligns with our expectations. With increased depth range, the majority of the experiments maintain similar performance, grounding the reliability of these benchmarks. Additionally, while these methods deliver promising outcomes on standard datasets, they reveal several limitations from this benchmarking process, indicating that the challenges of dynamic SLAM remain unresolved.

4.2 BENCHMARKING DETECTION MODEL

As discussed in Section 3.2, the mainstream of dynamic SLAM approaches adopts semantic attributes to discard potential dynamic landmarks during estimation, typically through object detection (e.g., SSD [72], YOLOv5 [64]) or semantic segmentation (e.g., SegNet [71], Mask R-CNN [65]). Although we have already highlighted the generaliza-

tion restrictions of these modules, it is worth investigating their impact on visual SLAM performance. In other words, we intend to answer the question, '*Do more accurate detection/segmentation networks improve dynamic SLAM performance?*', with further benchmarking experiments.

4.2.1. EXPERIMENT SETUP

Within this section, we select two representative dynamic SLAM approaches, Dynamic-VINS (VO variation) and DynaSLAM, to explore their relations with corresponding YOLOv5 and Mask R-CNN, respectively. Given that GRADE sequences contain a variety of moving entities while TUM RGB-D primarily features dynamic humans, we believe that benchmarking TUM RGB-D walking sequences using single-class detectors of varying accuracy levels is more straightforward. Otherwise, investigating how specific class accuracies influence SLAM performance becomes more intricate.

Indeed, given a series of model weights, detector accuracy rankings may vary across different test sets. For instance, the average precision (AP) tested on the COCO [83] dataset cannot indicate similar performance on the TUM sequences, potentially leading to inaccurate inference. Therefore, to solidify the reliability of this benchmark, we have constructed a specific image dataset using all RGB images from TUM RGB-D walking sequences (referred to as the TUM image set). We then evaluated this image set to obtain corresponding AP metrics. As for the ground truth, we manually annotated all human masks for instance segmentation, comprising 3579 images, 5362 human instances, and 130 background samples, as illustrated in Fig. 4.2.

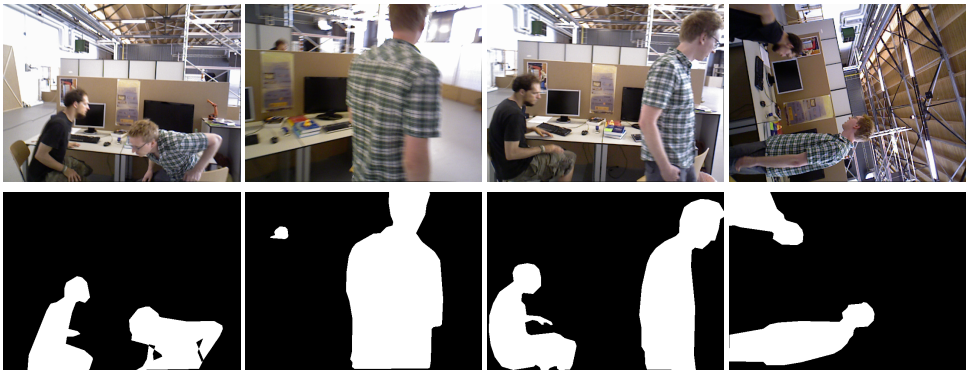


Figure 4.2: Examples of our TUM image set and corresponding human instance masks. Top row: Image data from TUM RGB-D walking sequences [25]. Bottom row: Ground truth instance masks annotated by ourselves.

Thus, for the remainder of this benchmarking, we prioritize training multiple single-class detectors that focus on human detection/segmentation. These series of model weights, with varying accuracy levels, are then integrated separately into dynamic SLAM systems for testing on the TUM RGB-D walking sequences. In this way, the evaluated AP results on the TUM image set directly correlate with their SLAM performance, enabling us to investigate the underlying relations.

4.2.2. DATA/MODEL FORMULATION

To acquire these model weights, the intuitive way is to train from scratch using various combinations of the GRADE (synthetic) and COCO (real-world) datasets. The inclusion of GRADE here can also demonstrate the visual realism of our synthetic data. To clarify first, the training process utilizes the COCO training sets and all generated data from GRADE (termed as GRADE image set), while the evaluation process involves the TUM image set and COCO validation sets.

To train YOLO and Mask R-CNN models, we use both a subset of the GRADE image set, which we will refer to as S-GRADE, and a larger one, A-GRADE. Images with a high probability of being occluded based on the depth and RGB information are automatically discarded. S-GRADE has 18K frames, of which 16.2K have humans in them and 1.8K are only background. S-GRADE comprises images only from indoor sequences without flying objects, with added motion blur on RGB images using a random rolling shutter noise ($\mu = 0.015$, $\sigma = 0.006$) and a fixed exposure time of 0.01 seconds. A-GRADE consists of all available data, with images containing flying objects and additional scenarios (e.g., outdoor city scenes). As for A-GRADE, we apply a random exposure time between 0 and 0.1 seconds for each sequence, with additional updates on the segmentation mask and bounding boxes to account for this motion blur.

As for COCO, we only utilize the subset of image data containing humans in the frame. Among these, we randomly sample 1256 training and 120 validation images, totaling ~ 2% and ~ 4% of the corresponding training/validation set (termed as S-COCO). Our BASELINEs are the models of networks pre-trained with the COCO dataset. We evaluate the performance with the COCO standard metric (mAP@[.5, .95], AP in this work) and the PASCAL VOC's metric (mAP@.5, AP50 in this work). More in-depth explanations of this training process can be found in [31, 84].

4.2.3. EXPERIMENTAL RESULTS

Table 4.3: YOLOv5 bounding box evaluation results. We put in **bold** the best result and in *italics* the second best.

Training Set	Pre-Training Set	COCO		TUM	
		AP50	AP	AP50	AP
BASELINE	—	0.753	0.492	0.916	0.722
S-COCO	—	0.492	0.242	0.661	0.365
S-GRADE	—	0.206	0.109	0.616	0.425
S-GRADE-E50	—	0.234	0.116	0.683	0.431
A-GRADE	—	0.176	0.093	0.637	0.459
A-GRADE-E50	—	0.282	0.154	0.798	0.613
S-COCO	S-GRADE	0.561	0.302	0.744	0.488
S-COCO	A-GRADE	0.540	0.299	0.762	0.514
COCO	S-GRADE	**0.801**	*0.544*	0.931	0.778
COCO	A-GRADE	*0.797*	0.542	0.932	**0.786**
S-GRADE + S-COCO	—	0.590	0.334	0.855	0.648
A-GRADE + S-COCO	—	0.527	0.289	0.801	0.597
S-GRADE + COCO	—	**0.801**	**0.547**	**0.938**	**0.786**
A-GRADE + COCO	—	0.764	0.503	*0.936*	*0.778*

Table 4.4: ATE RMSE [m] (upper line) and tracking rate (lower line) of Dynamic-VINS tested on the TUM RGB-D *fr3/walking* sequences using our trained YOLOv5 models. We put in **bold** the best result and in *italics* the second best for each sequence.

	S-COCO	S-GRADE	A-GRADE	T: S-GRADE F: S-COCO	T: S-GRADE F: COCO	T: A-GRADE F: S-COCO	T: A-GRADE F: COCO	Mix S-GRADE S-COCO	Mix S-GRADE COCO	Mix A-GRADE S-COCO	Mix A-GRADE COCO	BASELINE
w_half	0.064	**0.048**	0.061	0.059	0.066	0.081	0.060	0.064	*0.053*	0.059	0.062	0.069
	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
w_xyz	0.052	0.050	0.049	0.049	0.045	0.046	0.047	0.046	*0.043*	0.055	0.046	**0.037**
	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
w_rpy	0.133	0.224	0.137	0.116	0.116	0.120	0.119	0.149	0.126	0.132	*0.115*	**0.114**
	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
w_static	0.302	0.199	0.248	0.216	0.203	**0.182**	0.310	0.352	0.293	0.227	*0.183*	0.218
	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
Average	0.138	0.130	0.124	0.110	0.108	*0.107*	0.134	0.153	0.129	0.118	**0.102**	0.110
	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97

Table 4.5: Mask R-CNN instance segmentation evaluation results. We put in **bold** the best result and in *italics* the second best.

Training Set	Pre-Training Set	Threshold 0.7				Threshold 0.05			
		COCO		TUM		COCO		TUM	
		AP50	AP	AP50	AP	AP50	AP	AP50	AP
BASELINE	—	0.705	0.432	0.887	0.674	0.817	0.479	0.922	0.692
COCO	—	0.681	0.410	0.838	0.584	0.801	0.461	0.890	0.611
S-COCO	—	0.351	0.155	0.543	0.231	0.392	0.168	0.568	0.241
S-GRADE	—	0.100	0.043	0.509	0.264	0.117	0.048	0.561	0.283
A-GRADE	—	0.178	0.088	0.709	0.408	0.214	0.100	0.749	0.425
S-COCO	S-GRADE	0.401	0.195	0.665	0.374	0.465	0.216	0.694	0.387
S-COCO	A-GRADE	0.460	0.231	0.758	0.449	0.515	0.247	0.780	0.458
COCO	S-GRADE	*0.682*	*0.415*	*0.858*	*0.611*	*0.805*	*0.467*	*0.905*	*0.633*
COCO	A-GRADE	**0.710**	**0.430**	**0.869**	**0.638**	**0.813**	**0.476**	**0.908**	**0.660**
S-GRADE + S-COCO	—	0.268	0.126	0.608	0.321	0.344	0.149	0.661	0.346
A-GRADE + S-COCO	—	0.283	0.138	0.746	0.467	0.355	0.155	0.779	0.483
S-GRADE + COCO	—	0.671	0.401	0.849	0.603	0.790	0.452	0.896	0.626
A-GRADE + COCO	—	0.540	0.306	0.846	0.587	0.669	0.355	0.888	0.608

Table 4.6: ATE RMSE [m] (upper line) and tracking rate (lower line) of DynaSLAM tested on the TUM RGB-D *fr3/walking* sequences using our trained Mask R-CNN models. We put in **bold** the best result and in *italics* the second best for each sequence.

	S-COCO	S-GRADE	A-GRADE	T: S-GRADE F: S-COCO	T: S-GRADE F: COCO	T: A-GRADE F: S-COCO	T: A-GRADE F: COCO	Mix S-GRADE S-COCO	Mix S-GRADE COCO	Mix A-GRADE S-COCO	Mix A-GRADE COCO	BASELINE
w_half	0.031	0.034	0.032	0.030	**0.028**	0.030	*0.029*	0.031	*0.029*	0.031	0.030	0.030
	0.87	1.00	**1.00**	0.87	1.00	*1.00*	*1.00*	1.00	1.00	**1.00**	1.00	**1.00**
w_xyz	0.017	0.017	*0.016*	*0.016*	*0.016*	*0.016*	**0.015**	*0.016*	0.017	**0.015**	**0.015**	*0.016*
	0.99	0.94	0.91	**1.00**	0.91	0.91	0.91	0.94	0.91	0.91	0.91	0.92
w_rpy	0.034	0.104	**0.031**	0.060	0.037	*0.033*	0.034	0.039	0.038	0.043	0.035	0.040
	0.81	0.90	**0.96**	0.75	0.87	0.85	0.84	*0.92*	0.87	0.92	0.83	0.85
w_static	0.010	*0.007*	**0.006**	*0.007*	0.008	*0.007*	0.008	*0.007*	*0.007*	**0.006**	*0.007*	*0.007*
	1.00	1.00	**1.00**	**1.00**	0.84	0.97	0.84	**1.00**	0.84	*1.00*	0.97	0.98
Average	0.023	0.041	**0.021**	0.028	0.022	*0.022*	*0.022*	0.023	0.023	0.024	0.022	0.023
	0.92	0.96	**0.97**	0.91	0.90	0.93	0.90	*0.96*	0.90	0.96	0.93	0.94

As can be inferred from Table 4.3 and Table 4.5, the models trained with synthetic data have good generalization capabilities on real-world images. This process also indicates that using synthetic data alongside real-world images during training can significantly improve both detection and masking results.

Table 4.4 and Table 4.6 report the dynamic SLAM performance with different detectors. Each experiment is executed through three runs. Given these model weights with varying accuracy, we computed the baseline results again for both methods to be able to report their tracking rates. While for Dynamic-VINS there is no influence on the tracking rate, DynaSLAM is highly affected by the model used. For example, the model

pre-trained on S-GRADE and fine-tuned on S-COCO shows an increase in the tracking time of 8% on the *w_xyz* sequence. This holds true for most of the experiments in Table 4.6. Moreover, despite the lower segmentation performance, these configurations can present comparable ATE and TR as the baseline, or even outperform. The notable one is the model trained only with A-GRADE, exhibiting very compelling SLAM performance, but with an AP of 0.408 on the TUM image set. Conversely, the best detector with the highest AP does not necessarily correspond to a better visual SLAM performance. The remaining results indicate that there is a benefit to using a detection or segmentation network that adopts synthetic data during training. On average, the best tracking performance is obtained using the model trained on A-GRADE, while the model trained on the mixed A-GRADE+COCO dataset serves as the second-best model with general ATE results on both methods.

5
DynaPix SLAM

The goal of this chapter is to present our robust visual SLAM system for dynamic indoor environments, which consists of two novel core modules: a pixel-wise motion probability estimator (Section 5.1) and an enhanced pose optimization process (Section 5.2).To evaluate our method, we apply our DynaPix SLAM against ORB-SLAM2 and DynaSLAM on both TUM RGB-D and GRADE sequences, along with extended experiments on both static and dynamic versions of the GRADE ones (Section 5.3).

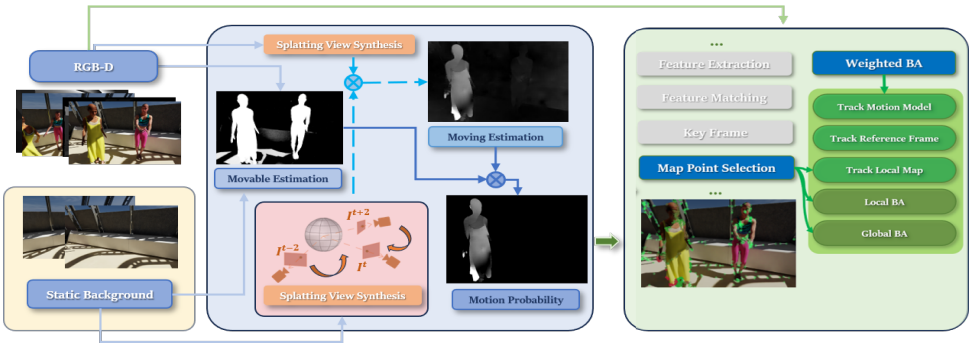


Figure 5.1: The DynaPix architecture consist of two main blocks, the motion probability estimation (blue box), and the modified ORB-SLAM2 (green box). We use RGB-D and corresponding background images to extract movable (Sec. 5.1.1) and moving regions (Sec. 5.1.2) on the current frame. The estimated moving probabilities are then integrated into all colored blocks of our SLAM backend (Sec. 5.2).

Given the preliminary discussions from Chapter 3 and Chapter 4, we can summarize the primary limitations on dynamic SLAM as follows:

- Over-reliance on semantic information and the quality of underlying detection networks. The moving objects not belonging to the predefined dynamic classes can continuously impact performance. This limitation also leads to incapabilities in detecting moving shadows, reflections, and the object's partial movements.
- Oversights in robustness assessment, especially regarding tracking failure issues. This can be attributed to the discarding of much useful visual information based on semantic attributes instead of focusing on the actual motion states.
- Dynamic SLAM typically excels under specific scenarios but may underperform when applied in different scenes, even in static environments. This limited generalization can be further restricted by dependencies on detection networks.

To solve these, our proposed visual SLAM approach, DynaPix SLAM, intends to employ motion attributes for a semantic-free pose estimation process. Our goal is to achieve reduced localization errors and higher tracking durations across various scenarios, including real-world TUM RGB-D dynamic sequences and both static and dynamic synthesized GRADE sequences.

The DynaPix SLAM, designed for dynamic indoor environments, mainly consists of two novel core modules: a pixel-wise motion probability estimator and an improved pose optimization process. Fig. 5.1 briefly demonstrates our DynaPix SLAM system. We first take RGB-D sequences and static background images as system inputs. The static background images represent images taken at the same time instant and camera states (e.g., position, velocity, intrinsic parameters), containing no dynamic entities and associated variations, such as shadows and illumination changes. These background images can be either synthetically generated [31] or inpainted through various state-of-the-art techniques [13, 85, 86]. Then, we decouple the pixel motion probability estimation process into two stages, using a combination of corresponding background images and adjacent frames to identify dynamic (moving) regions. These modules consider the motion of shadows or reflections in the environment and are capable of detecting specific moving parts of deformable objects, as shown in Fig. 5.1. With the usage of probabilistic motion information, useful image features are then retained throughout the estimation pipeline, rather than getting blindly discarded by the usage of binary masks. To incorporate these motion probabilities into the pose optimization process of the SLAM backend, a series of modifications are introduced in ORB-SLAM2. These include a map point selection process and a weighted-BA acting in both the *front end tracking* and *back end optimization* modules.

5.1 Pixel-wise Motion Probability

Before proceeding, we make a distinction between *movable* regions, i.e., potential dynamic regions or regions in the frame where motion *can* occur, and *moving* regions, i.e., regions which are actually moving or about to move at the current time instant. The goal of this module is to estimate moving regions in the current image frame using assigned probability values in place of binary masks.

As discussed in Sec. 3.2.2, learning-based flow estimation networks [87, 88] are commonly used to identify moving regions. However, the deployment in indoor dynamic environments poses specific challenges, including increased false correspondences in textureless areas (e.g. empty walls, floors), noisy estimation due to incomplete elimination of camera motion, and misclassification of foreground objects. To address these issues, our proposed probabilistic motion estimation is composed of two submodules: *movable region* (Sec. 5.1.1) and *moving region* estimation (Sec. 5.1.2). Movable regions are computed through background differencing and generate distributions covering potential moving objects with the corresponding shadows/reflections. Moving regions are then estimated through a rectified flow differencing mechanism which uses a novel combination of splatting view synthesis and static background with dynamic flow subtraction. The two are then fused (Sec. 5.1.3) to obtain the final pixel-wise motion probability. With this estimator, we manage to overcome general shortcomings of classic semantic-based detectors, such as imprecise detection/segmentation or limitation to predefined

categories, while successfully reducing the estimated errors due to the direct usage of optical flow methods.

5.1.1. MOVABLE REGION ESTIMATION

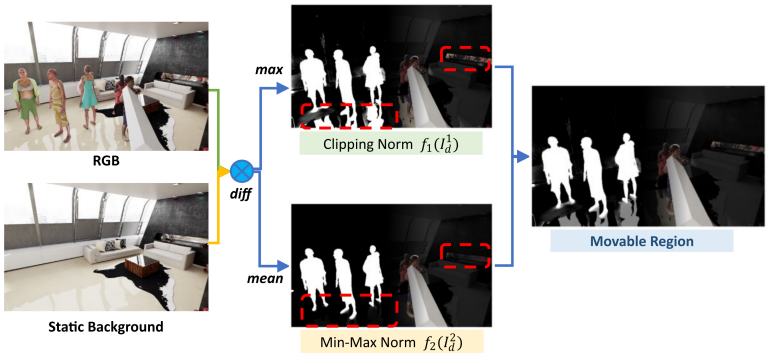


Figure 5.2: The process of probability estimation of movable regions. From the left: inputs, applied transformations, and results.

The *movable* distribution estimation serves as prior confidence to estimate motion attributes on the current frame. The static background images used in the differencing process can be either synthetically generated for simulated scenes by removing all potentially dynamic objects from the scene, or inpainted using E2FGVI [85] for real-world sequences. Making use of static background images, we observe that the difference between dynamic scenes and static scenes can provide reliable information about where the motion may occur, especially by capturing shadows and reflections that affect the environment. Indeed, the presence of such dynamic objects not only occludes the static background in the frame with the object itself, but brings variations to the surroundings by influencing the lighting conditions. Therefore, we subtract to the observed RGB frame (in dynamic scenes) I the corresponding static frame I_{bg}:

$$I_{diff}(x,y) = |I(x,y) - I_{bg}(x,y)| \in [0,255]^3 \tag{5.1}$$

$I_{diff}(x,y)$ is the absolute difference of RGB image I and static background image I_{bg} at location (x,y) over RGB color channels. We then apply $I_d^1 \in [0,255]^{H\times W}$ and $I_d^2 \in [0,255]^{H\times W}$ to represent I_{diff} as, respectively, the maximum and the average value of the pixel at that location:

$$\begin{aligned} I_d^1(x,y) &= \max(I_{diff}(x,y)) \\ I_d^2(x,y) &= \text{mean}(I_{diff}(x,y)) \end{aligned} \tag{5.2}$$

The movable probability for any pixel, p_m, is then computed with these two terms:

$$p_m = \lambda \cdot f_1(I_d^1) + (1-\lambda) \cdot f_2(I_d^2) \tag{5.3}$$

where f_1 and f_2 are scaling methods to project I_d^1 and I_d^2 to $[0,1]$. f_1 applies clipping normalization over the specific interval, in our case $[15,35]$. This is used to define an interval to reduce the effect of noise, change of illumination, and similar factors that can influence the RGB values of corresponding frames. f_2 adopts a min-max normalization within the current frame. λ is used to weight the two terms, and can be expressed as:

$$\lambda = \frac{1}{2} + \frac{1}{\exp(0.04 \cdot \max(I_d^2)) + 1} \in [0,1] \tag{5.4}$$

λ is necessary because any given RGB frame, I, may be relative to a scene without any movable objects. The image would then be highly similar to the corresponding static background image, I_{bg}. Applying then f_2 to the difference between the two would cause exploding factors due to the min-max normalization. The formulation can effectively reduce this effect by adjusting the participation of two terms.

With the combination of f_1 and f_2, the movable regions can be captured within a probabilistic distribution. In Fig. 5.2, we can observe that the first term f_1 is better at capturing unnoticeable variations including shadows and reflections, while f_2 can effectively reduce noise/error.

5.1.2. Moving Region Estimation

To determine the pixel-wise motion attributes, the problem can be formulated as the pixel displacement across the frames in 3D Euclidean space. This displacement can be further projected to the current 2D image frame for observations. The neighboring frames are reprojected to the current view to eliminate camera motion. Then, we adopt FlowFormer [89] as our underlying optical flow estimation module to provide correspondences for pixel-wise moving estimation. Ideally, when comparing the current frame with the reprojected frame, the static pixels should remain at the same coordinates, while the moving pixels show obvious displacements. This can be expressed as follows:

$$I^t(x,y) = \begin{cases} \text{static} & \text{if} \quad \text{dist} = \sqrt{||x-x'||^2 + ||y-y'||^2} \approx 0 \\ \text{moving} & \text{else} \end{cases} \tag{5.5}$$

where t denotes the frame timestamp, and (x', y'), represents the coordinates of the matched pixel in the reprojected frame. However, this expression requires further corrections when considering the incomplete elimination of camera motion and false pixel correspondences across the frames. Hence, we propose *splatting-based view synthesis* for accurate projection, and *static/dynamic flow differencing* for reliable moving region estimation.

Splatting-based View Synthesis. The goal is to synthesize image observations from other viewpoints to the corresponding location. The common approach adopts homography transforms to perform image reprojection [90, 91]. However, these transformations assume that the observed points belong to the same plane regardless of their depth information. This clearly causes displacements between the current frame I^t and the reprojected frame $\widetilde{I}^{t+i}$. To overcome this we follow the idea of softmax splatting [92] for more accurate view synthesis.

Given camera intrinsic matrix K, depth map at adjacent frame Z^{t+i}, we project the pixels of frame I^{t+i} into 3D space to recover their 3D information, then reproject these 3D points $\mathbf{X}^{t+i}$ to the current viewpoint with an initially estimated transformation $\{R, \mathbf{t}\}$:

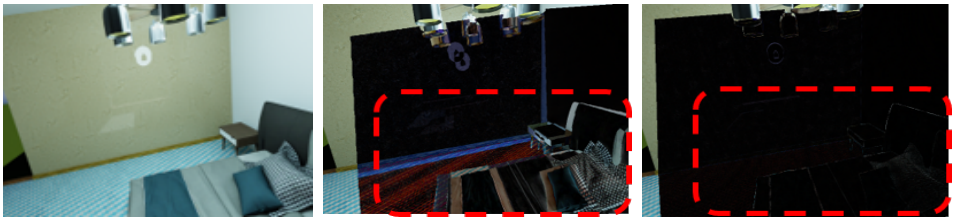
$$\begin{aligned} \mathbf{X}^{t+i} &= K^{-1}\mathbf{x}^{t+i}Z^{t+i} \\ \widetilde{\mathbf{x}}^{t+i} &= K(R\mathbf{X}^{t+i} + \mathbf{t}) \end{aligned} \tag{5.6}$$

where $\mathbf{x}$ denotes the 2D pixel coordinates, and $\widetilde{\mathbf{x}}^{t+i}$ represents the reprojected coordinate for each pixel at frame $\widetilde{I}^{t+i}$ when observing from the current viewpoint.

With these prior correspondences between the frame I^{t+i} and the desired transformed frame $\widetilde{I}^{t+i}$, each pixel in the frame $\widetilde{I}^{t+i}$ is synthesized by participation from adjacent pixels, which can be expressed as:

$$\widetilde{I}^{t+i}(x, y) = \frac{\vec{\Sigma}(\exp(z') \cdot I^{t+i}(x', y'))}{\vec{\Sigma}(\exp(z'))} \tag{5.7}$$

where $\vec{\Sigma}(\cdot)$ denotes the summation of all contributed pixels from the original frame I^{t+i}, $\exp(z')$ serves as the weight in this summation which relates to the depth of each pixel in frame I^{t+i}, and (x', y') represents the corresponding coordinates of contributed pixels in I^{t+i}. More details can be found in [92]. In Fig. 5.3, we show the advantage of using splatted views over homography transforms.



(a) Frame I^t (b) Densely reconstructed map (c) Semantically annotated map

Figure 5.3: Frame difference between a reprojected frame $\widetilde{I}^{t+i}$ and Frame I^t. While there is evident noise when subtracting homography-transformed images, using the splatted view synthesis achieves noise-free results.

Differencing Flow for Moving Estimation. Having obtained the reprojected view $\widetilde{I}^{t+i}$ from the splatting synthesis module, the static regions can be aligned correctly to the current view I^t. At this stage, the moving region estimation can be simply formulated as the flow estimation between these two frames by $\mathscr{F}(I^t, \widetilde{I}^{t+i})$, where the distance between each correspondence in Eq. 5.5 can be represented by flow magnitudes. This, despite displaying considerable effects in reducing errors, can not completely remove either the camera motion effects or false pixel correspondences that frequently occur in texture-less and ambiguous texture-rich areas during the flow estimation process. Interestingly, we observe a similar distribution of errors when performing flow estimation only on static background images. Therefore, we apply the flow estimation on static backgrounds as well, using the same procedure explained above, to further compensate

for the errors in the same regions through:

$$\mathscr{M}(I^t, \tilde{I}^{t+i}) = \min(\mathscr{F}(I^t, \tilde{I}^{t+i}), \mathscr{F}(I^t, \tilde{I}^{t+i}) - \mathscr{F}(I^t_{bg}, \tilde{I}^{t+i}_{bg})) \tag{5.8}$$

where $\mathscr{M}(I^t, \tilde{I}^{t+i})$ is the distribution of flow magnitudes over the frame. The low-pass filter is applied to avoid higher flow magnitudes due to subtraction on noisy estimations.

5.1.3. Final Motion Probability

Assuming that motion is consistent over a short period of time, the motion attribute of each pixel should be similar or little varying in the neighboring frames. Therefore, we finalize the moving region estimation at the current frame $\mathscr{M}^t$ across multiple frames:

$$\mathscr{M}^t = \frac{1}{2n} \sum_{j \in J} \left(\mathscr{M}(I^t, \tilde{I}^{t+j}) + \mathscr{M}(I^t, \tilde{I}^{t-j}) \right) \tag{5.9}$$

Where J is a set of time offsets and n is its cardinality. We adopt $J = [2]$ in our implementation to represent the moving region estimation at the frame t with respect to the next-previous and next-future frames.

Finally, the actual motion probability can be formulated as a blend of Eq. 5.9 and the movable probability p^t_m obtained in Eq. 5.3. For every frame t, we can compute

$$P^t = p^t_m \cdot \mathscr{M}^t \tag{5.10}$$

Through this multi-step processing involving splatted frame synthesis, flow differencing, crossed-frame calculation, and movable region constraints, the estimation errors arising from false correspondences and incomplete elimination of camera motion are progressively reduced. This results in the effective motion estimation covering moving parts of the objects, as well as their shadows, and reflections, as illustrated in Fig. 5.1.

5.2 Camera Pose Optimization

To incorporate the estimated motion probability into visual SLAM system, we introduce a series of modifications to the ORB-SLAM2 [55] framework. Before proceeding, we first briefly introduce ORB-SLAM2. The tracking module consists of the first-stage coarse estimation (i.e. *track motion model*, *track reference frame*) followed by a more precise second-stage estimation as *track local map*, while the backend module contains *local BA* and *global BA* for the optimization of camera poses and map point locations. Based on this, we improve the map point selection process (Section 5.2.1) and weighted bundle adjustment (Section 5.2.2). The latter directly affects both the tracking and backend optimization modules. Different from the previous work [93, 94, 13, 27], in which temporarily stationary movable objects are removed, our insights lie in that stationary objects (or stationary parts) can also be fully utilized to improve overall performance, whereas still preventing their negative influence once they resume the state of motion.

5.2.1. Map Point Selection

In ORB-SLAM2, map points correspond to features belonging to any identified keyframe and determine the accuracy of second-stage estimation and backend optimization. Here,

we only allow reliable features as map points. This is because features belonging to dynamic objects will change their positions in 3D space during the experiment, introducing wrongful estimation information in the stored map. To identify those, we modify the selection strategy of map points of the framework.

Given an image, all its features can be indicated as $K = \{k_1, \ldots, k_n\}$, where each k_i represents the coordinates of the feature in the frame. If this image was to be identified as keyframe, and its features updated as map points we can obtain their motion state, i.e. the estimated motion probability, with Eq. 5.10. Only the features satisfying $P(k_i) \leq p_{add}$ can be then further selected as map points. In our case $p_{add} = 0.05$ to ensure that static map points participate in *track local map* module and *local/global BA*. On the other hand, the stored map points from earlier keyframes may transition to moving states at the current frame, impacting the estimation process. Relying on the existing descriptor-based matching mechanism in ORB-SLAM2, the correspondences are established between partially observed map points from earlier keyframes and all image features in the current frame. We then conduct the map point deletion if $P(k_i) \geq p_{del}$ using again Eq. 5.10 as feature motion probability and $p_{del} = 0.1$. While our motion probability estimation is threshold-free, the inclusion of these limits here is necessary to improve overall estimation accuracy. However, note that the map point motion probability is anyway retained in the following steps, including the bundle adjustment procedures.

5.2.2. WEIGHTED BUNDLE ADJUSTMENT

It is crucial to retain numerous features to prevent tracking failures, even if those belong to *potentially* but *not moving* or slightly moving objects. We first aim to conduct a coarse estimation by using all the features to improve robustness. However, this severely affects the estimation accuracy due to the presence of dynamic features and may lead to false estimated poses, impacting the subsequent optimization procedures. For this reason, inspired by previous works [95, 77, 78], we assign weight to each error term during the BA process. This is formulated as:

$$\{R^*, t^*\} = \arg\min_{R,t} \frac{1}{2} \sum_{i=1}^{n} w_i ||\mathbf{x}_i - \pi(R\mathbf{X}_i + t)||_2^2 \tag{5.11}$$

where R and t represent the camera orientation and translation. $\mathbf{X}_i \in \mathbb{R}^3$ denote a point location in the world frame. These are temporary 3D points projected either from features or map points [55]. $\mathbf{x}_i \in \mathbb{R}^2$ represent its matched coordinates in the current image frame. The function $\pi_{(\cdot)}$ is the projection from 3D camera coordinates to 2D plane coordinates. Since the BA works on either the map points or the features of the previous/current frames, we can obtain the weight for each one of these points k_i with Eq. 5.10 by defining $w_i = 1 - P(k_i)$. Clearly, if $P(k_i)$ is low, the static point has a high probability of being static and receives a higher weight, contributing more to the optimization process. At the same time, points falling in dynamic areas are still retained but have less effect on the pose optimization. This procedure impacts both *track motion model* and *track reference frame* procedures of ORB-SLAM2 using all the features extracted from the image, as well as *track local map* and *local/global BA* using filtered map points. Moreover, different from the previous work, we implement this weighting procedure into both the tracking module and the backend optimization module. We

also observe that in those BA methods, the weight of each feature decreases during subsequent optimization iterations based on Gauss-Newton or Levenburg-Marquardt algorithms, effectively diminishing the difference between static and dynamic features. As opposed to that, we keep the last obtained weighting factor w_i in our BA procedure to fully retain this information.

Figure 5.4: Example of inpainting on the TUM-RGBD dataset.

5.3 Experimental Results

Table 5.1: ATE RMSE [m] and Tracking Rate (TR) of the GRADE sequences in both dynamic and static scenarios. Each experiment is executed through 10 runs.

		STATIC SEQUENCE								DYNAMIC SEQUENCE							
		DynaPix		ORB-SLAM2		DynaPix-D		DynaSLAM		DynaPix		ORB-SLAM2		DynaPix-D		DynaSLAM	
		ATE[m]	TR	ATE[m]	TR	ATE[m]	TR	ATE[m]	TR	ATE[m]	TR	ATE[m]	TR	ATE[m]	TR	ATE[m]	TR
FH	mean	0.006	1.00	0.010	1.00	0.007	1.00	0.012	1.00	0.035	1.00	0.248	1.00	0.023	1.00	0.232	0.98
	std	0.000	0.00	0.002	0.00	0.000	0.00	0.005	0.00	0.016	0.00	0.103	0.00	0.006	0.00	0.041	0.02
F	mean	0.230	0.86	0.330	0.90	0.325	0.88	0.529	0.82	0.291	0.20	0.359	0.35	0.571	0.37	0.864	0.42
	std	0.426	0.00	0.507	0.01	0.498	0.01	0.526	0.04	0.108	0.01	0.156	0.03	0.265	0.20	0.217	0.07
DH	mean	0.005	0.18	0.005	0.18	0.004	0.18	0.013	0.07	0.006	0.17	0.005	0.18	0.010	0.14	0.011	0.10
	std	0.001	0.00	0.001	0.01	0.001	0.00	0.009	0.03	0.006	0.00	0.001	0.01	0.011	0.04	0.003	0.02
D	mean	0.023	0.98	0.018	0.97	0.019	0.98	0.024	0.98	0.032	0.99	0.317	0.99	0.041	0.96	0.050	0.89
	std	0.006	0.01	0.002	0.03	0.004	0.02	0.008	0.02	0.010	0.02	0.043	0.01	0.026	0.03	0.009	0.05
WOH	mean	0.012	0.54	0.013	0.54	0.011	0.54	0.015	0.54	0.009	0.54	0.016	0.54	0.010	0.54	0.012	0.54
	std	0.008	0.00	0.010	0.00	0.013	0.00	0.017	0.00	0.001	0.00	0.008	0.00	0.004	0.00	0.002	0.00
WO	mean	0.040	0.44	0.038	0.83	0.206	0.87	0.043	0.78	0.023	0.20	0.168	0.20	0.641	0.20	0.083	0.08
	std	0.023	0.35	0.021	0.32	0.399	0.21	0.022	0.28	0.002	0.00	0.022	0.00	0.386	0.00	0.010	0.00
SH	mean	0.010	1.00	0.012	1.00	0.012	1.00	0.010	1.00	-	-	-	-	-	-	-	-
	std	0.001	0.00	0.002	0.00	0.003	0.00	0.001	0.00	-	-	-	-	-	-	-	-
S	mean	0.010	1.00	0.011	1.00	0.009	1.00	0.010	1.00	-	-	-	-	-	-	-	-
	std	0.001	0.00	0.001	0.00	0.001	0.00	0.002	0.00	-	-	-	-	-	-	-	-
Average		0.042	0.75	0.055	0.80	0.074	0.81	0.082	0.78	0.066	0.52	0.185	0.54	0.216	0.54	0.209	0.50

We use sequences from both the TUM-RGBD [25] and the GRADE [31] in our evaluations. Additional processing is necessary on the TUM-RGBD data to obtain the static background images. For that we use E2FGVI [85] video inpainting. We adjust the input

Table 5.2: ATE RMSE [m] and Tracking Rate (TR) of TUM RGB-D *fr3/walking* sequences. Each experiment is executed through 10 runs.

		DynaPix		ORB-SLAM2		DynaPix-D		DynaSLAM	
		ATE [m]	TR	ATE [m]	TR	ATE [m]	TR	ATE [m]	TR
/w_half	mean	0.030	1.00	0.607	0.79	0.023	1.00	0.029	1.00
	std	0.002	0.00	0.180	0.11	0.001	0.00	0.001	0.00
/w_static	mean	0.012	1.00	0.355	1.00	0.007	1.00	0.007	0.98
	std	0.002	0.00	0.121	0.00	0.001	0.00	0.000	0.00
/w_rpy	mean	0.043	1.00	0.744	0.99	0.123	0.98	0.040	0.86
	std	0.007	0.00	0.115	0.01	0.107	0.03	0.008	0.032
/w_xyz	mean	0.018	1.00	0.732	0.84	0.014	1.00	0.016	0.92
	std	0.002	0.00	0.102	0.11	0.000	0.00	0.001	0.00
Average		0.026	1.00	0.610	0.91	0.041	0.99	0.023	0.94

frames strategy of E2FGVI with a 50-frames sliding window approach with 100 frames bootstrap to overcome the high GPU usage of the method. We then extract the reference frames based on the closest covisible ones using ground-truth poses, rather than the original selection based on time intervals. An example of the inpainted frames is provided in Fig. 5.4. The GRADE's framework instead provides a way to re-render images captured in dynamic scenes without the dynamic objects through their experiment repetition tool. By doing so, we obtained a frame-by-frame correspondence of background images. In our experiments, for both the inpainting and the splatting view synthesis, we use the ground truth pose of the camera for the necessary transformations. While this is a limiting factor of the current proposed approach, note that a neighboring *frame-by-frame* pose variation can be estimated or optimized through different VO, VIO, DNN or other modules with good approximations.

For our tests, we apply DynaPix SLAM to compare the results against ORB-SLAM2, its underlying SLAM framework. Moreover, we also introduce DynaPix-D, which combines DynaPix with DynaSLAM [13], a popular state-of-the-art dynamic SLAM approach. While DynaSLAM combines Mask-RCNN to mask pre-defined dynamic classes and ORB-SLAM2 as its backend, DynaPix-D modifies the map point selection based on those same mask filtering. However, we maintain the same strategies on weighted BA in both tracking and backend modules as DynaPix SLAM. We utilize the widely accepted RMSE of absolute trajectory error (ATE) to evaluate pose estimation accuracy, and tracking rate (TR) to evaluate the system robustness.

Our results on the GRADE dataset are reported in Tab. 5.1, while the ones for the TUM-RGBD sequences are in Tab. 5.2. For each combination of sequence and method we perform ten experiment runs and report the mean and standard deviation on both metrics, as well as the overall averages. We use the static sequences of GRADE to show that our DynaPix does not degrade the performance of the used SLAM framework in those situations and to further compare the improvements between static and dynamic scenes.

Starting from the synthetic data, we can see that, in general, the experiments performed better on the static version of each sequence with respect to the corresponding dynamic ones. This was expected and holds for all methods. We can already see how using the TR alongside the ATE is essential to analyze these results. Taking the F

experiment in Tab. 5.1 as an example we can see that, while the ATE is similar to the ORB-SLAM2, the TR is 90% for the static sequence but only 35% on the dynamic one. We can also notice that DynaPix does not adversely affect the performance of the original ORB-SLAM2 when performing SLAM on static experiments, with the exception of the WO experiments in which the TR of DynaPix is almost half of the one with ORB-SLAM2. However, the standard variations on these experiments indicate how unsure these two particular results are in both ATE and TR metrics. Considering DynaPix-D and DynaSLAM results on the static sequences, we observe that our approach has better results on both TR and ATE in various experiments like F, DH, D, WO, as well as 10% on the average ATE and ~ 4% on average TR.

Considering now the experiments run on data with dynamic entities, DynaPix and DynaPix-D consistently overcome the corresponding methods with considerable margins. The TR of WO with DynaPix-D, for example, is 2.5 times better than DynaSLAM, although with a far worse ATE linked to the longer tracking time. The ATE of the same sequence obtained by DynaPix is only ~ 13% than the original one obtained through ORB-SLAM2, but with the same TR. An exception to that is the F sequence on both DynaPix and DynaPix-D, which have respectively 15% and 5% shorter tracking times. This experiment has the camera facing a featureless wall, which can make the SLAM backend stop working on some runs since no recovery procedure is devised for such situations. Despite this, DynaPix performs 3 times better than ORB-SLAM2 in ATE with similar tracking rates despite the 15% drop on the F experiment. On the other hand, DynaPix-D shows a 10% relative TR improvement with respect to DynaSLAM with a marginally worse ATE of 2 cm, despite the significantly worse ATE on the WO sequence of about 60 cm. It is interesting to notice how the dynamics of the WOH sequence seem not to affect the results, given that the metrics are close on both the static and dynamic tests. This is probably due to the camera facing a featureless area during the experiment, while ORB-SLAM2 does not provide robust recovery procedures for such situations. Overall, we can see that, on average, DynaPix performs better than DynaPix-D and ORB-SLAM on both static and dynamic sequences, making it the best overall method. This, holds especially if considering that longer tracking times can be linked to higher ATE.

Having verified that our method performs better with respect to both ORB-SLAM2 and DynaSLAM on the GRADE data, we proceed to analyze the results of our experiments with real data, which are reported in Tab. 5.1. With a 23 times improvement on the ATE of DynaPix and 100% TR of DynaPix, we perform better than any compared approach. DynaPix[-D] methods bring a consistent TR improvement alongside better ATE. The only variation to that is in the w_rpy experiment, in which the ATE performs on average worse but with a wide standard deviation, indicating that the method is not stable to fast rotations of the camera as desired. This, however, is linked to a 12% higher TR, which can clearly adversely affect the ATE metric. Thus, we can conclude that, while on synthetic dynamic sequences, we are still limited by occlusions and textureless areas that cause drift and tracking loss, on real-world ones our method has the best performance.

6 CONCLUSION

6.1 SUMMARY

Although SLAM has undergone significant development and has been considered a mature research field, several open problems in current visual SLAM algorithms still hinder their widespread applications. Among these, dynamic environments pose great challenges through short-term moving entities (as *dynamic SLAM*) and variations in longer time scales (as *lifelong SLAM*). Within this thesis, we mainly focused on investigating the effect of moving objects on the overall performance of visual SLAM systems, with an emphasis on their accuracy, robustness, and generalizability.

Notably, the necessity of testing processes for visual SLAM is often overlooked. In our view, visual SLAM systems universally lack the ability to generalize to out-of-distribution scenarios. This problem is compounded by the insufficiencies of current simulators and datasets, which fail to replicate more diverse and realistic dynamic environments as expected. Despite GRADE serving as one of the most suitable platforms to construct desired simulations, more endeavors are expected to enrich this field and facilitate the development of dynamic SLAM. On the other hand, robustness capabilities are less assessed in many studies, resulting in many systems with appealing accuracy results but inconsistent tracking operations.

To ground our insights, we benchmarked various SLAM methods and found that in static sequences, most demonstrate high localization accuracy, aligning with expectations and validating the effectiveness of GRADE for SLAM testing. In dynamic sequences, while some methods show promising localization accuracy, tracking failures are common. Challenges persist with dynamic objects and occlusions, as these methods depend heavily on the quality of object detection networks, struggle with camera occlusions, and are incapable of detecting unknown objects and partial movements.

Furthermore, the investigation on detector accuracy relations also indicates that the more accurate detectors do not necessarily lead to improved dynamic SLAM performance. Current methods tend to remove all potential dynamic objects during the estimation process, addressing less on assessing their actual motion states. Nevertheless, over-deletion directly leads to frequent tracking failures, especially in crowded dynamic environments. We believe that more static landmarks should be expected during the operation, and an external motion estimation can ensure a more stable tracking process.

Finally, we proposed DynaPix SLAM, a novel dynamic SLAM method based on pixel-wise motion probability and an improved pose optimization process. We first introduced a two-staged method to compute per-pixel motion probabilities by blending *movable* and *moving* estimations. These estimations are obtained through a static/dynamic differencing on both image frames and optical flows, respectively. The motion prob-

abilities are then applied in the ORB-SLAM2 framework with a refined mechanism to select map points while retaining these probabilities during the tracking and backend optimization procedures. Our extensive experiments on both real-world and synthetic scenes show that DynaPix SLAM consistently outperforms the ORB-SLAM2 and DynaSLAM methods. Moreover, given the results on the static version of GRADE sequences, we can also infer that DynaPix SLAM can exhibit more generalized capabilities across both static and dynamic scenes, without exhibiting degradation in static environments.

6.2 FUTURE WORK

To address the current limitations of this approach, future works can be summarised as include:

- Camera Pose Utilization: Future iterations will explore the integration of estimated camera poses into the motion probability estimation process. Due to the difficulty in accessing the ground truth information, the involvement of estimated results can verify the usability of our approach from a practical view.
- Recovery Mechanisms for Textureless Areas: Recognizing the challenges posed by textureless surfaces to tracking reliability, research will focus on developing robust recovery procedures to avoid long-term lost states. This may involve the inclusion of structure recognition instead of only relying on point descriptions.
- Semantic Information Integration: The next phase will also experiment with other elegant integrations of semantic information into DynaPix SLAM, aiming to further enhance the system's understanding and categorization of dynamic objects.
- Module Validation through Ablation Studies: To quantify the impact of each component within the SLAM framework, comprehensive ablation studies will be essential. To evaluate the necessity and efficiency of each module, these experiments will be performed on the motion estimator and weighted BA separately.
- Learning-Based Motion Estimation: Advancing the pixel-wise motion estimator from an empirical design to a learning-based model is anticipated. This approach will potentially improve the adaptability of the system to various dynamic conditions without heavy reliance on pre-defined heuristics.

These elements will guide the ongoing efforts to refine visual SLAM technologies, ensuring that they remain at the forefront of innovation in robotic perception and autonomous navigation.

BIBLIOGRAPHY

[1] Randall C. Smith and Peter Cheeseman. "On the Representation and Estimation of Spatial Uncertainty". In: *The International Journal of Robotics Research* 5.4 (1986), pp. 56–68. DOI: 10.1177/027836498600500404.

[2] Cesar Cadena et al. "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age". In: *IEEE Transactions on Robotics* 32.6 (2016), pp. 1309–1332. DOI: 10.1109/TRO.2016.2624754.

[3] Xuesong Shi et al. "Are we ready for service robots? the openloris-scene datasets for lifelong slam". In: *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2020, pp. 3139–3145.

[4] Guillaume Bresson et al. "Simultaneous localization and mapping: A survey of current trends in autonomous driving". In: *IEEE Transactions on Intelligent Vehicles* 2.3 (2017), pp. 194–220.

[5] Guray Sonugur. "A Review of quadrotor UAV: Control and SLAM methodologies ranging from conventional to innovative approaches". In: *Robotics and Autonomous Systems* (2022), p. 104342.

[6] Li Jinyu et al. "Survey and evaluation of monocular visual-inertial SLAM algorithms for augmented reality". In: *Virtual Reality & Intelligent Hardware* 1.4 (2019), pp. 386–410.

[7] Yuanzhi Liu et al. *Simultaneous Localization and Mapping Related Datasets: A Comprehensive Survey*. 2021. arXiv: 2102.04036 [cs.RO].

[8] Khalid Yousif, Alireza Bab-Hadiashar, and Reza Hoseinnezhad. "An overview to visual odometry and visual SLAM: Applications to mobile robotics". In: *Intelligent Industrial Systems* 1.4 (2015), pp. 289–311.

[9] Andrew J. Davison et al. "MonoSLAM: Real-Time Single Camera SLAM". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.6 (2007), pp. 1052–1067. DOI: 10.1109/TPAMI.2007.1049.

[10] C. Kerl, J. Sturm, and D. Cremers. "Dense Visual SLAM for RGB-D Cameras". In: *Proc. of the Int. Conf. on Intelligent Robot Systems (IROS)*. 2013.

[11] R. Wang, M. Schwörer, and D. Cremers. "Stereo DSO: Large-Scale Direct Sparse Visual Odometry with Stereo Cameras". In: *International Conference on Computer Vision (ICCV)*. Venice, Italy, Oct. 2017.

[12] Lachlan Nicholson, Michael Milford, and Niko Sunderhauf. "QuadricSLAM: Dual Quadrics from Object Detections as Landmarks in Object-oriented SLAM". In: *IEEE Robotics and Automation Letters* PP (Aug. 2018), pp. 1–1. DOI: 10.1109/LRA.2018.2866205.

[13] Berta Bescos et al. “DynaSLAM: Tracking, Mapping, and Inpainting in Dynamic Scenes”. In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 4076–4083. DOI: 10.1109/LRA.2018.2860039.

[14] Antoni Rosinol et al. “Kimera: an Open-Source Library for Real-Time Metric-Semantic Localization and Mapping”. In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2020. URL: https://github.com/MIT-SPARK/Kimera.

[15] Wei Xu and Fu Zhang. “FAST-LIO: A Fast, Robust LiDAR-Inertial Odometry Package by Tightly-Coupled Iterated Kalman Filter”. In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 3317–3324. DOI: 10.1109/LRA.2021.3064227.

[16] Thomas Whelan et al. “ElasticFusion: Real-time dense SLAM and light source estimation”. In: *The International Journal of Robotics Research* 35.14 (2016), pp. 1697–1716. DOI: 10.1177/0278364916669237. eprint: https://doi.org/10.1177/0278364916669237. URL: https://doi.org/10.1177/0278364916669237.

[17] Feng Lu and Evangelos Milios. “Globally consistent range scan alignment for environment mapping”. In: *Autonomous robots* 4 (1997), pp. 333–349.

[18] Arshiya Mahmoudi, Mahdi Mortazavi, and Mehdi Sabzehparvar. “Accommodating the multi-state constraint Kalman filter for visual-inertial navigation in a moving and stationary flight”. In: *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering* 236.7 (2022), pp. 1295–1303. DOI: 10.1177/09544100211029742. eprint: https://doi.org/10.1177/09544100211029742. URL: https://doi.org/10.1177/09544100211029742.

[19] Christian Forster et al. “On-Manifold Preintegration for Real-Time Visual–Inertial Odometry”. In: *Trans. Rob.* 33.1 (Feb. 2017), pp. 1–21. ISSN: 1552-3098. DOI: 10.1109/TRO.2016.2597321. URL: https://doi.org/10.1109/TRO.2016.2597321.

[20] Bill Triggs et al. “Bundle adjustment—a modern synthesis”. In: *Vision Algorithms: Theory and Practice: International Workshop on Vision Algorithms Corfu, Greece, September 21–22, 1999 Proceedings*. Springer. 2000, pp. 298–372.

[21] Luca Carlone et al. “Initialization techniques for 3D SLAM: A survey on rotation estimation and its use in pose graph optimization”. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 4597–4604. DOI: 10.1109/ICRA.2015.7139836.

[22] Mathieu Labbe and Francois Michaud. “Appearance-based loop closure detection for online large-scale and long-term operation”. In: *IEEE Transactions on Robotics* 29.3 (2013), pp. 734–745.

[23] Muhamad Risqi U Saputra, Andrew Markham, and Niki Trigoni. “Visual SLAM and structure from motion in dynamic environments: A survey”. In: *ACM Computing Surveys (CSUR)* 51.2 (2018), pp. 1–36.

[24] Xingming Wu et al. “A Robust SLAM towards Dynamic Scenes Involving Non-rigid Objects”. In: *2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA)*. 2020, pp. 1641–1646. DOI: 10.1109/ICIEA48937.2020.9248394.

[25] J. Sturm et al. “A Benchmark for the Evaluation of RGB-D SLAM Systems”. In: *Proc. of the International Conference on Intelligent Robot Systems (IROS)*. Oct. 2012.

[26] Andreas Geiger et al. “Vision meets Robotics: The KITTI Dataset”. In: *International Journal of Robotics Research (IJRR)* (2013).

[27] Chao Yu et al. “DS-SLAM: A Semantic Visual SLAM towards Dynamic Environments”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 1168–1174. DOI: 10.1109/IROS.2018.8593691.

[28] Fangwei Zhong et al. “Detect-SLAM: Making Object Detection and SLAM Mutually Beneficial”. In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2018, pp. 1001–1010. DOI: 10.1109/WACV.2018.00115.

[29] Shihao Shen et al. “Dytanvo: Joint refinement of visual odometry and motion segmentation in dynamic environments”. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 4048–4055.

[30] Wenshan Wang, Yaoyu Hu, and Sebastian Scherer. “TartanVO: A Generalizable Learning-based VO”. In: (2020).

[31] Elia Bonetto, Chenghao Xu, and Aamir Ahmad. *GRADE: Generating Realistic Animated Dynamic Environments for Robotics Research*. 2023. DOI: 10.48550/ARXIV.2303.04466. URL: https://arxiv.org/abs/2303.04466.

[32] Julio A Placed et al. “A survey on active simultaneous localization and mapping: State of the art and new frontiers”. In: *IEEE Transactions on Robotics* (2023).

[33] N. Koenig and A. Howard. “Design and use paradigms for Gazebo, an open-source multi-robot simulator”. In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Vol. 3. 2004, 2149–2154 vol.3. DOI: 10.1109/IROS.2004.1389727.

[34] Morgan Quigley et al. “ROS: an open-source Robot Operating System”. In: *ICRA workshop on open source software*. Vol. 3. 3.2. Kobe, Japan. 2009, p. 5.

[35] Alexey Dosovitskiy et al. “CARLA: An Open Urban Driving Simulator”. In: *Proceedings of the 1st Annual Conference on Robot Learning*. 2017, pp. 1–16.

[36] Olivier Michel. “Webots: Symbiosis Between Virtual and Real Mobile Robots”. In: *Virtual Worlds*. Ed. by Jean-Claude Heudin. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 254–263. ISBN: 978-3-540-68686-6.

[37] E. Rohmer, S. P. N. Singh, and M. Freese. “CoppeliaSim (formerly V-REP): a Versatile and Scalable Robot Simulation Framework”. In: *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*. 2013.

[38] Christine Connolly. “Technology and applications of ABB RobotStudio”. In: *Industrial Robot: An International Journal* 36.6 (2009), pp. 540–545.

[39] Jiafei Duan et al. “A Survey of Embodied AI: From Simulators to Research Tasks”. In: *IEEE Transactions on Emerging Topics in Computational Intelligence* 6.2 (2022), pp. 230–244. DOI: 10.1109/TETCI.2022.3141105.

[40] Fei Xia et al. “Gibson env: real-world perception for embodied agents”. In: *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. IEEE. 2018.

[41] Manolis Savva* et al. “Habitat: A Platform for Embodied AI Research”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019.

[42] “AI2-THOR: An Interactive 3D Environment for Visual AI”. In: *ArXiv* abs/1712.05474 (2017).

[43] Shital Shah et al. “AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles”. In: *Field and Service Robotics*. 2017. eprint: arXiv:1705.05065. URL: https://arxiv.org/abs/1705.05065.

[44] Matthias Mueller et al. “Sim4CV: A Photo-Realistic Simulator for Computer Vision Applications”. In: *International Journal of Computer Vision* 126 (Sept. 2018). DOI: 10.1007/s11263-018-1073-7.

[45] David Hall et al. “BenchBot environments for active robotics (BEAR): Simulated data for active scene understanding research”. In: *The International Journal of Robotics Research* 41.3 (2022), pp. 259–269.

[46] Meysam Madadi et al. “Learning Cloth Dynamics: 3D + Texture Garment Reconstruction Benchmark”. In: *Proceedings of the NeurIPS 2020 Competition and Demonstration Track, PMLR*. Vol. 133. 2021, pp. 57–76.

[47] Huan Fu et al. “3d-front: 3d furnished rooms with layouts and semantics”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 10933–10942.

[48] Matthew Loper et al. “SMPL: A Skinned Multi-Person Linear Model”. In: *ACM Trans. Graphics (Proc. SIGGRAPH Asia)* 34.6 (Oct. 2015), 248:1–248:16.

[49] Naureen Mahmood et al. “AMASS: Archive of Motion Capture as Surface Shapes”. In: *International Conference on Computer Vision*. Oct. 2019, pp. 5442–5451.

[50] Angel X. Chang et al. *ShapeNet: An Information-Rich 3D Model Repository*. Tech. rep. arXiv:1512.03012 [cs.GR]. Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[51] Anthony G. Francis et al., eds. *Google Scanned Objects: A High-Quality Dataset of 3D Scanned Household Items*. 2022.

[52] Michael Burri et al. “The EuRoC micro aerial vehicle datasets”. In: *The International Journal of Robotics Research* (2016). DOI: 10.1177/0278364915620033. URL: http://ijr.sagepub.com/content/early/2016/01/21/0278364915620033.abstract.

[53] Wenshan Wang et al. “TartanAir: A Dataset to Push the Limits of Visual SLAM”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 4909–4916. DOI: 10.1109/IROS45743.2020.9341801.

[54] Mihai Bujanca et al. “Robust SLAM Systems: Are We There Yet?” In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 5320–5327. DOI: 10.1109/IROS51168.2021.9636814.

[55] Raúl Mur-Artal and Juan D. Tardós. “ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras”. In: *IEEE Transactions on Robotics* 33.5 (2017), pp. 1255–1262. DOI: 10.1109/TRO.2017.2705103.

[56] H. Durrant-Whyte and T. Bailey. “Simultaneous localization and mapping: part I”. In: *IEEE Robotics Automation Magazine* 13.2 (2006), pp. 99–110. DOI: 10.1109/MRA.2006.1638022.

[57] Carlos Campos et al. “ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM”. In: *IEEE Transactions on Robotics* 37.6 (2021), pp. 1874–1890. DOI: 10.1109/TRO.2021.3075644.

[58] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. “SVO: Fast Semi-Direct Monocular Visual Odometry”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2014.

[59] Tong Qin, Peiliang Li, and Shaojie Shen. “VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator”. In: *IEEE Transactions on Robotics* 34.4 (2018), pp. 1004–1020. DOI: 10.1109/TRO.2018.2853729.

[60] Tong Qin et al. “A general optimization-based framework for local odometry estimation with multiple sensors”. In: *arXiv preprint arXiv:1901.03638* (2019).

[61] J. Engel, V. Koltun, and D. Cremers. “Direct Sparse Odometry”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (Mar. 2018).

[62] Yuxiang Sun, Ming Liu, and Max Q.-H. Meng. “Improving RGB-D SLAM in dynamic environments: A motion removal approach”. In: *Robotics and Autonomous Systems* 89 (2017), pp. 110–122. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2016.11.012. URL: https://www.sciencedirect.com/science/article/pii/S0921889015302232.

[63] Rainer Kümmerle et al. “G2o: A general framework for graph optimization”. In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 3607–3613. DOI: 10.1109/ICRA.2011.5979949.

[64] Joseph Redmon and Ali Farhadi. “YOLOv3: An Incremental Improvement”. In: (Apr. 2018).

[65] Kaiming He et al. “Mask R-CNN”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017.

[66] Chenjie Wang et al. “DymSLAM: 4D Dynamic Scene Reconstruction Based on Geometrical Motion Segmentation”. In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 550–557. DOI: 10.1109/LRA.2020.3045647.

[67] Irene Ballester et al. “DOT: Dynamic Object Tracking for Visual SLAM”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 11705–11711. DOI: 10.1109/ICRA48506.2021.9561452.

[68] Ao Li et al. “DP-SLAM: A visual SLAM with moving probability towards dynamic environments”. In: *Information Sciences* 556 (2021), pp. 128–142. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2020.12.019.

[69] Jiyu Cheng, Hong Zhang, and Max Q.-H. Meng. “Improving Visual Localization Accuracy in Dynamic Environments Based on Dynamic Region Removal”. In: *IEEE Transactions on Automation Science and Engineering* 17.3 (2020), pp. 1585–1596. DOI: 10.1109/TASE.2020.2964938.

[70] Yanan Wang et al. “DRG-SLAM: A Semantic RGB-D SLAM using Geometric Features for Indoor Dynamic Scene”. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 1352–1359. DOI: 10.1109/IROS47612.2022.9981238.

[71] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. “SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.12 (2017), pp. 2481–2495. DOI: 10.1109/TPAMI.2016.2644615.

[72] Wei Liu et al. “Ssd: Single shot multibox detector”. In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer. 2016, pp. 21–37.

[73] Wenxin Wu et al. “YOLO-SLAM: A semantic SLAM system towards dynamic environment with geometric constraint”. In: *Neural Computing and Applications* (2022), pp. 1–16.

[74] Seungwon Song et al. “DynaVINS: A Visual-Inertial SLAM for Dynamic Environments”. In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 11523–11530. DOI: 10.1109/LRA.2022.3203231.

[75] Yu Lu et al. “RGB-D SLAM Using Scene Flow in Dynamic Environments”. In: *2022 28th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*. 2022, pp. 1–6. DOI: 10.1109/M2VIP55626.2022.10041080.

[76] Tianwei Zhang et al. “FlowFusion: Dynamic Dense RGB-D SLAM Based on Optical Flow”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 7322–7328. DOI: 10.1109/ICRA40945.2020.9197349.

[77] Yuanhong Zhong et al. “WF-SLAM: A Robust VSLAM for Dynamic Scenarios via Weighted Features”. In: *IEEE Sensors Journal* 22.11 (2022), pp. 10818–10827. DOI: 10.1109/JSEN.2022.3169340.

[78] Jiaming He et al. “OVD-SLAM: An Online Visual SLAM for Dynamic Environments”. In: *IEEE Sensors Journal* (2023), pp. 1–1. DOI: 10.1109/JSEN.2023.3270534.

[79] Mathieu Labbé and François Michaud. “RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation”. In: *Journal of Field Robotics* 36.2 (2019), pp. 416–446. DOI: https://doi.org/10.1002/rob.21831. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21831. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21831.

[80] Raluca Scona et al. “StaticFusion: Background Reconstruction for Dense RGB-D SLAM in Dynamic Environments”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 3849–3856. DOI: 10.1109/ICRA.2018.8460681.

[81] Ran Long et al. “RGB-D-Inertial SLAM in Indoor Dynamic Environments with Long-term Large Occlusion”. In: *arXiv preprint arXiv:2303.13316* (2023).

[82] Martin Runz, Maud Buffier, and Lourdes Agapito. “MaskFusion: Real-Time Recognition, Tracking and Reconstruction of Multiple Moving Objects”. In: *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 2018, pp. 10–20. DOI: 10.1109/ISMAR.2018.00024.

[83] Tsung-Yi Lin et al. “Microsoft coco: Common objects in context”. In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer. 2014, pp. 740–755.

[84] Elia Bonetto, Chenghao Xu, and Aamir Ahmad. “Learning from synthetic data generated with GRADE”. In: *ICRA2023 Workshop on Pretraining for Robotics (PT4R)*. June 2023. URL: https://openreview.net/forum?id=SUIOuV2y-Ce.

[85] Zhen Li et al. “Towards An End-to-End Framework for Flow-Guided Video Inpainting”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[86] Ashkan Mirzaei et al. “SPIn-NeRF: Multiview Segmentation and Perceptual Inpainting with Neural Radiance Fields”. In: *CVPR*. 2023.

[87] Deqing Sun et al. “PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume”. In: 2018.

[88] A. Dosovitskiy et al. “FlowNet: Learning Optical Flow with Convolutional Networks”. In: *IEEE International Conference on Computer Vision (ICCV)*. 2015. URL: http://lmb.informatik.uni-freiburg.de/Publications/2015/DFIB15.

[89] Zhaoyang Huang et al. “FlowFormer: A Transformer Architecture for Optical Flow”. In: *ECCV* (2022).

[90] Yuxiang Sun, Ming Liu, and Max Q-H Meng. “Improving RGB-D SLAM in dynamic environments: A motion removal approach”. In: *Robotics and Autonomous Systems* 89 (2017), pp. 110–122.

[91] Yuxiang Sun, Ming Liu, and Max Q-H Meng. “Motion removal for reliable RGB-D SLAM in dynamic environments”. In: *Robotics and Autonomous Systems* 108 (2018), pp. 115–128.

[92] Simon Niklaus and Feng Liu. “Softmax splatting for video frame interpolation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 5437–5446.

[93] Tete Ji, Chen Wang, and Lihua Xie. “Towards Real-time Semantic RGB-D SLAM in Dynamic Environments”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 11175–11181. DOI: 10.1109/ICRA48506.2021.9561743.

[94] Jiacheng Zhang et al. “DCS-SLAM: A Semantic SLAM with Moving Cluster towards Dynamic Environments”. In: *2022 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. 2022, pp. 1923–1928. DOI: 10.1109/ROBIO55434.2022.10011980.

[95] Xinggang Hu et al. “CFP-SLAM: A Real-time Visual SLAM Based on Coarse-to-Fine Probability in Dynamic Environments”. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 4399–4406. DOI: 10.1109/IROS47612.2022.9981826.

APPENDIX

A OVERVIEW

In this supplementary material, we present additional information about both our approach and our experimental analysis. In Sec. B we detail our modifications to the E2FGVI framework with examples of inpainted frames on TUM-RGBD dataset. In Sec. C we show the complete workflow with explanatory images of both the *movable* and *moving* probability synthesis as well as additional visualization examples of our method. Finally, in Sec. D, we report all our extended results and some additional box plots based on the tracking rate.

B INPAINTING OF TUM-RGBD DATASET

As mentioned in Sec. **??**, the static background images are essential for our implementation. Those are not directly available for the TUM-RGBD sequences. We use the *end-to-end flow-guided video inpainting* (E2FGVI) technique to eliminate dynamic objects. With an NVIDIA Quadro P5000 with 16 GB of memory, E2FGVI can only load a limited number of frames. To overcome this limitation for longer sequences, we adopt a 50-frame sliding window approach with 100 frames bootstrap. In this way, the inpainting module can process 50 new frames with 50 initially inpainted frames from the last step. The initially inpainted frames can also be adjusted during the new processing cycle, which can be formulated as:

$$I^i_{inpaint}(x,y) = \lambda \cdot I^i_{init}(x,y) + (1-\lambda) \cdot I^i_{new}(x,y) \tag{6.1}$$

where $\lambda = 0.5$ in our deployment. Despite the original framework has remarkable performance on static/slightly moving camera views, it is difficult to adapt to varying environments with long-term moving cameras. The original framework takes use of *neighboring frames* and *reference frames* (extracted based on specific time intervals) to inpaint the current frame. In order to further provide sufficient background information in long-term sequences, we modify the selection strategy of the reference frames to remove irrelevant frames that are observing irrelevant areas. Given the ground-truth camera poses, we select the frames with the closest viewpoint as reference frames which should not be overlapped with the neighboring frames. In this way, these new reference frames can largely cover the possible background regions to improve the inpainting process.

As shown in Fig. iii, we show the inpainted frames from various viewpoints in the TUM-RGBD dataset, where our modified E2FGVI can also generalize to blurry images and provide reliable inpainting performance in different camera angles. We also exhibit some failure cases in Fig. i, the main reason lies in that neighboring/reference frames at other timestamps providing false information, such as the objects (e.g. chairs, books) actually staying at other positions in the current frame. Another reason lies in the fact that

reference frames sometimes can not provide sufficient information about the occluded background, leading to a large blur in some cases.

Figure i: Visualization of failure cases of inpainting process.

C MOTION PROBABILITY DEMONSTRATIONS

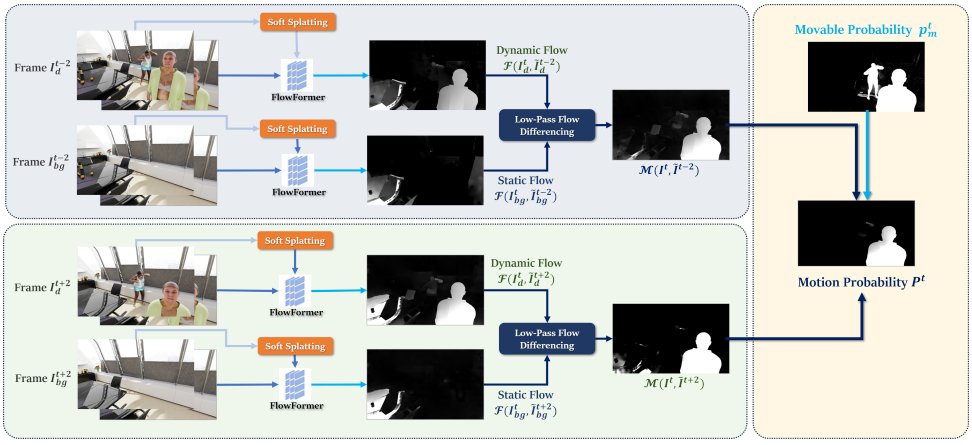


Figure ii: Flow diagram of moving estimation (Sec. 5.1.2) and final motion probability (Sec. 5.1.3).

In this section, we demonstrate the progressive improvements of our two-staged probability estimation module as shown in Fig.ii. In terms of the movable estimation part, the combination of f_1 and f_2 in Eq. 5.3 addresses a trade-off on reducing the noisy difference value and capturing the shadow/reflection areas. More illustrations for this process are exhibited in Fig. iv. The movable distribution acts as a confidence score to further constrain the following moving estimation to effectively eliminates the inherent errors on the static regions (see Fig. vi).

As we discussed in Sec. 5.1.2, the view synthesis based on soft-splatting [92] can effectively reduce the aligning error between the reprojected frame $\widetilde{I}^{t+i}$ and current

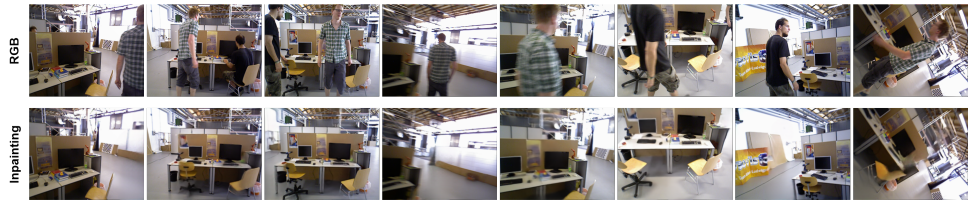


Figure iii: Visualization of inpainted background images from TUM-RGBD dataset, covering blurry images and various viewpoints.

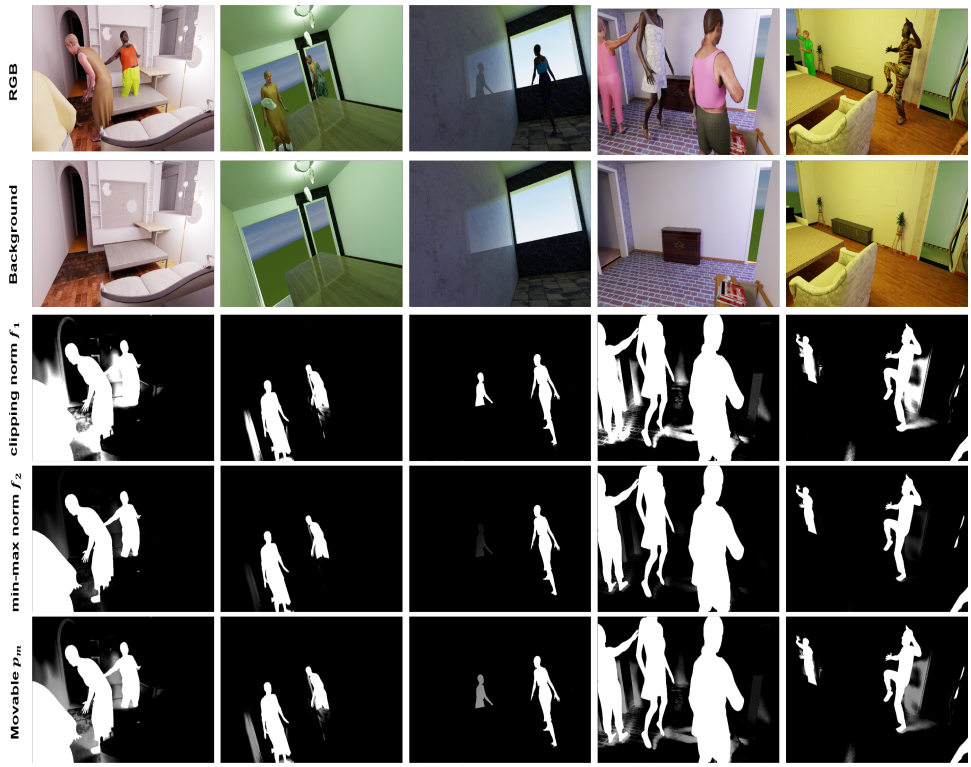


Figure iv: Visualization of movable region estimation, covering shadows, reflections, and movable objects.

frame I^t, significantly decreasing the errors in flow estimation generated from incomplete elimination of camera motion. In Fig. v we can see the computation of the optical flow on splatted frames gives us a more effective elimination of camera motion than the homography-based method. This is indicated by the coloring of the frame which goes towards white in static regions. Moreover, we can see that there exist similarities between flows estimated on static and dynamic scenes, as shown in details in Fig. vi. Given this insight, we developed a flow differencing approach to further eliminate residual camera motion. This can decrease the flow estimation error on the common static regions, while the low-pass filter stores the original estimation result in dynamic regions to avoid false

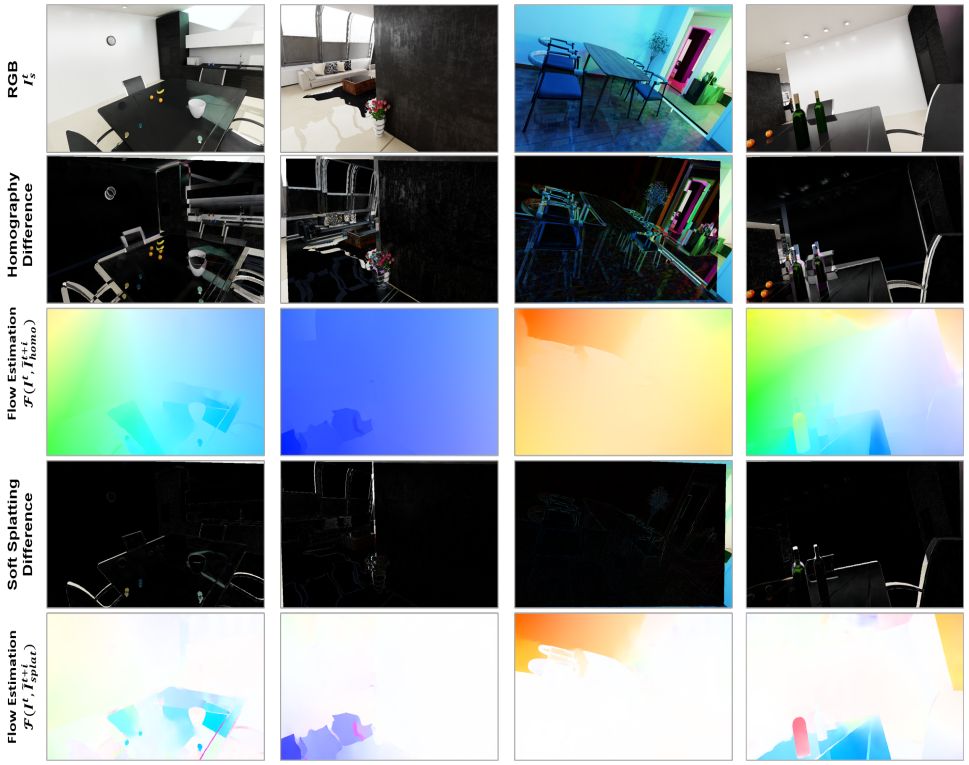


Figure v: Visualization of the performance of projecting frames through **Soft Splatting** or **Homography Transformation** techniques using frames for static scenes. The second line and fourth line differences the frame $\widetilde{I}^{t+i}$ and I^t to illustrate the alignment performance across the frames, while the third line and fifth line demonstrate their effects in flow estimation $\mathscr{F}(I^t, \widetilde{I}^{t+i})$, where flow magnitude should ideally approach to 0 in static regions as discussed in Eq. 5.5.

subtraction.

We also exhibit some failure cases in Fig. vii, which mainly result from the false correspondences in textureless regions (e.g. wall, floor). The specific reason lies in that underlying flow estimation networks fail to capture the correct pixel correspondences in these regions, leading to false moving estimation $\mathscr{M}(I^t, \widetilde{I}^{t+i})$ on the static regions. Despite many of these can be constrained by multiplying them with movable probability, this module sometimes fails to provide accurate motion estimation when there is occlusion. The occlusion directly influences the background differencing process and falsely classifies many regions as movable, being incapable of further reducing the errors generated from flow estimation.

D Detailed SLAM Evaluation Experiments

In Sec. **??**, we exhibit the necessary experimental results to demonstrate our effectiveness in both synthetic sequences and real-world sequences, followed by the analysis by

Figure vi: Visualization of the moving estimation and final motion probability. The second line and third line represent the flow estimated on dynamic scenes $\mathscr{F}(I_d^t, \widetilde{I}_d^{t+i})$ and static scenes $\mathscr{F}(I_s^t, \widetilde{I}_s^{t+i})$ respectively, exhibiting similar estimation errors in common static regions. The fourth line demonstrates the moving estimation $\mathscr{M}(I^t, \widetilde{I}^{t+i})$ through flow differencing. The fifth line denotes the blended motion probability P^t by multiplying movable probability p_m^t

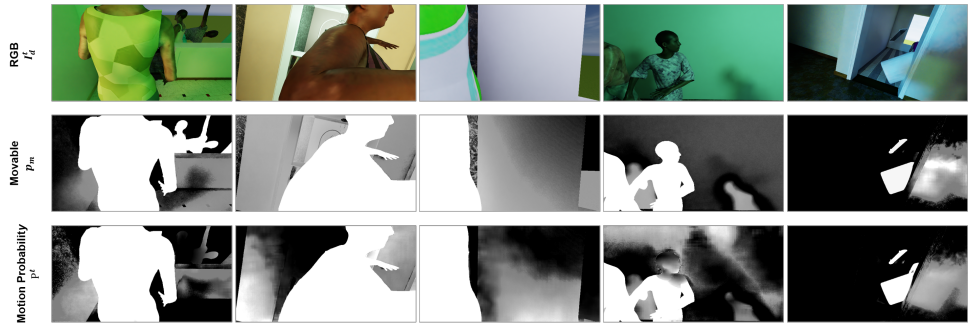


Figure vii: Visualization of failure cases on motion probability.

comparing performance in static/dynamic scenes. Here, we first report the detailed experimental result of each one of the ten runs for each sequence. Then, we analyze the distribution of such results using box plots based on tracking rate amounts to provide more insights into these results.

The detailed results help us put in perspective what we expressed in the paper. For example, we can see how in both sequences F and WO there are wide variations in the outputs of both the tracking rate and ATE, both for the static and dynamic case. This indicates that in those sequences there are critical points which make the method fail in some situations based on the stochastic nature of the underlying SLAM frameworks. These failures are mostly linked to featureless areas, that make the method lose track of the camera movement, and the absence of a subsequent strong recovery procedure. We

can also see how in the tracking rates there is little variability in most situations for both our and the benchmark methods.

Although the average value of both absolute trajectory error (ATE) and tracking rate (TR), with the corresponding standard deviations, can provide an overall evaluation of various methods, comparing the ATE with various TR is also necessary to gather insights on the relations between these two. For this reason we provide a series of plots, one for each experiment, in which, given a tracking rate interval, we report the corresponding ATE statistics for the four methods. We defined four non-overlapping TR intervals, which are *high* (≥ 0.95), *middle* (≥ 0.85), *low* (≥ 0.70), and *rare* (≤ 0.70) tracking rate. Ideally, the estimation distribution should lie in the left-bottom corner, i.e. low trajectory error and high tracking rate, without any presence in other sectors. Anyway, given a sector, the lower the ATE the better. From these plots we can see how DynaPix and DynaPix-D are often located in that region. Notably, with the exception of the *WO - Static Scene* experiment, both are always located in the left-most populated box with better performance than the corresponding method both in terms of ATE and variance of the results, making them not only better but stabler. This holds true for both the synthetic and real-world datasets and especially, but not only, in FH, WOH, S, D, DH, *halfsphere*, *static* and *rpy* sequences. Moreover, these plots also evidence the fact that DynaSLAM is often located *alone* in the right-most section, indicating poor tracking rates in almost all synthetic experiments and in both the *halfsphere* and *rpy* TUM-RGBD sequences. This is clearly reflected by DynaPix-D and evidence of the influence of semantic masks and how blindly discarding features because belonging to a potentially dynamic object can worsen the performance of the method.

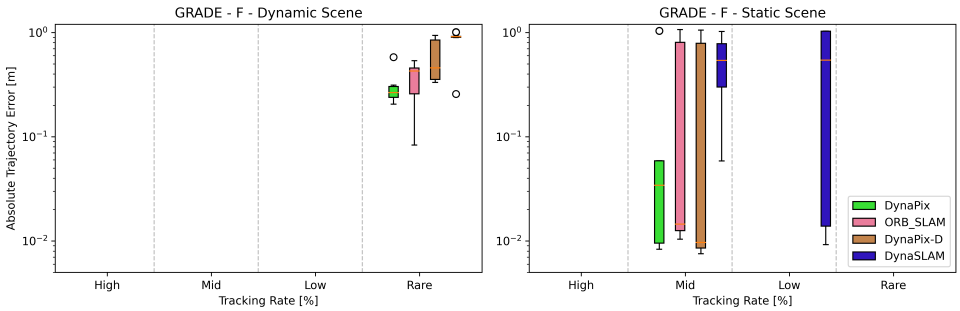


Figure viii: Box-Plot of results of the four methods tested on the F static and dynamic sequences using the tracking rate buckets and the ATE results.

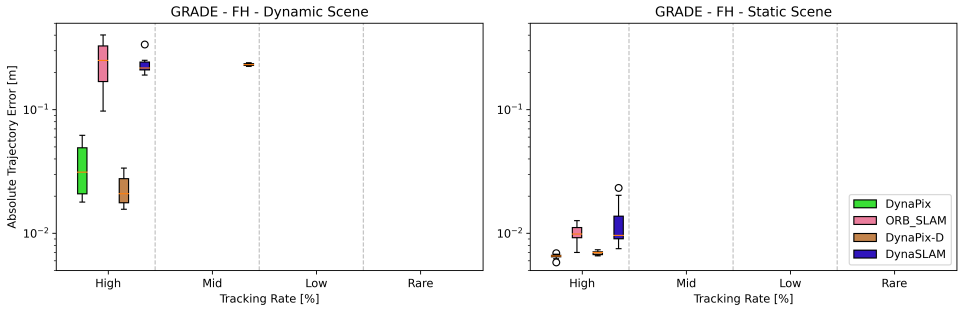


Figure ix: Box-Plot of results of the four methods tested on the FH static and dynamic sequences using the tracking rate buckets and the ATE results.

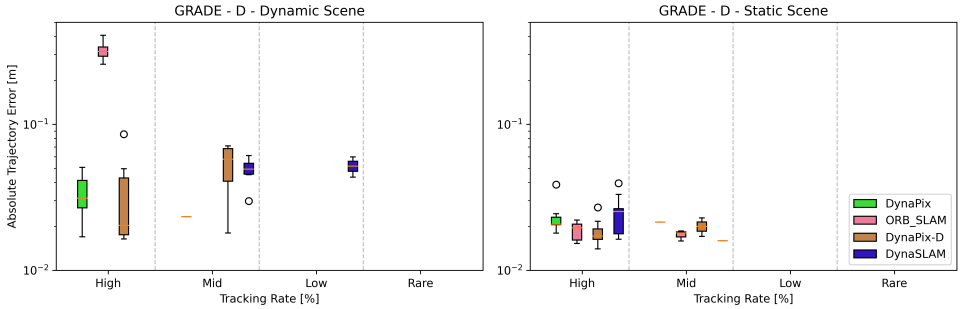


Figure x: Box-Plot of results of the four methods tested on the D static and dynamic sequences using the tracking rate buckets and the ATE results.

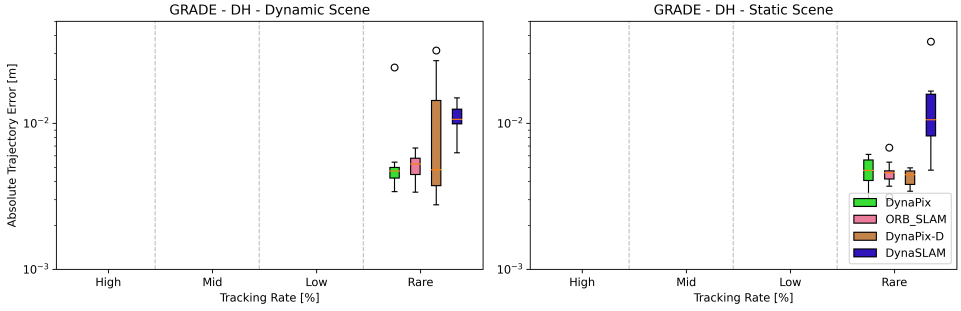


Figure xi: Box-Plot of results of the four methods tested on the DH static and dynamic sequences using the tracking rate buckets and the ATE results.

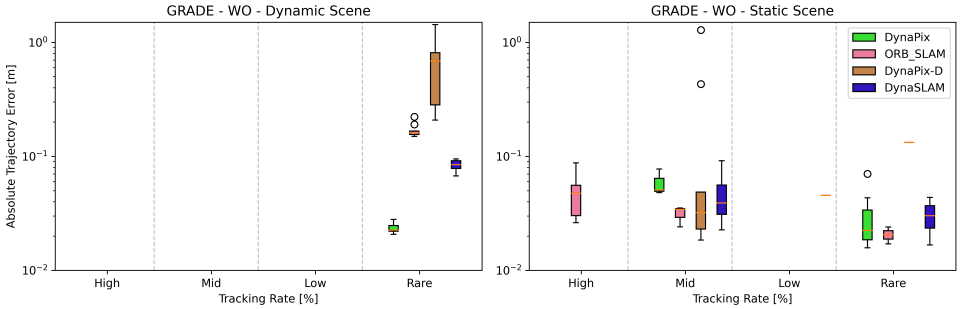


Figure xii: Box-Plot of results of the four methods tested on the WO static and dynamic sequences using the tracking rate buckets and the ATE results.

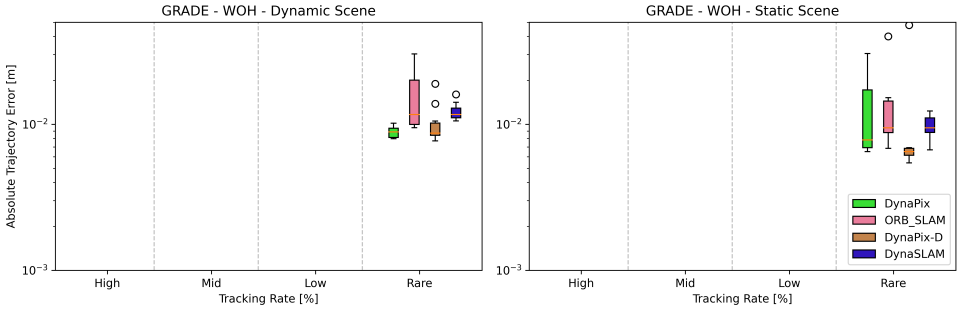


Figure xiii: Box-Plot of results of the four methods tested on the WOH static and dynamic sequences using the tracking rate buckets and the ATE results.

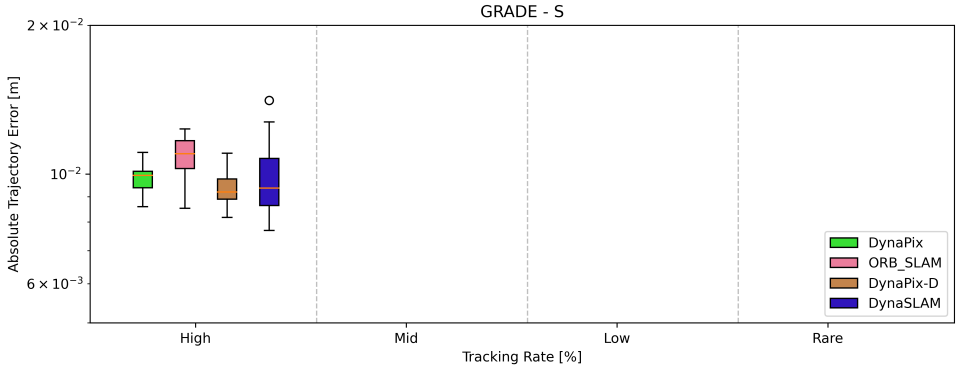


Figure xiv: Box-Plot of results of the four methods tested on the S sequence using the tracking rate buckets and the ATE results.

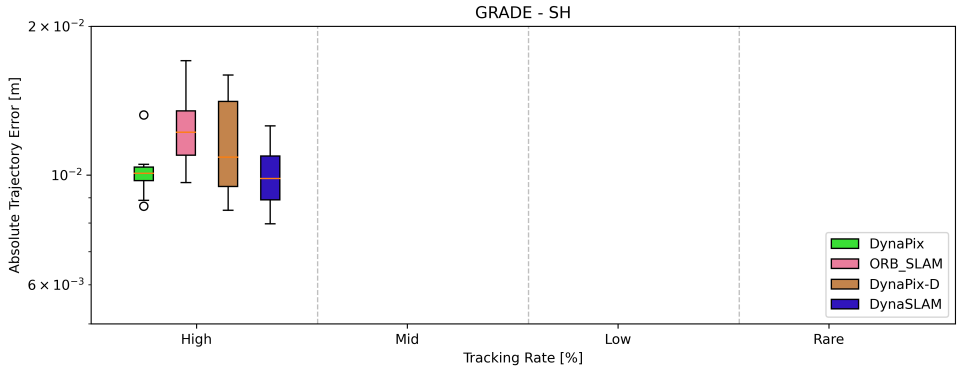


Figure xv: Box-Plot of results of the four methods tested on the SH sequence using the tracking rate buckets and the ATE results.

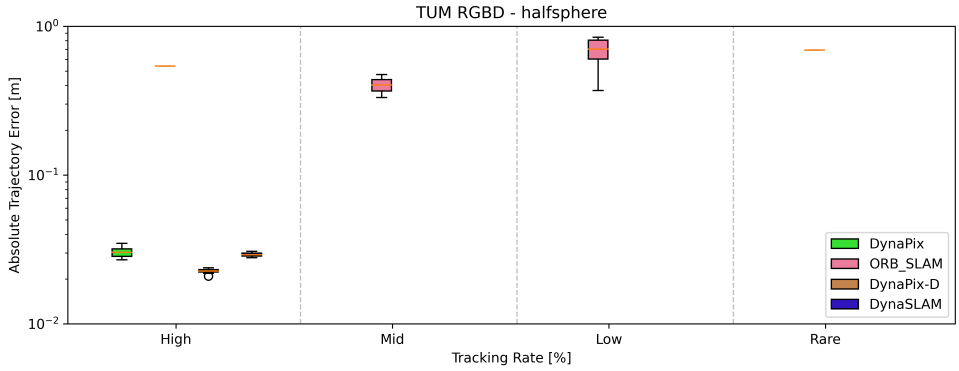


Figure xvi: Box-Plot of results of the four methods tested on the TUM-RGBD *fr3_walking/halfsphere* sequence using the tracking rate buckets and the ATE results.

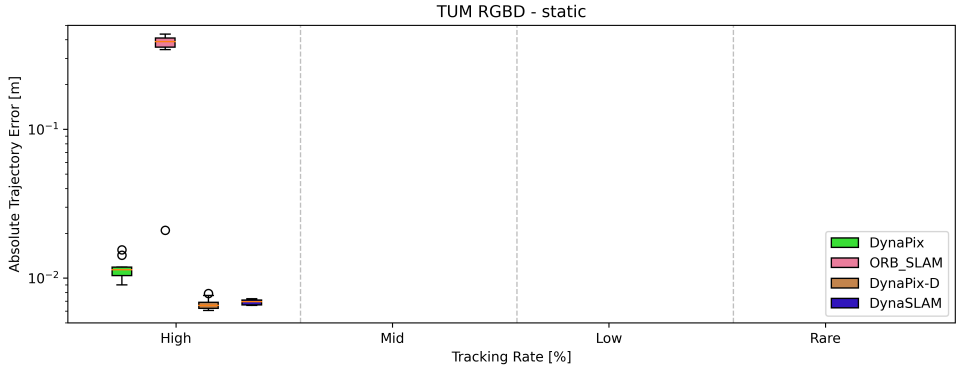


Figure xvii: Box-Plot of results of the four methods tested on the TUM-RGBD *fr3_walking/static* sequence using the tracking rate buckets and the ATE results.

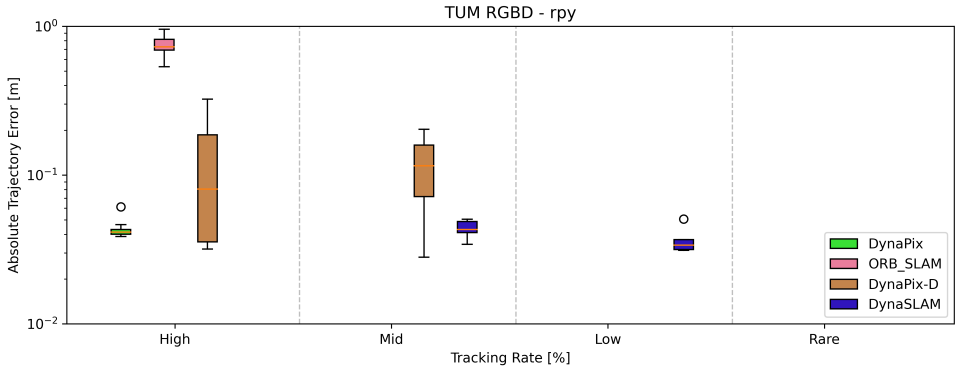


Figure xviii: Box-Plot of results of the four methods tested on the TUM-RGBD *fr3_walking/rpy* sequence using the tracking rate buckets and the ATE results.

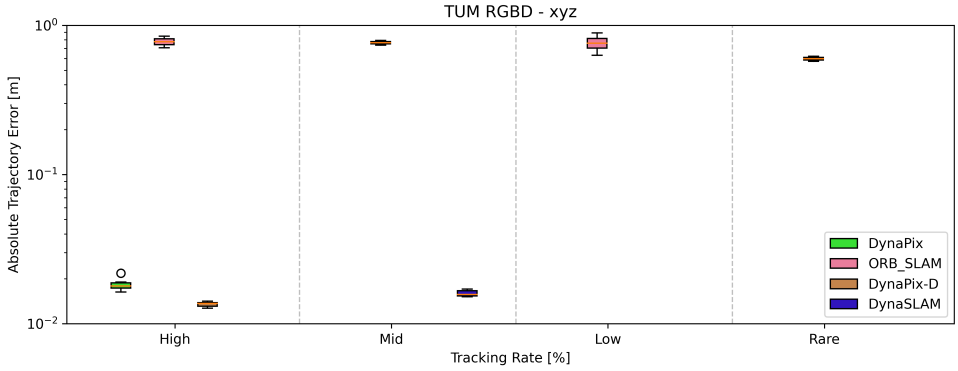


Figure xix: Box-Plot of results of the four methods tested on the TUM-RGBD *fr3_walking/xyz* sequence using the tracking rate buckets and the ATE results.