Computer Vision Group
TUM School of Computation, Information and Technology
Technical University of Munich

Automatically Differentiable Dense Visual Dynamic SLAM for RGB-D Cameras

Scientific work for obtaining the academic degree
Master of Science (M.Sc.) in Mechatronics and Robotics
at the TUM School of Computation, Information and Technology of the Technical University of Munich

Supervisor Prof. Dr. Daniel Cremers
Computer Vision Group

Advisor Mariia Gladkova, M.Sc.
Computer Vision Group

Submitted by Jingkun Feng

Submitted on December 20, 2023 in Garching

Disclaimer

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.

Garching, December 20, 2023

(Signature)

Abstract

Visual Simultaneous Localization and Mapping (vSLAM) in dynamic environments remains a critical challenge for real-world applications. Most existing visual SLAM approaches rely on the static environment assumption, while works addressing dynamic environments require considerable engineering effort, showing limited scalability. This thesis proposes a fully differentiable, dense, dynamic RGB-D SLAM pipeline by leveraging deep learning to avoid heavily engineered components. Building upon ∇SLAM [1], we propose a deep-learning-based visual odometry that performs direct image alignment on feature and uncertainty maps learned by a convolutional neural network (CNN). Additionally, the differentiable Levenberg-Marquardt algorithm and the convolutional M-Estimator are employed to enhance robustness and accuracy in tracking. In line with past works, our pipeline first experimented with the motion segmentation method that combines semantic and geometric segmentation to identify moving objects. Then, we propose a novel moving instance segmentation method by exploiting the uncertainty maps predicted by the CNN. For mapping, the differentiable PointFusion method of ∇SLAM is preserved. Our experiments on synthetic and real-world data demonstrate the effectiveness of our system, showing promising performance in tracking and producing meaningful 4D reconstruction of dynamic environments. In our evaluation, we identify vital limitations of the proposed pipeline and open up potential directions for future work.

Acknowledgements

I am immensely grateful for my time at TUM and all those who helped me grow during the years. First and foremost, I want to thank my advisor, Mariia Gladkova, for her support and guidance. Mariia always led me out of the mist and guided me towards problems worth working on while giving me the freedom to explore. Particularly, her energy and enthusiasm for engaging in research and mentoring students kept encouraging me. This lightened my desire to open up my academic path.

I am fortunate to have worked with Dominik Muhle, Dr. Krishna Murthy Jatavallabhula, and Prof. Daniel Cremers during this thesis. Particular thanks to Mariia and Dominik for initializing this monthly meeting, where I have been inspired by many juicy discussions and their valuable insights.

Thanks to all of my friends and colleagues who we have shared our joy and offered our hands to each other: Lei Cheng, Yang Gao, Yu Lei, Shuwei Hong, Chuyi Wu, Jiahui Wang, and Zhuoyao Zeng.

Special thanks to my two beloved cats, Garfield and Miumiu.

My time during my bachelor's and master's in Germany wouldn't have been the same without the company of my dear girlfriend back then and now my dearest wife, Yongyu Chryseis He.

Finally, none of this would have been possible without the love and support of my parents.

Contents

Chapter 1

Introduction

Simultaneous Localization and Mapping (SLAM) is a cornerstone technology in the rapidly advancing field of robotics and autonomous systems. Its capability to track the system's location while concurrently constructing a map of an unknown environment is pivotal for real-world applications such as autonomous navigation. Particularly, in the realm of visual dynamic SLAM, the challenges intensify as the system must contend with not only mapping and localization but also the inherent complexities of dynamic, ever-changing environments. Despite the recent considerable progress in visual SLAM, appropriately handling dynamic environmental properties remains challenging. On the other hand, leveraging neural networks offers promising avenues in relative sub-tasks and enhances the efficacy of visual SLAM systems. This thesis explores the possibility of further empowering the performance of visual SLAM systems by proposing an automatically differentiable SLAM pipeline for RGB-D images, which provides robust camera tracking and estimates and reconstructs moving rigid bodies in dynamic environments.

1.1 Challenges of Dynamic Visual SLAM

The use of automatic systems such as robots, autonomous vehicles, and drones stands highly on their ability to interact with the environment, which always consists of static and dynamic objects. Thus, for safety and interacting purposes, the system must accurately perceive the dynamic 3-dimensional world in real time.

Visual SLAM, one of the most essential problems in computer vision, has drawn significant interest in robotic applications over the past few years [2]. The majority of existing works have been conducted to develop accurate and robust visual SLAM systems based on the static environment assumption [3] [4]. Although they perform well in static or quasi-static environments, the dynamic participants in environments usually cause catastrophic failure of these methods. Some works have been carried out with weighting factors designed under the prior assumption of input data distribution; some have proposed using an M-estimator. They enhance the robustness to the changes of illumination, inevitable sensor noise, and slight dynamics but cannot still handle complex environmental dynamics in an open world.

With the increasing interest in deploying visual SLAM in real-world tasks, dynamic SLAM has been introduced to target environments with multiple moving entities. Most works estimate camera motion by trivially eliminating the dynamic pixels, while only a few pieces have been conducted to simultaneously track the camera and objects in the scene. Nonetheless, previous works demand considerable engineering effort, as each module must be carefully designed and tuned. As a consequence, this restricts their ready deployment in real-world settings.

Contemporary learning-based methods have shown superiority over classic geometric-based methods in handling more challenging conditions, such as lack of features, poor illumination due to haziness, and significant ego-motions [5][6]. Yet, these methods struggle in dynamic environments if they do not account for moving objects at training, as these objects would introduce unpredictable variations in lighting or occlusions.

1.2 Contributions

The primary intention of this work is not to defeat or surpass any existing dynamic SLAM systems but rather to offer another possibility for the development of visual SLAM in cooperation with other robotic tasks by proposing an end-to-end pipeline.

In this work, we propose an automatically differentiable dense visual SLAM pipeline using RGB-D images, which has robust performance in dynamic environments, coupled with the capability to track and reconstruct moving rigid objects. Extending and adapting the existing differentiable SLAM system, ∇SLAM [1], as our development baseline, we first replace the differentiable Iterative Closest Points (ICP) in ∇SLAM with the unrolled conventional direct method. The combined photometric and geometric errors are minimized using a differentiable Levenberg-Marquardt algorithm in a coarse-to-fine optimization framework. To improve efficiency, our pipeline leverages the inverse compositional approach for optimization. It has been proven that the inverse compositional approach performs comparable to the widely used forward compositional approach and is yet more efficient [7]. Subsequently, inspired by works such as [8], [9], and [5], we propose a convolutional neural network to learn deep feature maps and deep uncertainty maps. We formulate a deep feature-metric residual and down-weight the impact of feature vectors in uncertain regions, which are most likely under distinct lighting changes or contain moving objects.

We introduce two approaches with individual characteristics to handle the moving objects in the scene. The first one utilizes a pretrained segmentation model, MaskRCNN[10], to provide prior information on the foreground objects. Among the predicted foreground objects, we generate a binary mask to filter the regions predicted to be of movable object classes, such as people, animals, etc. Although MaskRCNN can provide temporally consistent segmentation masks to the classes in its train data, it shows limited capability to generalize to novel object categories. To mitigate this limitation, we introduce the second approach, exploiting the uncertainty map to propose dynamic regions. These regions are down-sampled to sets of points. We fed these points as prompts to a generic segmentation model, FastSAM [11], in order to extract segmentation masks. Thanks to the powerful segmentation model, our system can detect all moving objects in the scene. Similar to camera motion estimation, we track the moving objects using bundle adjustment based on the probabilistic feature-metric direct image alignment.

We conduct an extensive evaluation of the proposed pipeline on multiple datasets according to two configurations:

- **Moving camera in static environments.** This configuration reflects the static environment assumption of primary classic optimization-based visual SLAM methods. Evaluation in such a configuration validates our approach's essential capability to track the camera movement with sufficient accuracy and robustness.

- **Moving camera in dynamic environments.** Accurate camera and object motion estimation and robustness to dynamic objects is one of the main objectives of our approach.

If our target is achieved, we validate it with this most realistic and complex configuration.

Moreover, we perform ablation studies to validate the contribution of each component. We also provide a qualitative simulation of the convergence basin of the feature-metric direct image alignment. The simulation demonstrates this method's robustness under significant lighting changes and scenes with multiple dynamic objects.

1.3 Outline

In Chapter 2, it is presented a comprehensive overview of visual SLAM approaches, ranging from general visual SLAM to dense SLAM with RGB-D inputs, and extending to deep-learning-based SLAM techniques. Additionally, we cover the domain of motion segmentation and, ultimately, the integration of these methods into dynamic SLAM.

Chapter 3 provides the necessary background knowledge of the topic. To ease understanding, we first unify the notation we use throughout this thesis. After that, it is introduced the mathematical foundations of the approach developed in this thesis, including the pinhole camera model, rigid-body transformation of camera and objects, and the computational graph.

Chapter 4 details the developed approaches in this thesis. Beginning with an overview of the pipeline, each involved component is detailed following their sequential order in the pipeline, covering the probabilistic feature-metric direct image alignment, the inverse compositional approach, the convolutional neural network to learn feature map and feature-metric uncertainty map, the moving object segmentation methods, and the differentiable association of detected instances.

Chapter 5 presents the evaluation results of the developed system over multiple representative datasets. We demonstrate the system's effectiveness in estimating camera motion and tracking dynamic objects. Quantitative and qualitative results compared to multiple baselines are presented and analyzed in this chapter.

Moreover, in 6, we identify and discuss the contribution of each proposed component in our framework through ablation studies, followed by a discussion about the empirically observed limitations of the system, presenting ideas of possible extensions and future research directions.

Eventually, we summarize our findings of the project in Chapter 7.

Chapter 2

Related Work

2.1 Visual SLAM

Visual SLAM, as a particular form of SLAM, focuses on sensor recordings in monocular, stereo, or RGB-D images. Classic solutions can be divided into feature-based methods (indirect methods), which rely on feature detection and matching, and direct methods, which optimize an objective function over pixel-wise photometric error [2]. Compared to indirect methods, direct methods make use of more information from the image, including lines and intensity variations, which exhibit its capability to work in texture-less scenes [12][13][14]. However, optimization over photometric errors is more difficult since it has a small convergence basin and can easily be stuck in a sub-optimal solution. In this project, we try to avoid this problem by extending direct image alignment to the domain of feature maps inspired by works like [8][5][9].

2.2 Dense RGB-D SLAM

The RGB-D sensor provides explicit depth measurements, simplifying the depth estimation and easing the development of dense SLAM. KinectFusion [15] proposed an RGB-D SLAM approach with a truncated signed distance function (TSDF) based map representation, which is fast and robust in small environments. Successive works [16][17] extend the same principle to large-scale environments using appropriate data structures, e.g., octree map representation. Surface elements (surfels) [18] are another frequently applied map representation in the domain of RGB-D SLAM. Similar to point clouds, a map of surfels encodes extra information - typically a radius and a normal - in addition to the spatial information in its elements. In contrast to TSDF, surfels produce inherently fewer memory footprints. Hence, they can avoid the storage overhead due to switching representation between mapping and tracking, which is typical of TSDF-based fusion methods. PointFusion [19] proposed a surfel-based RGB-D SLAM approach for large environments with local and global loop closure. ElasticFusion [20] took further advantage of surfel-based fusion and introduced a loop closure optimization through non-rigid surface deformations.

2.3 Deep-learning-based SLAM

Deep learning has shown its capability in many computer vision tasks, e.g., classification. It has recently been used to redefine a subset of components of the complete SLAM system to

be differentiable or even directly applied to SLAM. Numerous works have attempted to train models for sub-tasks of visual SLAM, for instance, feature detection [21] and matching [22], optical flow estimation [23], and depth estimation [24][25] [26]. In particular, SuperPoint [21] and its follow-up, SuperGlue [22], were designed to perform feature detection and feature matching and have shown better robustness than hand-crafted features. It has been reported that deep-learning-based methods deliver superior performance in these upstream tasks and, therefore, can facilitate solving the downstream visual SLAM tasks [27] [28] [29] [30].

Additionally, visual representation learning for SLAM-related tasks is of great interest. In contrast to methods that study the geometry part of the task, these works focus on learning robust and invariant dense representations. For example, CodeSLAM [31] presents a dense representation of scene geometry, which is conditioned on intensity data from a single image and generated from a small number of parameters. On the other hand, GN-Net [8] and LM-Reloc [32] propose novel loss functions based on the classic Gauss-Newton and Levenberg-Marquardt optimization algorithms to learn feature-metric representations of the RGB map. Their loss functions constrain the reprojection errors of pixel correspondences instead of the relative pose error between consecutive frames. Likewise, [9] focuses on learning robust and invariant features for camera pose estimation, which can re-localize in large environments given coarse pose priors. Intentionally, these methods tackle re-localization, but their learned features are robust to temporal and seasonal scene changes and can be directly applied for image alignment.

In addition to re-localization, feature-metric representation has also been widely researched to improve the accuracy and robustness of visual odometry. For instance, [33] proposes to use feature-metric loss instead of photometric loss for self-supervised ego-motion estimation. In a concurrent work, Xu et al. extend the idea of [34] employing a convolutional network to learn a pixel-wise feature map along with a deep feature-metric uncertainty map to formulate a deep probabilistic feature-metric residual.

Furthermore, several researchers have attempted to construct SLAM systems that are end-to-end trainable [35][36][37]. ∇SLAM [1] implements differentiable versions of iterative closest points (ICP), Levenberg-Marquardt optimization, and ray casting, allowing for errors in the map to be propagated back to sensor measurements, without sacrificing accuracy. Despite ∇SLAM's full differentiability, it imitates classic methods in a differentiable manner without trainable parameters, meaning it cannot be trained. Thus, its performance is bounded by the accuracy of the classic counterparts of its components. In contrast, iMap[38], NICE-SLAM [39], and NICER-SLAM [40], using a differentiable render with neural implicit representations, neural radiance fields, has an end-to-end trainable pipeline. Meanwhile, Teed et al. propose DROID-SLAM [6], which adopts the state-of-the-art dense optical flow estimation network [23] with a novel implicit global bundle adjustment optimization model to the problem of visual odometry. It outperforms leading methods on a variety of challenging datasets, including Euroc [41] and TartanAir [42] datasets. Nonetheless, it trained the visual odometry beforehand and combined it with a well-designed back-end in the inference time, meaning it is not end-to-end trainable. Based on these works, with the increasing involvement of advanced and well-designed learning-based models, it is anticipated that the upcoming advancement in dense SLAM will lead to enhanced accuracy and robustness.

2.4 Motion Segmentation

Motion segmentation is dedicated to identifying dynamic pixels in an image, which probably capture moving instances. This task is particularly important in dynamic visual SLAM as it facilitates the SLAM system in robust camera tracking and in tracking moving objects. In existing dynamic SLAM systems, moving instances are mostly identified in three ways: 1. Initialization of background/foreground; 2. Utilizing geometry constraints and optical flow; 3. Employing deep learning models[30]. Approaches relying on geometric cues investigate the properties of epipolar geometry to segment static and dynamic features. At the same time, optical flow, corresponding to the motion field in an image, can also be used to segment a moving object [43][44]. Additionally, geometric segmentation can refine masks provided by an instance segmentation approach [45]. In the meantime, researchers have also been using deep learning approaches to tackle motion segmentation. Thanks to the great potential of neural networks, deep features, deep optical flow, and other deep components have been catalyzing new ideas to segment moving objects in the environment [46][47].

Among these three solutions, the first alternative is currently the most popular choice, showing convincing effectiveness in a considerable number of dynamic SLAM systems, such as [48], [49], [45] and [16]. They leverage segmentation models to obtain prior environmental knowledge, splitting background and foreground objects. However, widely used learning-based methods such as Mask R-CNN[10] and DeepLab[50] learn to segment instances under the supervision of labels. Using supervised learning heavily restricts these methods' ability to generalize to real-world scenarios because of a limited range of labels in training datasets. Dynamic SLAM systems mentioned above, acquiring semantic information from these networks, suffer naturally from the bounded generalization of these supervised methods.

Recently, segmentation approaches have addressed this limitation differently. For example, CutLER[51] exploits unsupervised learning to achieve more generalizable object detection and instance segmentation. It utilizes the MaskCut[51] to generate pseudo-masks of multiple instances of an image and train a detector using the pseudo-masks of unlabeled data. On the other hand, Kirillov et al. proposes a promptable segmentation system, Segment Anything (SAM), which simplifies the segmentation process as a foundational model. SAM's meticulously crafted network architecture supports interactive segmentation using different prompts. Together with its successor works, such as [11] [53], SAM revolutionizes the field of computer vision by providing cutting-edge performance and, most importantly, the possibility of using different types of prompts from keypoints, bounding boxes to masks, and even text to select the segmentation of interest. This novel feature opens up interesting research directions in further exploiting semantic information in various tasks, including dynamic visual SLAM. In this work, we exploit this potential by integrating the FastSAM [11] in our pipeline and using points spawning from an uncertainty map as prompts to segment moving instances.

2.5 Dynamic SLAM

As a specific case of semantic SLAM, dynamic SLAM mainly focuses on identifying and appropriately handling the active entities in environments. As introduced in the last section, SLAM systems can detect moving objects using deep-learning-based segmentation models, geometric cues, and neural networks. Once objects in motion are identified, multiple ways exist to leverage this information to enhance SLAM. Several existing approaches suggest treating moving objects as outliers and eliminating them from the map [48][54]. For instance, DynaSLAM [48] tackles the tasks by detecting moving objects and filtering them before feeding

the visual cues into the visual odometry pipeline. He et al. propose an online vSLAM system without predefined dynamic labels for dynamic environments, which is built on ORB-SLAM3 [56] and combines semantic information, depth information, and optical flow to distinguish foreground and background and recognize dynamic regions. Though they achieve superior performance in estimating camera trajectories, trivially ignoring pixels capturing moving objects leads to loss of information on images. Consequently, methods like these provide restricted aid to scenarios where robots interact with other dynamic participants around them.

On the contrary, [49][45] [16] have developed methods that effectively reduce the impact of dynamic elements in scenes. They estimate ego camera motion while simultaneously tracking and reconstructing the moving objects. Grinvald et al. introduce a novel multi-object TSDF formulation [57] that can track and reconstruct dynamic objects. This formulation allows for maintaining a single volume for the entire scene with all objects therein. Nevertheless, these solutions are often heavily engineered and remain improvable, with scope for substantial enhancements.

Chapter 3

Preliminaries

In this chapter, we first unify the notations used in the thesis and then revisit the important concepts, including the rigid body transformation, the pinhole camera projection, and the computational graph, which are prerequisites to understanding the problem we tackle and the proposed approaches.

3.1 Notation

Throughout the paper, we distinguish scalars, vectors, and matrices by denoting them differently, i.e., scalars with regular letters s, vectors with small bold letters $\mathbf{v}$, and matrices with capital bold letters $\mathbf{M}$. Moreover, we differ homogeneous coordinates $\bar{\mathbf{p}}$ from conventional coordinates $\mathbf{p}$.

Localization is a problem of estimating transformation between different coordinate systems. Therefore, in this thesis, we may encounter vast transformation calculations. For this, we denote coordinate systems using the symbol $\mathcal{F}$ with a subscript to identify a specific frame. For example, $\mathcal{F}_W$ represents the world coordinate system, and $\mathcal{F}_C$ is the camera coordinate system. The transformation from $\mathcal{F}_W$ to $\mathcal{F}_C$ is written as $\mathbf{T}_{WC}$.

3.2 Rigid Body Transformation

3.2.1 Euclidean Transformation between Coordinate Systems

Rigid body transformation, also known as Euclidean transformation, interprets the movement of an object in Euclidean space with respect to a coordinate system while preserving its size and shape - the Euclidean distance between every pair of points of the body. Formally, a rigid body transformation T in 3-dimensional Euclidean space is a map:

$$T : \mathbb{R}^3 \mapsto \mathbb{R}^3 \tag{3.1}$$

such that for any points $\mathbf{p}, \mathbf{q} \in \mathbb{R}^3$ belong to the body denoted as a set of points X the following conditions hold true:

$$\|\mathbf{p} - \mathbf{q}\| = \|T(\mathbf{p}) - T(\mathbf{q})\| \quad \forall \mathbf{p}, \mathbf{q} \in X \tag{3.2}$$

$$T(\mathbf{p}) \times T(\mathbf{q}) = T(\mathbf{p} \times \mathbf{q}) \quad \forall \mathbf{p}, \mathbf{q} \in X \tag{3.3}$$

In general, a rigid-body transformation consists of a rotation and a translation. Various ways are used to represent the rotation, including axis-angle, Euler angle, and quaternions.

To ease the read, we mainly employ the rotation matrix $\mathbf{R}$, a 3×3 orthogonal matrix with the following special properties:

$$\det(\mathbf{R}) = 1 \tag{3.4}$$

$$\mathbf{R}\mathbf{R}^T = \mathbf{I} \tag{3.5}$$

It belongs to the special orthogonal group defined as:

$$SO(n) = \left\{\mathbf{R} \in \mathbb{R}^{n\times n} | \mathbf{R}\mathbf{R}^T = \mathbf{I}, \det \mathbf{R} = 1\right\} \tag{3.6}$$

which particularly specifies the rotation in 3D space. Together with a translation vector $\mathbf{t} \in \mathbb{R}^3$, a rigid body transformation T of a 3-dimensional coordinates $\mathbf{p}$ can be then expressed as:

$$T(\mathbf{p}) = \mathbf{R}\mathbf{p} + \mathbf{t} \tag{3.7}$$

The 3-dimensional coordinates $\mathbf{p} = [x, y, z]^T$ can be expended to the so-called homogeneous coordinates, $\bar{\mathbf{p}} = [x.y, z.1]^T$. We can also integrate the rotation matrix $\mathbf{R} \in SO(3)$ and the translation vector $\mathbf{t} \in \mathbb{R}^3$ in a 4×4 matrix called transformation matrix $\mathbf{T}$ as:

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \tag{3.8}$$

The resulting matrix $\mathbf{T}$ is in the special Euclidean group $SE(3)$ defined as:

$$SE(3) = \left\{ \mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4} \quad | \quad \mathbf{R} \in SO(3), \mathbf{t} \in \mathbb{R}^3 \right\} \tag{3.9}$$

Then, we can formulate a rigid body transformation easily with a matrix multiplication as $T(\mathbf{p}) = \mathbf{T}\bar{\mathbf{p}}$. On top of that, transformations can be chained as matrix products:

$$T_2(T_1(\mathbf{p})) = \mathbf{T}_2\,\mathbf{T}_1\,\bar{\mathbf{p}} \tag{3.10}$$

Note that this matrix multiplication results in a homogeneous coordinate, which should be converted to a conventional Euclidean coordinate afterwards.

Since the Rotation matrix is orthogonal, its transpose $\mathbf{R}^T$ denotes an inverse rotation. This is particularly useful as it allows us to represent the inverse transformation as the inverse of matrix $\mathbf{T}$:

$$\mathbf{T}^{-1} = \begin{bmatrix} \mathbf{R}^T & -\mathbf{R}^T\mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \tag{3.11}$$

Additionally, we represent the identity transformation matrix as a 4×4 identity matrix $\mathbf{I}_{4\times 4}$

3.2.2 Lie Algebra - Parameterization of Rigid Body Transformation in Optimization

An appropriate parameterization of the rigid body transformation is vital in optimization problems. As interpreted above, the translation and rotation components in a rigid body transformation consist of six degrees of freedom. Parameterizing the translation component in three dimensions, denoted as $\mathbf{t}$, is straightforward. It comprises three elements and possesses three degrees of freedom, corresponding to movements along the x, y, and z axes.

The situation for the rotation component, however, is more complex. It contains nine elements when it is presented as a 3 by 3 matrix. Despite this, it only has three degrees of freedom. This discrepancy arises because of the constraints imposed on the matrix that belongs to $SE(3)$ (Equation (3.6)). Directly representing the constraints of rotation matrices in

optimization problems can be challenging and may limit the range of optimization strategies that can be applied.

As introduced above, the transformation belongs to the special Euclidean group $SE(3)$, a subgroup of the Lie group. It encompasses the space of rigid body transformation in three dimensions. Most importantly, it is a differentiable manifold. This property is significant because it implies we can leverage the Lie algebra, which links to the tangent space at the identity transformation $\mathbf{I}_{4\times4}$.

The Lie algebra representation of a transformation $\mathbf{T}$, denoted as $\xi \in se(3)$, can be expressed as a six-entry vector $[\boldsymbol{\omega}^T, \mathbf{v}^T]^T \in \mathbb{R}^6$. Here, $\boldsymbol{\omega} \in \mathbb{R}^3$ specifies the rotation and can be understood as akin to angular velocity. Likewise, $\mathbf{v} \in \mathbb{R}^3$ indicates the translation, analogous to linear velocity.

We can transform from $\mathbf{T}$ to ξ through the exponential map:

$$exp : se(3) \mapsto SE(3) \tag{3.12}$$

This mapping has a closed-form expression that involves matrix exponentiation and the application of Rodrigues' formula. The inverse mapping, from ξ back to $\mathbf{T}$, is achieved via the logarithm map, defined as:

$$log : SE(3) \mapsto se(3) \tag{3.13}$$

It's worth mentioning that when $\xi = 0$, it corresponds to the identity transformation $\mathbf{I}_{4\times4}$. This relation between $SE(3)$ and $se(3)$ greatly facilitates the optimization of rigid body transformations.

3.2.3 Camera-World Transformation

Most visual odometry algorithms in standard visual SLAM assume that all the observed changes are attributed to the camera motion in space relative to an inertial frame, usually the world frame.

Given a point $\mathbf{p}$ observed by the camera denoted as $\mathbf{p}_C$ w.r.t the camera frame $\mathcal{F}_C$, we can represent the same point in the world frame $\mathcal{F}_W$ by using the relative transformation between $\mathcal{F}_C$ and $\mathcal{F}_W$:

$$\bar{\mathbf{p}}_W = \mathbf{T}_{WC}\bar{\mathbf{p}}_C \tag{3.14}$$

The applied relative transformation here also corresponds to the relative position and orientation of the coordinate frame $\mathbf{F}_C$ as seen in the world coordinate frame $\mathcal{F}_W$. We call this the camera pose. Analogously, we defined the object pose similarly, which is detained in the next section. A schematic visualization of this relative transformation is shown in Figure 3.1.

Tracking camera frame-to-frame estimates the relative transformation $\mathbf{T}^W_{C_tC_{t+1}}$ between one camera frame $\mathcal{F}_{C_t}$ and the other camera frame $\mathcal{F}_{C_{t+1}}$ in the world frame, see Figure 3.2. To ease the reading, we omit the superscript when the relative frame is fixed, such as the world frame. Conventionally, we attach the world coordinate frame to the first camera frame:

$$\mathbf{T}_{WC_0} = \mathbf{I}_{4\times4} \tag{3.15}$$

and convert the successors' pose by concatenating the relative transformations as:

$$\mathbf{T}_{WC_{t+1}} = \mathbf{T}_{WC_0} \dots \mathbf{T}_{C_{t-1}C_t}\mathbf{T}_{C_tC_{t+1}} \tag{3.16}$$

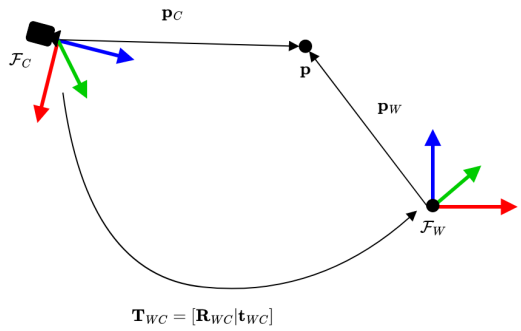


Figure 3.1: Camera-world transformation. World and camera coordinate frames are denoted as $\mathcal{F}_W$ and $\mathcal{F}_C$ respectively. The coordinate frame's axes are colored, x in red, y in green, and z in blue. $\mathbf{T}_{WC}$ represents the camera pose, interpreting the position and orientation of the camera as seen from the world coordinate frame. Meanwhile, $\mathbf{T}_{WC}$ is the rigid body transformation of coordinates in $\mathcal{F}_C$ to $\mathcal{F}_W$.

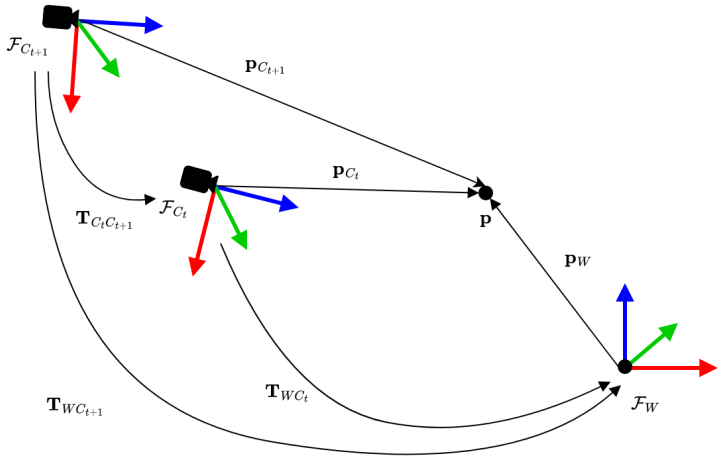


Figure 3.2: Relative camera pose transformation. Given two coordinates of the camera at different time $\mathcal{F}_{C_t}$ and $\mathcal{F}_{C_{t+1}}$ as well as the world frame $\mathcal{F}_W$, $\mathbf{T}_{C_tC_{t+1}}$ represents the relative transformation from $\mathbf{p}_{C_{t+1}}$ to $\mathbf{p}_{C_t}$.

3.2.4 Moving Object Transformation

In the context of dynamic SLAM, in addition to camera poses, we also have to account for moving objects' motion. The observed object motion is a combination of the camera motion as well as its ego motion. Object tracking is accomplished by first estimating the camera's relative transformations w.r.t to the object at a given timestamp j, then converting the transformation to the world coordinate frame.

Let's break down the process and first take a look at estimating camera motion relative to a fixed coordinate frame attached to an object $\mathcal{F}_{O_j}$. Considering the object is temporally static, while the camera is moving, we wish to compute the transformation of the camera from time $j > i$ to i observed in $\mathcal{F}_{O_j}$ denoted as $\mathbf{T}^O_{C_iC_j}$. As estimating the camera relative poses in the world coordinate frame, we express $\mathbf{T}^O_{C_iC_j}$ as:

$$\mathbf{T}^O_{C_iC_j} = (\mathbf{T}^O_{OC_i})^{-1}\,\mathbf{T}^O_{OC_j} \tag{3.17}$$

Given the initial pose of the camera $\mathcal{F}_{C_i}$ in the object frame $\mathbf{T}^W_{C_iO} = \mathbf{T}^O_{C_iO}$, we compute the pose

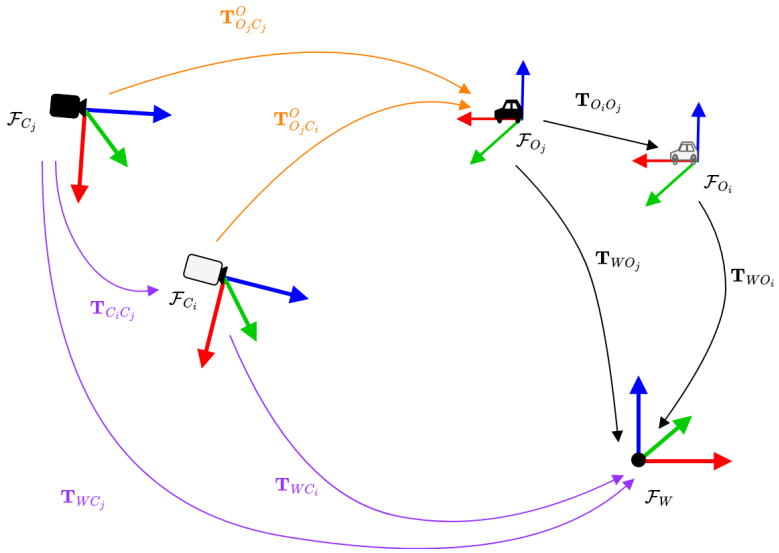


Figure 3.3: Camera and object motions from times i to $j > i$. $\mathbf{T}_{C_iC_j}$ represents camera motion, while $\mathbf{T}_{O_iO_j}$ defines the object motion. Transformations in purple depict camera tracking and those in orange illustrate the computation in Equation (3.18).

of the camera $\mathcal{F}_{C_j}$ in the object frame by concatenating transformations:

$$\begin{aligned}\mathbf{T}^O_{C_jO} &= \mathbf{T}^O_{C_jC_i}\,\mathbf{T}^O_{C_iO} \\ &= (\mathbf{T}^O_{C_iC_j})^{-1}\,\mathbf{T}^O_{C_iO}\end{aligned} \tag{3.18}$$

Given the camera motion from j to i expressed in the world coordinate frame $\mathbf{T}^W_{C_iC_j}$, we can obtain the object pose in world frame by transforming the above result to the world coordinate frame. Sequentially, combining this with camera tracking, we have the ultimate expression for object tracking:

$$\begin{aligned}\mathbf{T}_{WO_j} &= \mathbf{T}_{WC_j}\mathbf{T}^O_{C_jO} \\ &= \mathbf{T}_{WC_j}\mathbf{T}^O_{C_jC_i}\mathbf{T}^O_{C_iO} \\ &= \mathbf{T}_{WC_i}\mathbf{T}_{C_iC_j}\mathbf{T}^O_{C_jC_i}\mathbf{T}^O_{C_iO}\end{aligned} \tag{3.19}$$

3.3 Pinhole Camera Model and Jacobians

We define the image space domain as $\Omega \subset \mathbb{N}^2$, where an RGB-D frame is composed of a depth map $\mathbf{D}$ of depth values $d : \Omega \to \mathbb{R}$ and a colour image $\mathbf{C}$ of 3-channel (**R**ed, **G**reen, **B**lue) colour values in each pixel $\mathbf{c} : \Omega \to \mathbb{N}^3$. The colour map can be converted to an intensity map using

$$\mathbf{I} = 0.299 \cdot \mathbf{R} + 0.587 \cdot \mathbf{G} + 0.114 \cdot \mathbf{B} \tag{3.20}$$

The re-projection of an image point $\mathbf{x}$ with the inverse depth $\rho = \frac{1}{d}$ from a camera frame $\mathcal{F}_{C_i}$ to another camera frame $\mathcal{F}_{C_j}$ is formulated according to the so-called warping function as:

$$\mathbf{x}' = \Pi_c(\mathbf{T}_{ij} \cdot \Pi_c^{-1}(\mathbf{x}, \rho)) \tag{3.21}$$

We calculate the relative transformation $\mathbf{T}_{ij}$ between two frames in the above equation with

$$\mathbf{T}_{ij} = \mathbf{C}_j \circ \mathbf{C}_i^{-1} \tag{3.22}$$

$\mathbf{C}_i$ and $\mathbf{C}_j$ here refer to the transformation from the inertial (world) coordinate system to the respective camera-fixed coordinate system:

$$\mathbf{C}_i = \mathbf{T}_{WC_i}, \quad \mathbf{C}_j = \mathbf{T}_{WC_j} \tag{3.23}$$

i.e. the i-th and j-th camera poses. Furthermore, Π_c in equation (3.21) is the pinhole projection function

$$\Pi_c(\mathbf{p}) = \begin{bmatrix} f_x \frac{X}{Z} + c_x \\ f_y \frac{Y}{Z} + c_y \end{bmatrix} \tag{3.24}$$

and Π_c^{-1} is the inverse projection:

$$\Pi_c^{-1}(\mathbf{x}, \rho) = \begin{bmatrix} \frac{p_x - c_x}{f_x} \\ \frac{p_y - c_y}{f_y} \\ 1 \\ \\ \rho \end{bmatrix} \tag{3.25}$$

given a number of 3D points written in homogeneous coordinates $\bar{\mathbf{p}} = (X, Y, Z, W)^T$, pixel coordinates $\mathbf{x} = (u, v)^T$, and camera intrinsic parameters $c = \{f_x, f_y, c_x, c_y\}$.

To facilitate optimization, it is essential to compute the Jacobians of residuals with respect to camera poses $\mathbf{C}_i$, $\mathbf{C}_j$, and inverse depth ρ. We employ the local parameterizations $e^{\xi_i}\mathbf{C}_i$ and $e^{\xi_j}\mathbf{C}_j$, while considering the inverse depth d as a vector in $\mathbb{R}^1$. Note that here $\xi \in \mathfrak{se}(3)$ is the Lie algebra of the Lie group $\mathbf{T} \in \mathcal{SE}(3)$, and they are a map from one to the other using exponential and logarithm maps as

$$\mathbf{T} = \exp(\xi), \quad \xi = \log(\mathbf{T}) \tag{3.26}$$

The Jacobians of both the projection and inverse projection functions are computed as follows:

$$\frac{\partial \Pi_c(\mathbf{p})}{\partial \mathbf{p}} = \begin{bmatrix} f_x\rho & 0 & -f_x X\rho^2 & 0 \\ f_y\rho & 0 & -f_y X\rho^2 & 0 \end{bmatrix} \tag{3.27}$$

$$\frac{\partial \Pi_c^{-1}(\mathbf{x}, \rho)}{\partial \rho} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.28}$$

Considering the local parameterization, we compute the Jacobian of the 3D point transformation and reorganize it using the adjoint operator:

$$\begin{aligned} \mathbf{p}' &= \exp(\xi_j) \cdot \mathbf{C}_j \cdot (\exp(\xi_i) \cdot \mathbf{C}_i)^{-1} \cdot \mathbf{p} \\ &= \exp(\xi_j) \cdot \mathbf{C}_j \cdot \mathbf{C}_i^{-1} \cdot \exp(-\xi_i) \cdot \mathbf{p} \\ &= \exp(\xi_j) \cdot \exp(-\mathrm{Adj}_{\mathbf{C}_j \mathbf{C}_i^{-1}} \xi_i) \cdot \mathbf{C}_j \cdot \mathbf{C}_i^{-1} \cdot \mathbf{p} \end{aligned} \tag{3.29}$$

This repositioning allows us to compute the Jacobians using the established generators:

$$\begin{aligned} \frac{\partial \mathbf{p}'}{\partial \xi_j} &= \left[\begin{array}{c|c} W'\mathbf{I}_{3\times 3} & -\hat{\mathbf{p}'} \\ \hline \mathbf{0}_{3\times 1} & \mathbf{0}_{3\times 1} \end{array}\right] \\ &= \begin{bmatrix} W' & 0 & 0 & 0 & Z' & -Y' \\ 0 & W' & 0 & -Z' & 0 & X' \\ 0 & 0 & W' & Y' & -X' & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{aligned} \tag{3.30}$$

with the skew-symmetric matrix of the transformed 3D point $\hat{\mathbf{p}}'$. By applying the chain rule, we can calculate the complete Jacobians w.r.t. the twist ξ as:

$$\frac{\partial \mathbf{x}'}{\partial \xi} = \frac{\partial \Pi_c(\mathbf{p}')}{\partial \mathbf{p}'} \frac{\partial \mathbf{p}'}{\partial \xi} \tag{3.31}$$

and w.r.t. the inverse depth d as:

$$\frac{\partial \mathbf{x}'}{\partial \rho} = \frac{\partial \Pi_c(\mathbf{p}')}{\partial \mathbf{p}'} \frac{\partial \mathbf{p}'}{\partial \mathbf{p}} \frac{\partial \Pi_c^{-1}(\mathbf{x}, \rho)}{\partial \rho} \tag{3.32}$$

3.4 Differentiability and Computational Graphs

Differentiability is a fundamental concept in calculus that refers to the ability of a function to have a derivative at each point in its domain. A function $f(x)$ is considered differentiable at a point x if the derivative $f'(x)$ exists at that point. This means that there is a unique tangent to the curve of the function at that point, and the slope of this tangent is the value of the derivative.

Differentiability is crucial in deep learning for several reasons. First, deep learning models like neural networks are typically trained using gradient-based optimization techniques like gradient descent. Applying these techniques is only possible if we can compute the gradients of the loss function with respect to the model's parameters, i.e., the loss function is differentiable. While directly computing the gradients of the loss function is usually cumbersome, in practice, we employ back-propagation to ease the computation. We formulate the loss function as a computational graph to compute each node's gradients with respect to its input. Then, we calculate the gradients of the loss w.r.t each weight by applying the chain rule. Therefore, the differentiability of every operation involved in the loss function is necessary for using back-propagation.

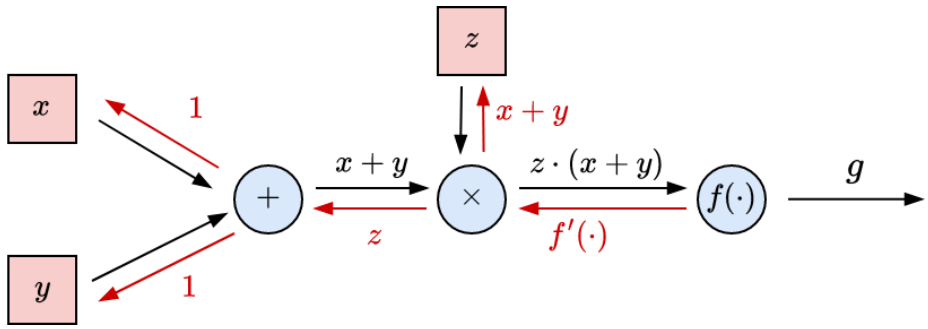


Figure 3.4: A computation graph. The red and blue nodes represent variables and operations, respectively. Directional edges represent data flow. Black edges indicate a forward pass with results from the upstream steps above the arrows, while red edges show local gradients w.r.t. the source variables in the backward pass. This graph computes the function $f(z \cdot (x+y))$ with f being an arbitrary differentiable function. The gradients w.r.t. any variables can be computed using the chain rule. For instance, the gradient w.r.t. to x is $\frac{\partial g}{\partial x} = f'(z(x+y)) \cdot z \cdot 1$.

As introduced above, the computational graph is the foundation of gradient-based learning architectures in deep learning, as it typically represents all functions and approximations. A computational graph is formally defined as a directed acyclic graph, which we denote as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Here, $\mathcal{V}$ is a set of nodes $v \in \mathcal{V}$ holding variables and operators, and the directed edges $\mathcal{E}$ represent the control flow. Moreover, each node specifies the computation rules for

the gradients of its outputs relative to its inputs, ensuring the differentiability of the graph. Function representation in computational graphs offers flexibility, allowing for nesting and composition while preserving differentiability. For instance, an example computational graph for computing g using the function $f(z \cdot (x + y))$ is illustrated in Figure 3.4.

A standard SLAM pipeline often consists of several components that are inherently non-differentiable. Specifying gradients in a computational graph is impossible for some forward computations in these components. For instance, in a dense 3D SLAM pipeline like [15], elements such as nonlinear least squares modules using the Levenberg-Marquardt algorithm and raycasting routines are not differentiable. Additionally, in particular operations like sampling, although gradients exist, they are zero in almost all cases. Consequently, a pipeline involving these operations has extremely sparse gradient flows, which poses a significant challenge in the context of gradient-based learning methods.

Chapter 4

Methodology

In this chapter, we present the core of this paper by detailing the proposed pipeline and the corresponding components. The schematic overview of the pipeline is illustrated in Figure 4.1. We ensure the pipeline's differentiability by integrating end-to-end deep-learning-based components and introducing differentiable counterparts for essentially non-differentiable components due to various facts. Here, we interpret our pipeline component by component, following the process order of our pipeline (Section 4.1-4.9). At the same time, we will explain the differentiable solutions to methods that are originally non-differentiable, such as the Levenberg-Marquardt optimization and the Hungarian algorithm. After introducing the methods, we detail the implementation of the components, presenting the network architectures and training setups (Section 4.10).

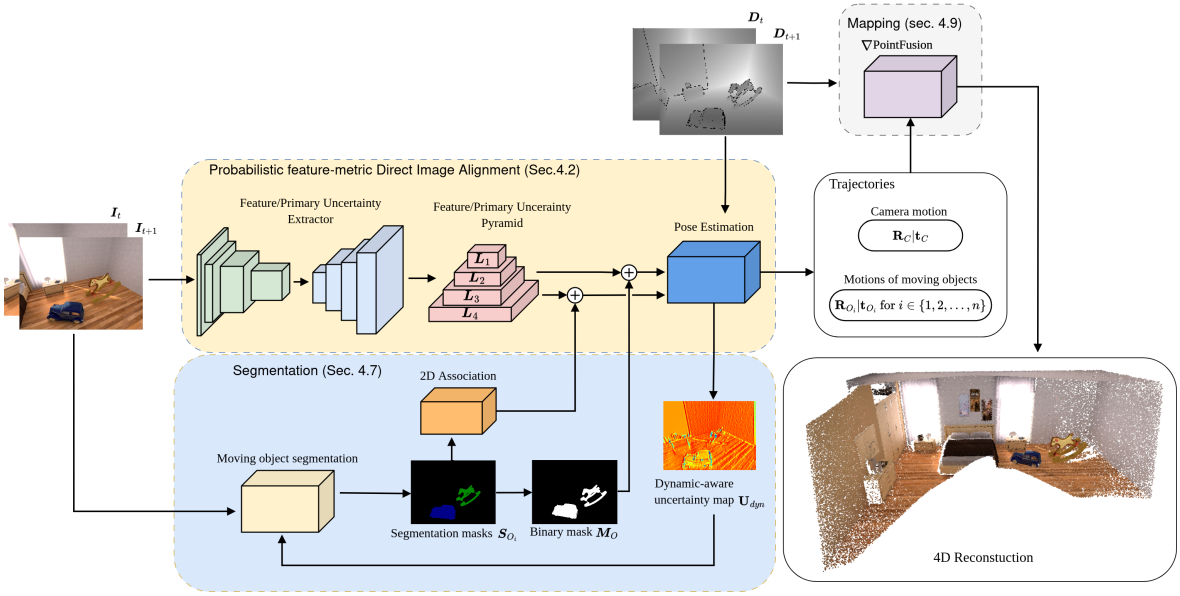


Figure 4.1: System overview. Our system takes consecutive pairs of RGB-D frames as input. We perform camera and object tracking using the probabilistic feature-metric direct image alignment (Section 4.1). To this end, incoming RGB images are fed into the feature/primary uncertainty extractor, which predicts dense high-dimensional feature pyramids and primary uncertainty pyramids for each frame. Then, we minimised the residuals that combine feature-metric and geometric terms using the differentiable Levenberg-Marquardt algorithm and the deep m-estimator in a coarse-to-fine fashion. Meanwhile, the deep m-estimator predicts the secondary dense uncertainties that focus on dynamic regions. We exploit these dynamic-aware uncertainty maps to segment moving objects (Section 4.7). The segmentation masks are applied to track dynamic objects, while the binary mask generated from them is utilized to refine the camera pose estimates. As result, our system returns camera and object trajectories and reconstructs the dynamic environment with time in 3D.

4.1 Direct Image Alignment

The first stage of this project consists of extending ∇SLAM [1] by replacing the ICP with Direct Image Alignment (DIA) in visual odometry. DIA is conventionally defined as a method to estimate the camera poses based on the assumption of intensity consistency of the environment observed in different perspectives, minimizing pixel-wise photometric errors. However, this assumption holds merely for the Lambertian surfaces. Moreover, the intensity map is a highly non-convex function, which results in the optimization often being easily trapped in a local minimum. Thus, we first combine the photometric term with a geometric residual as done in existing works [49][45] to enhance the robustness of the DIA, where the geometric residual implies the consistency of the depth measurement in every co-visible pixels across different frames. This is referred to as the combined DIA from the conventional DIA in the remainder of the thesis distinguished.

Given two RGB-D frames at timestamp t_i and t_j, we use $\mathbf{I}_t$ and $\mathbf{D}_t$ to denote the mapping from pixel coordinates to intensity and depth maps respectively, where the index $t \in \{i, j\}$ indicates the timestamp. The objectiveness here is to find a rigid body transformation $\mathbf{T}_{ij}$ that minimizes the following cost function:

$$E_{track} = \min_{\xi_{ij} \in se(3)} (E_{photo} + \omega E_{geo}) \tag{4.1}$$

This cost function is similar to the tracking mechanisms used in other RGB-D SLAM systems [19]. The key distinction, however, lies in our approach's adaptability to dynamic environments. Our approach tracks multiple instances simultaneously like [49] and [45] .

The photometric term in Equation (4.1) constrains the intensity consistency of pixels across frames:

$$E_{photo} = \sum_{x \in \Omega_j} \|\mathbf{r}_I\|_2^2 \tag{4.2}$$

where $\mathbf{r}_I$ is the photometric residual. It can be mathematically formulated as:

$$r_I = \mathbf{I}_i(\mathbf{x}') - \mathbf{I}_j(\mathbf{x}) \tag{4.3}$$

with coordinates of pixels $\mathbf{x} \in \Omega_j$. Respectively, the warped pixel coordinates $\mathbf{x}'$ can be obtained by applying the warping function described in Equation (3.21).

The geometric term in previous works [45][16][49] is mostly defined as a point-to-plane ICP [58] or even point-to-point ICP. In contrast, we define our geometric term similarly to the photometric residual, replacing the intensity maps $\mathbf{I}$ in Equations (4.3) and (4.2) with depth maps $\mathbf{D}$:

$$E_{geo} = \sum_{x \in \Omega_j} \|\mathbf{r}_D\|_2^2 \tag{4.4}$$

Here, we define the geometric residual $\mathbf{r}_D$ as:

$$r_D = \mathbf{D}_i(\mathbf{x}') - \left[\mathbf{T}_{ij} \cdot \Pi^{-1}\left(\mathbf{x}_j\right)\right]_Z \tag{4.5}$$

The operation $[\cdot]_Z$ is a look-up function which returns the z-component of a coordinate. This geometric error trivially measures the depth difference between inverse projected 3D points of the reference frame and the transformed points from the target frame.

The coefficient ω in Equation (4.1) is a scalar factor to balance the contribution of these two residual terms. This weight scale is set as $\omega = 0.01$. This specific value is selected to

ensure that the individual Chi-square errors have similar magnitude as suggested in [5]. Additionally, this factor is particularly useful when we combine geometric and feature-metric residuals as it will appropriately scale the features and uncertainties to achieve an optimal balance with the geometric term in the training process.

Having defined the cost functions, we calculate the Jacobian necessary for the optimization. The Jacobian of the photometric error w.r.t. the twist $\xi_{ij} \in \mathfrak{se}(3) \in \mathbb{R}^6$ of the rigid body transformation $\mathbf{T}_{ij}$ can be computed using the chain rule:

$$\mathbf{J}_I = \nabla \mathbf{I}^T \, \mathbf{J}_w = \nabla \mathbf{I}^T \, \mathbf{J}_P \, \mathbf{J}_T \tag{4.6}$$

where $\nabla \mathbf{I}$ is the image gradient, which can be expanded as:

$$\nabla \mathbf{I} = \begin{bmatrix} \frac{\partial \mathbf{I}}{\partial x} \\ \frac{\partial \mathbf{I}}{\partial y} \end{bmatrix} \tag{4.7}$$

The remaining terms $\mathbf{J}_w = \mathbf{J}_P \, \mathbf{J}_T$ is described in detail in Section 3.3. Analogously, we can compute Jacobian of the geometric error with

$$\mathbf{J}_D = \nabla \mathbf{D}^T \, \mathbf{J}_P \, \mathbf{J}_T - \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}^T \mathbf{J}_T \tag{4.8}$$

where $\nabla \mathbf{D}$ computes the depth image gradient:

$$\nabla \mathbf{D} = \begin{bmatrix} \frac{\partial \mathbf{D}}{\partial x} \\ \frac{\partial \mathbf{D}}{\partial y} \end{bmatrix} \tag{4.9}$$

The overall residual combining the photometric and geometric terms can be written as:

$$\mathbf{r} = \begin{bmatrix} r_I \\ r_D \end{bmatrix} \in \mathbb{R}^{2 \times 1} \tag{4.10}$$

and respectively the Jacobian is

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_I \\ \mathbf{J}_D \end{bmatrix} \in \mathbb{R}^{2 \times 6} \tag{4.11}$$

for each $\mathbf{x} \in \Omega_j$. To increase the radius of the convergence basin, the optimization is performed from coarse to fine.

As shown in Figure 4.2, the DIA can be formulated in a computational graph. Since warping and interpolation are essentially differentiable, so is the unrolled combined DIA if we applied gradient-based optimization or Gauss-Newton optimization. However, due to the limitations of these optimization methods to different extents, the community prefers using the Levenberg-Marquardt optimization algorithm (LM), which adapts itself with a damping factor between gradient descent and Gauss-Newton. However, the adaptive damping factor is determined at every iteration, making the LM algorithm not differentiable. ∇SLAM and many other works [36][34] introduce multi-layer perceptrons to predict the damping factor based on the computed Jacobian. In this way, we can mitigate the conditional selection schema to decide the damping factor and mimic the decision process in a differentiable fashion instead. The differentiable LM algorithm is introduced later in Section 4.5.

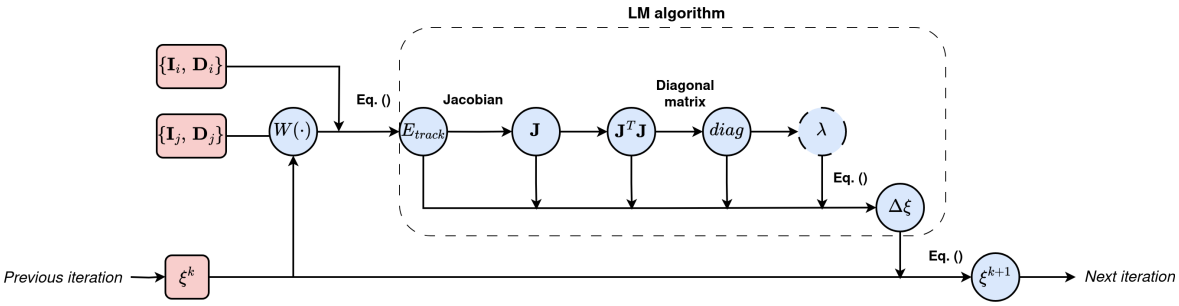


Figure 4.2: A single iteration of the DIA represented in a computational graph. For each update iteration, we warp the previous frame to the current frame using the initial pose or the estimated pose from the last iteration. Jacobians and residuals are computed and leveraged in the Levenberg-Marquardt algorithm as described in Equation (4.21). Since the LM algorithm involves non-differentiable operations in determining the damping factor λ (drawn in a dashed circle), gradient flow in this graph is not possible.

4.2 Probabilistic Feature-metric Direct Image Alignment

Instead of image intensity, the probabilistic feature-metric DIA performs over learned feature representation of the colour images. We utilize the power of a convolutional neural network (CNN) to extract a multi-level hierarchy of features. To enhance their robustness and ensure better generalization across various datasets, we perform L2-normalization over these features along the channels.

Given two images at different timestamp, $\mathbf{I}_i$ and $\mathbf{I}_j$, a CNN extracts a D_l-dimensional feature map $\mathbf{F}^l \in \mathbb{R}^{W_l \times H_l \times D_l}$ at each level $l \in \{1, ..., L\}$ for each of them. Following the spirit of direct alignment, we defined a feature difference by substituting the image intensity in Equation (4.3) with the learned feature-metric representation $\mathbf{F}$:

$$\mathbf{r}_F = \mathbf{F}_i(\mathbf{x}') - \mathbf{F}_j(\mathbf{x}) \tag{4.12}$$

In addition to the feature maps, our CNN can also learn complex visual priors. Therefore, we employ it to further improve the robustness of our method. To this end, we introduce an uncertainty encoder alongside the feature encoder to predict a pixel-wise uncertainty map $\boldsymbol{\sigma}^l \in \mathbb{R}^{H_l \times W_l}$. To combine the uncertainties with the feature-metric representation, we leverage the probabilistic feature-metric residual proposed by Xu et al. in [5]. For each pixel, this residual is defined as an uncertainty-normalised feature difference:

$$\mathbf{r}_{PF} = \frac{\mathbf{r}_F}{\sigma_F} = \frac{\mathbf{F}_i(\mathbf{x}') - \mathbf{F}_j(\mathbf{x})}{\sqrt{\sigma_i^2(\mathbf{x}') + \sigma_j^2(\mathbf{x})}} \tag{4.13}$$

where $\mathbf{x}'$ is the warped pixel coordinates from $\mathbf{x}$. $\mathbf{r}_F$ is the feature difference between the correspondences on the feature map as defined in Equation (4.12) and σ_F is the joint uncertainty combining the individual uncertainties. Equation (4.13) steers the two feature maps to be as similar as possible while downweighing the uncertain features.

Given this probabilistic feature-metric residual, we compute the feature-metric error over all non-occluded pixels in Ω_j as:

$$E_{PF} = \sum_{x \in \Omega_j} \|\mathbf{r}_{PF}\|_2^2 \tag{4.14}$$

This nonlinear least-squares cost combined with the geometric costs described in Equations (4.4) and (4.5) is progressively minimized from an initial pose estimate $\mathbf{T}_0$ using the differentiable LM algorithm in a coarse-to-fine manner.

Existing works have proposed various ways to formulate and apply similar uncertainty maps [9][59]. Our choice is natural since our CNN model is modified based on the one proposed in [5]. However, we do not explicitly compare this formulation to other existing methods. Hence, this design choice may be revisited in future work.

Classic image alignment operates on the image intensity, which is not robust to rapid changes like strongly varying lighting and occlusions caused by moving objects. In contrast, this learned representation, inspired by past works on using deep features for camera tracking, is robust to significant illumination changes. Meanwhile, it provides meaningful gradients for successful alignment with a more extensive and smoother convergence basin. While our approach also addresses the formed nonlinear least-squares problem in a coarse-to-fine manner like the classic method, it diverges from the standard practice of using Gaussian image pyramids. Instead, our feature pyramids are learned hierarchically at multiple levels, allowing us to extract contextual information from a broader receptive field, which enhances the alignment process's effectiveness.

4.3 Inverse Compositional Approach

The DIA, as a special type of the Lukas-Kanade algorithm [7], is conventionally solved using the forward compositional approach. In contrast, we employ the inverse compositional (IC) approach for probabilistic feature-metric DIA, since the IC approach is more efficient compared to the forward approach without sacrificing accuracy in pose estimation [7].

In DIA, an image $\mathbf{I}_i$ transformed using the warp parameters ξ_{ij} is compared with its consecutive frame $\mathbf{I}_j(\mathbf{x})$, also known as the template in image alignment. We write down this transformation explicitly as:

$$\mathbf{I}_i(w(\mathbf{x}, \rho, \xi)) = \mathbf{I}_i\big(\Pi_c(\exp(\xi_{ij}) \cdot \Pi_c^{-1}(\mathbf{x}, \rho))\big) \tag{4.15}$$

and omit the inverse depth ρ within the warping function w in the following to ease the reading.

Minimizing the photometric cost with respect to the warp parameters is a nonlinear optimization task. Therefore, we iteratively solve for the warp parameters $\xi^+ = \xi \circ \Delta\xi$. The warp increment $\Delta\xi$ can be obtained by linearizing the following equation using the first-order Taylor expansion at every iteration

$$\min_{\Delta\xi} \|\mathbf{I}_i(w(\mathbf{x}, \xi + \Delta\xi)) - \mathbf{I}_j(\mathbf{x})\|_2^2 \tag{4.16}$$

$$\min_{\Delta\xi} \|\mathbf{I}_i(w(\mathbf{x}, \xi)) + \underbrace{\frac{\partial \mathbf{I}_i(w(\mathbf{x}, \xi))}{\partial \xi}}_{\text{Steepest descent image}} \Delta\xi - \mathbf{I}_j(\mathbf{x})\|_2^2 \tag{4.17}$$

Then, we can solve the linearized equation using optimization algorithms such as the LM algorithm. However, the steepest descent image here depends on the warp parameters, which are updated iteratively. This means the formulated optimization problem requires repeated computation of the steepest descent image at every iteration.

On the other hand, using the IC algorithm to solve this non-linear optimization problem avoids the re-computing of Jacobain components. Unlike Equation (4.16), we linearize the

objective by applying the warp increments $\Delta\xi$ to the template $\mathbf{I}_j(\mathbf{x})$:

$$\min_{\Delta\xi} \|\mathbf{I}_i(w(\mathbf{x}, \xi)) - \mathbf{I}_j(w(\mathbf{x}, \Delta\xi))\|_2^2 \tag{4.18}$$

$$\min_{\Delta\xi} \|\mathbf{I}_i(w(\mathbf{x}, \xi)) - \mathbf{I}_j(w(\mathbf{x})) - \frac{\partial \mathbf{I}_j(w(\mathbf{x}))}{\partial \xi}\Delta\xi))\|_2^2 \tag{4.19}$$

The iterative update of the warp parameters must be adjusted accordingly using the inverse increments as:

$$\xi^+ = \xi \circ (\Delta\xi)^{-1} \tag{4.20}$$

$$\Delta\xi = (\mathbf{J}^T\mathbf{W}\mathbf{J} + \lambda\,\mathrm{diag}(\mathbf{J}^T\mathbf{W}\mathbf{J}))^{-1}\mathbf{J}^T\mathbf{W}\mathbf{r} \tag{4.21}$$

As the term $\partial \mathbf{I}_j(\mathbf{x})/\partial\xi$ in Equation (4.19) does not depend on ξ, the IC algorithm computes this term beforehand and chains it with other Jacobian components, preventing the re-computation at the iterations. Hence, it is a more efficient algorithm.

We leverage the IC algorithm in our pipeline mainly for its efficiency. Moreover, as the deep probabilistic feature-metric DIA employed in our pipeline results in an identical optimization problem to the classic DIA, integrating it with the IC algorithm requires nearly no extra effort.

4.4 Deep M-Estimator

To further enhance the robustness of the motion tracking in our pipeline, we apply the M-Estimator. The M-Estimator is designed to be more robust to outliers than trivial estimators like least squares. In the context of visual SLAM, they can provide more reliable estimates of a camera's motion and gain better environment reconstruction, even when some sensor readings are inaccurate or noisy. In particular, in a dynamic environment, where moving objects are often distinct sources of outliers in camera motion estimation, the application of the M-Estimator significantly increases the system's performance.

The robust version of DIA adapts the objective function (4.1) with an additional weight matrix $\mathbf{W}$:

$$\underset{\mathbf{T}_{ij}\in\mathcal{SE}(3)}{\arg\min}\ \mathbf{r}^T\mathbf{W}\mathbf{r} \tag{4.22}$$

Accordingly, the optimization using the Levenberg-Marquardt update step is adjusted as follows:

$$(\mathbf{J}^T\mathbf{W}\mathbf{J} + \lambda\,\mathrm{diag}(\mathbf{J}^T\mathbf{W}\mathbf{J}))^{-1}\Delta\xi = \mathbf{J}^T\mathbf{W}\mathbf{r} \tag{4.23}$$

The diagonal weight matrix $\mathbf{W}$ in this equation is contingent on the residual and is selected in accordance with the desired robust loss function ρ. This matrix is integral in determining how the loss function responds to different residual values, in particular, the outliers.

The Huber estimator is one of the most applied m-estimators in visual SLAM [60]. To implement a Huber estimator in a visual SLAM system, one replaces the L2 loss with the Huber loss [61] in the cost terms. The Huber loss is a combination of the mean squared error function, also known as L2 loss, and the absolute value function, L1 loss. It is mathematically formulated as:

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2 & \text{for } |a| \le \delta, \\ \delta(|a| - \frac{1}{2}\delta) & \text{otherwise,} \end{cases} \tag{4.24}$$

This combination is crafted such that it is differentiable, i.e. the derivatives are continuous. However, the parameter δ it uses requires careful tuning to balance between robustness and sensitivity. Moreover, it fully relies on the residuals computed at the previous step and is restricted to a particular error model.

Keeping these limitations in mind, in line with [34], we use a fully convolutional network called deep M-Estimator ϕ_M to learn a weight matrix $\mathbf{W}_\theta$ from not only the residual but also the encoded feature maps of the two input frames:

$$\mathbf{W}_\theta = \phi_M\big(\mathbf{F}_i(\mathbf{x}'_k), \mathbf{F}_j(\mathbf{x}), \mathbf{r}_k\big) \tag{4.25}$$

The subscript k here indicates the k-th update iteration.

The deep M-Estimator enables the unrolled optimization algorithm to effectively filter out non-essential data elements like outliers. Furthermore, using this model prevents us from determining the robust function of a particular error model. Instead, we condition the robust function itself on the input. Thus, this method enhances the algorithm's accuracy by following closely the relevant information.

Although the weight matrix described here has similar functionality as the uncertainty map introduced in Section 4.2, we decouple their contribution by training our model for two stages. Firstly, we use the identity matrix as the weight matrix and train the model to predict primary uncertainties $\boldsymbol{\sigma} \in \mathbb{R}^{H\times W}$. At this stage, we train the model on static sequences only. In this way, we ensure that the uncertainty maps can downweigh features learned from pixels with strong lighting changes and sensory noises. In the second stage, we activate the weights $\mathbf{W}_\theta \in \mathbb{R}^{H\times W}$ of the deep M-Estimator and train on dynamic sequences. Through this, the weight matrix serves as the secondary uncertainty map that encodes uncertainties caused by dynamic pixels, while the primary uncertainty map facilitates the invariance to other outlier sources. Thus, throughout this paper, we also denote the weight matrix $\mathbf{W}_\theta$ as the so-called dynamic-aware uncertainties $\mathbf{U}_{dyn}$ to emphasize its function. Our experiments showed the effectiveness of this two-stage training strategy (see Section 4.10.2).

4.5 Deep Levenberg-Marquardt Algorithm

We optimize the cost function by employing the Levenberg-Marquardt (LM) algorithm. However, the original LM algorithm presents non-differentiable characteristics due to two key challenges:

- The algorithm requires the specification of a termination condition. The iterative optimization terminates either when the cost is below the predefined convergence threshold or when a particular number of iterations is reached. This condition control flow makes the algorithm's output non-differentiable with respect to the input.

- The algorithm updates the damping factor λ in Equation (4.21) according to the currently computed cost in each iteration. It increases λ if the cost does not decline after a step; otherwise, it reduces this factor. This way, the LM algorithm adapts the optimization steps between Gauss-Newton optimization and gradient descent. However, this if-else-based schema prevents the differentiability of the algorithm.

When the solution of the LM algorithm is non-differentiable with respect to the input feature maps, it becomes impossible to learn feature representation through back-propagation. As suggested in previous works [1][36], we can fixate the number of iterations and omit the convergence threshold to addressee the first non-differentiable property. This is considered

as incomplete optimization. LM algorithm is known for possessing fast convergence speed regarding iteration numbers. Hence, using sufficient iterations, usually a small value, shows no obvious disadvantages in most cases but reduces memory usage instead.

The second difficulty has been tackled differently in past works. Jatavallabhula et al. proposed a soft reparameterization of the damping mechanism. They defined smooth gating functions based on the logistic function to approximate the damping factor in each iteration. This approximation enables differentiability in the LM algorithm. However, it offers no learning capability, limiting its scalability. Conversely, other works [36][34] propose to predict the damping coefficient using neural networks.

To address these challenges, we adapt the fully connected neural network introduced in [34]. The network predicts the damping factor λ_θ based on the matrices $\mathbf{J}^T\mathbf{W}\mathbf{J}$ and $\mathbf{J}^T\mathbf{W}\mathbf{r}$:

$$\lambda_\theta = \phi_{LM}\left(\mathbf{J}^T\mathbf{W}\mathbf{J},\, \mathbf{J}^T\mathbf{W}\mathbf{r}\right) \tag{4.26}$$

This concept allows the output of the LM update process to carry crucial information about the predicted damping parameter.

4.6 Pose Initialization

A relatively accurate pose initialization is crucial for DIA. Though leveraging the Gaussian image pyramids can enlarge the convergence basin of frame-to-frame tracking, these coarse-to-fine optimization methods are particularly sensitive to results estimated at coarser levels. Poor initial estimates propagated down to finer levels are often the failure source of a SLAM pipeline, leading to long convergence time or even divergence.

Traditional methods use an identity pose or assume a constant motion model for initialization. For instance, in [62] Gladkova et al. suggests a set of pose candidates based on assumptions such as no motion and constant motion. They choose the candidate with the smallest cost as the valid initialization. These assumptions, however, hold only in strictly limited conditions like slow motion and small baseline. For example, when they are not valid in wide baseline settings, an iterative optimization process can become trapped in an incorrect local minimum. To tackle this issue, as suggested in past works [5] [59], we train a convolutional neural network to provide an initial pose for the direct alignment.

Our network has a compact and effective design (see Section 4.10). It takes concatenated outputs from the coarsest-level view encoder of two frames as input and offers an effective starting point for the optimization process at the coarsest level:

$$\xi_0 = \phi_T\left(\{\mathbf{W}_i^1, \mathbf{W}_j^1\}\right) \tag{4.27}$$

Several works [34][5] suggest generating the final prediction as the weighted average of a set of hypothetical poses. However, no ablation studies in their work are provided to show the effectiveness of this design. Unlike them, we maintain a compact network design by making the final prediction, comprising a 3D Euler angle and a 3D translation vector directly.

4.7 Moving Object Segmentation

The dynamic regions, as a crucial factor that affects the accuracy as well as the robustness of a visual SLAM system in dynamic environments, are handled in our pipeline using segmentation like in other works such as [49], [48], [16], and [45]. In this thesis, we proposed

two alternatives to segment the moving instances. The first approach follows the idea of [45], combining the prior segmentation from Mask R-CNN [10] and a refinement step using geometric cues (Section 4.7.1). The other approach employs the promotable properties of the up-to-date work in semantic segmentation, FastSAM [11], leveraging the advantages of the dynamic-aware uncertainty map predicted by the deep M-Estimator at the previous stage (Section 4.4) to provide promising segmentation of moving instances (Section 4.7.2).

4.7.1 Mask R-CNN Refined by Geometric Segmentation

As suggested in previous works [49] [45] [16], we utilize the top-performing instance segmentation network Mask R-CNN [10] to detect and segment objects of classes which are likely dynamic such as people. Mask R-CNN is a convolutional neural network (CNN) developed on the top of the two-stage detector, Faster R-CNN [63]. It works by designing a new branch, in addition to the classification and regression branches of Faster R-CNN, to predict the object masks.

More recently, Li et al. have rolled out an improved Mask R-CNN model with a standard vision transformer (ViT) as the backbone in [64], which has potentially superior scaling properties. However, in our experiments, the latest model showed no distinct improvement in the segmentation results but had noticeably slower inference than the original model, evidencing a general limitation of the ViT-based methods. Therefore, we chose the pertrained model of the original Mask R-CNN to ensure that our system can predict instance segmentation in a reasonable time range. Examples of instance segmentation using Mask R-CNN on different datasets are shown in Figure 4.3.

Our approach uses the results of Mask R-CNN to provide prior information on dynamic entities in the observed environment. We run the segmentation on a limited set of labels to segment only objects of chosen classes that are naively considered motion initiators, such as humans, animals, cars, etc.

(a) Segmentation result of *TUM RGB-D Dataset* **(b)** Segmentation result of *Bonn Dynamic RGB-D Dataset*

Figure 4.3: Segmentation results using Mask R-CNN with class labels of interest. On the one hand, persons in scenes, even only partially visible, can be segmented by Mask R-CNN with few failures, while moving objects that are not included in the COCO dataset [65] such as the balloon in (a) can not be detected. On the other hand, although "chair" is one of the labels in COCO, it is not essential an initiator since its movement can only triggered by creatures. That said, we assume it is stationary at first and classify whether it is moving in the following stage using geometric cues. This way, we maintain as much information as possible for camera tracking.

The segmentation results of Mask R-CNN can only bring limited benefit to our pipeline if

no improvement strategies are applied. On the one hand, mask R-CNN excels in object detection yet has a notifiable limitation that restricts it from being directly used in our pipeline. Similar to other up-to-date semantic segmentation methods, it often yields imprecise (over-smoothed) object boundaries (see Figure 4.3). Moreover, as a supervised learning method, it can only predict categories in the training dataset. For instance, it cannot detect the balloon in Figure 4.4b. This characteristic naturally bounds the generalization of the model in the open-world setting.

On the other hand, we select merely a set of interests, which is essentially motion initiator, to balance the speed and accuracy. The selection further narrows down the categories that the model needs to segment. However, a moving object in reality is often triggered into motion by other active objects. With the predefined labels, Mask R-CNN cannot detect these passively moving objects.

Considering this limitation, to improve the segmentation results of Mask R-CNN, Rünz et al. concluded that informative RGB-D data allowing for over-segmentation of the image can be used to refine the imprecise object boundaries generated by semantic segmentation models [45]. They propose a geometric segmentation approach, studying the geometric cues hidden in input depth data. This approach is adjusted and employed in our work.

According to [45], an edginess map is generated by investigating the depth discontinuity ϕ_d and surface concavity ϕ_c. The depth discontinuity term ϕ_d is defined as:

$$\phi_d = \max_{i \in \mathcal{N}} |(\mathbf{v}_i - \mathbf{v}) \cdot \mathbf{n}| \tag{4.28}$$

Conceptually, computing the gradient of the depth map can also extract contours of discontinuity depth, resulting in a comparable edginess map as Equation (4.28). Additionally, the concavity term ϕ_c is formulated as:

$$\phi_c = \max_{i \in \mathcal{N}} \begin{cases} 0 & \text{if } (\mathbf{v}_i - \mathbf{v}) \cdot \mathbf{n} < 0 \\ 1 - (\mathbf{n}_i \cdot \mathbf{n}) & \text{otherwise} \end{cases} \tag{4.29}$$

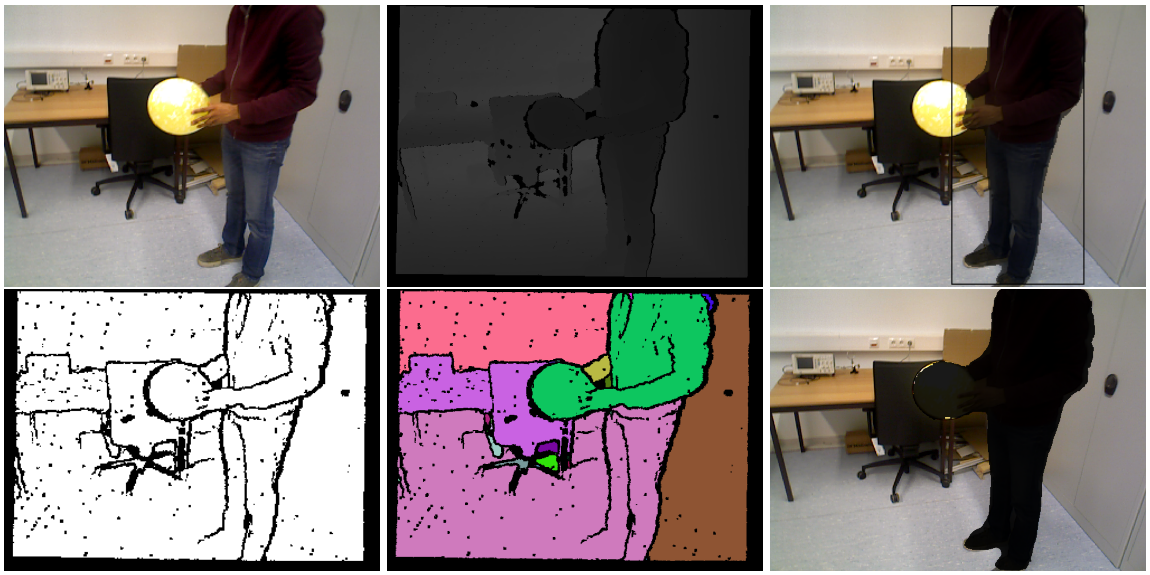
In the above two equations, $\mathbf{v}$ represents vertices, while $\mathbf{n}$ denotes normals. Vertex maps are typically obtained by back-projecting the depth map $\mathbf{D}$ into 3D space. Subsequently, we can compute normals from the vertex map. The subscript i indicates that the values are computed on a neighbouring pixel within a local neighbourhood $\mathcal{N}$. In particular, this approach defines a pixel to be on an edge when

$$\phi_d + \kappa \phi_c > \theta \tag{4.30}$$

where θ is a threshold, and κ is the relative weight to balance the two terms. Eventually, the image is segmented by running a connected-components analysis (CCA) algorithm on the edginess map. This method captures the intuition that the image area usually portrays geometric edges 1. when there is either a manifest change in depth or 2. where the surface bends sharply, often concave regions.

Nonetheless, this approach struggles to distinguish physically connecting objects. For instance, a walking person can not be segmented from the ground as his feet touch the floor (details can be found in Section 6.2.2). Mitigating this requires careful adjustment of the weighting factor, which balances the contribution of the concavity of objects and discontinuity of depth. However, tuning this coefficient is not trivial and is very case-sensitive.

Attempting to address this problem, we experiment with various methods and outline our insights and the results in Section 6.2.2. However, based on extensive experiments, we found that despite the disadvantages mentioned above, this geometric segmentation method



(a) Example from *Bonn RGB-D Dynamic Dataset*

(b) Example from *TUM RGB-D Dataset*

Figure 4.4: Segmentation results combining Mask R-CNN and geometric segmentation. For each example, images shown from left to right from up to down represent 1. RGB input; 2. depth input; 3. results of Mask R-CNN with specified classes of interest; 4. geometric edges; 5. geometric segmentation, and 6. final segmentation. These two examples demonstrate that our method can identify and segment passively moving objects, such as the balloon in the first example and the chair in the second. However, the chair is only partially segmented, which reflects one of the limitations of this method.

provides stable results in most cases. Moreover, we can account for objects subject to external forces in the fused segmentation by adjusting the merging mechanism introduced in [45].

To merge the geometric and instance segmentation, Rünz et al. performs geometric segmentation for each frame, while instance segmentation using Mask R-CNN is applied asynchronously for nondeterministic frames. According to their two-stage segmentation merging schema, they first map the geometric segmentation $M_g = \{C_0, C_1, ..., C_n\}$ to the existing semantic segmentation $M_s = \{L_0, L_1, ..., L_n\}$ by identifying the maximum overlap which should

be greater than the threshold φ. Formally, a component $C_i \in M_g$ with the number of pixels denoted as $|C_i|$ is considered as a candidate to be mapped to a component $L_k \in M_s$ if:

$$\frac{|C_i \cap L_k|}{|C_i|} \geq \varphi \tag{4.31}$$

After that, they assign each component similarly with object labels generated by rendering all existing models. This step requires a very accurate tracking of the camera and objects. We also note that Rünz et al. selected a broader set of interesting categories than us, including not solely the motion initiators. In our case, we must also generate and maintain the masks of externally manipulated objects. Therefore, Directly integrating this schema into our pipeline is unrealistic.

Instead, we simplify the mapping from masks to models by warping the merged results from the previous step using estimated camera motion. We first apply Mask R-CNN for every incoming frame to predict prior segmentation and fuse it with the geometric segmentation using Equation (4.31). Then, the predicted masks from the previous frame are propagated to the current frame using the warping function. By computing the overlap of the newly generated and the propagated masks, we generate temporally consistent masks of the passively moving instances. Unlike [45], we perform the segmentation merging for every incoming frame. Besides, we select $\varphi = 45\%$ to ensure larger moved objects are merged as well. Figure 4.4) shows some example results, while more examples and a discussion about the limitations of this method are detailed in Section 6.2.2. Eventually, the generated segmentation is utilized as binary masks to eliminate the dynamic pixels that tend to cause failure in tracking the camera. With this approach, we do not explicitly track the objects, i.e., our pipeline using this approach is capable of achieving accurate and robust camera motion estimation in dynamic environments, but it lacks the ability to track objects. Thus, we introduce the second segmentation approach in the following section, with which our pipeline can track both the camera and objects.

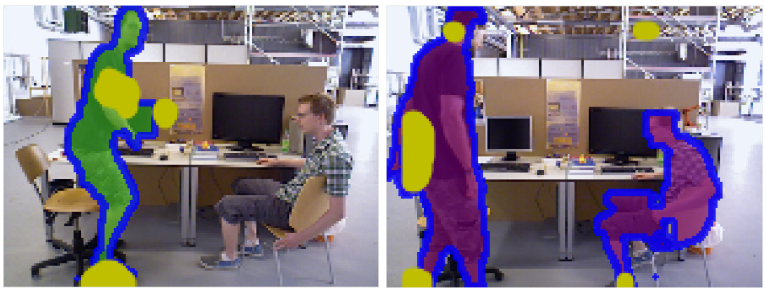
4.7.2 Uncertainty Prompted Moving Object Segmentation

Our visual odometry model predicts two pixel-wise uncertainty maps for every incoming frame. Thanks to our two-stage training, we can decouple the influence of two types of factors, i.e., the significantly varying illumination and the dynamic pixels, encoding them with two individual uncertainty maps. The one that encodes the motion captured by pixels offers informative data, which empowers the development of our second moving instance segmentation approach.

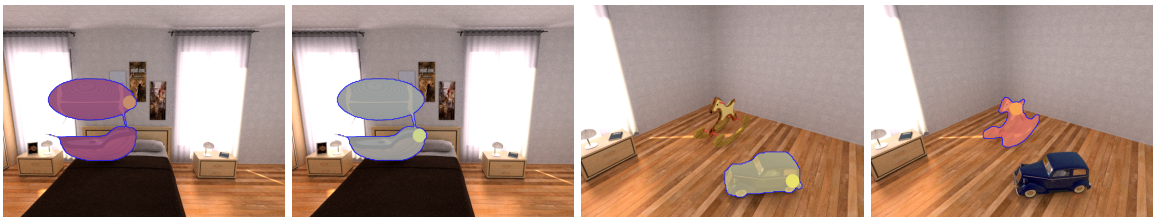
Pixels encompassing motion elements usually cannot hold the photometric as well as geometric consistency across frames, becoming one of the most frequent failure sources in tracking. Our CNN model predicts promising uncertainty maps which identify regions that possibly capture moving entities, assigning them with higher uncertainty - lower weights. This capability coincides with the intent to move instance segmentation. Therefore, we propose to exploit a novel moving instance segmentation approach leveraging the learned uncertainty maps.

Alongside these uncertainty maps, we leverage the prompt features of a recently rolled out generic segmentation approach, FastSAM [11]. This approach provides the opportunity to generate segmentation of interests using prompt inputs such as points, bounding boxes, and segmentation masks predicted by the model itself (Note that until the completion of this

thesis, the feature of using masks inferred by other models such as Mask R-CNN is still under development). To this end, we have to convert the uncertainty maps to a type of available prompts for which we choose to use points.



(a) Segmentation results using all points at once for images from *TUM RGB-D*



(b) Segmentation results combining individual uncertainty prompt points with background prompts for images from *Co-Fusion*

Figure 4.5: Segmentation results using uncertainty-prompted moving object segmentation. The upper examples show segmentation results using all uncertain points (coloured yellow) to prompt FastSAM [11]. This way, instances are grouped in one mask, which can be used to eliminate dynamic regions, enhancing the robustness and accuracy of camera tracking. The lower examples are generated by inputting a single uncertain point and background points to prompt the segmentation. The results can be used to identify individual moving instances. Thus, in addition to improving camera pose estimates, they are useful for object tracking.

Pixels depicting dynamic elements are assigned with distinctly lower certainty values than the static regions. However, the uncertainty maps are not perfect. They appear with noises, possibly due to illumination changes not fully covered by the primary uncertainty encoder. Thus, we apply a chain of average and max pooling to extract representative information from the maps. The pixel coordinates extracted in this way are often more reliable than raw, noisy uncertainty maps. Moreover, using pooling to sample points preserves the differentibility of our pipeline.

Figure 4.5 conveys the segmentation results generated by this approach. As it shows, if we feed all points to FastSAM at once, the model segments all pixels of interest using a single label and cannot distinguish between instances. To this end, one can potentially cluster the sample points and provide FastSAM with the grouped prompts set by set to produce instance segmentation. Instead of clustering, we choose to maintain the number of points at n_{FG} by controlling the kernel size and the number of pooling operations. These points are used to prompt the foreground object. In addition, we extract n_{BG} points with the highest certainty to indicate the background. Then, together with the background points, the foreground points are sent one by one to FastSAM. The model returns segmentation masks of identical numbers as the foreground points. We employ the same merge technique described in Equation (4.31) to fuse masks with large overlaps and obtain the ultimate set of instance segmentation masks $\mathcal{S} = \{\mathbf{S}_1, \mathbf{S}_2, ..., \mathbf{S}_n\}$ with $n \leq n_{FG}$.

The first two visualizations in Figure 4.5b exemplify one of the main advantages of employing FastSAM. Note that the object prompt points in these two examples are located on different parts of the airship. The connection between the upper and lower parts of the air-

ship is visually hard to identify. However, FastSAM can provide us with segmentation of the whole object correctly without further requirements. The inherently top performance of this state-of-the-art segmentation method mitigates post-processing, making our pipeline more concise, although this capability is traded with a long inference time, which is discussed in Section 6.2.5.

4.8 Dynamic Visual Odometry

In reality, moving objects can be either rigid or non-rigid. Since the starting points for solving them are often different, they are both challenging problems in themselves. Note that we refer to non-articulated rigid instances by rigid bodies and consider articulated bodies as non-rigid. We propose two approaches to handle non-rigid and rigid bodies to provide adapted solutions to dynamic objects. For non-rigid bodies, we eliminate them using segmentation masks to perform robust camera motion estimation based solely on the remaining static pixels. We also estimate the camera poses using the filtered data for rigid bodies. Meanwhile, we track the object's motion using the adopted feature-metric direct image alignment introduced in this section.

4.8.1 Camera tracking

Unlike visual odometry designed for static environments, which performs camera tracking using all non-occluded pixels, dynamic visual odometry usually omits the pixels capturing moving instances. Following this idea, we eliminate the dynamic pixels using the segmentation masks generated in the previous step. Given a union set of the segmentation masks $\mathbf{M}_O$ which indicates all pixels in motion, for camera tracking, we minimize the combined error:

$$\underset{\xi_{C_iC_j}\in se(3)}{\operatorname{argmin}} \sum_{\mathbf{x}\in(\Omega_j\setminus\mathbf{M}_O)} \left(\mathbf{r}_{PF}^T\mathbf{W}_{PF}\mathbf{r}_{PF}+\mathbf{r}_D^T\mathbf{W}_D\mathbf{r}_D\right) \tag{4.32}$$

where $\mathbf{r}_{PF}$ and $\mathbf{r}_D$ are the probabilistic feature-metric and geometric residual, respectively. We solve for the pose by iteratively optimizing this combined error, leveraging the differentiable LM algorithm over the feature pyramids.

4.8.2 Moving Object Tracking

Like camera tracking, we track identified objects from frame to frame and estimate their trajectories based on the probabilistic feature-metric DIA introduced in 4.2. Note that we mainly focus on tracking the objects in this work, for which we assume the initial relative poses between objects and camera denoted as $\mathbf{T}_{C_rO_r}$ are given. We formulate the residuals identically to (4.1), but only considering the pixels within an estimated segmentation mask $\mathbf{S}_i \in \mathcal{S}$. The transformation of the object O_i w.r.t. to the local coordinate frame fixed on the reference camera C_r is calculated by minimizing the cost function:

$$E_{O_i}(\xi_{O_rO_l}^{C_r}) = \sum_{\mathbf{x}\in\mathcal{S}_i} \left(\mathbf{r}_{PF}^T\mathbf{W}_{PF}\mathbf{r}_{PF}+\mathbf{r}_D^T\mathbf{W}_D\mathbf{r}_D\right) \tag{4.33}$$

where the feature-metric residual is formulated as:

$$\mathbf{r}_{PF}(\xi_{O_rO_l}^{C_r}) = \frac{\mathbf{F}_l(\mathbf{x}')-\mathbf{F}_r(\mathbf{x})}{\sqrt{\boldsymbol{\sigma}_l(\mathbf{x}')^2+\boldsymbol{\sigma}_r(\mathbf{x})^2}} \tag{4.34}$$

and the geometric residual as:

$$\mathbf{r}_D(\xi^{C_r}_{O_rO_l}) = \left[\left(\Pi_c(\exp(\xi^{C_r}_{O_rO_l}) \cdot \Pi_c^{-1}(\mathbf{x}, \rho))\right)\right]_Z - \mathbf{D}_r\left(\mathbf{x}'\right) \tag{4.35}$$

with the warped pixel coordinate $\mathbf{x}'$ using the warping function. We optimise the above cost function using the IC approach (Section 4.3) with a differentiable Levenberg-Marquardt method (Section 4.5) in a coarse-to-fine manner. Additionally, $\xi^{C_r}_{O_rO_l}$ is initialized by the Pose-Net which is discussed in 4.6. Since objects of small size or at further distance occupy minor image space, pyramidal down-sampling with fixed levels might significantly impact the alignment success due to potential over-smoothed pixels, i.e., information loss, for small objects on the coarse levels. To overcome this, inspired by DirectTracker [62], we adapt the scaling levels based on the size of objects online, while the downsample scales remain the same as for camera tracking.

4.8.3 2D Association

In the association step of our process, we use the estimated relative motion of objects to align the segmentation mask from the current frame with the previous frame. We employ the Hungarian algorithm [66] to establish one-to-one correspondences between objects across these frames.

As the base for matching, we compute the distance matrix $\mathbf{D}$ using the 2D Intersection over Union (IoU) metric (see Section A for detailed definition). Conventionally, a correspondence between objects is considered valid and accepted if the similarity score, as measured by the IoU metric, exceeds a predefined minimum threshold. However, the association violates differentiability in this way.

To this end, we employ the Hungarian Network (Hnet) [67]. Hnet is a differentiable version of the Hungarian algorithm [66] using a deep neural network shown in 4.6. It estimates the association matrix $\tilde{\mathbf{A}}$, given the distance matrix $\mathbf{D}$ as input. These two matrices are of identical dimensions.

This differentiable implementation allows it to be integrated with other deep-learning tasks that require permutation invariance in end-to-end training, which makes it a good fit in our pipeline. Besides, this implementation is simpler yet more efficient than the existing alternative [68].

Once matches are established, the segmentation masks from the current frame that successfully align with those from the previous frame are assigned a tracking ID corresponding to their matched counterpart. Their tracklets, or the sequences of observed positions over time, are then integrated into the corresponding object trajectories.

In classic approaches like [62], in cases where an object in the current frame does not find a match in the previous frame, they investigate whether it has been consistently tracked in 3D in earlier sequences. If this is the case, the object and its corresponding warped mask are retained in memory. This handles scenarios where the input segmentation masks lack temporal consistency, ensuring that objects are accurately tracked despite inconsistent segmentation across frames.

However, we do not tackle this issue with the above approach due to the concern about breaking the pipeline's differentiability. Instead, we consider any objects in the current frame that cannot be assigned with an existing ID in the previous frame as new objects. Consequently, our method highly relies on the temporal consistency of segmentation, which reflects a key limitation of our pipeline.

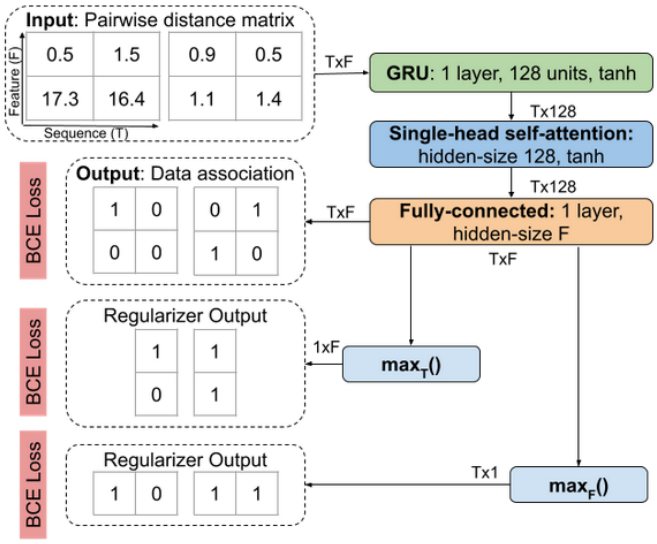


Figure 4.6: Block diagram of Hungarian Network [67]. This architecture comprises a gated recurrent unit (GRU), a single-head self-attention network [69], and a fully connected network. The GRU decouples the information from the pairwise input distance matrix along the temporal and feature-wise dimensions. Its output is fed into the self-attention network, which estimates the correct associations at each time step. The final fully connected network estimates the association matrix $\tilde{\mathbf{A}}$ using a Sigmoid activation function. The association is addressed as a multi-class, multi-label classification task.

4.9 Mapping with Differentiable PointFusion

In addition to localization, the other core task in SLAM is mapping, for which we adopt the differentiable PointFusion approach proposed in ∇SLAM [1] and reconstruct the scene with moving components using surfels. As this is not part of our contribution, we overview the approach here and refer the readers to the original paper [1] for technical details.

Differentiable PointFusion (∇PointFusion) addresses the differentiable but non-smooth operations in conventional dense mapping and fusion, such as clipping, indexing, thresholding, and decision-making, by leveraging locally smooth functions and soft association. It allows us to incrementally fuse redundant observations of the same map element and maintain the final reconstructed map in a manageable size.

Note that although the original implementation of PointFusion [19] supports dynamic scenes, segmenting dynamic objects and updating reconstruction even of moving objects, this function is not realized in ∇PointFusion. The ability of the visual odometry in our pipeline to detect and track moving objects equips ∇PointFusion with the capability of reconstructing dynamic environments.

4.10 Implementation Details

The full system is implemented in Python by extending ∇SLAM [1]. We developed the differentiable front-end by adapting and reusing several components from [5] and [34]. The differentiability of the system is guaranteed by taking advantage of the auto-diff properties of the PyTorch framework [70]. There are 16M learnable parameters in our implementation. The architectures of each component and the training setup are outlined in the following

subsections.

4.10.1 Architectures

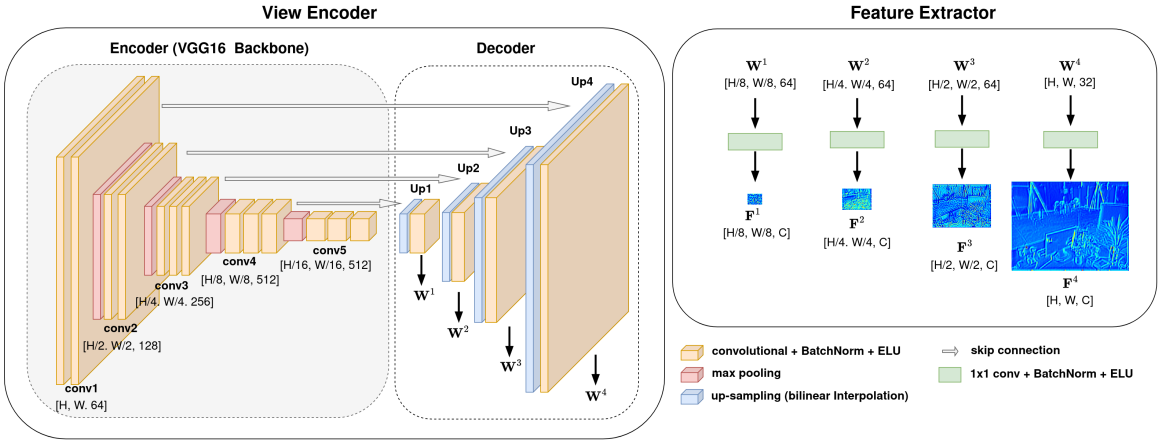


Figure 4.7: View encoder and feature extractor. The view encoder on the left side is constructed in U-Net architecture [71], consisting of an encoder and a decoder. The VGG16 backbone [72] is used in the encoder with additional batch normalization layers between the convolutional layers and activation function. The decoder is built by chaining up-sampling layers (blue) and basic convolutional blocks (yellow), which groups a convolutional layer, a batch normalization layer, and an ELU layer. The results of the decoder $\mathbf{W}^l$ at each level $l \in \{1,2,3,4\}$ are sent to the feature extractor, where the ultimate feature maps $\mathbf{F}^l$ used for direct alignment are generated. The feature extractor on the right side is an adaptation block (green) grouped by a 1×1 convolutional layer, a batch normalization layer, and an ELU layer. Dimension of the resulting intermediate and final feature maps are represented as [Height, Width, Channel].

Figure 4.7 shows the architecture of the view encoder and feature extractor. Within a U-Net-like architecture, the encoder extracts progressively richer semantic information and a larger spatial context of the image with decreasing resolution. The subsequent decoder adapts feature maps for alignments, restoring the resolution and maintaining low-level information with skip connections.

The view encoder encodes the input RGB images, extracting low-lever features $\mathbf{W}^l$ at pyramidal levels $l \in \{1,2,3,4\}$, which are utilized in the following feature extractor, uncertainty extractor, and initialization pose network. Lv et al. and Xu et al. proposed to use concatenated RGB-D data of two frames as input of the network for view representation learning [34][5]. However, we believe this manner offers merely duplicated information to the network and yet obtains restricted benefits. Besides, using concatenated RGB-D images as input may not be a good design choice, as the depth images from realistic data possess boundaries of zero values (see Figure 5.2 in Section 5.1). Thus, our view encoder only requires RGB images of the current frame as input. Similar to [8], the design of the view encoder is based on the U-Net architecture [71]; it is constructed with VGG16 backbone [72] and a 4-level pyramid using bilinear up-sampling in the decoder. The intermediary outputs of the decoder, $\mathbf{W}_l$, are fed into the feature extractor, which predicts 16-dimensional feature maps. Moreover, $\mathbf{W}_l$ are sent to an uncertainty encoder to estimate scalar pixel-wise uncertainty maps. Its architecture is shown in figure 4.8.

The architecture of our initialization pose network is depicted in Figure 4.9, which is adapted from [5]. It predicts an initial pose on the coarsest level of the coarse-to-fine optimization. Its input is a concatenation of the outputs of the two frames from the view encoder.

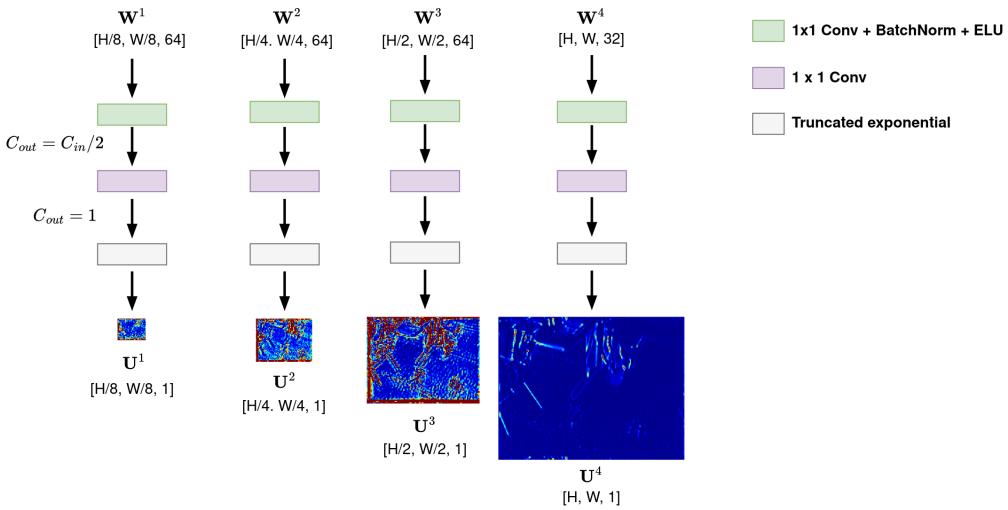


Figure 4.8: Uncertainty Encoder. The uncertainty encoder takes the results on each pyramid level of the decoder, $\mathbf{W}^l$, and predicts the scalar pixel-wise uncertainty maps, $\mathbf{U}^l \in \mathbb{R}^{H^l \times W^l}$. Each pyramid level $l \in \{1,2,3,4\}$ is an adaptation block (green) followed by a 1×1 convolutional layer and a truncated exponential layer.

In contrast to [5], our pose net returns a single prediction rather than a weighted sum of multiple hypotheses. In this way, we save the effort of choosing the number of hypotheses without observing any distinct declines in performance. The predicted pose is parameterized with 3 Euler angles and a 3-dimensional translation vector.

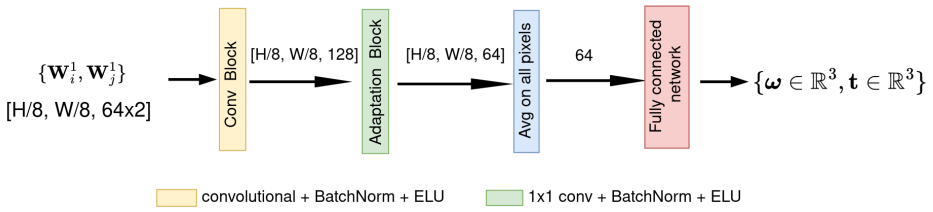


Figure 4.9: Initialization pose network. The initialization Pose-Net takes the concatenated coarsest view-features $\{\mathbf{W}_i^1, \mathbf{W}_j^1\}$ from the decoder and predicts an initial pose, which is parameterized as 3 Euler angles and a 3D translation vector. It consists of a basic convolutional block (yellow) and an adaptation block (green) followed by an average operation over all pixels (blue) and a fully connected network (red). Dimension of the resulting intermediate feature maps are represented as [Height, Width, Channel].

4.10.2 Training

The model is trained in a supervised fashion using RGB data from the *TUM RGB-D* [73] and the *Bonn Dynamic RGB-D* [74] datasets. The initial pose estimate from the pose network and the estimated pose $\hat{\mathbf{T}_{ij}} = \exp(\hat{\boldsymbol{\xi}_{ij}})$ are compared to the ground truth pose $\mathbf{T}_{ij} = \exp(\boldsymbol{\xi}_{ij})$. For this purpose, we use the 3D End-Point-Error (EPE) as suggested in [5] to supervise the training, which is mathematically defined as:

$$L = \frac{1}{|\mathcal{P}|} \sum_{i \in \mathcal{L}} \sum_{\mathbf{p}_j \in \mathcal{P}} \| \exp(\hat{\boldsymbol{\xi}_{ij}}) \mathbf{p}_j - \exp(\boldsymbol{\xi}_{ij}) \mathbf{p}_B \|^2 \tag{4.36}$$

It measures the Euclidean distance between back-projected 3D points in the frame $\mathcal{F}_j$ transformed to the frame $\mathcal{F}_i$ using the ground truth and the estimated transformation, respectively.

$\mathcal{L} = \{1, 2, 3, 4\}$ represents the pyramid levels and $\exp(\hat{\xi}_{ij})$ is the pose estimate after iterative update using the LM algorithm on every pyramid level. This loss allows the resulting gradients in every component of the visual odometry to back-propagate to the input sensory data and update all the learning weights. Moreover, 3D EPE can balance the translation and rotation with a single loss compared to combining the translation and rotation relative pose error. The two uncertainties introduced in Section 4.2 and in Section 4.4 are learned end-to-end and implicitly constrained by the 3D EPE.

We train the model from scratch using ADAM [75] and learning rate decay. The learning rate is initialized at 0.0005 and gradually reduced every five epochs until the 20th epoch. All neural networks in the model are trained together from the *TUM RGB-D* the *Bonn RGB-D Dynamic* datasets in two separate stages. In the first stage, we train the model using only sequences of static scenarios in datasets. Subsequently, the dynamic-aware uncertainties $\mathbf{U}_{dyn}$ are trained in particular on the dynamic sequences in the second stage.

For each stage, the data are split into three subsets, including a train set (80%), a validation set (10%), and a test set (10%). We select several sequences entirely from the datasets for testing, while the remaining sequences are split into training (last 90% of each sequence) and validation (first 10%) sets. The datasets' splitting is detailed in Table 4.1.

Our splitting schema is modified from [5], where they inversely take the first 95% for training and the rest for validation. We've observed that the last part of the sequences mainly consists of slower motions than the former. This results in the validation set being much easier than the training set. Consequently, whether the model is over-fitting becomes tricky to determine. Our splitting schema can mitigate this issue as the first 10% of the sequences that we used for validation consists of a richer pattern of motions than those at the end.

Dataset		TUM RGB-D [73]		Bonn RGB-D Dynamic [74]	
Split	Setup	Sequences	#frames	Sequences	#frames
Train-validation	static	fr1_desk2, fr1_floor, fr1_room, fr1_xyz, fr1_rpy, fr1_plant, fr1_teddy, fr2_360_hemisphere, fr2_large_no_loop, fr2_large_with_loop, fr2_pioneer_slam, fr2_pioneer_slam2, fr2_pioneer_slam3, fr2_xyz, fr2_rpy, fr2_dishes, fr2_360_kidnap, fr2_desk_with_person, fr3_long_office_household	Train: 31053 Val: 3856	static, static_close_far	Train: 10407 Val: 1298
	dynamic	fr3_walking_halfsphere, fr3_walking_rpy, fr3_sitting_rpy, fr3_sitting_static, fr3_sitting_xyz]	Train: 5802 Val: 719	crowd, kidnapping_box2, moving_nonobstructing_box, placing_obstructing_box, balloon, placing_nonobstructing_box3, moving_nonobstructing_box2, balloon_tracking, moving_obstructing_box, removing_nonobstructing_box, placing_nonobstructing_box2, synchronous, removing_obstructing_box, crowd2, synchronous2, removing_nonobstructing_box2, moving_obstructing_box2, kidnapping_box, person_tracking	Train: 12617 Val: 1551
Test		fr1_360, fr1_desk, fr2_desk, fr2_pioneer_360, fr3_walking_static, fr3_walking_xyz	5889	person_tracking2, crowd3, placing_nonobstructing_box, balloon2, balloon_tracking2	3030

Table 4.1: Dataset splits. We train and test our model using sequences from *TUM RGB-D* [73] and *Bonn RGB-D Dynamic* [74]. We split the data into three subsets, train, val, and test sets, with the ratios of approximately 80%, 10%, and 10%. We selected several sequences for testing, while the remaining sequences were used to construct the train and validation sets. The first 10% of frames of sequences in the train-validation split is used as validation, and the rest are applied for training.

Chapter 5

Evaluation

We evaluated the proposed method quantitatively using synthetic and real sequences with ground truth data from various datasets. In this chapter, we present the evaluation results after introducing the datasets and metrics used in our experiments.

5.1 Datasets

To validate the effectiveness of the proposed system, we select two synthetic and two realistic datasets, including pure static scenes and dynamic scenes captured by static and moving cameras. The dataset we select includes both RGB and depth images. Among them, the synthetic datasets can provide us with well-synchronized colour and depth image pairs, while we must associate them with each other for realistic datasets. The deep-learning components are trained using partitions of the *TUM RGB-D* [73] dataset and the *Bonn RGB-D Dynamic* [74] dataset. Thus, we merely evaluate the remaining test splits of these two datasets. Since we do not in particular target the tracking of non-rigid objects like humans in scenes, for the dynamic sequences in the realistic datasets that involve non-rigid moving objects, we eliminate them using segmentation masks to improve the robustness of the camera motion estimation relying on the extracted static background of the scene.

5.1.1 ICL-NUIM Dataset

(a) Noise-free RGB **(b)** Noise-free depth **(c)** RGB with noise **(d)** Depth with noise

Figure 5.1: Representative RGB-D image pairs of the *ICL-NUIM* dataset [76]. (a) and (b) are RGB and depth data without sensor noise extracted from the "living room" scene, while (c) and (d) extracted from "office" are modelled with sensor noise.

The *ICL-NUIM* dataset [76] contains two different static indoor scenes, namely "living room" and "office", consisting of a collection of synthetic handheld RGB-D camera sequences. Both scenes come with perfect ground truth camera trajectory for evaluating the algorithms'

accuracy. As a synthetic dataset, it provides well-synchronized colour and depth image pairs for every frame. Moreover, in addition to synthetic imagery (see Figure 5.1a and Figure 5.1b), this dataset provides typically observed real-world artefacts in both RGB and depth data by carefully modelling sensor noise (see Figures 5.1c and 5.1d). The noisy colour and depth data are used to simulate realistic conditions in our evaluation. Since this dataset is much easier than the actual applicant scenarios of dynamic SLAM, we mainly apply it to demonstrate that the proposed system performs comparable to the baseline proposed in ∇SLAM [1].

5.1.2 TUM RGB-D Dataset

(a) RGB and depth input of static environment **(b)** RGB and depth input input of dynamic environment

Figure 5.2: Example images from the *TUM RGB-D* dataset [73]. Example (a) taken from the *freiburg1_desk* sequence consists of a purely static environment, while (b) from the *freiburg3_walking_xyz* sequence presents a dynamic environment with moving people. The real depth input is nosier than the synthetic data (s. Figure 5.1d), containing more invalid pixels, especially near the border.

The *TUM RGB-D* dataset [73] provides a rich collection of RGB and depth images for evaluating the performance of visual SLAM algorithms. It includes sequences with and without moving objects, adding to the complexity and realism of the SLAM challenges it presents. However, its RGB and depth data captured by the Kinect camera are in different frame rates. Although this has been carefully taken care of by synchronizing the data as prepossessing, we cannot fully mitigate its impact on the approach, which relies on a good association between colour and depth images. This is discussed in detail in section 6.2.2. Moreover, the moving instances in this dataset are restricted to only one class, people, which is non-rigid. They're not in the scope of our targeted categories. Therefore, when evaluating the system on this dataset, we mainly eliminate the impact of the moving people by filtering the related pixels using segmentation masks generated by methods discussed in Chapter 4.

5.1.3 Bonn RGB-D Dynamic Dataset

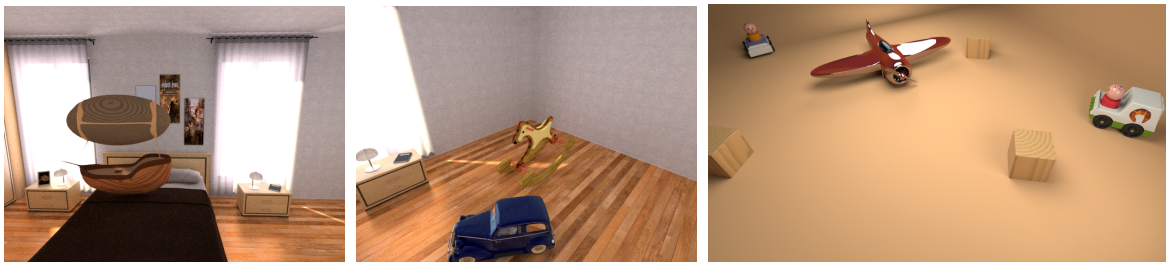
The *Bonn RGB-D Dynamic* dataset [74] is a successor of *TUM RGB-D* but concentrates more on dynamic scenarios. It contains 24 highly dynamic sequences where people perform different tasks, such as interacting with objects or walking around and two static sequences with the ground truth pose of the sensor. Identical to *TUM RGB-D* dataset, our approach can not completely resolve the synchronization issue of RGB and depth data in this dataset.

5.1.4 Co-Fusion Dataset

The *Co-Fusion* dataset, which is introduced along with the work Co-Fusion [49], contains two synthetic indoor dynamic scenes. It has been widely used in evaluating other dynamic visual SLAM algorithms. However, the moving objects' classes in these two sequences rarely appear

(a) Static indoor environment **(b)** Moving people **(c)** Playing with a balloon **(d)** Placing a cardboard box

Figure 5.3: Example images from the *Bonn RGB-D Dynamic* dataset [74]. Four typical representative scenarios in the *Bonn RGB-D Dynamic* dataset are shown here. Among them, the first one (a) illustrates a static environment. The rest shows various dynamic scenes. (b) visualizes multiple synchronously moving people, while (c) and (d) present images of a person manipulating different objects, including balloons and cardboard boxes.



(a) Airship **(b)** Toy car and rocking horse **(c)** Moving toy cars and a static airplane model

(d) Mask of airship **(e)** Masks of toy car and horse **(f)** Masks of toy cars

Figure 5.4: Example RGB images and segmentation masks from the *Co-Fusion* dataset. Input RGB images are shown on the upper row, while the corresponding object masks are listed below. Examples (a), (b), (d), and (e) visualize the three moving objects in the *room* sequence, including the airship, the toy car, and the rocking wooden horse. The remainder shows the two moving toy cars in the *car* sequence. Note that all other objects not coloured in the masks are not moving.

in datasets of classification or segmentation tasks and therefore highly limit the performance of the segmentation models which are trained in a supervised manner, such as Mask R-CNN [10] which we use in one of our proposals. Different from the other selected datasets, the Co-Fusion dataset includes ground truth segmentation masks of the moving instances, with which we can evaluate the effectiveness of our segmentation model quantitatively by calculating the Intersection over Union (IoU).

5.2 Metrics

To verify the effectiveness of our method from various perspectives, we employ multiple metrics during evaluation. Typical metrics for evaluating visual SLAM or visual odometry methods include the Absolute Trajectory Error (ATE [m]) and the Relative Pose Error of rotation and translation (RPE_{tran} [m/Frame] and RPE_{rot} [Deg/Frame]) introduced in [73]. The

ATE assesses the total trajectory accuracy by aggregating positional discrepancies between ground truth and reconstructed locations. In contrast, the RPE focuses on local motion discrepancies, serving as an indicator of drift. We evaluate the translation and rotation RPE in separate steps because it is not intuitive to understand from the numeric result how the distribution between the two errors is if they are embedded in one error.

Moreover, to ensure these measures are independent of scene length, they are commonly represented as the root-mean-square error (RMSE). To demonstrate the effectiveness of our segmentation model, we calculate the intersection over union (IoU) of the predicted and ground truth segmentation masks, particularly for the *Co-Fusion* dataset. These metrics' detailed definitions and mathematical formulation are provided in Appendix A. Furthermore, we visualize the 3D reconstruction of the particular scenes to provide intuitive results.

5.3 Evaluation Results

We summarize the evaluation results into two categories depending on the type of environment, i.e., the static environment (Section 5.4) and the dynamic environment (Section 5.5). In this way, we first demonstrate our method's ability to estimate camera trajectories. This is important because our object pose estimation is calculated by concatenating the object poses estimated in a local coordinate and the camera transformation, as described in Section 3.2.4. Without accurate camera tracking, our method cannot guarantee satisfying object-tracking results.

Then, we evaluated it in a comprehensive setting with the camera moving in dynamic environments. Since our pipeline does not include a method that explicitly distinguishes elastic and rigid objects, we manually categorize the sequences we use in evaluation. If a sequence contains a non-rigid moving instance, we merely filter all moving objects using the segmentation masks so that our system can achieve more accurate and robust camera tracking. If all the dynamic objects in a sequence are rigid, we track both the camera and objects for this sequence. In evaluation, we compare our methods with non-learning and learning-based, static and dynamic approaches.

We choose **∇SLAM** [1] as our development baseline, but it is not initially crafted for dynamic SLAM. Its visual odometry is a differentiable implementation of the pure geometric Point-to-Plane ICP method [77], known for its robustness to lighting changes. Therefore, we only compare our pipeline to it on sequences with a static environment. Besides, there are no trainable parameters in this framework. Hence, we categorize it as one of the non-learning approaches.

In the case of non-learning and static approaches, in addition to ∇SLAM, an **RGB-D VO** method [78] that minimizes the photometric consistency of aligned images is considered in comparing camera motion estimation. In the assessment, we also include the evaluation of **ColorICP** [79], a deviation of the above methods, which implements geometric constraint in addition to photo consistency. Comparing our approach to these methods is expected to highlight the advantages of learning-based visual odometry, particularly in terms of a broader convergence basin, which helps avoid being trapped in local minima (related discussion is presented in Section 6.2.1) and, therefore, enhances accuracy.

As learning-based approaches, we refer to SLAM systems that involve deep learning methods for either segmentation or visual odometry. In evaluation, we provide comparisons with **DROID-SLAM** [6], **DynaSLAM** (DS)[48], and **MaskFusion** (MF)[45]. DynaSLAM and MaskFusion rely on deep neural networks to obtain prior knowledge of possible moving objects, followed by refinement using a geometric approach. The main difference between them in

segmentation lies in using different geometric methods. Moreover, DynaSLAM uses feature-based visual odometry to track the camera, while MaskFusion and ReFusion estimate camera motions in a direct fashion. MaskFusion minimizes photometric and geometric errors to track the camera and moving objects in the environment. DROID-SLAM does not come with a dynamic object segmentation module and estimates solely camera motion. However, it has a novel learning-based visual odometry pipeline. It shows state-of-the-art performance on a broad range of data, including the most challenging TartanAir Dataset [42]. This dataset contains a considerable amount of dynamic sequences. Despite that, DROID-SLAM can still achieve promising performance, which makes it an interesting approach to include in our comparison.

Additionally, our method is compared to **ReFusion** (RF) [74] in dynamic sequences, which represents the non-leaning-based, dynamic methods. ReFusion optimizes the error function defined with TSDF and tracks the camera solely. It exploits residuals after the initial pose estimation to identify moving entities. It does not employ deep-learning methods to segment objects in the scene and can not track objects. However, its residual-based method for filtering dynamic objects significantly improves the accuracy of camera motion estimation.

We've used a large portion of the *TUM RGB-D* and *Bonn RGB-D Dynamic* datasets to train our visual odometry (see Table 4.1). Comparing our system's performance to others on the train sequences might not be objective. Thus, we identify the test sequences with "*" in the quantitative results on the *TUM RGB-D* and the *Bonn RGB-D Dynamic* datasets.

5.4 Moving Camera in Static Environment

The assumption that the camera is the only moving object in the environment has been the foundation of primary classic visual SLAM systems. In this experiment, we selected sequences from synthetic and real datasets that fulfil this configuration. Since no moving instances exist in the selected sequences, we deactivate the segmentation module in our pipeline and evaluate solely the learning-based visual odometry. We compare our method with non-learning and learning-based, static approaches to validate our approach's effectiveness in terms of camera tracking accuracy.

	ColorICP[79]	RGB-D VO[78]	∇SLAM[1]	Ours	DROID[6]
living_room_traj0	0.2802	0.3603	**0.0954**	0.1281	0.6369
living_room_traj1	0.0171	0.5951	**0.0284**	0.0926	0.2954
living_room_traj2	0.1327	0.2931	**0.0334**	0.1008	0.8094
living_room_traj3	0.8168	0.8524	**0.3784**	0.4532	0.6545
office_traj0	0.2047	0.1710	0.0446	**0.0191**	0.5958
office_traj1	0.4062	0.5366	0.9921	**0.1360**	0.4122
office_traj2	0.2827	0.2289	0.0339	**0.0473**	1.0089
office_traj3	**0.0572**	0.2289	0.9553	0.1084	0.7044
Mean	0.2747	0.4083	0.3202	**0.1357**	0.6397

Table 5.1: Absolute Trajectory Error (RMS) on the *ICL-NUIM* dataset. ATE is presented with the unit [m]. The second-best results are underlined, and the bests are shown in bold. Compared to the other methods, ours achieves the best performance on some sequences while performing reasonably compared to the best methods on the rest.

Table 5.1 shows the evaluation results on the *ICL-NUIM* dataset [76], while several qualitative results of our approach are shown in Figure 5.5. Despite DROID-SLAM's surprisingly large errors, compared to other methods, we observed no distinct performance gaps. Our

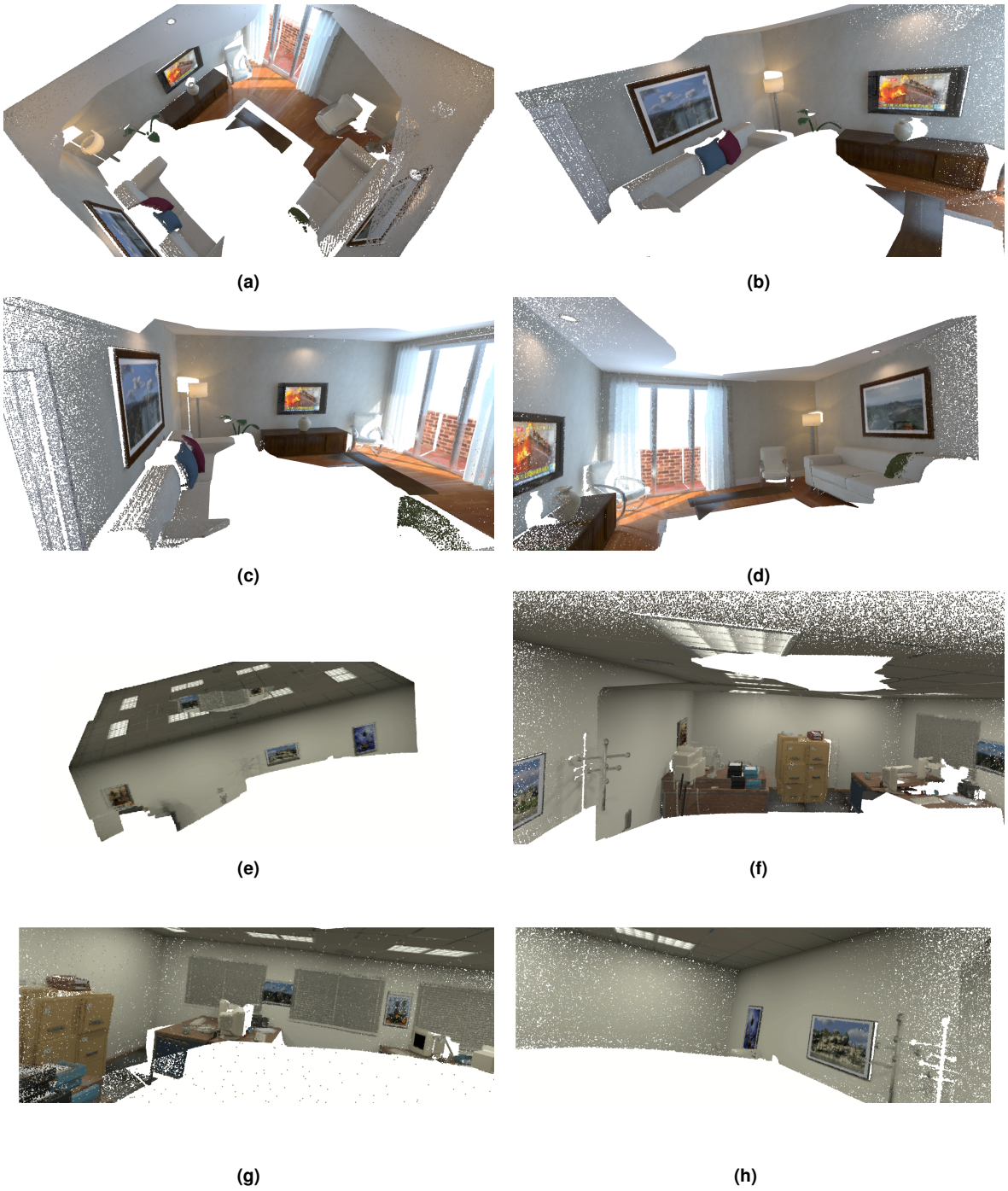


Figure 5.5: Reconstruction results of sequences from *ICL-NUIM* dataset. For better observation, we reconstruct the scene using the noise-free depth data and the camera poses are estimated using noisy depth input. Figure (a)-(d) shows the reconstruction of the *living-room* sequences from different viewpoints, while the reminder illustrates the results of the *office* sequences.

method achieves the best performance over all scenarios. Besides, we cannot spot clear misalignment in the reconstruction. The main reason is that these sequences themselves are relatively simple. The relative displacement between frames is smooth, and the ego-camera has no rapid pose changes. However, based on the results, we can still conclude that, in these sequences, the performance of our method is comparable to the baseline methods, which preliminarily verifies the feasibility of our method.

Table 5.2 and Table 5.3 present the quantitative results on static environment sequences of the *TUM RGB-D* dataset [73]. Compared to classic methods, including ColorICP, RGB-D VO, and ∇SLAM, our method tracks camera motions with lower average ATE and RPE across almost all trajectories. On the other hand, compared to the learning-based method, DROID-SLAM, which has the lowest ATE in general, our method exhibits enhanced accuracy in estimating relative poses. we presume that the bundle adjustment optimization invoked in DROID-SLAM contributes the most to this performance gap in the ATE.

	Non-leanring-based			Learning-based	
	ColorICP [79]	RGB-D VO [78]	∇SLAM [1]	Ours	DROID [6]
fr1_360*	0.1457	0.1994	0.1812	0.1010	**0.0669**
fr1_desk*	0.1375	0.0909	0.2636	0.0792	**0.0177**
fr1_desk2	0.2077	0.9024	0.3630	0.0796	**0.0268**
fr1_floor	0.3673	0.7517	0.7136	0.2248	**0.0222**
fr1_room	0.2894	0.9392	0.4849	0.1573	**0.0429**
fr1_xyz	0.0603	0.1056	0.1614	0.0820	**0.0101**
fr1_rpy	0.0453	0.0607	0.0487	0.0430	**0.0233**
fr1_plant	0.1631	0.6992	0.4090	0.0824	**0.0166**
fr1_teddy	0.1390	0.8626	0.5138	0.1669	**0.0358**
fr2_desk*	0.5571	1.3994	1.4237	0.9178	**0.0105**
fr2_360_hemisphere	x	0.9366	0.9460	**0.3655**	0.8712
fr2_large_no_loop	1.7268	1.6488	1.9762	1.1007	**0.9495**
fr2_large_with_loop	x	1.4734	1.4123	1.2061	**1.1131**
fr2_pioneer_360*	1.7701	1.6859	1.6656	**0.2534**	1.7422
fr2_pioneer_slam	1.7795	1.7762	1.8406	0.5549	**0.1837**
fr2_pioneer_slam2	1.0337	1.6338	1.2648	**0.6217**	1.0060
fr2_pioneer_slam3	1.0338	1.8659	1.5458	**0.8649**	1.1698
fr2_xyz	0.3086	0.3607	0.2719	0.1150	**0.0046**
fr2_rpy	0.0606	0.0765	0.0715	0.0564	**0.0120**
fr2_dishes	0.2750	0.8369	0.7549	0.2410	**0.0146**
fr2_360_kidnap	x	1.5388	1.4253	0.8922	**1.3397**
fr2_desk_with_person	0.4700	1.5465	0.9499	0.1725	**0.0170**
fr3_teddy	x	0.7603	0.7505	0.2491	**0.0107**
fr3_long_office_household	0.2853	0.5797	1.7370	0.5498	**0.0284**
Average	0.5428	0.9471	0.8823	0.3824	**0.3640**

Table 5.2: Absolute Trajectory Error (RMS) on static environment sequences of *TUM RGB-D* dataset. This table summarizes the ATE [m] evaluated on the *TUM RGB-D* static sequences. We compare our method to the RGB-D VO, ColorICP, ∇SLAM, and DROID-SLAM. The sequences not involved in the training are marked with "*". Fail estimations are denoted by "x". The best results are stressed in bold, while the second-best results are shown with underlines.

5.5 Moving Camera in Dynamic Environment

The camera moves occasionally in real-world scenarios, and objects in the environment can also be dynamic. The performance of our approach in handling this complexity is evaluated in this particular configuration. Among all the datasets we used, we have four of them that contain representative sequences of this setting, including *TUM RGB-D*, *Bonn RGB-D Dynamic*, and *Co-Fusion* datasets.

Table 5.5 and 5.5 summarize the evaluation results on the dynamic sequences or the *TUM RGB-D* dataset. The evaluation results on the *Bonn RGB-D Dynamic* dataset are shown in Table 5.6 and Table 5.7. Compared to non-learning-based methods without a segmentation module, including RGB-D VO, ColorICP and ∇SLAM, our methods show improvement in estimating camera motions. On the other hand, although DROID-SLAM does not have a segmentation module, it is found to be more accurate than our methods on the dynamic

	Non-leanring-based						Learning-based			
	ColorICP		RGB-D VO		SLAM		Ours		DROID	
	RPE_{trans}	RPE_{rot}	RPE_{trans}	RPE_{rot}	RPE_{trans}	RPE_{rot}	RPE_{trans}	RPE_{rot}	RPE_{trans}	RPE_{rot}
fr1_360*	0.0045	0.6803	0.0087	15.6504	0.0053	**0.6315**	**0.0037**	0.6637	0.0295	5.5272
fr1_desk*	0.0125	0.8644	0.0096	0.6639	0.0092	**0.5252**	**0.0080**	0.5961	0.0565	4.6464
fr1_desk2	0.0135	0.9941	0.0437	11.0466	0.0357	1.4220	**0.0089**	**0.7120**	0.0647	5.8853
fr1_floor	0.0222	0.9752	0.0345	6.5636	0.0235	**0.9125**	**0.0211**	1.0183	0.0411	3.5380
fr1_room	0.0069	0.4688	0.0316	16.5310	0.0115	0.6135	**0.0064**	**0.4634**	0.0447	4.7038
fr1_xyz	0.0052	0.3388	0.0052	0.4196	0.0045	**0.2909**	**0.0035**	0.3400	0.0382	2.1248
fr1_rpy	0.0020	1.0294	0.0028	11.4244	**0.0018**	**0.8466**	0.0020	0.9494	0.0143	5.7986
fr1_plant	0.0072	0.5651	0.0193	16.6459	0.0066	**0.5114**	**0.0062**	0.6295	0.0566	4.4496
fr1_teddy	0.0092	0.5769	0.0215	14.1236	0.0095	**0.5453**	**0.0079**	0.7149	0.0582	5.5342
fr2_desk*	0.0158	0.5089	0.0240	2.1736	0.0140	**0.4634**	**0.0132**	0.4713	0.0901	2.5560
fr2_360_hemisphere	x	x	0.0432	50.3544	0.0443	5.6602	**0.0147**	**0.3680**	0.0537	2.9343
fr2_large_no_loop	0.0817	**2.1908**	0.0904	27.8773	0.0827	2.2240	**0.0760**	2.1980	0.3296	7.1936
fr2_large_with_loop	x	x	0.0576	28.7691	0.0585	1.7831	**0.0516**	**1.5377**	0.1660	7.2342
fr2_pioneer_360*	0.0120	0.9577	0.0288	21.1785	0.0257	0.7716	**0.0105**	**0.7302**	0.0272	3.6863
fr2_pioneer_slam	0.0310	1.7811	0.0485	46.4098	0.0316	1.8302	0.0263	**1.6878**	**0.0252**	4.5299
fr2_pioneer_slam2	0.0466	**2.3164**	0.1169	42.4740	0.0482	2.4569	**0.0391**	2.3567	0.0594	4.4346
fr2_pioneer_slam3	0.0432	2.0008	0.0608	34.8880	0.0362	1.9307	**0.0199**	**1.7581**	0.0241	4.7357
fr2_xyz	0.0019	**0.2197**	0.0030	6.2224	0.0026	0.2247	**0.0018**	0.2320	0.0063	0.4738
fr2_rpy	0.0008	**0.1897**	0.0013	11.0813	0.0008	0.2248	**0.0008**	0.1926	0.0029	0.7726
fr2_dishes	0.0112	0.7381	0.0269	35.9551	0.0103	0.5893	**0.0029**	**0.2897**	0.0201	1.5152
fr2_360_kidnap	x	x	0.0209	26.6967	**0.0058**	**0.3120**	0.0068	0.3129	0.1330	3.0609
fr2_desk_with_person	0.0080	**0.3430**	0.0305	5.4565	0.0102	0.3919	**0.0068**	0.3577	0.0327	1.4022
fr3_teddy	x	x	0.0092	18.7393	0.0058	1.5902	**0.0039**	**0.5394**	0.0357	3.2697
fr3_long_office_household	0.0046	0.2541	0.0077	0.3892	0.0044	**0.2704**	**0.0035**	0.2936	0.0312	1.7115
Average	0.0170	0.8997	0.0311	18.8223	0.0204	1.1259	**0.0144**	**0.8089**	0.0600	3.8216

Table 5.3: Relative Pose Error (RMS) on static environment sequences of *TUM RGB-D* dataset. We compare our method to the RGB-D VO, ColorICP, ∇SLAM, and DROID-SLAM. In this table, the test sequences are marked with "*" for a fair comparison. Fail estimations are denoted by "x". The best results are stressed in bold, while the second-best results are shown with underlines.

sequences from *TUM RGB-D* in terms of the ATE. It achieves even comparable results to the dynamic SLAM methods. DROID-SLAM's robust and superior performance is attributed to its state-of-the-art learning-based visual odometry and a large amount of challenging train data. Additionally, DynaSLAM, as the only method that tracks the camera using sparse feature points, exhibits the best outcomes in ATE and RPE_{trans} in the *TUM RGB-D* dynamic sequences. However, our methods exhibit competitive performance in relative pose estimation, in general. Despite the absence of the RPE_{rots} of the last three methods, ours consistently yields the best outcomes in RPE on the *Bonn RGB-D Dynamic* dataset, which is, in general, much more challenging than the *TUM RGB-D* dataset. From this, we can presume that our methods' better performance benefits from the uncertainty-based tracking algorithm.

Moreover, among the two segmentation approaches we introduced, the pipeline integrated with the uncertainty-based segmentation approach (see Section 4.7.2) can track ego-motion more accurately than the other using geometric segmentation. Nonetheless, generally, compared to all state-of-the-art methods, the proposed approaches lagged slightly behind in tracking the camera within dynamic environments.

When evaluating our method on the *Co-Fusion* dataset, we compare with **Co-Fusion** [49] instead of MaskFusion because we cannot generate reasonable results using the public implementation of MaskFusion, especially for moving objects. Co-Fusion is also a visual SLAM system that separates and maintains multiple object models by motion. Conventional non-dynamic SLAM/VO systems are not designed to effectively track camera motion in dynamic environments. For this reason, in this experiment, we limit non-dynamic and non-learning approaches to our baseline ∇SLAM. This approach is chosen specifically to demonstrate how moving objects affect the accuracy of camera motion estimation.

The quantitative evaluation results on the *Co-Fusion* dataset are shown in Table 5.8. Com-

		NL + NM			L + NM	L + M		NL + M		
		ColorICP [79]	RGB-D VO [78]	∇SLAM [1]	DROID [6]	Ours+GS	Ours+US	DS [48]	RF [74]	MF [45]
Slightly dynamic	fr3_s_static	0.0293	0.0224	0.0151	0.0067	0.0164	0.0095	**0.0064**	0.0110	0.0154
	fr3_s_xyz	0.2995	0.1718	0.1586	0.0288	0.1306	0.0963	**0.0146**	0.0315	0.0614
	fr3_s_rpy	0.0504	0.0581	0.0489	**0.0158**	0.0450	0.0482	0.0673	0.1996	0.0924
	fr3_s_halfsphere	0.2562	0.3808	0.2391	0.0217	0.1953	0.0676	**0.0196**	0.0387	0.0606
Highly dynamics	fr3_w_static*	0.0222	0.0223	0.0226	0.0169	0.0215	0.0226	**0.0067**	0.0170	0.5021
	fr3_w_xyz*	0.2839	0.2887	0.2806	0.0191	0.2736	0.2084	**0.0161**	0.0809	0.3457
	fr3_w_rpy	0.1553	0.1675	0.1628	0.0601	0.1566	0.0941	**0.0393**	0.2660	0.6605
	fr3_w_halfsphere	0.3817	0.4340	0.3834	0.0309	0.3601	0.2739	**0.0278**	0.0584	0.3906
	Average	0.1848	0.1932	0.1639	0.0250	0.1499	0.1026	**0.0247**	0.0879	0.2661

Table 5.4: Absolute Trajectory Error (RMS) on TUM RGB-D dataset dynamic environment sequences. The sequences are categorized into two groups depending on the degree of dynamics presented and compare our method to the RGB-D VO, ColorICP, ∇SLAM, DROID-SLAM, DynaSLAM (DS), ReFusion (RF), and MaskFusion (MF). Note that we only include RPE_{trans} of the last three methods, which are excerpted from [80]. We identify the two segmentation approaches introduced in this thesis using GS for the one described in Section 6.5 and US for the other in Section 6.5, respectively. We also marked the test sequences with "*". Fail estimations are denoted by "x". The best results are stressed in bold, while the second-best results are shown with underlines.

		NL + NM						L + NM		L + M				NL + M		
		ColorICP[79]		RGB-D VO[19]		∇SLAM[1]		DROID[6]		Ours+GS		Ours+US		DS[48]	RF[74]	MF[45]
Setting	Seq.	RPE_{trans}	RPE_{rot}	RPE_{trans}	RPE_{rot}	RPE_{trans}	RPE_{rot}	RPE_{trans}	RPE_{trans}	RPE_{trans}	RPE_{rot}	RPE_{trans}	RPE_{rot}	RPE_{trans}	RPE_{trans}	RPE_{trans}
Slightly dynamic	fr3_s_static	0.248	0.145	1.262	0.372	0.184	0.177	0.584	0.665	0.078	0.066	0.072	**0.033**	**0.012**	0.036	0.045
	fr3_s_xyz	0.422	0.299	1.554	0.465	0.337	0.371	1.072	3.220	0.330	0.201	0.187	**0.165**	**0.020**	0.047	0.054
	fr3_s_rpy	0.672	0.450	x	x	0.694	0.555	2.076	1.104	0.345	0.239	0.170	**0.198**	**0.131**	0.650	0.357
	fr3_s_halfsphere	0.583	0.461	1.008	0.426	0.437	0.528	2.953	3.161	0.341	0.248	0.177	**0.208**	**0.043**	0.048	0.084
Highly dynamics	fr3_w_static*	0.879	0.265	2.884	0.682	0.818	0.275	0.357	0.567	0.563	0.107	0.374	**0.084**	**0.010**	0.040	0.601
	fr3_w_xyz*	1.888	0.480	3.325	0.761	1.654	0.510	3.129	1.567	0.906	0.254	0.729	**0.220**	**0.023**	0.094	0.318
	fr3_w_rpy	1.910	0.563	24.767	2.959	1.773	0.652	2.667	3.417	1.061	0.313	0.926	**0.279**	**0.059**	0.280	0.570
	fr3_w_halfsphere	1.666	0.574	2.644	0.666	1.454	0.661	3.832	2.973	0.848	0.286	0.749	**0.257**	**0.039**	0.056	0.391
	Average	1.033	0.405	5.349	0.904	0.919	0.466	2.084	2.084	0.559	0.214	0.423	**0.180**	**0.042**	0.156	0.302

Table 5.5: Relative Pose Error (RMS) on *TUM RGB-D* dataset dynamic environment sequences. We categorize the sequences into two groups depending on the degree of dynamics presented. We compare our method to the RGB-D VO, ColorICP, ∇SLAM, DROID-SLAM, DynaSLAM (DS), ReFusion (RF), and MaskFusion (MF). We only include RPE_{trans} of the last three methods, excerpted from [80]. We distinguish the two segmentation approaches introduced in this thesis using GS for the one described in Section 4.7.1 and US for the other in Section 4.7.2, respectively. Test sequences are marked with "*". Fail estimations are denoted by "x". The best results are stressed in bold, while the second-best results are shown with underlines.

pared to our development baseline, ∇SLAM, the proposed method performs more accurate camera tracking. Additionally, our method tracks the objects in the environment with satisfactory accuracy. This is mainly attributed to the moving object segmentation in our pipeline. Nevertheless, despite the strengths in our segmentation method, Co-Fusion surpasses the proposed method with a large performance gap. Reasons for this could be multiple.

On the one hand, Co-Fusion has a well-engineered pipeline. It performs frame-to-model tracking, estimating directly the global object poses. Although it relies on classic, i.e., non-learning residuals, which combine photometric colour error and point-to-plane ICP registration error, it shows superior performance in tracking.

On the other hand, our visual odometry system relies on frame-to-frame image alignment. We estimated the object poses in the local camera coordinate and computed the global object poses by transforming, which accumulated the error from both camera and object tracking. This could be the reason for our system's poorer accuracy in object tracking compared to Co-Fusion. Furthermore, our visual odometry operates on feature maps extracted from colour images. The *car4* sequence is a large number of homogeneous surfaces, which are mostly of identical colour values. We do not particularly include sequences with similar properties

	NL + NM			L + M		NL+ M		
Sequence	ColorICP	RGB-D VO	SLAM	Ours+GS	Ours+US	**DS**	RF	MF
balloon	0.128	0.207	0.137	0.091	0.071	**0.030**	0.175	0.0912
balloon2	0.134	0.191	0.122	0.085	0.065	**0.029**	0.254	0.0312
balloon_tracking	0.262	0.388	0.253	0.229	0.209	**0.049**	0.302	0.1442
balloon_tracking2	0.174	0.301	0.280	0.228	0.208	**0.035**	0.322	0.0847
crowd	x	0.073	0.071	0.013	0.033	**0.016**	0.204	-
crowd2	0.074	0.080	0.073	0.011	0.031	**0.031**	0.155	-
crowd3	0.062	0.060	0.059	0.012	0.032	**0.038**	0.137	0.1548
kidnap_box	0.233	0.550	0.350	0.224	0.204	**0.029**	0.148	-
kidnap_box2	0.165	0.487	0.250	0.275	0.255	**0.035**	0.161	0.1298
mov_nonObs_box	0.171	0.281	0.207	0.219	0.199	0.232	**0.071**	-
mov_nonObs_box2	0.259	0.478	0.248	0.331	0.311	**0.039**	0.179	0.121
mov_obs_box	0.178	0.219	0.261	0.122	0.102	**0.044**	0.343	-
mov_obs_box2	0.220	0.260	0.259	0.172	**0.132**	0.263	0.528	0.170
person_tracking	0.264	0.625	0.268	0.345	0.325	**0.061**	0.289	0.216
person_tracking2	0.258	0.346	0.241	0.489	0.469	**0.078**	0.463	-
placing_nonObs_box	0.174	0.136	0.185	0.1789	0.118	0.575	**0.106**	0.214
placing_nonObs_box2	0.123	0.156	0.124	0.079	0.059	**0.021**	0.141	-
placing_nonObs_box3	0.191	0.205	0.157	0.134	0.114	**0.058**	0.174	-
placing_obs_box	0.123	0.306	0.227	0.113	**0.093**	0.255	0.571	0.349
removing_nonObs_box	0.108	0.130	0.110	0.062	0.042	**0.016**	0.041	0.051
removing_nonObs_box2	0.083	0.173	0.137	0.109	0.089	**0.021**	0.111	-
removing_obs_box	0.150	0.155	0.174	0.114	**0.094**	0.291	0.222	-
synchronous	0.006	0.004	0.006	0.004	**0.003**	0.015	0.441	-
synchronous2	0.023	0.025	0.022	0.018	0.018	**0.009**	0.022	-
Average	0.155	0.243	0.176	0.147	0.134	**0.0946**	0.2317	0.1464

Table 5.6: Absolute Trajectory Error (RMS) on the dynamic scenes of the *Bonn RGB-D Dynamic* dataset. Here, we evaluate our deep-learning-based methods (L) equipped with two different segmentation approaches (M), i.e., geometric-refined segmentation using Mask R-CNN (GS) and uncertainty-based segmentation using FastSAM (US). The listed methods are divided in groups. The first group employs non-learning-based visual odometry (NL) and does not apply motion segmentation (NM), which includes the ColorICP, RGB-D VO, and ∇SLAM. The second group consists of DynaSLAM (DS), ReFusion (RF), and MaskFusion (MF). These methods employ non-learning-based tracking (NL) and has the capability to identify moving objects (M). Failures are marked with "x". Since results of DS, RF, and MF are cited from [80] and [81], some results are not available, which are noted with "-". Additionally, we stress that the best and second-best results are denoted using bold fonts and underlines.

in training, which might be the main reason for the poor performance in this particular sequence. On the other hand, the airship in the *room4* sequence is split into two instances by Co-Fusion. In contrast, thanks to the superior performance of FastSAM, our system segments it as one, which is the potential reason for its better performance in tracking this particular object.

Table 5.9 presents the average IoU between the predicted masks and the ground truth labels across all frames in which instance masks are predicted. Our first segmentation highly relies on the prior information provided by Mask R-CNN, which, however, cannot segment the moving instances in these sequences correctly. In contrast, our second segmentation method can generate prompt points with less noise from the predicted dynamic-aware uncertainties. In general, with accurate prompts, FastSAM can produce fine segmentation masks. Thus, our segmentation method achieves an overall higher average IoU than Co-Fusion.

	NL + NM						L + M				NL + M		
	ColorICP		RGB-D VO		∇SLAM		Ours + GS		Ours + US		DS	RF	MF
	RPE_{trans}	RPE_{rot}	RPE_{trans}	RPE_{rot}	RPE_{trans}	RPE_{rot}	RPE_{trans}	RPE_{rot}	RPE_{trans}	RPE_{rot}	RPE_{trans}	RPE_{trans}	RPE_{trans}
balloon	0.041	2.342	0.041	15.333	0.040	**1.748**	0.031	2.356	**0.028**	2.056	0.435	0.277	1.043
balloon2*	0.029	3.165	0.037	29.039	0.026	**2.662**	0.027	3.032	**0.024**	2.782	0.512	1.600	1.599
balloon_tracking	0.071	5.589	0.049	21.448	0.073	**4.799**	0.045	5.602	**0.042**	5.452	0.925	0.935	0.826
balloon_tracking2*	0.105	6.809	0.076	11.000	0.083	**5.471**	0.059	8.113	**0.056**	7.763	0.886	0.772	1.298
crowd	x	x	0.010	22.882	0.010	**1.920**	0.011	2.363	**0.010**	2.063	-	-	-
crowd2	0.013	2.817	**0.013**	28.365	0.014	**2.513**	0.014	4.163	0.013	3.813	-	-	-
crowd3*	0.009	3.006	**0.009**	29.412	0.009	**2.030**	0.009	2.589	0.009	2.429	0.253	0.310	0.230
kidnapping_box	0.074	4.380	**0.050**	6.406	0.076	3.836	0.066	3.728	0.063	**3.428**	-	-	-
kidnapping_box2	0.064	3.026	0.045	7.916	0.070	**2.517**	0.047	2.826	**0.044**	2.606	0.977	1.230	1.101
moving_nonobstructing_box	0.031	1.412	0.030	1.532	0.037	1.156	0.019	0.933	**0.016**	**0.663**	-	-	-
moving_nonobstructing_box2	0.071	3.821	**0.047**	8.331	0.079	**3.046**	0.054	3.529	0.051	3.389	1.179	1.429	1.260
moving_obstructing_box	0.061	4.725	0.080	10.559	0.059	**3.810**	0.042	4.345	**0.039**	4.045	-	-	-
moving_obstructing_box2	0.040	2.186	0.057	6.345	0.050	1.825	0.031	1.926	**0.028**	**1.626**	1.003	0.765	0.577
person_tracking	**0.069**	8.147	0.073	9.156	0.076	**7.259**	0.075	7.967	0.072	8.217	1.038	1.226	1.545
person_tracking2*	0.061	6.289	0.082	11.387	0.066	**5.097**	0.054	5.862	**0.051**	5.992	-	-	-
placing_nonobstructing_box*	0.016	2.398	0.021	3.419	0.016	2.120	0.008	2.118	**0.005**	**1.998**	0.442	0.111	0.373
placing_nonobstructing_box2	0.019	2.162	0.018	6.871	0.021	1.814	0.008	1.749	**0.005**	**1.449**	-	-	-
placing_nonobstructing_box3	0.020	2.145	0.018	40.581	0.029	**1.751**	0.010	1.912	**0.007**	1.972	-	-	-
placing_obstructing_box	0.027	2.203	0.030	7.859	0.034	1.984	0.015	1.981	**0.012**	**1.681**	0.632	0.675	0.643
removing_nonobstructing_box	0.017	0.985	0.016	7.153	0.017	0.879	0.006	0.715	**0.003**	**0.395**	0.325	0.440	0.424
removing_nonobstructing_box2	0.027	2.517	0.019	12.761	0.026	**2.098**	0.016	2.471	**0.013**	2.521	-	-	-
removing_obstructing_box	0.016	2.257	0.028	6.495	0.018	1.842	0.004	1.971	**0.001**	**1.911**	-	-	-
synchronous	0.001	0.495	0.002	2.050	0.001	**0.355**	0.001	0.919	**0.001**	0.879	-	-	-
synchronous2	0.003	0.717	0.005	21.154	0.003	**0.488**	0.003	1.463	**0.003**	1.293	-	-	-
Average	0.038	3.200	0.036	13.644	0.039	**2.626**	0.027	3.110	**0.025**	2.934	0.717	0.814	0.847

Table 5.7: Relative Pose Error (RMS) on dynamic sequences of the *Bonn RBG-D Dynamic* dataset. In this evaluation, our method is compared to the RGB-D VO, ColorICP, ∇SLAM, DynaSLAM (DS), ReFusion (RF), and MaskFusion (MF). Note that we only include RPE_{trans} of the last three methods, which are excerpted from [80]. We identify the two segmentation approaches introduced in this thesis using GS for the one described in Section 4.7.1 and US for the other in Section 4.7.2, respectively. Test sequences are marked with "*". Fail estimations are denoted by "x". The best results are stressed in **bold**, while the second-best results are shown with underlines.

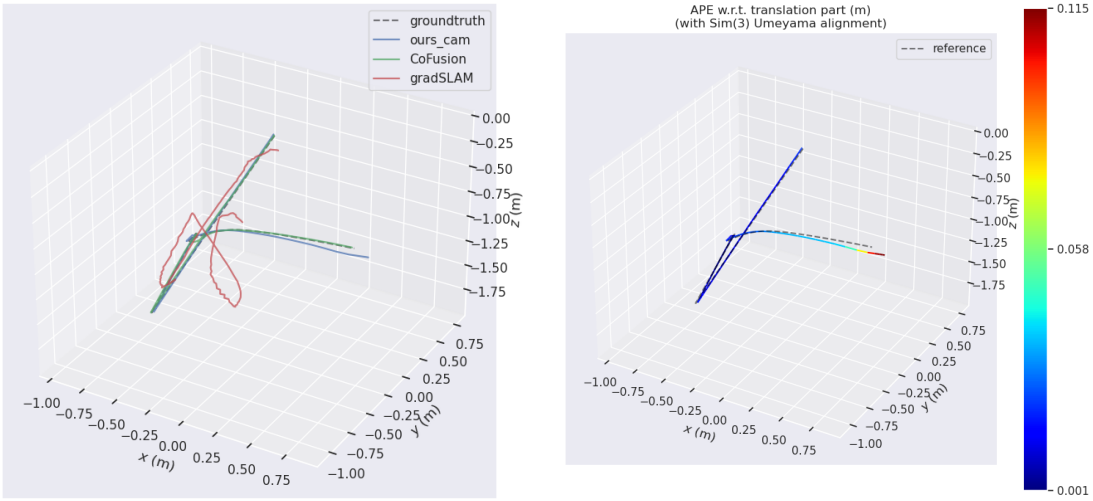
		∇SLAM [1]			Ours			CoFusion [49]		
		ATE	RPE_{trans}	RPE_{rot}	ATE	RPE_{trans}	RPE_{rot}	ATE	RPE_{trans}	RPE_{rot}
room4	cam	0.3839	0.0211	0.8467	0.0267	0.0048	0.0977	**0.0141**	**0.0006**	**0.0003**
	ship	-	-	-	**0.0096**	0.1328	0.0150	0.0298 / 0.0217	**0.0170 / 0.0092**	**0.0255 / 0.0061**
	horse	-	-	-	1.0266	0.1949	3.6004	**0.0612**	**0.0061**	**0.6729**
	car	-	-	-	0.0687	0.0437	0.0620	**0.0032**	**0.0091**	**0.0009**
car4	cam	x	x	x	0.3737	0.0083	0.0210	**0.0273**	**0.0011**	**0.0009**
	truck1	-	-	-	0.2249	0.0106	0.7793	**0.0103**	**0.0010**	**0.0045**
	car2	-	-	-	0.1269	0.0281	0.4560	**0.0070**	**0.0033**	**0.0059**

Table 5.8: Evaluation results on the *Co-Fusion* Dataset. We compare the proposed method with our development baseline ∇SLAM and Co-Fusion [49]. Ours outperforms the baseline, achieving better results in camera tracking and providing object tracking. However, compared to Co-Fusion, there is still a distinct bias between our method and this state-of-the-art method. The best results are shown in bold.

Additionally, the estimated trajectories compared to ground truth trajectories are visualized in Figure 5.6 and Figure 5.7 illustrates reconstructions generated by our method. Although the trajectory estimated by our system has a larger error than confusion, thanks to our mapping method, we can maintain better object shape reconstruction when the errors are in an acceptable range.

Sequence	Object	Ours	Co-Fusion
room4	Airship	0.85	0.43 / 0.62
	Horse	0.78	0.48
	Car	0.88	0.88
car4	Truck	0.81	0.79
	Car	0.76	0.68

Table 5.9: Average IoU. We calculate the average IoU between the predicted masks and the ground truth labels across all frames where the masks are predicted. We only compare the mask predicted by the uncertainty-prompted moving instance segmentation approach introduced in Section. As Mask R-CNN cannot generate reliable masks for the instances in these sequences, our first segmentation approach fails to provide usable instance masks.



(a) Camera trajectories

(b) Color-coded ATE of ours

Figure 5.6: Visualization of estimated camera trajectories. Figure (a) shows the camera trajectories estimated by ∇SLAM (gradSLAM), Co-Fusion, and our method. The trajectories of Co-Fusion almost perfectly overlap with the ground truth. Our method's estimate is slightly off the ground truth. ∇SLAM has relatively good tracking at the beginning of the sequence, while it struggles to estimate the camera motion after moving objects enter the view. Figure (b) visualizes the color-coded ATE on our estimation. The increasing errors near the end of the sequence depict that our method cannot maintain its tracking performance when multiple objects are in view.

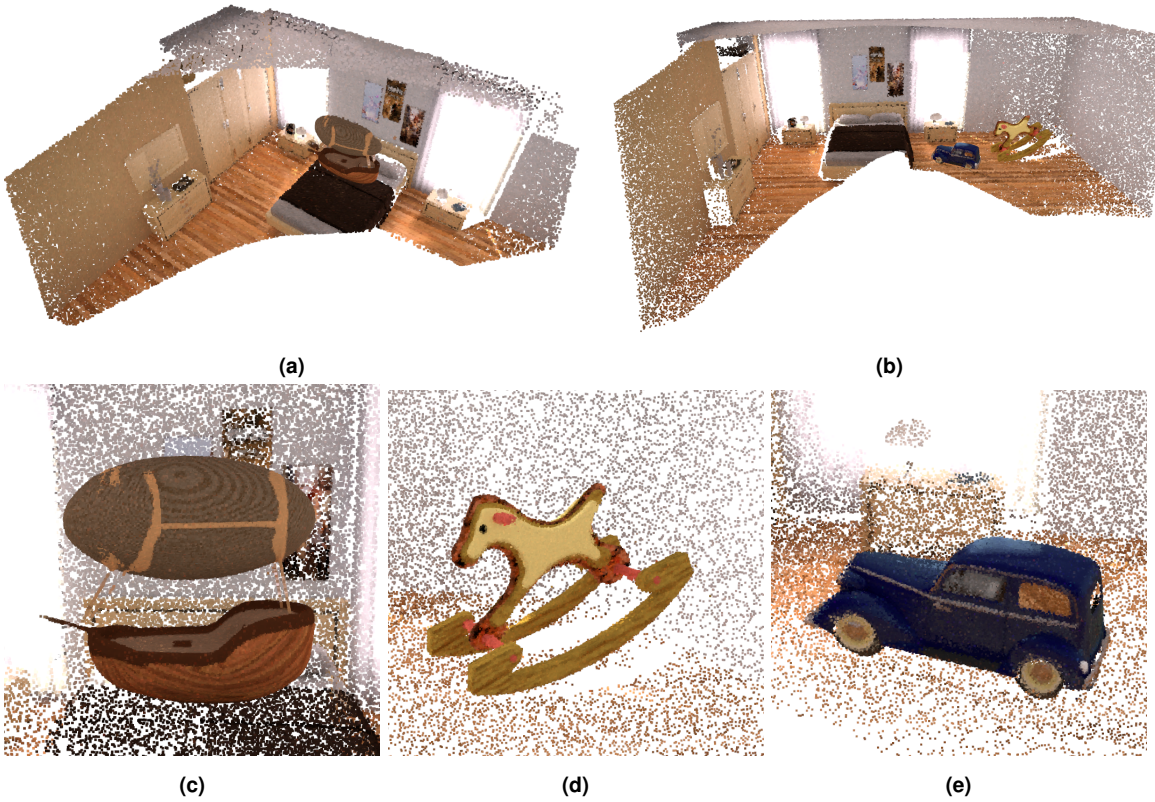


Figure 5.7: Reconstruction results of the *room* sequence from *Co-Fusion* dataset. Figures (a) and (b) show the whole environment's reconstruction, including moving entities. In particular, the airship in Figure (b) is erased from the visualization as it moves out of the camera view. Figure (c)-(e) shows the reconstruction of individual moving objects in detail. The misaligned texture of the airship shown in Figure (c) reflects an inaccurate tracking at the first few frames when the object enters the scene.

Chapter 6

Ablation Studies and Discussion

While the effectiveness of each stage in our pipeline is interdependent on the performance of the others - for instance, subpar segmentation could lead to inadequate reconstruction - it remains beneficial to evaluate the distinct components individually. In this chapter, an ablation study is presented to validate the contribution of each component in the pipeline. After that, we discuss the pipeline's limitations revealed in the experiments, followed by ideas for improvement and directions for future research.

6.1 Ablation Studies

Here, we present a comprehensive ablation study to elucidate the impact of individual components. We denote our system component, dense feature map, dense uncertainty map, deep initial pose prediction, and motion segmentation as **F**, **U**, **P**, and **M**, respectively. In this study, we conduct the following settings:

- **F**: In this variation, we perform the probabilistic feature-metric DIA, while the uncertainty prediction is substituted with an identity uncertainty. Additionally, we deactivate the pose initialization, assuming an identity motion and utilizing the prediction from the previous optimization instead.
- **F+P**: We additionally initialize the pose using Pose Net for the coarsest optimisation level, maintaining other aspects.
- **F+U**: This configuration disables the initial pose prediction, relying solely on the proposed probabilistic feature-metric direct image alignment for motion estimation.
- **F+P+U**: This is a version of our method without the dynamic-aware uncertainties and the associated motion segmentation.
- **Ours**: This represents the complete version of our dynamics-aware probabilistic feature-metric tracking system, including all components (**F+P+U+M**).

These variations are applied in a coarse-to-fine optimization process, adhering to the same number of iterations and pyramid levels outlined in our proposed method (see section 4.10. Table 6.1 summarises evaluation results on test sequences of the *TUM RGB-D* [73] and *Bonn RGB-D Dynamic* [74] datasets. Comparisons are exhibited by the Absolute Trajectory Error (ATE) and the Relative Pose Error (RPE_{trans} and RPE_{rot}) described in Section 5.2. It can be seen that the complete system achieves the best results overall across different sequences. Additionally, it can be observed that the accuracy tends to increase when additional components are integrated with the probabilistic feature-metric DIA, meaning every component contributes to enhancing our system.

Sequence	F			F + P			F + U			F + P + U			Ours		
	ATE	RPE_{trans}	RPE_{rot}	ATE	RPE_{trans}	RPE_{rot}	ATE	RPE_{trans}	RPE_{rot}	ATE	RPE_{trans}	RPE_{rot}	ATE	RPE_{trans}	RPE_{rot}
fr1_360	0.1819	0.0094	1.1138	0.1568	0.0101	0.6711	0.1645	0.0046	0.6575	**0.1010**	**0.0037**	**0.6637**	**0.1010**	**0.0037**	**0.6637**
fr1_desk	0.7984	0.0200	0.5985	0.5145	0.0179	0.9084	0.1024	0.0082	0.5961	**0.0792**	**0.0080**	**0.5961**	**0.0792**	**0.0080**	**0.5961**
fr2_desk	1.6240	0.0392	1.2632	1.3240	0.0385	1.0680	0.9729	0.0133	0.4695	**0.9178**	**0.0132**	**0.4713**	**0.9178**	**0.0132**	**0.4713**
fr2_pioneer_360	1.7455	0.2534	2.0123	1.7684	0.0474	1.7020	0.7417	0.0272	0.7302	**0.2534**	**0.0105**	**0.7302**	**0.2534**	**0.0105**	**0.7302**
fr3_walking_static	0.0184	0.0005	0.1917	0.0236	0.0005	0.1817	0.0215	0.0006	0.1982	0.0235	0.7130	0.1566	**0.0226**	**0.3736**	**0.0835**
fr3_walking_xyz	0.2793	0.0049	0.4434	0.2853	0.0047	0.0047	0.2736	0.0045	0.4519	0.2684	1.0564	0.3036	**0.2084**	**0.7294**	**0.2197**
balloon2	0.1711	0.0274	2.5248	0.1665	0.0332	3.3890	0.1355	0.0401	3.6352	0.1448	0.0367	3.5318	**0.0648**	**0.0280**	**2.0559**
balloon_tracking2	0.2683	0.0759	6.6432	0.2885	0.0659	7.7445	0.2793	0.0780	8.5404	0.2877	0.0692	8.6126	**0.2077**	**0.0562**	**7.7626**
crowd3	0.0535	0.0123	2.8557	0.0676	0.0093	2.9696	0.0706	0.0090	3.1230	0.0723	0.0094	3.0891	**0.0323**	**0.0094**	**2.4291**
person_tracking2	0.6632	0.0617	4.7066	0.4305	0.0670	6.0333	0.4907	0.0664	6.4411	0.5492	0.0645	6.3616	**0.4690**	**0.0515**	**5.9916**
placing_nonobs_box	0.1736	0.0236	2.2067	0.1451	0.0167	2.4952	0.1758	0.0169	2.5557	0.1789	0.0177	2.4121	**0.0989**	**0.0161**	**0.6630**
Average	0.5434	0.0480	2.2327	0.4701	0.0283	2.4698	0.3117	0.0244	2.4908	0.2615	0.1820	2.4481	**0.1577**	**0.1806**	**2.7436**

Table 6.1: Ablation study results. We evaluate different combinations of the proposed components on the test sequences using ATE [m], RPR_{trans} [m/f], and RPE_{rot} [Deg/f]. The best results are shown in bold. For the static sequences in *TUM RGB-D* dataset, the full system has identical performance to the F+P+U version because we deactivate the motion segmentation module for static scenes. In this table, it can be observed that we achieved the best performance with our ultimate system after integrating the components one after one.

6.2 Discussion

6.2.1 Convergence Basin

We conducted an experiment visualising the cost landscape to analyze the impact of initial pose prediction in our system, similar to [5]. Given that the pose is a six-dimensional vector, it's not practically feasible to sample costs across all pose components, and visualizing a 6D cost landscape poses significant challenges. Thus, as suggested in [5], we fix 4 degrees of freedom, keeping the rotation and z-translation components constant, and sample only pose combinations at the x and y translations near the ground truth pose. The results of our simulation are shown in Figure 6.1. It can be seen that the pose initialized by the pose network is closer to the ground truth pose, shown as the global minimum in the cost landscape. This reveals that our pose prediction network effectively guides the estimation toward the convergence basin near the global minimum.

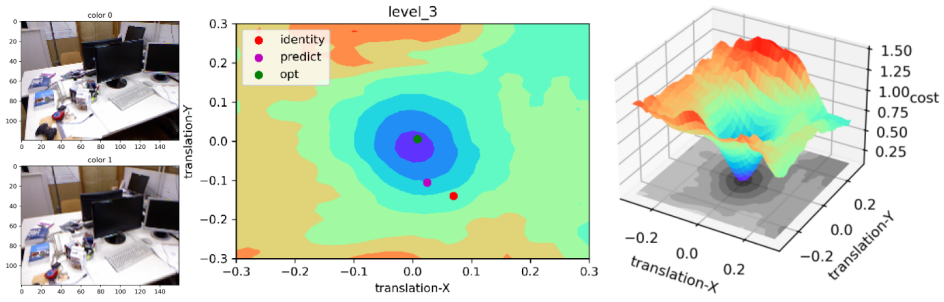


Figure 6.1: Convergence basin. The input image pairs are on the left side, while the 2D and 3D landscapes are on the right. The pose initialized with identity transformation (red), the pose initialized by the pose network (purple), and the final estimation result (green) are projected on the 2D landscape contour shown in the middle.

6.2.2 Limitations of Geometric Segmentation

The first segmentation approach introduced in this thesis is inspired by the geometric segmentation applied in MaskFusion [45], which other researchers have widely used for segmenting objects in RGB-D data [82][83][84][85]. The idea is to generate an edginess map and apply connected-components analysis (CCA) on the edginess map to generate over-segmentation results. The edge is determined by investigating the object's concavity and depth discontinuity.

Despite the quick determination of over-segmented results, we have found crucial limitations of this approach. First, this approach struggles to distinguish physically connecting objects. For instance, it is illustrated in Figure 6.2 that this approach fails to separate a walking person from the background. In this example, the camera captures the entire body of a walking person. On the one hand, the person's feet temporally touch the ground while he is stepping on the floor, which contradicts the first assumption mentioned above. On the other hand, his shoes show no clear concavity in the image due to the observation angle. These properties hypothetically confuse the edginess determination schema (see Figure 6.2f).

The second limitation of this approach is the requirement of the vast engineering effort. Mitigating the problem mentioned above requires careful adjustment of the weighting factor, which balances the contribution of the concavity of objects and discontinuity of depth. However, tuning the coefficient is not trivial and is very case-sensitive. One certain value may perform well in a sequence but fails in the others.

Considering these limitations, we experimented with enhancing the detected edges by infusing an additional input ϕ_e in Equation (4.30):

$$\phi_d + \kappa_c \phi_c + \kappa_e \phi_e > \theta \tag{6.1}$$

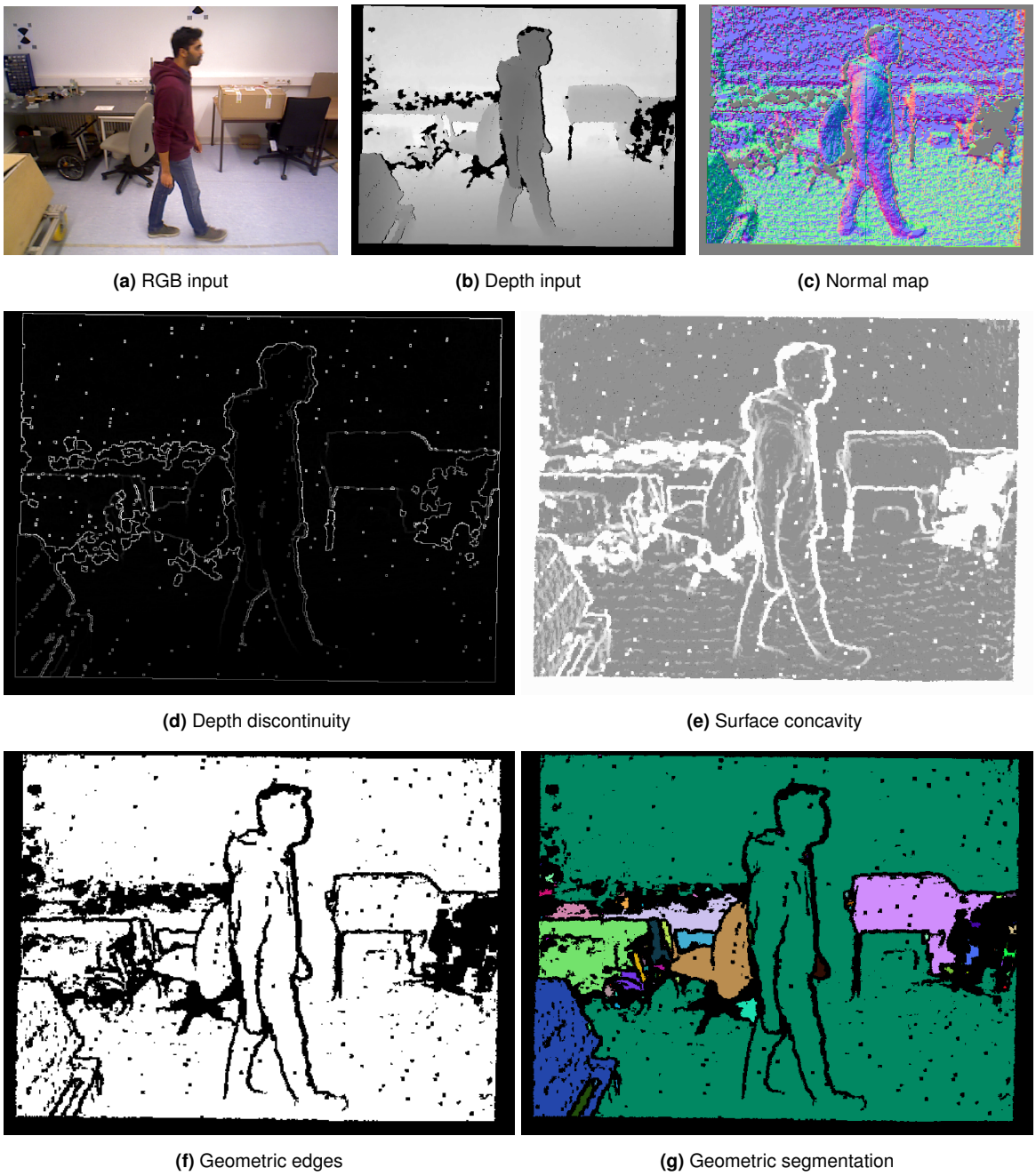
Here, ϕ_e is the edge map generated using the Sobel operator over the input colour image. We also know that when we apply the Sobel convolutional kernels on an image, we obtain the image gradients in the x and y directions. Thus, the edge map can be formulated using the image gradient:

$$\phi_e = \sqrt{\nabla_x \mathbf{I}^2 + \nabla_y \mathbf{I}^2} \tag{6.2}$$

As shown in Figure 6.3, the new term in the Equation allows us to obtain clearer edges. However, the enhanced edges separate the motion initiator, the human, and the triggered moving objects, the balloon. This separation violates our intention of merging the geometric and semantic instance segmentation to maintain active and passive dynamic objects. Moreover, the additional edge extraction term and the associated coefficient complicate Equation (4.30) further, making the parameter tuning more difficult.

6.2.3 Comparing Edge-based Segmentation to Geometric Segmentation

Inheriting the idea of studying the rich information of input data from [45], as the second experiment, we directly employ a CNN model called DexiNed [86], allowing for robust edge detection. It takes a colour image as input and generates fine and detailed edginess maps. Compared to the geometric edges extraction used in MaskFusion [45], DexiNed produces more consistent boundary detection, preventing parameter adjustment for various scenarios. Likewise, analyzing the connected components of the edginess map allows us to over-segment the images (see Figure 6.4). This is referred to as edge-based segmentation in the remainder



(a) RGB input **(b)** Depth input **(c)** Normal map

(d) Depth discontinuity **(e)** Surface concavity

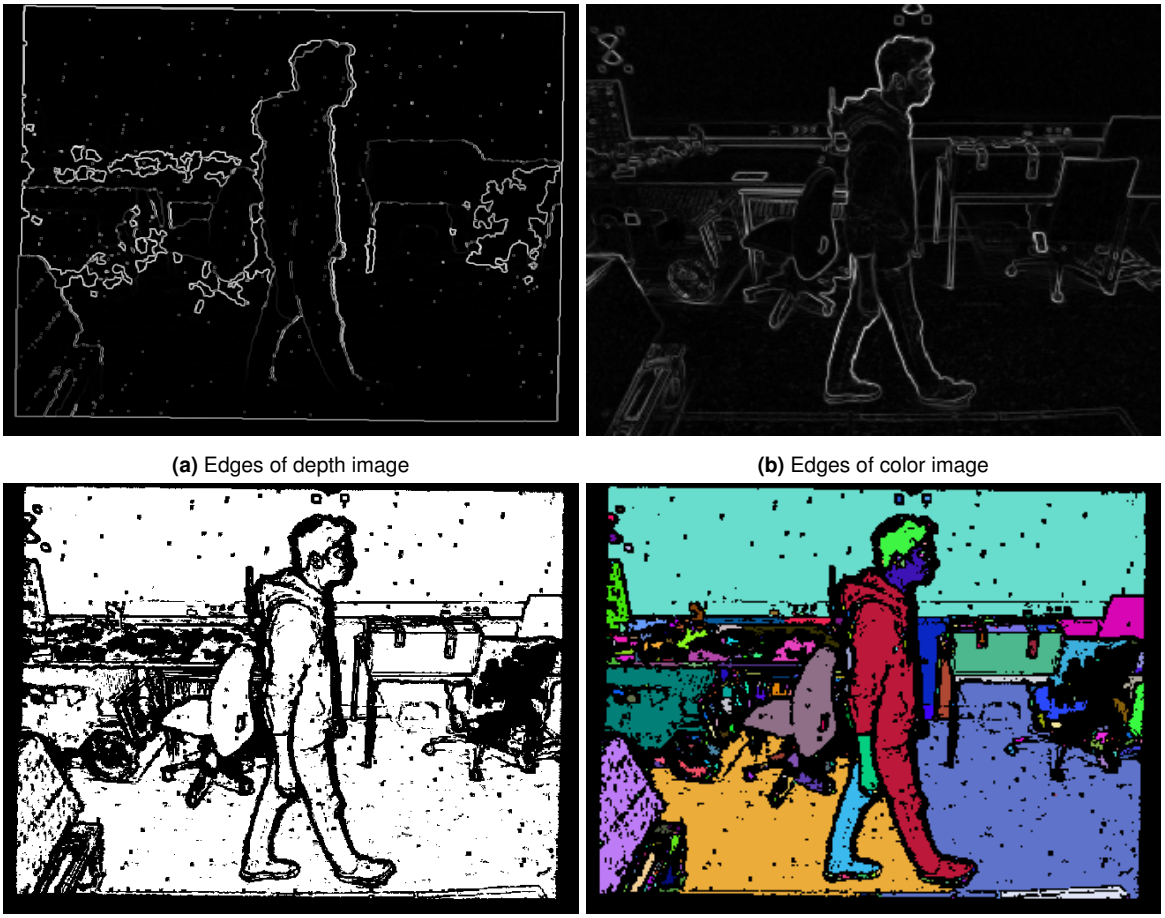
(f) Geometric edges **(g)** Geometric segmentation

Figure 6.2: Geometric segmentation results using methods introduced in MaskFusion [45]. (a) and (b) show the RGB and depth input of the sequence *person_tracking* from the *Bonn RGB-D Dynamic Dataset*. Visualization of the vertex map is the same as the input depth, while the normal map is shown in (C). Additionally, (d) and (e) illustrate the depth discontinuity and surface concavity terms accordingly. The combined geometric edginess map is depicted in (f), and (d) visualizes the final segmentation after applying CCA.

of this thesis.

Instead of studying the depth discontinuity and the surface concavity, we directly exploit the object boundaries by leveraging a state-of-the-art edge detection deep learning method, DexiNed [86]. Likewise, we apply the connected component analysis on the generated edges to obtain an over-segmented image. The method we choose has better capability in handling the aforementioned issue, succeeding in segmenting the person from the background. Results

are shown in Figure 6.4



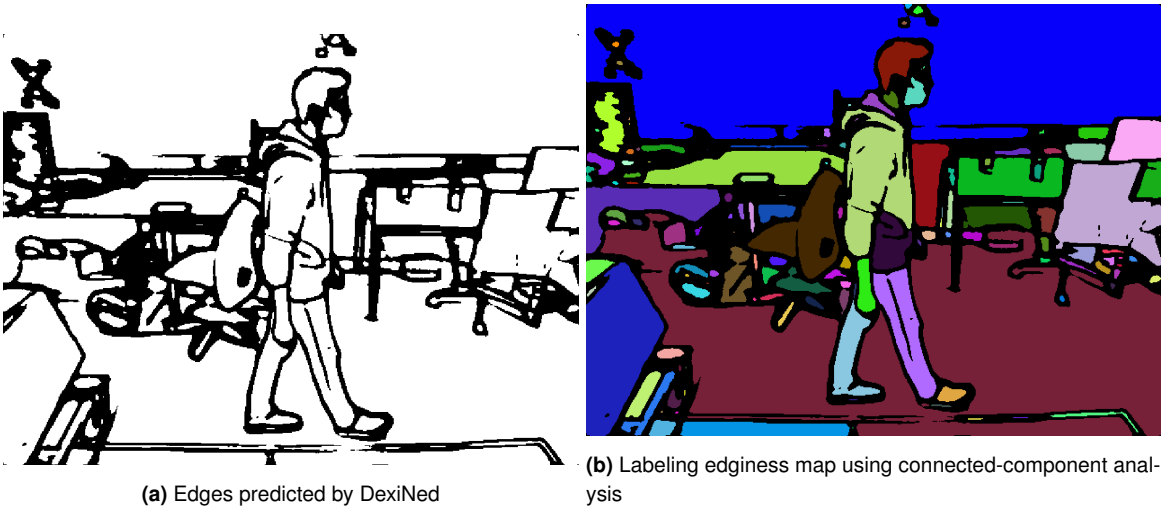
(a) Edges of depth image

(b) Edges of color image

(c) Improved edges

(d) Improved results

Figure 6.3: Improved segmentation results using augmented edges.



(a) Edges predicted by DexiNed

(b) Labeling edginess map using connected-component analysis

Figure 6.4: Segmentation results using the deep edges detector and connected-component analysis. (a) illustrates the edges detected by DexiNed [86], while (d) visualizes the final over-segmented image by labelling the connected components of the edginess map.

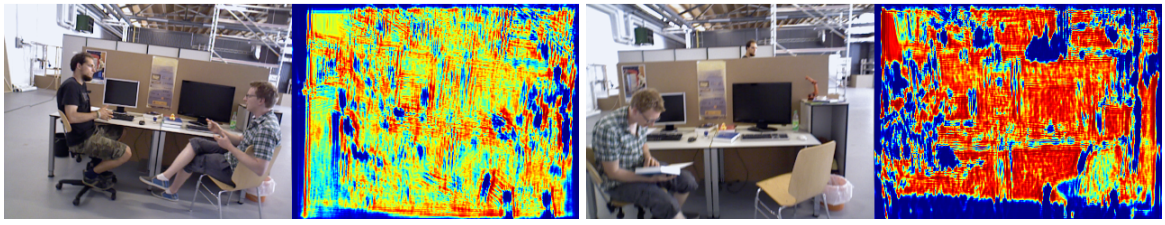
Disparate from the geometric segmentation method, using DexiNed is practical and easy

to scale in various scenarios. The edges DexiNed extracted are less noisy than the augmented approach, making it easier to obtain smooth results by applying CCA. More than that, using neural networks helps maintain the pipeline's differentiability, enabling learning more task-specific weights by training for downstream tasks end-to-end.

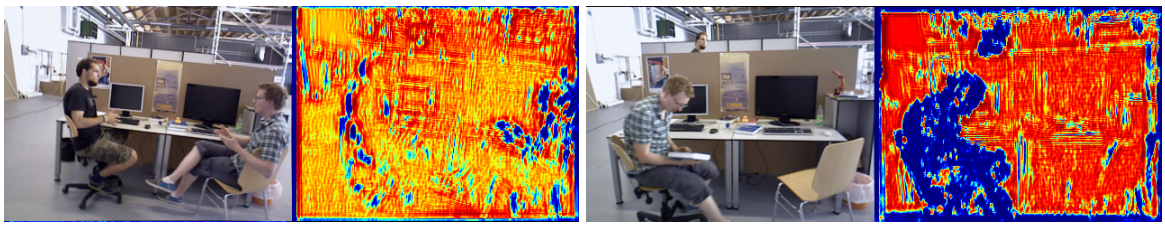
Although edge-based segmentation combining DexiNed and CCA can achieve results comparable to augmented geometric segmentation, it suffers from individual limitations. Similar to the augmented approach, the extracted edges by DexiNed split the active and passive moving objects. In this case, fusing this over-segmentation with the segmentation results of Mask R-CNN can not generate a segmentation result that includes both the motion initiator and moving objects triggered by them. As we use Mask R-CNN as a source of strong prior information, applying the segmentation merge approach introduced in Section 4.7.1 can solely obtain the instances identified by Mask R-CNN with finer boundaries. However, Results like this cannot fulfil our intention, so we do not integrate this approach into the pipeline.

6.2.4 Benefits of Two-Stage Training

Uncertainty can result from different influence factors, including noisy sensory data, challenging illumination, and moving objects. We train our model using a two-stage schema to decouple the uncertainty resulting from illumination and dynamic pixels. A control model is also trained in one stage without separating static and dynamic scenario sequences. After that, we visualized the uncertainties on a dynamic sequence predicted by these models individually in Figure 6.5. As shown in Figure 6.5a, uncertainty maps generated by the model trained using one-stage appear uncertain about regions under lighting changes and capturing moving instances. On the other hand, the model trained with two stages concentrates more on the dynamic regions after the second training stage using only dynamic sequences (see 6.5b).



(a) One-stage training



(b) Two-stage training (second stage)

Figure 6.5: Uncertainty maps. This figure visualizes the input colour image and the respective uncertainty maps estimated by models trained differently. Regions that the model is more certain about are coloured more reddish. Inversely, uncertain regions are more blueish.

6.2.5 Limitation of Uncertainty-based Segmentation

The accuracy of the prompts fed to the FastSAM model [11] is significant for producing high-quality segmentation masks. Besides, how we prompt the segmentation is crucial in precisely identifying individual instances. The proposed uncertainty-based approach highly relies on the sensitivity and accuracy of the motion-aware uncertainties. Ideally, it has to be agile enough to detect every perturbation in residuals caused by dynamic factors and, meanwhile, accurate enough to localize the dynamic pixels precisely with low uncertainties. However, despite a considerable amount of train data, the uncertainty maps predicted by our model often contain noises, leading to failure in motion segmentation as shown in Figure 6.6.

Figure 6.6: Failure case in uncertainty-based segmentation. As the dense uncertainty map is noisy for various reasons, incorrect points remain in the prompt set, located outside the dynamic object, in this case, the moving person. These points lead to the segmentation of the static background.

However, it is not trivial to pinpoint the elements that trigger false positives. Numerous factors can result in uncertain information for image alignment, including appearance variety due to seasonal or illumination changes, occlusion caused by moving objects in scenes, and inherent noises of sensors. The uncertainty maps are trained on gradient flows propagated from the 3D EPE, which means that their training constraints are not explicitly defined and might include factors other than dynamic pixels. Although we can mitigate the effect of other factors using a two-stage training schema, we cannot fully decouple the uncertainties caused by the motion of third parties from all possible influences.

A potential solution could be training the uncertainty maps with annotated data, e.g. ground truth motion segmentation. The ground truth segmentation can serve as prior uncertainties as we can assign the static pixels with the highest certainty 1 and the dynamics with 0. This prior information can potentially guide the uncertainty encoder to concentrate on the dynamic regions. Besides, substituting the predicted camera poses with the ground truth poses would also be helpful. Using ground truth camera poses, we can expect the 3D EPE to be around zero for all static pixels, while the errors for pixels capturing motion are much higher. In this way, the uncertainty encoder can learn to dampen the impact of the dynamic regions more effectively.

In addition, better approaches to generating prompts would be helpful in improving the accuracy of the predicted dense uncertainties. The pooling strategy in the current approach ensures the pipeline's differentiability, but it is ineffective in many cases. For one thing, it generates prompts for every local patch of the image plane, which is the main source of false positives. Thresholding them with respect to their certainty values can mitigate the issue in the inference time, as we show on the Co-Fusion datasets (Section 4.7.2). For another thing, clustering the points using geometric cues can bring a significant performance boost in terms of time and accuracy. In the current approach, prompt points of foreground objects are fed individually to the segmentation model. The generated masks are merged by calculating the

relative overlay of every possible combination of all candidates. Hence, applying clustering can potentially improve our system's performance. To preserve the pipeline's differentiability, differentiable clustering methods [87] can be exploited.

6.3 Other Limitations

Several other limitations remain except for the limitations discussed in the previous sections. For instance, the moving object segmentation masks are hard-associated with pixels, resulting in extremely sparse gradient flows in the computational graph of our pipeline. That said, our pipeline is merely locally differentiable.

Additionally, the proposed uncertainty-based segmentation approach fully relies on FastSAM [11]. Although FastSAM improves the inference speed by about 50 times faster than SAM [11], we obtained an average inference time of around 480 ms per frame in our experiments using the *TUM RGB-D* dataset. This long processing time limits the possibility of deploying our system in real-time robotic tasks.

Furthermore, our approach encounter limitations that hinder it from achieving comparable performance to the state-of-the-art dynamic SLAM methods. Though it is validated that the feature-metric visual odometry in the pipeline has promising advantages in estimating relative poses, it cannot overcome the accumulated drift over time, resulting in relatively high ATE. One key reason is that our pipeline lacks back-end optimization, a crucial component of SLAM's superior performance. Another reason is that we prevented many engineering factors that could boost the performance yet damage the pipeline's differentiability.

6.4 Future Work

During the development and evaluation of the algorithm outlined in this thesis, we have pinpointed multiple avenues for future research. These avenues encompass:

- Small or distant objects occupy only a few pixels. These objects offer limited geometric and photometric information in image frames, affecting both the tracking and segmentation modules. Tracking them can result in ignorance or errors in motion estimation. Particularly, applying image alignment on them in a coarse-to-fine manner can hardly succeed. Even for feature-based methods, which rely on the correspondence matching, tracking visibly small objects in image sequences is also very challenging. Developing more agile algorithms can potentially address these limitations.

- The two proposed alternative moving instance segmentation approaches showed individual limitations in our experiment as discussed in Section 6.2. Neither shows satisfactory stability and reliability in segmenting all moving entities in scenes. In this thesis, we attempted to dive into the up-to-date unified segmentation models [52][53][11], but our designed approach fails to provide temporal consistent segmentation, which leads to inaccurate or failed object tracking. Revisiting the proposed moving instance segmentation method opens up opportunities for future work. One improvement can be achieved by studying dynamic-aware uncertainties, seeking a better way to formulate and train the deep uncertainty estimation [88][89][90]. Possible improvements could be prompting FastSAM with segmentation masks predicted in the previous frame to generate consistent segmentation across frames [52][11].

- The differentiable mapping and map fusion component, namely ∇PointFusion, in our pipeline is fully adopted from ∇SLAM [1]. However, this component is a differentiable approximation of the original method [19], having no trainable parameters and inheriting limitations from its predecessor. In addition to previously widely used volumetric or point-based representation in mapping, the implicit neural radian field (NeRF)[91] shows its great potential in many recent works [38][39] [92][93]. More recently, [94] have proposed 3D Gaussian Splatting (3DGS), showing improved properties compared to NeRF. To the best of the author's knowledge, [95] and [96] are the first works that apply 3DGS for scene rendering and refining camera tracking in dense SLAM for monocular and RGB-D camera, achieving convincing results.

  NeRF and 3DGS are intentionally introduced for differentiable rendering. Continuous learning allows them to complete the object's shape, predicting their backside without seeing it. Their cutting-edge rendering performance and, more importantly, their differentiability make them appealing research topics in the scope of differentiable visual SLAM.

- The current implementation is a prototype built with high simplicity for testing and experimenting. Thus, it still struggles to work in real time due to the computational bottleneck in the optimization solver and the involved segmentation models. The former can be tackled using the CUDA accelerated implementation similar to MaskFusion [45] and Co-Fusion [49]. However, the latter involves further research since the state-of-the-art segmentation approach mostly trades computational time for accuracy, and their inference time is incompatible with real-time applications.

  Past works mitigate this issue by running deep-learning segmentation in a keyframe-based manner and involving a frame-wise geometrical segmentation approach [45]. Nevertheless, the geometrical segmentation approach requires empirically hand-crafted parameters to generate reasonable yet reliable segmentation. Reliable geometrical segmentation approaches and more efficient neural networks for segmentation will be researched.

- Further research in the dynamic visual odometry and dense, dynamic SLAM is fundamentally limited by a lack of suitable datasets. Although we comprehensively evaluated the proposed systems by combining results from multiple datasets, we are bound when training our systems. The datasets we use and other existing datasets have their own limitations. For instance, the TartanAir dataset [42] covers a wide range of indoor and outdoor scenes, yet it includes a limited amount of moving objects in its sequences. Although the recent AirDOS Dataset [97] offers some improvements in this aspect, both of these two synthetic datasets are too challenging. Their cameras attached to drones undergo extremely dynamic, 6 DoF motions. In the case of the AirDOS dataset especially, the captured environments are highly dynamic and cluttered with articulated objects - humans.

  In addition to synthetic data, the urge for comprehensive real-world datasets for researching dynamic dense SLAM cannot be ignored. Learning-based methods, particularly those that involve supervised learning, are data-hungry. Generally, no satisfying performance can be expected without a comparable amount of data fed at the learning stage. Hence, the further elaboration of more real-world datasets and improved, closer-to-reality synthetic datasets would allow for advancing the development of methods in dense dynamic SLAM

- Given the lack of available data for supervised training, self-supervised visual representation learning is gaining significant interest, particularly for its application in depth and motion estimations. Some approaches define losses by manipulating the data without explicit labels. These manipulations include techniques such as predicting spatial transformations [98][99] and reconstructing input data [100][101][33] etc.

  Our feature-metric direct image alignment component is data-driven like any other learning-based method. Its performance heavily depends on the amount of data on which it is trained. However, despite the increasing popularity of SLAM relative research, we still encounter a tight availability of labelled data. An interesting research direction is to develop the visual odometry component so that we can train it in a self-supervised manner without concern for data shortage.

- The proposed differentiable SLAM framework benefits not only SLAM or related computer vision tasks but also robotic tasks such as object manipulation and robot navigation. Our pipeline's differentiability enables training downstream components end-to-end, propagating the loss back to the input sensory data. Experiments to illustrate this capability of our pipeline can solidify our idea and encourage follow-up work.

Chapter 7

Conclusion

In this master's thesis, we have presented an automatically differentiable dense visual SLAM pipeline for RGB-D image, capable of tracking and reconstructing moving objects in dynamic environments while maintaining robust camera trajectory estimation. Based on the existing ∇SLAM [1] framework, the pipeline is designed by chaining multiple differentiable components. For robust tracking, we have empowered DIA with deep learning, expanding 3-dimensional image space to high-dimensional latent space, and proposed probabilistic feature-metric DIA. The pose estimation is carried out by the differentiable LM-solver and strengthened using the deep M-Estimator. Additionally, a pose network that operates on the coarsest feature map is adopted to provide reliable pose initialization to the LM-solver.

Furthermore, we have provided two segmentation approaches. Both can generate reasonable segmentation masks that can be applied to eliminate dynamic pixels. In particular, based on the newly proposed FastSAM [11], we have proposed an uncertainty-based segmentation approach, invoking the dynamic-aware uncertainties predicted by a CNN in the pipeline, is capable to generate instance segmentation that is applicable for object tracking. To guarantee the differentiability of the whole pipeline, we further employed the Deep Hungarian Network [67] to associate segmented objects. Last but not least, a 4D reconstruction of the scene containing objects moving over time is generated by the differentiable PointFusion.

This pipeline, as the key contribution of this thesis, can be trained end-to-end. The modular design with easily replaceable components empowers the pipeline to be adapted and used for training deep-learning models for various tasks, ranging from upstream tasks, such as motion segmentation, to learning-based visual odometry. Moreover, the pipeline can be integrated with other trainable models, such as networks designed for robot manipulation and navigation. This key property of the pipeline is capable to enhance deep-learning-based algorithmic development in the robotic applications.

The primary intention of this work is not to surpass any existing dynamic SLAM systems but rather to offer another possibility for the development of visual SLAM in cooperation with other robotic tasks by proposing an end-to-end pipeline. Therefore, the performance bias compared to existing dynamic visual SLAM systems explained by the experiment results is within our expectations. The conducted experiments have shown the effectiveness of pipelines in challenging dynamic environments and revealed the contribution of individual components. Although the proposed approach conveys competitive performance in challenging dynamic environments, it still lags behind the state-of-the-art dynamic SLAM methods by a significant gap.

Nevertheless, the modular design of our pipeline allows us to further experiment and replace components with less effort, enabling future extension of our work. Within this thesis, the great advantage of an end-to-end differentiable pipeline, enabling gradients from

ultimate loss to flow back to input images, has not been demonstrated. A comprehensive verification of the pipeline's differentiability and adequate demonstration of its potential should be the focus of the subsequent research.

In conclusion, this thesis demonstrates the possibility of estimating ego camera motion and the motions of dynamic objects in scenes and reconstructing the observed environment using RGB-D images in a fully differentiable system. We have used probabilistic feature-metric direct image alignment for accurate and robust motion estimation. Concurrently, an uncertainty-based motion segmentation approach has been proposed based on the uncertainties in feature-metric visual odometry. Through experiments, we have demonstrated the effectiveness of our method in tracking camera and dynamic object motion and reconstructing dynamic environments in representative scenarios. Ultimately, the crucial limitations in the proposed approach has been pinpointed, highlighting vital areas for continued research. Overall, we believe our work is an important stepping stone for investigating a fully differentiable framework to address visual dynamic SLAM.

List of Figures

List of Tables

Bibliography

[1] Jatavallabhula, K. M. et al. *gradSLAM: Automagically differentiable SLAM*. Nov. 19, 2020. arXiv: 1910.10672[cs]. URL: http://arxiv.org/abs/1910.10672 (visited on 08/10/2022).

[2] Chen, W. et al. "An Overview on Visual SLAM: From Tradition to Semantic". In: *Remote Sensing* 14.13 (June 23, 2022), p. 3010. ISSN: 2072-4292. DOI: 10.3390/rs14133010. URL: https://www.mdpi.com/2072-4292/14/13/3010 (visited on 12/02/2022).

[3] Engel, J. et al. "Direct Sparse Odometry". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.3 (Mar. 1, 2018), pp. 611–625. ISSN: 0162-8828, 2160-9292. DOI: 10.1109/TPAMI.2017.2658577. URL: http://ieeexplore.ieee.org/document/7898369/ (visited on 12/05/2022).

[4] Zubizarreta, J. et al. "Direct Sparse Mapping". In: *IEEE Transactions on Robotics* 36.4 (Aug. 2020), pp. 1363–1370. ISSN: 1552-3098, 1941-0468. DOI: 10.1109/TRO.2020.2991614. arXiv: 1904.06577[cs]. URL: http://arxiv.org/abs/1904.06577 (visited on 01/19/2023).

[5] Xu, B. et al. "Deep Probabilistic Feature-Metric Tracking". In: *IEEE Robotics and Automation Letters* 6.1 (Jan. 2021). Conference Name: IEEE Robotics and Automation Letters, pp. 223–230. ISSN: 2377-3766. DOI: 10.1109/LRA.2020.3039216.

[6] Teed, Z. et al. *DROID-SLAM: Deep Visual SLAM for Monocular, Stereo, and RGB-D Cameras*. Feb. 2, 2022. DOI: 10.48550/arXiv.2108.10869. arXiv: 2108.10869[cs]. URL: http://arxiv.org/abs/2108.10869 (visited on 01/03/2023).

[7] Baker, S. et al. "Lucas-Kanade 20 Years On: Part 5". In: ().

[8] Stumberg, L. von et al. *GN-Net: The Gauss-Newton Loss for Multi-Weather Relocalization*. Nov. 27, 2019. DOI: 10.48550/arXiv.1904.11932. arXiv: 1904.11932[cs]. URL: http://arxiv.org/abs/1904.11932 (visited on 01/16/2023).

[9] Sarlin, P.-E. et al. *Back to the Feature: Learning Robust Camera Localization from Pixels to Pose*. Apr. 7, 2021. DOI: 10.48550/arXiv.2103.09213. arXiv: 2103.09213[cs]. URL: http://arxiv.org/abs/2103.09213 (visited on 08/10/2023).

[10] He, K. et al. *Mask R-CNN*. Jan. 24, 2018. DOI: 10.48550/arXiv.1703.06870. arXiv: 1703.06870[cs]. URL: http://arxiv.org/abs/1703.06870 (visited on 07/18/2023).

[11] Zhao, X. et al. *Fast Segment Anything*. June 21, 2023. arXiv: 2306.12156[cs]. URL: http://arxiv.org/abs/2306.12156 (visited on 12/07/2023).

[12] Engel, J. et al. "LSD-SLAM: Large-Scale Direct Monocular SLAM". In: *Computer Vision – ECCV 2014*. Ed. by Fleet, D. et al. Vol. 8690. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2014, pp. 834–849. ISBN: 978-3-319-10604-5 978-3-319-10605-2. DOI: 10.1007/978-3-319-10605-2_54. URL: http://link.springer.com/10.1007/978-3-319-10605-2_54 (visited on 12/04/2022).

[13] Engel, J. et al. *Direct Sparse Odometry*. Oct. 7, 2016. DOI: 10.48550/arXiv.1607.02565. arXiv: 1607.02565[cs]. URL: http://arxiv.org/abs/1607.02565 (visited on 11/21/2023).

[14] Gao, X. et al. *LDSO: Direct Sparse Odometry with Loop Closure*. Aug. 3, 2018. DOI: 10.48550/arXiv.1808.01111. arXiv: 1808.01111[cs]. URL: http://arxiv.org/abs/1808.01111 (visited on 12/05/2022).

[15] Roth, H. et al. "Moving Volume KinectFusion". In: *Procedings of the British Machine Vision Conference 2012*. British Machine Vision Conference 2012. Surrey: British Machine Vision Association, 2012, pp. 112.1–112.11. ISBN: 978-1-901725-46-9. DOI: 10.5244/C.26.112. URL: http://www.bmva.org/bmvc/2012/BMVC/paper112/index.html (visited on 07/18/2023).

[16] Xu, B. et al. *MID-Fusion: Octree-based Object-Level Multi-Instance Dynamic SLAM*. arXiv.org. Dec. 19, 2018. URL: https://arxiv.org/abs/1812.07976v4 (visited on 06/18/2023).

[17] Huang, J. et al. *ClusterVO: Clustering Moving Instances and Estimating Visual Odometry for Self and Surroundings*. Mar. 29, 2020. DOI: 10.48550/arXiv.2003.12980. arXiv: 2003.12980[cs]. URL: http://arxiv.org/abs/2003.12980 (visited on 03/09/2023).

[18] Pfister, H. et al. "Surfels: surface elements as rendering primitives". In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques - SIGGRAPH '00*. the 27th annual conference. Not Known: ACM Press, 2000, pp. 335–342. ISBN: 978-1-58113-208-3. DOI: 10.1145/344779.344936. URL: http://portal.acm.org/citation.cfm?doid=344779.344936 (visited on 12/02/2022).

[19] Keller, M. et al. "Real-Time 3D Reconstruction in Dynamic Scenes Using Point-Based Fusion". In: *2013 International Conference on 3D Vision*. 2013 International Conference on 3D Vision (3DV). Seattle, WA, USA: IEEE, June 2013, pp. 1–8. ISBN: 978-0-7695-5067-1. DOI: 10.1109/3DV.2013.9. URL: http://ieeexplore.ieee.org/document/6599048/ (visited on 11/25/2022).

[20] Whelan, T. et al. "ElasticFusion: Dense SLAM Without A Pose Graph". In: *Robotics: Science and Systems XI*. Robotics: Science and Systems 2015. Robotics: Science and Systems Foundation, July 13, 2015. ISBN: 978-0-9923747-1-6. DOI: 10.15607/RSS.2015.XI.001. URL: http://www.roboticsproceedings.org/rss11/p01.pdf (visited on 12/02/2022).

[21] DeTone, D. et al. *SuperPoint: Self-Supervised Interest Point Detection and Description*. Apr. 19, 2018. DOI: 10.48550/arXiv.1712.07629. arXiv: 1712.07629[cs]. URL: http://arxiv.org/abs/1712.07629 (visited on 07/18/2023).

[22] Sarlin, P.-E. et al. *SuperGlue: Learning Feature Matching with Graph Neural Networks*. Mar. 28, 2020. DOI: 10.48550/arXiv.1911.11763. arXiv: 1911.11763[cs]. URL: http://arxiv.org/abs/1911.11763 (visited on 07/18/2023).

[23] Teed, Z. et al. "RAFT: Recurrent All-Pairs Field Transforms for Optical Flow". In: *arXiv:2003.12039 [cs]* (Aug. 25, 2020). arXiv: 2003.12039. URL: http://arxiv.org/abs/2003.12039 (visited on 01/05/2022).

[24] Teed, Z. et al. *DeepV2D: Video to Depth with Differentiable Structure from Motion*. Apr. 27, 2020. DOI: 10.48550/arXiv.1812.04605. arXiv: 1812.04605[cs]. URL: http://arxiv.org/abs/1812.04605 (visited on 07/06/2023).

[25] Choi, J. et al. *SelfTune: Metrically Scaled Monocular Depth Estimation through Self-Supervised Learning*. Mar. 10, 2022. arXiv: 2203.05332[cs]. URL: http://arxiv.org/abs/2203.05332 (visited on 12/02/2022).

[26] Tateno, K. et al. *CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction*. Apr. 11, 2017. arXiv: 1704.03489[cs]. URL: http://arxiv.org/abs/1704.03489 (visited on 11/23/2022).

[27] Chen, C. et al. *A Survey on Deep Learning for Localization and Mapping: Towards the Age of Spatial Machine Intelligence*. June 29, 2020. arXiv: 2006.12567[cs,eess]. URL: http://arxiv.org/abs/2006.12567 (visited on 11/23/2022).

[28] Chen, K. et al. *Semantic Visual Simultaneous Localization and Mapping: A Survey*. Sept. 14, 2022. DOI: 10.48550/arXiv.2209.06428. arXiv: 2209.06428[cs]. URL: http://arxiv.org/abs/2209.06428 (visited on 07/17/2023).

[29] Pu, H. et al. "Visual SLAM Integration With Semantic Segmentation and Deep Learning: A Review". In: *IEEE Sensors Journal* 23.19 (Oct. 2023). Conference Name: IEEE Sensors Journal, pp. 22119–22138. ISSN: 1558-1748. DOI: 10.1109/JSEN.2023.3306371. URL: https://ieeexplore.ieee.org/document/10227894 (visited on 11/21/2023).

[30] Saputra, M. R. U. et al. "Visual SLAM and Structure from Motion in Dynamic Environments: A Survey". In: *ACM Computing Surveys* 51.2 (Mar. 31, 2019), pp. 1–36. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3177853. URL: https://dl.acm.org/doi/10.1145/3177853 (visited on 11/16/2023).

[31] Bloesch, M. et al. *CodeSLAM - Learning a Compact, Optimisable Representation for Dense Visual SLAM*. Apr. 14, 2019. DOI: 10.48550/arXiv.1804.00874. arXiv: 1804.00874[cs]. URL: http://arxiv.org/abs/1804.00874 (visited on 01/20/2023).

[32] Stumberg, L. von et al. *LM-Reloc: Levenberg-Marquardt Based Direct Visual Relocalization*. Oct. 13, 2020. arXiv: 2010.06323[cs]. URL: http://arxiv.org/abs/2010.06323 (visited on 12/18/2023).

[33] Shu, C. et al. *Feature-metric Loss for Self-supervised Learning of Depth and Egomotion*. July 21, 2020. DOI: 10.48550/arXiv.2007.10603. arXiv: 2007.10603[cs]. URL: http://arxiv.org/abs/2007.10603 (visited on 06/29/2023).

[34] Lv, Z. et al. *Taking a Deeper Look at the Inverse Compositional Algorithm*. Apr. 8, 2019. DOI: 10.48550/arXiv.1812.06861. arXiv: 1812.06861[cs]. URL: http://arxiv.org/abs/1812.06861 (visited on 07/19/2023).

[35] Zhou, H. et al. *DeepTAM: Deep Tracking and Mapping*. Aug. 7, 2018. DOI: 10.48550/arXiv.1808.01900. arXiv: 1808.01900[cs]. URL: http://arxiv.org/abs/1808.01900 (visited on 07/18/2023).

[36] Tang, C. et al. *BA-Net: Dense Bundle Adjustment Network*. Aug. 25, 2019. DOI: 10.48550/arXiv.1806.04807. arXiv: 1806.04807[cs]. URL: http://arxiv.org/abs/1806.04807 (visited on 04/17/2023).

[37] Clark, R. et al. *LS-Net: Learning to Solve Nonlinear Least Squares for Monocular Stereo*. Sept. 9, 2018. DOI: 10.48550/arXiv.1809.02966. arXiv: 1809.02966[cs]. URL: http://arxiv.org/abs/1809.02966 (visited on 10/27/2023).

[38] Sucar, E. et al. *iMAP: Implicit Mapping and Positioning in Real-Time*. Sept. 13, 2021. DOI: 10.48550/arXiv.2103.12352. arXiv: 2103.12352[cs]. URL: http://arxiv.org/abs/2103.12352 (visited on 12/07/2023).

[39] Zhu, Z. et al. "NICE-SLAM: Neural Implicit Scalable Encoding for SLAM". In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). New Orleans, LA, USA: IEEE, June 2022, pp. 12776–12786. ISBN: 978-1-66546-946-3. DOI: 10.1109/CVPR52688.2022.01245. URL: https://ieeexplore.ieee.org/document/9878912/ (visited on 11/23/2022).

[40] Zhu, Z. et al. *NICER-SLAM: Neural Implicit Scene Encoding for RGB SLAM*. arXiv.org. Feb. 7, 2023. URL: https://arxiv.org/abs/2302.03594v1 (visited on 12/10/2023).

[41] Burri, M. et al. "The EuRoC micro aerial vehicle datasets". In: *The International Journal of Robotics Research* 35.10 (Sept. 1, 2016). Publisher: SAGE Publications Ltd STM, pp. 1157–1163. ISSN: 0278-3649. DOI: 10.1177/0278364915620033. URL: https://doi.org/10.1177/0278364915620033 (visited on 01/05/2022).

[42] Wang, W. et al. *TartanAir: A Dataset to Push the Limits of Visual SLAM*. Aug. 8, 2020. DOI: 10.48550/arXiv.2003.14338. arXiv: 2003.14338[cs]. URL: http://arxiv.org/abs/2003.14338 (visited on 06/29/2023).

[43] Zhang, T. et al. *FlowFusion: Dynamic Dense RGB-D SLAM Based on Optical Flow*. Mar. 11, 2020. arXiv: 2003.05102[cs]. URL: http://arxiv.org/abs/2003.05102 (visited on 10/29/2023).

[44] Chang, J. et al. "A Real-Time Dynamic Object Segmentation Framework for SLAM System in Dynamic Scenes". In: *IEEE Transactions on Instrumentation and Measurement* 70 (2021). Conference Name: IEEE Transactions on Instrumentation and Measurement, pp. 1–9. ISSN: 1557-9662. DOI: 10.1109/TIM.2021.3109718. URL: https://ieeexplore.ieee.org/abstract/document/9527213/authors#authors (visited on 11/16/2023).

[45] Rünz, M. et al. *MaskFusion: Real-Time Recognition, Tracking and Reconstruction of Multiple Moving Objects*. Oct. 22, 2018. arXiv: 1804.09194[cs]. URL: http://arxiv.org/abs/1804.09194 (visited on 08/10/2022).

[46] Yang, G. et al. *Learning to Segment Rigid Motions from Two Frames*. Jan. 10, 2021. DOI: 10.48550/arXiv.2101.03694. arXiv: 2101.03694[cs]. URL: http://arxiv.org/abs/2101.03694 (visited on 07/04/2023).

[47] Ye, W. et al. *DeFlowSLAM: Self-Supervised Scene Motion Decomposition for Dynamic Dense SLAM*. Jan. 13, 2023. DOI: 10.48550/arXiv.2207.08794. arXiv: 2207.08794[cs]. URL: http://arxiv.org/abs/2207.08794 (visited on 11/16/2023).

[48] Bescos, B. et al. "DynaSLAM: Tracking, Mapping and Inpainting in Dynamic Scenes". In: *IEEE Robotics and Automation Letters* 3.4 (Oct. 2018), pp. 4076–4083. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2018.2860039. arXiv: 1806.05620[cs]. URL: http://arxiv.org/abs/1806.05620 (visited on 12/12/2022).

[49] Rünz, M. et al. "Co-Fusion: Real-time Segmentation, Tracking and Fusion of Multiple Objects". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. May 2017, pp. 4471–4478. DOI: 10.1109/ICRA.2017.7989518. arXiv: 1706.06629[cs]. URL: http://arxiv.org/abs/1706.06629 (visited on 12/05/2022).

[50] Chen, L.-C. et al. *DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs*. May 11, 2017. DOI: 10.48550/arXiv.1606.00915. arXiv: 1606.00915[cs]. URL: http://arxiv.org/abs/1606.00915 (visited on 07/18/2023).

[51] Wang, X. et al. *Cut and Learn for Unsupervised Object Detection and Instance Segmentation*. Jan. 26, 2023. DOI: 10.48550/arXiv.2301.11320. arXiv: 2301.11320[cs]. URL: http://arxiv.org/abs/2301.11320 (visited on 09/28/2023).

[52] Kirillov, A. et al. *Segment Anything*. Apr. 5, 2023. DOI: 10.48550/arXiv.2304.02643. arXiv: 2304.02643[cs]. URL: http://arxiv.org/abs/2304.02643 (visited on 12/07/2023).

[53] Zhang, C. et al. *Faster Segment Anything: Towards Lightweight SAM for Mobile Applications*. July 1, 2023. arXiv: 2306.14289[cs]. URL: http://arxiv.org/abs/2306.14289 (visited on 12/07/2023).

[54] Ballester, I. et al. *DOT: Dynamic Object Tracking for Visual SLAM*. Sept. 30, 2020. DOI: 10.48550/arXiv.2010.00052. arXiv: 2010.00052[cs]. URL: http://arxiv.org/abs/2010.00052 (visited on 07/18/2023).

[55] He, J. et al. "OVD-SLAM: An Online Visual SLAM for Dynamic Environments". In: *IEEE Sensors Journal* 23.12 (June 15, 2023), pp. 13210–13219. ISSN: 1530-437X, 1558-1748, 2379-9153. DOI: 10.1109/JSEN.2023.3270534. URL: https://ieeexplore.ieee.org/document/10113832/ (visited on 11/19/2023).

[56] Campos, C. et al. "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM". In: *IEEE Transactions on Robotics* 37.6 (Dec. 2021), pp. 1874–1890. ISSN: 1552-3098, 1941-0468. DOI: 10.1109/TRO.2021.3075644. arXiv: 2007.11898[cs]. URL: http://arxiv.org/abs/2007.11898 (visited on 12/13/2023).

[57] Grinvald, M. et al. *TSDF++: A Multi-Object Formulation for Dynamic Object Tracking and Reconstruction*. May 16, 2021. DOI: 10.48550/arXiv.2105.07468. arXiv: 2105.07468[cs]. URL: http://arxiv.org/abs/2105.07468 (visited on 05/03/2023).

[58] Low, K.-L. "Linear Least-Squares Optimization for Point-to-Plane ICP Surface Registration". In: ().

[59] Yang, N. et al. *D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry*. Mar. 28, 2020. DOI: 10.48550/arXiv.2003.01060. arXiv: 2003.01060[cs]. URL: http://arxiv.org/abs/2003.01060 (visited on 12/02/2022).

[60] Hartley, R. et al. *Multiple view geometry in computer vision*. OCLC: 804793563. Cambridge: Cambridge University Press, 2004. ISBN: 978-0-511-18618-9. URL: http://public.eblib.com/choice/publicfullrecord.aspx?p=256634 (visited on 10/02/2018).

[61] Huber, P. J. "Robust Estimation of a Location Parameter". In: *The Annals of Mathematical Statistics* 35.1 (Mar. 1964). Publisher: Institute of Mathematical Statistics, pp. 73–101. ISSN: 0003-4851, 2168-8990. DOI: 10.1214/aoms/1177703732. URL: https://projecteuclid.org/journals/annals-of-mathematical-statistics/volume-35/issue-1/Robust-Estimation-of-a-Location-Parameter/10.1214/aoms/1177703732.full (visited on 12/19/2023).

[62] Gladkova, M. et al. *DirectTracker: 3D Multi-Object Tracking Using Direct Image Alignment and Photometric Bundle Adjustment*. Sept. 29, 2022. DOI: 10.48550/arXiv.2209.14965. arXiv: 2209.14965[cs]. URL: http://arxiv.org/abs/2209.14965 (visited on 02/01/2023).

[63] Ren, S. et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. Jan. 6, 2016. DOI: 10.48550/arXiv.1506.01497. arXiv: 1506.01497[cs]. URL: http://arxiv.org/abs/1506.01497 (visited on 12/05/2023).

[64] Li, Y. et al. *Benchmarking Detection Transfer Learning with Vision Transformers*. Nov. 22, 2021. arXiv: 2111.11429[cs]. URL: http://arxiv.org/abs/2111.11429 (visited on 12/02/2023).

[65] Lin, T.-Y. et al. *Microsoft COCO: Common Objects in Context*. Feb. 20, 2015. DOI: 10.48550/arXiv.1405.0312. arXiv: 1405.0312[cs]. URL: http://arxiv.org/abs/1405.0312 (visited on 12/04/2023).

[66] Kuhn, H. W. "The Hungarian method for the assignment problem". In: *Naval Research Logistics Quarterly* 2.1 (1955). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nav.3800020109, pp. 83–97. ISSN: 1931-9193. DOI: 10.1002/nav.3800020109. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800020109 (visited on 12/12/2023).

[67] Adavanne, S. et al. *Differentiable Tracking-Based Training of Deep Learning Sound Source Localizers*. Oct. 29, 2021. arXiv: 2111.00030[cs,eess]. URL: http://arxiv.org/abs/2111.00030 (visited on 12/15/2023).

[68] Xu, Y. et al. *How To Train Your Deep Multi-Object Tracker*. Apr. 23, 2020. DOI: 10.48550/arXiv.1906.06618. arXiv: 1906.06618[cs]. URL: http://arxiv.org/abs/1906.06618 (visited on 12/15/2023).

[69] Vaswani, A. et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017. URL: https://papers.nips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html (visited on 12/15/2023).

[70] Paszke, A. et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Dec. 3, 2019. DOI: 10.48550/arXiv.1912.01703. arXiv: 1912.01703[cs,stat]. URL: http://arxiv.org/abs/1912.01703 (visited on 12/20/2023).

[71] Ronneberger, O. et al. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. May 18, 2015. DOI: 10.48550/arXiv.1505.04597. arXiv: 1505.04597[cs]. URL: http://arxiv.org/abs/1505.04597 (visited on 11/17/2023).

[72] Simonyan, K. et al. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. Apr. 10, 2015. DOI: 10.48550/arXiv.1409.1556. arXiv: 1409.1556[cs]. URL: http://arxiv.org/abs/1409.1556 (visited on 12/20/2023).

[73] Sturm, J. et al. "A benchmark for the evaluation of RGB-D SLAM systems". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2012). Vilamoura-Algarve, Portugal: IEEE, Oct. 2012, pp. 573–580. ISBN: 978-1-4673-1736-8 978-1-4673-1737-5 978-1-4673-1735-1. DOI: 10.1109/IROS.2012.6385773. URL: http://ieeexplore.ieee.org/document/6385773/ (visited on 11/14/2023).

[74] Palazzolo, E. et al. *ReFusion: 3D Reconstruction in Dynamic Environments for RGB-D Cameras Exploiting Residuals*. Aug. 28, 2019. arXiv: 1905.02082[cs]. URL: http://arxiv.org/abs/1905.02082 (visited on 11/09/2023).

[75] Kingma, D. P. et al. *Adam: A Method for Stochastic Optimization*. Jan. 29, 2017. DOI: 10.48550/arXiv.1412.6980. arXiv: 1412.6980[cs]. URL: http://arxiv.org/abs/1412.6980 (visited on 12/19/2023).

[76] Handa, A. et al. “A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM”. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014 IEEE International Conference on Robotics and Automation (ICRA). Hong Kong, China: IEEE, May 2014, pp. 1524–1531. ISBN: 978-1-4799-3685-4. DOI: 10.1109/ICRA.2014.6907054. URL: http://ieeexplore.ieee.org/document/6907054/ (visited on 11/14/2023).

[77] Rusinkiewicz, S. et al. “Efficient variants of the ICP algorithm”. In: *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*. Third International Conference on 3-D Digital Imaging and Modeling. Quebec City, Que., Canada: IEEE Comput. Soc, 2001, pp. 145–152. ISBN: 978-0-7695-0984-6. DOI: 10.1109/IM.2001.924423. URL: http://ieeexplore.ieee.org/document/924423/ (visited on 11/18/2023).

[78] Steinbrucker, F. et al. “Real-time visual odometry from dense RGB-D images”. In: *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops). Barcelona, Spain: IEEE, Nov. 2011, pp. 719–722. ISBN: 978-1-4673-0063-6 978-1-4673-0062-9 978-1-4673-0061-2. DOI: 10.1109/ICCVW.2011.6130321. URL: http://ieeexplore.ieee.org/document/6130321/ (visited on 08/30/2023).

[79] Park, J. et al. “Colored Point Cloud Registration Revisited”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017 IEEE International Conference on Computer Vision (ICCV). Venice: IEEE, Oct. 2017, pp. 143–152. ISBN: 978-1-5386-1032-9. DOI: 10.1109/ICCV.2017.25. URL: http://ieeexplore.ieee.org/document/8237287/ (visited on 01/03/2023).

[80] Yu, M.-F. et al. “SCP-SLAM: Accelerating DynaSLAM With Static Confidence Propagation”. In: *2023 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*. 2023 IEEE Conference Virtual Reality and 3D User Interfaces (VR). Shanghai, China: IEEE, Mar. 2023, pp. 509–518. ISBN: 9798350348156. DOI: 10.1109/VR55154.2023.00066. URL: https://ieeexplore.ieee.org/document/10108443/ (visited on 12/13/2023).

[81] Singh, G. et al. “Fast Semantic-Aware Motion State Detection for Visual SLAM in Dynamic Environment”. In: *IEEE Transactions on Intelligent Transportation Systems* 23.12 (Dec. 2022). Conference Name: IEEE Transactions on Intelligent Transportation Systems, pp. 23014–23030. ISSN: 1558-0016. DOI: 10.1109/TITS.2022.3213694. URL: https://ieeexplore.ieee.org/document/9928030 (visited on 12/19/2023).

[82] Karpathy, A. et al. “Object discovery in 3D scenes via shape analysis”. In: *2013 IEEE International Conference on Robotics and Automation*. 2013 IEEE International Conference on Robotics and Automation (ICRA). Karlsruhe, Germany: IEEE, May 2013, pp. 2088–2095. ISBN: 978-1-4673-5643-5 978-1-4673-5641-1. DOI: 10.1109/ICRA.2013.6630857. URL: http://ieeexplore.ieee.org/document/6630857/ (visited on 12/19/2023).

[83] Finman, R. et al. “Efficient incremental map segmentation in dense RGB-D maps”. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014 IEEE International Conference on Robotics and Automation (ICRA). ISSN: 1050-4729. May 2014, pp. 5488–5494. DOI: 10.1109/ICRA.2014.6907666. URL: https://ieeexplore.ieee.org/document/6907666 (visited on 12/19/2023).

[84] Tateno, K. et al. “Real-time and scalable incremental segmentation on dense SLAM”. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Hamburg, Germany: IEEE, Sept. 2015, pp. 4465–4472. ISBN: 978-1-4799-9994-1. DOI: 10.1109/IROS.2015.7354011. URL: http://ieeexplore.ieee.org/document/7354011/ (visited on 10/15/2023).

[85] Ückermann, A. et al. “3D scene segmentation for autonomous robot grasping”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. ISSN: 2153-0866. Oct. 2012, pp. 1734–1740. DOI: 10.1109/IROS.2012.6385692. URL: https://ieeexplore.ieee.org/document/6385692 (visited on 12/19/2023).

[86] Soria, X. et al. “Dense Extreme Inception Network: Towards a Robust CNN Model for Edge Detection”. In: 2020 IEEE Winter Conference on Applications of Computer Vision (WACV). IEEE Computer Society, Mar. 1, 2020, pp. 1912–1921. ISBN: 978-1-72816-553-0. DOI: 10.1109/WACV45572.2020.9093290. URL: https://www.computer.org/csdl/proceedings-article/wacv/2020/09093290/1jPbjFHmwi4 (visited on 12/05/2023).

[87] Chang, J. et al. “Deep Adaptive Image Clustering”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017 IEEE International Conference on Computer Vision (ICCV). Venice: IEEE, Oct. 2017, pp. 5880–5888. ISBN: 978-1-5386-1032-9. DOI: 10.1109/ICCV.2017.626. URL: http://ieeexplore.ieee.org/document/8237888/ (visited on 12/19/2023).

[88] Kendall, A. et al. *What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?* Oct. 5, 2017. DOI: 10.48550/arXiv.1703.04977. arXiv: 1703.04977[cs]. URL: http://arxiv.org/abs/1703.04977 (visited on 12/19/2023).

[89] Cipolla, R. et al. “Multi-task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Salt Lake City, UT, USA: IEEE, June 2018, pp. 7482–7491. ISBN: 978-1-5386-6420-9. DOI: 10.1109/CVPR.2018.00781. URL: https://ieeexplore.ieee.org/document/8578879/ (visited on 12/19/2023).

[90] Sandström, E. et al. *UncLe-SLAM: Uncertainty Learning for Dense Neural SLAM*. Sept. 6, 2023. arXiv: 2306.11048[cs]. URL: http://arxiv.org/abs/2306.11048 (visited on 11/08/2023).

[91] Mildenhall, B. et al. *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis*. Aug. 3, 2020. DOI: 10.48550/arXiv.2003.08934. arXiv: 2003.08934[cs]. URL: http://arxiv.org/abs/2003.08934 (visited on 12/14/2023).

[92] Ming, Y. et al. *iDF-SLAM: End-to-End RGB-D SLAM with Neural Implicit Mapping and Deep Feature Tracking*. Sept. 16, 2022. arXiv: 2209.07919[cs]. URL: http://arxiv.org/abs/2209.07919 (visited on 11/25/2022).

[93] Rosinol, A. et al. *NeRF-SLAM: Real-Time Dense Monocular SLAM with Neural Radiance Fields*. Oct. 24, 2022. DOI: 10.48550/arXiv.2210.13641. arXiv: 2210.13641[cs]. URL: http://arxiv.org/abs/2210.13641 (visited on 03/16/2023).

[94] Kerbl, B. et al. *3D Gaussian Splatting for Real-Time Radiance Field Rendering*. Aug. 8, 2023. DOI: 10.48550/arXiv.2308.04079. arXiv: 2308.04079[cs]. URL: http://arxiv.org/abs/2308.04079 (visited on 12/14/2023).

[95] Keetha, N. et al. *SplaTAM: Splat, Track & Map 3D Gaussians for Dense RGB-D SLAM*. Dec. 4, 2023. DOI: 10.48550/arXiv.2312.02126. arXiv: 2312.02126[cs]. URL: http://arxiv.org/abs/2312.02126 (visited on 12/14/2023).

[96] Matsuki, H. et al. *Gaussian Splatting SLAM*. Dec. 11, 2023. DOI: 10.48550/arXiv.2312.06741. arXiv: 2312.06741[cs]. URL: http://arxiv.org/abs/2312.06741 (visited on 12/15/2023).

[97] Qiu, Y. et al. "AirDOS: Dynamic SLAM benefits from Articulated Objects". In: *2022 International Conference on Robotics and Automation (ICRA)*. May 23, 2022, pp. 8047–8053. DOI: 10.1109/ICRA46639.2022.9811667. arXiv: 2109.09903[cs]. URL: http://arxiv.org/abs/2109.09903 (visited on 12/08/2023).

[98] Doersch, C. et al. *Unsupervised Visual Representation Learning by Context Prediction*. Jan. 16, 2016. DOI: 10.48550/arXiv.1505.05192. arXiv: 1505.05192[cs]. URL: http://arxiv.org/abs/1505.05192 (visited on 12/19/2023).

[99] Noroozi, M. et al. *Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles*. Aug. 22, 2017. DOI: 10.48550/arXiv.1603.09246. arXiv: 1603.09246[cs]. URL: http://arxiv.org/abs/1603.09246 (visited on 12/19/2023).

[100] Masci, J. et al. "Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction". In: *Artificial Neural Networks and Machine Learning – ICANN 2011*. Ed. by Honkela, T. et al. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2011, pp. 52–59. ISBN: 978-3-642-21735-7. DOI: 10.1007/978-3-642-21735-7_7.

[101] Vincent, P. et al. "Extracting and composing robust features with denoising autoencoders". In: *Proceedings of the 25th international conference on Machine learning - ICML '08*. the 25th international conference. Helsinki, Finland: ACM Press, 2008, pp. 1096–1103. ISBN: 978-1-60558-205-4. DOI: 10.1145/1390156.1390294. URL: http://portal.acm.org/citation.cfm?doid=1390156.1390294 (visited on 12/19/2023).

[102] Umeyama, S. "Least-squares estimation of transformation parameters between two point patterns". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13.4 (Apr. 1991). Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 376–380. ISSN: 1939-3539. DOI: 10.1109/34.88573.

Appendix A

Evaluation Metrics

A.1 Absolute Trajectory Error

Absolute Trajectory error [73] is an intuitive metric to evaluate the global consistency of the estimated trajectory by measuring the root-mean-square error between predicted camera poses and ground truth. We denote estimated, and the true camera poses as $\mathbf{Q} \in SE(3)$ and $\mathbf{P} \in SE(3)$, respectively. Besides, the translation of a rigid body transformation matrix in $SE(3)$ is defined with an operation $trans(\cdot)$, and the rotation part with $rot(\cdot)$. Because the estimated and the ground-truth trajectory can be in arbitrary coordinate frames, we first need to align them by using, for example, the Umeyama alignment method [102], which brings out a transformation matrix $\mathbf{S}$. Then, we define the absolute trajectory error matrix at time step i as

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

To ensure an objective evaluation of the system performance on various datasets and sequences, we compute the root mean square error (RMSE) over all time indices (n) of the translational part

$$\text{ATE} := \sqrt{\frac{1}{n}\sum_{i=1}^{n} ||trans(\mathbf{E}_i)||^2} \tag{A.2}$$

A.2 Relative Pose Error

To evaluate the local accuracy of the estimated trajectory, relative pose error (RPE) is in particular useful [73]. It compares the reconstructed relative transformation between nearby poses to the ground truth transformation. Assume $\mathbf{Q_i}$ is the estimated pose of the current frame and $\mathbf{Q_{i+1}}$ the next frame, given the respective ground-truth poses $\mathbf{P_i}$ and $\mathbf{P_{i+1}}$, the RPE matrix can be formulated as

$$\mathbf{F}_i := (\mathbf{Q}_i^{-1}\mathbf{Q}_{i+1})^{-1}(\mathbf{P}_i^{-1}\mathbf{P}_{i+1}) \tag{A.3}$$

$\mathbf{F}_i$ consists of a translation and a rotational error. For a clear distinction in evaluation, we normally compute them separately. Similar to absolute trajectory error, we calculate the RMSE of the translational component $trans(\mathbf{F}_i)$ over all possible time intervals

$$\text{RPE}_{trans} := \sqrt{\frac{1}{m}\sum_{i=1}^{m} ||trans(\mathbf{F}_i)||^2} \quad \text{for} \quad i = 1, ..., n \tag{A.4}$$

where $m = n-1$. Similar to ATE and translational RPE, we compute RMSE over the rotational relative errors of all paired estimated and ground truth poses:

$$\text{RPE}_{rot} := \sqrt{\frac{1}{m}\sum_{i=1}^{m} \|\angle \mathbf{F}_i\|^2} \quad \text{for} \quad i = 1, ..., n \tag{A.5}$$

where $\angle \mathbf{F}_i$ is defined as:

$$\angle \mathbf{F}_i := \arccos\left(\frac{tr\,(rot(\mathbf{F}_i)) - 1}{2}\right) \tag{A.6}$$

A.3 Intersection Over Union

Intersection over Union (IoU) is a metric used to evaluate the accuracy of an object detector on a particular dataset. It's often used in the context of evaluating models for tasks such as object detection, segmentation, and computer vision problems where the prediction involves determining the position and size of objects. Formally, it is calculated as the ratio of the area of overlap between the predicted bounding box and the ground truth bounding box to the area of the union of these two boxes.

In this thesis, we calculate the IoU differently. In image segmentation, a rectangular area is not always the case since objects can come in regular or irregular shapes. Therefore, we need a pixel-wise analysis. We do not generate bounding boxes for every segmentation. Instead, we compute IoU using the coverage area of the segmentation. Given the area of two regions denoted as A and B, we represent the IoU as:

$$\text{IoU} = \frac{A \cap B}{A \cup B} \tag{A.7}$$

Although over-smooth segmentation often displays slightly different areas than the object actually occupies, we see no obvious disadvantages of using mask area instead of bonding box area to compute IoU. When we use bounding boxes to include irregular objects, a large area that does not belong to the object is also considered. Compared to that area, the area difference due to over-smooth segmentation has no adverse impact on the accuracy of the evaluation.