Imperial College London

Department of Computing

Object-level Dynamic SLAM

Binbin Xu

17th June 2022

Supervised by Dr Stefan Leutenegger

Co-supervised by Professor Andrew J. Davison

Copyright Declaration

Abstract

Visual Simultaneous Localisation and Mapping (SLAM) can estimate a camera's pose in an unknown environment and reconstruct an online map of it. Despite the advances in many real-time dense SLAM systems, most still assume a static environment, which is not a valid assumption in many real-world scenarios. This thesis aims to enable dense visual SLAM to run robustly in a dynamic environment, knowing *where* the sensor is in the environment, and, also importantly, *what* and *where* objects are in the surrounding environment for better scene understanding.

The contributions in this thesis are threefold. The first one presents one of the first object-level dynamic SLAM systems that robustly track camera pose while detecting, tracking, and reconstructing all the objects in dynamic scenes. It can continuously fuse geometric, semantic, and motion information for each object into an octree-based volumetric representation.

One of the challenges in tracking moving objects is that the object motion can easily break the illumination constancy assumption. In our second contribution, we address this issue by proposing a dense feature-metric alignment to robustly estimate camera and object poses. We will show how to learn dense feature maps and feature-metric uncertainties in a self-supervised way. They formulate a probabilistic feature-metric residual, which can be efficiently solved using Gauss-Newton optimisation and easily coupled with other residuals.

So far, we only reconstruct objects' geometry from the sensor data. Our third contribution further incorporates category-level shape prior to the object mapping. Conditioning on the depth measurement, the learned implicit function completes the unseen part while reconstructing the observed part accurately. It can yield better reconstruction completeness and more accurate object pose estimation.

These three contributions in this thesis have advanced the state of the art in visual SLAM. We hope such object-level dynamic SLAM systems will help robots intelligently interact with the human-existing world.

Acknowledgements

A long journey has come to an end and I am very thankful for all those who have helped, supported, and inspired me, not only for my PHD studies but also over and beyond.

I am first very grateful to my supervisor Dr Stefan Leutenegger who has always been extremely supportive of my research and my, sometimes, risky ideas. His solid knowledge in state estimation has guided me every step of the way through my studies and provided strong support in my work. I also very much appreciate the time and efforts he generously spent during our “morning coffee” calls, which has proved very supportive in a hard time during the COVID lockdown.

I am also very appreciative of my co-supervisor Professor Andrew Davison. His vision of long-term research and insightful wisdom have helped shape my research trajectory. He has built an incredible research community that has provided valuable opportunities to discuss research ideas. I feel very honoured and blessed to be able to work with such a SLAM legend.

I would like to thank my examiners, Prof. Christos Bouganis and Prof. Laurent Kneip, for taking the time to evaluate my thesis and to provide valuable feedback on my work. I am also grateful to the many teachers, professors, and mentors I met in China, Japan and UK throughout my life who, official or not, have helped me learn to be the person I am today.

I was also very fortunate to work with many talented people in both Stefan’s and Andy’s groups. I would like to thank Shuaifeng for his companionship inside and outside the school as we worked through the challenges of PhD and COVID together. Zoe for all the joys and concerns we shared together in our PhD life. Dimos for numerous hours we spent in the Vicon room. Chris for many inspiring discussions on Math. Nils, and Sotiris for all the memorable times we shared during the virtual coffee time and spent together in the office. Xingxing for all the hours we spent discussing various research ideas. Michael for spending his precious time teaching me Jacobians in front of the whiteboard. Wenbin, Sajad, Ronnie, John, and Patrick for their patient guidance when I started my PhD. Yifei for the time we spent together discussing the visual-inertial SLAM system and the supervision experience I learned during this collaboration. Jan, Tristan, and Andrea for their patient guidance on my formula derivation and valuable suggestions on the PhD milestones. I have had fruitful discussions with many other lab members, including Masha, Ed, Anna, Simon B., Simon S., Charlie, Dan, Stephen, Robert, Hide, Xin, Shikun, Edgar, Kentaro, Dorian. I would also like to thank Iosifina and Amani for the various support I received related to my PhD studies.

I would like to also thank my Facebook mentor, Lingni, when I conducted my research internship at Facebook Reality Labs. Conducting a remote internship is hard but her patient guidance makes the internship experience pretty enjoyable. I would like also to thank the colleagues I met during the internship, including Yuting, Tanner, Tianye, Zhaoyang, Julian, Christopher, Steve.

Lastly but most importantly, I deeply thank my parents, Yinbao and Yin, for their unconditional support. Due to COVID, I could not fly back to China for the past two years and I owe my greatest thank to them for their understanding. I also appreciate my uncle, Chi, who has fostered my interest in robotics and encouraged me to pursue a PhD since my childhood. Without their trust, it is hard to imagine that from a small town in China, I could pursue my academic dreams in Tokyo and then in London. Lastly I greatly appreciate my partner Ying who has accompanied me through both Tokyo and London and we have enjoyed such amazing life together. Life is very exciting and colourful even with many troubles on the roads in between. The support from all the people I met on this journey make this work possible. I will carry with what I learned in this study for my next journey.

Binbin Xu

London, UK, February, 2022

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Contents of Chapter

1.1 Motivation

Simultaneous Localization and Mapping (SLAM) techniques simultaneously estimate a map of an unknown environment and a robot pose within that map. Research in the real-time visual SLAM field has experienced rapid progress. It started from the beginning of sparse SLAM [Davison et al., 2007, Klein and Murray, 2007, Leutenegger et al., 2014], and was able to enter in to dense SLAM [Newcombe et al., 2011a, Whelan et al., 2016, Dai et al., 2017, Laidlow et al., 2017, Loop et al., 2016, Vespa et al., 2018] thanks to the increased computational power of the Graphics processing unit (GPU) and cheap depth sensors. In the past few years, many people worked on exploiting the power of a Deep Neural Network (DNN) from large amounts of training data and inserting the learned prior information inside the SLAM framework. This has enabled the SLAM system to create a global dense map from a monocular camera [Bloesch et al., 2018, Zhou et al., 2018a, Czarnowski et al., 2020] or have a better semantic and instance scene understanding [McCormac et al., 2017, McCormac et al., 2018, Sucar et al., 2020, Zhi et al., 2019]. The fast-evolving research in SLAM has, since, benefited various applications, such as robotics and Virtual Reality(VR) / Augmented Reality(AR), more than ever.

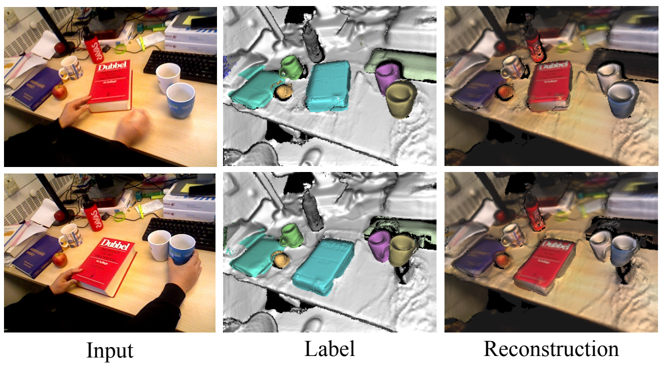
Despite this progress, most of these works still are based on the fundamental assumption of a static environment, within which points in the 3D world would always have the same spatial position in the global world and the only moving object being the camera. This assumption enabled the success of early phases of development as it creates a robust epipolar geometry constraint, conveniently eliminating the chicken-and-egg problem between reconstructing structure and estimating motion. A camera pose can be estimated between a live frame and its reference frame, which is based on the assumption that the relative transformation between those two images is caused only by the camera motion. It is this basic yet strong assumption that allows a joint probabilistic inference (sparse SLAM [Durrant-

Whyte and Bailey, 2006]) or an alternative optimisation (dense SLAM [Newcombe et al., 2011a]) of the map and the pose to solve the SLAM problem. Any moving object in the environment should be treated as outlier to the static model and intentionally removed from the tracking and mapping process.

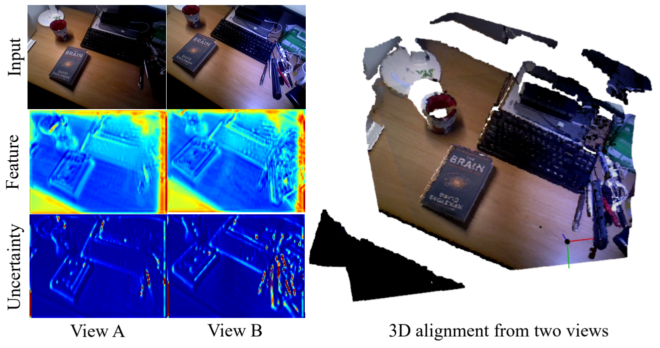
This idealized setup, therefore, can only deal with a small portion of dynamic parts and distances itself from real-world applications as environments do change, especially places where humans exist. This assumption of a static environment for SLAM system has served as an inspirational springboard for SLAM development, but a continuation with the same methods would halt our advancements for robust applications and better scene understandings. Robust SLAM working in the dynamic environment is still an open problem and this leads to the goal of this PhD thesis.

The definition of *a dynamic environment* in this thesis is a scene where objects are moving under the perception of a camera sensor. Our main interest in this thesis targets moving rigid objects and intentionally excludes non-rigid objects, such as human hands or bodies. We also consider changing illuminations as part of a dynamic environment. Instead of solely reconstructing a single static and clean background model for robust camera tracking and ignoring all possible moving objects, our ultimate goal is to build *a multi-instance dynamic system that can consistently and reliably estimate geometric, semantic, and motion properties for each object in the scene.* This thesis has made a few contributions towards this ultimate goal. I believe, similar to human perception, an awareness of instances in the map would be a more proper solution to the dynamic SLAM issue and can lead to a semantically meaningful scene representation. Augmenting semantic and object information in a map is also a significant step for robotic agents to advance beyond obstacle avoidance and achieve environment interactions for human agents.

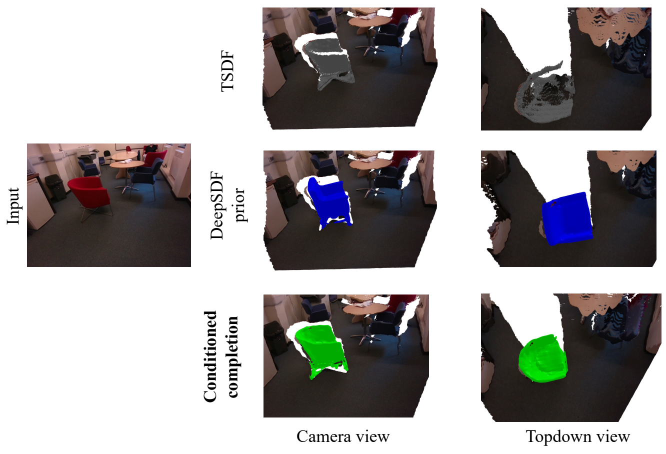
In this thesis, we begin with presenting a novel object-level dynamic SLAM



(a) MID-Fusion [Xu et al., 2019]



(b) Deep Probabilistic Feature-metric Tracking [Xu et al., 2021a]



(c) Object-level Dynamic SLAM with Map Completion

Figure 1.1: Demonstrations of each contributed system in this thesis

system called 'MID-Fusion' [Xu et al., 2019] (shown on the Figure 1.1a), which continuously estimates the pose, semantic class, and dense geometry of each object in the scene. The 3D geometry of each object is reconstructed in an efficient octree-based truncated signed distance field (TSDF) volume [Vespa et al., 2018] that can be naturally applied for further robotic applications such as exploration and manipulation. The poses of the camera and objects are estimated via a probabilistic combination of photometric and geometric residuals that are also commonly used in many direct SLAM systems. This separation of each object in the scene to its individual representation can naturally handle the real-world dynamic scene. The object instances are first detected and segmented using Mask R-CNN [He et al., 2017] and further refined using motion residuals. The semantic information is efficiently fused using a Bayesian update scheme. The often imperfect object boundary from 2D segmentation is refined via depth segmentation and further refined with 3D foreground probabilistic fusion.

In MID-Fusion, our experiments showed that object-level representation can lead to more robust camera tracking in a dynamic scene and better scene understanding. However, we also found the limitations in the conventional photometric tracking residual, which stringently requires brightness constancy and good initialization (close to the global minimum). These requirements often cannot be met in reality, especially for objects that have non-Lambertian surfaces (e.g. typical plastic objects). To overcome these limitations, we explore a new tracking pipeline [Xu et al., 2021a] that uses Convolutional Neural Networks (CNN) to predict a good initial pose, and learns features and feature-metric uncertainties that can be robust to lighting changes. We solve this novel residual using the Gauss-Newton algorithm and unroll this optimisation step to learn deep features and its associated uncertainties in an end-to-end manner. In the experiments, we have shown that this residual is robust to lighting changes and have a larger convergence basin, as shown in Figure 1.1b. It provides better tracking accuracy than classic residuals or pure learning-based approaches and can be naturally combined with other residuals, such as Iterative Closest Point (ICP) residual, to further improve performance.

In terms of object mapping, we found that the traditional TSDF fusion [Newcombe et al., 2011a, Vespa et al., 2018] adopted in MID-Fusion can only reconstruct the parts observed from the sensor. The unobserved parts, due to occlusion or being behind the surface, cannot be reconstructed, resulting in many incomplete meshes. To tackle this problem, we propose to learn an implicit occupancy field that conditions on both observed reconstruction and category-shape prior. We further propose to jointly optimise this occupancy field and object pose, which enables more robust object pose estimation and better object reconstruction qualities. We demonstrate the effectiveness of our proposed system in both the synthetic and real-world experiments, as one example shown in Figure 1.1c.

A brief historical review of SLAM, from static SLAM to dynamic SLAM, is given below. More specific discussions on work closely related to the contributions in this thesis will be presented in each individual chapter.

1.2 Static SLAM: from Sparse, to Dense, to Semantic

Although this thesis focuses primarily on dynamic SLAM, most existing dynamic SLAM systems borrow ideas heavily from the existing static SLAM systems. [Engel, 2017] introduces a taxonomy of visual SLAM, categorizing systems into *direct vs. indirect* and *sparse vs. dense*. The first axis, *direct vs. indirect*, is determined whether the input measurements for camera pose and geometry estimation is directly from the actual sensor value or pre-computed from features extracted from the image(s). The other axis, *sparse vs. dense*, is determined whether the input measurements are only from a sparse selected set of independent points (usually corners) or all image pixels.

A real-time visual SLAM is one that can complete all processing in real time, at the rate of operation or framerate of the camera. It is composed of at least two main components: tracking (online camera pose estimation) and mapping (fusing past observations into a coherent environment model). The front-end tracking component requires a high framerate and low latency, while the back-end mapping can run slower than the camera framerate.

Most early successful visual SLAM systems were sparse and indirect, due to limited computation capabilities. MonoSLAM is one of the first real-time single camera visual SLAM systems [Davison et al., 2007]. It uses a joint state to represent the camera pose and an extended Kalman filter (EKF) method to build a point cloud map. When new features are observed, they are filtered into the current map with a joint Gaussian uncertainty. However, when the map grows larger, the filter update has an $O(N^2)$ complexity, restricting the increase of the map size. Thus, Klein and Murray, later proposed Parallel Tracking and Mapping (PTAM) [Klein and Murray, 2007], which uses a keyframe-based bundle adjustment, rather than a filtering method. In their work, feature points are associated with keyframes, which are then selected based on the structural sparsity of the problem. Real-time camera tracking given a map and slow-speed bundle adjustment optimisation to update maps run in parallel, enabling computationally expensive bundle adjustment into a real-time SLAM work. With the increasing computation power and advanced feature keypoint descriptors, such as SIFT [Lowe, 1999], SURF [Bay et al., 2006], ORB [Rublee et al., 2011], and BRISK [Leutenegger et al., 2011], Sparse indirect SLAM systems have become mature and can provide accurate and reliable camera tracking in mostly static environments. One of the modern SLAM systems, ORB-SLAM [Mur-Artal and Tardós, 2017] is such an example.

One of the drawbacks of these sparse systems is that they can only estimate the sparse geometry of the surrounding environment due to its reliance on sparse 3D landmarks from sparse 2D keypoints. However, dense geometry of the map is desirable for some robotic applications, such as collision avoidance and scene understanding. With the emergence of commodity graphics processing units (GPUs), dense SLAM has also started to rise. DTAM is one of the first real-time monocular dense SLAM systems that estimate the dense geometry of keyframes by minimizing the photometric error between the live frame and the reference keyframes with a small baseline into a perspective cost volume.

The emergence of low-cost 3D sensing equipment that can directly measure depth information has further boosted the research of dense SLAM. KinectFusion [Newcombe et al., 2011a] is one of the earliest systems that can build a dense 3D volumetric reconstruction of arbitrary environments in real-time by only acquiring depth information from a Kinect sensor. The map representation of KinectFusion is based on a volumetric data structure, Truncated Signed Distance Function (TSDF) [Curless and Levoy, 1996], which provides an implicit and computationally efficient way to represent the scene and the surface. The parallel structure inside the KinectFusion with the usage of GPU also improves its real-time performance. The initial design of KinectFusion maps the 3D geometry as a regularly spaced 3D grid, and thus the memory usage scales with the size of the represented volume rather than the surface. This limits its ability to perform large-scale mappings. Some following work proposed to use more efficient data structures, such as N^3 trees [Chen et al., 2013], octrees [Vespa et al., 2018], and voxel hashing [Nießner et al., 2013, Kahler et al., 2015]. Other following works focused on solving the drift in camera tracking. BundleFusion [Dai et al., 2017] constructs a globally consistent 3D model by using a robust pose estimation method based on both sparse features (SIFT [Lowe, 1999]) and dense (geometric and photometric) constraints. In the map updating process, a de-integration operation was coupled with a conventional integration process to remove integration errors and frames can be reintegrated with new poses when loop closure is detected. ElasticFusion [Whelan et al., 2016] achieves global consistency by applying elastic map deformation of surfel-based map representation upon loop closure. [Laidlow et al., 2017] extended it to RGB-D-Inertial sensors with the camera tracking replaced by a more robust approach using tightly-coupled visual-inertial odometry. [Vespa et al., 2018] proposed an octree-based volumetric mapping that can support both SDF and occupancy mapping, and is efficient enough to run in real-time on CPU. Our first work, MID-Fusion, is developed using this map representation for objects, including background.

In addition to geometric information, other information can also be estimated from the input measurements. Thanks to the advancement of neural network research, one important direction is semantic SLAM, which aims to integrate local semantic information from 2D input images to build a global 3D semantic map inside the SLAM framework for better scene understanding. SemanticFusion [McCormac et al., 2017] combines the ElasticFusion [Whelan et al., 2016] with a semantic segmentation CNN [Noh et al., 2015] in a Bayesian update scheme to create a semantically fused dense reconstruction. [Nakajima et al., 2018] speeded up the segmentation with the help of geometric segmentation on depth images and only run expensive semantic segmentation on keyframes.

Moving forward from dense semantic mapping, object-level representation provides semantic map representation that can naturally differentiate different instances in the same semantic class and is very important for understanding the relationship between objects in the scene. An early version is SLAM++ [Salas-Moreno et al., 2013] that can recognize pre-defined and repeated objects in the environment in real-time. However, it requires collecting all possibly appearing object instances in the environment with very detailed geometric information. Fusion++ [McCormac et al., 2018] reconstructs arbitrary object-centric maps from 2D CNN detections [He et al., 2017] in TSDF volumes. [Sünderhauf et al., 2017] combines ORB-SLAM2 [Mur-Artal and Tardós, 2017] with a Single-shot Multi-box Detector (SSD) approach [Liu et al., 2016] to detect instance labels in the 3D world and generates a global object-oriented semantic map. Kimera [Rosinol et al., 2020] creates a dense mesh reconstruction with a VIO-frontend and provides a pose-graph optimisation backend used upon loop closure. It can additionally provide a semantically annotated scene graph map for better scene understanding. Rather than reconstructing object geometry from scratch, some other works concentrate on extracting a compact object representation. [Nicholson et al., 2018] generates object-centric maps using 3D quadric surface representation. DirectShape models object shapes using PCA models and optimises these shapes using geometric, photometric, and silhouette information [Wang et al., 2020]. Deep-SLAM++ [Hu et al., 2019], NodeSLAM [Sucar et al., 2020], FroDo [Runz et al., 2020], and DSP-SLAM [Wang et al., 2021] represent objects in a compact latent vector that can be learned from category-level object CAD models in ShapeNet dataset [Chang et al., 2015]. The object representations explore the variance and similarities inside an object category and this learnt object prior is used to represent object geometries. Other works also explore understanding the inter-relationship between objects in the scene by learning a scene graph representation [Wald et al., 2020]. However, most of these works only target static environments, as multi-view consistency of static world points is required to localise the shape prior models.

One thing to be noted is that the computational requirements of SLAM algorithms scale with the quantity of data they need to process. It is determined not only by the resolution of the camera, but also by the design choice of the input to the tracking, the density of the mapping model, and the network architecture, where included. Therefore, a real-time SLAM system design is influenced by a variety of factors, from frame resolution to point selection threshold and reconstruction density, resulting in different hardware platform requirements. With the progress of state-of-the-art SLAM algorithms to dense mapping and notably semantic prediction, real-time SLAM systems demand at least desktop-grade CPUs and often one or even a few high-end GPUs for acceleration, especially for deep neural network inference. Many works have also been proposed in order to optimise real-time performance on low-power platforms for mobile robot applications [Boikos and Bouganis, 2017].

1.3 Dynamic SLAM: from Background Reconstruction to Object-level

By far, all the research mentioned above and most existing approaches have a basic assumption that the environment is mostly static, and that dynamic objects can be treated as outliers, to a certain limit and usually in a very small portion, to the static model, and are ignored intentionally in the tracking and reconstruction step. However, real-world scenarios are often changing, especially in the places where humans exist. Therefore, existing dynamic systems that were originally designed for static environments cannot work robustly in real-world dynamic scenes. To overcome this issue, some work are coming up recently to enable SLAM to work again in dynamic environments. In the remaining part of this section, I will introduce some related work on the dynamic SLAM and categorize them into three parts based on the condition if they integrate moving objects into the map and the condition if they tackle deformable objects.

1.3.1 Removal of moving objects

When dynamic objects occupy an important part of the scene, visual SLAM systems that do not specifically address dynamic content tend to confuse the motion of the dynamic objects with the camera's ego-motion, leading to wrong camera pose estimation and distorted geometry reconstruction. Many works have proposed to address this issue. [Jaimez et al., 2017] proposed a method to jointly estimate visual odometry and scene flow under a dynamic environment. They estimate a dominant rigid motion in the over-segmented clusters as the initially estimated camera motion. Then the static parts are used to refine the camera motion estimation and the moving parts are used to refine a piece-wise rigid scene flow. StaticFusion [Scona et al., 2018] leverages the reconstructed model to reduce the overall drift and jointly estimates the frame-to-model tracking and static/dynamic segmentation. The frame-to-model motion estimation is formulated by geometric (ICP) and photometric (RGB) reprojection residuals. It is also weighted together by the segmentation score, which is used to separate static and dynamic parts. The segmentation term is composed of three parts, camera motion (ICP+RGB) residuals, depth inconsistency prior, and a smoothness regularization.

In addition to these geometric solutions, [Barnes et al., 2018] proposed a self-supervised learning approach to segment dynamic objects in the scene. It built a prior 3D static map using a camera and LIDAR in the data collection step. The prior static map is built by collecting data in the target environment in multiple traversals and only the points that appear in different traversals are considered static and remain in the prior map. The 3D points that are collected only in one traversal are considered as belonging to moving objects and removed from the 3D map. In the training step, they predict both disparity and ephemerality masks from a single RGB image using a convolutional encoder-multi-decoder network architecture. [Bescós et al., 2018] proposed to use Mask R-CNN [He et al., 2017] to detect prior dynamic objects, such as people, vehicles and animals. Then they perform ORB-SLAM2 tracking module [Mur-Artal and Tardós, 2017] on the regions outside the prior dynamic objects. Then based on this estimated camera pose, their system checks the depth inconsistency to detect moving objects that are not a prior dynamic. After refining the static regions, camera pose tracking is also refined. Using the estimated camera, dynamic regions and the neighbouring regions are inpainted using the corresponding information from keyframes.

1.3.2 Integration of moving rigid objects

Instead of only reconstructing a static background, some works explore how to track and reconstruct rigid moving objects inside the environment. Co-Fusion [Rünz and Agapito, 2017] is a system that is extended from ElasticFusion and can segment, track, and reconstruct several moving objects. The segmentation is mainly based on motion between two consecutive frames using a fully connected Conditional Random Field (CRF), where ICP cost is used as the unary potentials. Alternatively, semantic segmentation can also be used to determine objects on each frame, yet in an offline case. Co-Fusion assumes that several rigid bodies are moving in the scene and uses geometric and photometric terms to track 6 Degrees of Freedom (DoF) rigid pose for each object. The reconstruction and tracking use the same method proposed in the ElasticFusion [Whelan et al., 2016]. The following work MaskFusion[Rünz and Agapito, 2018] replaces the segmentation module of Co-Fusion with Mask R-CNN [He et al., 2017] to segment instances in the scene online. To compensate for the imperfect mask boundary and sometimes missed detections from Mask R-CNN, they combine it with a geometric segmentation method [Tateno et al., 2015] to provide a better boundary and use the rendered masks from reconstructed models in case of failed recognition. Both Co-Fusion and MaskFusion use surfels to represent map models, which is memory efficient but cannot directly provide free space information in the map, and neither surface connectivity. MID-Fusion [Xu et al., 2019], presented in Chapter 3 of this thesis, leverages a memory efficient octree-based volumetric representation of a Signed Distance Field (SDF) and further conducts semantic fusion for each detected object. EM-Fusion [Strecke and Stuckler, 2019] proposes to estimate object pose by directly align the object SDF with the input frame. A similar work is proposed by [Bârsan et al., 2018], targeting outdoor environments. From stereo cameras, depth maps are first calculated using Efficient Large Stereo Matching [Geiger et al., 2011] or DispNet [Mayer et al., 2016]. Then the dynamic and potentially dynamic objects are detected using a Multi-task Network Cascades (MNC) [Dai et al., 2016]. In parallel, sparse scene flow are calculated on the current frame and the previous frame. Based on the estimated scene flow, camera visual odometry and each instance's motion are further calculated. Based on the estimated motion and the instance masks, corresponding information on each frame is fused to each instance's volumes using InfiniTAM [Kahler et al., 2015].

Similar to visual static SLAM, instead of alternating the optimisation of tracking and mapping as most dense SLAM systems do, another direction is to formulate a joint probabilistic inference on map and pose for higher object tracking accuracy [Durrant-Whyte and Bailey, 2006], with the caveat of sacrificing the dense map representation and depth fusion. This is particularly useful in the outdoor environment, especially for autonomous driving applications. [Li et al., 2018a] proposes a stereo vision-based system that can track robustly both the camera pose and 3D semantic objects in dynamic environments. It creates a dynamic object bundle adjustment (BA) approach to fuse temporal sparse feature correspondences and the semantic 3D measurement model for object pose, velocity and point cloud estimations. DynaSLAM-II [Bescós et al., 2021] extends ORB-SLAM II [Mur-Artal and Tardós, 2017] to dynamic environments by representing objects as sparse pointclouds. It jointly optimises the camera pose, object poses, and geometries in an object-level pose graph optimisation. ClusterSLAM [Huang et al., 2019] formulates object detection and tracking as a clustering problem of landmark movements and solves it in a batch optimization scheme. Following that, they reformulated it as an online VO SLAM that also considers semantic detection [Huang et al., 2020].

In addition to different map representation choices for these object-level SLAM systems, there are also different motion models for object movements. While the majority of these dense object-level SLAM systems, such as [Rünz and Agapito, 2018, Xu et al., 2019], use a zero-velocity motion model to track objects, some other works, such as [Bescós et al., 2021], use a constant velocity motion model, or a white-noise-on-acceleration prior [Barfoot, 2017], for example in the [Huang et al., 2020].

1.3.3 Integration of deformable objects

We introduced some work above on integrating multiple possible moving objects in the SLAM framework. However, in those works, the non-rigid deformable objects are either out of the scope or treated as a composition of several rigid bodies. There is another category of work specifically targeting at reconstructing 3D deformable objects, especially human hands and bodies.

Although there have been many work being proposed to reconstruct non-rigid objects, they are often off-line and require multiple sensor settings. DynamicFusion [Newcombe et al., 2015] is the first real-time dense reconstruction in dynamic environments using a single RGB-D camera. It extends the KinectFusion [Newcombe et al., 2011a] to model the dynamic scene by estimating a dense volumetric 6D motion field, which can warp the static surface into the dynamic scene input. To efficiently estimate the motion field in real-time, the field is based on a sparse set of rigid node motions and then refined through interpolation.

Since that, many other dense SLAM algorithms were proposed to capture the dynamic environment. VolumeDeform [Innmann et al., 2016] proposes using both depth and colour correspondences in the data association part for motion field estimation. They also define the motion field points on the discretised volumetric grids, on the same level of volume representation, instead of the interpolated field. Later, Fusion4D extends the dynamic scene construction to a multiple RGB-D camera set-up [Dou et al., 2016]. It estimates non-rigid tracking based on a 2D dense correspondence field within images using a learning-based method. This provides a more robust initialization to tackle fast motions. It also proposed to use key-volumes, an idea similar to keyframe, instead of one fixed canonical model, to solve the large topology change.

1.3.4 Position of thesis and quality metrics

The position of this thesis fits into the direction described in Section 1.3.2 since we believe that, similar to human perception, an awareness of instances in the map is significant for robots to perceive and interact with the changing environment.

This direction is defined as *object-level dynamic SLAM* and aims to provide accurate and robust pose estimation of the camera sensor in dynamic scenes while incrementally building dense object-level maps of the surrounding environment. The pose of the camera is typically estimated by minimising the error between the sensor observation and the static background environment that has been generated. The system then estimates the object poses of moving objects by minimising the error between the newly observed object information and the reconstructed object models. The camera pose estimation is then used to integrate the newly captured background information to improve the accuracy of the current background map and is combined with the estimated object poses to improve the accuracy of the corresponding object models.

As such, the main quality metrics used for object-level dynamic SLAM can be in three folders: the accuracy of the camera pose estimation, the accuracy of the moving object pose estimation, and/or the accuracy of the generated object maps, including the background map.

The accuracy of the pose is defined as the distance (error) between the real-world position with respect to the origin point and the estimated position of the recovered pose. To quantify the pose estimation performance on the entire captured trajectory, one of the widely adopted metrics is the Root-Mean-Square-Error (RMSE) of the Absolute Trajectory Error (ATE). When global consistency is not enforced, other metrics can also be used to evaluate the pose estimation accuracy, such as relative pose error (RPE) metrics [Sturm et al., 2012] that compares the estimated relative transformations between nearby poses to the ground truth relative transformations, or 3D End-Point-Error (EPE). In this thesis, we chose the widely used TUM RGB-D dataset [Sturm et al., 2012] as our main benchmark to evaluate the camera pose estimation accuracy.

When the origin of the estimated object map is aligned with the origin of the ground truth object model, the accuracy of the object pose estimation can also be quantified using the same metrics, such as ATE, RPE, or 3D EPE. Since there are no available real-world benchmark datasets containing ground truth origin and the trajectory of semantic objects, we chose to use synthetic datasets, such as MovingObjects3D [Lv et al., 2019] that contains objects from ShapeNet [Chang et al., 2015] with random motions,to evaluate the object pose estimation accuracy.

The quality of object-level map reconstruction is also important. Similar to the surface reconstruction accuracy evaluation in dense SLAM systems, we quantify the reconstruction quality by computing the mean distances from each point in the reconstructed map to the nearest surface in the ground truth 3D model. The most commonly used metric is chamfer distance, and there are also some metrics, such as IoU or completeness. To obtain ground truth object models, we rendered some sequences with moving objects using the ground truth 3D CAD models from InteriorNet [Li et al., 2018b] and Shapenet [Chang et al., 2015].

In addition to quantitative evaluations, we also conducted extensive qualitative evaluations on pose estimation accuracy and object reconstruction quality in this thesis. The details will be discussed in the respective chapters.

1.4 Contributions

We described three contributions in this thesis to tackle tracking and reconstructing rigid moving objects. The main results have been presented in three different research papers. The full list of publications done in conjunction with this work as well as the video materials that provides visualisation of the algorithms are given in Section 1.5. The motivation and contribution of each paper are briefly discussed below.

1.4.1 Paper I: Octree-based Object-Level Multi-Instance Dynamic SLAM.

Research Question:
Can we design an object-level dynamic SLAM algorithm that can robustly estimate camera pose and also accurately estimate all the objects' geometric, semantic and motion information?

Context:
We propose a new multi-instance dynamic RGB-D SLAM system using an object-level octree-based volumetric representation. It can provide robust camera tracking in dynamic environments and at the same time, continuously estimate geometric, semantic, and motion properties for arbitrary objects in the scene. For each incoming frame, we perform instance segmentation to detect objects and refine mask boundaries using geometric and motion information. Meanwhile, we estimate the pose of each existing moving object using an object-oriented tracking method and robustly track the camera pose against the static scene. Based on the estimated camera pose and object poses, we associate segmented masks with existing models and incrementally fuse corresponding colour, depth, semantic, and foreground object probabilities into each object model. In contrast to existing approaches, our system is the first system to generate an object-level dynamic volumetric map from a single RGB-D camera. Our method can run at 2-3 Hz on a CPU, excluding the instance segmentation part. We demonstrate its effectiveness by quantitatively and qualitatively testing it on both synthetic and real-world sequences.

This object-level dynamic SLAM and the corresponding experiments are presented in Chapter 3.

Reference:

Binbin Xu, Wenbin Li, Dimos Tzoumanikas, Michael Bloesch, Andrew Davison, Stefan Leutenegger (2019). **MID-Fusion: Octree-based Object-Level Multi-Instance Dynamic SLAM**. *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).* [Xu et al., 2019].

1.4.2 Paper II: Deep Probabilistic Feature-metric Tracking

Research Question:

Can we have a better tracking algorithm to estimate camera and object poses under wide baseline and challenging lighting conditions?

Context:

Dense image alignment from RGB-D images remains a critical issue for real-world applications, especially under challenging lighting conditions and in a wide baseline setting. In this paper, we propose a new framework to learn a pixel-wise deep feature map and a deep feature-metric uncertainty map predicted by a Convolutional Neural Network (CNN), which together formulate a deep probabilistic feature-metric residual of the two-view constraint that can be minimised using Gauss-Newton in a coarse-to-fine optimisation framework. Furthermore, our network predicts a deep initial pose for faster and more reliable convergence. The optimisation steps are differentiable and unrolled to train in an end-to-end fashion. Due to its probabilistic essence, our approach can easily couple with other residuals, where we show a combination with ICP. Experimental results demonstrate state-of-the-art performances on the TUM RGB-D dataset and the 3D rigid object tracking dataset. We further demonstrate our method's robustness and convergence qualitatively.

The algorithm and the corresponding experimental results on object and camera trackings are presented in Chapter 4.

Reference:

Binbin Xu, Andrew J. Davison, Stefan Leutenegger (2021). **Deep Probabilistic Feature-metric Tracking**. *IEEE Robotics and Automation Letters (RA-L)*, Vol. 6, No. 1,pp. 223-230, 2021. [Xu et al., 2021a].

This paper was elected in ICRA 2021 presentation and received a **RA-L Best Paper Honorable Mention Award**.

1.4.3 Paper III: Object-level Dynamic SLAM with Map Completion

Research Question:

Can we incorporate object shape prior into object mapping and can this shape prior improve the object reconstruction quality and also pose estimation accuracy?

Context:

We propose a novel object-level dynamic SLAM system that can simultaneously segment, track, and reconstruct objects in dynamic scenes. It can further predict and complete the full geometry of the reconstructed objects by conditioning on the measured depth and category-level canonical shape prior, leading to better tracking accuracy. For each incoming RGB-D frame, we perform instance segmentation to detect objects and build data associations between the detection and the existing object maps. A new object model will be created for each unmatched detection. For each matched object, we jointly optimise its pose and latent representations using geometric and differential rendering residuals towards its shape prior and completed geometry. Our approach shows better tracking and reconstruction performance compared to methods using traditional volumetric or pure shape prior approaches. We evaluate its effectiveness by quantitatively and qualitatively testing it in both synthetic and real-world sequences.

The algorithm and the corresponding experimental results on object reconstruction and pose estimation are presented in Chapter 5.

Reference:
Binbin Xu, Andrew J. Davison, Stefan Leutenegger (2022). **Object-level Dynamic SLAM with Map Completion**. *(under submission).*

1.5 Publications

The work described in this thesis resulted in the following publications:

- Binbin Xu, Wenbin Li, Dimos Tzoumanikas, Michael Bloesch, Andrew Davison, Stefan Leutenegger (2019). **MID-Fusion: Octree-based Object-Level Multi-Instance Dynamic SLAM**. *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).* [Xu et al., 2019].
- Binbin Xu, Andrew J. Davison, Stefan Leutenegger (2021). **Deep Probabilistic Feature-metric Tracking**. *IEEE Robotics and Automation Letters (RA-L)*, Vol. 6, No. 1,pp. 223-230, 2021. (**RA-L Best Paper Honorable Mention Award** and selected in ICRA 2021 presentation) [Xu et al., 2021a].

and the following work that is currently under preparation for submission:

- Binbin Xu, Andrew J. Davison, Stefan Leutenegger (2022). **Object-level Dynamic SLAM with Map Completion**. *(under submission).*

The following video material provides visualisation of some of the algorithms developed in this thesis:

- MID-Fusion supplementary video at: `https://youtu.be/gturboNl9gg`.
- Deep Probabilistic Feature-metric Tracking supplementary video at: `https://youtu.be/6pMosl6ZAPE`.

While not described directly, the following work was conducted in conjunction with this thesis:

- Binbin Xu*, Lingni Ma*, Yuting Ye, Tanner Schmidt, Christopher D. Twigg, and Steven Lovegrove (2021). **DiForm: Identity-Disentangled Neural Deformation Model for Dynamic Meshes**. *Arxiv preprint arXiv:2109.15299*. [Xu et al., 2021b]
  This work was conducted during a research internship at Facebook Reality Labs Research.

*: equal contribution

1.6 Thesis Structure

The remainder of this thesis is structured as follows:

Chapter 2 introduces basic notation, the transformation groups and sensor models used in dense SLAM, and provides a primer on non-linear least-squares optimisation methods and tracking methods used in this work as well as a brief introduction on the map representation and deep neural networks related to this work.

Chapter 3 describes an octree-based object-level dynamic SLAM system. It can provide robust camera tracking in dynamic environments and at the same time, continuously estimate geometric, semantic, and motion properties for arbitrary objects in the scene. It is one of the first systems to generate an object-level dynamic volumetric map from a single RGB-D camera and can run at 2-3 Hz on a CPU, excluding the instance segmentation part. We demonstrate its effectiveness in robust camera tracking in a dynamic scene and accurate object geometry reconstruction.

Chapter 4 describes a probabilistic deep feature-metric tracking method to overcome the limitations of photometric residual used in Chapter 3. It is quantitatively evaluated on camera tracking and object tracking benchmarks and shows state-of-the-art performance. Qualitative demonstrations also show its robustness to lighting changes and large convergence basin.

Chapter 5 further improves object mapping component by incorporating category-level shape prior. Conditioning on both actual observations and latent codes, a learnt implicit function can predict complete object shape geometry. Experiments demonstrate this shape completion leads to better object reconstruction quality and also better object pose estimations.

Chapter 6 concludes this thesis with a summary of the results presented and discussions for promising future work.

Chapter 2

Preliminaries

In this chapter, we present fundamental concepts and related works that form the foundations for the algorithms presented in this thesis. In terms of layout, we start with the mathematical notation. We continue with the three-dimensional sensor models that are used inside our state estimation algorithms. We then introduce three-dimensional geometry knowledge, including the three-dimensional pose transformation groups that are used to represent the states of our sensor and object representations, and proceed with the nonlinear least-squares optimisation algorithms that are used to solve state estimation problems. Then we present some related works that have been served as a fundamental part in our work, especially the parts that are used to estimate the poses of the sensors and the object models in the environment, as well as the parts that are used for background and object map representations. We conclude with a general introduction to deep neural networks that are very closely related to the works presented in this thesis.

Contents of Chapter

2.1 Notation

This section introduces the notations used throughout this thesis. We will recap in each following chapter again the notations that are used in the respective work.

2.1.1 General notation

a	A lower-case symbol denotes a scalar.
$\mathbf{a}$	A bold lower-case symbol denotes an m-dimensional column vector.
$\mathbf{A}$	A bold capital symbol denotes an $m \times n$ matrix.
$\mathbf{I}$	The identity matrix, optionally with dimensions as subscript.
$\mathbf{0}$	The zero matrix, optionally with dimensions as subscript.
$[\cdot]^\times$	The cross-product matrix that produces a skew symmetric matrix from a 3D vector such that $\mathbf{a} \times \mathbf{b} = [\mathbf{a}]^\times \mathbf{b}$. Given the vector $\mathbf{a} = [a_x, a_y, a_z]^\top$, $[\mathbf{a}]^\times$ can be computed by: $$[\mathbf{a}]^\times = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}. \quad (2.1)$$ The properties of the cross product operation can be applied, such as the anticommutative property: $[\mathbf{a}]^\times \mathbf{b} = -[\mathbf{b}]^\times \mathbf{a}$, as well as the skew-symmetric property: $([\mathbf{a}]^\times)^\top = -[\mathbf{a}]^\times$. The combined properties can lead some useful equation, such as $\mathbf{a}^\top [\mathbf{b}]^\times = (-[\mathbf{b}]^\times \mathbf{a})^\top$.
$\mathcal{A}$	A calligraphic capital symbol denotes a set.

2.1.2 Spaces and manifolds

$\mathbb{R}$	The set of real numbers.

$\mathbb{R}^m$	The vector space of real m-dimensional vectors.
$\mathbb{R}^{m \times n}$	The vector space of real $m \times n$-dimensional matrices.
$SO(3)$	Special Orthogonal group: the group of 3D rotations.
$so(3)$	lie algebra associated with $SO(3)$.
$SE(3)$	Special Euclidean group: the group of 3D rigid transformations.
$\boxplus$	The "box-plus" operator that applies a small perturbation expressed in a tangent space to a manifold state, e.g. $SO(3) \times \mathbb{R}^3 \longrightarrow SO(3)$.

2.1.3 Frames and transformations

$\underrightarrow{\mathcal{F}}_A$	A cartesian coordinate frame in $\mathbb{R}^3$.
${}_A\mathbf{v}$	A vector $\mathbf{v}$ expressed in the frame $\underrightarrow{\mathcal{F}}_A$, for example a 3D world vertex position in the frame $\underrightarrow{\mathcal{F}}_A$.
${}_A\mathbf{r}_P$	The position vector from the origin of $\underrightarrow{\mathcal{F}}_A$ to the point P represented in the coordinate frame $\underrightarrow{\mathcal{F}}_A$.
${}_A\mathbf{r}_{BC}$	The vector that represents the vector from $\underrightarrow{\mathcal{F}}_B$ to $\underrightarrow{\mathcal{F}}_C$, represented in $\underrightarrow{\mathcal{F}}_A$.
$\mathbf{C}_{AB}$	The rotation matrix that transforms the vector ${}_B\mathbf{v}$ expressed in $\underrightarrow{\mathcal{F}}_B$ to one expressed in $\underrightarrow{\mathcal{F}}_A$ as: ${}_A\mathbf{v} = \mathbf{C}_{AB}\,{}_B\mathbf{v}$. The inverse rotation $\mathbf{C}_{BA}$ can be computed as: $\mathbf{C}_{BA} = \mathbf{C}_{AB}^{-1} = \mathbf{C}_{AB}^{\top}$.
${}_A\mathbf{r}_{AB}$	The translation vector that represents the vector from $\underrightarrow{\mathcal{F}}_A$ to $\underrightarrow{\mathcal{F}}_B$, represented in $\underrightarrow{\mathcal{F}}_A$.
$\boldsymbol{T}_{AB}$	The transformation matrix that transforms homogeneous vectors from $\underrightarrow{\mathcal{F}}_B$ to $\underrightarrow{\mathcal{F}}_A$ as ${}_A\boldsymbol{r}_P = \boldsymbol{T}_{AB}\,{}_B\boldsymbol{r}_P$.
$\boldsymbol{u}_A$	2D pixel position represented on the frame $\underrightarrow{\mathcal{F}}_A$

2.2 Camera Model

One of the most important and widely used sensors on the robot platform is the camera as it is cheap and can capture very rich texture environment information. We use the pinhole camera model to model the RGB and RGB-D cameras used in this thesis.

The pinhole camera model can be mathematically described using the perspective projection function. Assuming lens undistortion has been applied to the images, a 2D pixel coordinate $\boldsymbol{u}_I = (u, v)$, on the 2D image plane I is projected from a 3D point position ${}_C\mathbf{r}_p = (x, y, z)$ in the camera coordinate $\underrightarrow{\mathcal{F}}_C$.

$$\boldsymbol{u}_I = \mathbf{K}[\boldsymbol{\pi}({}_C\mathbf{r}_p)], \tag{2.2}$$

where $\boldsymbol{\pi}(\cdot)$ is a perspective projection function and removes bottom row from the homogeneous point representation

$$\boldsymbol{\pi}({}_C\mathbf{r}_p) = \boldsymbol{\pi}\left(\begin{bmatrix} x \\ y \\ z \end{bmatrix}\right) = \frac{1}{z}\begin{bmatrix} x \\ y \end{bmatrix}. \tag{2.3}$$

$\mathbf{K}$ is the camera intrinsic matrix to map the normalized image coordinates to the actual pixel coordinates:

$$\mathbf{K} = \begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.4}$$

The intrinsic matrix $\mathbf{K}$ is composed of the camera focal length expressed in horizontal, f_u, and vertical axis, f_v, as well as the offset of the image origin from the optical axis intersection in horizontal c_u, and in vertical pixels, c_v. These values can be estimated in the calibration stage to remove the lens effects.

If the depth of a certain pixel, d, is known, we can recover its corresponding 3D point position by inverting the perspective projection function:

$$ {}_C\mathbf{r}_p = \boldsymbol{\pi}^{-1}[\mathbf{K}^{-1}(\boldsymbol{u}_I), d], \tag{2.5}$$

where $\mathbf{K}^{-1}$ is the inverse of the intrinsic matrix to normalize the pixel coordinates to the normalized pixel coordinates:

$$\mathbf{K}^{-1} = \begin{bmatrix} \frac{1}{f_u} & 0 & -\frac{c_u}{f_u} \\ 0 & \frac{1}{f_v} & -\frac{c_v}{f_v} \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.6}$$

and $\boldsymbol{\pi}^{-1}()$ is a back-projection function to recover the 3D point position when its depth is known.

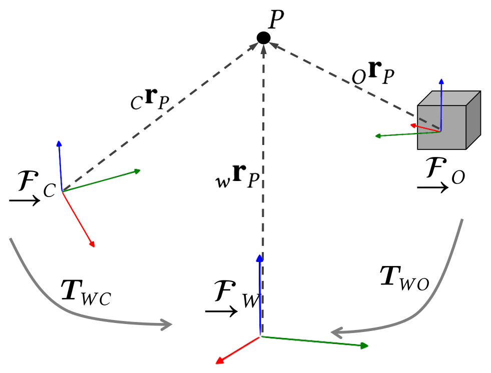
2.3 Transformations

In this thesis, we proposed several novel systems to tackle the object-level visual SLAM problem. Each camera frame, object pose, and the global world scene are represented in their own respective frames. Rigid transformations between these coordinate frames are used to represent the corresponding camera pose and object poses in these scenes.

A camera pose is defined as the relative transformation of the camera coordinate w.r.t. the world coordinate and belongs to Special Euclidean group, $SE(3)$. It consists of a 3DoF rotation and a 3DoF translation. One common parametrization choice for it is using a 4×4 homogeneous matrix $\boldsymbol{T}_{AB}$, and we express it as $\boldsymbol{T}_{AB} = (\mathbf{C}_{AB}, {}_A\mathbf{r}_{AB}) \in (SO(3) \times \mathbf{R}^3)$, from $\underrightarrow{\mathcal{F}}_B$ to $\underrightarrow{\mathcal{F}}_A$. The 3×3 rotation matrix $\mathbf{C}$ belongs to Special Orthogonal group, $SO(3)$, such that $\mathbf{C}^T\mathbf{C} = \mathbf{I}$ and $det(\mathbf{C}) = 1$.

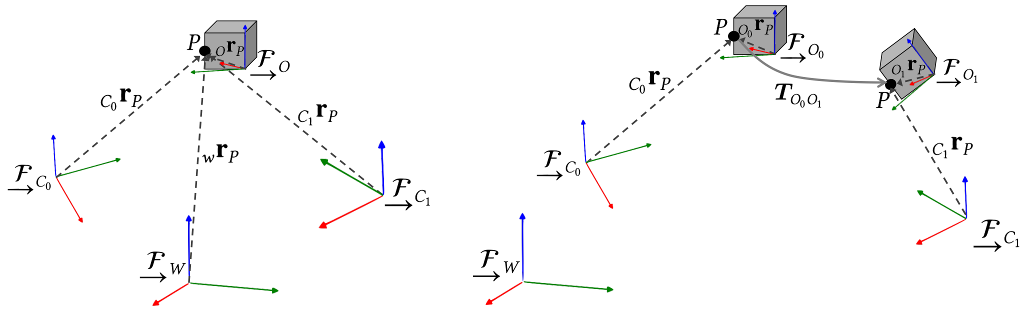
The relationship between $\boldsymbol{T}_{AB}$, the rotation matrix $\mathbf{C}_{AB}$ and the position vector ${}_A\mathbf{r}_{AB}$ is given by:

$$\boldsymbol{T}_{AB} = \begin{bmatrix} \mathbf{C}_{AB} & {}_A\mathbf{r}_{AB} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}, \tag{2.7}$$



P
${}_C\mathbf{r}_P$
${}_O\mathbf{r}_P$
$\underrightarrow{\mathcal{F}}_C$
${}_W\mathbf{r}_P$
$\underrightarrow{\mathcal{F}}_O$
$\boldsymbol{T}_{WC}$
$\underrightarrow{\mathcal{F}}_W$
$\boldsymbol{T}_{WO}$

Figure 2.1: A depiction of a point P expressed in the world coordinate $\underrightarrow{\mathcal{F}}_W$, camera coordinate $\underrightarrow{\mathcal{F}}_C$ and object coordinate $\underrightarrow{\mathcal{F}}_O$ as ${}_W\mathbf{r}_P$, ${}_C\mathbf{r}_P$, and ${}_O\mathbf{r}_P$, respectively. The relative transformation of the camera coordinate w.r.t the world coordinate represents the camera pose. The relative transformation of the object coordinate w.r.t the world coordinate represents the object pose.



P
${}_O\mathbf{r}_P$
$\underrightarrow{\mathcal{F}}_O$
${}_{C_0}\mathbf{r}_P$
${}_{C_1}\mathbf{r}_P$
${}_W\mathbf{r}_P$
$\underrightarrow{\mathcal{F}}_{C_0}$
$\underrightarrow{\mathcal{F}}_{C_1}$
$\underrightarrow{\mathcal{F}}_W$
P
${}_{O_0}\mathbf{r}_P$
$\underrightarrow{\mathcal{F}}_{O_0}$
${}_{O_1}\mathbf{r}_P$
$\underrightarrow{\mathcal{F}}_{O_1}$
${}_{C_0}\mathbf{r}_P$
$\boldsymbol{T}_{O_0O_1}$
P
${}_{C_1}\mathbf{r}_P$
$\underrightarrow{\mathcal{F}}_{C_0}$
$\underrightarrow{\mathcal{F}}_{C_1}$
$\underrightarrow{\mathcal{F}}_W$

(a) Static scene (b) Dynamic scene

Figure 2.2: Multi-view constraint in static and dynamic scenes

while its inverse transformation $\boldsymbol{T}_{BA}$ can be computed as:

$$\boldsymbol{T}_{BA} = \boldsymbol{T}_{AB}^{-1} = \begin{bmatrix} \mathbf{C}_{AB}^{\top} & -\mathbf{C}_{AB}^{\top}\,{}_A\mathbf{r}_{AB} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{2.8}$$

Similarly, an object pose is defined as an $SE(3)$ transformation between the object coordinate and the world coordinate.

A visualisation of a point P expressed in these three coordinate frames and with the transformations among them is shown in Figure 2.1. Part of the works in this thesis focuses on estimating the camera pose $\boldsymbol{T}_{WC_i}$ and the object pose $\boldsymbol{T}_{WO_i}$ at the timestamp i by utilizing the correspondence in the multiple view constraint. The world coordinate $\underrightarrow{\mathcal{F}}_W$ can be arbitrarily defined, though conventionally it is chosen to coincide with the camera coordinate on the first frame. If the world and the object are static, the 3D landmark P has a constant spatial position vector ${}_W\mathbf{r}_P$ in the world coordinate $\underrightarrow{\mathcal{F}}_W$. So we have the following multi-view constraint when the same static landmark is observed at different timestamps in the camera coordinate ${}_{C_i}\mathbf{r}_P$:

$$\boldsymbol{T}_{WC_0}\,{}_{C_0}\mathbf{r}_P = \boldsymbol{T}_{WC_1}\,{}_{C_1}\mathbf{r}_P = {}_W\mathbf{r}_P. \tag{2.9}$$

The difference of 3D location between ${}_{C_0}\mathbf{r}_P$ and ${}_{C_1}\mathbf{r}_P$ is only caused by camera motion. However, when the object is moving, the constraint in Equation (2.9) does not hold anymore since the ${}_W\mathbf{r}_P$ can also change. Instead, we use the rigid body constraint that the same 3D point on a rigid body remains a constant position in the object body frame, which is set as the object canonical space. Then we can take advantage of the following constraint:

$$\boldsymbol{T}_{WO_0}^{-1}\,\boldsymbol{T}_{WC_0}\,{}_{C_0}\mathbf{r}_P = \boldsymbol{T}_{WO_1}^{-1}\,\boldsymbol{T}_{WC_1}\,{}_{C_1}\mathbf{r}_P = {}_O\mathbf{r}_P. \tag{2.10}$$

Figure 2.2 visualizes the constraint difference. Here we slightly abuse the notation ${}_A\mathbf{r}_P$ to express the homogeneous coordinates of a vector, ${}_A\boldsymbol{r}_P = [{}_A\mathbf{r}_P^\top, 1]^\top$.

2.3.1 Lie Algebra

The homogeneous matrix is convenient to express transformations via simple multiplications. However, its 9-parameter rotation matrix form is over-parameterized for 3DoF rotations as its elements are not independent. There are also other parameterisations for rotations, such as Euler angles, angle-axis, quaternions, and lie algebra. Each of these parameterisations can be useful for particular tasks. However, the representations that have more than three parameters must have associated constraints to limit their degrees of freedom. The representations that have exactly three parameters have associated singularities, and thus there is no perfect representation that is minimal and also free of singularities. In our work, we primarily employ lie algebra to parametrise rotations for iterative pose optimisation.

Both $SE(3)$ and $SO(3)$ are matrix Lie groups. A group is mathematically defined as a set of elements together with an operation that applies to any two of its elements to form a third element that also belongs to the group. A group also must satisfy four conditions called the group axioms. A Lie group is a specific group that is also a differential manifold, and whose group operations are also smooth. A matrix Lie group further specifies that the elements of the group are matrices, the combination operation is matrix multiplication, and the inversion operation is matrix inversion.

As a smooth manifold, a Lie group has an associated Lie algebra, which is the local tangent space around the identity of the group. For rotations, the linear algebra associated with $SO(3)$ is defined as

$$so(3) = \{\mathbf{\Phi} = \boldsymbol{\phi}^{\wedge} \in \mathbb{R}^{3\times 3} \mid \boldsymbol{\phi} \in \mathbb{R}^{3}\}. \tag{2.11}$$

$(\cdot)^{\wedge}$ is the equivalent operation symbol as the $[\cdot]^{\times}$ and its inverse operation symbol is $(\cdot)^{\vee}$:

$$\boldsymbol{\phi} = \mathbf{\Phi}^{\vee} \tag{2.12}$$

The exponential map is the key to map each element from $so(3)$ to $SO(3)$ using Rodrigues' rotation formula:

$$\mathbf{C} = \exp(\boldsymbol{\phi}^{\wedge}). \tag{2.13}$$

Inversely, we can also map from $SO(3)$ to $so(3)$:

$$\boldsymbol{\phi} = \log(\mathbf{C})^{\vee}. \tag{2.14}$$

The exponential mapping is subjective as its inverse log mapping is not unique. We can find multiple elements in $so(3)$ mapping to the same element in $SO(3)$. Lie algebra can be converted to the axis-angle form of a rotation matrix by setting

$\boldsymbol{\phi} = \phi\boldsymbol{\alpha}$, where $\phi = |\boldsymbol{\phi}|$ is the rotation angle and $\boldsymbol{\alpha} = \boldsymbol{\phi}/\phi$ is the unit-length axis of rotation. In practice, we limit $|\phi|$ to be smaller than π to limit the log mapping candidates.

The associated Lie algebra is an especially natural place to perform iterative updates on the lie group element. It creates a vector space with the same dimensionality as the group's number of DoF, allowing each step to freely move within the entire vector space. In each iterative optimisation, a small update presented in a minimal parameterisation belonging to $\mathbb{R}^3$ can be used to update the group element on the manifold by:

$$\begin{aligned} \mathbf{C} &= \overline{\mathbf{C}} \boxplus \delta\boldsymbol{\phi} & (2.15) \\ &= \exp(\delta\boldsymbol{\phi}^\wedge)\overline{\mathbf{C}}, & (2.16) \end{aligned}$$

where the update $\delta\boldsymbol{\phi}$ is computed around the currently estimated $\overline{\mathbf{C}} \in SO(3)$.

Similarly, rigid transformations can also be expressed in a minimal representation $\boldsymbol{\xi} \in \mathbb{R}^6$:

$$se(3) = \{\boldsymbol{\xi}^\wedge = \begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{\phi} \end{bmatrix}^\wedge = \begin{bmatrix} \boldsymbol{\phi}^\wedge & \boldsymbol{\rho} \\ \mathbf{0}^\mathrm{T} & 0 \end{bmatrix} \in \mathbb{R}^{4\times4} \mid \boldsymbol{\rho}, \boldsymbol{\phi} \in \mathbb{R}^3\}. \tag{2.17}$$

In terms of exponential mapping, there are two choices to express it, leading to different expressions for Jacobian computations and pose updating. The orientation exponential mapping is the same as the one in $SO(3)$. For translation part, one is to map $\boldsymbol{\rho}$ via left Jacobian: $\mathbf{r} = \mathbf{J}\boldsymbol{\rho} \in \mathbb{R}^3$. This is referred to as the $SE(3)$ way as it is consistent in $SE(3)$ space. The other is to directly map $\boldsymbol{\rho}$ to the translation component: $\mathbf{r} = \boldsymbol{\rho} \in \mathbb{R}^3$. We refer to it as $SO(3)$ way as it separates the translation part from the orientation part. In this thesis, we adopt the second choice. When a small pose update $\boldsymbol{\xi} \in \mathbb{R}^6$ is computed around the currently estimated rotation $\overline{\mathbf{C}}$ and translation $\overline{\boldsymbol{\rho}}$

$$\delta\boldsymbol{\xi} = [\delta\boldsymbol{\rho}, \delta\boldsymbol{\phi}] \in \mathbb{R}^6, \tag{2.18}$$

it can be applied back on the $SE(3)$ manifold as:

$$\boldsymbol{\rho} = \overline{\boldsymbol{\rho}} \boxplus \delta\boldsymbol{\rho}, \mathbf{C} = \overline{\mathbf{C}} \boxplus \delta\boldsymbol{\phi}. \tag{2.19}$$

More details of the lie groups and lie algebra in computer vision and robotics can be found in [Eade, 2014, Bloesch et al., 2016, Barfoot, 2017].

2.4 Nonlinear Least-Squares Optimisation

Modern SLAM systems typically formulate the tracking and mapping problem as nonlinear least-squares optimisation problems. For a set of state parameters, $\mathbf{x} \in \mathbb{R}^m$, we aim to find the parameters $\tilde{\mathbf{x}}$ that can minimize the differences between the predicted observations, $h(\mathbf{x})$, with the actual measurements, $\mathbf{z}$. This difference is measured by an error function:

$$\mathbf{e}(\mathbf{x}) = h(\mathbf{x}) - \mathbf{z}. \tag{2.20}$$

The error function is also called a residual function in some other papers. It can be further formulated in a cost function:

$$E(\mathbf{x}) = \rho(||\mathbf{e}(\mathbf{x})||_{\mathbf{W}}). \tag{2.21}$$

$\mathbf{W}$ is symmetric positive-definite (and often diagonal if we assume the measurement in each different is independent) weighting matrix and can be formulated in the Mahalanobis norm $|| \cdot ||_{\mathbf{W}}$. In robotics, this weighted matrix is often defined by the inverse covariance matrix associated with the measurement. $\rho(\cdot)$ is a robust loss function that is used to down-weigh outliers in the optimisation problem, and is often chosen empirically or based on the distribution of the residuals [Concha and Civera, 2015, MacTavish and Barfoot, 2015] .

2.4.1 Gauss-Newton Algorithm

Equation (2.21) is typically solved using a non-linear least-squares optimisation algorithm, such as Gauss-Newton or Levenberg-Marquardt algorithm. The convergence of these approaches, however, is not guaranteed and it may lead to a local minimum [Eade, 2009].

Each iteration k of the Gauss-Newton system solves the state update $\delta\mathbf{x}$ for

$$\mathbf{x}_{k+1} = \mathbf{x}_k \boxplus \delta\mathbf{x} \tag{2.22}$$

by linearizing the residual function $\mathbf{e}(\mathbf{x})$ using first-order Taylor expansion

$$\begin{aligned} E(\mathbf{x}) &= \rho(\|\mathbf{e}(\mathbf{x}_{k+1})\|_{\mathbf{W}}) && (2.23) \\ &= \rho(\|\mathbf{e}(\mathbf{x}_k \boxplus \delta\mathbf{x})\|_{\mathbf{W}}) && (2.24) \\ &\approx \rho(\|\mathbf{e}(\mathbf{x}_k) + \mathbf{J}\delta\mathbf{x}\|_{\mathbf{W}}), && (2.25) \end{aligned}$$

where the Jacobian matrix $\mathbf{J}$ is a function of $\mathbf{x}_k$,

$$\mathbf{J} = \left.\frac{\partial\mathbf{e}(\mathbf{x})}{\partial\mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_k}, \tag{2.26}$$

and needs to be re-evaluated at each iteration. Equation (2.26) can be analytically computed using the chain rule.

To minimize Equation (2.25), we can set the derivative w.r.t. the parameter update to be zero, and obtain the normal equation:

$$\delta\mathbf{x} = -(\mathbf{J}^T\mathbf{W}\mathbf{J})^{-1}\mathbf{J}^T\mathbf{W}\mathbf{x}_k. \tag{2.27}$$

For brevity the robust loss function $\rho()$ and its derivative are dropped from Equation (2.27). In practice, they are typically included in the weight matrix $\mathbf{W}$.

Here $\mathbf{J}^T\mathbf{W}\mathbf{J}$ is an approximation to the true Hessian of Equation (2.21) with respect to the parameters $\mathbf{x}$ to speed up computation. As the approximate Hessian can become ill-conditioned to compute its inverse, a damping term is added in practice. If the damping parameter λ is adaptable in each iteration, this results in the Levenberg–Marquardt (trust-region) update equation. Different λ values can adjust the parameter update $\delta\mathbf{x}$ between the Gauss-Newton update (smaller λ) and gradient descent (larger λ). The choice of the damping parameter determines the speed of which the algorithm converges to a local optima.

The iterative optimisation begins with an initial guess of the parameters, $\mathbf{x}_0$ and then iteratively updates until the updates become sufficiently small, or a certain number of iterations is reached or stops early when the residual starts to go up. Besides, when working with image inputs, instead of repeating this process on the same resolution, a coarse-to-fine approach is typically applied. This speeds up the optimisation and makes it less likely to get stuck in the local minimum by providing a wider basin of convergence.

2.4.2 Inverse Compositional Algorithm

In a non-linear least squares framework, the Jacobian in Equation (2.26) needs to be re-computed at each iteration, as it is a function of $\mathbf{x}_k$. To avoid recomputing it at every iteration, the inverse compositional algorithm [Baker and Matthews, 2004] pre-computes the Jacobian to save computational resource. This is achieved by computing the parameter update on the measurement part, instead of the predicted observation part:

$$E(\mathbf{x}) = \rho(||\mathbf{e}(\mathbf{x}_{k+1})||_{\mathbf{W}}) \tag{2.28}$$

$$= \rho(||h(\mathbf{x}_k) - \mathbf{z}(\delta\mathbf{x})||_{\mathbf{W}}). \tag{2.29}$$

Accordingly, the state parameter is updated in the inverse way in each iteration:

$$\mathbf{x}_{k+1} = \mathbf{x}_k \boxplus (\delta\mathbf{x})^{-1}. \tag{2.30}$$

In this way, the Jacobian

$$\mathbf{J} = \frac{\partial\mathbf{z}(0)}{\partial\mathbf{x}} \tag{2.31}$$

in the linearization function

$$E(\mathbf{x}) = \rho(||h(\mathbf{x}_k) - \mathbf{z}(\delta\mathbf{x})||_{\mathbf{W}}) \tag{2.32}$$

$$\approx \rho(||h(\mathbf{x}_k) - \mathbf{z}(0) - \mathbf{J}\delta\mathbf{x}||_{\mathbf{W}}), \tag{2.33}$$

does not depend on $\mathbf{x}_k$ and thus can be pre-computed to speed up the optimisation.

2.4.3 Jacobians in Rigid Transformations

The non-linearity in the optimisation problem largely comes from the orientation component and thus the derivative of a transformed point with respect to the Lie algebra parameters that transforms it is of particular importance. Here we give a short summary of the lie algebra Jacobians that will be applied in the following chapters.

To optimise the $\boldsymbol{T}_{AB}$ in the forward rigid transformation:

$${}_A\mathbf{r}_p = \boldsymbol{T}_{AB}\, {}_B\mathbf{r}_p, \tag{2.34}$$

The 3×6 Jacobian of ${}_A\mathbf{r}_p$ with respect to the state update $\delta\xi$ for ξ_{AB} is:

$$\frac{\partial {}_A\mathbf{r}_p}{\partial \delta\xi} = \left[\; -\left[{}_A\mathbf{r}_p\right]^\times \;\;|\, \mathbf{I} \;\right]. \tag{2.35}$$

To optimise the inverse rigid transformation $\boldsymbol{T}_{AB}^{-1}$:

$${}_B\mathbf{r}_p = \boldsymbol{T}_{AB}^{-1}\, {}_A\mathbf{r}_p, \tag{2.36}$$

The 3×6 Jacobian of ${}_B\mathbf{r}_p$ with respect to the state update $\delta\xi$ for ξ_{AB} is:

$$\frac{\partial {}_B\mathbf{r}_p}{\partial \delta\xi} = \left[\; \mathbf{C}_{AB}^T \left[{}_A\mathbf{r}_p\right]^\times \;\;|\; -\mathbf{C}_{AB}^T \;\right]. \tag{2.37}$$

The notation of ${}_A\mathbf{r}_p$ is slightly abused here. It is expressed in the homogeneous coordinates when expressing transformation in Equations (2.34) and (2.36) and in the inhomogeneous coordinates when expressing Jacobians in Equations (2.35) and (2.37).

2.5 Dense Tracking

In this thesis, our camera and object tracking components are in the category of dense direct SLAM that alternates between the tracking and mapping steps without explicit feature extraction and matching steps. Unlike sparse tracking, which estimates a camera/object pose by minimising the reprojection error over a set of sparse keypoints, dense tracking optimises over all the pixel intensities and geometric information.

2.5.1 Iterative Closest Point (ICP) tracking

When using a depth camera, fast Iterative Closest Point (ICP) [Rusinkiewicz and Levoy, 2001] is often used to register an incoming live depth frame to a reference pointcloud or a reference 3D reconstruction model [Newcombe et al., 2011a] by minimizing the distances between the corresponding points. In each iteration, projective data association [Blais and Levine, 1995] is used to build dense correspondences between the two pointclouds rather than searching for the closest two points in terms of Euclidean distance. After each optimisation iteration, the estimated transform is applied to the source point cloud, a new set of associations are built again based on the projectively closest points and the procedure is repeated.

In ICP tracking, we first back-project each pixel $\boldsymbol{u}_L$ in the incoming live depth image to a 3D vertex point ${}_{C_L}\mathbf{v}$ in the live camera frame, $\underrightarrow{\mathcal{F}}_{C_L}$ using Equation (2.5). This creates a live 3D vertex map, which can be transformed into the reference frame $\underrightarrow{\mathcal{F}}_R$using the current estimate of the camera pose with the reference camera pose, $\boldsymbol{T}_{WC_R}^{-1}\boldsymbol{T}_{WC_L}$. The correspondence $\boldsymbol{u}_R$ in the reference image for $\boldsymbol{u}_L$ can be found by re-projecting the live pixel $\boldsymbol{u}_L$ into the reference image plane:

$$\boldsymbol{u}_R = \mathbf{K}\boldsymbol{\pi}(\boldsymbol{T}_{WC_R}^{-1}\boldsymbol{T}_{WC_L}(\boldsymbol{\pi}^{-1}\mathbf{K}^{-1}(\boldsymbol{u}_L, D_L[\boldsymbol{u}_L]))). \tag{2.38}$$

The ICP residual is defined as the point-to-plane ICP residual [Chen and Medioni, 1992, Newcombe et al., 2011a] using the reference surface normal ${}_W\mathbf{n}^r$ in the world coordinate $\underrightarrow{\mathcal{F}}_W$ as well as the correspondence vertex distance error:

$$e_{\text{ICP}}(\boldsymbol{T}_{WC_L}) = {}_W\mathbf{n}^r[\boldsymbol{u}_R] \cdot \left(\boldsymbol{T}_{WC_L}\, {}_{C_L}\mathbf{v}[\boldsymbol{u}_L] - {}_W\mathbf{v}^r[\boldsymbol{u}_R]\right). \tag{2.39}$$

The reference vertex $\mathbf{v}^r[\boldsymbol{u}_R]$ and reference normal $\mathbf{n}^r[\boldsymbol{u}_R]$ are found via raycasting [Parker et al., 1998] to the reference 3D model. Here we choose the reference normal in the residual, as the reference normal vector is estimated from the fused reference TSDF model that contains much less noise than the live depth image. To handle the outlier noise on the live depth image, we first run bilateral filtering on the depth image and also filter out the live vertices whose vertice distance $\boldsymbol{T}_{WC_L}\, {}_{C_L}\mathbf{v}[\boldsymbol{u}_L] - {}_W\mathbf{v}^r[\boldsymbol{u}_R]$ or normal divergence ${}_W\mathbf{n}^r[\boldsymbol{u}_R] \cdot \mathbf{C}_{WC_L}\, {}_{C_L}\mathbf{n}[\boldsymbol{u}_L]$ with the reference point are too large. This residual has been found to work quite well for projective data association, as it allows the correspondences to "slide" on the surfaces.

The ICP residual in Equation (2.39) can be iteratively solved using the Gauss-Newton algorithm, as described in Section 2.4.1. The state parameter to be optimised is ξ_{WC_L} and its Jacobian can be computed following the chain rule and using Equation (2.35):

$$\mathbf{J}_{\text{ICP}}(\xi_{WC_L}) = \frac{\partial e_{\text{ICP}}(\boldsymbol{T}_{WC_L})}{\partial \delta \xi} = -\left[\ \left(\boldsymbol{T}_{WC_L}\, {}_{C_L}\mathbf{v}[\boldsymbol{u}_L]\right) \times {}_W\mathbf{n}^r[\boldsymbol{u}_R] \quad \middle|\ {}_W\mathbf{n}^r[\boldsymbol{u}_R]\ \right]. \tag{2.40}$$

Here we describe the Jacobian in the forward compositional way. The inverse compositional Jacobian will be described in the later chapters when it is used in the corresponding system. ξ_{WC_L} can be iteratively updated using Equation (2.27). The inverse of the approximate Hessian matrix can be efficiently solved using Cholesky Decomposition, QR decomposition, or Singular Value Decomposition.

2.5.2 Photometric (RGB) tracking

Photometric (RGB) tracking minimizes the photometric, i.e. pixel intensity, difference between a live RGB image and a reference one. It has been widely adopted in many SLAM systems to optimise a relative transformation that aligns the live frame to a reference frame since it was proposed in [Steinbrücker et al., 2011]. It works by rendering the live frame, $\underrightarrow{\mathcal{F}}_L$ into the reference frame, $\underrightarrow{\mathcal{F}}_R$, by back-projection and re-projection operations and then minimizing the photometric residual:

$$\begin{aligned} e_{RGB}(0) &= I_R[\boldsymbol{u}_R] - {}_RI_L && (2.41)\\ &= I_R[\boldsymbol{u}_R] - I_L[\mathbf{K}\boldsymbol{\pi}(\boldsymbol{T}_{WL}^{-1}(\boldsymbol{T}_{WR}\,\boldsymbol{\pi}^{-1}\mathbf{K}^{-1}(\boldsymbol{u}_R, D_R[\boldsymbol{u}_R])))] && (2.42)\end{aligned}$$

In the following chapter, we seek to optimise the transformation $\boldsymbol{T}_{WL}$ using the depth rendered from the 3D reconstruction model, which has higher depth quality than raw depth measurements.

Some intermediate terms in Equation (2.42) have their physical meaning in this image rendering process and can be denoted as:

$$e_{RGB}(0) = I_R[\boldsymbol{u}_R] - I_L\,(\underbrace{\mathbf{K}\boldsymbol{\pi}\,(\overbrace{\boldsymbol{T}_{WL}^{-1}\,\underbrace{(\boldsymbol{T}_{WR}\,\boldsymbol{\pi}^{-1}\mathbf{K}^{-1}(\boldsymbol{u}_R, D_R[\boldsymbol{u}_R]))}_{{}_W\mathbf{v}}}^{{}_L\mathbf{v}}))}_{\boldsymbol{u}_L} \qquad (2.43)$$

With small perturbation, i.e. parameter update, $\delta\xi$ performed on the $\boldsymbol{T}_{WL}$ we can obtain the following equation:

$$e_{RGB}(\delta\xi) = I_R[\boldsymbol{u}_R] - I_L(\mathbf{K}\boldsymbol{\pi}(\boldsymbol{T}_{WL}^{-1}\,\exp((-\delta\xi)^{\wedge})_W\mathbf{v})) \qquad (2.44)$$

Thus, we can get the photometric Jacobian formula as:

$$\begin{aligned} J &= \frac{\partial(e_{RGB}(\delta\xi) - e_{RGB}(0))}{\partial(\delta\xi)} && (2.45)\\ &= -\frac{\partial\left(I_L(\mathbf{K}\boldsymbol{\pi}(\boldsymbol{T}_{WL}^{-1}\,\exp((-\delta\xi)^{\wedge})_W\mathbf{v}) - I_L(\mathbf{K}\boldsymbol{\pi}(\boldsymbol{T}_{WL}^{-1}\,({}_W\mathbf{v})\right)}{\partial(\delta\xi)} && (2.46)\end{aligned}$$

Using the chain rule, we can decompose Equation (2.46) as three parts:

$$J = -\frac{\partial\big(I_L(\boldsymbol{u}_L)\big)}{\partial(\boldsymbol{u}_L)}\frac{\partial(\boldsymbol{u}_L)}{\partial({}_L\mathbf{v})}\frac{\partial({}_L\mathbf{v})}{\partial(\delta\xi)}. \tag{2.47}$$

1. The first term is the image gradient of the rendered image $\triangledown_R I_L$ on the pixel coordinate $\boldsymbol{u}_L$ and it requires (bilinear-)interpolation if the projection pixel coordinate $\boldsymbol{u}_L$ is sub-pixel.

2. The second item is the derivative towards perspective projection from the 3D world vertex ${}_L\mathbf{v}$, (x, y, z) onto the 2D pixel position $\boldsymbol{u}_L$, expanded as (u, v) in the coordinate $\underrightarrow{\mathcal{F}}_L$.
   Using Equations (2.3) and (2.4), we can get

$$u = f_x\frac{x}{z} + c_x, \tag{2.48}$$

$$v = f_y\frac{y}{z} + c_y, \tag{2.49}$$

   and then the second term in the Jacobian equation would be:

$$\frac{\partial(\boldsymbol{u}_L)}{\partial({}_L\mathbf{v})} = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} & \frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial v}{\partial z} \end{bmatrix} = \begin{bmatrix} \frac{f_x}{z} & 0 & -\frac{f_x x}{z^2} \\ 0 & \frac{f_y}{z} & -\frac{f_y y}{z^2} \end{bmatrix}. \tag{2.50}$$

3. The third term is the derivative towards the vertex ${}_L\mathbf{v}$ in the live coordinate $\underrightarrow{\mathcal{F}}_L$ transformed form the ${}_W\mathbf{v}$ under the small perturbation $\delta\xi$ on the transformation $\boldsymbol{T}_{WL}$. Using Equation (2.37), we can get

$$\frac{\partial({}_L\mathbf{v})}{\partial(\delta\xi)} = \frac{\partial(\boldsymbol{T}_{WL}^{-1}\,{}_W\mathbf{v})}{\partial(\delta\xi)} \tag{2.51}$$

$$= \begin{bmatrix} \mathbf{C}_{WL}^T\,[{}_W\mathbf{v}]^\times & -\mathbf{C}_{WL}^T \\ \mathbf{0}^T & \mathbf{0}^T \end{bmatrix} \tag{2.52}$$

   The notation of ${}_W\mathbf{v}$ is slightly abused here. It is expressed in the homogeneous coordinate from Equation (2.51) and inhomogeneous coordinate from Equation (2.52).

Inserting Equation (2.50), Equation (2.52) into Equation (2.47), we can get the Jacobian used in the photometric tracking:

$$J = \triangledown_R I_L(\boldsymbol{u}_L) \begin{bmatrix} \frac{f_x}{z} & 0 & -\frac{f_x x}{z^2} \\ 0 & \frac{f_y}{z} & -\frac{f_y y}{z^2} \end{bmatrix} \begin{bmatrix} -\mathbf{C}_{WL}^T & \mathbf{C}_{WL}^T \left[{}_W\mathbf{v}\right]^\times \end{bmatrix} \tag{2.53}$$

2.6 Map Representations

As discussed above, the quality of depth map estimation has a large impact on tracking performance. While we can measure depth directly from a depth sensor, the measurement depth also has its own reliable range and measurement uncertainty. Fusing depth maps from multiple measurements can remove outliers and reduce noise as well as the measurement uncertainties. It can also reduce the odometry drift and further improve the tracking accuracy when combined with other optimisation methods, such as pose graph optimisation, or factor graph optimisation. The exact depth fusion method largely depends on the map representation choice, which has been explored in various ways and is still one of the hottest topics in the SLAM community nowadays.

Scene representation in visual SLAM initially focuses on building sparse pointcloud maps, coupled well with sparse indirect feature-based SLAM systems [Davison, 2003, Klein and Murray, 2007, Mur-Artal and Tardós, 2017]. With the advancement of commodity Graphic Processing Units (GPU) and especially the cheap depth sensors, dense mapping has also become a popular choice SLAM since it is useful for other robotics tasks, such as path planning and collision avoidance. Dense mapping typically fuses multiple view depth measurements into a global volumetric representation such as a signed distance function [Newcombe et al., 2011a] or occupancy map [Hornung et al., 2013]. The initial design of dense mapping typically use a regularly spaced 3D voxel grid and thus the memory consumption scales with the size of the represented volume rather than the surface, limiting its ability to perform large-scale mapping. Some following work proposed to use more efficient data structures, such as N^3 trees [Chen et al., 2013], Octrees [Vespa et al., 2018], and voxel hashing [Nießner et al., 2013]. Dense mapping can also be generated in the mesh representation directly [Rosinol et al., 2019] or via augmenting pointclouds with knowledge of the surface orientation [Whelan et al., 2016].

Recently, learning-based representations start to gain popularity in the research community. Recent works, such as GQN [Eslami et al., 2018] and CodeSLAM [Bloesch et al., 2018] use keyframe-based view-based latent code representations from a variational auto-encoder (VAE) for scene representations. In addition to the view-dependent approaches, several methods directly learn 3D-aware neural representations by augmenting the classic volumetric representations with 3D learning features [Sitzmann et al., 2019, Park et al., 2020]. They have tried to learn a 3D-structured latent model , but memory cost of 3D convolutions on the explicit 3D grid resolution often limit its representation capability. Instead of augmenting classical representation with deep features, direct parametrizing 3D scenes implicitly using network weights has also gained much attention very recently. Neural implicit representations can implicitly represent 3D scenes using the weights of neural networks (often MLPs) which can predict the 3D geometry in occupancy or SDF values at any given 3D query position, yielding unlimited resolutions. It has been applied in both view-conditioned ones, including Neural Radiance Fields (NeRF) [Mildenhall et al., 2020], and 3D-aware representations, including DeepSDF [Park et al., 2019] and Occupancy Networks [Mescheder et al., 2019].

In Chapter 3 we will show how an efficient octree-based TSDF volume representation is used to reconstruct moving rigid objects. In Chapter 5, we will show how to use learning-based representations to utilize category-level shape prior and to predict a complete shape geometry of a moving object.

2.7 Deep Neural Networks

Deep neural networks (DNN) have led to significant performance improvements in almost every computer vision task, including semantic segmentation, depth prediction, 3D reconstruction, and scene understanding [He et al., 2016, He et al., 2017, Eigen et al., 2014, Ummenhofer et al., 2016]. It has also been applied in many robotic applications and combined well with traditional model-based methods. A deep neural network is typically a parameteric computational graph that is composed of biologically-inspired "neurons". Each neuron works as a weighted summation of inputs that is followed by a nonlinear activation function to produce an output for the next neuron layer. The purpose of a DNN training is to learn an approximation of an unknown mapping $f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$ by updating neuron weights θ through back-propagation to minimise a designed loss function, which is often supervised over $(\mathbf{x} \in \mathcal{X}, \mathbf{y} \in \mathcal{Y})$ pairs in the training set. It is often not computationally feasible to do the minimisation overall training pairs in one pass, and the backpropagation training is conducted instead over mini-batches of training data. As the mini-batches are randomly drawn from the full training set and reshuffled in each epoch, the optimisation procedure resembles stochastic gradient descent that may help avoid local minima [Goodfellow et al., 2016]. This is repeated until convergence on the training data and periodically monitored on the designed loss on a reserved validation to avoid overfitting. After the training finishes, it is expected to work well on unseen testing data sampled from a similar data distribution as the training data.

Multi Layer Perceptron

One of the simplest neural networks, called multilayer perceptrons (MLP) or fully-connected networks, consists of solely fully connected (FC) layers and non-linear activation functions. In each layer, a neuron is densely connected to all neurons in the previous and next adjacent layer. There are no connections between neurons within the same layer.

The representation power of an MLP is that theoretically, it is a universal function approximators [Hornik, 1989] and thus can approximate solutions for extremely complex high-dimensional nonlinear problems in machine learning. Unlike convolutional neural networks (CNN) or recurrent neural networks (RNN), it also has little inductive bias due to its fully connected design. Therefore, the unconstrained MLP tends to overfit training data and does not have translation invariance and thus making it less efficient at some 2D and 3D vision tasks. Despite this, MLP has been used in a variety of popular network architectures [Simonyan and Zisserman, 2015, Qi et al., 2017a]. In recent years, MLPs have also been back to the research communities' attention and have been applied in implicit representation for 3D shape [Park et al., 2019], 3D scene representation [Sitzmann et al., 2020], and 3D scene view synthesis [Mildenhall et al., 2020] as well as vision transformers [Vaswani et al., 2017] due to its strong representation power.

In Chapter 5, we will show how to use a coordinate-based MLP to learn implicit map representations for category-level shape prior and conditioned shape completion.

Convolutional Neural Networks

Compared with MLPs, Convolutional Neural Networks (CNN) incorporate stronger inductive biases and can learn translation-invariant features that are suitable for many 2D and 3D vision tasks. CNN typically assumes a regular grid pattern in the input data and is composed of a sequence of layers that usually consist of convolutional layers, pooling layers and optional fully connected layers. The weights to be optimised in the training time are in convolutional kernels and fully connected layers.

Each convolutional layer learns small filter kernels that are convolved against the entire input. This enables parameter sharing as the salient features detected at a certain position can also be detectable at other spatial positions, enabling translation invariance in CNN. It also helps control the number of trainable parameters in CNN, allowing more efficient computation and deeper network design. Another property brought by the convolutional layer is the designed assumption of local connectivity. The convolution operation means the spatial extent of the connectivity of each neuron, i.e. receptive field, is limited to a local region. It helps solve the scaling issue of network size with the input dimension that exists in MLP. By carefully designing the receptive field of the convolutional layer, it is possible to replace large filter kernels with several small filters, enabling a more compact network size. Modern *deep* neural networks often have many convolutional layers, with deeper layers learning more abstract and higher-level features from the previous layers. Pooling layers are periodically inserted between convolutional layers to downsample the output feature maps. This helps reduce the dimensions of the input to the following convolutional layers and thus reduce the trainable parameter and the computation amount. This can also mitigate the overfitting issue. Average pooling and maximum pooling are the two most commonly used pooling types. Fully connected layers are typically placed at the end of CNN for feature fusion and classification.

In Chapter 4, we will show how to use Convolutional Neural Networks to learn dense features for robust camera and object tracking in a self-supervised way.

Chapter 3

Octree-based Object-Level Multi-Instance Dynamic SLAM

In this chapter, we propose a new multi-instance dynamic RGB-D SLAM system using an object-level octree-based volumetric representation. It can provide robust camera tracking in dynamic environments and at the same time, continuously estimate geometric, semantic, and motion properties for arbitrary objects in the scene. For each incoming frame, we perform instance segmentation to detect objects and refine mask boundaries using geometric and motion information. Meanwhile, we estimate the pose of each existing moving object using an object-centric tracking method and robustly track the camera pose against the static scene. Based on the estimated camera pose and object poses, we associate segmented masks with existing models and incrementally fuse corresponding colour, depth, semantic, and foreground object probabilities into each object model. In contrast to existing approaches, our system is the first system to generate an object-level dynamic volumetric map from a single RGB-D camera. Our method can run at 2-3 Hz on a CPU, excluding the instance segmentation part. We demonstrate its effectiveness by quantitatively and qualitatively testing it on both synthetic and real-world sequences.

Contents of Chapter

Parts of this Chapter appear in:

Binbin Xu, Wenbin Li, Dimos Tzoumanikas, Michael Bloesch, Andrew Davison, Stefan Leutenegger (2019). MID-Fusion: Octree-based Object-Level Multi-Instance Dynamic SLAM. *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2019.* [Xu et al., 2019]

3.1 Introduction

In Simultaneous Localisation and Mapping (SLAM) both, the map of the unknown environment as well as the robot pose within it, are concurrently estimated from the data of its on-board sensors only. In recent years, the field of SLAM has experienced rapid progress. It started from sparse SLAM [Davison et al., 2007, Klein and Murray, 2007], and evolved into dense SLAM [Newcombe et al., 2011a] thanks to the increased computational power of GPU and affordability of depth sensors. More recently, many people have begun to leverage Deep Neural Networks and their ability to learn from large amounts of training data to improve SLAM. This fast-evolving research in SLAM has, since then, lead to strong progress in various fields of applications, such as robotics, Virtual Reality (VR), and Augmented Reality (AR).

Despite this progress, much work is still based on the fundamental assumption of a static environment, within which points in the 3D world always maintain the same spatial position in the global world, with the only moving object being the camera. This assumption enabled the success of early phases of development as it alleviated the chicken-and-egg problem between map estimation and sensor pose estimation. A camera pose can be estimated between a live frame and a reference frame, which is based on the assumption that the relative transformation between those two images is caused only by the camera motion. It is this basic, yet strong, assumption that allowed a joint probabilistic inference (sparse SLAM [Durrant-Whyte and Bailey, 2006]) or an alternating optimisation (dense SLAM [Engel et al., 2017]) of map and pose relationship to solve SLAM. Any moving objects in the environment would be treated as outliers to the static model and are intentionally ignored by tracking and mapping.

This idealised setup, therefore, can only handle a small amount of dynamic elements and disqualifies itself from many real-world applications as environments,

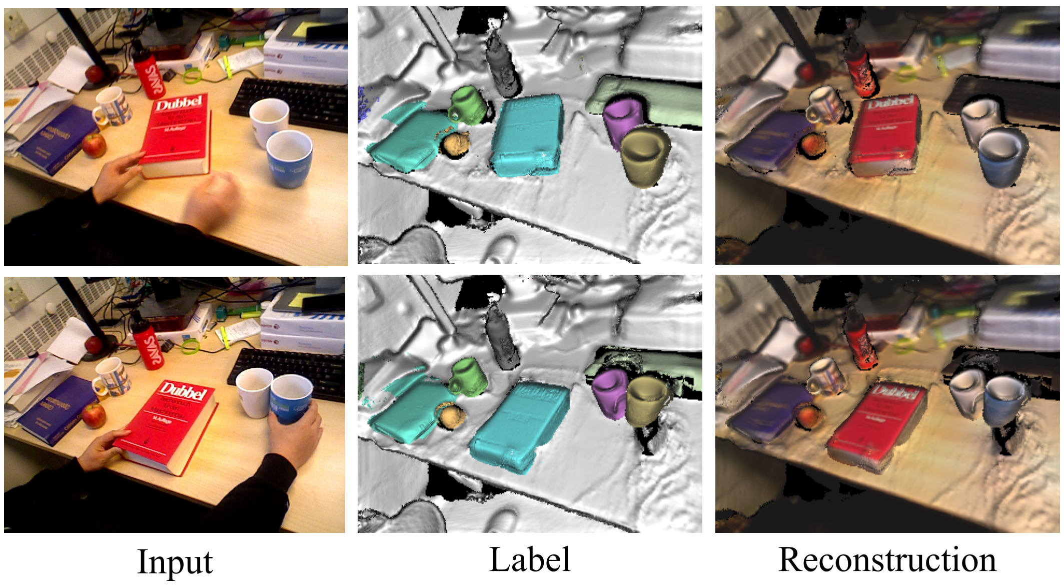


Figure 3.1: An overview of our system. Given RGB-D images, our system builds an object-level dense volumetric map that deals with dynamic objects and ignores people. Next to the input image we show the labelled object models as well as the coloured reconstruction.

especially where humans are present, change constantly. A robust SLAM system, which works in highly dynamic environments, is still an open problem, which we seek to address in this work.

Although dynamic SLAM has been studied for a couple of decades [Wang et al., 2003], approaches based on visual dense SLAM have only recently been explored. They can be categorised into three main directions. One deforms the whole world in a non-rigid manner in order to include a deformable/moving object [Newcombe et al., 2015]. The second specifically aims at building a single static background model, while ignoring all possibly moving objects and thus improving the accuracy of camera tracking [Jaimez et al., 2017, Scona et al., 2018, Barnes et al., 2018, Bescós et al., 2018]. The third models dynamic components by creating sub-maps for every possibly rigidly moving object in the scene while fusing corresponding information into these sub-maps [Rünz and Agapito, 2017, Bârsan et al., 2018, Rünz and Agapito, 2018]. We are more interested in the third direction since we believe that, similar to human perception, an awareness of instances in the map would be a more proper solution for robots to perceive the changing environment and has higher potential to achieve a meaningful map representation. However, most existing approaches build maps using a collection of surfels, which is difficult to be used directly for robotic tasks. The only two systems that support sub-map volumetric map, we know of so far, are [Bârsan et al., 2018] and [McCormac et al., 2018]. However, the former has been specifically designed for an outdoor stereo camera setting and the latter only deals with static environments. Here, we propose the first object-level dynamic volumetric map for indoor environment applications, where free space and surface connectivity can be represented for each object model. We further improve its memory efficiency by utilising an octree-based structure. Despite showing some promising results based on deep learning, most methods [Rünz and Agapito, 2017, Rünz and Agapito, 2018, Bârsan et al., 2018] simply leverage predictions from neural network without much refinement in the map fusion. In this chapter, we integrate and refine semantic predictions by fusing them into object models.

The main contributions in this chapter are divided into four main parts. We propose

1. the first RGB-D multi-instance dynamic SLAM system using a volumetric representation,
2. a more robust tracking method utilising weighting via measurement uncertainty and being re-parametrised for object tracking,
3. an integrated segmentation using geometric, photometric, and semantic information,
4. a probabilistic fusion of semantic distribution and a foreground object probability into octree-based object models.

3.2 Related Works

In the majority of SLAM systems the environment is assumed to be static. To tackle dynamic environment in real-world applications, several solutions have recently been proposed and they can be mainly categorised into three directions as introduced in last section. We will introduce and compare the last two types of approaches in further details in this section. One straightforward way for dynamic SLAM is to segment dynamic objects out as outliers and intentionally ignore them from tracking and reconstruction to avoid corruption in the pose estimation. StaticFusion [Scona et al., 2018] performs segmentation by coupling camera motion residuals, depth inconsistency and a regularisation term. Barnes *et al.* [Barnes et al., 2018] learn to segment possibly moving objects in a self-supervised way, which is limited by the availability of training data and may often misclassify static objects. Bescos *et al.* [Bescós et al., 2018] combine Mask-RCNN [He et al., 2017] with depth inconsistency checking to segment moving objects and further inpaint those areas with static background. Those methods provide a more robust approach in dynamic scene than conventional SLAM methods, however, information regarding the moving objects is lost. Instead, our approach aims to simultaneously track and reconstruct static background and dynamic and static objects in the scene, while at the same time, provide state-of-the-art tracking accuracy.

There are three approaches, to our knowledge, which provide similar functionality as ours and can reconstruct multiple moving objects in the scene – the third way to tackle dynamic SLAM. Co-Fusion [Rünz and Agapito, 2017] segments objects by either ICP motion segmentation or semantic segmentation and then tracks objects separately based on ElasticFusion [Whelan et al., 2016]. MaskFusion [Rünz and Agapito, 2018] segments objects using a combination of instance segmentation from Mask-RCNN and geometric edges, and tracks objects using the same approach as Co-Fusion. Both Co-Fusion and MaskFusion use surfels to represent map models, which is also memory efficient but cannot directly provide free space information in the map, and neither surface connectivity. DynSLAM [Bârsan et al., 2018] focuses on outdoor environments using stereo cameras. In contrast, our system focuses on indoor environments consisting of many (potentially) moving objects using a single RGB-D camera.

In terms of differences in system components, our system further differentiates itself from above approaches. In camera tracking, we weighted photometric and geometric terms by their measurement uncertainty, instead of a single weight such as in [Whelan et al., 2016]. Also, to be robust to depth loss, we derive two terms from different frames to complement one another. To track objects, all previous methods use a virtual camera pose, which is not robust to object rotation due to a large lever arm effect in its optimisation derivative. We found best robustness by re-parametrising it into object coordinate. To generate object masks, we combine both information to provide better boundary conditions, instead of using just motion or just semantic information. When fusing information to object models, we fuse not only depth and colour information, but also semantic and foreground predictions while previous methods just take predictions from neural network without any refinement. In terms of speed, all three of the above require one or even two powerful GPUs, while our method, despite running only on CPU, is capable of performing at a similar speed to DynSLAM [Bârsan et al., 2018], thanks to our efficient octree-based data structure for object models.

Another very recent work related to ours is Fusion++[McCormac et al., 2018], which generates an object-level volumetric map yet in static environments. In addition to handling dynamic scenes, our system utilises a joint photometric and geometric tracking to robustly track both camera and object poses while Fusion++ only use geometric tracking to estimate camera pose. Furthermore, to have a better object mask boundary for fusion and tracking, we combine geometric, motion and existing model information to refine mask boundary instead of directly using predicted mask as was done in Fusion++. In terms of map representation, Fusion++ is based on discrete voxel grids, which suffers from scalability issues, while we represent all our object models in memory-efficient octree structures.

3.3 Notations and Preliminaries

In this chapter, we will use the following notation: a reference coordinate frame is denoted $\underrightarrow{\mathcal{F}}_A$. The homogeneous transformation from $\underrightarrow{\mathcal{F}}_B$ to $\underrightarrow{\mathcal{F}}_A$ is denoted as $\boldsymbol{T}_{AB}$, which is composed of a rotation matrix $\mathbf{C}_{AB}$ and a translation vector ${}_A\mathbf{r}_{AB}$. For each pair of images, we distinguish them as live (L) and reference (R) image. For example, a live RGB-D image contains the intensity image I_L and depth image D_L, with 2D pixel positions denoted as $\boldsymbol{u}_L$ and pixel lookup (including bilinear interpolation) denoted as $[\cdot]$. Perspective projection and back-projection are denoted π and π^{-1}, respectively.

In our system, we store every detected object into a separate object coordinate frame $\underrightarrow{\mathcal{F}}_{\mathbf{O}_n}$, with $n \in \{0 \ldots, N\}$ where N is the total number of objects (excluding background) and 0 denotes background. We assume a canonical static volumetric model is stored in each object coordinate frame, forming the basis of our multi-instance SLAM system. In addition, each object is also associated with a COCO dataset [Lin et al., 2014] semantic class label $c_n \in \{0, \ldots, 80\}$, a probability distribution over its potential semantic class labels, a current pose w.r.t. the world coordinate $\boldsymbol{T}_{WO_n}$, and a binary label $s \in \{0, 1\}$ denoting whether the object is believed to be in motion or not. Each object is represented in an separate octree structure, where every voxel stores Signed Distance Function (SDF) value, intensity, foreground probability and the corresponding weights.

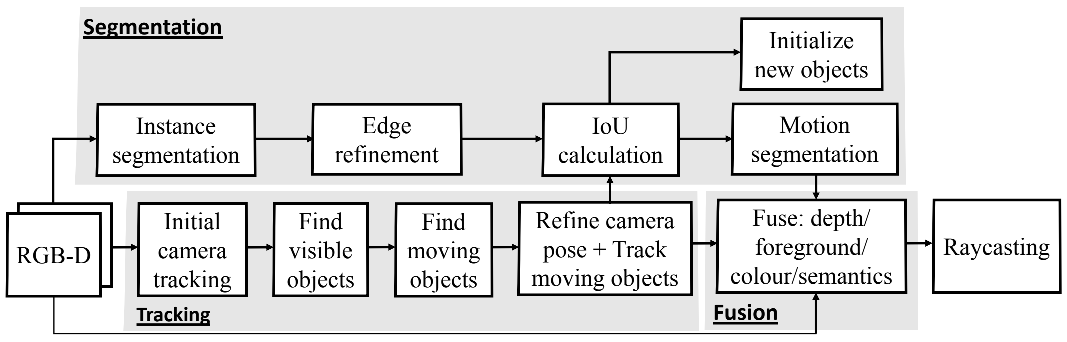


Figure 3.2: The pipeline of the proposed method

3.4 Method

3.4.1 System Overview

Figure 3.2 shows the pipeline of our proposed system. It is composed of four parts: segmentation, tracking, fusion and raycasting. Each input RGB-D image is processed by Mask R-CNN to perform instance segmentation, which is followed by geometric edge segmentation and motion residuals from tracking to refine mask boundaries (Section 3.4.4). For the tracking, we first track the camera against all vertices excluding the human mask area (Section 3.4.2) and then raycast from this pose to find which objects are currently visible in this frame. This can also help create a preliminary association between local object masks with existing object models. We evaluate motion residuals for each object to determine if it is in motion or not, then track moving objects (Section 3.4.3) and refine the camera pose against the static world – which includes currently static objects (Section 3.4.2). Using estimated poses of the camera and objects, we refine the data association between object detections and object models and then fuse the corresponding depth and colour information, as well as predicted semantic and foreground probabilities into the object models (Section 3.4.5). Detection of visible objects as well as raycasting is explained in Section 3.4.6.

3.4.2 RGB-D Camera Tracking

This part estimates the live camera pose $\boldsymbol{T}_{WC_L}$ and is composed of two steps. First, it tracks against all model vertices while masking out detected people; second, it tracks against all static scene parts. Both steps are conducted by minimising the dense point-to-plane ICP residual e_g and photometric (RGB) residual e_p, which are weighted by individual measurement uncertainty, w_g and w_p.

$$E_{\text{track}}(\boldsymbol{T}_{WC_L}) = \frac{1}{2}\left(\sum_{\boldsymbol{u}_L \in M_L} w_\mathrm{g}\,\rho(e_\mathrm{g}) + \sum_{\boldsymbol{u}_R \in M_R} w_\mathrm{p}\,\rho(e_\mathrm{p})\right), \tag{3.1}$$

where ρ represents the Cauchy loss function and M is a mask excluding invalid correspondences (for ICP), occlusions (for RGB), and humans.

For the ICP residual, we use the method proposed in [Newcombe et al., 2011a] to minimise point-plane depth error between the live depth map and the rendered depth map of the model on the reference frame:

$$e_\mathrm{g}(\boldsymbol{T}_{WC_L}) = {}_W\mathbf{n}^r[\boldsymbol{u}_R] \cdot (\boldsymbol{T}_{WC_L}\,{}_{C_L}\mathbf{v}[\boldsymbol{u}_L] - {}_W\mathbf{v}^r[\boldsymbol{u}_R]), \tag{3.2}$$

where ${}_{C_L}\mathbf{v}$ is live vertex map in the camera coordinate by back-projection and ${}_W\mathbf{v}^r$ and ${}_W\mathbf{n}^r$ are the rendered vertex map and normal map expressed in world coordinates. For each pixel $\boldsymbol{u}_L$ on the live depth map, its correspondence $\boldsymbol{u}_R$ on the rendered depth map can be found using projective data association:

$$\boldsymbol{u}_R = \pi(\boldsymbol{T}_{WC_R}^{-1}\boldsymbol{T}_{WC_L}(\pi^{-1}(\boldsymbol{u}_L, D_L[\boldsymbol{u}_L]))), \tag{3.3}$$

where $\boldsymbol{T}_{WC_R}$ is the camera pose of the reference frame.

For maximum robustness, we combine the ICP residual with a photometric one by rendering a depth map from model in the reference frame and using that depth map to align photometric consistency:

$$e_\mathrm{p}(\boldsymbol{T}_{WC_L}) = I_R[\boldsymbol{u}_R] - I_L[\pi(\boldsymbol{T}_{WC_L}^{-1}(\boldsymbol{T}_{WC_R}\,\pi^{-1}(\boldsymbol{u}_R, D_R^r[\boldsymbol{u}_R])))]. \tag{3.4}$$

Different from previous approaches [Rünz and Agapito, 2017], we evaluate the photometric residuals using rendered reference depth map other than the raw depth map on live frame or reference frame for the de-noised depth quality from models. This choice furthermore improves the robustness of tracking when raw input depth is not available, e.g. when the camera is too close to a surface.

We further introduce a measurement uncertainty weight to combine ICP and RGB residuals. For RGB residuals, the measurement uncertainty is assumed to be constant for all pixels. For ICP residuals, the quality of input depth map is related to the structure of the depth sensor and the depth range. We adopted the inverse covariance definition for depth measurement uncertainty in [Laidlow et al., 2017]. Given the sensor parameters, i.e. baseline b, disparity d, focal length f, and the uncertainties in the x-y plane σ_{xy} and disparity direction σ_z, the standard deviation σ_D for depth sensor measurement in the x, y, z coordinates can be modelled as:

$$\sigma_D = (\frac{D_L[\boldsymbol{u}_L]}{f}\sigma_{xy}, \frac{D_L[\boldsymbol{u}_L]}{f}\sigma_{xy}, \frac{D_L^2[\boldsymbol{u}_L]}{fb}\sigma_z). \tag{3.5}$$

The weight for ICP residuals using the inverse covariance of measurement uncertainty is then defined as:

$$w_g = \frac{1}{({}_W\mathbf{n}^r)^T \, {}_W\mathbf{n}^r \sigma_D^T \sigma_D}. \tag{3.6}$$

The cost function is minimised using the Gauss-Newton approach in a three-level coarse-to-fine scheme. The necessary Jacobians can be referred in Chapter 2.

After performing an initial camera tracking, we raycast to find visible objects in the view. To find which objects are in motion, we evaluate $E_{\text{track}}(\boldsymbol{T}_{WC_L})$ once again on the finest level on the *live* frame. To this end, the RGB residual needs to be re-formulated in the live frame as:

$$e_{\text{p}}(\boldsymbol{T}_{WC_L}) = I_L[\boldsymbol{u}_L] - I_R[\pi(\boldsymbol{T}_{WC_R}^{-1}(\boldsymbol{T}_{WC_L}\pi^{-1}(\boldsymbol{u}_L, D_L^r[\boldsymbol{u}_L])))]. \tag{3.7}$$

To detect textureless objects in motion, we also include the geometric loss (Equation (3.2)), similar to the joint residual defined in Equation (3.1). We apply a threshold to the combined residual $E_{\text{track}}(\boldsymbol{T}_{WC_L})$ to find the motion inliers. If the inlier ratio is lower than 0.9 in the object's rendered mask, then we consider that object is moving and refine its pose as described in Section 3.4.3. The camera pose is then refined by tracking against only static objects using the same objective function and optimisation strategy explained above. Note that this hard threshold for inlier ratio is a hyperparameter that can be tuned for different objects, but we consistently use this number in all our experiments. Since it is a threshold applied to the joint photometric and geometric residuals, it may miss objects that do not contain enough information in both texture and geometric structures.

3.4.3 Object Pose Estimation

In this part, we describe how to estimate the pose of moving objects. As opposed to virtual camera based tracking [Rünz and Agapito, 2017, Barnes et al., 2018], we propose to employ an object-centric approach, which is less prone to bad initial pose guesses. We still use a joint dense ICP and RGB tracking, weighted in the same way as Equation (3.1), just with different ICP and RGB residual definitions. In the present formulation, we estimate the current relative pose between object and camera, $\boldsymbol{T}_{C_L O_L}$, by aligning the live vertex map expressed in the live object frame with the rendered vertex map expressed in the reference object frame:

$$e_{\text{g}}(\boldsymbol{T}_{C_L O_L}) = \mathbf{C}^{-1}_{WO_R}\, {}_{W}\mathbf{n}^{r}[\boldsymbol{u}_R] \cdot (\boldsymbol{T}^{-1}_{C_L O_L}\, {}_{C_L}\mathbf{v}[\boldsymbol{u}_L] - \boldsymbol{T}^{-1}_{WO_R}\, {}_{W}\mathbf{v}^{r}[\boldsymbol{u}_R]). \tag{3.8}$$

The formulation is based on the assumption that each object coordinate frame yields a static canonical object model and thus the point clouds must align. The proposed parameterisation leads to more stable tracking due to a smaller lever arm effect of the rotation. When computing the partial derivative of the above cost w.r.t. the rotation [Bloesch et al., 2016] we get a term proportional to $\mathbf{C}^{-1}_{C_L O_L}({}_{C_L}\mathbf{v}[\boldsymbol{u}_L] - {}_{C_L}\mathbf{r}_{C_L O_L})$ which is small since we choose the object frame to be centred. In analogy, we also re-formulate the RGB residual as:

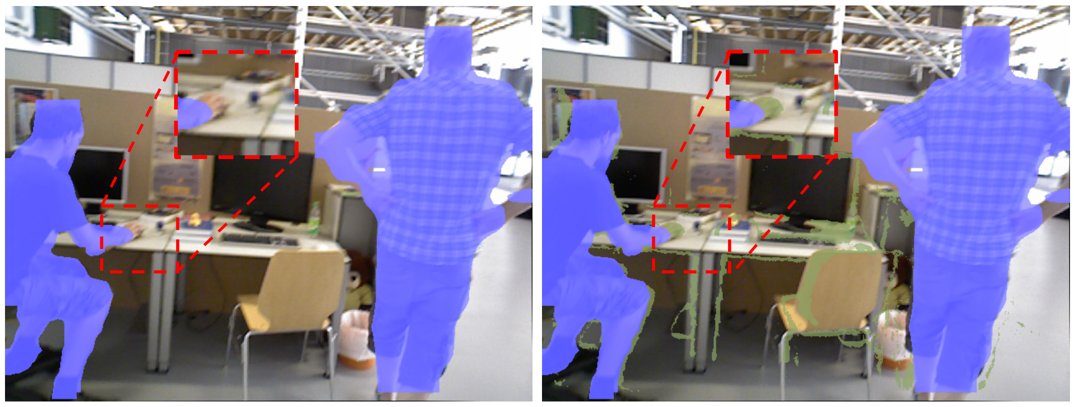
$$e_{\mathrm{p}}(\boldsymbol{T}_{C_L O_L}) = I_R[\boldsymbol{u}_R] - I_L[\pi(\boldsymbol{T}_{C_L O_L}\boldsymbol{T}_{C_R O_R}^{-1}(\pi^{-1}(\boldsymbol{u}_R, D_R^r[\boldsymbol{u}_R])))]. \tag{3.9}$$

The above cost function is also optimised using Gauss-Newton approach in a three-level coarse-to-fine scheme with $\boldsymbol{T}_{C_L O_L}$ initialised as $\boldsymbol{T}_{C_L O_R}$.

3.4.4 Combined Semantic-Geometric-Motion Segmentation

For each RGB-D frame, we use Mask R-CNN [He et al., 2017] to find semantic instances, followed by geometric edge refinement to solve leaked mask boundaries [Rünz and Agapito, 2018]. Then we render instance masks for each map object to the live frame by means of raycasting (explained in Section 3.4.6). We associate local segmentation masks, which are generated from Mask R-CNN and geometric refinement, with existing object models by calculating the intersection of union (IoU) with the rendered masks. We assign the local segmentation mask to the rendered mask which has the largest intersection and where the intersection is larger than 0.5. In comparison to [Rünz and Agapito, 2018], we do not require predicted semantic label of the local segmentation mask to be the same as object semantic class since the prediction may be subject to high uncertainty. Instead, we trust probabilistic fusion of semantic predictions to refine the objects' semantic labels (described in Section 3.4.5).

For segmentation masks that do not belong to any existing objects, a new object model will be initialised (described in Section 3.4.5). For objects without associated local segmentation masks, i.e. Mask R-CNN has no corresponding detection, we choose its rendered mask from the model for the subsequent fusion process. Since the rendered masks are associated with the object models, we do not integrate the foreground probability and semantic predictions for these undetected objects in this frame.



(a) Instance segmentation with geometric refinement (hand is missed out)

(b) Instance segmentation with geometric and motion refinements

Figure 3.3: Combination of semantic, geometric and motion segmentations. Mask regions that refined by geometric segmentation is in blue and the one further refined by motion residual is shown in green.

After associating segmentation masks with object models, we further refine the segmentation masks based on motion residuals of object tracking. We evaluate Equation (3.1) again on the finest level, however, this time we evaluate photometric residual on the live frame:

$$e_{\mathrm{p}}(\boldsymbol{T}_{WC_L}) = I_L[\boldsymbol{u}_L] - I_R[\pi(\boldsymbol{T}_{C_R O_R}\,\boldsymbol{T}^{-1}_{C_L O_L}\,(\pi^{-1}(\boldsymbol{u}_L, D^r_L[\boldsymbol{u}_L]))))]. \tag{3.10}$$

Pixels whose joint ICP and RGB residuals are too high are treated as outliers and filtered out in the segmentation mask. This can help detect the moving objects when the object segmentation component misses the detection of some moving parts, as one example shows in Figure 3.3.

Before integration, we also generate a foreground mask based on the local segmentation mask. The use of foreground probabilities is inspired by the foreground/background probabilities introduced in [McCormac et al., 2018] and allows to avoid spurious integration due to wrong segmentation masks. Information in both foreground and background regions are integrated into the models. In order to avoid impairing the efficiency of the octree structure, we use dilated segmentation masks as background mask. Pixels in the foreground are assigned an foreground probability of 1.0 while pixels in the dilated background are assigned 0. For undetected existing objects that Mask R-CNN fails on, we assign an foreground probability of 0.5 to their foreground due to their lower possibility of existence.

3.4.5 Object-level Fusion

From each frame, we integrate depth, colour, semantics and foreground probability information into object models using foreground and background masks. Using the relative pose $\boldsymbol{T}_{O_n C_L}$ and depth, the Truncated SDF (TSDF) is updated following the approach of Vespa et al. [Vespa et al., 2018]. Concurrently within the same voxels, colour and foreground probability are updated using a weighted average. For semantic fusion, we refine the semantic class probability distribution for each model using averaging, instead of Bayesian updating which often leads to overconfidence when used with Mask R-CNN predictions[McCormac et al., 2018].

For every segmentation mask that cannot be associated with any existing objects, we initialise a new object model whose coordinate frame is centred around the object itself. We back-project all points in the mask into world coordinates and then find the centre and size of these point clouds. To account for possible occlusions, we initialise the TSDF volume size to be 3 times the point cloud size to avoid additional padding. We choose the volume resolution such that each voxel size is slightly bigger than 1mm in order to support detailed object reconstruction. With the octree-based structure, the unused voxels will not be initialised and the whole system remains memory-efficient. The initial object translation in $\boldsymbol{T}_{W O_n}$ is chosen as the left side corner of the object volume and the orientation is aligned with world coordinates.

3.4.6 Raycasting

We perform raycasting from camera pose to object models to render depth, normal, intensity, and instance label maps. We use a similar method as proposed in [McCormac et al., 2018]. However, as shown in the pipeline Figure 3.2, our system involves at least four raycasting operations: depth rendering in tracking, finding visible objects, IoU calculation, and visualisation. The first two operations create a preliminary data association based on the estimated object pose in the last frame. After camera and object motion refinement, the last two operations can refine the data associations for dense mapping. Besides, it would be computationally expensive if we continuously raycast all objects on each stage. To speed up, we raycast all objects only once to find visible objects from the existing object models, and avoid raycasting to invisible objects in the remaining steps of this frame. This is based on the assumption that objects' poses between consecutive frames do not change dramatically.

For each ray originating from camera center ${}_W\mathbf{c}_L$ in the direction of ${}_C\mathbf{r}$, we warp both the origin and the direction into the object model coordinate. For each object $\mathbf{O}_m$, there will be a ray originating from $\boldsymbol{T}_{WO_m}^{-1}\,{}_W\mathbf{c}_L$ in the direction of $\mathbf{C}_{CO_m}^{-1}\,{}_C\mathbf{r}$. Within each object octree volume, we apply the field interpolation method [Vespa et al., 2018] to look for voxel blocks. We store the ray length of the nearest intersection by far to avoid searching past that point in another object volume. Since the ray is originated from the same point, even in a different object coordinate, the ray length magnitude can be directly used to compare raycasting intersection length. We use the closest intersected voxel for raycasting result and only raycast voxels whose foreground probability is higher than 0.5. For one voxel that is generated in both background and object volumes, we give a priority to voxel in the object volumes. If two object voxels have same intersection length, we prefer the one with higher existence probability. Objects whose voxels are not raycasted at all are considered to be invisible from this view.

3.5 Experiments

We evaluate our system on a Linux system with an Intel Core i7-7700 CPU at 3.50GHz with 32GB memory. Mask R-CNN segmentation is pre-computed on the GPU using the publicly available weights and implementation [Wu et al., 2016] without fine-tuning. Each object is stored in a separate octree-based volumetric model, modified based on source code of Supereight [Vespa et al., 2018].

3.5.1 Robust Camera Pose Estimation

We first evaluate the camera tracking accuracy in dynamic environments using the widely used TUM RGB-D dataset [Sturm et al., 2012]. The dataset provides RGB-D sequences with ground truth camera trajectory, recorded by a motion capture system. We report the commonly used Root-Mean-Square-Error (RMSE) of the Absolute Trajectory Error (ATE). To evaluate the effect of different cameras motion and environment change conditions, 6 different sequences are investigated. In the f3s sequences, two people were sitting in the desk while engaging in slightly dynamic movements. In the f3w sequences, two people were engaging in highly dynamic movements. For both types of sequences, three different camera movements were involved: static with the camera kept static manually, xyz with the camera moving along the x-y-z axes, and halfsphere with the camera moving following the trajectory of a 1m diameter half sphere.

We compare our method with five state-of-the-art dynamic SLAM approaches: joint visual odometry and scene flow (VO-SF) [Jaimez et al., 2017], StaticFusion (SF) [Scona et al., 2018], DynaSLAM (DS) [Bescós et al., 2018], Co-Fusion (CF) [Rünz and Agapito, 2017], and MaskFusion (MF) [Rünz and Agapito, 2018]. VO-SF [Jaimez et al., 2017], SF [Scona et al., 2018], and DS [Bescós et al., 2018] were designed for reconstructing the static background with dynamic parts ignored (or even inpainted as in DS [Bescós et al., 2018]). CF [Rünz and Agapito, 2017] and MF [Rünz and

Agapito, 2018] were designed for multi-object reconstruction. In all these methods, DS [Bescós et al., 2018] is the only method using feature-based sparse tracking (not reconstructing moving objects at all), while the remaining ones use dense tracking methods as ours. For fair comparison, we compare first with dense tracking methods and take DS [Bescós et al., 2018] as an additional reference. Table 3.1 reports our experimental results.

From the Table 3.1, we can see that our system achieves best results in almost all sequences among dense tracking method. Our method even outperforms VO-SF and SF, which were designed especially for robust camera tracking in a dynamic environment. Figure 3.4 shows two inputs and the reconstruction results in the challenging "f3w halfsphere" sequence. We highlight rejected segmentation masks in the input images, with the geometrically refined mask labelled as human in blue and the high residual regions during motion refinement in green. It can be noted in Figure 3.4 that even when Mask R-CNN fails to recognise a person, our combined segmentation using geometric and motion refinement can still reject it. This, combined with the high quality depth rendered from our octree-based TSDF model, leads to our robust and accurate camera tracking estimation in these highly dynamic scenes. DynaSLAM achieves best tracking accuracy in almost all sequences while it is the only sparse feature-based SLAM system among the tested approaches. It shows the potential advantage of feature trackers in dynamic environments. As part of future work, it would be very interesting to combine a feature-based approach with direct dense tracking/mapping methods to further improve camera tracking accuracy and robustness. This could also help to overcome current failure-cases in challenging conditions such as very reflective scenes or fast motions.

Table 3.1: Quantitative comparison of camera tracking

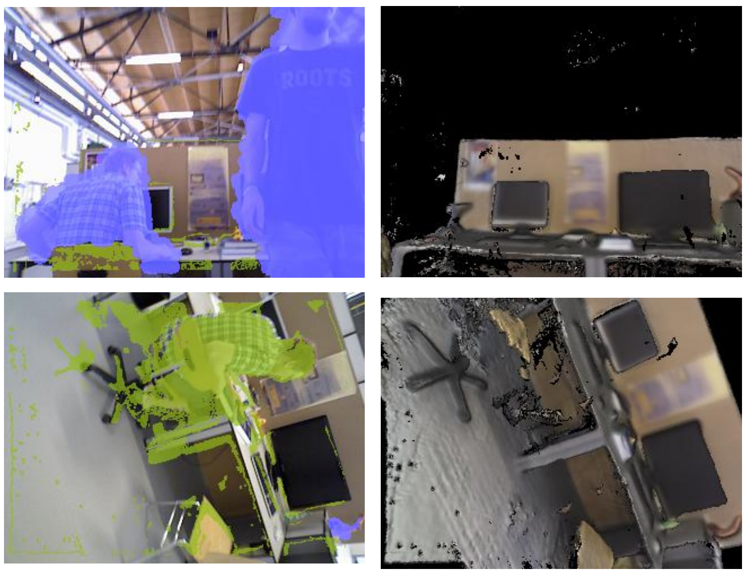
Sequence	ATE RMSE (cm)					
	VO-SF	SF	CF	MF	**Ours**	DS*
f3s static	2.9	1.3	1.1	2.1	**1.0**	-
f3s xyz	11.1	4.0	**2.7**	3.1	6.2	**1.5**
f3s halfsphere	18.0	4.0	3.6	5.2	**3.1**	**1.7**
f3w static	32.7	**1.4**	55.1	3.5	2.3	**0.6**
f3w xyz	87.4	12.7	69.6	10.4	**6.8**	**1.5**
f3w halfsphere	73.9	39.1	80.3	10.6	**3.8**	**2.5**

*: feature-based sparse approach. The others are dense-tracking approaches.

3.5.2 Object Reconstruction Evaluation for Other Components

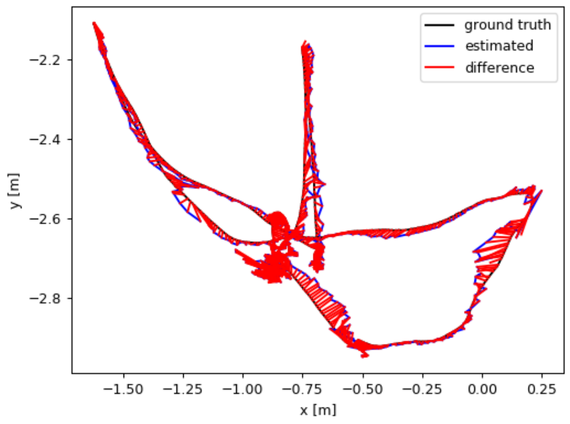
We also tested our method within a fully controlled synthetic environment using photo-realistic rendering and trajectory simulation [Li et al., 2018b]. We selected a typical indoor scene with a sofa and a chair being translated and rotated in front of the camera. We implicitly evaluate object pose estimation accuracy via object reconstruction error.

To evaluate the effect of segmentation, we replaced the segmentation pipeline with ground truth masks (G.T. Seg.). We also compared our object-oriented tracker with virtual camera (V.C.) tracking to see if our parametrisation improves tracking accuracy. We further compare with Co-Fusion(CF) [Rünz and Agapito, 2017] using their public code. Table 3.2 reports the mean and standard deviation of reconstruction error in these experiments. The results show that our system can achieve more accurate object reconstructions. The difference between using ground truth masks and our own segmentation component is negligible in the specific example. The higher error obtained using virtual camera tracking demonstrates the reliability of our object-centric tracking, especially for large object rotations. Figure 3.5 shows the visualisation comparison results on the sofa reconstruction.



Input with estimated masks Reconstruction

(a) Robust detection of dynamic objects and reconstruction of static background



(b) Ground truth (blue) with estimated (red) trajectories

Figure 3.4: Robust camera tracking and background reconstruction in a dynamic environment (in “f3w halfsphere” sequence). Moving persons are rejected due to the semantic labelling of Mask R-CNN (in blue) or during motion refinement (in green).

3.5.3 Real-world Applications

We demonstrated our proposed method in various scenarios to show its capabilities. Figure 3.6 shows the results in two scenes, “rotated book” and “cup and bottle”. For

Table 3.2: Object reconstruction error (avg./std., in cm)

Method	CF	Ours with v.c. tracking	**Ours**	Ours with GT-Seg
Sofa	1.72/1.62	1.68/1.90	**0.74/0.79**	0.46/0.59
Chair	1.19/**1.33**	1.13/1.58	**1.00**/1.66	0.92/1.74

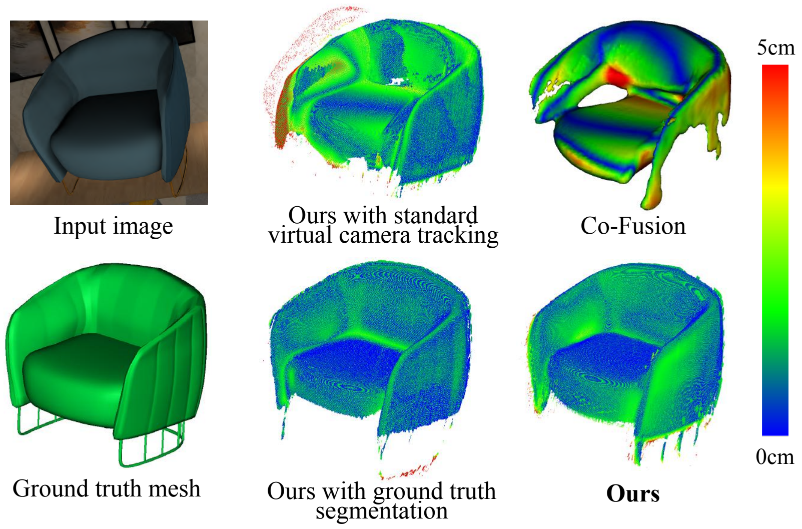
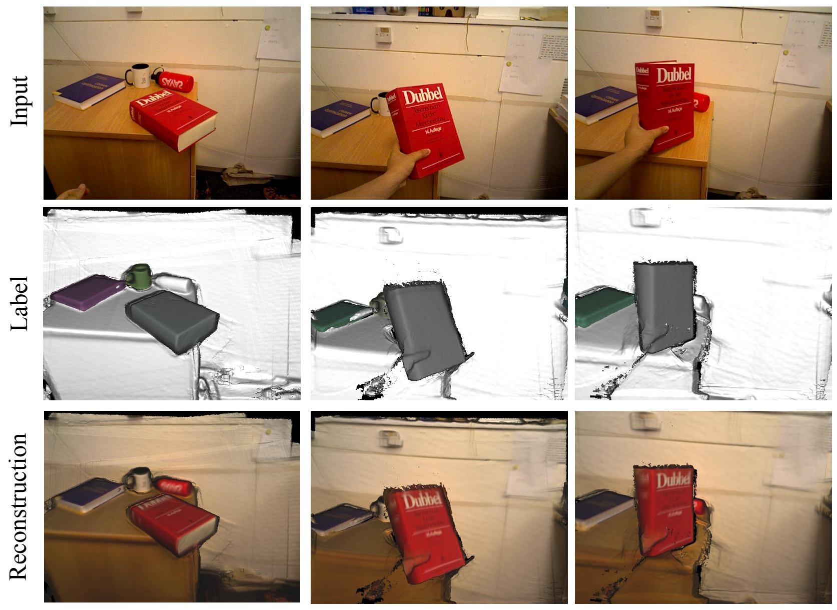

Input image
Ours with standard virtual camera tracking
Co-Fusion
5cm
0cm
Ground truth mesh
Ours with ground truth segmentation
Ours

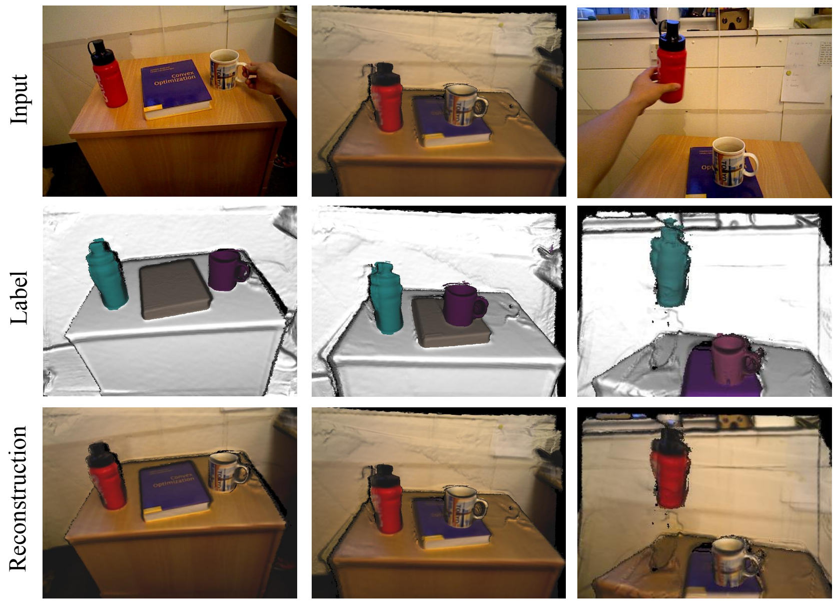
Figure 3.5: Comparison of reconstruction error for a moving sofa.

each input image, we provide label image and reconstruction to show the detailed reconstruction, reliable tracking, and segmentation. With separate volumetric maps for each object, our object models do not collide with each other, which is more suitable for multiple instance SLAM than a surfel-based system.

Figure 3.7 also shows a scene where our system can simultaneously support the robust tracking of more than 6 moving objects while maintaining a highly detailed reconstruction. As a qualitative comparison, we also show the reconstruction from Co-Fusion, which did not segment and reconstruct these moving objects successfully because the motion was not of sufficient magnitude. In addition, surfel-based systems, such as Co-Fusion and MaskFusion, do not provide the same level of details per object. On the contrary, our system can maintain highly detailed reconstructions and nevertheless keep efficient memory usage thanks to the octree data structure. More results can be seen in the video attachment.



(a) "rotated book" scene



(b) "cup and bottle" scene

Figure 3.6: Qualitative demonstration: input RGB (top row), semantic class prediction (middle row) and geometry reconstruction result (bottom row).

3.5.4 Runtime Analysis

We evaluated the average computational time for the components of our dynamic SLAM system in different sequences with approximately 3 to 6 objects being moved.

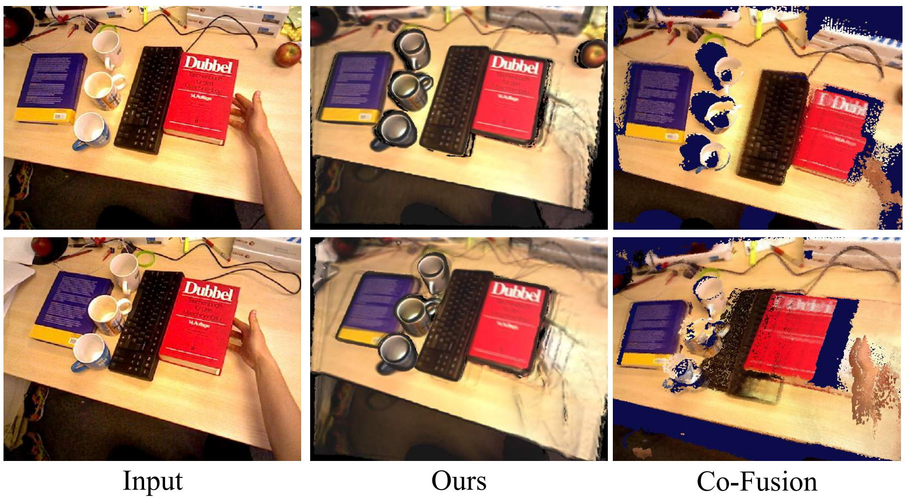


Figure 3.7: Qualitative comparison with Co-Fusion: input RGB (left column), our reconstruction results (middle) and Co-Fusion results (right column).

Table 3.3: Run-time analysis of system components (ms)

Components	Tracking	Segmentation	Integration	Raycasting
Time (ms)	43/MO	10/VO	12/VO.	8/VO

Processing time (all on CPU) for each frame averages 400 ms with more than 25 objects being generated in the scene. When a new object is detected, the initialisation takes around 10 ms per object. Tracking time scales mainly with moving objects (MO) while segmentation, integration and raycasting scales with visible objects (VO). A more-detailed breakdown of computation time for each component is shown in Table 3.3.

We would like to highlight that our current system only runs on CPU without being highly optimised for performance yet. We believe a high frame-rate version of our system is achievable by exploiting GPU parallelisation.

3.6 Conclusions and Discussions

We present a novel approach for multi-instance dynamic SLAM using an octree-based volumetric representation. It robustly tracks camera pose in dynamic environments and continuously estimates dense geometry, semantics, and object foreground probabilities. Experimental results in various scenarios demonstrate the effectiveness of our method in dynamic indoor environments. We hope our method paves the way for new applications in indoor robotic applications, where an awareness of environment change, free space, and object-level information will uplift the next generation of mobile robots.

Despite working generally well in most cases, our proposed method still has several issues, which may limit its wide applications in real-world scenarios. One issue we experimentally found is that the photometric loss employed in our object tracking Equation (3.10) component can only work for objects showing less glossy reflections. The camera view change or the object surface normal change caused by object motion will violate the brightness constancy assumption used in photometric tracking, even for objects with lambertian surfaces. In the next chapter, we will present how we can learn some robust deep features and associated feature-metric uncertainties to provide robust tracking under strong lighting changes and wide baseline conditions. Besides, the projective data association method in raycasting implies that the camera and objects do not exhibit large motion in consecutive frames. This is a strong assumption used in many dense SLAM systems, but is not always valid in practice. Some other works used different motion models, such as constant velocity [Bescós et al., 2021] or a white-noise-on-acceleration prior [Huang et al., 2020]. These SLAM systems, however, lack the capability to reconstruct dense geometries for object models.

Chapter 4

Deep Probabilistic Feature-metric Tracking

Dense image alignment from RGB-D images remains a critical issue for real-world applications, especially under challenging lighting conditions and in a wide baseline setting. In this chapter, we propose a new framework to learn a pixel-wise deep feature map and a deep feature-metric uncertainty map predicted by a Convolutional Neural Network (CNN), which together formulate a deep probabilistic feature-metric residual of the two-view constraint that can be minimised using Gauss-Newton in a coarse-to-fine optimisation framework. Furthermore, our network predicts a deep initial pose for faster and more reliable convergence. The optimisation steps are differentiable and unrolled to train in an end-to-end fashion. Due to its probabilistic essence, our approach can easily couple with other residuals, where we show a combination with ICP. Experimental results demonstrate state-of-the-art performances on the TUM RGB-D dataset and the 3D rigid object tracking dataset. We further demonstrate our method's robustness and convergence qualitatively.

Contents of Chapter

Parts of this Chapter appear in:

Binbin Xu, Andrew J. Davison, Stefan Leutenegger (2021). Deep Probabilistic Feature-metric Tracking. *IEEE Robotics and Automation Letters (RA-L), Vol. 6, No. 1, pp. 223-230, 2021.* [Xu et al., 2021a]

It has also been selected in ICRA 2021 presentation and received a **RA-L Best Paper Honorable Mention Award**.

4.1 Introduction

Dense image alignment [Lucas and Kanade, 1981] using the photometric residual has been widely applied in 2D tracking [Shi and Tomasi, 1994], 3D object tracking [Xu et al., 2019], optical flow [Horn and Schunck, 1981], and SLAM [Newcombe et al., 2011b]. In visual SLAM, it leads to two types of estimator designs: direct sparse [Engel et al., 2017] and direct dense type [Newcombe et al., 2011b]. There has been an argument that dense methods that utilise information from all image pixels should exhibit better performance in terms of robustness and accuracy. However, this is not necessarily the case in reality, as investigated in [Platinsky et al., 2017], especially compared to the performance achieved by systems using the indirect sparse residual formulation (reprojection error) [Mur-Artal and Tardós, 2017].

One reason is that lighting change and reflection in real scenes break the brightness constancy assumption [Horn and Schunck, 1981] commonly used in dense image alignment. Thus the resulting dense photometric residual cannot be well explained by the Gaussian distribution assumed in the Gauss-Newton scheme, which is in contrast to reprojection error minimisation that may still work robustly as long as sparse feature matches can be established. Secondly, the photometric residual considers only very local color consistency, which requires a good initialisation close to the global minimum. This leads to a poorer estimation accuracy when the baseline gets larger. On the contrary, the keypoint reprojection residual models a global constraint using a sparse feature descriptor, leading to better convergence properties.

In this chapter, we are trying to address these issues by replacing raw intensity image alignment with deep feature map alignment. Different from the existing learning-based feature-metric alignment [Czarnowski et al., 2017, Lv et al., 2019, Tang and Tan, 2019, Schmidt et al., 2017, von Stumberg et al., 2020], we argue that the feature-metric residual should incorporate not simply the feature difference

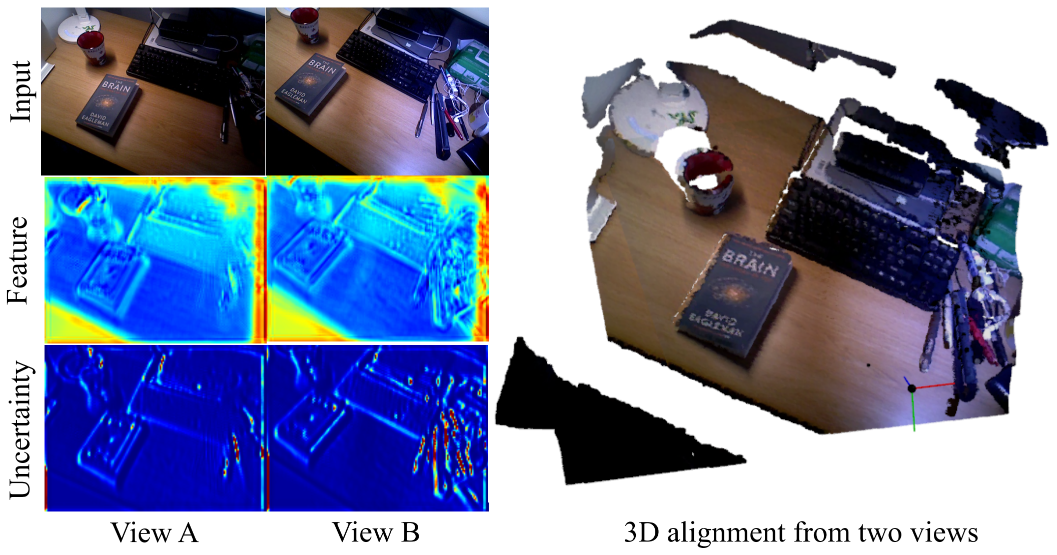


Figure 4.1: We propose a probabilistic feature-metric tracking method that estimates dense feature and uncertainty maps from a pair of RGB-D images to optimise the relative pose between them. Our method can handle strong lighting changes and large motion scenarios by leveraging features that are robust to lighting changes, e.g. on the desk surface, and predicting high uncertainties on areas that the network cannot handle, e.g. for the strong lighting changes near the pens.

but also the corresponding uncertainty. Predictions from neural networks inherently are uncertain, which can be estimated [Kendall and Gal, 2017]. Secondly, and also importantly, SLAM has most successfully been posed as a probabilistic problem, where uncertainty of the residuals has to be known [Thrun et al., 2005], in particular when fusing different sensors and residuals. We will show how our feature-metric residuals can be combined with geometric ICP residuals using uncertainties to further improve results. The proposed probabilistic feature-metric residuals are minimised using coarse-to-fine Gauss-Newton optimisation. To ensure that the learned feature-metric cost landscape is suitable for the Gauss-Newton optimisation, we unroll the iterative optimisation steps and train the whole pipeline end-to-end. To handle the initialisation issue in the wide baseline case, we include training pairs with varied baselines and propose to replace the identity initialisation with a predicted initial pose from a pose network. This can improve the system convergence by bringing the initialisation into the convergence basin of the correct minimum.

As shown in Figure 4.1, the proposed method can handle large motion and strong illumination variance. The learned features are robust to lighting changes in most regions, e.g. reflection on desk surface, and the uncertainty map (red means high uncertainty) can downweigh the region, e.g. pens, where the feature predictions are uncertain. In summary, we make the following contributions:

1. We propose a dense probabilistic feature-metric residual, where a CNN predicts both feature and uncertainty maps used for non-linear least-squares minimisation to estimate the relative camera or object pose.
2. In our CNN architecture, we propose a coupled feature encoder and pose predictor network, which combines the learning-based initial pose prediction and the learned features/uncertainties for pose optimisation, and train them together end-to-end.
3. We further demonstrate how our proposed probabilistic feature-metric residual can easily lend itself to integration with other residuals, where a classic ICP residual is showcased.

We evaluate our proposed method on the TUM RGB-D SLAM dataset [Sturm et al., 2012] and MovingObjects3D rigid motion dataset [Lv et al., 2019]. We provide ablation studies to validate each contribution component. We further provide a qualitative evaluation on the convergence basin and demonstrate the robustness under strong lighting changes.

4.2 Related Work

Feature-metric Alignment: To relax the brightness constancy constraint in direct image alignment, several recent works have exploited the feature-metric alignment by utilising features from neural networks. [Jaramillo et al., 2017, Czarnowski et al., 2017] replace image intensity with high-dimensional features extracted from a pre-trained neural network for tracking and show a better robustness than using image intensity. However, the pre-trained features are not naturally consistent across different views and the redundancy in the pre-trained very high-dimensional features means a high cost of memory and computation time.

[Schmidt et al., 2017] proposes to learn a robust feature descriptor suitable for estimating dense correspondence in different lighting conditions and viewpoints using the contrastive loss [Hadsell et al., 2006]. [von Stumberg et al., 2020] combines the contrastive loss with a Gauss-Newton loss, which includes a 2-dimensional pixel position uncertainty, to train dense features. However, both of these works generate a feature map good for correspondence matching rather than alignment. The composed residuals do not necessarily fit well with the least square optimisation used for pose estimation. This is why [Schmidt et al., 2017] requires a RANSAC step for refinement and [von Stumberg et al., 2020] is only used for re-localisation.

Recently, some methods start to explore how to combine the feature map learning more tightly with the least-square optimisation of camera tracking, based on the differentiable property of iterative optimisation. [Wang et al., 2018] learn feature maps for 2D image tracking in the Lucas-Kanade framework. [Tang and Tan, 2019] propose feature-metric bundle adjustment for 3D reconstruction. [Bloesch et al., 2019] propose to use feature maps for depth prediction and pose estimation. However, these works only consider a spatial correlation in feature generation, ignoring the temporal correlation in input image pairs. Quite related to our work, [Lv et al., 2019] propose a spatio-temporal feature encoder by concatenating two views for the network input and further propose an m-estimator network and damping network for pose optimisation. However, different from ours, none of these works exploit feature-metric uncertainty in their settings, nor combine a pose predictor to boost convergence.

Deep Pose Prediction: A different way to estimate pose from a pair of images is to leverage CNN predictions directly [Zhou et al., 2017, Ummenhofer et al., 2016]. Learning a direct mapping from input images to 6D relative pose skips potential convergence issues of least-squares optimisation. However, it requires a large number of model parameters and a vast amount of training data, while not necessarily generalising to new scenes.

To improve accuracy and generalisation, some recent works include coarse-to-fine estimation [Zhou et al., 2018a] and iterative refinement [Li et al., 2018c] to estimate a relative transformation. Despite some shared network parameters in iterations, these works still come with a much larger model capacity (i.e. parameter number) than the ones using optimisation – even those with learned features – and do not necessarily show an advantage in terms of pose accuracy. To better leverage both types of approaches, we propose a coarse-to-fine optimisation using learned features and uncertainties, plus a direct pose prediction on the coarsest layer serving as an *initial guess*, which takes the output from the coarsest level two-view encoder as an input to make it compact.

Uncertainty Learning: Safety considerations have prompted recent works on uncertainty estimation of deep learning, as discussed in [Kendall and Gal, 2017] and applied to several tasks [Kendall et al., 2018]. [Tateno et al., 2017] fuse the predicted depth into a monocular SLAM system and estimate the depth uncertainties via its difference with the nearest key-frame. [Zhou et al., 2018a] propose to estimate both depth and pose uncertainty in their depth and pose prediction networks. [Liu et al., 2019] formulate the depth uncertainty differently using a probability volume. Recently, D3VO [Yang et al., 2020] propose to estimate the photometric uncertainties and predict a relative pose to initialise the pose optimisation. Most of these works, if not all, model the uncertainty based on the difference between the prediction and the ground truth values. In contrast to these works, we propose a novel feature-metric uncertainty and learn it without ground truth feature maps available in the training. Instead, we formulate the uncertainty in a novel probabilistic feature-metric residual and learn it implicitly as part of the least-squares optimisation. The learned features and uncertainties should lead to a better optimised pose via training back-propagation.

4.3 Method

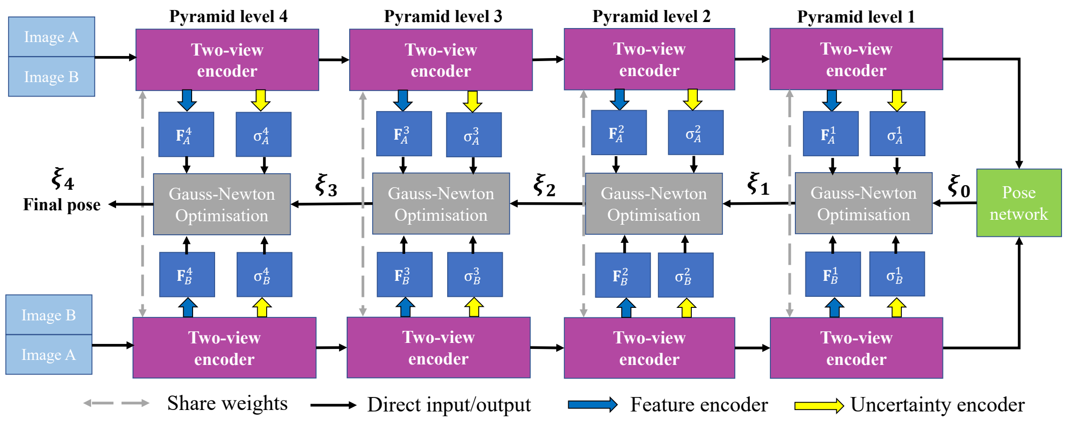

Pyramid level 4
Pyramid level 3
Pyramid level 2
Pyramid level 1
Image A
Image B
Two-view encoder
F_A^4 σ_A^4 F_A^3 σ_A^3 F_A^2 σ_A^2 F_A^1 σ_A^1
ξ_4
Final pose
Gauss-Newton Optimisation
ξ_3 ξ_2 ξ_1 ξ_0
Pose network
F_B^4 σ_B^4 F_B^3 σ_B^3 F_B^2 σ_B^2 F_B^1 σ_B^1
Image B
Image A
Share weights
Direct input/output
Feature encoder
Uncertainty encoder

Figure 4.2: Overview of our proposed deep probabilistic feature-metric tracking method. For two views, we input image A and image B, by concatenating them as $\{A, B\}$ and $\{B, A\}$, respectively, to our two-view encoder pyramid network. At each pyramid level, we extract the output from the two-view encoder and feed it into the feature encoder and uncertainty encoder separately to extract dense feature and uncertainty maps. Then we optimise the pose by minimising the proposed probabilistic feature-metric residual, which is initialised by the pose from the coarser level. On the coarsest level, we concatenate the outputs of the two views from the two frames and run through the pose network to obtain an initial pose prediction.

Figure 4.2 shows an overview of our system. For a pair of RGB-D frames, frame A $\underrightarrow{\mathcal{F}}_A$ and frame B $\underrightarrow{\mathcal{F}}_B$, our aim is to estimate its relative transformation $\boldsymbol{T}_{AB} = (\mathbf{C}_{AB}, {}_A\mathbf{r}_{AB}) \in (SO(3) \times \mathbf{R}^3)$, from $\underrightarrow{\mathcal{F}}_B$ to $\underrightarrow{\mathcal{F}}_A$. We represent $\boldsymbol{T}_{AB}$ in twist coordinates ξ by $\boldsymbol{T}_{AB}(\xi) = \exp(\xi_{AB})$. Each frame has a depth map $\mathbf{D}$ and a color image $\mathbf{I}$. The network components in our whole system are denoted as ϕ, with the two-view spatio-temporal encoder ϕ_θ, the feature encoder ϕ_F, the uncertainty encoder ϕ_σ, and the pose network ϕ_T. The weights are shared across the two views for ϕ_θ, ϕ_F, and ϕ_σ. The architecture details of all our network components can be found in the Section 4.3.5.

To extract the spatial and temporal correlation between two frames, we first concatenate the input colour and depth image along the feature channel and feed them through the two-view spatio-temporal encoder pyramid network:

$$\mathbf{W}_A^i = \phi_\theta(\{\mathbf{I}_A, \mathbf{D}_A, \mathbf{I}_B, \mathbf{D}_B\}), \qquad \mathbf{W}_B^i = \phi_\theta(\{\mathbf{I}_B, \mathbf{D}_B, \mathbf{I}_A, \mathbf{D}_A\}), \tag{4.1}$$

where $\mathbf{W}_A^i$ and $\mathbf{W}_B^i$ are the outputs of the two-view encoder at level i, $i \in 1, 2, 3, 4$, for frame A and B respectively and $\{,\}$ is the concatenation operation. On each pyramid level, we extract the dense feature and uncertainty maps by feeding the two-view encoder outputs into the feature encoder branch and the uncertainty encoder branch:

$$\mathbf{F}_X^i = \phi_F(\mathbf{W}_X^i), \qquad \sigma_X^i = \phi_\sigma(\mathbf{W}_X^i), \tag{4.2}$$

where $X \in A, B$. i will be omitted later when we explain operation on the same pyramid level. Different from [Lv et al., 2019] which averages the output features map into one single channel, we maintain a same high-dimensional feature map at different pyramid levels. This choice is motivated by the hypothesis that higher dimensionality should lead to higher discriminative power of the features – which we support in the experimental section.

4.3.1 Probabilistic Feature-metric Residual for Pose Estimation

In probabilistic estimation that assumes an underlying Gaussian distribution of the residuals, we equivalently minimise the weighted least squares, with the inverse covariance matrix acting as the weight. Given the dense feature and uncertainty maps on two views and an estimated pose ξ_{AB}, we propose a probabilistic feature-metric residual as an uncertainty-normalised feature difference:

$$\mathbf{r}_f(\xi_{AB}) = \frac{\bar{\mathbf{r}}_f(\xi_{AB})}{\sigma_f(\xi_{AB})} = \frac{\mathbf{F}_A[\boldsymbol{u}_A(\xi_{AB})] - \mathbf{F}_B[\boldsymbol{u}_B(\xi_0)]}{\sqrt{\sigma_A^2[\boldsymbol{u}_A(\xi_{AB})] + \sigma_B^2[\boldsymbol{u}_B(\xi_0)]}}, \tag{4.3}$$

where $\boldsymbol{u}_A$ and $\boldsymbol{u}_B$ are a pair of pixel correspondences on the two frames. $\boldsymbol{u}$ represents image pixel coordinates. $\boldsymbol{u}_B(\xi_0)$ means $\boldsymbol{u}_B$ is perturbed under zero transformation ξ_0. $\bar{\mathbf{r}}_f$ is the feature difference between the correspondences on the feature map and σ_f is the joint uncertainty estimate for the correspondence that we obtain as a combination from the individual uncertainties. Note that this assumes isotropic uncertainty w.r.t. each feature dimension – a simplification we chose (for speed) that may be revisited. Section 4.3.1 encourages the feature map from two different views to be as similar as possible while downweighs the features that the network is uncertain about from the either view with the predicted uncertainties. As shown in example Figure 4.1, the trained features are robust to moderate lighting, reflection and view perspective variances and the trained uncertainties handle the uncertain features caused by the extreme lighting changes (lower right corner). The dense correspondence lookup is implemented via warping from frame B to frame A through ξ_{AB}, which can be defined as:

$$\boldsymbol{u}_A(\xi_{AB}) = \pi(\boldsymbol{T}_{AB}(\xi)\pi^{-1}(\boldsymbol{u}_B, D_B[\boldsymbol{u}_B])), \tag{4.4}$$

where [.] represents the pixel lookup (including bilinear interpolation). π and π^{-1} denote the projection function to the image plane and the back-projection function to 3D (homogeneous) coordinates, respectively. By inserting Section 4.3.1 into a Lucas-Kanade framework [Lucas and Kanade, 1981], we formulate the pose estimation problem of an optimal pose ξ^* as:

$$\xi^* = \underset{\xi}{\operatorname{argmin}} \frac{1}{2} \sum_{\boldsymbol{u}_B \in \mathcal{V}} \mathbf{r}_f^T(\xi)\mathbf{r}_f(\xi), \tag{4.5}$$

i.e. summing all residuals over non-occluded pixels in B, $\mathcal{V}$, which can be iteratively solved by e.g. the Gauss-Newton method. To speed up the computation, we choose the inverse compositional formulation [Baker and Matthews, 2004] that updates poses by applying the incremental pose on frame B. It allows for a more efficient computation of the feature-metric Jacobians. In each iteration, the pose is updated by $\Delta\xi$ as:

$$\begin{aligned} \xi_{k+1} &= \xi_k \circ \Delta\xi^{-1}, && (4.6) \\ \Delta\xi &= -(\mathbf{J}_f^T \mathbf{J}_f)^{-1}(\mathbf{J}_f^T \mathbf{r}_f). && (4.7) \end{aligned}$$

$\mathbf{J}_f$ is the Jacobian of the probabilistic feature-metric residual $\boldsymbol{r}_f$ w.r.t. the relative pose ξ_{AB}:

$$\mathbf{J}_f = \frac{\partial \mathbf{r}_f}{\partial \xi_{AB}} = -\left(\frac{\nabla \mathbf{F}_B}{\sigma_f(\xi_{AB})} + \frac{\bar{\mathbf{r}}_f(\xi_{AB})\sigma_B \nabla \sigma_B}{\sigma_f^3(\xi_{AB})} \right) \frac{\partial \boldsymbol{u}_B}{\partial \xi_0}, \tag{4.8}$$

where $\nabla \mathbf{F}_B$ and $\nabla \sigma_B$ are the gradients of the feature maps and uncertainty maps along the two pixel dimensions in frame B, respectively. Under this formulation, only the components of $\sigma_f(\xi)$ and $\bar{\mathbf{r}}_f(\xi)$ need to be re-evaluated in each iteration, which can be shared when computing the residuals in Section 4.3.1. All the other components in Section 4.3.1 can be pre-computed to speed up the computation.

4.3.2 A Probabilistic Combination with ICP Residual

As an uncertainty-driven residual, our proposed residual can be naturally combined with other residuals. For example, if we assume that depth measurement from an RGB-D sensor is reliable, we can further combine the feature-metric residual with an ICP residual to directly add a geometric constraint. The combined residual equation is:

$$\xi^* = \underset{\xi}{\operatorname{argmin}}\, \mathbf{r}_f^T(\xi)\mathbf{r}_f(\xi) + w_g \mathbf{r}_g^T(\xi)\Sigma_g^{-1}\mathbf{r}_g(\xi), \tag{4.9}$$

where $\boldsymbol{r}_g$ and Σ_g are the ICP residual and uncertainty, respectively, and w_g is the weight for ICP residual. The above equation can still be iteratively solved via the Gauss-Newton method. The detailed definitions of the ICP residual and Jacobian can be found in [Rusinkiewicz and Levoy, 2001]. As there are no regularisation terms in Section 4.3.1, our learned uncertainty is a scale-free parameter. When combining with other residuals of different magnitudes, we need to scale them properly before fine-tuning to bootstrap the training. The scale of ICP weight w_g is chosen (as $w_g = 0.01$) such that the individual Chi-square errors are of similar magnitude, after which the joint ICP/feature-metric training will scale the features and feature-metric uncertainties to be best balanced with the ICP.

4.3.3 Coarse-to-fine Optimisation and Initialisation

The cost functions in Section 4.3.1 and Section 4.3.2 can be optimised in a coarse-to-fine way using damped Gauss-Newton optimisations, which is applied on 4 pyramid levels, with a fixed number of rolled-out iterations, i.e. 3, on each level. We added a small damping constant in Section 4.3.1 to prevent the matrix inversion to be ill-conditioned. Coarse-to-fine optimisation methods are sensitive to coarse-level estimation, where the incorrect estimations will be propagated to finer levels and the iterative optimisation may get stuck in a wrong local minimum, especially in a wide-baseline setting. To tackle this issue, we train a pose network to bootstrap the optimisation by predicting an initial relative pose on the coarsest level, instead of using a conventional identity pose initialisation. To make the network compact, the concatenated outputs from the coarsest-level two-view encoder on the two frames serve as the inputs to our pose prediction network:

$$\xi_0 = \phi_T(\{\mathbf{W}_A^1, \mathbf{W}_B^1\}). \tag{4.10}$$

To account for the multi-modal information on the coarse level, the deep initial pose network outputs K pose hypotheses, which are parameterised as 3 Euler angles and 3D translation vectors, and a respective confidence probability for each hypothesis. The final predicted pose is the weighted average of all hypotheses.

4.3.4 Training Setup

The predicted initial pose and the estimated poses per pyramid level are compared to the ground truth pose and the resulting gradients in the optimisation are used for back-propagation to update all the learning weights. To balance influence of rotation vs. translation, we use the 3D End-Point-Error (EPE) as the training loss: given the ground truth relative transformation $\boldsymbol{T}_{AB}(\xi)$ and the estimated/predicted pose $\boldsymbol{T}_{AB}(\xi_i)$, the loss is composed as:

$$L = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{I}} \sum_{{}_B\mathbf{v} \in \mathcal{V}} \| \boldsymbol{T}_{AB}(\xi)\, {}_B\mathbf{v} - \boldsymbol{T}_{AB}(\xi_i)\, {}_B\mathbf{v} \|_2^2 , \tag{4.11}$$

where $\mathcal{V}$ is the set of backprojected 3D points ${}_B\mathbf{v}$ in the frame B, $\mathcal{I} = \{0, 1, 2, 3, 4\}$ denotes the pyramid levels, ξ_0 is the predicted pose from the pose network and the other ξ_i are the estimated poses at the final iteration of Gauss-Newton optimisations on the respective pyramid level. This formulation enables the network to learn both feature and uncertainty representations in an end-to-end fashion, without the need for a ground truth feature map or ground truth correspondences, and without requiring an explicit definition of the uncertainty model. We set the feature map channels to be 8. Note that the uncertainty is defined as a scalar value. We unroll the Gauss-Newton optimisation and train all the models together from scratch using ADAM [Kingma and Ba, 2015] for 30 epochs, with a learning rate initialized at 0.0005 and reduced at epochs [5, 10, 20]. When combining the ICP residual, we do a further fine-tuning for 10 epochs.

4.3.5 Implementation Details

Figure 4.3 shows the architecture of our two-view encoder which takes the input from a pair of RGB-D images and extracts spatio-temporal correlation information from that. It is constructed into a 4-level pyramid architecture, where each level outputs a higher-dimension information. The architecture is modified from [Lv et al., 2019], however, we do not perform an average operation to extract feature

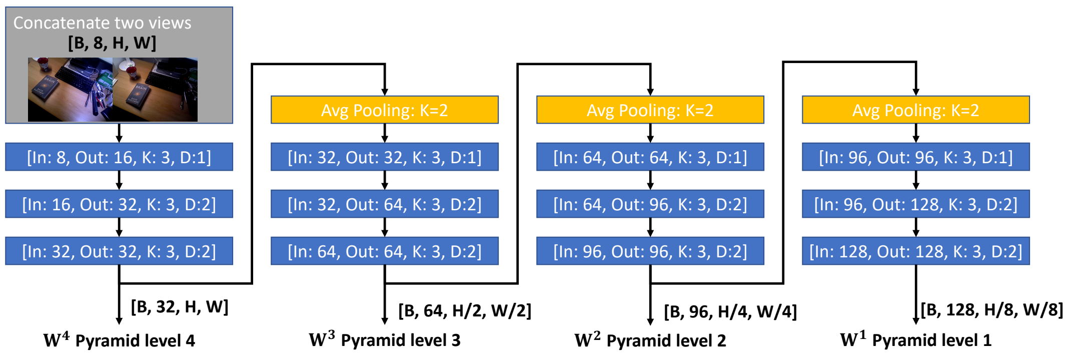

Concatenate two views
[B, 8, H, W]
[In: 8, Out: 16, K: 3, D:1]
[In: 16, Out: 32, K: 3, D:2]
[In: 32, Out: 32, K: 3, D:2]
[B, 32, H, W]
W⁴ Pyramid level 4
Avg Pooling: K=2
[In: 32, Out: 32, K: 3, D:1]
[In: 32, Out: 64, K: 3, D:2]
[In: 64, Out: 64, K: 3, D:2]
[B, 64, H/2, W/2]
W³ Pyramid level 3
Avg Pooling: K=2
[In: 64, Out: 64, K: 3, D:1]
[In: 64, Out: 96, K: 3, D:2]
[In: 96, Out: 96, K: 3, D:2]
[B, 96, H/4, W/4]
W² Pyramid level 2
Avg Pooling: K=2
[In: 96, Out: 96, K: 3, D:1]
[In: 96, Out: 128, K: 3, D:2]
[In: 128, Out: 128, K: 3, D:2]
[B, 128, H/8, W/8]
W¹ Pyramid level 1

Figure 4.3: The architecture of our two-view encoder. It is composed of basic convolutional blocks (blue) and average pooling operations (yellow). The basic convolutional block is grouped by a convolutional layer and followed by a BatchNorm layer, and a ELU layer. [In, Out, K, D] represents [Input channel, Output channel, Kernel size, Dilation] with stride always being 1.

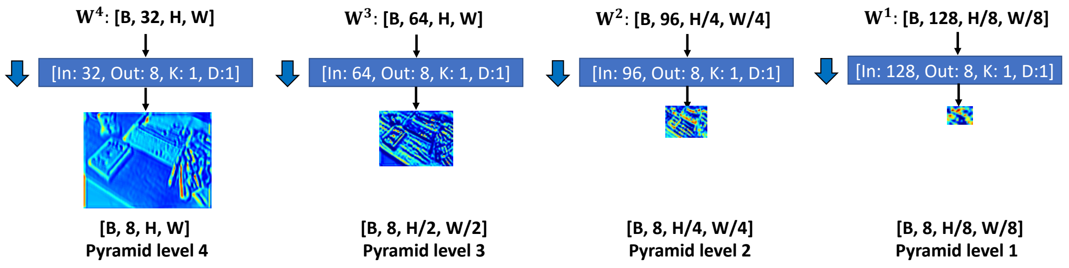
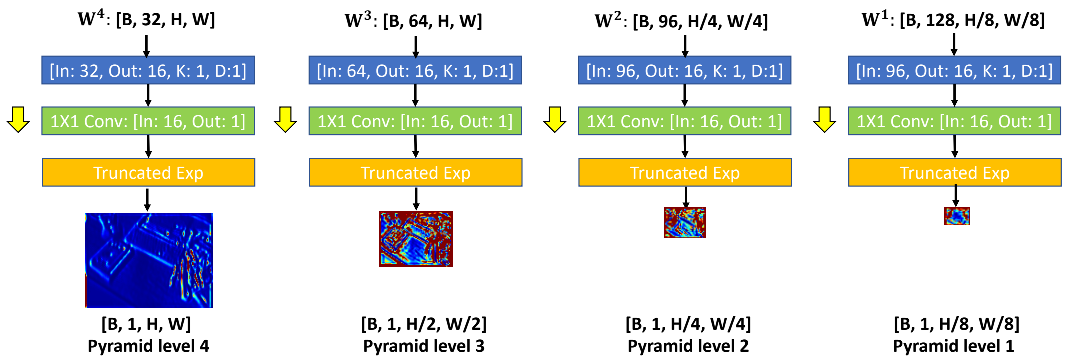


Figure 4.4: The architecture of our feature encoder. On each pyramid level, it is a basic convolutional block that is group by a 1 by 1 convolutional layer, a BatchNorm layer, and a ELU layer. [In, Out, K, D] represents [Input channel, Output channel, Kernel size, Dilation] with stride always being 1.

maps. Instead, we send the outputs to the feature encoder and the uncertainty encoder to estimate the feature and uncertainty maps.

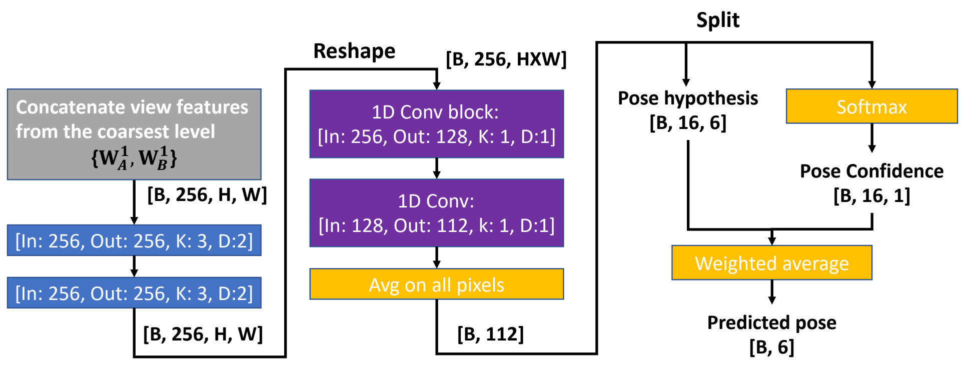
Figure 4.4 shows the architecture of our feature encoder on each pyramid level. It takes the input from the two-view encoder and predicts an 8-dimensional feature map.

Figure 4.5 shows the architecture of our uncertainty encoder on each pyramid level. It takes the input from the two-view encoder and predicts a 1-dimensional uncertainty map. To ensure positive uncertainty values, we assume the output from the 1 by 1 convolutional layer is a logarithmised uncertainty and use the



W^4: [B, 32, H, W]
[In: 32, Out: 16, K: 1, D:1]
1X1 Conv: [In: 16, Out: 1]
Truncated Exp
[B, 1, H, W]
Pyramid level 4
W^3: [B, 64, H, W]
[In: 64, Out: 16, K: 1, D:1]
1X1 Conv: [In: 16, Out: 1]
Truncated Exp
[B, 1, H/2, W/2]
Pyramid level 3
W^2: [B, 96, H/4, W/4]
[In: 96, Out: 16, K: 1, D:1]
1X1 Conv: [In: 16, Out: 1]
Truncated Exp
[B, 1, H/4, W/4]
Pyramid level 2
W^1: [B, 128, H/8, W/8]
[In: 96, Out: 16, K: 1, D:1]
1X1 Conv: [In: 16, Out: 1]
Truncated Exp
[B, 1, H/8, W/8]
Pyramid level 1

Figure 4.5: The architecture of our uncertainty encoder. On each pyramid level, it is composed by a basic convolutional block, followed by a 1 by 1 convolutional layer and a truncated exponential operation.



Concatenate view features from the coarsest level $\{W_A^1, W_B^1\}$
[B, 256, H, W]
[In: 256, Out: 256, K: 3, D:2]
[In: 256, Out: 256, K: 3, D:2]
[B, 256, H, W]
Reshape
[B, 256, HXW]
1D Conv block: [In: 256, Out: 128, K: 1, D:1]
1D Conv: [In: 128, Out: 112, k: 1, D:1]
Avg on all pixels
[B, 112]
Split
Pose hypothesis [B, 16, 6]
Softmax
Pose Confidence [B, 16, 1]
Weighted average
Predicted pose [B, 6]

Figure 4.6: The architecture of our pose network for initial pose prediction.

exponentiation operation to recover the true uncertainty. We experimentally found that this leads to better performance than the Gaussian distribution assumption. The output is then truncated to avoid gradient explosion.

Figure 4.6 shows the architecture of our pose network to predict an initial pose on the coarsest level of the coarse-to-fine Gauss-Newton optimisation. It takes the input from a concatenation of the outputs of the two frames from the two-view encoder at the coarsest level. Similar to [Zhou et al., 2018a], the initial pose network also predicts multiple pose hypotheses and then fuse them together using their respective confidences. Here, we choose the hypotheses number to be 16. The pose is parameterised with 3 Euler angles and a 3-dimensional translation vector.

4.4 Experiments

4.4.1 Quantitative Evaluation and Discussion

We first evaluate our method on the **TUM RGB-D** SLAM dataset [Sturm et al., 2012]. It contains various camera motions, lighting conditions, and scene structures. A natural extension is to apply it to 3D rigid object motion estimation, which we test on the **MovingObjects3D** dataset [Lv et al., 2019]. It is a synthetic dataset rendered from Blender and contains 6 different categories of moving objects under varied illumination changes . We trained different network weights on each benchmark dataset separately.

DeepIC [Lv et al., 2019] is chosen as our main baseline method, which also learns dense feature map for pose optimisation. It also learns an m-estimator network and a damping network in its Levenberg–Marquardt optimisation, but without feature-metric uncertainty or initial pose predictions. To have a fair comparison, we use the same experimental setting as theirs. We randomly subsampled frames B at intervals {1,2,4,8} relative to frame A from TUM RGB-D dataset [Sturm et al., 2012] and {1,2,4} from MovingObjects3D dataset [Lv et al., 2019] to generate various motion magnitudes and tracking difficulties as the training pairs. A comparison to this approach would show the importance of the uncertainty prediction and the initial pose prediction in our proposed method. We use the network weights provided by the authors in our evaluations. We further implemented **DeepIC+P**, an augmented variant of DeepIC [Lv et al., 2019], with our pose prediction network to initialise their optimisation. The same number of iterations and pyramid levels are used as in our method. A comparison to it would further verify the contribution of our proposed probabilistic feature-metric loss.

To have a comparison to deep pose prediction methods that directly predict a relative transformation from two views, we implemented a **coarse-to-fine PoseNet**, similar to the tracking part in DeepTAM [Zhou et al., 2018a]. It is implemented on four pyramid levels for coarse-to-fine pose refinements, where the predicted pose from a coarser pyramid level would be used to bootstrap the prediction on a finer level. The network architecture is similar to our pose network but with different network weights on different pyramid levels. A comparison to it would show a benefit of our learning-based optimisation approach for pose estimation. We further included the iterative refinement idea from [Li et al., 2018c] to the coarse-to-fine PoseNet approach. The **iterative PoseNet** has 3 iteration refinements on each pyramid level. All the learning-based comparison approaches are trained end-to-end using the loss in Equation (4.11).

For the non-learning approaches, we compare our method to the pure geometric Point-to-Plane **ICP** method [Rusinkiewicz and Levoy, 2001], which is essentially robust to illumination changes. We also include an **RGB-D VO** method [Steinbrücker et al., 2011] in the camera motion evaluation. The ICP and RGB-D VO approaches are implemented in Open3D [Zhou et al., 2018b]. A comparison to these approaches would show benefits of learning-based approaches, in terms of larger convergence basin and better accuracy, even under challenging lighting conditions.

To reveal the contribution of each component, we provide a detailed ablation study. We denote our system component, dense feature map, dense uncertainty map, deep initial pose prediction as F, U, P, respectively. We select the following settings. **Ours (F)**: We replace the uncertainty prediction with an identity uncertainty and disable the pose prediction with an identity pose initialisation. **Ours (F+P)**: We replace the uncertainty prediction with an identity uncertainty. **Ours (F+U)**: We disable the pose prediction and only use the proposed probabilistic feature-metric residual for alignment. **Ours (F+U+P)**: A full version of our probabilistic feature-metric tracking system. **Ours+ICP**: A combination of the probabilistic feature-metric and ICP residuals. All these combinations are implemented in coarse-to-fine optimisations, with the same number of iterations and pyramid levels as in the proposed method.

The evaluation metrics are the 3D EPE loss in Equation (4.11) and the relative pose error (RPE) metrics defined in TUM RGB-D dataset [Sturm et al., 2012].

TUM RGB-D Dataset: We use the same setting as DeepIC [Lv et al., 2019], where sequences 'fr1/360', 'fr1/desk', 'fr2/360', and 'fr2/pioneer360' are used for testing and the remaining sequences are split into training (first 95% of each sequence) and validation (last 5%). Images are transformed to a resolution of 160×120, with depth values outside of 0.5m to 5.0m being ignored. We use a mixture of all subsampled keyframe intervals to train our network, and evaluate the methods separately for each keyframe interval.

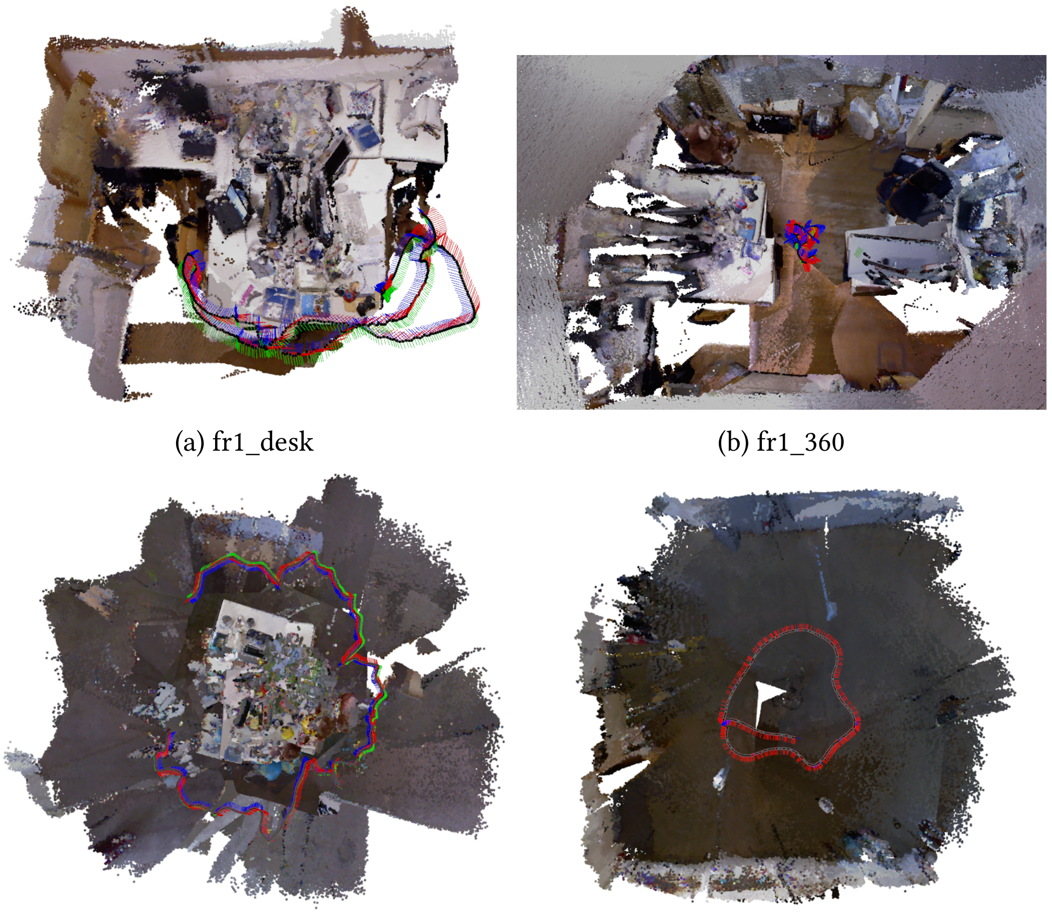
Table 4.1 summarises the results on the TUM RGB-D dataset. Our method outperforms all the other state-of-the-art learning-based approaches, as well as the non-learning RGB-D VO, and ICP methods, from small baselines to large baselines. Compared with all ablation variants, our full version (F+U+P) achieves the best performance. The addition of uncertainty estimation complements the high-dimensional feature-metric alignment to improve the tracking accuracy. The predicted initial pose further improves the accuracy by bringing the estimation close the correct minimum, especially in the large motion scenarios. After fine-tuning the probabilistic combination with ICP loss, it can be seen that the performance is further improved in most cases (except KF 4 where the performance drops a bit), showing the validity of the probabilistic combination.

We have further developed a prototype visual odometry system, where the camera pose is estimated by our proposed method. Figure 4.7 shows the resulting trajectories and point clouds back-projected from all frames in the sequence from our test split of the TUM RGB-D dataset [Sturm et al., 2012].

Despite being a pure frame-to-frame tracking system without components of

Method	3D EPE (cm) / RPE translation (cm) / RPE rotation (Deg)			
	KF 1	KF 2	KF 4	KF 8
ICP [Rusinkiewicz and Levoy, 2001]	2.53/1.25/0.75	5.12/2.57/1.47	13.21/5.73/3.70	28.80/10.54/7.89
RGB-D VO [Steinbrücker et al., 2011]	2.31/1.03/0.55	4.38/2.81/1.39	12.67/5.95/3.99	31.13/13.83/9.20
Coarse-to-fine PoseNet [Zhou et al., 2018a]	1.88/1.91/0.80	3.08/3.76/1.42	5.82/7.30/2.76	15.43/13.16/5.73
Iterative PoseNet [Zhou et al., 2018a, Li et al., 2018c]	1.76/1.86/0.84	2.70/3.61/1.53	4.75/7.28/2.73	12.74/13.12/5.23
DeepIC [Lv et al., 2019]	1.31/0.69/0.45	1.57/1.14/0.63	2.53/2.09/1.10	11.03/5.88/3.76
DeepIC+P, adapted from [Lv et al., 2019]	1.26/0.69/0.44	1.46/1.13/0.60	2.32/2.68/1.10	8.20/5.06/3.73
Ours (F)	1.25/0.67/0.44	1.49/1.14/0.60	2.50/2.78/1.14	11.70/12.20/4.37
Ours (F+P)	1.24/0.65/0.44	1.42/1.04/0.57	2.04/2.06/0.81	7.35/6.71/2.89
Ours (F+U)	1.23/0.58/0.41	1.40/0.86/0.50	2.33/1.99/0.87	13.24/12.92/4.59
Ours (F+U+P)	1.23/0.57/**0.40**	1.38/0.80/0.48	**1.71**/**1.22**/**0.64**	5.48/4.89/2.12
Ours+ICP	**1.22**/**0.54**/**0.40**	**1.33**/**0.76**/**0.47**	1.78/1.26/0.66	**4.82**/**4.57**/**2.00**

Table 4.1: Results on our test split in TUM RGB-D Dataset. KF denotes the frame intervals.



(a) fr1_desk (b) fr1_360

(c) fr2_desk (d) fr2_360

Figure 4.7: Trajectories delivered by our system on test split of TUM RGB-D dataset. We back-projected point clouds from all frames to visualise the alignment.

keyframing and loop closure optimisations, drift caused by incremental misalignment qualitatively remains small. The qualitative results can be found in the supplementary video.

MovingObjects3D Dataset: MovingObjects3D dataset contains 6 different categories of objects moving in front of the camera under various illumination changes. We follow the dataset setting, where the categories of 'boat' and 'motorbike' are used as the testing set and the other categories are split into training (first 95% sequences of each category) and validation (last 5%), to test tracking performance for unseen objects. For the non-learning-based ICP [Rusinkiewicz and Levoy, 2001] approach, we provide ground truth object masks for them to test their optimal performances. For the learning-based approaches, we reply on those systems to distinguish the object motion from the background, given the ground truth object and camera motions. Table 4.2 reports the results, which again show the superior performance of our method and confirm the contribution of each proposed component.

Figure 4.8 visualises our tracking result on the test split of MovingObjects3D dataset. As can be seen, our proposed method can provide a good alignment for objects under large motion and lighting changes. A combination with ICP can provide a further refinement in the pose estimation.

Ablation Study on the Choice of Channel Dimension: As examined in [Czarnowski et al., 2017], multi-dimensional feature map from network can improve tracking robustness. In Table 4.1 and Table 4.2, **Ours (F)**, with higher-dimension features, outperforms [Lv et al., 2019] in most cases, even without uncertainty or pose predictions. On the other hand, a higher dimension of feature maps usually bring a higher computational cost. In this part, we experimentally evaluate the effect of the channel dimension of the feature map and the uncertainty map. We fix the uncertainty channel to be 1 when we vary the feature channels and fix the feature channel to be 8 when we vary the uncertainty channels between 1 and the same feature channel, i.e. 8. Table 4.3 summarises accuracy and inference time on the TUM RGB-D dataset [Sturm et al., 2012]. Note that the accuracy increases when we increase the channel dimension of feature map, albeit with diminishing gains at dimensions higher than 8. When we increase the channel dimension of the uncertainty map, the accuracy very slightly increases for small baselines and slightly decreases for large baselines, validating the original choice of scalar uncertainty prediction.

Method	3D EPE (cm) / RPE translation (cm) / RPE rotation (Deg)		
	KF 1	KF 2	KF 4
ICP [Rusinkiewicz and Levoy, 2001]	3.31/9.75/**2.74**	9.63/19.72/8.31	19.98/41.40/16.64
Coarse-to-fine PoseNet [Zhou et al., 2018a]	2.62/10.10/4.14	5.01/20.19/8.29	9.63/38.96/16.02
Iterative PoseNet [Zhou et al., 2018a, Li et al., 2018c]	2.55/10.08/4.14	4.96/20.16/8.28	9.60/38.91/16.00
DeepIC [Lv et al., 2019]	2.91/9.73/3.74	5.94/19.60/7.41	12.96/38.39/14.71
DeepIC+P, adapted from [Lv et al., 2019]	2.66/9.78/3.76	5.14/19.72/7.67	9.90/38.50/15.17
Ours (F)	2.52/9.34/3.57	5.04/18.90/7.26	10.49/37.19/14.39
Ours (F+P)	2.64/9.59/3.64	5.14/19.42/7.43	9.97/37.01/14.32
Ours (F+U)	2.20/8.62/3.43	4.53/17.90/7.19	9.86/36.18/14.50
Ours (F+U+P)	2.17/8.44/3.22	4.47/17.86/6.91	9.26/36.443/14.22
Ours+ICP	**1.93**/**7.84**/2.93	**4.12**/**16.94**/**6.29**	**8.93**/**35.39**/**13.14**

Table 4.2: Results on our test split of MovingObjects3D Dataset.

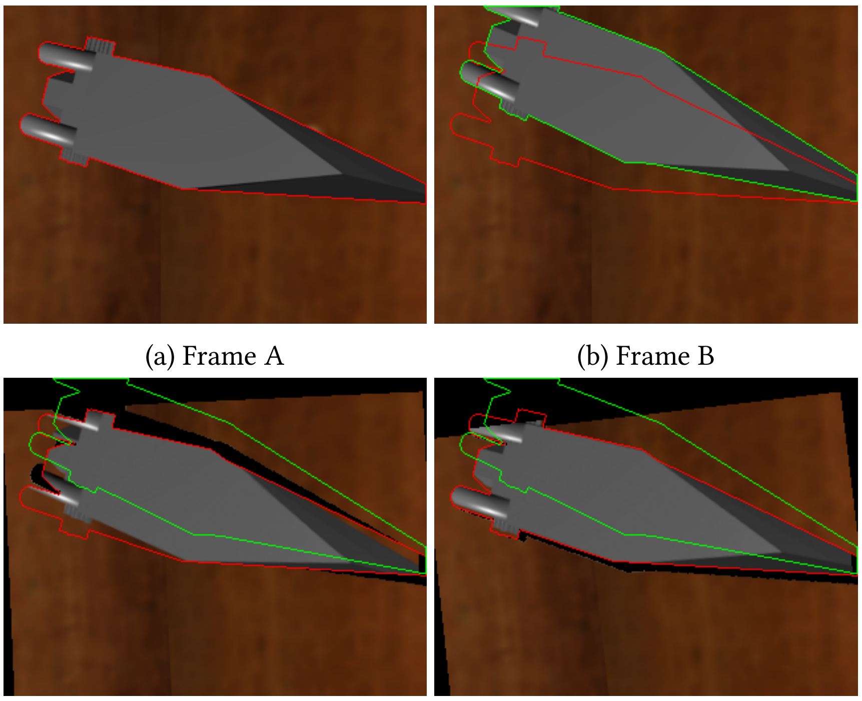


Figure 4.8: Qualitative results on MovingObjects3D dataset. Object motion between the frame A and frame B is estimated using our proposed method (c) and a further combination with ICP (d). The object is warped from frame A to B using the estimated motion for visualization. The ground truth object boundaries in *A* and *B* are colored in red and color, respectively. Black regions in the warped image are caused by occlusion.

In addition to accuracy, the increase of channel dimension in either feature or uncertainty map dimension would increase the GPU memory usage and reduce the inference speed. As a compromise of all these factors, we choose the feature dimension to be 8 and the uncertainty dimension to be 1 in all our other experiments.

Model Size and Computation Time: Our system implemented in PyTorch has 1.83M learnable parameters. The average forward inference time for a pair of RGB-D image in the resolution of 160×120 on a GTX 1080 platform is 7.29ms. After integrating ICP, it is 9.84ms (i.e. +35%) on the same platform. Our network has a larger model size than DeepIC, which has 662K learnable parameters, but shows a comparable inference speed to its 7.6ms inference time on the same platform.

Map	C	3D EPE (cm) / RPE translation (cm) / RPE rotation (Deg)				Time
		KF 1	KF 2	KF 4	KF8	(ms)
	1	1.23/0.58/0.41	1.37/0.83/0.50	1.86/1.48/0.74	8.15/6.09/2.93	5.41
F	3	1.23/**0.57**/**0.40**	1.36/**0.78**/**0.48**	1.72/1.24/0.64	5.92/5.05/2.20	6.25
U=1	8	1.23/**0.57**/**0.40**	1.38/0.80/**0.48**	1.71/1.22/0.64	**5.48**/**4.89**/**2.12**	7.29
	16	**1.22**/**0.57**/**0.40**	**1.35**/**0.78**/**0.48**	**1.66**/**1.21**/**0.62**	5.72/4.94/2.22	11.67
U	1	1.23/0.57/**0.40**	1.38/0.80/**0.48**	**1.71**/**1.22**/**0.64**	**5.48**/**4.89**/**2.12**	7.29
F=8	8	**1.22**/**0.55**/**0.40**	**1.37**/**0.79**/0.49	1.74/1.35/0.67	6.15/5.58/2.38	9.13

Table 4.3: Ablation study of the channel dimension effect on our test split in TUM RGB-D. F, U, C abbreviate the feature map, uncertainty, and the channel dimension. Time is the average inference time for a pair of input RGB-D images (size 160×120).

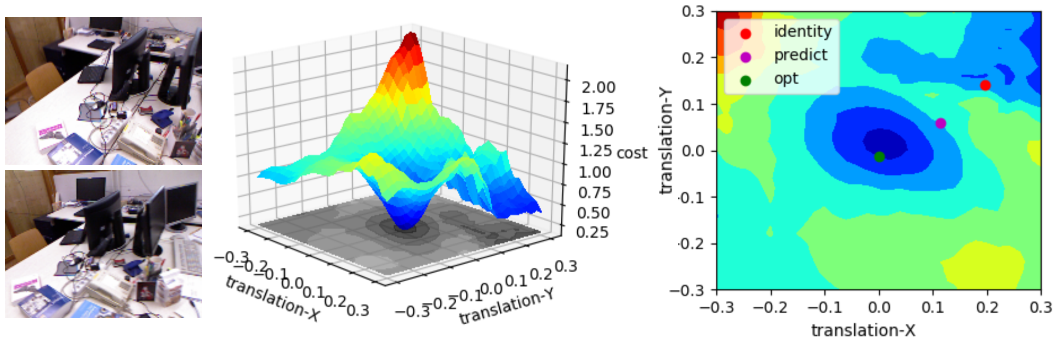


Figure 4.9: Visualisation of cost landscape of x and y translation for the feature-metric loss on the coarsest level. From left to right: input, cost landscape 3D, and 2D projection of cost landscape.

Compared to the classic approaches implemented in Open3D [Zhou et al., 2018b], our network is also faster than the point-to-plane ICP [Rusinkiewicz and Levoy, 2001] (310ms). However, we would also expect to see a significant boost to the ICP inference time when it is efficiently implemented in CUDA [Newcombe et al., 2011a] or CPU [Vespa et al., 2018].

We also studied the effect of the input image resolution. With increased resolution (256×192), accuracy slightly improves on the small baselines, i.e. KF 1 and 2, however, slightly deteriorates on KF 4 and 8 while the computation increases to 15.29ms (i.e. +111%). Therefore, we set 160×120 as main setting for training and testing.

4.4.2 Qualitative Evaluation and Discussions

Convergence Basin: To analyse the effect of the initial pose prediction in our system, we perform a cost landscape visualisation experiment. Since ξ is a 6D vector, it is computationally infeasible to sample cost on all possible pose components and also difficult to visualise the 6D cost landscape. Therefore, we choose to fix the rotation and z-translation components and only sample the pose combinations at the x and y translations around the ground truth pose. Figure 4.9 shows one example on our test split from the TUM RGB-D dataset using the an interval of

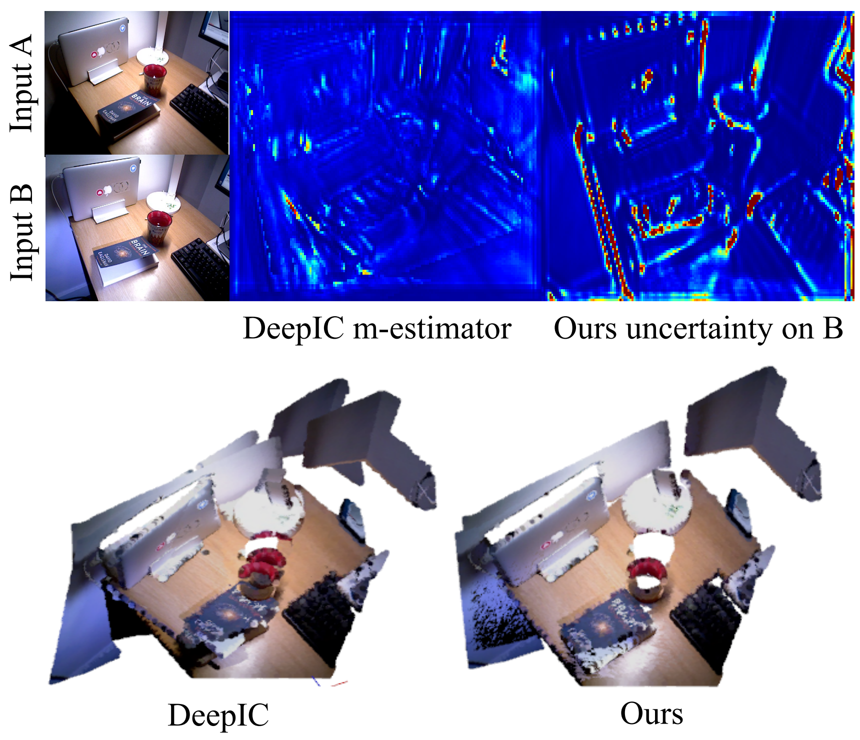


Figure 4.10: Qualitative evaluation in challenging lighting. Notice our uncertainty estimation is more sensitive to the lighting changes than the learned m-estimator in DeepIC (higher value is in red and lower value is in blue).

8 frames. It can be seen that our pose prediction network brings the estimation into the convergence basin near the global minimum otherwise the conventional identity pose initialisation would lead the optimisation to a wrong local minimum.

Challenging Illuminations: Uncertainty prediction is significant for deploying neural network on robotic applications. DeepIC [Lv et al., 2019] proposed a learned robust cost function m-estimator to downweigh the residual outliers. To evaluate our learned uncertainty and also to compare to DeepIC's learned m-estimator, we captured sequences using an RGB-D camera while we were waving a flashlight to create illumination changes. The collected sequences contain both local and global lighting, reflection, and shading variances across the images. Since we don't have ground truth poses on these frames, we warp the point cloud from one frame to another using the estimated transformation between them and visualise the 3D pointcloud alignment of the two views. We test it using the weights trained from the TUM RGB-D dataset without fine-tuning. Figure 4.10 shows one example. It can be seen that our method provides more robust pose estimation under those lighting changes. This is partially because our estimated uncertainty can more reliablely capture illumination variance, e.g. on the book and desk surface, than DeepIC's m-estimator. Please refer to the supplementary video for more results and details.

4.5 Conclusion and Discussions

We presented a deep probabilistic feature-metric two-frame RGB-D tracking method by combining the power of deep learning for feature learning, uncertainty estimation and pose prediction in a learning-based optimisation framework. It makes our method compact and outperform the state of the art methods on camera motion and rigid object motion estimation benchmarks. Challenging experiments have shown an accurate and robust performance under large motion and strong lighting change scenarios, which is significant and currently lacking, in real-world robotic applications. We further showcased how our proposed residual can easily be combined with commonly used ICP residual in practice.

A strong assumption in this work is that the geometric information from the depth sensor is mostly stable and accurate. The geometric information has been exploited in the feature descriptor generation, warping function, and also the combined geometric residual. An important future direction will be to explore how to predict the uncertainties in the geometric component and then formulate them in the proposed feature-metric residual. Continuing from here, we would also like to explore how to better combine the probabilistic feature-metric residuals with other residuals.

In the training step, the unrolled optimisation architecture implemented a fixed number of update steps at each optimisation level. We assumed that the optimisation should also converge after 3 update steps at each level in the test time. Although this is a widely adopted practice, for example as in [Lv et al., 2019, Bloesch et al., 2019], and we verified that using an adaptive number of update steps in the test time does not necessarily improve the estimation results, an extra evaluation and proof on the convergence guarantee could be possible in future work.

We would also like to highlight that our system is trained end-to-end in a self-supervised way without explicitly defining the ground-truth uncertainties or feature maps. This allows flexibility to further improve our performance in solving difficult conditions, for example, lighting changes or occlusions, when more targeted training data is given.

Also, we aim to apply our tracking method to full dense SLAM systems, including object-level and dynamic SLAM systems. The frame-to-frame tracking designed in our prototype visual odometry system in Figure 4.7 inevitably accumulates drift. Possible solutions would be to extend our proposed feature-metric residual Section 4.3.1 to keyframe-based SLAM systems or to fuse the depth into a global frame and conduct frame-to-model tracking to reduce drift.

In the next chapter, we will explore how to utilise the shape prior information in object-level SLAM systems, in addition to the multi-view constraint. We will focus on the improvement of object mapping by taking advantage of the category-level shape priors and show that object shape completion can also improve the tracking accuracy of moving objects.

Chapter 5

Object-level Dynamic SLAM with Map Completion

In this chapter, we propose a novel object-level dynamic SLAM system that can simultaneously segment, track, and reconstruct objects in dynamic scenes. It can further predict and complete their full geometries by conditioning on reconstructions from measured depth and a category-level shape prior with the aim that completed object geometry leads to better object reconstruction and tracking accuracy. For each incoming RGB-D frame, we perform instance segmentation to detect objects and build data associations between the detection and the existing object maps. A new object map will be created for each unmatched detection. For each matched object, we jointly optimise its pose and latent geometry representations using geometric residual and differential rendering residual towards its shape prior and completed geometry. Our approach shows better tracking and reconstruction performance compared to methods using traditional volumetric mapping or learned shape prior approaches. We evaluate its effectiveness by quantitatively and qualitatively testing it in both synthetic and real-world sequences.

Contents of Chapter

This chapter is currently under submission.

5.1 Introduction

Simultaneous Localisation and Mapping (SLAM) research aims to concurrently estimate both the scene geometry of the unknown environment as well as the robot pose within it from the data of its on-board sensors only. It has rapidly progressed from sparse SLAM [Davison et al., 2007, Klein and Murray, 2007] into dense SLAM [Newcombe et al., 2011a, Vespa et al., 2018], and recently into semantic object-level SLAM [McCormac et al., 2017, McCormac et al., 2018]. This fast-evolving research has enabled many robotic applications. Despite this, most SLAM research still assumes a static scene, where points in the 3D world maintain constant spatial positions in a global coordinate. Any information violating this assumption, such as moving objects in the environment, would be treated as outliers and are intentionally ignored in tracking and mapping steps.

This setup, however, can only handle a small amount of dynamic elements, excluding itself from many real-world applications as environments, particularly where humans are involved, are continually changing. A robust SLAM system capable of handling highly dynamic environments, therefore, is desirable. Most current dynamic SLAM research can be categorised into three main directions. One maps the whole changing world in a non-rigid deformable representation to deal with the changing topology of deformable/moving objects [Newcombe et al., 2015]. The second aims at improving the robustness and accuracy of camera tracking by ignoring all possibly moving objects and building a single static background model [Jaimez et al., 2017, Scona et al., 2018, Bescós et al., 2018]. The third models dynamic environments by creating object-centric maps for each possibly moving object in the scene while fusing corresponding information into these object-level maps [Rünz and Agapito, 2018, Xu et al., 2019]. Object-level tracking and mapping can be conducted for each object and camera motion against the static part of the map. This paper aligns with the last direction as we believe that, similar to

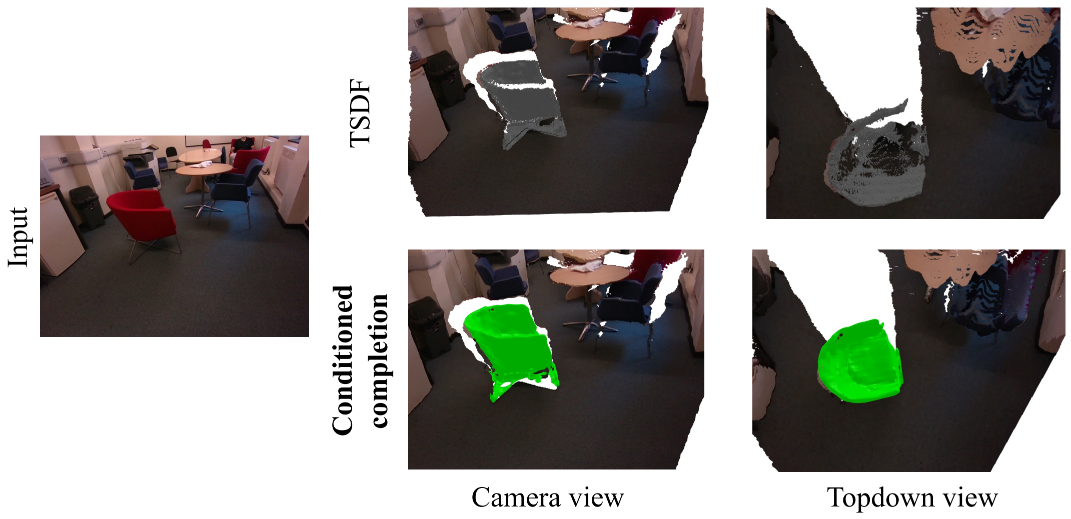


Figure 5.1: Given RGB-D images, our system builds object-level dense dynamic maps that can robustly track camera pose and object poses while completing the missing sensor information using object priors. Compared to the classic TSDF maps, our object maps fill in unobserved parts and their latent codes can be optimised jointly with object poses. Interfered regions by humans can be detected and intentionally removed in the system. The background pointclouds are projected for pure visualisation purpose.

human perception, an instance-awareness of the surrounding environment can help intelligent robots perceive the scene changes and enables meaningful interactions with the surrounding environment.

By far most existing object-level dynamic SLAM systems mentioned above adopt the classic map representation that have been exploited in the static SLAM systems, such as pointclouds [Bescós et al., 2021], surfels [Rünz and Agapito, 2018] or volumetric maps [Xu et al., 2019]. This leads to partial or incomplete object maps as only the observable information can be fused into the object models. Information in unseen parts can not be filled unless an object or the sensor is moved actively. Contrary to reconstructing objects from scratch, some works recently explored learning-based category-level object shape priors and build object-level maps based on learned shape priors [Sucar et al., 2020, Wang et al., 2021]. The object geometry and pose are usually optimised via differentiable rendering. However, most of these systems are only applicable in static scenes. Besides, despite being able to generate complete object geometry, object shape priors cannot capture complex geometry details as the bottleneck of its latent representation can only interpolate shapes inside the training datasets [Park et al., 2019]. When combined with dense image alignment, such as photometric or ICP residuals, this inconsistency between the measurement and the object prior model inevitably leads to inaccurate object motion trajectory estimates.

This work stands in the middle between reconstructing object geometry from scratch and mapping using object shape priors. We reconstruct the observable part by continuously fusing depth measurements into a volumetric canonical space and predict the complete geometry by conditioning it on the fused volume. The resulting object geometry can preserve the details that have been observed in the past and simultaneously complete the missing geometry information. We also verified that this completed object geometry can further improve the accuracy of object tracking. The main contributions in this chapter can be summarised as follows:

1. we present, to the best of our knowledge, the first RGB-D object-level dynamic SLAM system that can complete unseen parts of objects using a shape prior encoded in neural networks while still reconstructing observed object parts accurately;
2. a joint optimisation of object pose and shape geometry based on geometric residuals and differentiable rendering;
3. extensive experiments of object tracking and reconstruction components on synthetic and real-world data to evaluate the benefits of object geometry completion for object-level SLAM.

The remainder of this chapter is organised as follows. Section II discusses the related work in greater detail, highlighting our originality. Section III explains the details in our system. Section IV describes the experiments and the comparisons with the existing methods, followed by the conclusion in the last section.

5.2 Related Works

Object-level dynamic SLAM Although object-level dynamic SLAM research can be dated back to [Wang et al., 2003], visual dense object-level dynamic SLAM has only been explored recently. From RGB-D sensor inputs, Co-Fusion [Rünz and Agapito, 2017] segments objects by either ICP motion segmentation or semantic segmentation and then tracks objects separately based on ElasticFusion [Whelan et al., 2016]. MaskFusion [Rünz and Agapito, 2018] segments objects using a combination of instance segmentation from Mask-RCNN and geometric edges, and tracks objects using the same approach as Co-Fusion. Both Co-Fusion and MaskFusion use surfels to represent map models, which is memory efficient but cannot directly provide free space information in the map, and neither surface connectivity. DynSLAM-II [Bescós et al., 2021] extends from ORB-SLAM II [Mur-Artal and Tardós, 2017] and formulates tracking using sparse feature descriptor matching. Object maps are represented using clusters of pointclouds, which can bring object poses and geometries into the pose graph optimisation but also lack space connection awareness. Instead, MID-Fusion [Xu et al., 2019] uses memory-efficient octree-based volumetric signed distance field (SDF) representation for objects and re-parametrises tracking residuals in object coordinates and weights. EM-Fusion [Strecke and Stuckler, 2019] similarly uses volumetric SDF object maps but formulates object tracking as direct alignment of input frames with the SDF representations. Their following work [Strecke and Stuckler, 2020] infers the missing object geometry by penalising the hull and intersection constraints. However, it did not explore shape prior information and requires heavy computation to optimise SDF field explicitly. Instead, we fuse the depth measurement into object-level SDF maps and predict completed object geometries by incorporating a shape prior in continuous occupancy fields.

SLAM with shape prior maps Instead of estimating both object geometry and poses from scratch, some approaches use a shape prior to represent objects. Since the coordinates of object shape priors and the run-time measurement are not necessarily aligned, a relative rigid transformation needs to be estimated. This is analogous to the localisation-only problem in SLAM. SLAM++ [Salas-Moreno et al., 2013] is one of the pioneering object-level RGB-D SLAM systems. It scanned objects in advance and then maps the detected instances at run-time by jointly optimising a pose graph of camera and object poses. Relying on pre-scanned objects, however, limits its ability to scale to unknown object models. Rather than using instance-level shape priors, several following works exploited category-level shape priors as there is limited variance in certain object categories. The category-level shape prior can be represented in PCA models as in DirectShape [Wang et al., 2020], in occupancy grids as in Deep-SLAM++ [Hu et al., 2019], in variational autoencoders as in NodeSLAM [Sucar et al., 2020], in autodecoders as in DSP-SLAM [Wang et al., 2021], or in mesh generation networks as in [Lin et al., 2019]. However, most of these works only target static environments, as multi-view consistency of static world points is required to localise the shape prior models. [Li et al., 2021] relax this restriction using a Bayesian filter to associate object detections on different frames and fuse the prior model by simply averaging the latent codes from each frame. However, object shape deviations cannot always be captured by the shape prior interpolation. The object tracking accuracy would be affected by the discrepancy between the prior shape model and the online measurement. We address this challenge by conditioning the completion network on the fused reconstruction model.

Object-level tracking To track moving objects in RGB-D sequences, several pioneering object-level works adopt the frame-to-model tracking methods from RGB-D SLAM systems [Rünz and Agapito, 2017] and parametrise them for object tracking [Xu et al., 2019, Bescós et al., 2021]. The classic photometric residual, however, is difficult to deal with as object lighting changes; several approaches explore using learning-based robust features to formulate object tracking in direct [Xu et al., 2021a] and in-direct ways [Wen and Bekris, 2021]. Parallel to learning category-level shape priors, Wang et al. [Wang et al., 2019] proposed to learn category-level pixel-wise correspondence from RGB-D images to the canonical space. The shape is implicitly defined from this correspondence and the frame-to-canonical transformation can be estimated from this noisy correspondence. Rempe et al. [Rempe et al., 2020] further proposed to generate more stable correspondences by accumulating temporal information from RGB-D sequences. Recently, Muller et al. [Muller et al., 2021] proposed to track moving objects and predict their complete geometry using such canonical correspondence representation. In this work, the object pose is initially predicted using canonical correspondence regression. However, we found it does not necessarily yield alignment to the canonical space and we further optimise the pose tracking using geometric residual and differential rendering.

5.3 Method

5.3.1 Notations and Preliminaries

In this work, we will use the following notation: a coordinate frame is denoted as $\underrightarrow{\mathcal{F}}_A$. The homogeneous transformation from $\underrightarrow{\mathcal{F}}_B$ to $\underrightarrow{\mathcal{F}}_A$ is denoted as $\boldsymbol{T}_{AB}$, which is composed of a rotation matrix $\mathbf{C}_{AB}$ and a translation vector ${}_A\mathbf{r}_{AB}$.

Every detected object is represented in its individual object coordinate frame $\underrightarrow{\mathcal{F}}_{O_n}$, with $n \in \{0 \ldots, N\}$, where N is the total number of objects (excluding background)

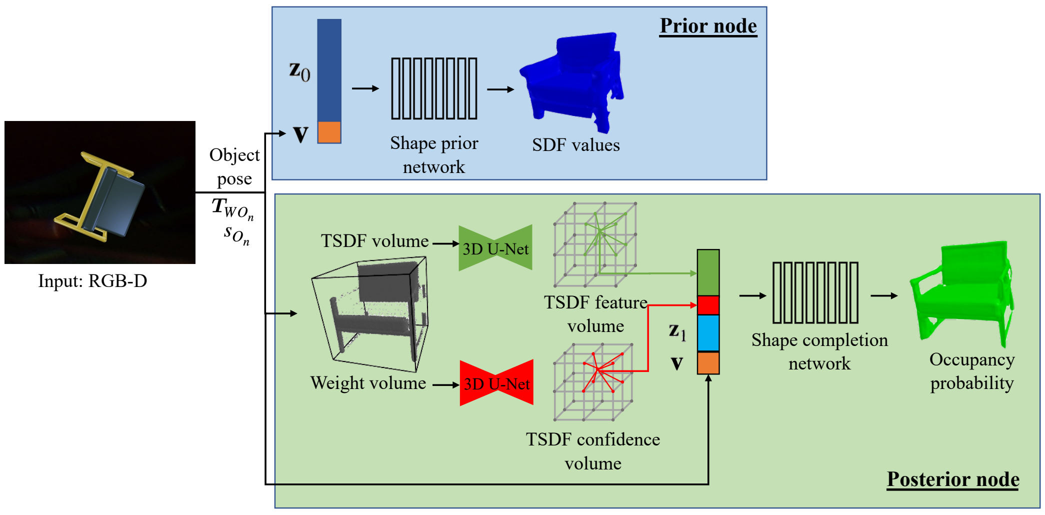


Figure 5.2: The overview of our object geometry representation.

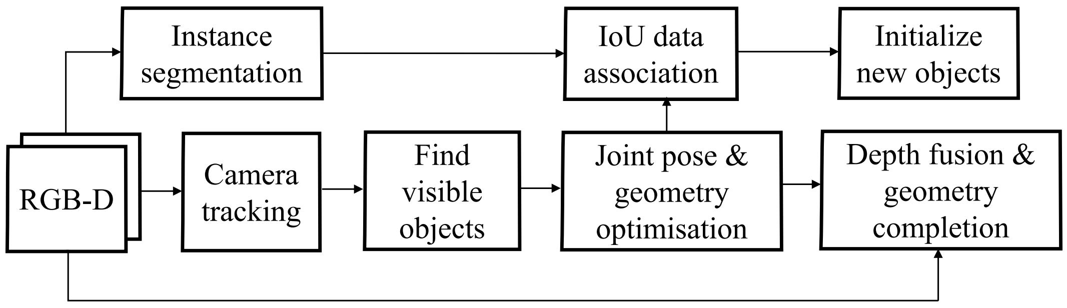


Figure 5.3: The pipeline of the proposed method

and 0 denotes background. We assume a canonical static volumetric model is stored in each object coordinate frame, forming the basis of our multi-instance SLAM system. To leverage the category-level shape prior, we need to align the canonical space with the one defined in training, otherwise the completion performance will be deteriorated since it cannot fully take advantage of the shape variances of the objects in the same category. The relative transformation between the current world coordinate and the corresponding object canonical space is defined as a joint state composed of a rigid transformation T_{WO_n} and the object scale s_{O_n}. We define the object pose using this joint state. T_{WO_n} needs to be continuously updated for a moving object but the object scale should be consistent across different frames.

5.3.2 System Overview

Figure 5.3 shows the pipeline of our proposed system. Each input RGB-D image is processed by Mask R-CNN to perform instance segmentation. The camera pose is tracked against background regions, excluding the human mask area and moving objects, similar to what has been proposed in MID-Fusion [Xu et al., 2019].

The object geometry is composed of two nodes with a shared object pose: prior node and posterior node, as shown in Figure 5.2. The prior node represents its category-level shape prior using a latent code $\mathbf{z}_0 \in \mathbb{R}^{64}$. It can be used to express the continuous SDF field s on any query 3D location in object canonical coordinate $\mathbf{v} \in \mathbb{R}^3$ using a DeepSDF shape prior network F_0 [Park et al., 2019] as

$$s = F_0(\mathbf{v}, \mathbf{z}_0). \tag{5.1}$$

The prior node is used to initialise the object pose and re-localise the object model when object tracking is lost as the prior geometry does not degrade even if the object pose deviates from the canonical space. The posterior node encodes a fused partial TSDF volume and its associated TSDF weight volume into TSDF feature volume θ_t and TSDF confidence volume θ_c separately using 3D-UNet [Çiçek et al., 2016]. Then a complete occupancy field o can be predicted using a shape completion network F_1:

$$o = F_1(\mathbf{v}, \theta_t[\mathbf{v}], \theta_c[\mathbf{v}], \mathbf{z}_1). \tag{5.2}$$

where $\theta_t[\mathbf{v}] \in \mathbb{R}^{32}$ and $\theta_c[\mathbf{v}] \in \mathbb{R}^1$ denote the feature vectors tri-linearly interpolated at the point $\mathbf{v}$ inside the volumes θ_t and θ_c, respectively. We additionally condition it on a latent code $\mathbf{z}_1 \in \mathbb{R}^{32}$ so that the hidden space can be optimised to generate novel shapes, as shown in Figure 5.4. The shape completion network F_1 has a similar architecture to the CONet [Peng et al., 2020] with an extra input of TSDF confidence weight to balance the depth measurement and network prediction and a instance-level latent code in the input to optimise the hidden space.

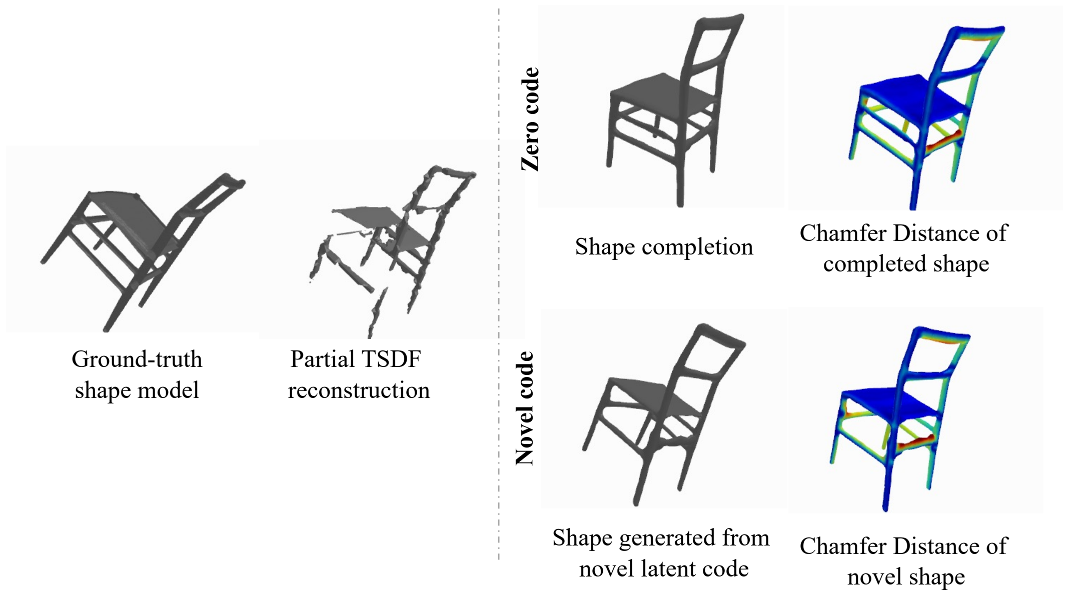

Ground-truth shape model
Partial TSDF reconstruction
Zero code
Shape completion
Chamfer Distance of completed shape
Novel code
Shape generated from novel latent code
Chamfer Distance of novel shape

Figure 5.4: Editing the conditioned latent code can change the geometry of the unobserved part in the object model.

We perform an efficient Axis Aligned Bounding Box (AABB) ray intersection test [Majercik et al., 2018] to find all the visible existing object models in the current viewpoint and render object masks for each visible models. An Intersection of Union (IoU) between the detections on the current frame and the rendered model masks is computed to build data associations between the current frame detections and existing object models. Then we track each object model and complete its geometry by performing a joint optimisation of pose and geometry (Section 5.3.3). Using estimated poses of the camera and objects, new depth measurements are fused into an object model and a complete shape geometry can be predicted by conditioning on the fused model. New objects are created for unmatched detections by initialising their initial object pose using object prior models.

5.3.3 Joint Optimisation of Object Pose and Geometry

Instead of defining an arbitrary canonical space for object coordinates [Xu et al., 2019], we need to estimate the 7DoF relative transformation, which is composed of

$\boldsymbol{T}_{WO_n}$ and s_{O_n}, to align the initialised object coordinate to the training canonical space for each object detection in order to take advantage of the learned prior information.

Initialisation Given an RGB-D frame (I_L, D_L) and detected object mask M_n, we predict their positions in the canonical space $\mathbf{v}$ and associated confidences w from back-projected pointcloud using a modified normalised object correspondence network F_n from [Rempe et al., 2020]:

$$_{C_L}\mathbf{v}(\boldsymbol{u}_L) = \pi^{-1}(\boldsymbol{u}_L, D_L[\boldsymbol{u}_L]), \forall \boldsymbol{u}_L \in M_n, \tag{5.3}$$

$$F_n\left(_{C_L}\mathbf{v}\right) \longrightarrow \mathbf{v}, w \tag{5.4}$$

Then we solve the 7-DoF relative transformation, scale s_{O_n}, rotation $\mathbf{C}_{C_L O_n}$, and translation $_{C_L}\mathbf{r}_{C_L O_n}$ from the predicted correspondence using the Umeyama algorithm [Umeyama, 1991] with SVD decomposition:

$$\underset{s_{O_n}, \boldsymbol{T}_{C_L O_n}}{\operatorname{argmin}} \sum_{\boldsymbol{u}_L \in M_n} w \left(\mathbf{v} - \frac{1}{s_{O_n}} \boldsymbol{T}^{-1}_{C_L O_n} \, {}_{C_L}\mathbf{v}(\boldsymbol{u}_L) \right). \tag{5.5}$$

For the latent codes $\mathbf{z}_0$ and $\mathbf{z}_1$, we use both zero code initialisations. We only run this pose initialisation step for new unmatched object detections. The initial pose solved from SVD decomposition, however, is necessary to be close to the ground-truth canonical pose, due to the unseen shapes or viewpoints, affecting the performance of shape completion.

Coarse Estimation To refine object canonical poses from the initial pose prediction, we jointly optimise it with the prior latent code $\mathbf{z}_0$ to minimise the 3D SDF loss E_{SDF} and 2D rendering loss E_{render}:

$$E_{\text{coarse}} = \lambda_{\text{s}} E_{\text{SDF}} + \lambda_{\text{r0}} E_{\text{render}} + \lambda_{\text{z0}} ||\mathbf{z}_0||. \tag{5.6}$$

The 3D SDF loss is defined to encourage the back-project depth points to align with the object surface, where the zero SDF value is defined

$$E_{\text{SDF}} = \sum_{\boldsymbol{u}_L \in M_n} F_0 \left(\frac{1}{s_{O_n}} \boldsymbol{T}^{-1}_{C_L O_n} {}_{C_L}\mathbf{v}(\boldsymbol{u}_L), \mathbf{z}_0 \right). \tag{5.7}$$

We cannot compute SDF residuals for empty space since ground-truth SDF values are not available at test time. Instead, for the non-surface areas, we use differentiable rendering to encourage the rendered depth D_L to be close to the measured depth $\tilde{D}_L$. We compute the rendering loss for the visible 3D space inside the object 3D bounding box:

$$E_{\text{render}} = \sum_{\boldsymbol{u}_L \in B_n} D_L[\boldsymbol{u}_L] - \tilde{D}_L[\boldsymbol{u}_L], \tag{5.8}$$

where

$$D_L[\boldsymbol{u}_L] = \sum_{i=1}^{N} w_i d_i, \tag{5.9}$$

and w_i is the ray-termination probability [Sucar et al., 2020] of sample i at depth d_i along the ray from the pixel $\boldsymbol{u}_L$,

$$w_i = o_i \prod_{j=1}^{i-1} (1 - o_j), \tag{5.10}$$

and B_n is the 2D bounding box rendering from the estimated object 3D bounding box on this frame. A continuous occupancy field can be extracted from the continuous prior SDF field as proposed in [Wang et al., 2021]:

$$o_i = -\frac{1}{2\sigma} F_0 \left(\frac{1}{s_{O_n}} \boldsymbol{T}^{-1}_{C_L O_n} \left(\pi^{-1}(\boldsymbol{u}_L, d_i) \right), \mathbf{z}_0 \right), \tag{5.11}$$

where σ is the truncation distance to control the transition.

By freezing the network weight in F_n, the cost function in Equation (5.6) can be iteratively solved using Gauss-Newton optimisation with analytical Jacobians. Since the prior shape is not necessarily aligned with the actual observation, it is unnecessary to sample every pixel ray. Instead, we run this optimisation on sparse ray samples, which can speed up the optimisation without losing much accuracy.

Dense Refinement To further align the object pose with the measurement and to complete the hidden part, we jointly optimise the posterior latent code $\mathbf{z}_1$ with the object pose by minimizing the 3D occupancy loss E_{occ} on both occupied and empty space (excluding the unknown 3D space) and a similar 2D rendering loss E_{render}:

$$E_{\text{refine}} = \lambda_{\text{o}} E_{\text{occ}} + \lambda_{\text{r1}} E_{\text{render}} + \lambda_{\text{z1}} \|\mathbf{z}_1\|. \tag{5.12}$$

The occupied space is defined on the back-projected points and the empty space is uniformly sampled in the background space as well as the foreground space before the depth measurement. The occupancy loss is defined using the binary cross-entropy loss between the predicted occupancy value $o_{\mathbf{v}}$ using Equation (5.2) and the ground-truth occupancy values $o^*_{\mathbf{v}}$ (0.5 for the occupied space and 0 for the empty space) for sampled points $\mathbf{v}$ inside the occupied and empty space:

$$E_{\text{occ}} = -\sum_{\mathbf{v}} [o_{\mathbf{v}} \log(o^*_{\mathbf{v}}) + (1 - o_{\mathbf{v}}) \log(1 - o^*_{\mathbf{v}})]. \tag{5.13}$$

Similar to the coarse estimation, we can also evaluate the 2D rendering loss using Equation (5.8). The difference is here we use the decoded continuous occupancy value for the sampled d_i using Equation (5.2), instead of converting it from the SDF field in Equation (5.11).

5.3.4 Training Setup

The learnable network parameters in this work includes three part, canonical correspondence network F_n, shape prior network F_0, shape posterior network F_1.

Figure 5.5 shows the architecture of our canonical correspondence network F_n. It takes the partial pointcloud ${}_{C_L}\mathbf{v}$ from the depth measurements as input and predicts its correspondence $\mathbf{v}$ in canonical space and the associated confidence w. We train the canonical correspondence network using the partial pointcloud generated from the synthetic shapenet dataset [Chang et al., 2015]. During training, we augmented

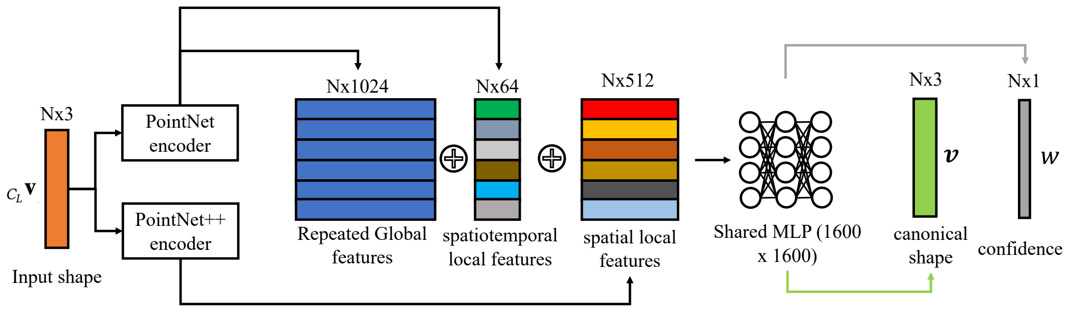


Figure 5.5: The architecture of our canonical correspondence network. The architecture is modified from [Rempe et al., 2020]. It extracts global features and spatiotemporal local features from the PointNet encoder [Qi et al., 2017a] and spatial local features from the PointNet++ encoder [Qi et al., 2017b]. These features are concatenated and passed to an MLP to regress the canonical shape correspondence and the associated confidence.

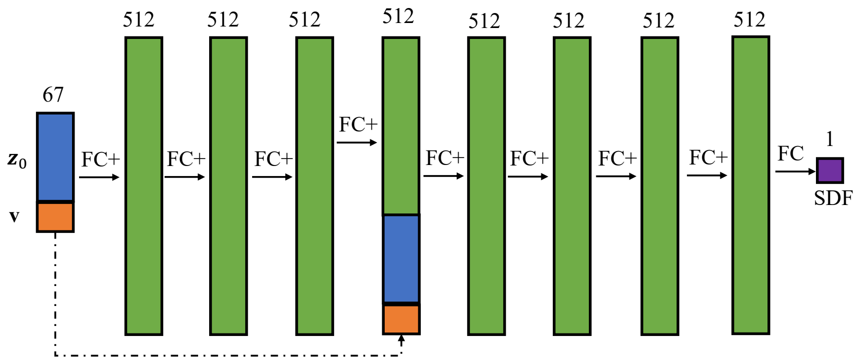


Figure 5.6: The architecture of our shape prior network, adopted from DeepSDF [Park et al., 2019]. The input vector is fed through a decoder, which contains eight fully-connected (FC) layers with one skip connection. FC+ denotes a FC with a following softplus activation and the last FC layer output a single SDF value.

the input pointcloud with random object pose and solve the 7DoF object poses using Equation (5.5). To help network prediction robust to outliers, we also added random depth outliers in the pointcloud generation to learn the correspondence confidence w in a self-supervised way. The solved pose is compared to the augmented ground-truth pose and the whole network is trained end-to-end since the estimation is differentiable.

We use the pre-trained off-shelf network weights for the category-level shape

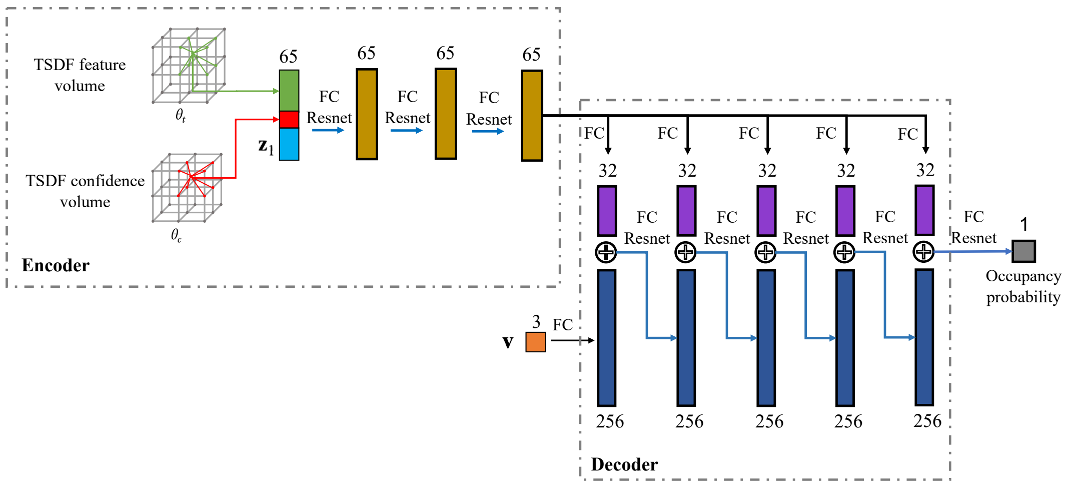


Figure 5.7: The architecture of our shape completion network, modified from CONet [Peng et al., 2020]. The encoder extracts the TSDF feature vector $\theta_t[\mathbf{v}] \in \mathbb{R}^{32}$ and the TSDF confidence vector $\theta_c[\mathbf{v}] \in \mathbb{R}^1$ from TSDF feature volume and TSDF confidence volume, respectively, and concatenates them with a latent code z_1 as an input to the network. It goes through 3 fully-connected (FC) ResNet-blocks to extract local latent features, which are then fed into an occupancy decoder [Mescheder et al., 2019] to predict occupancy probabilities on the position vector $\mathbf{v}$.

prior network F_0 [Park et al., 2019], which was also trained in the shapenet dataset [Chang et al., 2015]. Its architecture is visualized in Figure 5.6.

Figure 5.7 shows the architecture of our posterior shape completion network F_1. It takes the input of a TSDF feature volume and a TSDF confidence volume, which are extracted separately from a partial TSDF volume and its weight volume, and predicts a complete object geometry represented in a continuous occupancy function.

Since the partial observation in reality mostly happens due to self-occlusions and sometimes also due to occlusions from other objects, we rendered depth maps using objects in the shapenet dataset [Chang et al., 2015]. We trained the shape completion network on the rendered depth images to simulate partial depth observations. We use the occupancy loss defined in Equation (5.13) to encourage the completed shape to be similar to the ground-truth one. Similar to the latent code training in

DeepSDF [Park et al., 2019], different object shapes have their own latent codes, which are trained together with the network. We make different partial observations of the same object shape share the same latent code.

5.4 Experiments

5.4.1 Quantitative Reconstruction Evaluations

Experimental Setup

We validate the reconstruction quality of our method on object-level surface reconstruction tasks. We conduct a comparison on the chair category of the ShapeNet [Chang et al., 2015] dataset. The split of train/val/test sets follows the same setting in [Peng et al., 2020]. We randomly select 50 models from the test set to conduct quantitative evaluations. We generate input depth images by rendering images using uniformly sampled virtual camera viewpoints surrounding the CAD model. The hyperparameters used in inference optimization are chosen as $\sigma = 0.05$, $\lambda_s = 100$, $\lambda_{r0} = 2.5$, $\lambda_{z0} = 5$, $\lambda_o = 100$, $\lambda_{r1} = 1$, and $\lambda_{z1} = 1$.

Baseline Methods

To evaluate the object mapping, we compare with the following baseline methods:

- TSDF-fusion: We fuse the depth measurements into a TSDF volume grid as in [Newcombe et al., 2011a].
- DeepSDF mapping: We use the pre-trained decoder weight in [Park et al., 2019]. As the shape completion code is not provided, we optimise the SDF loss on the input pointcloud as well as the empty space constraint proposed in IGR [Gropp et al., 2020].

- CONet: We use the pre-trained network [Peng et al., 2020] and pass the accumulated pointcloud in the canonical space to generate the continuous occupancy field where the meshes are extracted.

Metrics

To quantitatively evaluate the quality and completeness of the shape reconstruction, we use the following metrics:

- IoU: We sample 100k points uniformly in the bounding box and evaluate on both the reconstructed and the ground-truth meshes whether each point is inside or outside. The final value is the fraction of intersection over union. Higher is better.
- Chamfer Distance: we sample 100k points on the surface of both the ground-truth and the reconstructed mesh. We compute the closest points from the reconstructed to the ground-truth mesh using kD-tree and vice-versa. We then compute the average of the L1 distances to the closest points in each direction. Lower is better.
- (In-)completeness: As the completeness of the object map is essential in this work, we also report completeness, which is the one-way chamfer distance from the ground-truth meshes to the reconstructed ones. This is to measure the closest distance from each ground-truth mesh points to the reconstruction. Lower is better.

Results and Discussions

We quantitatively evaluate how the view number of depth measurement would impact the reconstruction results of different methods. Figure 5.8 reports the result. It can be seen that our proposed method consistently show better reconstruction results from single view depth completion to multiple views. When the view number is limited, classic TSDF-Fusion shows worse result as it can only reconstruct the visible parts. CONet completes some missing information, but still struggles as it heavily depends on the input pointcloud. DeepSDF does not condition on the input and the latent code optimisation can fit the few depth measurement and shows better completion and reconstruction results. Our proposed method uses both the input information and shape prior information, yielding best performance. When more depth measurements are received, TSDF-Fusion and CONet start to fill in the missing information while DeepSDF struggles to leverage more measurement information. Our result also improves since we can also take advantage of the measurement information. Figure 5.9 shows an examples of reconstruction results by each method in the view number case of 1, 5, and 10.

5.4.2 Quantitative Tracking Evaluations

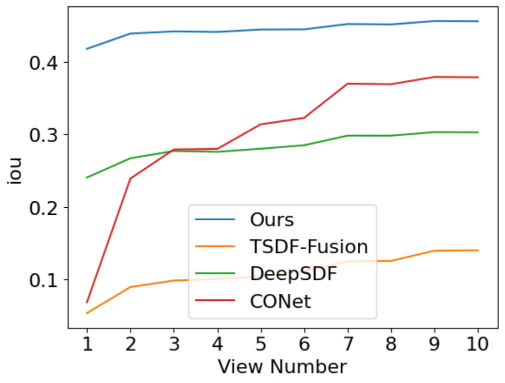
Experimental Setup

To quantitatively evaluate the object-level tracking and mapping performance, we randomly select 10 object models from the test split of the chair category in the ShapeNet [Chang et al., 2015] and render 200 frames using Blender. To ensure diversity of object motion, texture, and illuminations, we randomise four point light sources, camera viewpoint, and object trajectories. We then subsample the sequences using sampling intervals 1, 2, 4 in order to create small, medium and large motion subsets.

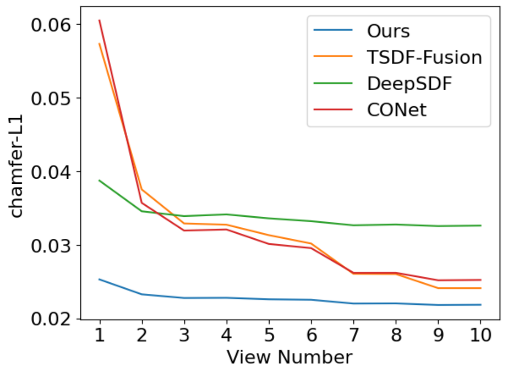
Baseline Methods

To evaluate the object tracking, we compare with the following baseline methods:

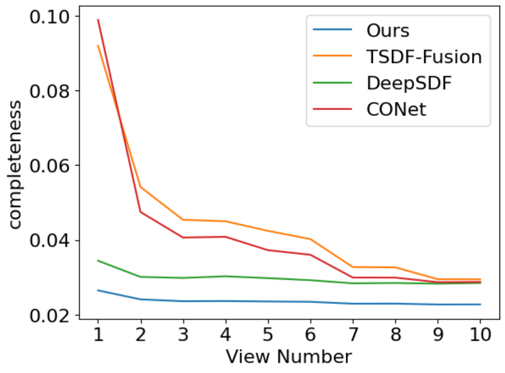
- RGB-D VO: a non-learning-based visual odometry method proposed in [Steinbrücker et al., 2011], which minimises the photometric loss between two frames. We re-parametrised it for object tracking.



(a) IoU



(b) Chamfer distance (L1)



(c) Completeness

Figure 5.8: Quantitative comparison of reconstruction quality and completion of our proposed methods v.s. classic TSDF-Fusion, learning-based DeepSDF and CONet. Our proposed method consistently show better reconstruction results from single view depth completion to multiple views.

- Point-to-Plane ICP: a non-learning geometric registration method [Rusinkiewicz and Levoy, 2001]
- Color ICP: a non-learning registration method using both color and geometric information [Park et al., 2017]
- Prior: a state-of-the-art object pose estimation using DeepSDF shape prior model. It is originally proposed in [Wang et al., 2021] for static object pose estimation and we re-parametrised it for estimating moving objects.
- NOCS: a state-of-the-art learning-based canonical correspondence regression

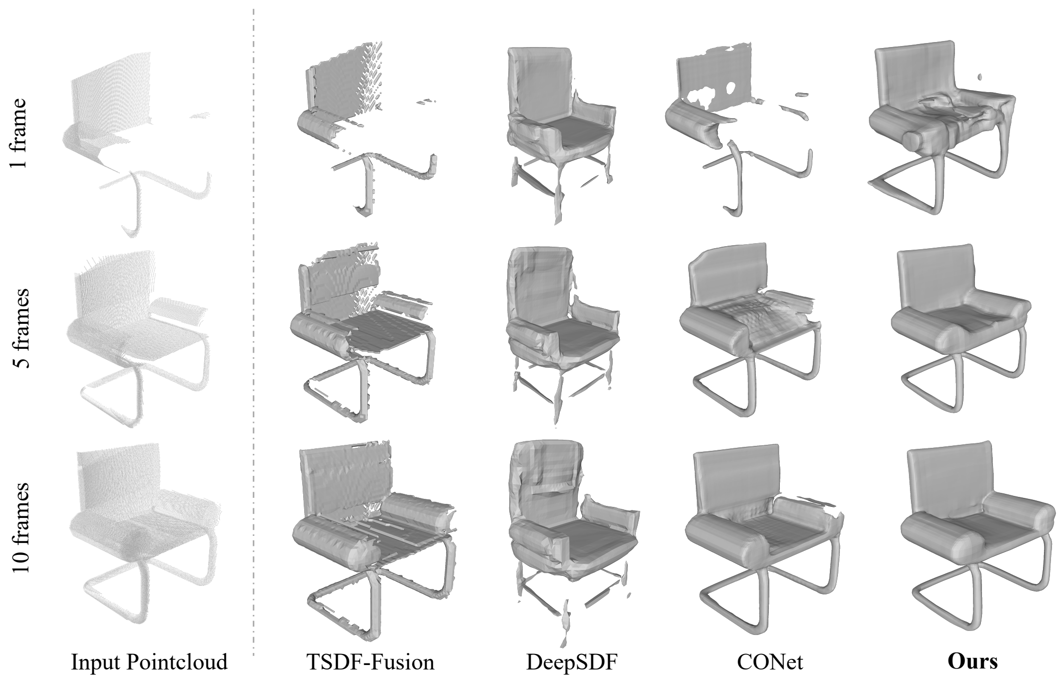


Figure 5.9: Qualitative Results on reconstructions. Our method is superior to all other methods in completing missing information and reconstructing fine details.

method. We adapted the network architecture proposed in [Rempe et al., 2020].

Metrics

To quantitatively evaluate the accuracy of the object tracking, we use the following metrics:

- ATE: Absolute. Trajectory Error defined in [Sturm et al., 2012], in the unit of of meters
- RPE_t: relative pose error (RPE) metrics in translation defined in [Sturm et al., 2012], in the unit of of metres
- RPE_R: relative pose error (RPE) metrics in rotation defined in [Sturm et al., 2012], in the unit of of degrees

- R_err: mean orientation error on each frame individually, in the unit of degrees

- t_err: mean translation error on each frame individually, in the unit of metres

The above metrics all indicate better tracking performance when the values are lower. To analyse the trajectory, we align the first frame of the estimated object pose to the ground-truth canonical space.

Results and Discussions

Table 5.1 reports the experimental results. It shows that our approach consistently outperforms both the non-learning-based and learning-based methods in small and large motion situations. For non-learning approaches, RGB-D VO [Steinbrücker et al., 2011], Point-to-Plane ICP [Rusinkiewicz and Levoy, 2001], and Color ICP [Park et al., 2017] only leverages the depth and intensity information from two-view measurements, without taking into account any object shape prior information. The single view canonical correspondence prediction from NOCS [Rempe et al., 2020] only considers shape prior information and does not take advantage of the multiview constraint. Our proposed method instead combines both multi-view constraint and shape prior information into object pose estimation. Similar to ours, the shape prior method [Wang et al., 2021] adopts category-level shape prior from DeepSDF [Park et al., 2019] and uses differential rendering to estimate object poses. However, latent code optimisation cannot necessarily capture the geometry deviation between training space and test shapes and thus affects the accuracy of pose estimation. It confirms that the object geometry completion and joint optimisation can improve the tracking accuracy.

method [unit]	ATE [m]	RPE_t [m]	RPE_R [°]	R_err [°]	t_err [m]
Ours	**0.030**	**0.027**	**3.845**	**3.931**	**0.040**
Prior	0.044	0.047	8.200	6.269	0.068
RGBD	0.254	0.106	18.47	32.25	0.314
Point2Plane	0.047	0.035	4.672	5.970	0.064
Color ICP	0.254	0.114	29.12	56.18	0.320
NOCS	0.074	0.059	23.46	37.87	0.085

(a) Keyframe gap-1

method [unit]	ATE [m]	RPE_t [m]	RPE_R [°]	R_err [°]	t_err [m]
Ours	**0.033**	**0.032**	**5.243**	**4.224**	**0.043**
Prior	0.046	0.052	11.91	8.063	0.063
RGBD	1.068	0.403	30.89	50.95	1.309
Point2Plane	0.070	0.056	8.570	9.384	0.087
Color ICP	0.536	0.351	36.69	60.56	0.568
NOCS	0.074	0.074	21.65	37.78	0.084

(b) Keyframe gap-2

method [unit]	ATE [m]	RPE_t [m]	RPE_R [°]	R_err [°]	t_err [m]
Ours	**0.034**	**0.038**	**6.767**	**4.834**	**0.044**
Prior	0.043	0.050	17.20	9.885	0.061
RGBD	1.942	0.866	43.34	68.86	2.177
Point2Plane	0.807	0.442	18.22	20.16	0.892
Color ICP	2.786	1.885	42.73	77.89	2.802
NOCS	0.073	0.085	26.95	35.41	0.083

(c) Keyframe gap-4

Table 5.1: Quantitative evaluation of object tracking method on the synthetic moving objects dataset.

5.4.3 Timing analysis

We implemented our system in PyTorch. The average inference time for a pair of RGB-D image in the resolution of 320×240 is 1.337s on a RTX 3090 platform. A more-detailed breakdown of computation time for each component is shown in Table 5.2a. A further breakdown of computation time on tracking components is shown in Table 5.2b.

Components	Tracking	Integration	Completion (visualization)
Time (s)	1.284	0.003	0.474

(a) System components

Components	Initialization	Coarse est.	Dense refinement
Time (ms)	0.643	0.150	1.129

(b) Object tracking components

Table 5.2: Run-time analysis (s)

We would like to highlight that our current implementation is prototyped in Python. We believe a real-time system can be achieved by exploiting C++ and further GPU parallelisation.

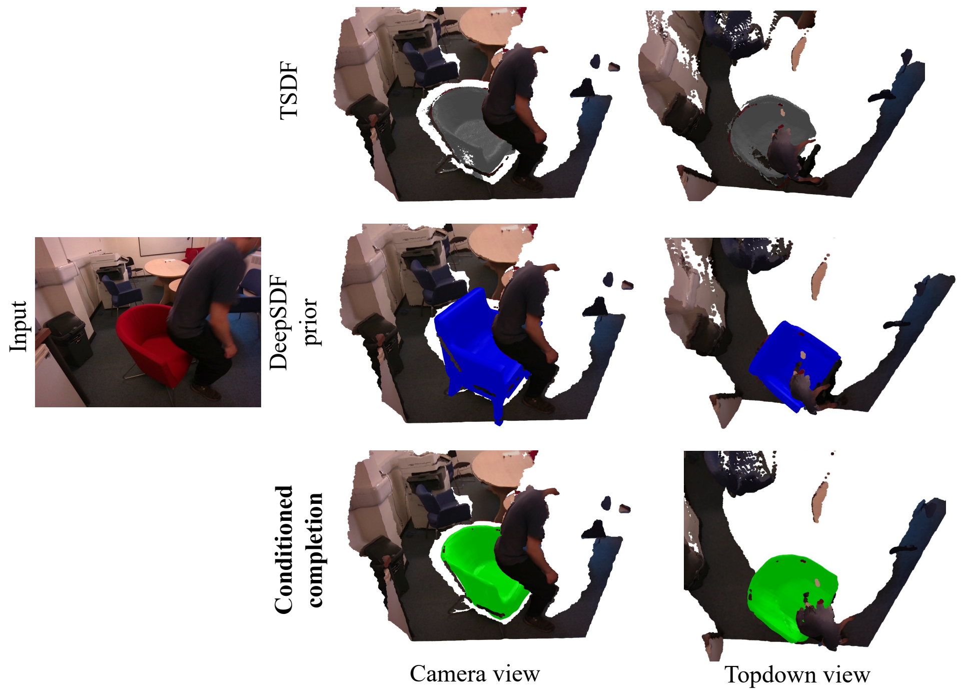
5.4.4 Qualitative Evaluations

We further demonstrate our proposed method in various real-world scenarios. Figure 5.10 shows the results in two different scenes. For each input image, we provide object reconstructions from the currently estimated camera viewpoint to visualise the observed part and from the top-down viewpoint to visualise the hidden part. As a qualitative comparison, we also show the reconstructions using classic TSDF fusion [Newcombe et al., 2011a] and the learned category-level DeepSDF object prior [Park et al., 2019], it can be seen that TSDF-Fusion can only reconstruct the visible parts, leaving many empty holes in the object models. DeepSDF, on the other hand, has watertight reconstructions, but does not match the measurement necessarily, especially for the objects that deviate from the training space. On the contrary, our system can maintain highly detailed reconstructions and generate watertight meshes by filling in the missing parts using category-level shape priors thanks to the conditioned completion. Figure 5.11 also shows a scene where our system can reconstruct the visible parts and complete the hidden information of a moving object. The object pose and object geometry for the moving object are optimised jointly.

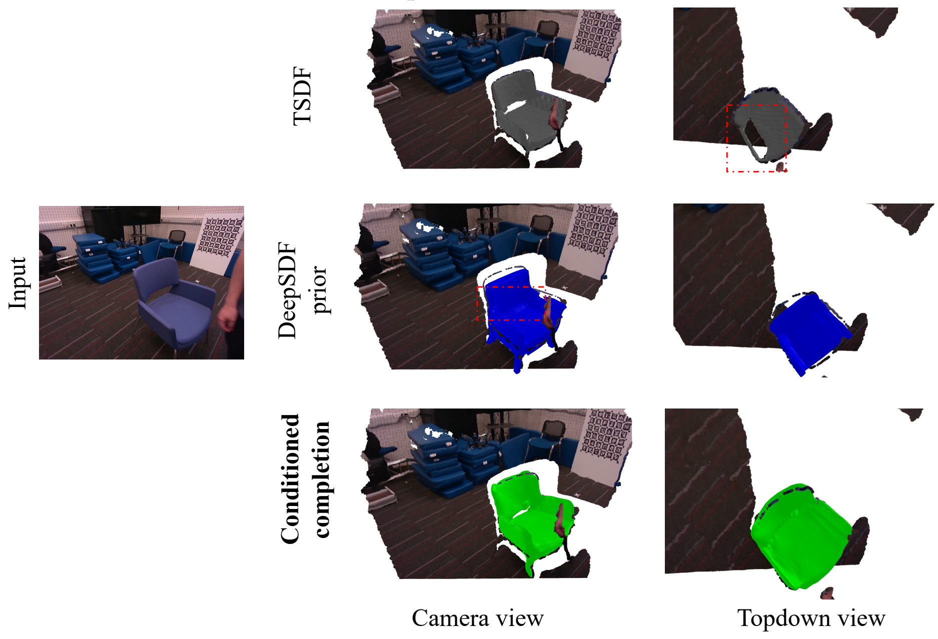
5.5 Conclusions and Discussions

We present a novel approach for object-level dynamic SLAM by incorporating learned category-level shape priors. It enables to reasonably complete the object geometry of unseen parts based on the prior knowledge, and provide more robust and accurate tracking accuracy, even under large frame-by-frame motion and in dynamic environments with moving human involved. Experimental results in various scenarios demonstrate the effectiveness of our method. We hope our method paves the way for a deeper understanding of exploring inter-instance relationships in object-level SLAM and can potentially benefit the robot applications of autonomous navigation and path planning.

Extending from what we learned in this work, there can be a few important directions. First, despite training entirely in synthetic data, our approach generalises well to the unseen objects in real-world experiments since our completion is conditioned on the fused TSDF reconstruction. However, we also observed that the completion performance would degrade as the target object severely deviates from the dataset models used in training. Besides, leveraging the category-level shape prior information requires alignment to the canonical space defined in the training data. However, when only part of an object is observed due to severe occlusions, the canonical correspondence network proposed in this work may not work well. Under such circumstances, the initialization of shape and pose becomes a highly challenging chicken-and-egg problem. Despite not being the focus of this work, this problem deserves more attention in future work.



(a) Completion of a red chair



(b) Completion of a blue chair

Figure 5.10: Qualitative comparison of classic TSDF volume representation (gray), DeepSDF shape prior representation (blue), and our conditioned completion representation (green): our representation can reconstruct the observed part more correctly than shape prior and complete the unseen part where TSDF representation fails.

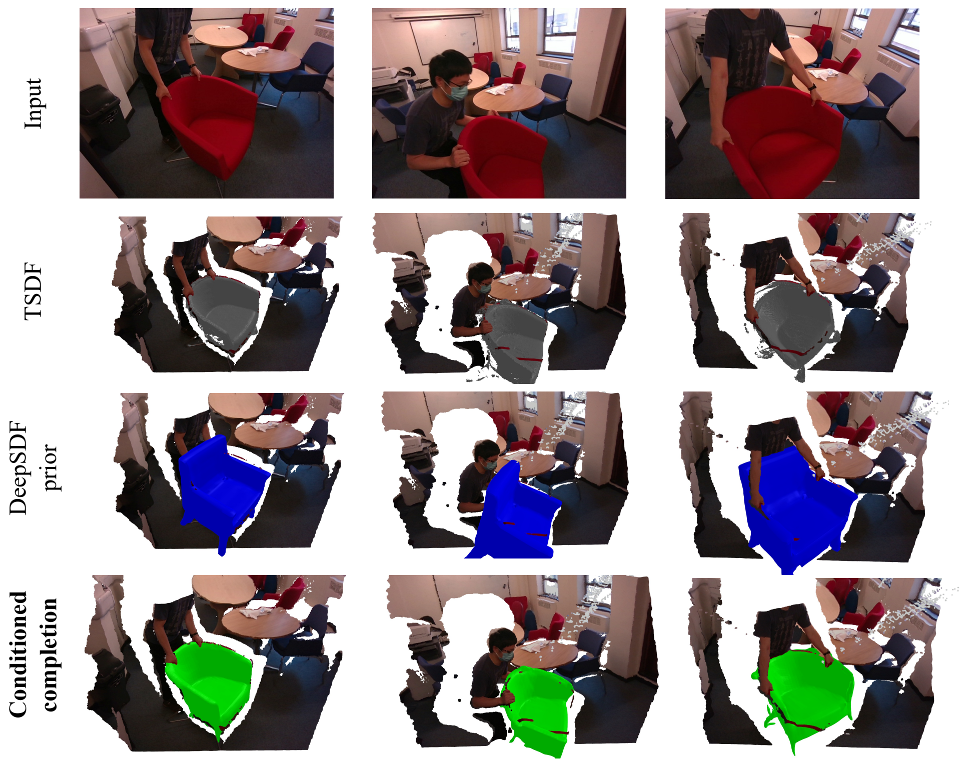


Figure 5.11: Segment, track, reconstruct and complete a moving chair. Background pointclouds are just for visualization.

Chapter 6

Conclusions

Contents of Chapter

6.1 Summary of results

In this thesis, we have presented several contributions towards advancing visual object-level SLAM in a dynamic environment by demonstrating their effectiveness qualitatively and quantitatively in synthetic and real-world scenes. We also discussed their respective limitations and proposed several possible solutions to address these shortcomings. In this section, we present a summary of the key contributions and results that have been discussed in greater detail in the previous chapters.

We first present MID-Fusion, a multi-instance dynamic SLAM using an octree-based volumetric representation, in Chapter 3. We have shown it can robustly estimate camera poses in dynamic environments and, at the same time, continuously estimate geometric, semantic, and motion properties for arbitrary static or moving objects in the scene. Each object, including the background, is reconstructed within its own object-oriented octree-based TSDF volume. Semantic classifications predicted independently from each frame are fused into semantic probabilistic distributions for each object using a simple and efficient Bayesian update scheme. Meanwhile, the pose of each existing moving object is estimated using the proposed object-oriented tracking method, and the camera pose can be robustly tracked against the static objects and background. Based on the estimated camera pose and object poses, we run projective data association to associate segmented masks with existing models and incrementally fuse corresponding colour, depth, semantic, and foreground object probabilities into each object model. Our system was one of the first to attempt to extend the traditional dense SLAM system to the domain of object-level dynamic SLAM, paving the way for many subsequent works in this area.

While the proposed system has been evaluated in experiments and shown to be effective in real-world experiments, it has a few limitations. First, it assumes a constant lighting condition to perform joint photometric and geometric tracking. The assumption could be violated by illumination changes in reality. Several other works, including ours presented in Chapter 4, have attempted to address this issue. Second, the projective data association method, which has also been widely used in dense SLAM systems, implies that the camera and objects move slowly. This is not necessarily valid in practice. There are a few other works adopting different motion models, such as constant velocity in [Bescós et al., 2021] or a white-noise-on-acceleration prior in [Huang et al., 2020]. These SLAM systems, however, do not estimate dense object geometries. More interesting and promising future directions that can be extended from this thesis will be discussed in Section 6.2.

To strengthen the robustness and accuracy of camera and object pose estimation under illumination changes and wide baseline situations, we present a novel dense image alignment method from RGB-D image inputs in Chapter 4. Dense feature maps and feature-metric uncertainty maps generated from a CNN can formulate a deep probabilistic feature-metric residual of the two-view constraint. This proposed probabilistic feature-metric residual can be efficiently minimised using Gauss-Newton in a coarse-to-fine manner. Furthermore, our network predicts an initial pose for faster and more reliable convergence by re-using the feature maps from the coarsest pyramid level. The whole pipeline can be trained end-to-end since the optimisation steps are differentiable. We have experimentally demonstrated that our method outperforms state-of-the-art methods in both learning and non-learning domains. We also show our approach can easily couple the proposed feature-metric residual with other residuals without manually scaling weight thanks to its probabilistic essence.

We learned from Chapter 4, have learnt that the deep features and feature-metric uncertainties can yield superior performance than raw pixel intensities in dense image alignment. Although we developed a prototype visual odometry system in our experiments, this can only be seen as part of a front-end component in a direct SLAM system. We expect that adding a back-end component, for example, factor graph optimisation of deep features with multi-view constraints, can further boost the performance. Several recent papers have also looked into this direction. Droid-SLAM presents a novel sparse SLAM system that is composed of a learning-based optical flow front-end and a back-end bundle adjustment (BA) layer [Teed and Deng, 2021]. Both the front-end and the back-end components are learned together in an end-to-end way with supervision from ground-truth poses, similar to what we did in Chapter 4. [Yoon et al., 2021] demonstrates that deep features and uncertainties may also be learned without supervision from ground-truth poses. Although they only demonstrate it in 3D lidar odometry, it should also be feasible to learn deep features for visual odometry in a similar way.

The final contribution, presented in Chapter 5, extends MID-Fusion (presented in Chapter 3) by incorporating shape prior into object map representations. We proposed a novel object-level SLAM system that can segment, track, reconstruct, and complete objects in dynamic scenes. The full object shape is predicted by conditioning on the measured depth and category-level shape prior. We proposed a joint optimisation for object pose and shape latent representation using geometric and differential rendering residuals towards its shape prior and completed geometry. Synthetic and real-world experiments have demonstrated that completed object geometry using shape prior can indeed improve the object reconstruction quality and lead to better object tracking accuracy.

6.2 Future works

This thesis has presented several contributions towards bringing deep learning to object-level dynamic SLAM. However, this thesis is not an exhaustive evaluation for all possible directions in this area. We believe object-level SLAM in a dynamic environment is critical for spatial AI and to enable intelligent robots to interact with real-world, especially in the scenes with humans. We hope the work presented in this thesis can open up new paths for future research. This section highlights some of possible directions.

Benchmark dataset for object-level dynamic SLAM

Despite the fast progress in the object-level (dynamic) SLAM research, there are very few public benchmark datasets available for researchers to test and benchmark their SLAM algorithms. Existing datasets for visual SLAM, such as ICL-NUIM [Handa et al., 2014] and TUM RGB-D datasets [Sturm et al., 2012], do not contain ground-truth object motion trajectories or ground-truth object reconstructions. [Judd and Gammell, 2019] introduced the oxford multimotion dataset, which collected several ground-truth object trajectories using motion capture markers. It includes different motion patterns and occlusions. However, object motion there only contains swing movements caused by gravity and 2D vehicle motion, lacking the important object motion pattern caused by human interactions. As well, there has not been a widely used object reconstruction benchmark. Existing dense visual object-level dynamic SLAM [Rünz and Agapito, 2017, Xu et al., 2019] mainly use their own synthetic dataset, which however only contains a few sequences. A dataset that contains various ground-truth object trajectories and ground-truth object models will be ideal. Generating synthetic datasets is easier as large-scale object models, such as ShapeNet [Chang et al., 2015], are easier to get access. On the other hand, real-world sequences containing sensor noise, motion blur, real-world object interaction with humans [Batra et al., 2020] are also important for SLAM benchmarks. It will be an important direction to release a dataset that includes both synthetic and real-world dynamic scenes.

Alternative object tracking methods

We have explored three object tracking algorithms in this thesis. They are all in the category of direct tracking approaches, with two (Chapter 3, Chapter 5) using frame-to-model tracking and one (Chapter 4) using frame-to-frame dense image alignment to estimate the relative transformation of object poses. However, we also noticed several alternative options to perform object tracking. One way is to do sparse indirect tracking, such as [Bescós et al., 2021, Wen and Bekris, 2021], which also contain a pose graph (in [Wen and Bekris, 2021]) or pose graph optimisation (in [Bescós et al., 2021]) to utilise multi-view constraint. Concurrently, [Teed and Deng, 2021] proposed a dense in-direct visual SLAM algorithm by learning feature correspondences using optical flow and constructing a bundle adjustment layer to perform global optimisation. We believe a similar dense-indirect method can also be developed for object tracking, benefited by the fast advancement in modern learning-based optical flow algorithms. Object prior information can also be brought into the optimisation constraints. Besides, most object tracking methods discussed before operate on discrete-time, returning estimations at fixed time stamps. It is also possible to estimate the continuous trajectories of the moving objects by also taking velocities and accelerations into account. This can impose additional physical constraints, e.g. non-holonomic motion constraints, and potentially also help tackle objects with fast motion and temporal occlusions.

Sensor fusion for object-level SLAM

This thesis develops object-level SLAM using an RGB-D sensor. It is also worth considering developing object-level SLAM systems with different sensor inputs. For example, IMU data can be fused with the camera input to further increase the robustness and accuracy of the sensor pose estimation, even under a dramatically changing environment. To track fast-moving objects, such as a ball thrown by people, we can also take advantage of an event camera, which offers lower latency, higher dynamic range, and lower power consumption than a normal camera. With the fast advancement of depth prediction accuracy from monocular camera images, we may also be able to get rid of the depth sensor and build a monocular version of the systems presented in this thesis.

Alternative map representations

As always being the core of SLAM, the choice of map representation affects the design of the SLAM system. This thesis has explored the choice of octree-based TSDF volume, continuous SDF field, continuous occupancy field, and also the combination of them. As one of the hottest directions at the time of writing, neural radiance field [Mildenhall et al., 2020] has been adopted in many fields, including object-level SLAM [Yuan et al., 2021]. However, [Yuan et al., 2021] can only one foreground object and requires multiple camera rigs as input sensors. It is expected that the neural radiance field will also be extended to the multi-instance dynamic object SLAM field. We can also learn the inter-instance relationship for objects in the scene. One example is shown in [Wald et al., 2020], where scene graph representation can be learnt for semantic representations.

Physical constraints in real world environments

This thesis presents several contributions to detecting, tracking, reconstructing, and completing objects in the scene. Although we can process multiple objects simultaneously and create an object-level map, each object is handled independently. The constraints and the relationships among objects are not explored yet in this thesis. With the advancement of 3D modelling of object geometries in dynamic scenes, we can also further infer the underlying physics of the 3D world and, importantly, bring these constraints into the optimisation. For example, we would expect the geometry of objects to not collide with each other. We should also expect that the motion of objects should respect certain physical rules, such as momenta, friction, and gravity. These physical properties can be, in turn, introduced into the geometry optimisation.

Integration with other robotics components

An object-awareness of the surrounding environment is important for intelligent robots to interact with the world. There are several potential robotic applications using our object-level SLAM systems. As our system demonstrates strong robustness and accuracy in dynamic indoor scenarios, a direct application will be to combine the SLAM systems with indoor navigation to enable an autonomous robot to reconstruct the room-scale environment or category-specific objects without requiring humans to leave the scene. Since our SLAM can handle moving objects and persons, the robot can co-exist with humans without disturbing their activities and thus the environment map can be updated spatially and temporally.

The prediction of object complete geometry in Chapter 5 suggests that it is sometimes unnecessary to have complete observations to reconstruct objects. Such object mapping and predictions may help robots better plan and navigate in a clustered indoor environment. Also, different object categories often correlate to the floorplan. For example, a sofa is usually located in the living room and a bed is usually located in the bedroom. Such high-level abstract of object-level location prior can also help robots better navigate in the indoor environment. We have already witnessed some progress [Chaplot et al., 2020] in this area. However, their mapping is static and based on 2D occupancy. We believe our mapping and tracking systems can further advance this area of research.

By analogy, our object-level dynamic SLAM can also be applied for manipulation and grasping tasks. As demonstrated in Chapter 3 and Chapter 5, our system can continuously track and fuse the geometric, semantic, and motion properties for every observed object. Having an awareness of the complete object geometry can help a robotic arm to navigate towards a goal. It also means that robot arms no longer need to wait for the objects to be static and then execute the tasks. It is also possible to conduct some collaboration tasks between human and robot arm(s).

Bibliography

[Baker and Matthews, 2004] Baker, S. and Matthews, I. (2004). Lucas-Kanade 20 years on: A unifying framework: Part 1. *International Journal of Computer Vision (IJCV)*, 56(3):221–255.

[Barfoot, 2017] Barfoot, T. D. (2017). *State Estimation for Robotics.* Cambridge University Press.

[Barnes et al., 2018] Barnes, D., Maddern, W., Pascoe, G., and Posner, I. (2018). Driven to distraction: Self-supervised distractor learning for robust monocular visual odometry in urban environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).*

[Bârsan et al., 2018] Bârsan, I. A., Liu, P., Pollefeys, M., and Geiger, A. (2018). Robust dense mapping for large-scale dynamic environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).*

[Batra et al., 2020] Batra, D., Chang, A. X., Chernova, S., Davison, A. J., Deng, J., Koltun, V., Levine, S., Malik, J., Mordatch, I., Mottaghi, R., Savva, M., and Su, H. (2020). Rearrangement: A challenge for embodied AI. *arXiv preprint arXiv:2011.01975.*

[Bay et al., 2006] Bay, H., Tuytelaars, T., and Van Gool, L. (2006). SURF: Speeded up robust features. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[Bescós et al., 2021] Bescós, B., Campos, C., Tardós, J. D., and Neira, J. (2021). Dynaslam II: tightly-coupled multi-object tracking and SLAM. *IEEE Robotics and Automation Letters*, 6(3):5191–5198.

[Bescós et al., 2018] Bescós, B., Fácil, J. M., Civera, J., and Neira, J. (2018). Dynaslam: Tracking, mapping and inpainting in dynamic scenes. *IEEE Robotics and Automation Letters*.

[Blais and Levine, 1995] Blais, G. and Levine, M. D. (1995). Registering Multiview Range Data to Create 3D Computer Objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 17(8):820–824.

[Bloesch et al., 2018] Bloesch, M., Czarnowski, J., Clark, R., Leutenegger, S., and Davison, A. J. (2018). CodeSLAM — learning a compact, optimisable representation for dense visual SLAM. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Bloesch et al., 2019] Bloesch, M., Laidlow, T., Clark, R., Leutenegger, S., and Davison, A. J. (2019). Learning meshes for dense visual SLAM. In *Proceedings of the International Conference on Computer Vision (ICCV)*.

[Bloesch et al., 2016] Bloesch, M., Sommer, H., Laidlow, T., Burri, M., Nützi, G., Fankhauser, P., Bellicoso, D., Gehring, C., Leutenegger, S., Hutter, M., and Siegwart, R. (2016). A Primer on the Differential Calculus of 3D Orientations. *CoRR*, abs/1606.0.

[Boikos and Bouganis, 2017] Boikos, K. and Bouganis, C.-S. (2017). A high-performance system-on-chip architecture for direct tracking for slam. In *2017*

27th International Conference on Field Programmable Logic and Applications (FPL), pages 1–7.

[Chang et al., 2015] Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., and Yu, F. (2015). ShapeNet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*.

[Chaplot et al., 2020] Chaplot, D. S., Gandhi, D. P., Gupta, A., and Salakhutdinov, R. R. (2020). Object goal navigation using goal-oriented semantic exploration. In *Neural Information Processing Systems (NIPS)*.

[Chen et al., 2013] Chen, J., Bautembach, D., and Izadi, S. (2013). Scalable real-time volumetric surface reconstruction. In *Proceedings of SIGGRAPH*.

[Chen and Medioni, 1992] Chen, Y. and Medioni, G. (1992). Object modeling by registration of multiple range images. *Image and Vision Computing (IVC)*, 10(3):145–155.

[Çiçek et al., 2016] Çiçek, Ö., Abdulkadir, A., Lienkamp, S. S., Brox, T., and Ronneberger, O. (2016). 3d u-net: learning dense volumetric segmentation from sparse annotation. In *Proceedings of the International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pages 424–432.

[Concha and Civera, 2015] Concha, A. and Civera, J. (2015). An evaluation of robust cost functions for rgb direct mapping. In *Proceedings of the European Conference on Mobile Robotics (ECMR)*.

[Curless and Levoy, 1996] Curless, B. and Levoy, M. (1996). A volumetric method for building complex models from range images. In *Proceedings of SIGGRAPH*.

[Czarnowski et al., 2020] Czarnowski, J., Laidlow, T., Clark, R., and Davison, A. J. (2020). Deepfactors: Real-time probabilistic dense monocular SLAM. *IEEE Robotics and Automation Letters*, 5(2):721–728.

[Czarnowski et al., 2017] Czarnowski, J., Leutenegger, S., and Davison, A. J. (2017). Semantic texture for robust dense tracking. In *Proceedings of the International Conference on Computer Vision Workshops (ICCVW)*.

[Dai et al., 2017] Dai, A., Nießner, M., Zollhöfer, M., Izadi, S., and Theobalt, C. (2017). BundleFusion: Real-time Globally Consistent 3D Reconstruction using On-the-fly Surface Re-integration. *ACM Transactions on Graphics (TOG)*, 36(3):24:1–24:18.

[Dai et al., 2016] Dai, J., He, K., and Sun, J. (2016). Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Davison, 2003] Davison, A. J. (2003). Real-Time Simultaneous Localisation and Mapping with a Single Camera. In *Proceedings of the International Conference on Computer Vision (ICCV)*.

[Davison et al., 2007] Davison, A. J., Molton, N. D., Reid, I., and Stasse, O. (2007). MonoSLAM: Real-Time Single Camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 29(6):1052–1067.

[Dou et al., 2016] Dou, M., Khamis, S., Degtyarev, Y., Davidson, P., Fanello, S., Kowdle, A., Escolano, S. O., Rhemann, C., Kim, D., Taylor, J., Kohli, P., Tankovich, V., and Izadi, S. (2016). Fusion4D: Real-time performance capture of challenging scenes. In *Proceedings of SIGGRAPH*.

[Durrant-Whyte and Bailey, 2006] Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms. *IEEE Robotics and Automation Magazine*, 13(2):99–110.

[Eade, 2009] Eade, E. (2009). Gauss-Newton / Levenberg-Marquardt Optimization. http://www.ethaneade.com/optimization.pdf.

[Eade, 2014] Eade, E. (2014). Lie groups for computer vision. http://www.ethaneade.com/lie_groups.pdf.

[Eigen et al., 2014] Eigen, D., Puhrsch, C., and Fergus, R. (2014). Depth Map Prediction from a Single Image using a Multi-Scale Deep Network. In *Neural Information Processing Systems (NIPS)*.

[Engel et al., 2017] Engel, J., Koltun, V., and Cremers, D. (2017). Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*.

[Engel, 2017] Engel, J.-J. (2017). *Large-Scale Direct SLAM and 3D Reconstruction in Real-Time*. PhD thesis, Technische Universität München.

[Eslami et al., 2018] Eslami, S. A., Rezende, D. J., Besse, F., Viola, F., Morcos, A. S., Garnelo, M., Ruderman, A., Rusu, A. A., Danihelka, I., Gregor, K., et al. (2018). Neural scene representation and rendering. *Science*, 360(6394):1204–1210.

[Geiger et al., 2011] Geiger, A., Ziegler, J., and Stiller, C. (2011). Stereoscan: Dense 3d reconstruction in real-time. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[Gropp et al., 2020] Gropp, A., Yariv, L., Haim, N., Atzmon, M., and Lipman, Y. (2020). Implicit geometric regularization for learning shapes. *Proceedings of the International Conference on Machine Learning (ICML)*.

[Hadsell et al., 2006] Hadsell, R., Chopra, S., and LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Handa et al., 2014] Handa, A., Whelan, T., McDonald, J. B., and Davison, A. J. (2014). A Benchmark for RGB-D Visual Odometry, 3D Reconstruction and SLAM.

In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

[He et al., 2017] He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask R-CNN. In *Proceedings of the International Conference on Computer Vision (ICCV)*.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Horn and Schunck, 1981] Horn, B. and Schunck, B. (1981). Determining optical flow. *Artificial Intelligence*, 17:185–203.

[Hornik, 1989] Hornik, K. (1989). Multilayer Feedforward Networks are Universal Approximators. *Journal of Neural Networks*, 2:359–366.

[Hornung et al., 2013] Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., and Burgard, W. (2013). OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*.

[Hu et al., 2019] Hu, L., Xu, W., Huang, K., and Kneip, L. (2019). Deep-slam++: Object-level rgbd slam based on class-specific deep shape priors. *arXiv preprint arXiv:1907.09691*.

[Huang et al., 2020] Huang, J., Yang, S., Mu, T.-J., and Hu, S.-M. (2020). Clustervo: Clustering moving instances and estimating visual odometry for self and surroundings. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Huang et al., 2019] Huang, J., Yang, S., Zhao, Z., Lai, Y.-K., and Hu, S.-M. (2019). Clusterslam: A slam backend for simultaneous rigid body clustering and motion estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5875–5884.

[Innmann et al., 2016] Innmann, M., Zollhöfer, M., Nießner, M., Theobalt, C., and Stamminger, M. (2016). VolumeDeform: Real-time Volumetric Non-rigid Reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[Jaimez et al., 2017] Jaimez, M., Kerl, C., Gonzalez-Jimenez, J., and Cremers, D. (2017). Fast odometry and scene flow from rgb-d cameras based on geometric clustering. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

[Jaramillo et al., 2017] Jaramillo, C., Taguchi, Y., and Feng, C. (2017). Direct multichannel tracking. In *Proceedings of the International Conference on 3D Vision (3DV)*.

[Judd and Gammell, 2019] Judd, K. M. and Gammell, J. D. (2019). The oxford multimotion dataset: Multiple se (3) motions with ground truth. *IEEE Robotics and Automation Letters*, 4(2):800–807.

[Kahler et al., 2015] Kahler, O., Prisacariu, V. A., Ren, C. Y., Sun, X., Torr, P. H. S., and Murray, D. W. (2015). Very High Frame Rate Volumetric Integration of Depth Images on Mobile Device. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*.

[Kendall and Gal, 2017] Kendall, A. and Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision? In *Neural Information Processing Systems (NIPS)*.

[Kendall et al., 2018] Kendall, A., Gal, Y., and Cipolla, R. (2018). Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Kingma and Ba, 2015] Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR).*

[Klein and Murray, 2007] Klein, G. and Murray, D. W. (2007). Parallel Tracking and Mapping for Small AR Workspaces. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR).*

[Laidlow et al., 2017] Laidlow, T., Bloesch, M., Li, W., and Leutenegger, S. (2017). Dense RGB-D-Inertial SLAM with map deformations. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS).*

[Leutenegger et al., 2011] Leutenegger, S., Chli, M., and Siegwart, R. (2011). BRISK: Binary robust invariance scalable keypoints. In *Proceedings of the International Conference on Computer Vision (ICCV).*

[Leutenegger et al., 2014] Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., and Furgale, P. (2014). Keyframe-based visual-inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334.

[Li et al., 2021] Li, K., Rezatofighi, H., and Reid, I. (2021). Moltr: Multiple object localization, tracking and reconstruction from monocular rgb videos. *IEEE Robotics and Automation Letters*, 6(2):3341–3348.

[Li et al., 2018a] Li, P., Qin, T., et al. (2018a). Stereo vision-based semantic 3d object and ego-motion tracking for autonomous driving. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 646–661.

[Li et al., 2018b] Li, W., Saeedi, S., McCormac, J., Clark, R., Tzoumanikas, D., Ye, Q., Huang, Y., Tang, R., and Leutenegger, S. (2018b). Interiornet: Mega-scale multi-sensor photo-realistic indoor scenes dataset. In *Proceedings of the British Machine Vision Conference (BMVC).*

[Li et al., 2018c] Li, Y., Wang, G., Ji, X., Xiang, Y., and Fox, D. (2018c). DeepIM: Deep iterative matching for 6d pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[Lin et al., 2019] Lin, C.-H., Wang, O., Russell, B. C., Shechtman, E., Kim, V. G., Fisher, M., and Lucey, S. (2019). Photometric mesh optimization for video-aligned 3d object reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Lin et al., 2014] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 740–755.

[Liu et al., 2019] Liu, C., Gu, J., Kim, K., Narasimhan, S. G., and Kautz, J. (2019). Neural rgb(r)d sensing: Depth and uncertainty from a video camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Liu et al., 2016] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). SSD: Single shot multibox detector. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[Loop et al., 2016] Loop, C., Cai, Q., Chou, P., and Orts-Escolano, S. (2016). A closed-form bayesian fusion equation using occupancy probabilities. In *Proceedings of the International Conference on 3D Vision (3DV)*.

[Lowe, 1999] Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision (ICCV)*.

[Lucas and Kanade, 1981] Lucas, B. D. and Kanade, T. (1981). An Iterative Image Registration Technique with an Application to Stereo Vision. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI).*

[Lv et al., 2019] Lv, Z., Dellaert, F., Rehg, J., and Geiger, A. (2019). Taking a deeper look at the inverse compositional algorithm. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[MacTavish and Barfoot, 2015] MacTavish, K. and Barfoot, T. D. (2015). At all costs: A comparison of robust cost functions for camera correspondence outliers. In *Proceedings of the Canadian Conference on Computer and Robot Vision (CRV).*

[Majercik et al., 2018] Majercik, A., Crassin, C., Shirley, P., and McGuire, M. (2018). A ray-box intersection algorithm and efficient dynamic voxel rendering. *Journal of Computer Graphics Techniques Vol*, 7(3):66–81.

[Mayer et al., 2016] Mayer, N., Ilg, E., Hausser, P., Fischer, P., Cremers, D., Dosovitskiy, A., and Brox, T. (2016). A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[McCormac et al., 2018] McCormac, J., Clark, R., Bloesch, M., Davison, A. J., and Leutenegger, S. (2018). Fusion++:volumetric object-level slam. In *Proceedings of the International Conference on 3D Vision (3DV).*

[McCormac et al., 2017] McCormac, J., Handa, A., Davison, A. J., and Leutenegger, S. (2017). SemanticFusion: Dense 3D semantic mapping with convolutional neural networks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).*

[Mescheder et al., 2019] Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., and Geiger, A. (2019). Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Mildenhall et al., 2020] Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2020). Nerf: Representing scenes as neural radiance fields for view synthesis. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[Muller et al., 2021] Muller, N., Wong, Y.-S., Mitra, N. J., Dai, A., and Nießner, M. (2021). Seeing behind objects for 3d multi-object tracking in rgb-d sequences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Mur-Artal and Tardós, 2017] Mur-Artal, R. and Tardós, J. D. (2017). ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Transactions on Robotics (T-RO)*, 33(5):1255–1262.

[Nakajima et al., 2018] Nakajima, Y., Tateno, K., Tombari, F., and Saito, H. (2018). Fast and accurate semantic mapping through geometric-based incremental segmentation. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 385–392.

[Newcombe et al., 2015] Newcombe, R. A., Fox, D., and Seitz, S. M. (2015). Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Newcombe et al., 2011a] Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohli, P., Shotton, J., Hodges, S., and Fitzgibbon, A. (2011a). KinectFusion: Real-Time Dense Surface Mapping and Tracking. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*.

[Newcombe et al., 2011b] Newcombe, R. A., Lovegrove, S., and Davison, A. J. (2011b). DTAM: Dense Tracking and Mapping in Real-Time. In *Proceedings of the International Conference on Computer Vision (ICCV).*

[Nicholson et al., 2018] Nicholson, L., Milford, M., and Sünderhauf, N. (2018). QuadricSLAM: Constrained Dual Quadrics from Object Detections as Landmarks in Object-oriented SLAM. *IEEE Robotics and Automation Letters.*

[Nießner et al., 2013] Nießner, M., Zollhöfer, M., Izadi, S., and Stamminger, M. (2013). Real-time 3D Reconstruction at Scale using Voxel Hashing. In *Proceedings of SIGGRAPH.*

[Noh et al., 2015] Noh, H., Hong, S., and Han, B. (2015). Learning deconvolution network for semantic segmentation. *arXiv preprint arXiv:1505.04366.*

[Park et al., 2017] Park, J., Zhou, Q.-Y., and Koltun, V. (2017). Colored point cloud registration revisited. In *Proceedings of the International Conference on Computer Vision (ICCV).*

[Park et al., 2019] Park, J. J., Florence, P., Straub, J., Newcombe, R., and Lovegrove, S. (2019). DeepSDF: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[Park et al., 2020] Park, K., Mousavian, A., Xiang, Y., and Fox, D. (2020). Latentfusion: End-to-end differentiable reconstruction and rendering for unseen object pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[Parker et al., 1998] Parker, S., Shirley, P., Livnat, Y., Hansen, C., and Sloan, P. (1998). Interactive Ray Tracing for Isosurface Rendering. In *Proceedings of Visualization.*

[Peng et al., 2020] Peng, S., Niemeyer, M., Mescheder, L., Pollefeys, M., and Geiger, A. (2020). Convolutional occupancy networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[Platinsky et al., 2017] Platinsky, L., Davison, A. J., and Leutenegger, S. (2017). Monocular visual odometry: Sparse joint optimisation or dense alternation? In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

[Qi et al., 2017a] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017a). Pointnet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 652–660.

[Qi et al., 2017b] Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017b). Pointnet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Neural Information Processing Systems (NIPS)*, pages 5099–5108.

[Rempe et al., 2020] Rempe, D., Birdal, T., Zhao, Y., Gojcic, Z., Sridhar, S., and Guibas, L. J. (2020). Caspr: Learning canonical spatiotemporal point cloud representations. *Neural Information Processing Systems (NIPS)*.

[Rosinol et al., 2020] Rosinol, A., Abate, M., Chang, Y., and Carlone, L. (2020). Kimera: an open-source library for real-time metric-semantic localization and mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

[Rosinol et al., 2019] Rosinol, A., Sattler, T., Pollefeys, M., and Carlone, L. (2019). Incremental visual-inertial 3d mesh generation with structural regularities. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

[Rublee et al., 2011] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). ORB: an efficient alternative to SIFT or SURF. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 2564–2571. IEEE.

[Rünz and Agapito, 2017] Rünz, M. and Agapito, L. (2017). Co-fusion: Real-time segmentation, tracking and fusion of multiple objects. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

[Rünz and Agapito, 2018] Rünz, M. and Agapito, L. (2018). Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects. *arXiv preprint arXiv:1804.09194*.

[Runz et al., 2020] Runz, M., Li, K., Tang, M., Ma, L., Kong, C., Schmidt, T., Reid, I., Agapito, L., Straub, J., Lovegrove, S., et al. (2020). Frodo: From detections to 3d objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Rusinkiewicz and Levoy, 2001] Rusinkiewicz, S. and Levoy, M. (2001). Efficient Variants of the ICP Algorithm. In *Proceedings of the IEEE International Workshop on 3D Digital Imaging and Modeling (3DIM)*.

[Salas-Moreno et al., 2013] Salas-Moreno, R. F., Newcombe, R. A., Strasdat, H., Kelly, P. H. J., and Davison, A. J. (2013). SLAM++: Simultaneous Localisation and Mapping at the Level of Objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Schmidt et al., 2017] Schmidt, T., Newcombe, R., and Fox, D. (2017). Self-supervised visual descriptor learning for dense correspondence. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

[Scona et al., 2018] Scona, R., Jaimez, M., Petillot, Y. R., Fallon, M., and Cremers, D. (2018). StaticFusion: Background reconstruction for dense rgb-d slam in dynamic environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).*

[Shi and Tomasi, 1994] Shi, J. and Tomasi, C. (1994). Good Features to Track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[Simonyan and Zisserman, 2015] Simonyan, K. and Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. In *Proceedings of the International Conference on Learning Representations (ICLR).*

[Sitzmann et al., 2020] Sitzmann, V., Martel, J., Bergman, A., Lindell, D., and Wetzstein, G. (2020). Implicit neural representations with periodic activation functions. In *Neural Information Processing Systems (NIPS).*

[Sitzmann et al., 2019] Sitzmann, V., Thies, J., Heide, F., Nießner, M., Wetzstein, G., and Zollhofer, M. (2019). Deepvoxels: Learning persistent 3d feature embeddings. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[Steinbrücker et al., 2011] Steinbrücker, F., Sturm, J., and Cremers, D. (2011). Real-Time Visual Odometry from Dense RGB-D Images. In *Workshop on Live Dense Reconstruction from Moving Cameras at ICCV.*

[Strecke and Stuckler, 2019] Strecke, M. and Stuckler, J. (2019). Em-fusion: Dynamic object-level slam with probabilistic data association. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[Strecke and Stuckler, 2020] Strecke, M. and Stuckler, J. (2020). Where does it end?-reasoning about hidden surfaces by object intersection constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[Sturm et al., 2012] Sturm, J., Engelhard, N., Endres, F., Burgard, W., and Cremers, D. (2012). A Benchmark for the Evaluation of RGB-D SLAM Systems. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS).*

[Sucar et al., 2020] Sucar, E., Wada, K., and Davison, A. J. (2020). NodeSLAM: Neural object descriptors for multi-view shape reconstruction. In *Proceedings of the International Conference on 3D Vision (3DV).*

[Sünderhauf et al., 2017] Sünderhauf, N., Pham, T. T., Latif, Y., Milford, M., and Reid, I. (2017). Meaningful maps with object-oriented semantic mapping. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS).*

[Tang and Tan, 2019] Tang, C. and Tan, P. (2019). BA-net: Dense bundle adjustment networks. In *Proceedings of the International Conference on Learning Representations (ICLR).*

[Tateno et al., 2017] Tateno, K., Tombari, F., Laina, I., and Navab, N. (2017). CNN-SLAM: Real-time dense monocular slam with learned depth prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[Tateno et al., 2015] Tateno, K., Tombari, F., and Navab, N. (2015). Real-time and scalable incremental segmentation on dense slam. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS).*

[Teed and Deng, 2021] Teed, Z. and Deng, J. (2021). DROID-SLAM: Deep Visual SLAM for Monocular, Stereo, and RGB-D Cameras. In *Neural Information Processing Systems (NIPS).*

[Thrun et al., 2005] Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics.* Cambridge: MIT Press.

[Umeyama, 1991] Umeyama, S. (1991). Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 13(04):376–380.

[Ummenhofer et al., 2016] Ummenhofer, B., Zhou, H., Uhrig, J., Mayer, N., Ilg, E., Dosovitskiy, A., and Brox, T. (2016). DeMoN: Depth and motion network for learning monocular stereo. *arXiv preprint arXiv:1612:02401*.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Neural Information Processing Systems (NIPS)*.

[Vespa et al., 2018] Vespa, E., Nikolov, N., Grimm, M., Nardi, L., Kelly, P. H., and Leutenegger, S. (2018). Efficient octree-based volumetric SLAM supporting signed-distance and occupancy mapping. *IEEE Robotics and Automation Letters*.

[von Stumberg et al., 2020] von Stumberg, L., Wenzel, P., Khan, Q., and Cremers, D. (2020). GN-Net: The gauss-newton loss for multi-weather relocalization. *IEEE Robotics and Automation Letters*, 5(2):890–897.

[Wald et al., 2020] Wald, J., Dhamo, H., Navab, N., and Tombari, F. (2020). Learning 3d semantic scene graphs from 3d indoor reconstructions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3961–3970.

[Wang et al., 2018] Wang, C., Galoogahi, H. K., Lin, C.-H., and Lucey, S. (2018). Deep-LK for efficient adaptive object tracking. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

[Wang et al., 2003] Wang, C.-C., Thorpe, C., and Thrun, S. (2003). Online simultaneous localization and mapping with detection and tracking of moving objects: theory and results from a ground vehicle in crowded urban areas. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

[Wang et al., 2019] Wang, H., Sridhar, S., Huang, J., Valentin, J., Song, S., and Guibas, L. J. (2019). Normalized object coordinate space for category-level 6d object pose and size estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[Wang et al., 2021] Wang, J., Rünz, M., and Agapito, L. (2021). Dsp-slam: Object oriented slam with deep shape priors. In *Proceedings of the International Conference on 3D Vision (3DV).*

[Wang et al., 2020] Wang, R., Yang, N., Stückler, J., and Cremers, D. (2020). Directshape: Photometric alignment of shape priors for visual vehicle pose and shape estimation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).*

[Wen and Bekris, 2021] Wen, B. and Bekris, K. E. (2021). Bundletrack: 6d pose tracking for novel objects without instance or category-level 3d models. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS).*

[Whelan et al., 2016] Whelan, T., Salas-Moreno, R. F., Glocker, B., Davison, A. J., and Leutenegger, S. (2016). ElasticFusion: Real-time dense SLAM and light source estimation. *International Journal of Robotics Research (IJRR)*, 35(14):1697–1716.

[Wu et al., 2016] Wu, Y. et al. (2016). Tensorpack. `https://github.com/tensorpack/`.

[Xu et al., 2021a] Xu, B., Davison, A., and Leutenegger, S. (2021a). Deep probabilistic feature-metric tracking. *IEEE Robotics and Automation Letters*, 6(1):223 – 230.

[Xu et al., 2019] Xu, B., Li, W., Tzoumanikas, D., Bloesch, M., Davison, A., and Leutenegger, S. (2019). MID-Fusion: Octree-based object-level multi-instance dynamic slam. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).*

[Xu et al., 2021b] Xu, B., Ma, L., Ye, Y., Schmidt, T., Twigg, C. D., and Lovegrove, S. (2021b). Identity-disentangled neural deformation model for dynamic meshes. *arXiv preprint arXiv:2109.15299*.

[Yang et al., 2020] Yang, N., von Stumberg, L., Wang, R., and Cremers, D. (2020). D3VO: Deep depth, deep pose and deep uncertainty for monocular visual odometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Yoon et al., 2021] Yoon, D. J., Zhang, H., Gridseth, M., Thomas, H., and Barfoot, T. D. (2021). Unsupervised learning of lidar features for use ina probabilistic trajectory estimator. *IEEE Robotics and Automation Letters*, 6(2):2130–2138.

[Yuan et al., 2021] Yuan, W., Lv, Z., Schmidt, T., and Lovegrove, S. (2021). Star: Self-supervised tracking and reconstruction of rigid objects in motion with neural rendering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Zhi et al., 2019] Zhi, S., Bloesch, M., Leutenegger, S., and Davison, A. J. (2019). SceneCode: Monocular dense semantic reconstruction using learned encoded scene representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Zhou et al., 2018a] Zhou, H., Ummenhofer, B., and Brox, T. (2018a). Deeptam: Deep tracking and mapping. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[Zhou et al., 2018b] Zhou, Q.-Y., Park, J., and Koltun, V. (2018b). Open3D: A modern library for 3D data processing. *arXiv preprint arXiv:1801.09847*.

[Zhou et al., 2017] Zhou, T., Brown, M., Snavely, N., and Lowe, D. G. (2017). Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.