Title

3D Semantic Scene Understanding and Reconstruction

Permalink

https://escholarship.org/uc/item/63q0405d

Author

Feng, Qiaojun

Publication Date

2023

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA SAN DIEGO

3D Semantic Scene Understanding and Reconstruction

A dissertation submitted in partial satisfaction of the requirements
for the degree Doctor of Philosophy

in

Electrical Engineering (Intelligent Systems, Robotics and Control)

by

Qiaojun Feng

Committee in charge:

Professor Nikolay Atanasov, Chair
Professor Henrik Christensen
Professor Hao Su
Professor Xiaolong Wang

2023

The Dissertation of Qiaojun Feng is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2023

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

This dissertation would not have been possible without my advisor Dr. Nikolay Atanasov. I was fortunate to start as Nikolay's Ph.D. student in the same year when Nikolay started as a professor. Nikolay is a very nice person. He has always viewed me as an independent researcher and shown his support to me. He really enjoys learning and discussing on our research topic and provides insightful and detailed guidance. My greatest thanks go to Nikolay, not only for his passion and dedication to research but also for his kindness and exemplary conduct, which mark him as a true role model. I could never wish for a better advisor than Nikolay.

I thank my Ph.D. committee members, Dr. Henrik Christensen, Dr. Hao Su and Dr. Xiaolong Wang, for showing willingness to learn about my research, accommodating the schedule to attend my proposal and defense, and reviewing and providing valuable feedback. I thank Dr. Bodi Yuan and Erik Nelson for hosting and mentoring me during the two summer internships. I thank my undergraduate advisors, Dr. Xin Pei, Dr. Jianjiang Feng and Dr. Mihaela van der Schaar, for bringing me into the research and encouraging me to pursue graduate education.

I was fortunate to have the chance to work with a lot of talented people. I want to thank my coauthors, Vikas Dhiman, Sai Jadhav, You-Yi Jau, Jinzhao Li, Yue Meng, Mo Shan, Aiwei Yin and Tianyu Zhao. I also want to thank the ERL members, Ibrahim Akbar, Abdullah Altawaitan, Arash Asgharivaskasi, Hanwen Cao, Kun Chen, Zhirui Dai, Yaobang Deng, Thai Duong, Siwei Guo, Chang Han, Yen-Ting Huang, Shumon Koga, Zhichao Li, Jialiang Liu, Kehan Long, Minh Pham, Harshini Rajachander, Hojoon Shin, Tianyu Wang, Youxing Wang, Yinzhuang Yi, Xinyang Yu, Gao Zhu and Ehsan Zobeidi. I also want to thank my friends, Weixiang Chen, Yusi Chen, Ji Dai, Chen Du, Zijian Fang, Junheng Hao, Huan Hu, Junfeng Huang, Jingling Li, Jianbo Liu, Kaiming Liu, Shijia Liu, Tianyi Liu, Zhuolin Ouyang, Pengluo Wang, Weijian Xu, Zhen Yang, Xiaotian Zhan, Shu Zhang, Ruoyu Zhao, Yubin Zou, and many other friends I miss to mention, for believing in and supporting me throughout the Ph.D. journey.

Finally I want to thank my parents, Luanling Wu and Fenghua Feng, my grandparents, Ailian Hu, Zhongguo Wu, Shuixiu Yu and Renshan Feng, for constantly caring me and supporting

me. Last but not least, I want to thank my love Yuli Han and our flurry creature Dousha, for the love and support as always.

Chapter 2, in part, is a reprint of the material as it appears in Q. Feng and N. Atanasov, ”Mesh Reconstruction from Aerial Images for Outdoor Terrain Mapping Using Joint 2D-3D Learning,” IEEE International Conference on Robotics and Automation (ICRA), pp. 5208-5214, 2021. Chapter 2, in part, has been submitted for publication of the material as it may appear in Q. Feng and N. Atanasov, ”TerrainMesh: Metric-Semantic Terrain Reconstruction from Aerial Images Using Joint 2D-3D Learning,” IEEE Transactions on Robotics, 2023. The dissertation author was the primary author of these papers.

Chapter 3, in part, is a reprint of the material as it appears in Q. Feng and N. Atanasov, ”Fully Convolutional Geometric Features for Category-level Object Alignment,” IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 8492-8498, 2020, and in T. Zhao, Q. Feng, S. Jadhav and N. Atanasov, ”CORSAIR: Convolutional Object Retrieval and Symmetry-AIded Registration,” IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 47-54, 2021, and in M. Shan, Q. Feng, Y. -Y. Jau and N. Atanasov, ”ELLIPSDF: Joint Object Pose and Shape Optimization with a Bi-level Ellipsoid and Signed Distance Function Description,” IEEE/CVF International Conference on Computer Vision (ICCV), pp. 5926-5935, 2021. The dissertation author was the primary author of these papers.

Chapter 4, in part, is a reprint of the material as it appears in Q. Feng, Y. Meng, M. Shan and N. Atanasov, ”Localization and Mapping using Instance-specific Mesh Models,” IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4985-4991, 2019, and in M. Shan, Q. Feng and N. Atanasov, ”OrcVIO: Object residual constrained Visual-Inertial Odometry,” IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5104-5111, 2020. Chapter 4, in part, is currently being prepared for submission for publication of the material as it may appear in M. Shan, Q. Feng, J. Li, A. Yin, V. Dhiman and N. Atanasov, ”OrcVIO: Object residual constrained Visual Inertial Odometry.” The dissertation author was the primary author of these papers.

VITA

2017	Bachelor of Engineering in Automation, Tsinghua University
2019	Master of Science in Electrical Engineering (Intelligent Systems, Robotics and Control), University of California San Diego
2023	Doctor of Philosophy in Electrical Engineering (Intelligent Systems, Robotics and Control), University of California San Diego

PUBLICATIONS

M. Shan, **Q. Feng**, J. Li, A. Yin, V. Dhiman and N. Atanasov, “OrcVIO: Object residual constrained Visual Inertial Odometry”, in preparation for submission.

Q. Feng and N. Atanasov, “TerrainMesh: Metric-Semantic Terrain Reconstruction from Aerial Images Using Joint 2D-3D Learning”, under review.

M. Shan, **Q. Feng**, Y. Jau and N. Atanasov, “ELLIPSDF: Optimizing Encoding and Similarity Transformation of Object Signed Distance Functions from Multi-view RGB-D Sequences”, IEEE/CVF International Conference on Computer Vision (ICCV), 2021.

T. Zhao, **Q. Feng**, S. Jadhav and N. Atanasov, “CORSAIR: Convolutional Object Retrieval and Symmetry-AIded Registration”, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021.

Q. Feng and N. Atanasov, “Mesh Reconstruction from Aerial Images for Outdoor Terrain Mapping Using Joint 2D-3D Learning”, IEEE International Conference on Robotics and Automation (ICRA), 2021.

Q. Feng and N. Atanasov, “Fully Convolutional Geometric Features for Category-level Object Alignment”, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020.

M. Shan, **Q. Feng** and N. Atanasov, “OrcVIO: Object residual constrained Visual-Inertial Odometry”, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020.

Q. Feng, Y. Meng, M. Shan and N. Atanasov, “Localization and Mapping using Instance-specific Mesh Models”, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019.

Q. Feng, Y. Meng and N. Atanasov, “Dense Spatial Segmentation from Sparse Semantic Information”, Workshop on Learning and Inference in Robotics at Robotics: Science and Systems

(RSS), 2018.

O. Atan, W. R. Zame, **Q. Feng** and M. van der Schaar, “Constructing effective personalized policies using counterfactual inference from biased data sets with many features”, Machine Learning, 2018.

H. Chen, X. Pei, Z. Zhang, D. Yao, **Q. Feng**, Z. Wang, “Driving Behavior Differences between Crash-involved and Crash-not-involved Drivers using Urban Traffic Surveillance Data”, IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI), 2016.

ABSTRACT OF THE DISSERTATION

3D Semantic Scene Understanding and Reconstruction

by

Qiaojun Feng

Doctor of Philosophy in Electrical Engineering (Intelligent Systems, Robotics and Control)

University of California San Diego, 2023

Professor Nikolay Atanasov, Chair

Semantic understanding and reconstruction of the surrounding 3D environment is a necessary requirement for intelligent robots to autonomously fulfill various tasks like environment exploration, surveillance, autonomous driving, indoor household and healthcare service, to name a few. Although the progress on semantic understanding of 2D images is impressive, building a 3D consistent, meaningful yet compact and scalable semantic map for robotics applications is still challenging. In this dissertation, we develop 3D semantic map approaches for robots in different form, including the dense semantic map and the object-level semantic map. We propose TerrainMesh, a dense semantic map in the form of 3D mesh, for the terrain reconstruction from the aerial images and the sparse depth measurements. The joint 2D-3D and geometric-semantic learning framework is proposed to reconstruct the local semantic meshes and the global semantic mesh can be constructed by merging the local meshes with the help of the SLAM algorithm. We investigate the object-level semantic map constructed from 3D measurements. We propose CORSAIR, a retrieval and registration algorithm for point cloud objects. A 3D sparse convolution neural network model is trained to extract the global features for similar shape retrieval and the local per-point features for correspondence generation for pose registration with the help of symmetry. We develop ELLIPSDF, a bi-level object shape model including a coarse level of 3D ellipsoid and a fine level of 3D continuous signed distance function (SDF). We also design the approach to initialize and optimize the object pose together with the bi-level shape from the multiple depth image observations. We also propose the object-level semantic map from the 2D images and investigate its connections with the localization task. We introduce the object mesh model and its observation model of semantic instance segmentation and semantic keypoints. We derive the observation residual function and minimize it to optimize both the object states and the camera poses. We develop OrcVIO, object residual constrained visual-inertial odometry with object ellipsoid and semantic keypoints model. We implement the observation residual model between the ellipsoid and its 2D semantic bounding box and semantic keypoints and connect this with MSCKF framework to implement the online tightly coupled estimation of object and IMU-camera states. The object-level semantic map also provides a meaningful yet efficient representation of the environment. Finally we discuss the potential directions to further extend the 3D semantic understanding technique for robotics.

Chapter 1

Introduction

1.1 Motivation

Human beings have always worked hard to build tools to make our lives better. Robots, which are expected to be highly autonomous and intelligent helpers, are definitely among the most advanced tools that humans expect to have. Since the word ”robot” was first introduced by a Czech writer Karel Čapek in his play *R.U.R.*, or *Rossum's Universal Robots* in 1921, there has been a phenomenal evolution on the robot's functionality and popularity in the past 100+ years. Undoubtedly, robots can take on many dirty, dangerous, and dull jobs now. However, we are not seeing that many of robots appearing in our daily life yet, besides some robot vacuums. Robots today are still nowhere near the science fiction robot figures like R2-D2. Many robots now can only work on a limited of tasks, in a fixed environment. There are several aspects that the current robots need to improve to get closer to be a versatile agent. In this dissertation, we focus on the semantic understanding and reconstruction of the scene or environment.

There are many industrial robots that can perform assembling task at a high precision in a well-organized and fixed factory environment. In this case, it might be possible to execute the task blindly without any understanding on the environment. However, in many situations, the environment is dynamic. Also, we want to have a robot to be capable to work in different environments, even the environment that it has never visited before. Therefore, it is important to have the ability to perceive and reconstruct the working environment of the robot. Besides the direct measurement from the visual sensors, such as the RGB color and the occupancy information, the semantic information is a higher level of information and has a more important role. Comparing with a binary occupancy map that only indicates whether the space is free or occupied, a semantic map that embed the dense semantic information, such as classifying the road, the building and the vegetation region, or annotating the object category, pose and shape, is definitely more useful. With such semantic representation of the environment, it would be easier for human operator to specify task using more natural-language-like instruction so that more people can benefit from robots without technical prerequisite. Also the robot can generalize to perform more complicated tasks enabled by the semantic map. For example, in environmental monitoring, a real-time semantic map can be useful for wildfire prevention and detection, and traversable route prediction for first responders. In object manipulation, the object category, pose and shape estimation can be useful to generate gripping pose. In safe autonomous navigation scenario, the autonomous car need to detect the other vehicles and pedestrians and share the road safely with them. We believe that the 3D semantic map reconstruction can release the potential of robot to be deployed in our daily life.

1.2 Problem Formulation

We consider the tasks of building the 3D semantic map for mobile robots that are equipped with various sensors. The sensors include the visual sensors like the monocular camera, stereo cameras and depth camera. The 3D LiDAR can also be a good complement to provide direct 3D measurements. The other sensors include the inertial measurement unit (IMU) or the GPS that support the localization tasks. We investigate two forms of semantic map. One is dense semantic map. We consider the 3D mesh representation of the scene with surface information. We want to also have the semantic information attached to the mesh besides the geometric shape. The other is a object-level map. We want to reconstruct a map consist of objects with category label, pose and shape reconstructed.

1.3 Related Work

In early days, researchers tried to estimate the semantic information of object classes [182, 151], which are mainly described by hand-crafted geometric parameters. The other early attempts on semantic mapping include the object detection and tracking for dynamic 3D scene analysis [102] and indoor spatial and semantic hierarchical representation recovery [56]. The computer vision technique made a great progress because of the large datasets like ImageNet [38] and the deep learning technique [63]. The object detection [150, 71] and semantic segmentation [169, 23] on 2D images has reached an impressive level of accuracy. Semantic object detection [212, 201] and segmentation [143, 170, 80] can also be performed on 3D data. Such stronger front-end perception ability largely improve the robot's ability to reconstruct semantic map. The semantic map reconstruction is used for robotics applications such as autonomous driving [180, 140, 145], household [157, 113, 142], health care [44], agriculture [119, 160] and disaster first response [132, 154]. For more related work on semantic mapping for robotics, the two recent surveys [98, 57] cover a more comprehensive list of works.

1.4 Contributions

The high-level goal of the dissertation is to develop the technique of building 3D semantic map to enhance the robot's understanding of the environment. We specifically focus on two forms of semantic map, the dense semantic map and the object-level semantic map.

In Chapter 2, we focus on the dense semantic mesh map. We introduce TerrainMesh, a metric-semantic terrain mesh reconstruction algorithm using the RGB image and sparse depths. We propose a joint 2D-3D learning method for metric-semantic mesh reconstruction using a novel 2-step coarse-to-fine strategy, composed of mesh initialization and mesh refinement stages. In the initialization stage, we use only the sparse depth measurements to fit a coarse mesh surface. In the refinement stage, we extract deep convolutional 2D image features and associate them with the initial mesh 3D vertices through perspective projection. The mesh is subsequently refined using a graph convolution model to predict both spatial coordinates and semantic features residuals of the vertices. Employing our TerrainMesh method in combination with feature- and keyframe-based odometry techniques allows reconstruction of global dense metric-semantic mesh models with utility in environmental monitoring and semantic navigation applications.

In Chapter 3, we focus on building the object-level semantic map using 3D measurements. We introduce CORSAIR, a fully Convolutional Object Retrieval and Symmetry-AIded Registration algorithm. We design a sparse fully convolutional network to jointly regress global and local point-cloud features, which are hierarchically correlated. The global feature enables similar model retrieval, while the local features allow object pose registration. We also introduce ELLIPSDF, an expressive yet compact bi-level Ellipsoid and Signed Distance Function model of object pose and shape, and an associated optimization algorithm to infer an object-level map from 3D measurements. These methods enable high-fidelity object-level map with object category label, pose and shape information included.

In Chapter 4, we focus on building the object-level semantic map using sequencial 2D RGB images without direct 3D measurement. We introduce an object mesh model and develop its observation model that generates the semantic instance segmentation and semantic keypoints leveraging differentiable mesh renderer. The observation model is used to optimize both the object states and the camera poses. We introduce OrcVIO, Object residual constrained Visual Inertial Odometry. We model the objects as 3D ellipsoids with coarse ellipsoid shape and fine semantic-keypoint shape. We define residuals relating object states and IMU-camera states to object semantic keypoint and bounding-box detections. The observation residuals are used under the Multi-State Constraint Kalman Filter (MSCKF) framework to implement online tightly coupled estimation of object and IMU-camera states. We have coupled the object-level map reconstruction and the camera pose refinement tasks and get a better estimation of both the ego state and the environment layout.

Chapter 2

Metric-Semantic Terrain Reconstruction from Aerial Images

Recent advances in sensing, computation, storage and communication hardware have set the stage for mobile robot systems to impact environmental monitoring, security and surveillance, agriculture, and many other applications. Constructing terrain maps onboard an unmanned aerial vehicle (UAV) using online sensor measurements provides critical situational awareness in such applications. This chapter considers the problem of building a metric-semantic terrain model, represented as a triangular mesh, of an outdoor environment using a sequence of overhead RGB images obtained onboard a UAV. Fig. 2.1 shows an example of inputs and mesh reconstruction.

We assume that the UAV is running a localization algorithm, based on visual-inertial odometry (VIO) [144] or simultaneous localization and mapping (SLAM) [18], which estimates its camera pose and the depths of a sparse set of tracked image keypoints. However, range sensors and, hence, dense depth information are not available during outdoor flight. One approach for terrain mapping is to recover depth images at each camera view using dense stereo or multi-view stereo (MVS) matching, fuse them to generate a point cloud, and triangulate a mesh surface. Multi-view stereo (MVS) [54] aims to estimate the depth at one frame using several different frames. Classical MVS methods perform patch matching with photometric and geometric consistency [165]. These methods generalize well although the performance can be affected by low texture, lighting variation, and occlusion. Recently, learning-based methods [177, 174, 164]

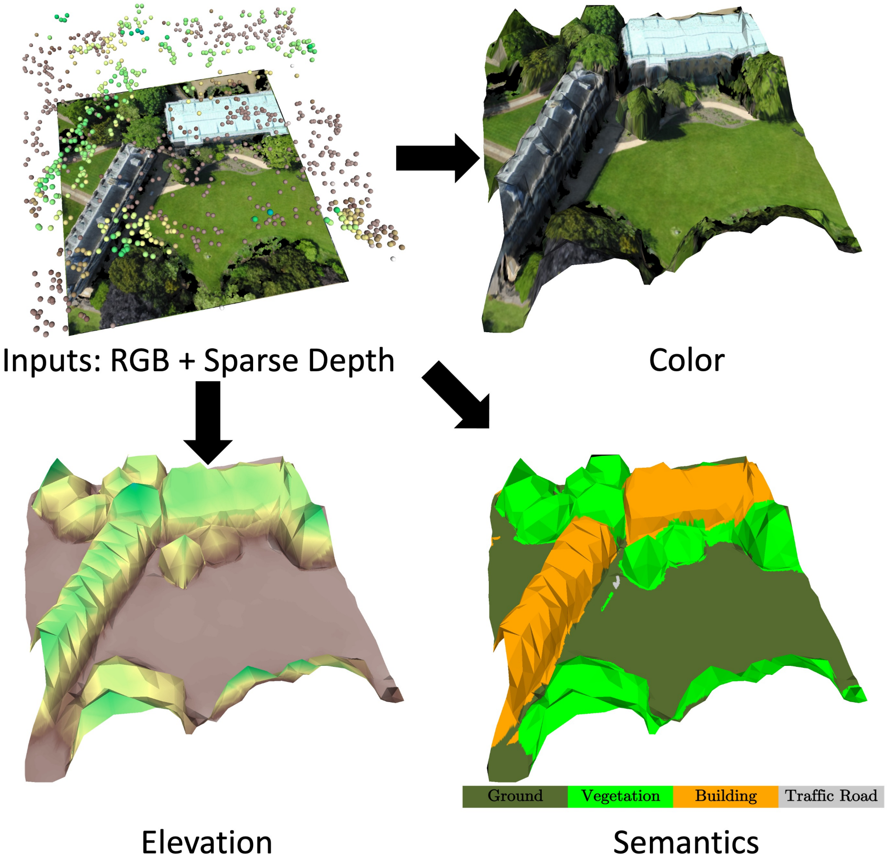

Figure 2.1. This chapter develops a method using aerial RGB images and sparse depth measurements (top-left) to reconstruct a semantic mesh of an outdoor terrain. The color, elevation, and semantics of the mesh are visualized in the top-right, bottom-left and bottom-right plots.

that fuse multi-view learned features across frames for depth recovery have achieved excellent performance. The learning-based MVS can tackle some of the challenges like severe occlusion, but generally labeled data is needed for training. While specialized sensors and algorithms exist for real-time stereo matching, they are restricted to a limited depth range, much smaller than the distances commonly present in aerial images. Apart from MVS, depth completion, the task of reconstructing a dense depth image from an RGB image with given sparse depth estimates, have shown promising performance on indoor [175] and outdoor datasets [59]. Ma et al. [115, 114] develop a deep network for depth completion that passes the sparse depth and RGB image inputs through convolution layers, ResNet encoder layers, transposed convolution decoder layers, and a 1x1 convolution filter. The model is trained either with supervision from ground-truth depth images or via photometric error self-supervision from calibrated RGB image pairs. Instead of consuming sparse depth images directly, Chen et al. [26] pre-process sparse depth images by generating a Euclidean distance transform of the keypoint locations and a nearest-neighbor depth fill map. The authors propose a multi-scale deep network that treats depth completion as residual prediction with respect to the nearest-neighbor depth fill maps. CodeVIO [214] uses a Conditional Variational Autoencoder (CVAE) to encode RGB and sparse depth inputs into a latent depth code and decode a dense depth image from the latent depth code. The sparse depth measurements are used to perform incremental depth code updates, allowing the depth reconstruction to be coupled with visual odometry estimation in the MSCKF filter [121]. However, aerial images are different from ground RGBD images used to train these models. Due to the limited availability of aerial image datasets for supervision, learning-based methods have not yet been widely adopted for outdoor terrain mapping.

Online terrain mapping requires efficient storage and updates of a 3D surface model. Storing dense depth information from aerial images requires significant memory and subsequent model reconstruction. An explicit surface representation using a polygonal mesh may be quite memory and computationally efficient but the vertices and faces need to be optimized to fit the environment geometry. FLaME [65] performs variational optimization over a time-varying Delaunay graph to obtain an inverse-depth mesh of the environment, using sparse depth measurements from a VIO algorithm. Voxblox [135] incrementally builds a voxel-based truncated signed distance field and can reconstruct a mesh as a post-processing step using the Marching Cubes algorithm [110]. Terrain Fusion [191] performs real-time terrain mapping by generating digital surface model (DSM) meshes at selected keyframes. The local meshes are converted into grid-maps and merged using multi-band fusion. Recently, learning methods have emerged as a promising approach for mesh reconstruction from limited or no 3D information. Bloesch et al. [11] propose a learning method to regress the image coordinates and depth of mesh vertices in a decoupled manner. This allows an in-plane 2D mesh to capture the image structure. Pixel2Mesh [190] treats a mesh as a graph and applies graph convolution [97] for vertex feature extraction and graph unpooling to subdivide the mesh for refinement. Using differentiable mesh rendering [93, 109], the 3D mesh structure of an object can be learned from 2D images [92, 50, 171]. Mesh R-CNN [61] simultaneously detects objects and reconstructs their 3D mesh shape. A coarse voxel representation is predicted first and then converted into a mesh for refinement. Recent works [131, 192] can generate mesh reconstructions of complete scenes, including object and human meshes and their poses, from a single RGB image.

Recently, there has also been increasing interest in supplementing geometric reconstruction with semantic information because many robotics tasks require semantic understanding. Semantic segmentation on the 2D images can be back-projected onto 3D space and multi-view information can be fused to annotate the 3D structure [185, 152]. Besides, semantic segmentation can be directly performed on the 3D point cloud [79] or the mesh [81]. Instead of treating the semantic annotation as the post-processing step after geometric reconstruction, researchers also investigate on how to jointly optimize geometric and semantic accuracy. Häne et al. [85] formulate a joint segmentation and dense reconstruction problem on voxels and show that appearance likelihoods and class-specific geometric priors help each other. Cherabier et al. [27] leverage variational energy minimization method for regularization to capture complex dependencies between the semantic labels and the 3D geometry. Guo et al. [67] jointly optimize the geometry and semantics by predicting the implicit neural representations of the signed distance, color and semantic field. However, few algorithms exist for joint metric-semantic reconstruction. Most works treat semantic classification as a post-processing step, decoupling it from 3D geometric reconstruction.

This chapter is based on the papers [48, 49]. In this chapter, we propose a joint 2D-3D learning method for metric-semantic mesh reconstruction using a novel coarse-to-fine strategy, composed of mesh initialization and mesh refinement stages. In the initialization stage, we use only the sparse depth measurements to fit a coarse mesh surface. In the refinement stage, we extract deep convolutional 2D image features and associate them with the initial mesh

3D vertices through perspective projection. The mesh is subsequently refined using a graph convolution model to predict both spatial coordinates and semantic features residuals of the vertices. We conduct extensive evaluation on simulated and real aerial datasets. We demonstrate empirically that the joint geometric-semantic training can outperform the geometric-only method. In summary, the **contributions** of this chapter are summarized as follows.

- We introduce a joint 2D-3D loss function, utilizing differentiable mesh rendering, for metric-semantic mesh reconstruction.
- We develop a two-stage coarse-to-fine mesh reconstruction approach, using a closed-form mesh vertex *initialization* from sparse depth measurements and a graph convolution network mesh vertex *refinement* from RGB, sparse depth measurements and semantic image features.
- We evaluate our metric-semantic mesh reconstruction algorithm on synthetic and photo-realistic aerial image datasets.

2.1 Problem: Semantic Mesh Reconstruction from RGB Images and Sparse Depth

Consider a UAV equipped with an RGB camera flying outdoor. Let $\mathbf{I}$ denote an RGB image. Obtaining dense depth images during outdoor flight is challenging due to the large distances and relative small variation. However, a VIO or SLAM algorithm can track and estimate the depth of a sparse set of image feature points. Let $\mathbf{D}^s$ be a *sparse* 2D matrix that contains estimated depths at the image feature locations and zeros everywhere else. Let $\mathbf{D}$ denote the *dense* ground-truth depth image. Let $\mathbf{S}$ denote an associated ground-truth semantic segmentation image. Assuming there are s semantic classes in total, we model $\mathbf{S}$ as a tensor with the same width and height as the RGB image $\mathbf{I}$ and third channel size s. Each element $\mathbf{S}_{i,j} \in [0,1]^s$ is a one-hot vector with 0s in all elements, except for a single 1 indicating the true semantic class.

Our goal is to construct an explicit model of the camera view using a 3D semantic triangle mesh $\mathcal{M} := (\mathbf{V}, \mathbf{C}, \mathcal{E}, \mathcal{F})$, where $\mathbf{V} \in \mathbb{R}^{n\times 3}$ are the vertex spatial coordinates, $\mathbf{C} \in \mathbb{R}^{n\times s}$ are the vertex semantic features, $[n] := \{1, \ldots, n\}$ is the set of vertex indices, $\mathcal{E} \subseteq [n] \times [n]$ are the edges, and $\mathcal{F} \subseteq [n] \times [n] \times [n]$ are the faces. Each row of the matrix $\mathbf{C}$ contains an unnormalized score vector for the s classes that can be converted into a probability distribution over the s classes using the softmax function [16].

Problem. *Given a finite set of RGB images $\{\mathbf{I}_k\}_k$ and corresponding sparse depth measurements $\left\{\mathbf{D}_k^s\right\}_k$, define a semantic mesh reconstruction function $\mathcal{M} = f(\mathbf{I}, \mathbf{D}^s; \theta)$ and optimize its parameters θ to fit the ground-truth depth $\{\mathbf{D}_k\}_k$ and semantic segmentation $\{\mathbf{S}_k\}_k$ images:*

$$\min_{\theta} \sum_{k} \ell(f(\mathbf{I}_k, \mathbf{D}_k^s; \theta); \mathbf{D}_k, \mathbf{S}_k) \tag{2.1}$$

where $\ell(\mathcal{M}; \mathbf{D}, \mathbf{S})$ is a loss function measuring the error between a 3D semantic mesh $\mathcal{M}$ and a depth image $\mathbf{D}$ plus a semantic image $\mathbf{S}$.

The choice of loss function ℓ is discussed in Sec. 2.2. We develop a machine learning approach consisting of an offline training phase and an online mesh reconstruction phase. During training, the parameters θ are optimized using a training set $\mathcal{D} := \left\{\mathbf{I}_k, \mathbf{D}_k^s, \mathbf{D}_k, \mathbf{S}_k\right\}_k$ with known ground-truth depth images and semantic segmentation images. During testing, given streaming RGB images $\mathbf{I}$ and sparse depth measurements $\mathbf{D}^s$, the optimized parameters θ^* are used in the model $f(\mathbf{I}, \mathbf{D}^s; \theta^*)$ to reconstruct the mesh vertex spatial coordinates $\mathbf{V}$ and semantic features $\mathbf{C}$. The mesh edges $\mathcal{E}$ and faces $\mathcal{F}$ are assumed fixed and known, and hence are not reconstructed by the model. For notational simplicity, we write the output of model f directly as the semantic mesh $\mathcal{M} = f(\mathbf{I}, \mathbf{D}^s; \theta^*)$. A keyframe-based VIO or SLAM algorithm estimates the positions $\mathbf{p}$ and orientations $\mathbf{R}$ of camera keyframes as well as the depth of sparse keypoint measurements associated with each keyframe. Our approach estimates a local mesh $\mathcal{M} = (\mathbf{V}, \mathbf{C}, \mathcal{E}, \mathcal{F})$ at each camera keyframe. The keyframe meshes can be converted to a global frame (with vertex coordinates $\mathbf{V}\mathbf{R}^\top + \mathbf{1}\mathbf{p}^\top$, where $\mathbf{1}$ is a $n \times 1$ vector filled with 1) and fused to obtain a complete consistent metric-semantic model of the environment.

2.2 Loss Functions for Semantic Mesh Reconstruction

We develop several loss functions to measure the consistency between a semantic mesh $\mathcal{M}$ and corresponding depth image $\mathbf{D}$ and semantic segmentation image $\mathbf{S}$. Since our problem focuses on optimizing the mesh, the loss function must be differentiable with respect to the mesh vertex spatial coordinates $\mathbf{V}$ and semantic features $\mathbf{C}$. We keep the mesh edges $\mathcal{E}$ and faces $\mathcal{F}$ fixed during the mesh optimization.

A loss function can be defined in the 2D image plane by rendering a depth image from $\mathcal{M}$ and comparing it with $\mathbf{D}$. The differentiable mesh renderer [109, 149] makes the 3D mesh rendering, e.g., from a 3D mesh to a 2D image, differentiable. Therefore, we can back-propagate the loss measured on the 2D images to the 3D mesh vertices. We leverage a differentiable mesh renderer to generate a depth image $\rho_D(\mathcal{M})$ and define a 2D loss function:

$$\ell_2(\mathcal{M}, \mathbf{D}) := \text{mean}(|\rho_D(\mathcal{M}) - \mathbf{D}|), \tag{2.2}$$

where $\text{mean}(\cdot)$ is a function taking the mean over all the valid pixels where both $\mathbf{D}$ and $\rho_D(\mathcal{M})$ have a depth value.

While ℓ_2 is a natural choice of a loss function in the image plane, it does not emphasize two important properties for mesh reconstruction. First, since ℓ_2 only considers a region in the image plane where both depth images have valid information, its minimization over $\mathcal{M}$ may encourage the mesh $\mathcal{M}$ to shrink to cover only a smaller image region. Second, ℓ_2 does not emphasize regions of large depth gradient variation (e.g., the side surface of a building), which may lead to inaccurate 3D reconstruction. To address these limitations, we define an additional loss function in the 3D spatial domain using two point clouds $\mathcal{P}_{\mathcal{M}}$ and $\mathcal{Q}_{\mathbf{D}}$ obtained from $\mathcal{M}$

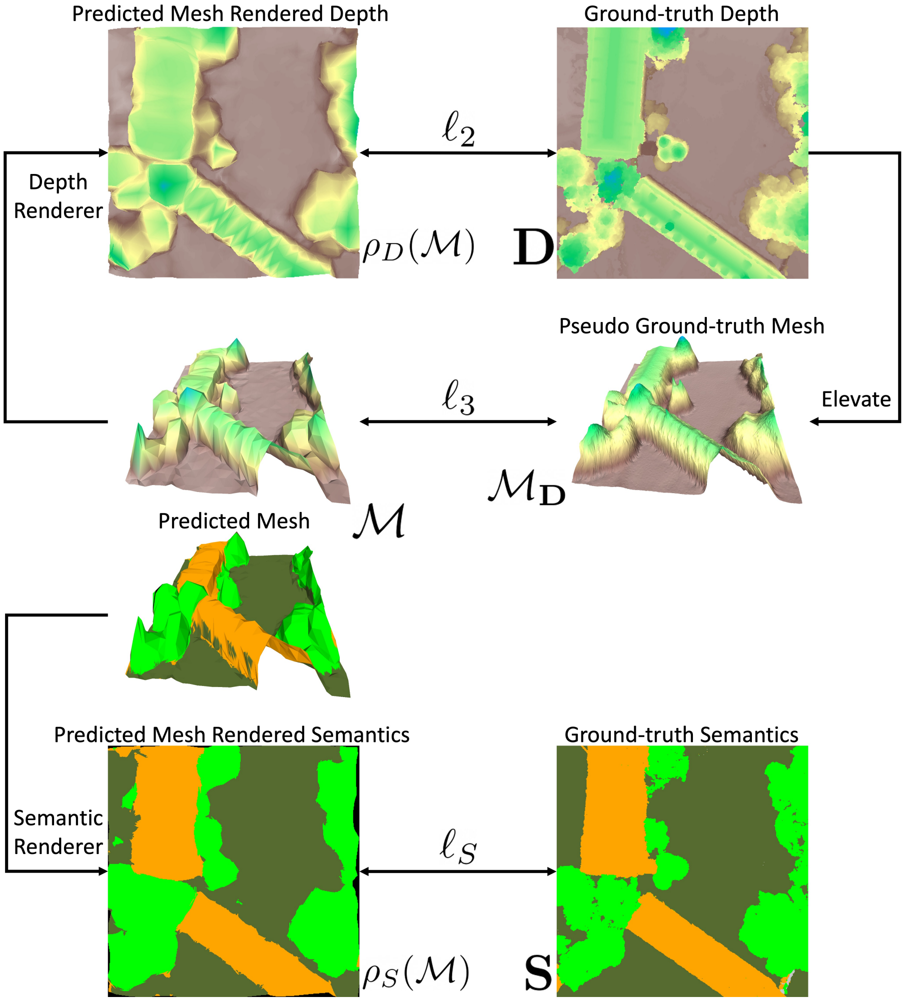

Figure 2.2. Loss function illustration: ℓ_2 compares rendered mesh depth $\rho_D(\mathcal{M})$ to a depth image $\mathbf{D}$, ℓ_3 compares a mesh $\mathcal{M}$ to an elevated mesh $\mathcal{M}_{\mathbf{D}}$ obtained from a depth image, and ℓ_S compares a rendered mesh semantic image $\rho_S(\mathcal{M})$ to a semantic segmentation image $\mathbf{S}$.

and $\mathbf{D}$, respectively:

$$\ell_3(\mathcal{M},\mathbf{D}) := \frac{1}{2}d(\mathcal{P}_{\mathcal{M}},\mathcal{Q}_{\mathbf{D}}) + \frac{1}{2}d(\mathcal{Q}_{\mathbf{D}},\mathcal{P}_{\mathcal{M}}), \tag{2.3}$$

where d is the asymmetric Chamfer point cloud distance [8]:

$$d(\mathcal{P},\mathcal{Q}) := \frac{1}{|\mathcal{P}|}\sum_{\mathbf{p}\in\mathcal{P}}\min_{\mathbf{q}\in\mathcal{Q}}\|\mathbf{p}-\mathbf{q}\|_2^2. \tag{2.4}$$

Note that squared Euclidean distance is used when calculating the Chamfer distance. To generate $\mathcal{P}_{\mathcal{M}}$, we sample on the faces of $\mathcal{M}$ uniformly using PyTorch3D library [149]. The loss function is differentiable with respect to the mesh vertices because the samples on the mesh faces can be represented as linear combinations of the mesh vertices using the barycentric coordinate introduced in Sec. 2.3.1. To generate $\mathcal{Q}_{\mathbf{D}}$, we may sample the depth image $\mathbf{D}$ uniformly and project the samples to 3D space but this will not generate sufficient samples in the regions of large depth gradient variation. Instead, we first generate a pseudo ground-truth mesh $\mathcal{M}_{\mathbf{D}}$ by densely sampling pixel locations in $\mathbf{D}$ as the mesh vertices and triangulating on the image plane to generate faces. We then sample the surface of $\mathcal{M}_{\mathbf{D}}$ uniformly to obtain $\mathcal{Q}_{\mathbf{D}}$. The sample number is set as 10000.

We also define two regularization terms to measure the smoothness of the mesh $\mathcal{M}$. The first is based on the Laplacian matrix $\mathbf{L} := \mathbf{G} - \mathbf{A} \in \mathbb{R}^{n \times n}$ of $\mathcal{M}$, where $\mathbf{G}$ is the vertex degree matrix and $\mathbf{A}$ is the adjacency matrix. We define a vertex regularization term based on the $\ell_{2,1}$-norm [130] of the degree-normalized Laplacian [173] $\mathbf{L}_n = \mathbf{G}^{-1}\mathbf{L} = \mathbf{I}_n - \mathbf{G}^{-1}\mathbf{A}$ where $\mathbf{I}_n$ is an identity matrix of size $n \times n$:

$$\ell_{\mathbf{V}}(\mathcal{M}) := \frac{1}{n} \|\mathbf{L}_n \mathbf{V}\|_{2,1}, \tag{2.5}$$

where n is the number of vertices. We also introduce a mesh edge regularization term to discourage long edges in the mesh

$$\ell_{\mathcal{E}}(\mathcal{M}) := \frac{1}{|\mathcal{E}|} \sum_{(i,j) \in \mathcal{E}} \|\mathbf{v}_i - \mathbf{v}_j\|_2, \tag{2.6}$$

where $\mathbf{v}_i \in \mathbb{R}^3$ are the coordinates of the i-th mesh vertex.

We also define a semantic loss function that relates the 3D mesh semantic information to the 2D semantic segmentation image by rendering the semantic mesh similar to (2.2). We define a differentiable semantic rendering function $\rho_S(\mathcal{M})$ which can generate a same-sized image as $\mathbf{S}$ with s channels, where s is the number of semantic classes. At each pixel, the s-dimensional vector stores the unnormalized scores representing the likelihoods of the s classes. We use a softmax function [16] $\sigma_i(\mathbf{x}) = \exp(\mathbf{x}_i)/\sum_{j=1}^{s}\exp(\mathbf{x}_j)$ to compute the probability distribution over the s classes $\sigma(\rho_S(\mathcal{M}))$. For the semantic segmentation task, we choose the Dice loss [41] :

$$\ell_{\mathbf{S}}(\mathcal{M},\mathbf{S}) := -\frac{2|\sigma(\rho_S(\mathcal{M}))\cdot\mathbf{S}|}{|\sigma(\rho_S(\mathcal{M}))|+|\mathbf{S}|}, \tag{2.7}$$

where $|\cdot|$ sums up all the absolute values of the elements. Note that $\mathbf{S}$ contains one-hot vectors while $\sigma(\rho_S(\mathcal{M}))$ stores probability vectors for the s classes. Therefore, $|\sigma(\rho_S(\mathcal{M}))\cdot\mathbf{S}|$ is the probabilistic intersection between two semantic segmentation images. In Sec. 2.4.7, we compare the Dice loss with three alternative semantic loss functions (cross-entropy loss, focal loss, and Jaccard loss). Finally, we apply Laplacian smoothing (2.5) to the vertex semantic features:

$$\ell_{\mathbf{C}}(\mathcal{M}) := \frac{1}{n}\|\mathbf{L}_n\mathbf{C}\|_{2,1}\,. \tag{2.8}$$

The complete loss function is:

$$\begin{aligned}\ell(\mathcal{M},\mathbf{D},\mathbf{S}) :=& w_2\ell_2(\mathcal{M},\mathbf{D}) + w_3\ell_3(\mathcal{M},\mathbf{D}) + w_{\mathbf{S}}\ell_{\mathbf{S}}(\mathcal{M},\mathbf{S}) \\ &+ w_{\mathbf{V}}\ell_{\mathbf{V}}(\mathcal{M}) + w_{\mathcal{E}}\ell_{\mathcal{E}}(\mathcal{M}) + w_{\mathbf{C}}\ell_{\mathbf{C}}(\mathcal{M})\end{aligned} \tag{2.9}$$

where the first two terms evaluate the error between $\mathcal{M}$ and $\mathbf{D}$, the following two terms encourage smoothness of the mesh structure, and the last two terms evaluate the error between $\mathcal{M}$ and $\mathbf{S}$ and regularize the semantic features, which affects both the geometric and semantic properties of the mesh. The scalars $w_2, w_3, w_{\mathbf{V}}, w_{\mathcal{E}}, w_{\mathbf{S}}, w_{\mathbf{C}} \in \mathbb{R}_{\geq 0}$ allow appropriate weighting of the different terms in (2.9). Fig. 2.2 illustrates the loss functions ℓ_2 in (2.2), ℓ_3 in (2.3), and ℓ_S in (2.7).

2.3 2D-3D Learning for Semantic Mesh Reconstruction

Inspired by depth completion techniques, we approach mesh reconstruction in two stages: *initialization* and *refinement*. In the initialization stage, we generate a mesh from the sparse depth measurements alone (Sec. 2.3.1). In the refinement stage, we optimize the mesh vertex

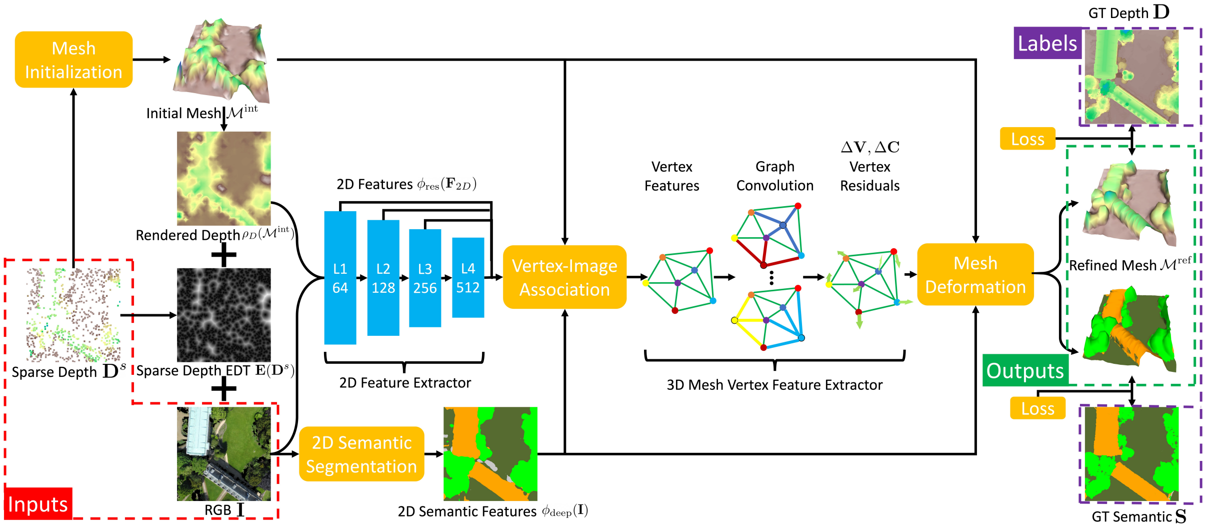
Mesh Initialization
Initial Mesh $\mathcal{M}^{\text{int}}$
Rendered Depth $\rho_D(\mathcal{M}^{\text{int}})$
Sparse Depth $\mathbf{D}^s$
Sparse Depth EDT $\mathbf{E}(\mathbf{D}^s)$
RGB $\mathbf{I}$
Inputs
2D Features $\phi_{\text{res}}(\mathbf{F}_{2D})$
L1 64
L2 128
L3 256
L4 512
2D Feature Extractor
Vertex-Image Association
2D Semantic Segmentation
2D Semantic Features $\phi_{\text{deep}}(\mathbf{I})$
Vertex Features
Graph Convolution
$\Delta\mathbf{V}, \Delta\mathbf{C}$
Vertex Residuals
3D Mesh Vertex Feature Extractor
Mesh Deformation
GT Depth $\mathbf{D}$
Labels
Loss
Refined Mesh $\mathcal{M}^{\text{ref}}$
Outputs
Loss
GT Semantic $\mathbf{S}$

Figure 2.3. Overview of our semantic mesh reconstruction architecture. In the initialization stage (Sec. 2.3.1), we use sparse depth to elevate a flat mesh from the image plane to 3D space (Fig. 2.4). In the refinement stage (Sec. 2.3.2, 2.3.3), we first combine the RGB image, a depth image rendered from the initial mesh, and a Euclidean Distance Transform of the sparse depth measurements to extract features using a 2D feature extractor. We have a 2D semantic segmentation model to generate 2D semantic features from the RGB image. The 2D features and the 2D semantic features are associated with the mesh vertices using camera projection at different stages (Fig. 2.5,2.6). The vertex spatial coordinates and the vertex semantic features, are regressed using graph convolution network (GCN) over the mesh. The refined output is a metric-semantic mesh (Fig. 2.7). The 2D feature extractor and GCN parameters are optimized jointly using the loss function in Sec. 2.2. The 2D semantic segmentation model is trained separately.

coordinates based on RGB image features (Sec. 2.3.2) and assign semantic categories to each vertex using image segmentation features (Sec. 2.3.3). An overview of our semantic mesh reconstruction model $\mathcal{M} = f(\mathbf{I}, \mathbf{D}^s; \theta)$ is shown in Fig. 2.3.

2.3.1 Mesh Initialization

Outdoor terrain structure can be viewed as a 2.5-D surface with height variation. Hence, we initialize a flat mesh surface and change the surface elevation based on the sparse depth measurements. The flat mesh is initialized with regular-grid vertices ($n = 1024$ in our experiments) over the image plane, and the edges and the faces connecting the vertices. See Fig. 2.4 for an illustration. Subsequently, our mesh reconstruction approach only optimizes the mesh vertices

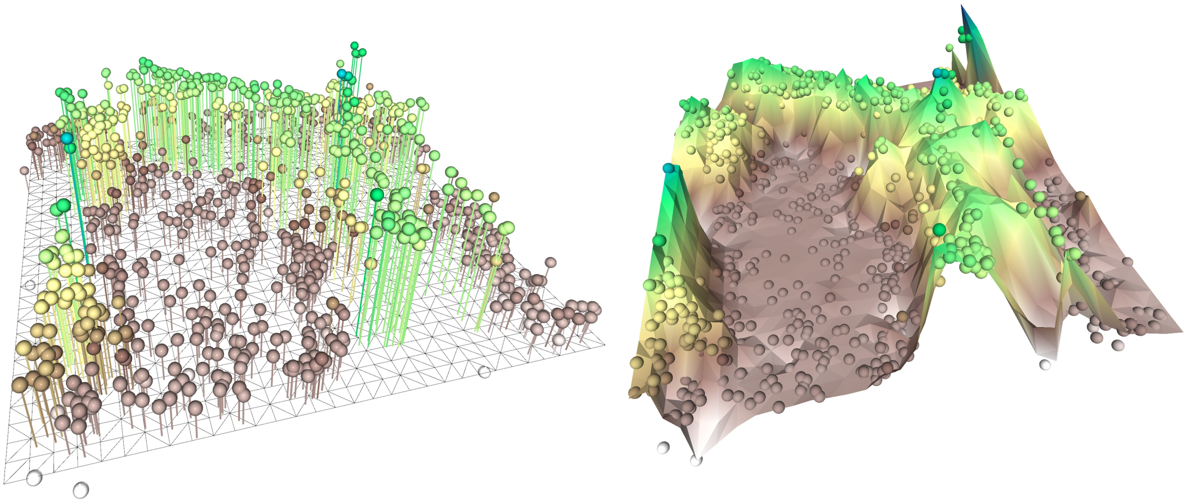

Figure 2.4. Mesh initialization stage (Sec. 2.3.1). Left: Sparse depth measurements (color dots) are used to determine vertex heights from a flat image-plane mesh (bottom wireframe). Right: The initialized mesh. Colors indicate elevation.

and keeps the edge and face topology fixed. The initialized mesh $\mathcal{M}^{\text{int}} = (\mathbf{V}^*, \mathbf{0})$ is used as an input to the mesh refinement stage, described in Sec. 2.3.2, 2.3.3. Since we do not update $\mathcal{E}, \mathcal{F}$, we will omit them for simplicity.

We constrain the mesh vertex deformation to the z-axis to change the vertex heights only. The coordinates of the i-th vertex of the flat mesh, $\mathbf{v}_i = [v_i^x, v_i^y, 1]$, are divided by a scalar inverse depth λ_i to obtain the i-th vertex coordinates $[v_i^x/\lambda_i, v_i^y/\lambda_i, 1/\lambda_i]$ of the initialized mesh. We concatenate λ_i to obtain a vector $\lambda \in \mathbb{R}^n$ of all vertex inverse depths.

Any point $\mathbf{p}$ on the mesh surface that lies in a specific triangle can be represented as a convex combination $\mathbf{p} = b_i\mathbf{v}_i + b_j\mathbf{v}_j + b_k\mathbf{v}_k$ of the triangle vertices $\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k$ with weights $b_i, b_j, b_k \in [0, 1]$ such that $b_i + b_j + b_k = 1$. The vector $[b_i, b_j, b_k]^\top$ is called the *barycentric coordinates* of $\mathbf{p}$. We use barycentric coordinates to relate the sparse depth measurements $\mathbf{D}^s$ to the vertex inverse depths λ, which is equivalent to a linear interpolation.

Let the valid measurements in the sparse depth image $\mathbf{D}^s$ be $\{(i, j), \mathbf{D}^s_{ij}\}$, where (i, j) are the pixel coordinates and $\mathbf{D}^s_{ij}$ are the corresponding depth measurements. Each pixel (i, j) falls within one triangle of the flat 2D mesh (see Fig. 2.4). Let $\mathbf{b}_{ij} \in \mathbb{R}^n$ be the barycentric coordinates of pixel (i, j), where at most three elements of $\mathbf{b}_{ij}$, corresponding to the three triangle vertices, are non-zero. The inverse depth $1/\mathbf{D}^s_{ij}$ is related to the vertex inverse depths λ through the barycentric coordinates [84], $\mathbf{b}_{ij}^\top \lambda = 1/\mathbf{D}^s_{ij}$. Stacking these equations for all valid pixels (i, j) in $\mathbf{D}^s$, we obtain:

$$\mathbf{B}\lambda = \rho, \tag{2.10}$$

where ρ is a vector of the valid inverse depth measurements in $\mathbf{D}^s$ with elements $1/\mathbf{D}^s_{ij}$. Using Laplacian regularization as in (2.5), we formulate a least-squares problem in λ:

$$\lambda^* = \arg\min_{\lambda} \left(\|\mathbf{B}\lambda - \rho\|_2^2 + w'_{\mathbf{V}} \|\mathbf{L}_n \lambda\|_2^2 \right). \tag{2.11}$$

The problem in (2.11) has a closed-form solution:

$$\lambda^* = (\mathbf{B}^\top \mathbf{B} + w'_{\mathbf{V}} \mathbf{L}_n^\top \mathbf{L}_n)^{-1} \mathbf{B}^\top \rho. \tag{2.12}$$

The regularization term, not only makes the initialized mesh smoother, but also guarantees that the solution exists even when the number of sparse depth measurements is smaller than the number of mesh vertices. Since the 2D mesh projection and $\mathbf{L}_n$ are pre-defined, the problem can be solved very efficiently, e.g., in less than 0.1 sec for a mesh with 1024 vertices. Given λ^*, we obtain an initialized mesh $\mathcal{M}^{\text{int}}$ with each vertex coordinate as $[v_i^x/\lambda_i^*, v_i^y/\lambda_i^*, 1/\lambda_i^*]$.

2.3.2 Geometric Mesh Refinement

Initialization using the sparse depth measurements only provides a reasonable mesh reconstruction but many details are missing. In the geometric refinement stage, we use a learning approach to extract features from both the 2D image and 3D initial mesh and regress mesh vertex spatial coordinate residuals. The ground-truth depth maps are used for supervision.

The photometric image information is useful for mesh refinement since man-made objects have sharp vertical surfaces, while natural terrain has noisy but limited depth variation. The sparse depth measurements also provide information about areas with large intensity variation. Inspired by Mesh R-CNN [61], we design a network that extracts features from the 2D image, associates them with the 3D vertices of the initial mesh, and uses them to refine the vertex spatial coordinates. Our network has 3 stages: feature extraction, vertex-image feature association, and vertex graph convolution.

Feature Extraction

We extract features from three sources: the RGB image $\mathbf{I}$, the rendered depth $\rho_D(\mathcal{M}^{\text{int}})$ from the initial mesh, and a Euclidean distance transform (EDT) $\mathbf{E}(\mathbf{D}^s)$ of the sparse depth measurements in the 2D image space, obtained by computing the Euclidean distance to the closest valid depth measurement pixel from each pixel coordinate. The three images are concatenated to form a 5-channel input (3-channels in $\mathbf{I}$, 1-channel in each $\rho_D(\mathcal{M}^{\text{int}})$ and $\mathbf{E}(\mathbf{D}^s)$):

$$\mathbf{F}_{2D} = \text{concat}(\mathbf{I}, \rho_D(\mathcal{M}^{\text{int}}), \mathbf{E}(\mathbf{D}^s)). \tag{2.13}$$

Four layers of features with different resolution and channels are extracted:

$$[\mathbf{L}_1, \mathbf{L}_2, \mathbf{L}_3, \mathbf{L}_4] = \phi_{\text{res}}(\mathbf{F}_{2D}; \theta_{2D}), \tag{2.14}$$

where ϕ_{res} is a ResNet model [72] with parameters θ_{2D}.

Vertex-Image Feature Association

Next, we construct 3D features for the mesh vertices by projecting each vertex to the image plane and interpolating the 2D image features. This idea is inspired by Pixel2Mesh [190], which projects mesh vertices onto the image plane and extracts features at the projected coordinates. To obtain multi-scale features, we associate the projected mesh vertices with the intermediate layer feature maps $[\mathbf{L}_1, \mathbf{L}_2, \mathbf{L}_3, \mathbf{L}_4]$ from (2.14). The vertex-image association step is illustrated in Fig. 2.5. All features corresponding to different channels are concatenated to

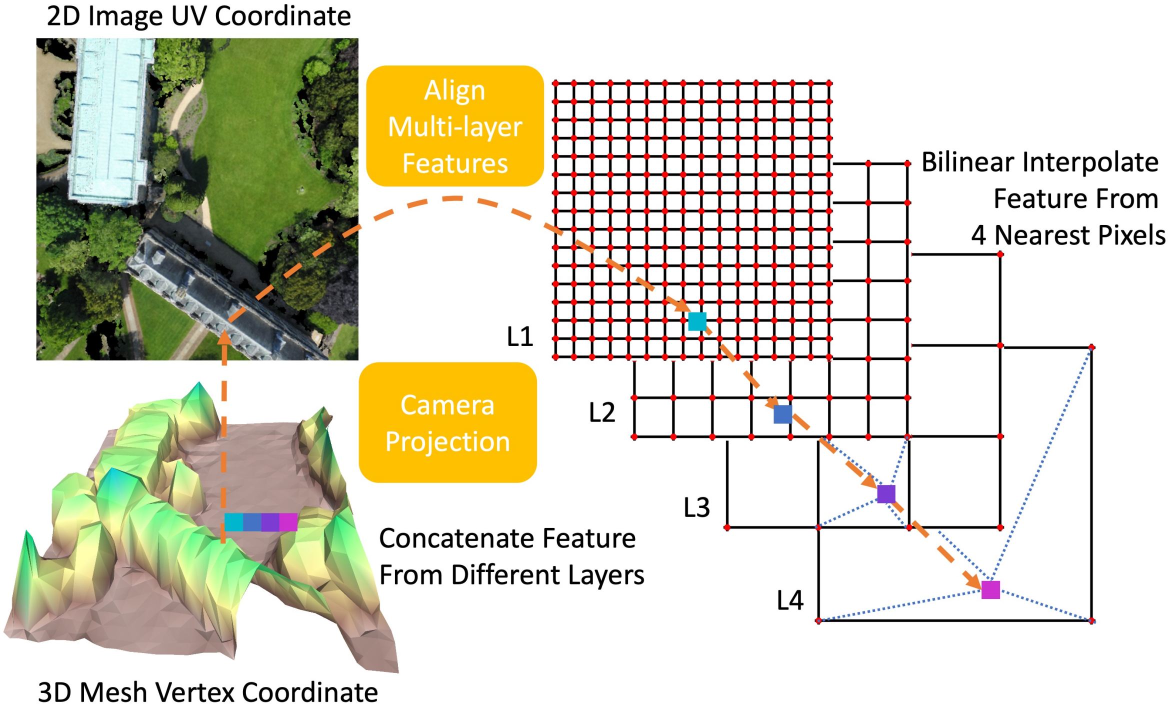
2D Image UV Coordinate
Align Multi-layer Features
Bilinear Interpolate Feature From 4 Nearest Pixels
L1
L2
L3
L4
Camera Projection
Concatenate Feature From Different Layers
3D Mesh Vertex Coordinate

Figure 2.5. Illustration of image feature to mesh vertex association. With known camera intrinsics, each mesh vertex can be projected in uv coordinates (range $[0,1]$) onto the image plane. Bilinear interpolation is used to associate image feature maps at different resolutions with the mesh vertices. The features across different resolutions are concatenated to form a composite vertex feature.

form composite vertex features. We define associate$(\cdot,\cdot)$ as the function that assigns 2D features to 3D mesh vertices:

$$\mathbf{V}_{g_{\text{in}}} = \text{associate}(\mathcal{M}, \phi_{\text{res}}(\mathbf{F}_{2D})), \tag{2.15}$$

where $\mathbf{V}_{g_{\text{in}}} \in \mathbb{R}^{n\times(l_1+l_2+l_3+l_4)}$ are the vertex features and l_i is the number of channels in feature map $\mathbf{L}_i$.

Vertex Graph Convolution

After the feature assignment, the mesh can be viewed as a graph with vertex features $\mathbf{V}_{g_{\text{in}}}$. Using the vertex features, a graph convolution network [97, 61] is a suitable architecture to predict coordinate deformation $\mathbf{\Delta V}$ for the vertex spatial coordinates to optimize the agreement between the refined mesh $\mathcal{M}^{\text{ref}} = (\mathbf{V} + \mathbf{\Delta V})$ and the ground truth depth $\mathbf{D}$ according to the loss in (2.9). To capture a larger region of feature influence, we use 3 layers of graph convolution $g_1^{\mathbf{V}}$,$g_2^{\mathbf{V}}$,$g_3^{\mathbf{V}}$, as follows:

$$\begin{aligned} \mathbf{V}_1^{\text{in}} &= \text{ReLU}(\mathbf{W}_1^{\mathbf{V}}\mathbf{V}_{g_{\text{in}}}) \\ \mathbf{V}_i^{\text{in}} &= \mathbf{V}_{i-1}^{\text{out}}, && i = 2,3, \\ \mathbf{V}_i^{\text{out}} &= \text{ReLU}(g_i^{\mathbf{V}}([\mathbf{V}_i^{\text{in}};\mathbf{V}];\theta_{g\mathbf{V}i})), && i = 1,2,3, \\ \Delta\mathbf{V} &= \mathbf{W}_2^{\mathbf{V}}[\mathbf{V}_3^{\text{out}};\mathbf{V}], \end{aligned} \tag{2.16}$$

where $\Delta\mathbf{V} \in \mathbb{R}^{n\times 3}$ is the matrix of spatial coordinate residuals, $\mathbf{W}_1^{\mathbf{V}}, \mathbf{W}_2^{\mathbf{V}}$ are weight matrices of the linear layers, $\theta_{g\mathbf{V}i}$ are the graph convolution layer weights, and ReLU is the Rectified Linear Unit activation function $\text{ReLU}(x) = \max(0,x)$. The trainable parameters for vertex graph convolution are $\theta_{3D\mathbf{V}} = [\mathbf{W}_1^{\mathbf{V}};\mathbf{W}_2^{\mathbf{V}};\theta_{g\mathbf{V}1};\theta_{g\mathbf{V}2};\theta_{g\mathbf{V}3}]$. It is possible to concatenate more stages of vertex-image feature association and graph convolution. At stage i, the previous stage's refined mesh $\mathcal{M}_{i-1}^{\text{ref}}$ is set as the initial mesh $\mathcal{M}_i^{\text{int}}$ and new vertex features are extracted via vertex-image feature association and fed to new graph convolution layers. All refined meshes at different stages are evaluated using the ground-truth depth map $\mathbf{D}$ using the loss functions defined in (2.9).

2.3.3 Semantic Mesh Reconstruction

To further enrich the environment representation, we introduce semantic information in the mesh reconstruction. By assigning per-vertex semantic features $\mathbf{C}$, we can interpolate the semantic information over the whole mesh using barycentric coordinates (see Fig. 2.6). To obtain a 2D semantic segmentation image from the mesh, we use the differentiable semantic renderer $\rho_{\mathbf{S}}$ introduced in (2.7). Both the vertex spatial coordinates $\mathbf{V}$ and the semantic features $\mathbf{C}$ can affect the rendered 2D semantic segmentation image $\rho_{\mathbf{S}}(\mathcal{M})$. Hence, by optimizing the semantic loss in (2.7), we can refine both the semantic features and the geometric structure of the mesh.

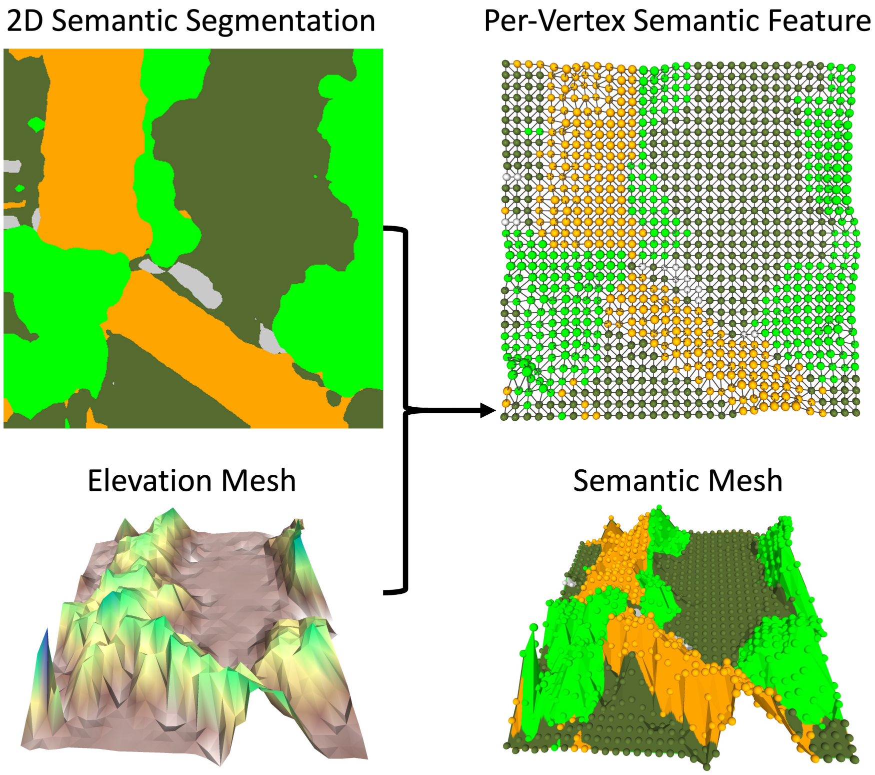

Figure 2.6. A 2D semantic segmentation feature map (top left) is used to generate semantic features for the 3D elevation mesh vertices (bottom left). Each mesh vertex is projected to the semantic segmentation feature map to retrieve an associated semantic feature (top right). Dense semantic features over the whole mesh can be obtained by interpolation on the mesh faces (bottom right).

We first obtain 2D semantic segmentation features $\phi_{\text{deep}}(\mathbf{I};\theta_{2Dsem})$ from the RGB image $\mathbf{I}$ using the DeepLabv3 model [24] with parameters θ_{2Dsem}. Then, we associate the mesh vertices to the 2D semantic feature map to get initial mesh vertex semantic features:

$$\mathbf{C} = \text{associate}(\mathcal{M}, \phi_{\text{deep}}(\mathbf{I})). \tag{2.17}$$

Fig. 2.6 illustrates the mesh vertex association with respect to the 2D semantic segmentation features. In the semantic refinement stage, we regress a semantic residual $\Delta\mathbf{C}$ for the semantic features. We use 3 layers of graph convolution $g_1^{\mathbf{C}}$,$g_2^{\mathbf{C}}$,$g_3^{\mathbf{C}}$. We also use the $\mathbf{V}_{g_{\text{in}}}$ extracted from ResNet in (2.15) as an input to the first graph convolution layer. Additionally, we concatenate

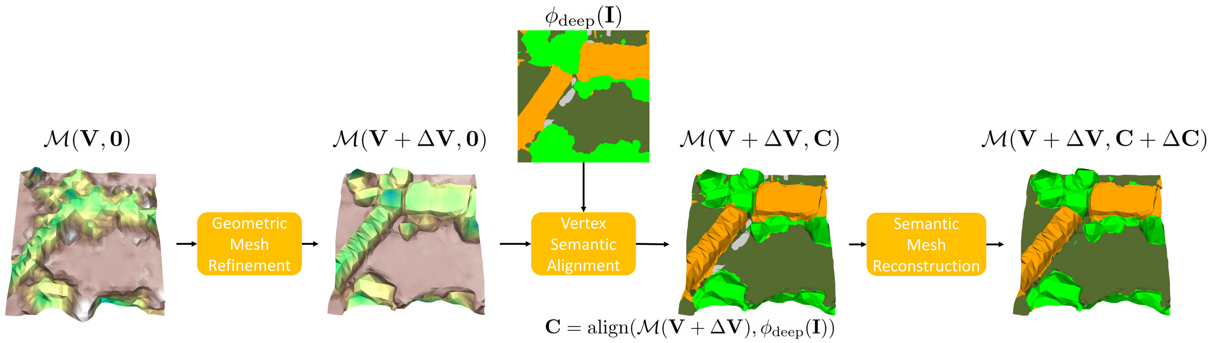
$\phi_{\text{deep}}(\mathbf{I})$
$\mathcal{M}(\mathbf{V},\mathbf{0})$
Geometric Mesh Refinement
$\mathcal{M}(\mathbf{V}+\Delta\mathbf{V},\mathbf{0})$
Vertex Semantic Alignment
$\mathcal{M}(\mathbf{V}+\Delta\mathbf{V},\mathbf{C})$
Semantic Mesh Reconstruction
$\mathcal{M}(\mathbf{V}+\Delta\mathbf{V},\mathbf{C}+\Delta\mathbf{C})$
$\mathbf{C}=\text{align}(\mathcal{M}(\mathbf{V}+\Delta\mathbf{V}),\phi_{\text{deep}}(\mathbf{I}))$

Figure 2.7. Mesh refinement stage: the mesh vertex spatial coordinate are refined to $\mathbf{V}+\Delta\mathbf{V}$ using graph convolution based on the RGB image features. Then, the semantic features $\mathbf{C}$ of the mesh vertices are initialized by projecting the vertices to the image plane and associating them with 2D semantic segmentation features. Finally, the vertex semantic features are refined to be $\mathbf{C}+\Delta\mathbf{C}$ using graph convolution.

the initial mesh vertex semantic features $\mathbf{C}$ in (2.17) to the graph convolution input:

$$\begin{aligned} \mathbf{C}_1^{\text{in}} &= \text{ReLU}(\mathbf{W}_1^{\mathbf{C}}\mathbf{V}_{g_{\text{in}}}) && \\ \mathbf{C}_i^{\text{in}} &= \mathbf{C}_{i-1}^{\text{out}}, && i=2,3, \\ \mathbf{C}_i^{\text{out}} &= \text{ReLU}(g_i^{\mathbf{C}}([\mathbf{C}_i^{\text{in}};\mathbf{V};\mathbf{C}];\theta_{g\mathbf{C}i})), && i=1,2,3, \\ \Delta\mathbf{C} &= \mathbf{W}_2^{\mathbf{C}}[\mathbf{C}_3^{\text{out}};\mathbf{V};\mathbf{C}], && \end{aligned} \tag{2.18}$$

where $\Delta\mathbf{C}\in\mathbb{R}^{n\times s}$ is the matrix of semantic residuals and $\mathbf{W}_1^{\mathbf{C}},\mathbf{W}_2^{\mathbf{C}}$ are two matrices as linear layers. The trainable parameters for vertex semantic graph convolution are

$$\theta_{3D\mathbf{C}} = [\mathbf{W}_1^{\mathbf{C}};\mathbf{W}_2^{\mathbf{C}};\theta_{g\mathbf{C}1};\theta_{g\mathbf{C}2};\theta_{g\mathbf{C}3}].$$

Now we can perform the joint geometric and semantic refinement. All trainable parameters for the 3D graph convolution are $\theta_{3D}=[\theta_{3D\mathbf{V}};\theta_{3D\mathbf{C}}]$. An illustration of the joint geometric and semantic refinement is provided in Fig. 2.7. For the initial mesh $\mathcal{M}(\mathbf{V},\mathbf{0})$, we first estimate the geometric residuals $\Delta\mathbf{V}$ (2.16) from Sec.2.3.2. On the geometrically refined mesh $\mathcal{M}(\mathbf{V}+\Delta\mathbf{V},\mathbf{0})$, we initialize the semantic features as in (2.17) to get $\mathcal{M}(\mathbf{V}+\Delta\mathbf{V},\mathbf{C})$. Then we estimate the semantic residuals $\Delta\mathbf{C}$ (2.18). The final joint geometric and semantic refined mesh is $\mathcal{M}^{\text{ref}} = (\mathbf{V} + \Delta\mathbf{V}, \mathbf{C} + \Delta\mathbf{C})$.

2.3.4 Global Mesh Merging

Given semantically annotated meshes obtained by our model from each camera view, a global mesh of the whole environment can be obtained by transforming each local mesh to the global frame using the camera pose trajectory and merging it into a combined global mesh. We design an approach to incrementally merge local meshes into a global mesh. Given a new local mesh obtained from a camera view with a known pose and the current global mesh, we update the global mesh by merging information from the local mesh.

First, we refine the global mesh vertices. We transform the global mesh to the local camera frame and project it onto the 2D image plane. If the resulting 2D global mesh covers an area over a certain threshold (e.g., 70%), we regard the local frame as duplicate and proceed to the next frame. Otherwise, we determine the overlapping parts between the 2D projections of the global and local meshes. We choose the vertices of the overlapped global mesh as a source point cloud and sample a point cloud from the overlapped local mesh vertices as a target point cloud. We perform non-rigid point cloud registration between the source and target point cloud using the Coherent Point Drift (CPD) algorithm [124]. Through this non-rigid transformation, we deform and refine the global mesh geometry based on the local mesh information.

Second, we introduce new vertices and faces into the global mesh from the non-overlap region of the local mesh. We remove the faces of the local mesh that overlap with the global mesh projection on the image plane. Through this step, we decouple the global mesh and the local mesh because their 2D projections do not intersect with each other any longer. We perform 2D constrained Delaunay triangulation [2] over the global and local mesh projections, keeping the edges of existing triangles in tact. Through this step, we connect the global mesh and the local mesh to obtain a new global mesh, which is lifted back to 3D using the vertex depth values and the known camera pose.

Fig. 2.8 and Fig. 2.9 demonstrate the mesh merging process, which results in a single

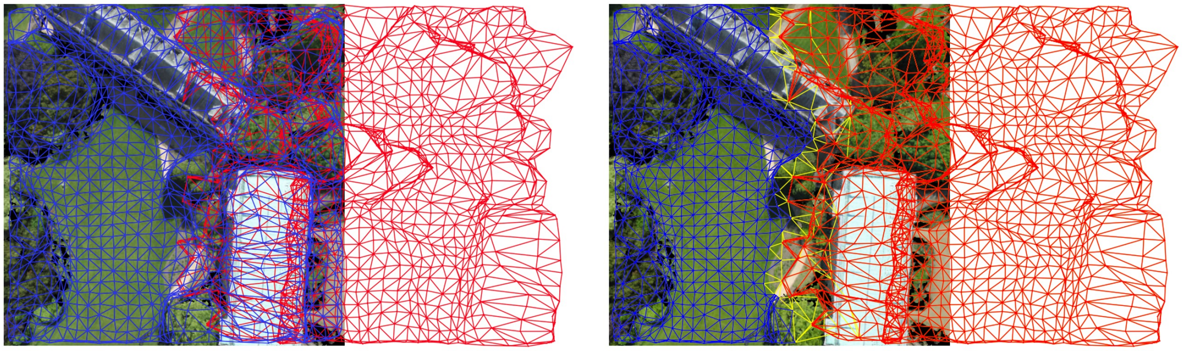

Figure 2.8. Red: global mesh. Blue: local mesh. Yellow: New edges after merging. Left: the refined global mesh overlaps with the local mesh. Right: the global and the local meshes are separated by removing overlapping faces and a new global mesh is generated via 2D constrained Delaunay triangulation.

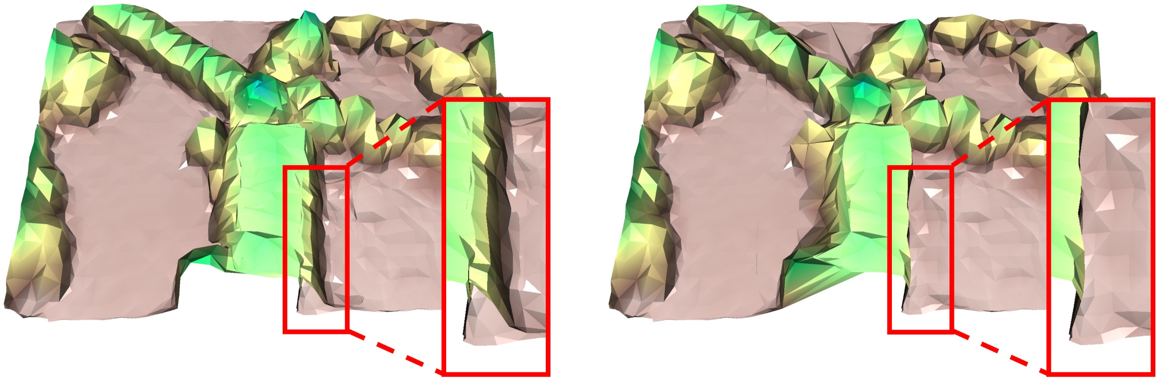

Figure 2.9. Left: Stacking two local meshes directly. Right: Merging two meshes with our proposed method in Sec. 2.3.4.

consistent global mesh and removes artifacts such as double layers in naïve mesh merging.

2.4 Experiments

In this section, we evaluate our metric-semantic mesh reconstruction approach using aerial image sequences generated from three open-source 3D datasets: WHU MVS/Stereo [108], SensatUrban [79], and STPLS3D [25]. We evaluate the model generalization ability by training and testing on different datasets. Ablation studies are included to show the effectiveness of our choices in the model design.

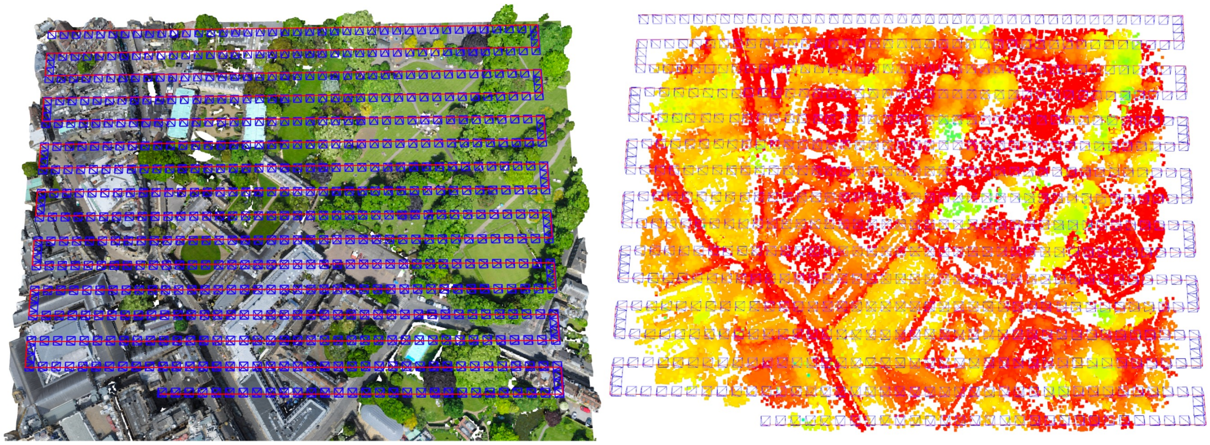

Figure 2.10. Left: Camera trajectory used to render RGBD images from a point cloud model generated from the SensatUrban dataset [79] Right: Sparse depth points and camera poses estimated by ORB-SLAM3. The color indicate elevation.

2.4.1 Datasets

Our mesh reconstruction approach requires ground-truth depth and semantic segmentation data for supervised training, which are generally not available and challenging to obtain from RGB aerial images. We used photo-realistic point cloud models covering several km^2 reconstructed from real aerial images in WHU MVS/Stereo and SensatUrban dataset to render RGB, depth, and semantic segmentation images. This provides accurate depth and semantic supervision data, while keeping the RGB images realistic, which is important for real-world applications of our model. We also use data from the synthetic STPLS3D dataset, which has more variation in the scene layout and the texture. We divide the large point cloud models in each dataset into different regions and generate different image sequences over them. Each camera trajectory follows a sweeping grid-pattern, which is common in drone flight planning (see Fig. 2.10). The camera trajectories are chosen to ensure enough image overlap for tracking and sparse depth reconstruction. RGBD images with resolution 512×512 are rendered along the trajectory from the ground-truth point cloud using PyTorch3D [149]. We keep the RGB aerial image resolution at around 0.2 meter/pixel. When semantic labels are available in the point cloud model, we also render semantic segmentation images with the same size as the RGBD images.

WHU Dataset

The WHU MVS/Stereo dataset [108] provides geo-calibrated RGBD images rendered from a highly accurate 3D digital surface model of a 6.7×2.2 km^2 area over Meitan County, Guizhou Province, China. The 3D DSM model is not publicly available, so we recover a dense point cloud from the RGBD images as a ground-truth 3D model. Semantic labels are also not available in this dataset so we only perform geometric reconstruction using the WHU dataset. We obtain sparse depth measurements $\mathbf{D}^s$ for each image by applying OpenSfM [118] to its four neighbor images with known camera intrinsic and extrinsic parameters. Since monocular structure from motion (SfM) suffers from scale ambiguity, we rescale the reconstructed point cloud obtained from OpenSfM to align it with the real 3D model. In reality the scale can be recovered from other sensor measurements like GPS or IMU. The point features reconstructed by OpenSfM are treated as sparse noisy depth measurements. The noise is due to feature detection and matching as well as the bundle adjustment step. We also obtain noiseless depth measurements with the same 2D sparsity pattern from the ground-truth depth images $\mathbf{D}$. We vary the number of available sparse depth measurements as 500, 1000, 2000. We generate 20 camera trajectory sequences with 200 images in each sequence, split into 14 for training, 2 for validation, and 4 for testing.

SensatUrban Dataset

The SensatUrban dataset [79] is a point cloud dataset obtained using photogrammetry in two urban areas in Birmingham and Cambridge, UK. Each 3D point in the dataset is labeled as one of 13 semantic classes. The Birmingham region covers an area of 1.2 km^2. The Cambridge region covers an area of 3.2 km^2. We only use the training set part of the data in which point cloud semantic labels are available. We keep 4 semantic categories (ground, vegetation, building, and traffic road) and merge or discard the remaining less-frequent categories. We used monocular ORB-SLAM3 [19] to estimate the camera poses and sparse feature depths on the SensatUrban dataset. Compared with OpenSfM, ORB-SLAM3 performs sequential optimization of the image sequences, instead of looping over all the images to find matching pairs. As a result, it runs faster (1-10 Hz) and may be deployed on an aerial robot directly. We use OrbSLAM3 in order to ensure that our method can operate incrementally in time and handle pose and sparse depth estimation errors typical for online SLAM algorithms. We also re-scale the reconstructed point cloud and camera poses to align with the real 3D model. Finally, we project the point cloud to each camera frame to derive a sparse depth image. We vary the number of available sparse depth measurements as 500, 1000, 2000, 4000. We generate 13 camera trajectory sequences with 660 images in each sequence, split into 8 for training, 2 for validation, and 3 for testing.

STPLS3D Dataset

The STPLS3D dataset [25] is a richly-annotated synthetic 3D aerial photogrammetry point cloud dataset with more than 16 km^2 of landscapes and up to 18 fine-grained semantic category annotations. To ensure the object placements in the virtual environments resemble real city blocks, the environments are built based on Geographic Information System (GIS) data that are publicly available. We use the same 4 semantic categories as for the SensatUrban dataset. We generated 38 camera trajectory sequences with 660 images in each sequence, split into 26 for training, 4 for validation, and 8 for testing. Camera keyframe poses and sparse depth measurements were estimated using ORB-SLAM3.

2.4.2 Implementation Details

During training, we use 1000 sparse depth measurements per image and generate a mesh model with 576/1024/2025 vertices. For the WHU dataset, the weights of the loss function in (2.9) are set to $[w_2, w_3, w_{\mathbf{V}}, w_{\mathcal{E}}, w_{\mathbf{S}}, w_{\mathbf{C}}] = [3, 1, 0.5, 0.01, 0, 0]$. For the SensatUrban and the STPLS3D dataset, the weights are set to $[w_2, w_3, w_{\mathbf{V}}, w_{\mathcal{E}}, w_{\mathbf{S}}, w_{\mathbf{C}}] = [5, 1, 0.5, 0.01, 5, 0.5]$ for the joint geometric-semantic training (Sec. 2.3.3). For the geometric training (Sec. 2.3.2), the last two weights are set to be 0. The loss weights are decided through evaluation on the validation set. We use the 2D loss ℓ_2 in (2.2) as the metric to choose the best model on the validation set.

The Chamfer distance d in the ℓ_3 loss (2.3) is computed using 10000 samples.

For the WHU experiments in Sec. 2.4.3, we use ResNet-18 for the 2D feature extraction. For the remaining ones including the generalization experiments, we use ResNet-34. The ResNet is initialized with the pretrained weights on ImageNet-1K. The ResNet and GCN parameters of our model are optimized jointly during the mesh refinement training using the Adam optimizer [95] with initial learning rate of 0.0005 for 100 epochs. For the semantic reconstruction task, we first train a DeepLabv3 model with ResNet-50 backbone [24] alone for 2D semantic segmentation on the SensatUrban training set. The ResNet-50 is initialized with the pretrained weights on ImageNet-1K. We use the Cross Entropy loss for training and set the class weight as [ground, vegetation, building, traffic road] = $[1,2,3,3]$. During the mesh semantic refinement step, we use the Dice loss in (2.7) and keep the same per-class weights. We use three graph convolution stages for the WHU dataset and two graph convolution stages for the SensatUrban and the STPLS3D dataset. For the joint geometric-semantic training, we concatenate two graph convolution stages, where the first stage predicts the geometric residual only and the second stage predicts both the geometric and the semantic residuals. All trainable parameters of the model in (2.1) are $\theta = [\theta_{2D}; \theta_{2Dsem}; \theta_{3D}]$, including the parameters of the 2D features extraction, 2D semantic segmentation, and 3D graph convolution models.

2.4.3 Geometric Reconstruction

Our experiments report the ℓ_2 error in (2.2) and the ℓ_3 error in (2.3) for the reconstructed meshes. The ℓ_2 error emphasizes the accuracy of the projected depth, while ℓ_3 emphasizes the regions of large depth variation.

For comparison, we define a baseline method that triangulates the sparse depth measurements directly to build a mesh. The baseline method performs Delaunay triangulation on the 2D image plane over the depth measurements and projects the flat mesh to 3D using the measured vertex depths. We refer to the baseline method as sparse-depth-triangulation (*SD-tri*). SD-tri defines vertices at all sparse depth measurements (500, 1000, or 2000) and, hence, may produce meshes with different number of vertices compared to other models.

First, we perform geometric reconstruction on the WHU dataset. The quantitative results from the comparison are reported in Table 2.1. All models are trained with 1000 sparse depth measurements and directly generalize to different numbers of sparse depth measurements. We compared three options for the 2D inputs provided to the mesh refinement stage: an RGB image only (RGB, 3-channels), an RGB image plus rendered depth from the initial mesh (RGB+RD, 4-channels), and an RGB image plus rendered depth from the initial mesh plus Euclidean distance transform (EDT) obtained from of the sparse depth measurements (RGB+RD+EDT, 5-channels). The model using RGB-only does not perform as well as the other two. The RGB+RD+EDT model has the best performance according to the ℓ_2 error metric. The RGB+RD method has similar performance in the ℓ_2 metric and smaller ℓ_3 error compared to RGB+RD+EDT. The RGB+RD model is used to generate our qualitative results in Fig. 2.11, 2.12, 2.13 with 1024-vertex meshes because it offers good performance according to both error metrics.

At the bottom of Table 2.1, we evaluate the mesh reconstruction accuracy with noisy sparse depth measurements obtained from OpenSfM. The measurements are noisy due to feature matching errors, local minima during bundle adjustment, and the simple projective camera model used for optimization. The average per image errors of the 500/1000/2000 sparse depth measurements were 1.011/1.017/1.023 meters, respectively. The baseline SD-tri method performs well in a noiseless setting but degenerates drastically when noise from the SfM feature reconstruction is introduced. In contrast, our model is more robust to noise due to two factors. First, our mesh initialization and refinement stages both include explicit mesh regularization terms (in (2.5) and (2.6)). Second, the image features extracted during the mesh refinement process help distinguish among different terrains and structures. The latter is clear from the improved accuracy of the refined, compared to the initialized, meshes. We also report the performance using a mesh with only 576 vertices. When the depth measurements are noisy, the 576-vertex mesh has lower ℓ_2 loss compared with the baseline method with similar number of vertices. It even has lower ℓ_3 loss compared with meshes with more vertices generated from the

Table 2.1. Quantitative evaluation on the WHU dataset [108]. The second column shows the number of available sparse depth measurements per image and indicates whether the measurements are noisy (Sec. 2.4.1). The *SD-tri* method triangulates a mesh using all the sparse depth measurements as vertices. The Regular-*n* model generates a regular mesh with *n* vertices and performs initialization and refinement steps as we propose in Sec. 2.3. The *Initialized* model constructs a mesh from the sparse depth (Sec. 2.3.1). The RGB, RGB+RD, RGB+RD+EDT methods refine the initialized mesh (Sec. 2.3.2), using different inputs respectively.

Error	Meshing	SD-tri	Regular-576			Regular-1024			
	Inputs	(vert = SD)	Initialized	RGB+RD	RGB+RD+EDT	Initialized	RGB	RGB+RD	RGB+RD+EDT
ℓ_2	500 w/o noise	1.492	2.069	1.670	**1.637**	1.861	1.575	1.289	**1.252**
	1000 w/o noise	1.172	1.834	1.596	**1.546**	1.535	1.298	1.124	**1.097**
	2000 w/o noise	0.916	1.941	1.551	**1.511**	1.344	1.144	1.045	**1.024**
ℓ_3	500 w/o noise	9.815	18.278	**13.438**	13.763	13.799	7.242	**5.647**	6.352
	1000 w/o noise	6.494	17.762	**12.938**	13.574	11.872	5.876	**4.911**	5.703
	2000 w/o noise	4.649	17.130	**12.483**	13.506	10.859	5.131	**4.477**	5.291
ℓ_2	500	1.865	2.294	1.809	**1.768**	2.155	1.828	1.486	**1.456**
	1000	1.632	2.056	1.701	**1.685**	1.826	1.535	1.319	**1.308**
	2000	1.485	1.717	1.655	**1.654**	1.629	1.364	**1.236**	1.241
ℓ_3	500	19.737	18.392	**12.974**	13.532	14.887	8.351	**6.157**	6.865
	1000	22.189	17.693	**12.258**	13.161	12.480	6.447	**5.266**	6.075
	2000	18.545	17.256	**11.856**	12.988	11.147	5.452	**4.793**	5.620

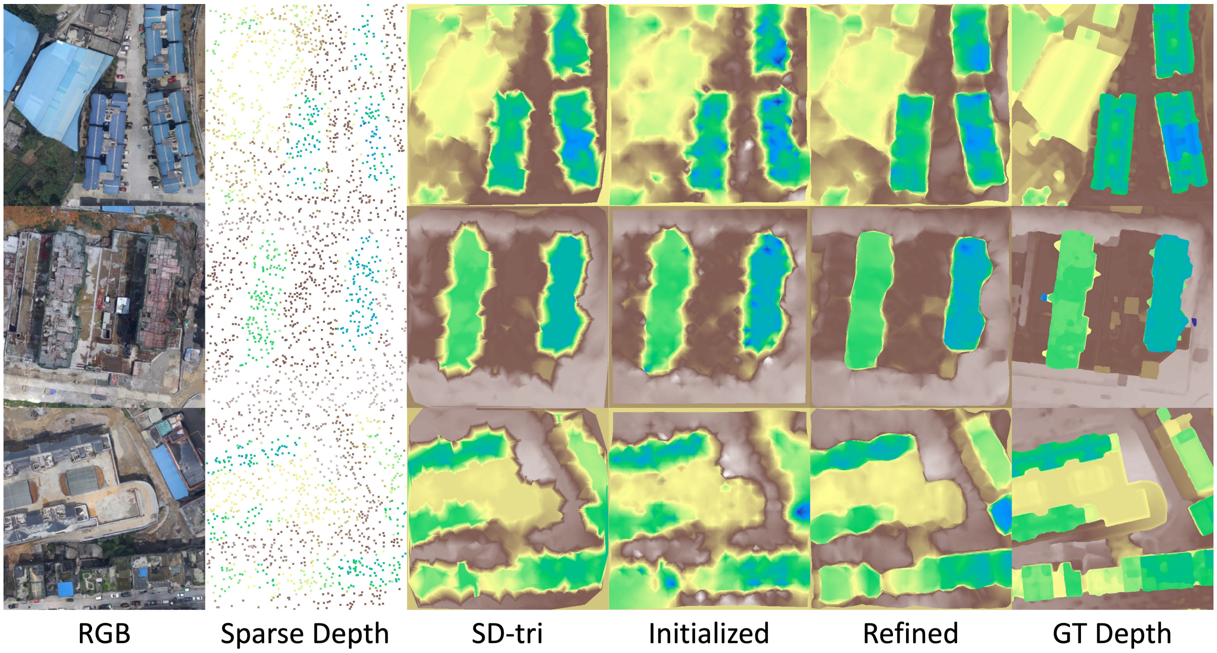

Figure 2.11. Mesh reconstructions on the WHU dataset [108] visualized as rendered depth images. The colors indicate the relative depth values. Column 1: RGB images. Column 2: sparse depth measurements (around 1000). Column 3: meshes reconstructed from sparse-depth triangulation. Column 4: meshes after initialization (Sec.2.3.1). Column 5: meshes after neural network refinement (Sec.2.3.2). Column 6: ground-truth depth images.

baseline method.

Qualitative results are presented in Fig. 2.11 and 2.12. Compared with SD-tri and initialized meshes, the refined meshes have smoother boundaries on the side surfaces of the buildings. The guidance from the image features allows the refined meshes to fit the 3D structure better. Fig. 2.13 shows a global mesh reconstruction obtained by transforming and merging 12 camera-view mesh reconstructions. The local meshes are transformed to global frame using the camera keyframe poses and no post-processing is used to merge them into a single global mesh.

2.4.4 Joint Geometric & Semantic Reconstruction

On the SemsatUrban dataset, we first perform geometric reconstruction with the same settings as in the WHU dataset. We train three models with different numbers of mesh vertices: $576 = 24^2$, $1024 = 32^2$ and $2025 = 45^2$. The quantitative results are reported in Table 2.2. As

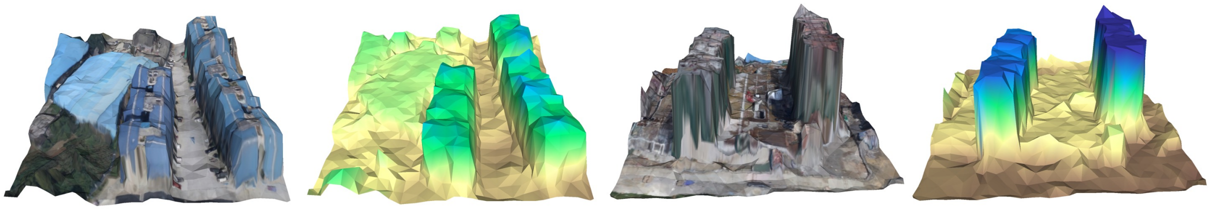

Figure 2.12. Reconstructed meshes painted with RGB texture and colors indicating elevations. These are associated with the first two rows in Fig. 2.11. The sharp vertical transitions of the buildings are reconstructed accurately.

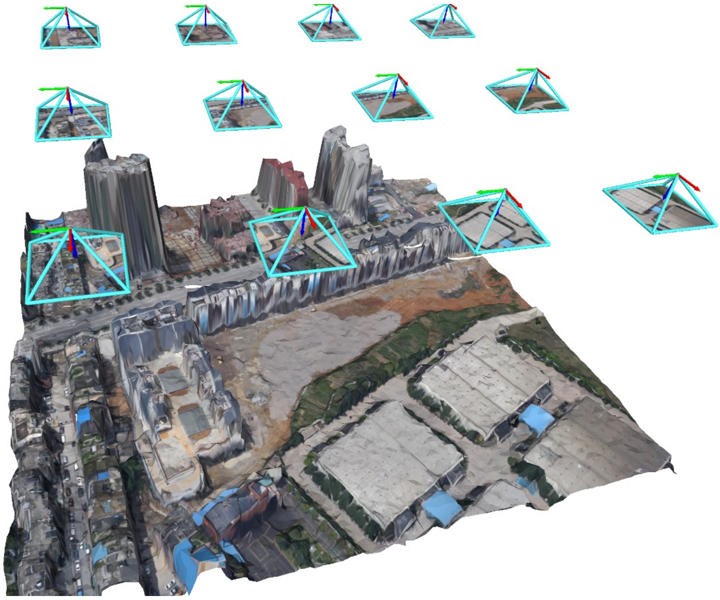

Figure 2.13. Complete environment model obtained by transforming to the global frame and merging local meshes from 12 camera views.

the number of sparse depth measurements increases, the baseline SD-tri method has better accuracy because the number of mesh vertices also increases. Our initialized meshes with fewer vertices are comparable with the SD-tri mesh, and the refined meshes are much better, especially according to the 3D metric ℓ_3. This shows that the joint 2D-3D loss in (2.9) enables our model to capture 3D structure details. Comparing the number of input depth measurements, we find that around 2000 measurements on the 512×512 image provide the best performance, while more do not noticeably improve the results. Regarding the number of mesh vertices, all three mesh sizes perform well. While we can see that the 1024-vertex mesh is generally better than

Table 2.2. Quantitative evaluation on the SensatUrban dataset [79]. The second column shows the number of available sparse depth measurements per image (Sec. 2.4.1). The baseline *SD-tri* method triangulates a mesh using all sparse depth measurements as vertices. The Regular-*n* model generates a regular mesh with *n* vertices and performs initialization and refinement steps (Sec. 2.3).

Error	Meshing	SD-tri	Regular-576		Regular-1024		Regular-2025	
	Inputs	(vert = SD)	Initialized	Refined	Initialized	Refined	Initialized	Refined
ℓ_2	500	2.018	2.050	1.175	2.204	**1.088**	1.992	1.171
	1000	1.843	1.841	1.096	1.865	**1.000**	2.033	1.153
	2000	1.715	1.796	1.120	1.700	**0.988**	1.752	1.209
	4000	1.647	1.834	1.181	1.662	**1.026**	1.630	1.283
ℓ_3	500	8.926	7.898	2.371	9.871	**2.128**	7.645	2.371
	1000	7.796	6.353	2.075	6.725	**1.815**	8.875	2.284
	2000	7.164	5.989	2.133	5.527	**1.745**	6.200	2.500
	4000	6.908	6.176	2.339	5.217	**1.844**	5.364	2.806

Table 2.3. Semantic segmentation per-class IoU for different geometric-semantic models. The definitions of the different models can be found in Sec. 2.4.4 and Sec. 2.4.7.

Class	Ground	Vegetation	Building	Traffic Road
Geo Init	0.642	0.810	0.846	0.643
Geo Refine	0.644	0.809	0.840	0.644
Cross Entropy	0.661	0.805	0.843	0.660
Focal	0.663	0.807	0.844	0.657
Jaccard	0.664	**0.826**	**0.863**	0.653
Our Model (Dice)	**0.674**	0.824	0.860	**0.663**
2D Seg	0.649	0.834	0.854	0.648

576-vertex mesh, the 2025-vertex mesh does not show an advantage over the 1024-vertex mesh. This indicates that good accuracy can be achieved with a light-weight storage-efficient mesh model.

We choose the 1024-vertex mesh to perform joint geometric-semantic reconstruction using 1000 sparse depth measurements. To evaluate the semantic reconstruction, we render a 2D semantic image from the mesh reconstruction and calculate the per-class Intersection over Union (IoU). For comparison, we report the IoU of the DeepLabv3 2D semantic segmentation model (named 2D Seg), the direct projection of the 2D semantic segmentation image onto the initial mesh as in (2.17) (named Geo Init) and the semantic segmentation projection onto the geometrically-refined mesh (named Geo Refine). Only 2D Seg is using a dense semantic image

Table 2.4. Geometric error for different metric-semantic models. The definitions of the different models and loss functions can be found in Sec. 2.4.4 and Sec. 2.4.7.

Class	Depth (ℓ_2)	Chamfer (ℓ_3)
Geo Init	1.866	6.725
Geo Refine	1.000	1.815
Cross Entropy	0.994	1.793
Focal	1.035	1.912
Jaccard	**0.976**	1.776
Our Model (Dice)	**0.976**	**1.763**

while the other methods store semantic features on the mesh vertices and interpolate through the semantic mesh renderer. As we can see in Table 2.3, our semantic residual refinement model improves the semantic segmentation performance compared to the direct projection of the 2D semantic segmentation image. Our approach also outperforms 2D Seg on most of the categories (ground, building, traffic Road) even though it is using only 0.4% of the points to store the semantic information (1024 mesh vertices vs 512 × 512 segmentation image). Further, we investigate whether the semantic mesh refinement affects the geometric reconstruction quality. In Table 2.4, we can see that our joint geometric-semantic mesh reconstruction achieves better geometric accuracy compared with purely geometric training. This can be explained by the fact that the semantic category information serves as regularization for the geometric properties. The results show that the geometric and semantic information help each other. More qualitative results for single-image reconstruction are provided in Fig. 2.14 and 2.15. Compared with SD-tri and initialized mesh, the refined mesh achieves higher reconstruction accuracy. The semantic refinement can improve the 2D semantic segmentation results. We can see that some noisy classification labels are removed after the refinement. In Fig. 2.16, we reconstruct several local metric-semantic meshes using the sparse depths from the tracked points from ORB-SLAM3. Then we transform them using estimated camera keyframe poses from ORB-SLAM3 to the global frame and combine them to form a global metric-semantic mesh without any further post-processing.

We also evaluated our model on the synthetic STPLS3D dataset. We use a 1024-vertex

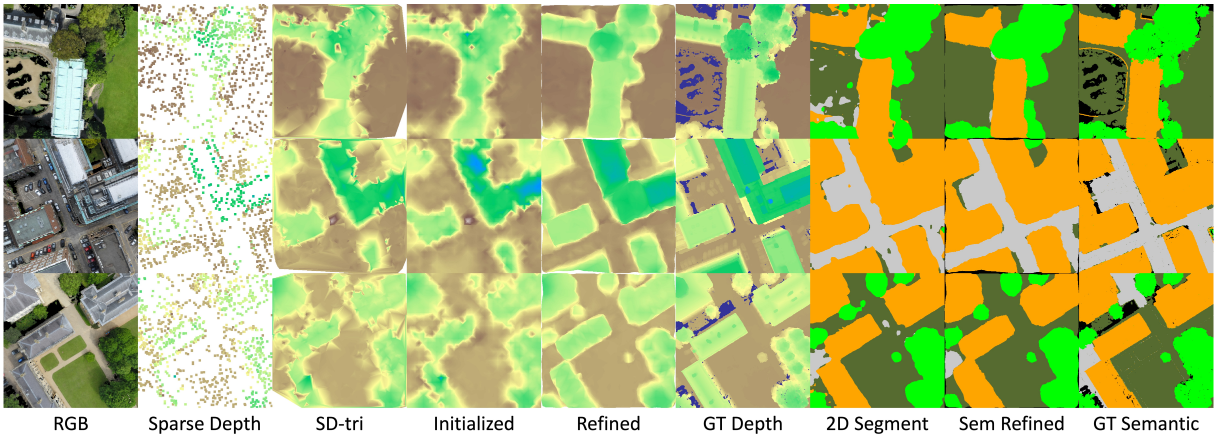

Figure 2.14. Mesh reconstructions on the SensatUrban dataset [79] visualized as rendered depth (colors indicate the relative depth values) and semantic images. The original 3D model is not fully complete so the RGB, GT Depth and GT Semantic may have little missing region. Column 1: RGB images. Column 2: sparse depth measurements (1000). Column 3: meshes reconstructed from sparse-depth triangulation. Column 4: meshes after initialization (Sec.2.3.1). Column 5: meshes after neural network refinement (Sec.2.3.2). Column 6: ground-truth depth images. Column 7: 2D semantic segmentation results. Column 8: meshes after neural network refinement (Sec.2.3.3). Column 9: ground-truth semantic segmentation maps.

Table 2.5. Geometric error on the STPLS3D dataset [25].

Class	Depth (ℓ_2)	Chamfer (ℓ_3)
SD-tri	2.039	11.832
Geo Init	1.996	10.417
Geo Refine	0.977	2.807
Sem Refine	**0.972**	**2.725**

mesh to perform geometric-only and joint geometric-semantic reconstruction using 1000 sparse depth measurements per image. The quantitative results are shown in Table 2.5. Our joint geometric-semantic mesh reconstruction method out-performs geometric-only mesh reconstruction, which verifies the effectiveness of fusing both geometric and semantic information.

2.4.5 Generalization Across Datasets

In this section, we evaluate the generalization ability of our model, trained on one dataset and applied to another. To align the datasets, we regenerated the RGB images and the sparse depth measurements on the WHU dataset to follow the same camera intrinsic parameters and

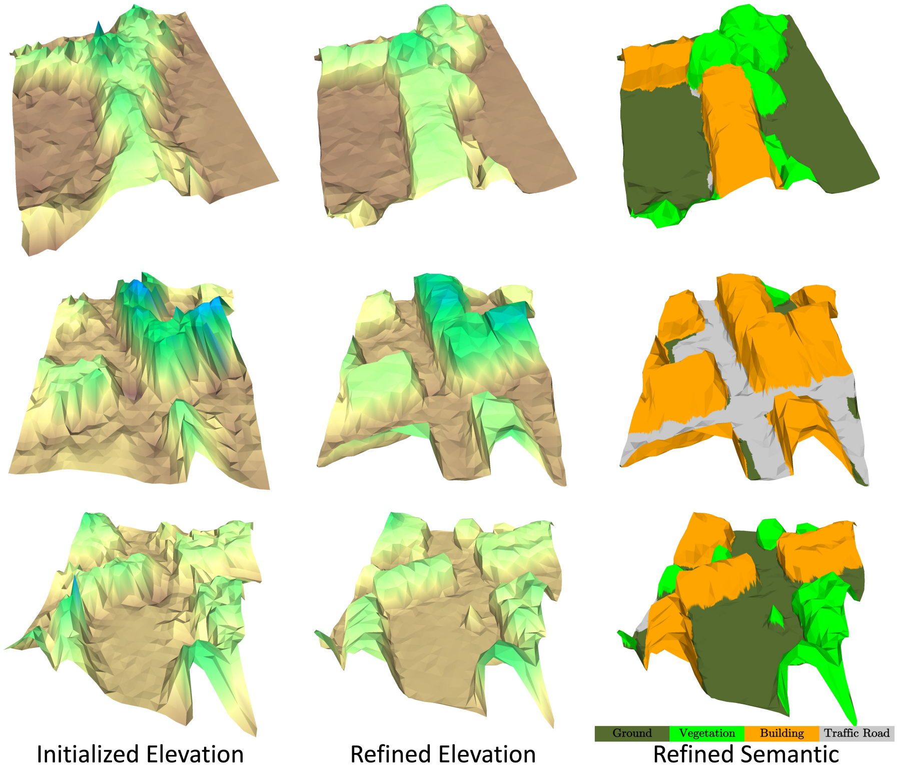

Figure 2.15. Reconstructed meshes painted with colors indicating elevations and semantic labels. These are associated with the three rows in Fig. 2.14. Column 1: initialized meshes. Column 2: refined meshes colored by elevation. Column 3: refined meshes colored by semantic categories.

trajectory patterns in the SensatUrban and the STPLS3D dataset so that there are 660 frames for each trajectory and ORB-SLAM3 is used to estimate sparse depth measurements and camera keyframe poses. It is challenging to achieve zero-shot generalization, so we also include a finetuning step. During finetuning, we only use 10% of the target domain training set and train for 30 epochs. A validation set (10% of the target domain validation set) is used to choose the best model with the 2D loss ℓ_2 as the metric. Usually, models trained on larger datasets show better generalization ability. We choose to use a model trained on WHU to generalize to SensatUrban and a model trained on STPLS3D to generalize to both WHU and SensatUrban. The average per image sparse depth errors in meters for 1000 depth samples were 1.771 on STPLS3D, 2.460 on WHU, and 1.597 on SensatUrban. The sparse depth measurements are generated through ORB-SLAM3.

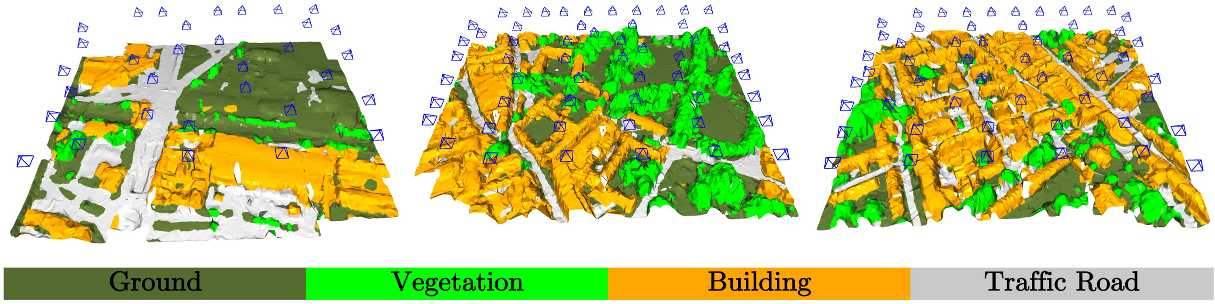

Figure 2.16. Global metric-semantic meshes reconstructed from three areas in the SensatUrban dataset [79] by fusing the local mesh reconstructions at the keyframe camera poses (shown in blue). The three global meshes are obtained from 32/55/54 local keyframe meshes, respectively.

Table 2.6. Generalization experiment: Geometric error on WHU dataset [108]. Brackets indicate the dataset used for training.

Class	Depth (ℓ_2)	Chamfer (ℓ_3)
SD-tri	3.628	40.431
Init	3.582	38.570
Refine (WHU)	2.253	11.332
Refine (STPLS3D)	20.043	1478.478
Refine (STPLS3D finetune)	2.047	10.796

First, we evaluate how the model trained on STPLS3D generalizes to WHU. The geometric error is reported in Table 2.6. The WHU dataset is more challenging due to the presence of denser and taller ($>$ 30m) buildings. The camera intrinsics and the flight pattern and height are different compared to the data generated in Sec. 2.4.3 so the numbers in Table 2.6 are not directly comparable with Table 2.1. Zero-shot generalization does not work for WHU, which is understandable given the large domain gap. The STPLS3D synthetic dataset uses scene layouts extracted from a U.S. Geological Survey (USGS) which covers cities in the United States, while the WHU dataset is collected in a Chinese city. After finetuning with only 10% of the original WHU training set, our model generalizes well to WHU, and even outperforms the model trained purely on WHU.

Next, we evaluate how models trained on WHU and STPLS3D generalize to SensatUrban. The results are presented in Table 2.7 and Table 2.8. We report only geometric error for the

Table 2.7. Generalization experiment: Geometric error on SensatUrban dataset [79]. Brackets indicate the dataset used for training.

Class	Depth (ℓ_2)	Chamfer (ℓ_3)
SD-tri	1.843	7.796
Init	1.865	6.725
Refine (SensatUrban)	1.000	1.815
Sem Refine (SensatUrban)	0.976	1.763
Refine (WHU)	1.439	3.827
Refine (WHU finetune)	1.112	2.223
Refine (STPLS3D)	1.442	3.973
Sem Refine (STPLS3D)	1.501	4.309
Refine (STPLS3D finetune)	1.021	1.992
Sem Refine (STPLS3D finetune)	1.043	2.111

Table 2.8. Generalization experiment: Semantic segmentation per-class IoU on SensatUrban dataset [79]. Brackets indicate the dataset used for training.

Class	Ground	Vegetation	Building	Traffic Road
2D Seg (SensatUrban)	0.649	0.834	0.854	0.648
Sem Refine (SensatUrban)	0.674	0.824	0.860	0.663
2D Seg (STPLS3D)	0.444	0.676	0.565	0.061
Sem Refine (STPLS3D)	0.528	0.667	0.608	0.038
2D Seg (STPLS3D finetune)	0.652	0.827	0.838	0.638
Sem Refine (STPLS3D finetune)	0.657	0.814	0.850	0.623

model trained on WHU. We can see that zero-shot generalization from WHU to SensatUrban improves the initialized meshes, while a finetuned model performs even better. We train a geometric-only model and a metric-semantic model on STPLS3D. In terms of geometric loss, both STPLS3D models generalize well to SensatUrban and their performance after finetuning is close to that of a model trained on SensatUrban. However, the metric-semantic model is slightly worse than the pure geometric model. In terms of semantic segmentation performance, zero-shot generalization does not perform well and especially fails on the traffic road category. After finetuning, the metric-semantic model can largely close the gap between itself and the SensatUrban model. Given that the RGB images from the synthetic scenes in STPLS3D have very different appearance, it is understandable that the semantic model that heavily relies on the RGB image might be harder to generalize compared to the geometric-only model.

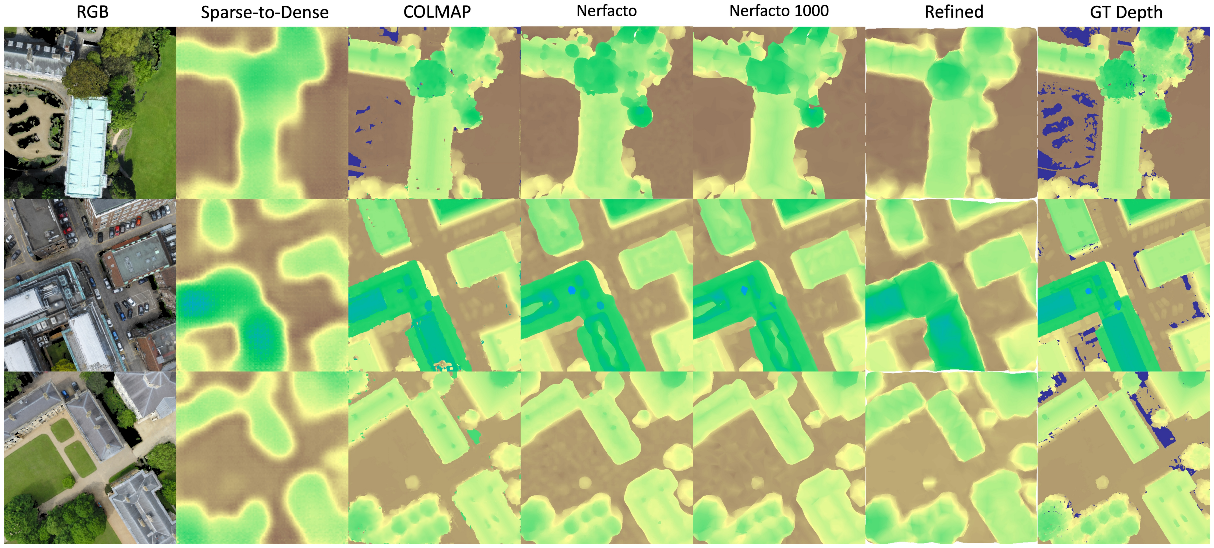

Figure 2.17. Depth reconstruction comparison on the SensatUrban dataset [79] among Sparse-to-Dense [115], COLMAP [165], Nerfacto [179], Nerfacto restricted to 1000 mesh vertices, and our method.

These experiments demonstrate promising generalization ability of our mesh reconstruction method, using limited data to finetune or even without finetuning in some cases. The model generalizes better in terms of geometric reconstruction than in terms of semantic classification. Nevertheless, it is exciting to see that a model trained on a synthetic dataset (STPLS3D) can generalize well to real data. This makes it possible to achieve good performance by training a model with inexpensive synthetic data that comes with free ground-truth labels and finetuning on a small set from the target domain.

2.4.6 Comparison with Other Methods

In this section, we compare our approach with depth completion, multi-view stereo, and NeRF techniques on the SensatUrban dataset [79] with 1000 sparse depth measurements. The qualitative results are shown in Fig. 2.17. The quantitative results are shown in Table 2.9. To evaluate the reconstruction quality of the different methods comprehensively, we report multiple 2D and 3D reconstruction accuracy metrics [177]. The threshold distance for 3D precision and recall is set to 0.5m due to the large scale of the reconstructed mesh.

Table 2.9. Geometric error comparison among Sparse-to-Dense [115], COLMAP [165], Nerfacto [179], and our method using metrics defined in [177]. Nerfacto Mesh (1000) uses about 1000 mesh vertices, similar to our method. The other two mesh methods use about 20000 vertices.

Method	2D			
	Abs Diff (ℓ_2) ↓	RMSE ↓	Abs Rel ↓	Sq Rel ↓
Initializaiton	1.865	3.210	0.029	0.218
Refinement	1.000	1.792	0.015	0.056
Semantic Refinement	0.976	1.715	0.014	0.053
Depth Completion - Sparse-to-Dense	1.987	2.949	0.029	0.148
MVS - COLMAP	0.425	1.741	0.006	0.054
MVS - COLMAP Mesh	0.588	1.612	0.008	0.039
NeRF - Nerfacto Mesh	1.217	2.444	0.016	0.102
NeRF - Nerfacto Mesh (1000)	1.178	2.395	0.016	0.099

Method	3D					
	Chamfer (ℓ_3) ↓	Accuracy ↓	Completeness ↓	Precision ↑	Recall ↑	F-score ↑
Initializaiton	6.692	1.319	1.173	0.192	0.202	0.197
Refinement	1.787	0.767	0.778	0.288	0.298	0.293
Semantic Refinement	1.735	0.754	0.772	0.293	0.301	0.297
Depth Completion - Sparse-to-Dense	6.136	1.283	1.432	0.139	0.137	0.138
MVS - COLMAP	3.358	0.638	0.914	0.464	0.405	0.432
MVS - COLMAP Mesh	3.379	0.980	0.724	0.320	0.361	0.339
NeRF - Nerfacto Mesh	5.818	1.364	0.947	0.181	0.221	0.199
NeRF - Nerfacto Mesh (1000)	5.351	1.258	1.018	0.195	0.212	0.203

Depth completion methods take a sparse depth image and other inputs, such as an RGB image, and recover a dense depth image. While there are many depth completion algorithms, few focus on the aerial image domain. We compare against the Sparse-to-Dense method [115], an end-to-end deep learning regression model, because it considers a similar problem setting and uses similar feature extraction as our approach. The Sparse-to-Dense model consists of a ResNet feature extraction encoder and per-pixel depth regression decoder. We trained both our model and Sparse-to-Dense with a ResNet-34 feature extractor, thus focusing the comparison on the performance of the pure 2D learning and per-pixel depth regression of Sparse-to-Dense versus the joint 2D-3D learning for mesh refinement of our method. The results in Fig. 2.17 and Table 2.9 show that the Sparse-to-Dense model is not as accurate as our method on the SensatUrban dataset, and it is beneficial to utilize our joint 2D-3D learning technique. At least on this dataset, it is challenging for Sparse-to-Dense to regress an accurate dense depth map, while, using the same 2D feature extraction network, our mesh initialization and refinement method performs better at reconstructing the 3D scene.

We also compare our method with COLMAP [165], a multi-view stereo technique that recovers 3D structure from a series of calibrated images using pixelwise view selection for depth and normal estimation. In this comparison, we used ground-truth camera poses and skip the SfM step. For each frame, we manually select neighboring frames within a radius of 50 m for multi-view stereo matching. We use the reconstructed dense depth images to obtain a global point cloud, generate a global mesh, and crop the mesh at each camera pose to obtain local meshes associated with each frame. For the meshing step, we compared Poisson reconstruction [94] and Delaunay mesh reconstruction [101] and found that due to point cloud noise the Delaunay reconstruction performs better. The COLMAP is the dense depth estimation result, while the COLMAP Mesh is the subsequent meshing result. For COLMAP, we convert the dense depth images to a point cloud and sample 10000 points to compute the ℓ_3 error. For COLMAP Mesh, we render the local mesh to get a rendered depth image to compute the ℓ_2 error. The results are shown in Fig. 2.17 and Table 2.9. The COLMAP method generally has better depth reconstruction measured by ℓ_2 error, while our method has lower ℓ_3 error because of the implicit regularization in our mesh reconstruction. The 3D error ℓ_3 can be large when outliers appear in the reconstruction. COLMAP achieves better reconstruction at the cost of heavy computation for the MVS step. It takes around 4 seconds per frame to recover a dense depth image using GPU, while meshing requires additional time. Our method is much faster, with 0.07 s per frame on a desktop with GeForce RTX 2080 Ti GPU and 0.45 s per frame on a Jetson AGX Xavier edge computing platform. When it comes to online mesh reconstruction on a resource-constrained platform, our method offers an advantage over MVS.

Finally, we compare with Nerfacto [179], a NeRF model that combines components from recent NeRF papers to achieve a balance between speed and quality. As a NeRF model, Nerfactor takes posed RGB images and constructs an implicit 3D scene represented by a deep neural network. We used ground-truth camera poses for each image and trained separate Nerfacto models for each test image sequences. One in ten frames was chosen as an evaluation image during training. Notice here no depth images are used for the NeRF training. We observed that depth images rendered directly from the trained Nerfacto model have inconsistent depth across frames. Therefore, we exported a mesh model using the Poisson surface reconstruction [94] implemented in Nerfstudio [179]. Nerfacto exports meshes with different vertex density. A dense mesh has about 20000 vertices for each camera view, while a sparse mesh has about 1000, which is similar to our method's mesh vertex density. To compute the 2D metrics in Table 2.9, we rendered depth images from the mesh. The evalution results for Nerfacto are shown in Fig. 2.17 and Table 2.9. Qualitatively, Nerfacto has similar reconstruction accuracy with our method but the quantitative metrics indicate that it is not as good as our method. Furthermore, Nerfacto needs to be trained for each novel environment and the training can hardly meet real-time requirement, while our method can make the inference faster.

2.4.7 Ablation Studies

Sec. 2.4.3 compared the effect of using RGB, rendered depth, and Euclidean distance transform as inputs for the mesh reconstruction model. This section reports additional ablation studies on the SensatUrban dataset. We use a 1024-vertex mesh model and 1000 sparse depth measurements for training and testing. We evaluate the effects of mesh initialization, types of 2D input data, and number of graph convolution stages on the geometric mesh reconstruction accuracy. We also evaluate the performance effect of joint metric-semantic training and the choice of a semantic loss function.

Mesh Initialization

An important aspect of our model in Sec. 2.3 is the separation of the mesh initialization stage from the mesh refinement stage. The mesh initialization stage allows the data-driven refinement stage to focus on learning the mesh vertex deformation residuals instead of absolute vertex coordinates. To demonstrate the effectiveness of this design, we compare our model to a baseline model which applies the refinement stage directly to a flat initial mesh. The baseline model, Flat Init, deforms a flat initial mesh with vertex depth specified by the mean of the sparse depth measurements. Table 2.10 shows that the Flat Init model makes the 2D-3D learning problem challenging, and the model performs even worse than purely geometric initialization as in Sec. 2.3.1.

2D Input Channels

In (2.13), we concatenate an RGB image $\mathbf{I}$ (RGB), rendered depth $\rho_D(\mathcal{M}^{\text{int}})$ (RD) and a Euclidean distance transform $\mathbf{E}(\mathbf{D}^s)$ (EDT) to form a 5-channel input image used for 2D feature extraction. Table 2.10 evaluates the role of the different 2D inputs on the overall mesh reconstruction performance. The results indicate that the RGB information plays the most important role in refining the initialized mesh. The model RD+EDT that does not use RGB features performs the worst. Adding RD and EDT inputs to the RGB gives an additional boost to

Table 2.10. Ablation study on geometric reconstruction error for geometric models. The definitions of the different models can be found in Sec. 2.4.7.

Class	Depth (ℓ_2)	Chamfer (ℓ_3)
Geo Init	1.865	6.725
Flat Init	4.646	20.655
RD+EDT	1.761	5.791
RGB	1.521	3.750
RGB+RD	1.070	2.138
1 Stage	1.015	1.828
Our Model	**1.000**	**1.815**

the accuracy.

Number of graph convolution stages

Table 2.10 also evaluates the effect of one (1 Stage) vs two (Our Model) graph convolution stages in the geometric mesh refinement (Sec. 2.3.2). The first GCN stage contributes the most to the geometric refinement, while the second GCN stage further refines the results.

Semantic Loss Function

Finally, we discuss the choice of a semantic loss function $\ell_{\mathbf{S}}$. Instead of the Dice loss in (2.7), three other semantic loss functions may be considered.

- The cross entropy loss is widely used for semantic segmentation. Given two stochastic vectors $\alpha, \beta \in [0,1]^s$, the cross entropy loss is defined as:

$$\begin{aligned} \mathrm{CE}(\alpha,\beta) &= -\sum_{i=1}^{s} \beta_i \log(\alpha_i), \\ \ell_{\mathbf{S}1}(\mathcal{M},\mathbf{S}) &:= \mathrm{mean}(\mathrm{CE}(\sigma(\rho_S(\mathcal{M})),\mathbf{S})), \end{aligned} \tag{2.19}$$

  where CE is applied to the elements $\sigma(\rho_{S,ij}(\mathcal{M})) \in [0,1]^s$ and $\mathbf{S}_{ij} \in [0,1]^s$ of the tensors of predicted and ground-truth semantic class probabilities.

- The focal loss [106] is a variation of cross entropy, focusing on hard misclassified examples:

$$\begin{aligned} \mathrm{FL}(\alpha,\beta) &= -\sum_i \beta_i (1-\alpha_i)\log(\alpha_i), \\ \ell_{\mathbf{S}2}(\mathcal{M},\mathbf{S}) &:= \mathrm{mean}(\mathrm{FL}(\sigma(\rho_S(\mathcal{M})),\mathbf{S})). \end{aligned} \tag{2.20}$$

- The Jaccard loss [87] measures the negative Intersection over Union (IoU) between the ground-truth and predicted semantic segmentation:

$$\ell_{\mathbf{S}3}(\mathcal{M},\mathbf{S}) := -\frac{|\sigma(\rho_S(\mathcal{M}))\cdot\mathbf{S}|}{|\sigma(\rho_S(\mathcal{M}))|+|\mathbf{S}|-|\sigma(\rho_S(\mathcal{M}))\cdot\mathbf{S}|}, \tag{2.21}$$

  where, as in (2.7), $|\cdot|$ sums up all the absolute values of the elements.

In Table 2.3, we see that the Jaccard loss in (2.21) leads to good segmentation performance, outperforming the Dice loss in (2.7) for some categories. The Cross Entropy and the Focal losses are not as good. In Table 2.4, we see that the Cross Entropy and the Jaccard loss both outperform the Focal loss when considering their effect on the geometric reconstruction accuracy. The Dice loss leads to the best geometric reconstruction accuracy. Considering the joint geometric and semantic performance, we elected to use the Dice loss for our final model.

2.4.8 Memory and Computation Complexity

The reconstructed mesh model is a more efficient representation than a dense depth image. A dense depth image requires $512\times 512 \approx 0.26M$ parameters, and a semantic image also requires the same number of parameters. Our mesh model with fixed face topology only needs storage of the 3D vertex coordinates and the semantic labels. With 1024 vertices, our semantic mesh model requires only 2% of the depth and semantic image parameters to obtain a high-fidelity reconstruction of a camera view. Our model has about 21M parameters (ResNet and GCN) and takes about 3GB GPU memory during inference. We report the inference time of our model on different computation platforms in Table 2.11. The results show that our mesh reconstruction

Table 2.11. Prediction time (in second) on different NVIDIA devices.

Platform	Initialization	Refinement	Total
Jetson AGX Xavier w/o GPU	0.45	0.75	1.20
Jetson AGX Xavier GPU	0.30	0.15	0.45
GeForce RTX 2080 Ti GPU	0.05	0.02	0.07

algorithm can achieve 2 Hz on an embedded NVIDIA Jetson AGX Xavier computer, making it applicable for real-time deployment onboard a robot system. Regarding timing the baseline algorithm for sparse-depth triangulation, we evaluated its run-time frequency to be at around 20 Hz on the same platform.

2.4.9 Limitations

Our 3D metric-semantic mesh reconstruction algorithm can run efficiently on an embedded computer but as a result the number mesh vertices used for reconstruction is limited, which in turn affects the geometric reconstruction accuracy. Furthermore, local meshes are generated using only a single camera frame without multi-view constraints, making it challenging to achieve consistent mesh merging into a global model. Potential avenues for future work that may improve the reconstruction quality include adaptively increasing the mesh vertices depending on the image feature distribution, considering techniques like deformable convolution [36] for associating the 3D mesh vertices with the 2D image features, utilizing sparse depth measurement uncertainty (e.g., keypoint covariances provided by SLAM) for weighted interpolation during the image features to vertex association, and improving the global mesh merging approach with multi-view constraints.

2.5 Summary

This chapter introduces an approach for 3D metric-semantic mesh reconstruction from RGB image and sparse depth measurements. Compared to methods that utilize only sparse depth for mesh initialization or triangulation, our approach provides more accurate geometric reconstruction by utilizing RGB image features. Compared to 2D semantic segmentation methods, our semantic reconstruction eliminates classification inaccuracies by inferring an underlying 3D mesh structure. The joint metric-semantic reconstruction approach improve geometric accuracy further by utilizing semantic information and provides memory savings compared to dense image depth and segmentation techniques. Employing our method in combination with feature- and keyframe-based odometry techniques allows reconstruction of global dense metric-semantic mesh models with utility in environmental monitoring and semantic navigation applications.

Acknowledgements

Chapter 2, in part, is a reprint of the material as it appears in Q. Feng and N. Atanasov, ”Mesh Reconstruction from Aerial Images for Outdoor Terrain Mapping Using Joint 2D-3D Learning,” IEEE International Conference on Robotics and Automation (ICRA), pp. 5208-5214, 2021. Chapter 2, in part, has been submitted for publication of the material as it may appear in Q. Feng and N. Atanasov, ”TerrainMesh: Metric-Semantic Terrain Reconstruction from Aerial Images Using Joint 2D-3D Learning,” IEEE Transactions on Robotics, 2023. The dissertation author was the primary author of these papers.

Chapter 3

Object Mapping from 3D Measurements

In Chapter 2, we discussed metric-semantic reconstruction of outdoor terrain. Our approach is suitable for reconstruction at a large scale but potentially low resolution. Due to the limited number of vertices we keep in the mesh representation, it is challenging to capture small objects, such as vehicles on the ground. A fine-grained, expressive yet compressed map representation is an important requirement as robots have limited storage and computation capabilities. An object-level map, which models objects including their semantic labels, poses, shapes, etc., is a promising environment representation. It is efficient as it ignores the irrelevant background and only focuses on the objects. It is still expressive as the objects offer key information for many robotics tasks including manipulation, navigation, and others. In this chapter, we focus on the object pose and shape estimation for object-level mapping using the 3D observations. We assume the semantic category label of a detected object is known and perform category-level object pose and shape estimation.

In recent years more and more affordable 3D sensors like depth cameras and LiDARs are equipped on the mobile robots to enhance their perception capability. The mapping procedure largely benefits from the 3D measurements, i.e., the 3D point clouds and the depth images. We consider two categories of methods for object reconstruction—model-based and model-free—depending on whether the reference object model is readily available.

When a reference model is available, we can match the model with the real 3D measurements. The task is converted to a 3D registration problem. One popular method is to extract point features, generate correspondences, and recover the pose. Besides, there are also correspondence-free methods that directly predict the 6 DoF pose from the RGB-D image [188, 21, 39]. Traditional methods using hand-crafted 3D features include Spin Image [88], FPFH [158], SHOT [162]. They mainly rely on simple local geometric information like normals and histogram. Recently learning-based methods have gained more attention. 3DMatch [204] applies 3D ConvNet on fixed-size 3D patch represented in truncated distance function. 3DFeat-Net [200] collects local points in a radius-fixed ball, uses a detector module to estimate the local orientation and generates the descriptor after alignment. 3DSmoothNet [62] proposes the idea of smoothed density value voxelization as the preprocessed input to reduce the sparsity of the input patch. FCGF [31] leverages the sparse convolution [30] to build fully-convolutional network in 3D space and use metric learning to learn the feature. The point cloud coordinates and associated features are used for registration to estimate the rigid transformation. These features can be generalized to various scenes. While the object or the category model is available [75, 15, 76], more information such as the object template can be leveraged to learn the 3D feature. Given the correspondence, RANSAC [51] or voting-based technique [29] can be used for finding the most promising pose hypothesis. The necessity of identical object model can be alleviated, by using the categorical model. Objects in the same category usually share similar structure with relatively consistent distribution of semantic keypoints [196], such as the wheels of the car and the chair legs. Category-level semantic keypoints [139] are predicted on RGB images and a deformable shape model is fitted to recover the pose. StarMap [211] extends to predict category-agnostic keypoints by predicting the keypoint's normalized canonical coordinate from RGB image observation. The Scan2CAD dataset [6] annotates 6D pose and scale of objects in the indoor scenes of ScanNet dataset [34] by aligning CAD models from ShapeNet dataset [20]. RGBD scans are converted to voxelized signed distance fields and a 3D CNN network is used to predict sparse keypoint correspondence, given a matching CAD model. NOCS [189] uses the idea of normalized canonical coordinates for a specific category and generates dense annotations covering the whole object surface. This model can predict the normalized canonical coordinates densely on a query RGB-D image and use them to recover the object pose. The shape variation between the object instances can make the registration challenging. The availability of massive object CAD datasets [20] makes it possible to select a similar-looking instance from the database to increase the accuracy of estimating the pose of the unknown instance. Hence, object retrieval is an important sub-problem for robust pose registration. Compared to category classification of 3D objects [43, 73], retrieval of specific CAD instances is more challenging due to the emphasis on shape similarity. Grabner et al. [64] render CAD model depth and embed the depth and RGB image observations jointly for CAD model retrieval. Dahnert et al. [33] use a 3D hourglass encoder-decoders structure to learn an embedding feature with triplet loss for shapes, implicitly represented using a signed distance field. Uy et al. [184] introduce a deformation-aware asymmetric distance across CAD models and learn an egocentric anisotropic distance field for latent embeddings. Combining the retrieval and the registration task, a more similar CAD model can be selected to recover the object pose at a higher accuracy.

When the reference model is not available, we need to reconstruct the 3D representation of the object. Although the 3D point cloud reconstruction can be at a very high-quality with accurate pose tracking [19, 205, 168], we specifically focus on a more complete and continuous shape representation including the surface. One option is to use 3D mesh. There are meshing approaches that generate the mesh from the point cloud [94, 9]. Besides the mesh explicit representation, implicit representation like signed-distance function (SDF) is also widely used for shape reconstruction. Discrete form of SDF stored in voxels [135] can be built from the RGB-D sequences. Recently the neural network is used to learn a continuous SDF function for shape reconstruction. DeepSDF [138] learns a continuous metric function of distance instead of binary classification function dividing inside or outside, which makes it suitable for gradient-based optimization. Subsequent works along the direction of DeepSDF include FroDO [159], MOLTR [104], and DualSDF [69]. FroDO leverages both point cloud and SDF representations, which defines sparse and dense losses to optimize the object shape. An extension of FroDO is MOLTR, which reconstructs an object shape by fusing multiple single-view shape codes to handle both static and dynamic objects. Similar to the coarse-to-fine shape estimation in FroDO and MOLTR, DualSDF uses two levels of granularity to represent 3D shapes. A shared latent space is employed to tightly couple the two levels, and a Gaussian prior is imposed on the latent space to enable sampling, interpolation, and optimization-based manipulation. DeepSDF and the derivatives offer models for accurate shape modeling but few of them consider object pose estimation.

This chapter is based on the papers [47, 208, 167]. We focus on object mapping from 3D point cloud observations. To be more specific, we estimate the object pose and shape at a categorical level, i.e., with the prior of the object category. Two approaches are presented in this chapter. One idea is to leverage a database of CAD models. We propose CORSAIR, a fully Convolutional Object Retrieval and Symmetry-AIded Registration method. The algorithm takes in an object point cloud with known category label and extracts a global object-shape embedding feature in addition to local point-wise features. The global feature is used to retrieve a similar object from a category CAD database, and the local features are used for robust pose registration between the observed and the retrieved object, leveraging symmetries present in the object shapes. The other idea is to use an implicit representation to model the shape. We propose ELLIPSDF, a bi-level object shape decoder model with a shared latent representation. On the coarse-level, an ellipsoid is used as a primitive shape to constrain the overall shape scale. On the fine-level, an expressive SDF model is used to preserve the object shape details. In summary, the **contributions** of this chapter are summarized as follows.

- We design a sparse fully convolutional network to jointly regress global and local point-cloud features, which are hierarchically correlated. The global feature enables similar model retrieval, while the local features allow object pose registration.
- We construct symmetry classes within an object instance based on the local features and aid the generation of promising feature pairs for robust registration.
- We propose a bi-level object model with coarse and fine levels, enabling joint optimization

of object pose and shape. The coarse-level uses a primitive shape for robust pose and scale initialization, and the fine-level uses SDF residual directly to allow accurate shape modeling. The two levels are coupled via a shared latent space.

- We design a cost function to measure the mismatch between the bi-level object model and the point cloud observations in the world frame.

3.1 Problem: Categorical Object Retrieval & Registration

Consider a robot, equipped with an RGBD camera, aiming to construct an object-level map of an unknown environment. Assume that the camera pose is estimated using an odometry algorithm, such as ORB-SLAM3 [19], and a convolutional neural network, such as Mask R-CNN [71], is used to detect and segment objects in each RGB image, and an object tracking algorithm, such as FairMOT [206], tracks the object detections over time. A partial point cloud observation $\mathbf{X} \in \mathbb{R}^{3\times N}$ of a tracked object instance can be obtained by accumulating the segmented RGBD pixels associated with the instance over time and projecting them to the world frame using the estimated camera pose trajectory.

Let $\mathcal{Y} := \left\{\mathbf{Y}_i \in \mathbb{R}^{3\times M_i}\right\}_i$ be a database of point cloud object models from the same category as $\mathbf{X}$. We assume the database was used offline for training the object detection and tracking models and is available to the robot. We consider the following joint object retrieval and registration problem.

Problem. *Given a query point cloud* $\mathbf{X} \in \mathbb{R}^{3\times N}$ *and a point cloud database* $\mathcal{Y} := \left\{\mathbf{Y}_i \in \mathbb{R}^{3\times M_i}\right\}_i$*, retrieve a point cloud* $\mathbf{Y}^* \in \mathcal{Y}$ *that is similar to* $\mathbf{X}$ *and estimate its rotation* $\mathbf{R}^* \in SO(3)$ *and translation* $\mathbf{p}^* \in \mathbb{R}^3$ *with respect to* $\mathbf{X}$ *such that the two point cloud align well:*

$$\mathbf{Y}^*, \mathbf{R}^*, \mathbf{p}^* = \underset{\mathbf{Y}\in\mathcal{Y}, \mathbf{R}\in SO(3), \mathbf{p}\in\mathbb{R}^3}{\arg\min} \; d(\mathbf{X}, \mathbf{R}\mathbf{Y} + \mathbf{p}\mathbf{1}^\top), \tag{3.1}$$

where $\mathbf{1}$ *is a vector with all elements equal to* 1 *and* d *is a distance metric between two point clouds.*

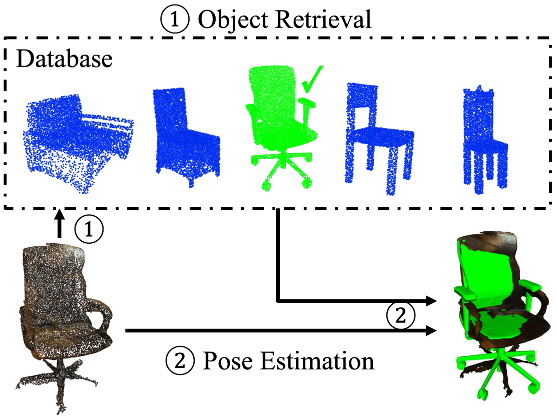

Figure 3.1. Given an observed object point-cloud, we focus on retrieving a similar object (green) from a category database (blue) and estimating its pose with respect to the input object. Performing retrieval and registration for object point-clouds observed online allow us to construct an object-level map of an unknown environment.

The objective of Problem 3.1 is to determine the world-frame pose of an object instance, observed online, which may or may not have been seen before. Retrieving a similar instance from the training database and registering it with the point cloud observation allows accurate pose estimation of the newly observed object. Fig. 3.1 illustrates the problem setting.

3.2 CORSAIR: Point Cloud Feature Extraction for Retrieval & Registration

Estimating the pose of novel objects based on a finite set of models from the same category can be challenging due to shape variation. In this chapter, we develop an approach for fully Convolutional Object Retrieval and Symmetry-AIded Registration (CORSAIR). We extend the point-wise FCGF feature extractor proposed in [31] with a global embedding network. We learn *local point-wise features* (Sec. 3.2.2) to enable robust matching and registration of point cloud with potentially different shapes. We learn *global object-level features* (Sec. 3.2.3) to enable retrieval of a point cloud from the database that is similar to the query point cloud. During inference (Sec. 3.2.4), we align a partially observed point cloud with a retrieved one, and exploit object symmetry to generate matching feature pairs for registration. Fig. 3.2 presents an

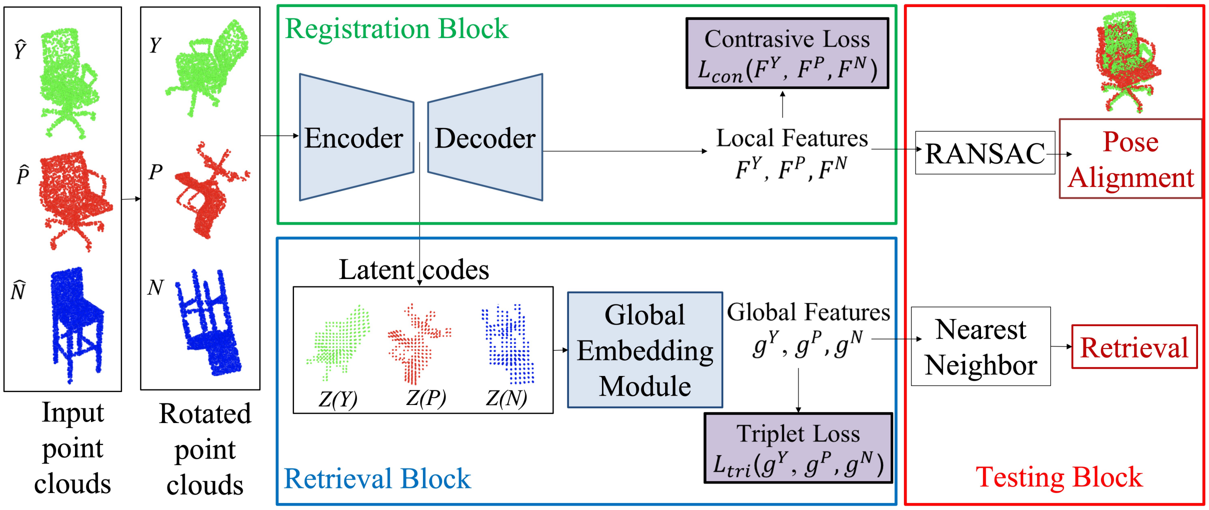

Figure 3.2. During training, given a point cloud and its corresponding positive and negative pairs $(\mathbf{Y}, \mathbf{P}, \mathbf{N})$, a Registration block is trained to generate local point-wise features (Sec. 3.2.2) and a Retrieval block is trained to generate a global shape embedding (Sec. 3.2.3). During testing (Sec. 3.2.4), given a query point cloud $\mathbf{X}$, we generate its global embedding $\mathbf{g}^{\mathbf{X}}$ and retrieve a similar instance using nearest neighbors in the embedding space. Local features are then generated for both point clouds, and matching pairs are used to recover the pose of the query using RANSAC.

overview of our approach.

3.2.1 Normalized Canonical Coordinate

The task of matching different objects is challenging. When scans are obtained from different object instance, the definition of good alignment becomes vague since a rigid transformation cannot eliminate the intrinsic shape difference. If the problem is restricted to aligning objects within the same category, we can leverage the conventions of object canonical pose. For example, a chair always has a seat and often a back and a canonical chair pose can be defined accordingly. The Normalized Object Coordinate Space (NOCS) is proposed in [189] and here we call this concept as Normalized Canonical Coordinate (NCC). The normalized object bounding box has a unit-length diagonal and is centered at the zero. Fig. 3.3 visualizes some examples of the chairs in the category NCC. The NCC of a object can be recovered given its pose and scale. NCC can bridge different objects with different poses and shapes as a intermediate pose-invariant shape representation.

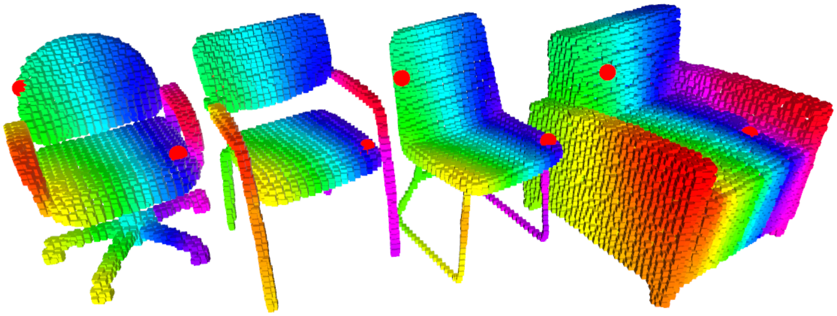

Figure 3.3. Visualization of different chairs in Normalized Canonical Coordinates (NCCs). Colors indicate the coordinates. The red dots annotate the same coordinate on different objects. Note that the red dot on the second chair is occluded by the back. Although not all points can be associated well in the NCCs, many positive matching pairs can be discovered.

3.2.2 Local Features for Pose Registration

We aim to predict matching pairs of point-wise local feature for pose registration between point clouds. During training, we first define matching pairs of points, rather than features. If two point clouds $\mathbf{X} \in \mathbb{R}^{3\times N}$ and $\mathbf{Y} \in \mathbb{R}^{3\times M}$ were already aligned in the same coordinate frame, matching point pairs can be extracted via:

$$p(\mathbf{X},\mathbf{Y}) = \{(i,j) \in \mathbb{N}^2 \mid \|\mathbf{x}_i - \mathbf{y}_j\| < \tau, i \leq N, j \leq M\}, \tag{3.2}$$

where $\tau > 0$ is a matching tolerance. Negative pairs can be obtained from the complement set of the positive pairs, ensuring that two negatively associated points are at least a margin τ away.

Instead of finding matching pairs within the same instance, we are trying to generate pairs between two different instances. Since the training set $\mathcal{Y}$ contains point clouds from different instances, which inherently carry geometric shape differences, we generate matching pairs in category-level normalized canonical coordinates (NCC) [189, 47]. We are not introducing new data but generate new matching pairs in the original database. This can be viewed as a data augmentation strategy for this specific feature learning task. Instead of dense correspondence annotation, we only need object pose annotations to convert the point clouds to NCC. Given the scale $s_{\mathbf{X}} \in \mathbb{R}$, rotation $\mathbf{R}_{\mathbf{X}} \in SO(3)$, and translation $\mathbf{p}_{\mathbf{X}} \in \mathbb{R}^3$ of a point cloud $\mathbf{X}$ during training, $\mathbf{X}$ can be converted to NCC via:

$$\mathbf{X}_{\text{NCC}} = s_{\mathbf{X}}^{-1} \cdot \mathbf{R}_{\mathbf{X}}^{\top} \left(\mathbf{X} - \mathbf{p}_{\mathbf{X}} \mathbf{1}^{\top} \right). \tag{3.3}$$

Thus, to generate matching pairs for different instances $\mathbf{X}$ and $\mathbf{Y}$ of the same category, we convert both into NCC, and obtain the positive pair set as $\mathcal{P}_{\mathcal{L}} = p(\mathbf{X}_{\text{NCC}}, \mathbf{Y}_{\text{NCC}})$. A negative pair set $\mathcal{N}_{\mathcal{L}}$ is obtained as a subset of the complement of $\mathcal{P}_{\mathcal{L}}$.

Given a point cloud $\mathbf{X} \in \mathbb{R}^{3 \times N}$, our model uses a sparse fully convolutional encoder-decoder architecture, illustrated in Fig. 3.4, to predict local point-wise features:

$$\mathbf{F}^{\mathbf{X}} = [\mathbf{f}_1^{\mathbf{x}}, \ldots, \mathbf{f}_N^{\mathbf{x}}] \in \mathbb{R}^{C \times N}, \tag{3.4}$$

where $\mathbf{f}_i^{\mathbf{x}}$ is the feature corresponding to the point $\mathbf{x}_i$. Sparse convolution [30] generalizes image convolution to arbitrary dimensions and coordinates and allows processing of spatially sparse inputs. Our model is an extension of the FCGF model [31] that adds an embedding module to the encoder output (bottleneck layer) to also retrieve a global feature. The training and role of the global feature for retrieval is described in Sec. 3.2.3.

We use metric learning to train the local feature extractor. Relying on the positive pairs $\mathcal{P}_{\mathcal{L}}$ and the negative pairs $\mathcal{N}_{\mathcal{L}}$ of matching points, we define a contrastive loss function for the features $\mathbf{F}^{\mathbf{X}}$ and $\mathbf{F}^{\mathbf{Y}}$ associated with two point clouds from the training set:

$$L_{\text{con}}(\mathbf{F}^{\mathbf{X}}, \mathbf{F}^{\mathbf{Y}}) = \sum_{(i,j) \in \mathcal{P}_{\mathcal{L}}} \max(0, \|\mathbf{f}_i^{\mathbf{x}} - \mathbf{f}_j^{\mathbf{y}}\|_2 - p_+)^2 + \sum_{(i,j) \in \mathcal{N}_{\mathcal{L}}} \max(0, p_- - \|\mathbf{f}_i^{\mathbf{x}} - \mathbf{f}_j^{\mathbf{y}}\|_2)^2, \tag{3.5}$$

where p_+ and p_- are the positive and negative thresholds. These thresholds are selected to ensure that points from the positive pairs move closer together and points from the negative pairs move farther apart in the feature space. We normalize the feature vectors to unit length and set

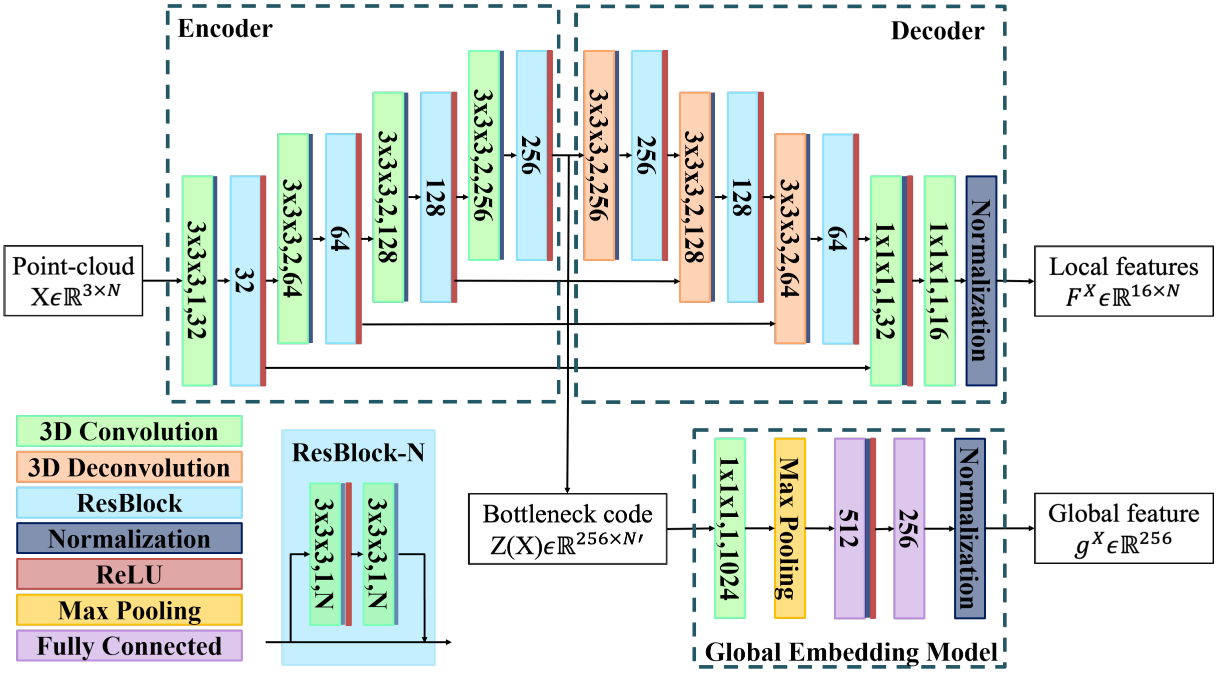

Figure 3.4. Our model extends the sparse convolutional encoder-decoder ResUNet structure proposed in FCGF [31] by adding an embedding module to the bottleneck latent code (encoder output). The output of our embedding module provides a global object shape feature, suitable for retrieval, while the decoder generates point-wise local features. The numbers in the 3D convolution and deconvolution blocks represent kernel sizes, strides, and output dimensions. The numbers in the fully connected blocks represent the output dimensions. Each ResBlock is composed of two 3D convolution layers and the number indicates the output dimensions.

$p_+ = 0.1$ and $p_- = 1.5$.

For each point cloud $\mathbf{Y}$ in the training set $\mathcal{Y}$, we choose similar point clouds $\mathbf{P}$ and dissimilar point clouds $\mathbf{N}$, which are defined precisely in Sec. 3.2.3. We, then, generate the positive pairs between $\mathbf{Y}$ and $\mathbf{P}$ using (3.2) as $\mathcal{P}_{\mathcal{L}} = p(\mathbf{Y}_{NCC}, \mathbf{P}_{NCC})$. We sample the negative pair set $\mathcal{N}_{\mathcal{L}}$ by taking random pairs between $\mathbf{Y}$ and $\mathbf{N}$ as well as from the complement of $\mathcal{P}_{\mathcal{L}}$. The local feature extractor model is trained with the contrastive loss in (3.5) but using the pairs from $\mathcal{P}_{\mathcal{L}}$ and $\mathcal{N}_{\mathcal{L}}$.

3.2.3 Global Feature for Object Retrieval

Extracting a similar point cloud from $\mathcal{Y}$ for a given query point cloud $\mathbf{X}$ is crucial for pose registration because only similar shapes provide consistent local geometric features for matching. Since the sparse convolutional model in Fig. 3.4 performs in providing local geometric features, we design a global shape descriptor by combining the local features. An input point cloud $\mathbf{X}$ is quantized into a sparse tensor and gets downsampled by the encoder into a point cloud $\mathbf{Z}(\mathbf{X}) \in \mathbb{R}^{256\times N'}$ with fewer points $N' < N$ but each in a higher (256 in our case) dimension. After the multiple convolution layers of the encoder, the bottleneck code $\mathbf{Z}(\mathbf{X})$ encodes a high-level structure feature which should be beneficial for shape retrieval. We introduce an embedding module to extract a single global feature $\mathbf{g}^{\mathbf{X}} \in \mathbb{R}^{256}$ from $\mathbf{Z}(\mathbf{X})$ as $\mathbf{g}^{\mathbf{X}} = g(\mathbf{Z}(\mathbf{X}))$. As shown in Fig. 3.4, the embedding module $g(\cdot)$, includes a fully convolutional layer, followed by maxpooling to combine the features from all points in $\mathbf{Z}(\mathbf{X})$ and pass them through several fully connected layers to obtain $\mathbf{g}^{\mathbf{X}}$.

We use metric learning for global feature training. To measure the similarity of point cloud $\mathbf{X} \in \mathbb{R}^{3\times N}$ with respect to $\mathbf{Y} \in \mathbb{R}^{3\times M}$, we define a single-direction Chamfer Distance (SCD):

$$d_{SCD}(\mathbf{X},\mathbf{Y}) = \frac{1}{N}\sum_{i=1}^{N}\min_{j=1}^{M}\|\mathbf{x}_i - \mathbf{y}_j\|_2^2. \tag{3.6}$$

The bi-directional similarity between the two point clouds is measured by the usual Chamfer distance:

$$d_{CD}(\mathbf{X},\mathbf{Y}) = d_{SCD}(\mathbf{X},\mathbf{Y}) + d_{SCD}(\mathbf{Y},\mathbf{X}). \tag{3.7}$$

Let $\mathbf{D} \in \mathbb{R}^{|\mathcal{Y}|\times|\mathcal{Y}|}$ encode the pair-wise Chamfer distance similarity of all point clouds in $\mathcal{Y}$ with elements:

$$\mathbf{D}_{i,j} = d_{CD}(\mathbf{Y}_i, \mathbf{Y}_j), \text{ for } \mathbf{Y}_i, \mathbf{Y}_j \in \mathcal{Y}. \tag{3.8}$$

A similarity ranking for $\mathbf{Y}_i \in \mathcal{Y}$ can be obtained by sorting the i-th column of $\mathbf{D}$ in ascending order. We define a positive set $\mathcal{P}_{\mathcal{G}}(\mathbf{Y}_i)$ and negative set $\mathcal{N}_{\mathcal{G}}(\mathbf{Y}_i)$ of point clouds associated with $\mathbf{Y}_i$ as follows:

$$\begin{aligned}\mathcal{P}_{\mathcal{G}}(\mathbf{Y}_i) &= \{\mathbf{Y}_j \mid \mathrm{Rank}_i(\mathbf{Y}_j) \leq \tau_+|\mathcal{Y}|, d_{CD}(\mathbf{Y}_i,\mathbf{Y}_j) \leq \delta_+\},\\ \mathcal{N}_{\mathcal{G}}(\mathbf{Y}_i) &= \{\mathbf{Y}_j \mid \mathrm{Rank}_i(\mathbf{Y}_j) \geq \tau_-|\mathcal{Y}|, d_{CD}(\mathbf{Y}_i,\mathbf{Y}_j) \geq \delta_-\},\end{aligned} \tag{3.9}$$

where $\text{Rank}_i(\mathbf{Y})$ returns an integer indicating the Chamfer distance ranking of $\mathbf{Y}$ to $\mathbf{Y}_i$, τ_+ and τ_- are the percentage of positive and negative point clouds we want to consider, and δ_+ and δ_- are the Chamfer distance thresholds. In our experiments, we set $\tau_+ = 0.1$, $\tau_- = 0.5$, $\delta_+ = 0.15$ and $\delta_- = 0.20$.

For a given point cloud $\mathbf{Y}$, we randomly select one positive object $\mathbf{P} \in \mathcal{P}_\mathcal{G}(\mathbf{Y})$ and one negative object $\mathbf{N} \in \mathcal{N}_\mathcal{G}(\mathbf{Y})$ and train the global embedding module with a triplet loss:

$$L_{\text{tri}}(\mathbf{g}^\mathbf{Y}, \mathbf{g}^\mathbf{P}, \mathbf{g}^\mathbf{N}) = \max(1 + \|\mathbf{g}^\mathbf{Y} - \mathbf{g}^\mathbf{P}\|_2 - \|\mathbf{g}^\mathbf{Y} - \mathbf{g}^\mathbf{N}\|_2, 0). \tag{3.10}$$

Similar to the contrastive loss for the local features, the triplet loss pushes similar point clouds closer and drags dissimilar point clouds apart in the global embedding space. The positive object $\mathbf{P}$ and the negative object $\mathbf{N}$ are also used to generate point pairs for local feature training in Sec. 3.2.2.

3.2.4 Retrieval and Registration Inference

Finally, we consider object retrieval and pose registration for a query point cloud $\mathbf{X}$ given the trained CORSAIR model in Fig. 3.4. The first step is to retrieve a similar object from $\mathcal{Y}$. The local and global features of all point clouds in $\mathcal{Y}$ are pre-computed offline. The query $\mathbf{X}$ is passed through the CORSAIR model to obtain its local features $\mathbf{F}^\mathbf{X}$ and global feature $\mathbf{g}^\mathbf{X}$. The instance from $\mathbf{Y}$ closest to $\mathbf{X}$ is retrieved via:

$$\mathbf{Y} = \operatorname*{arg\,min}_{\mathbf{Y_i} \in \mathcal{Y}} \|\mathbf{g}^\mathbf{X} - \mathbf{g}^{\mathbf{Y_i}}\|_2, \tag{3.11}$$

Next, we generate matching pairs between $\mathbf{X}$ and $\mathbf{Y}$ by searching for K nearest neighbors in the local feature space:

$$nn_K(\mathbf{F}^\mathbf{X}, \mathbf{F}^\mathbf{Y}) = \{(i, j_i) \mid j_i \in K\text{-}\operatorname*{arg\,min}_j \|\mathbf{f}_i^\mathbf{x} - \mathbf{f}_j^\mathbf{y}\|_2, i \leq N, j_i \leq M, i, j_i \in \mathbb{N}\}, \tag{3.12}$$

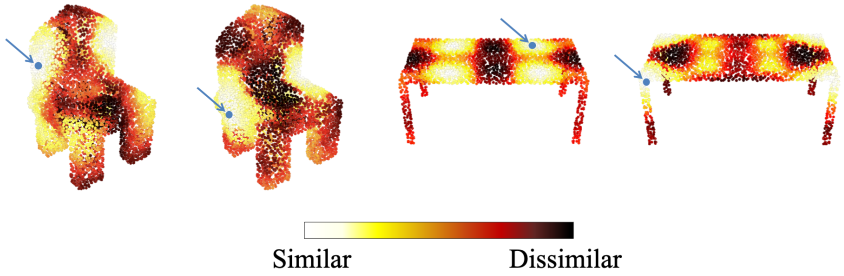

Figure 3.5. Heatmap of local feature similarity for different points on a chair (left) and a table (right) instances. Lighter points indicate points with similar features to the query point (blue dot), while darker ones have dissimilar features. The feature similarity heatmap visualizes the symmetric nature of the local features.

Algorithm 1. Symmetry-aided Segmentation

```
input: point cloud $\mathbf{X}$, point-wise local features $\mathbf{F}^{\mathbf{X}}$, number of matching feature pairs $M$, number of symmetry classes $G$
Randomly sample $\mathcal{S} \leftarrow \{\mathbf{x}_i | \mathbf{x}_i \in \mathbf{X}, i \leq n\}$
for $\mathbf{x}_i \in \mathcal{S}$ do
    $\mathcal{K} \leftarrow \{\mathbf{x}_j | (i,j) \in nn_K(\mathbf{f}_i^{\mathbf{x}}, \mathbf{F}^{\mathbf{X}})\}$
    Split $\mathbf{X}$ in clusters $\mathcal{C}_i = \{\widetilde{\mathbf{X}}_1, \ldots, \widetilde{\mathbf{X}}_G\}$ via GMEANS($\mathcal{K}$)
    $\sigma_i \leftarrow$ Standard Deviation of $\{|\widetilde{\mathbf{X}}_1|, \ldots, |\widetilde{\mathbf{X}}_G|\}$
output: $\mathcal{C}_i$ with the smallest $\sigma_i$
```

where we define K-$\arg\min_j$ as the set of K different values that make the function smaller than any other set of K indices. The correspondence candidates in $nn_K(\mathbf{F}^{\mathbf{X}}, \mathbf{F}^{\mathbf{Y}})$ are used to recover the rotation and translation of $\mathbf{Y}$ with respect to $\mathbf{X}$ via a robust pose estimation method such as RANSAC [51].

Artificial objects usually have one or more planes of symmetry. Since our model is able to generate rotation-invariant local features, the features from symmetrical areas can be similar, which significantly increases the risk of mismatch in (3.12). A feature-distance heatmap shown in Fig. 3.5 illustrates the symmetrical patterns. We propose a symmetry-aware method to add constraints in the nearest neighbor matching phase.

We split the point cloud with a symmetric segmentation method (see Alg. 1) and then constrain the nearest-neighbor matching to the corresponding parts. Given an object category, we assume that the number of symmetric classes G, computed as the number of symmetry planes

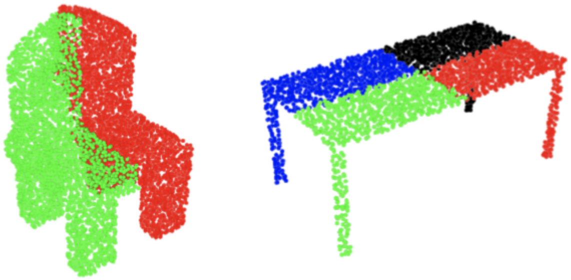

Figure 3.6. Different objects split according to their symmetry planes.

times 2, is known. For example, $G = 2$ for chairs and $G = 4$ for tables. We first extract the point-wise local features $\mathbf{F}^{\mathbf{X}}$ for the input point cloud $\mathbf{X}$. Second, we randomly sample n points from the point cloud. For each sampled point, we take its K nearest neighbors, $nn_K(\mathbf{f}_i^{\mathbf{X}}, \mathbf{F}^{\mathbf{X}})$, and perform G-means clustering using their 3D spatial coordinates. The object can then be split into G parts, $\{\widetilde{\mathbf{X}}_1, \ldots, \widetilde{\mathbf{X}}_G\}$, by the decision boundaries of G-means clustering as shown in Fig. 3.6. Each of the splits is considered as a candidate and we choose the most even split as our symmetric segmentation output. The evenness of a split is measured by the standard deviation σ of the sizes of the G parts.

We assume that the query and retrieved point clouds $\mathbf{X}$ and $\mathbf{Y}$ share the same symmetry property and split them with our symmetry segmentation method. Since there are multiple possible mappings between the subsets $\{\widetilde{\mathbf{X}}_{\mathbf{1}}, \ldots, \widetilde{\mathbf{X}}_G\}$ and $\{\widetilde{\mathbf{Y}}_{\mathbf{1}}, \ldots, \widetilde{\mathbf{Y}}_G\}$, we generate matching pairs using (3.12) for all the possible mappings. We also generate matching pairs without the symmetry constraints as a back-up for asymmetric objects. We supply these sets of matching pairs to RANSAC to estimate the rotation $\mathbf{R}$ and translation $\mathbf{p}$ that align $\mathbf{Y}$ with $\mathbf{X}$ for every possible matching pair set. The quality of alignment is evaluated by the single direction Chamfer distance $d_{SCD}(\mathbf{X}, \mathbf{RY} + \mathbf{p1}^\top)$ defined in (3.6). The rotation and translation with the best alignment quality are selected as the output of our symmetry-aware pose estimation method. Our symmetry-aware method performs well when the input point clouds have symmetrical structure.

3.3 Problem: Categorical Object Pose & Shape Optimization

Instead of the explicit point cloud representation proposed in Sec. 3.1, we can also consider using the implicit shape representation. We follow the same assumptions that an RGB-D camera is tracked. The camera moves in an unknown environment that contains N objects $\mathcal{O} \triangleq \{\mathbf{o}_n\}_{n=1}^{N}$. Each object $\mathbf{o}_n = (\mathbf{c}_n, \mathbf{i}_n)$ is an instance $\mathbf{i}_n$ of class $\mathbf{c}_n$, defined below. The object segmentations on the RGB-D images across frames are associated, which can be accumulated as an object point cloud.

There are two levels of implicit shape representation we consider here. The coarse level shape is represented using a *quadric shape* [70], $\left\{\mathbf{x} \in \mathbb{R}^3 \mid \underline{\mathbf{x}}^\top \mathbf{Q} \underline{\mathbf{x}} = 0\right\}$, where $\underline{\mathbf{x}} \triangleq [\mathbf{x}^\top, 1]^\top$ denotes the homogeneous coordinates of $\mathbf{x}$ and $\mathbf{Q} \in \mathbb{R}^{4\times 4}$ is a symmetric matrix. An axis-aligned ellipsoid centered at the origin is:

$$\mathcal{E}_{\mathbf{u}} \triangleq \left\{\mathbf{x} \in \mathbb{R}^3 \mid \mathbf{x}^\top \mathbf{U}^{-\top} \mathbf{U}^{-1} \mathbf{x} = 1\right\}, \tag{3.13}$$

where $\mathbf{U} \triangleq \operatorname{diag}(\mathbf{u})$ and the elements of the vector $\mathbf{u} \in \mathbb{R}^3$ specify the lengths of the semi-axes of $\mathcal{E}_{\mathbf{u}}$. An ellipsoid $\mathcal{E}_{\mathbf{u}}$ is a special case of a quadric shape with $\mathbf{Q} = \operatorname{diag}(\mathbf{U}^{-2}, -1)$. A quadric shape can also be defined in dual form, as the set of planes $\underline{\pi} = \mathbf{Q}\underline{\mathbf{x}}$ that are tangent to the shape surface at each $\mathbf{x}$. This dual quadric surface definition is $\left\{\pi \in \mathbb{R}^3 \mid \underline{\pi}^\top \mathbf{Q}^* \underline{\pi} = 0\right\}$, where $\mathbf{Q}^* = \operatorname{adj}(\mathbf{Q})$ is the adjugate of $\mathbf{Q}$. A dual quadric defined by $\mathbf{Q}^*$ can be scaled, rotated, or translated by a similarity transform $\mathbf{T} \in SIM(3)$ as $\mathbf{T}\mathbf{Q}^*\mathbf{T}^\top$. The fine shape of a rigid body is represented as $\left\{\mathbf{x} \in \mathbb{R}^3 \mid f(\mathbf{x}) = 0\right\}$ using the *signed distance function* (SDF) of a set $\mathcal{S} \subset \mathbb{R}^3$:

$$f(\mathbf{x}) = \begin{cases} -d(\mathbf{x}, \partial\mathcal{S}), & \mathbf{x} \in \mathcal{S}, \\ d(\mathbf{x}, \partial\mathcal{S}), & \mathbf{x} \notin \mathcal{S}, \end{cases} \tag{3.14}$$

where $d(\mathbf{x}, \partial\mathcal{S})$ denotes the Euclidean distance from a point $\mathbf{x} \in \mathbb{R}^3$ to the boundary $\partial\mathcal{S}$ of $\mathcal{S}$.

Definition 1. *An* object class *is a tuple* $\mathbf{c} \triangleq \left(\nu, \mathbf{z}, f_\theta, g_\phi\right)$*, where* $\nu \in \mathbb{N}$ *is the class identity, e.g., chair, table, sofa, and* $\mathbf{z} \in \mathbb{R}^d$ *is a latent code vector, encoding the average class shape. The class shape is represented in a canonical coordinate frame at two levels of granularity: coarse and fine. The coarse shape is specified by an ellipsoid* $\mathcal{E}_{\mathbf{u}}$ *in* (3.13) *with semi-axis lengths* $\mathbf{u} = g_\phi(\mathbf{z})$ *decoded from the latent code* $\mathbf{z}$ *via a function* $g_\phi : \mathbb{R}^d \mapsto \mathbb{R}^3$ *with parameters* ϕ*. The fine shape is specified by the signed distance* $f_\theta(\mathbf{x}, \mathbf{z})$ *in* (3.14) *from any* $\mathbf{x} \in \mathbb{R}^3$ *to the average shape surface, decoded from the latent code* $\mathbf{z} \in \mathbb{R}^d$ *via a function* $f_\theta : \mathbb{R}^3 \times \mathbb{R}^d \mapsto \mathbb{R}$ *with parameters* θ*.*

Definition 2. *An* object instance *of class* $\mathbf{c}$ *is a tuple* $\mathbf{i} \triangleq (\mathbf{T}, \delta\mathbf{z})$*, where* $\mathbf{T} \in SIM(3)$ *specifies the transformation from the global frame to the object instance frame, and* $\delta\mathbf{z} \in \mathbb{R}^d$ *is a deformation of the class average shape latent code* $\mathbf{z}$*.*

Our goal is to estimate the transformation and shape $\mathbf{i}_n := (\mathbf{T}_n, \delta\mathbf{z}_n)$ of each observed object n. We consider object instances independently and drop the subscript n when it is clear from the context. Given an object with multi-view segmentation, we use the depth $D(\mathbf{p})$ of each pixel $\mathbf{p}$ to obtain a set of points $\mathcal{X}(\mathbf{p})$ along the ray starting from the camera optical center and passing through $\mathbf{p}$. The sets $\mathcal{X}(\mathbf{p})$ is used to optimize the pose and shape of the object instance. For each ray, we choose three points, one lying on the observed surface, one a small distance $\varepsilon > 0$ in front of the surface, and one a small distance ε behind. Given $d \in \{0, \pm\varepsilon\}$, we obtain points $\mathbf{y} \in \mathbb{R}^3$ in the optical frame corresponding to the pixels $\mathbf{p}$:

$$\mathcal{Y}(\mathbf{p}) \triangleq \left\{ (\mathbf{y}, d) \middle| \mathbf{y} = \left(D(\mathbf{p}) + \frac{d}{\|\underline{\mathbf{p}}\|} \right) \underline{\mathbf{p}},\ d \in \{0, \pm\varepsilon\} \right\}, \tag{3.15}$$

and project them to the global frame using the known camera pose $\mathbf{C}$:

$$\mathcal{X}(\mathbf{p}) \triangleq \left\{ (\mathbf{x}, d) \middle| \mathbf{x} = \mathbf{P}\mathbf{C}\underline{\mathbf{y}},\ (\mathbf{y}, d) \in \mathcal{Y}(\mathbf{p}) \right\}. \tag{3.16}$$

where $\mathbf{P} = [\mathbf{I} \quad \mathbf{0}]$. We use $\{\mathcal{X}(\mathbf{p})\}$ to represent the combined distance-labeled point set of all the valid pixels of all the observed frames. We define an error function e_ϕ to measure the discrepancy between a distance-labeled point $(\mathbf{x}, d) \in \mathcal{X}(\mathbf{p})$ observed close to the instance surface and the coarse shape $\mathcal{E}_{\mathbf{u}}$ provided by $\mathbf{u} = g_{\phi}(\mathbf{z})$. Another error function e_{θ} is used for the difference between $(\mathbf{x}, d)$ and the SDF value $f_{\theta}(\mathbf{x}, \mathbf{z})$ predicted by the fine shape model. The overall error function is defined as:

$$e_{\theta,\phi}(\mathbf{T}, \mathbf{z}, \delta\mathbf{z}; \{\mathcal{X}(\mathbf{p})\}) \triangleq e_r(\delta\mathbf{z}) + \sum_{(\mathbf{x},d)\in\{\mathcal{X}(\mathbf{p})\}} e_{\theta}(\mathbf{T}, \mathbf{z}, \delta\mathbf{z}; \mathbf{x}, d) + e_{\phi}(\mathbf{T}, \mathbf{z}, \delta\mathbf{z}; \mathbf{x}, d), \tag{3.17}$$

where $e_r(\delta\mathbf{z})$ is a shape deformation regularization term. The coarse-shape error, e_{θ}, fine-shape error, e_{ϕ}, and the regularization, e_r are defined precisely in Sec. 3.4.2.

We distinguish between a training phase, where we optimize the parameters $\mathbf{z}$, θ, ϕ of an object class using offline data from instances with known mesh shapes, and a testing phase, where we optimize the pose $\mathbf{T}$ and shape deformation $\delta\mathbf{z}$ of a previously unseen instance from the same category using online distance data from an RGB-D camera. In training, we generate points $\{\mathcal{X}(\mathbf{p})\}$ close to the surface of each available mesh model in a canonical coordinate frame (with identity pose $\mathbf{I}_4$) and optimize the class shape parameters as defined in Definition 1 via:

$$\min_{\mathbf{z},\theta,\phi} e_{\theta,\phi}(\mathbf{I}_4, \mathbf{z}, \mathbf{0}; \{\mathcal{X}(\mathbf{p})\}). \tag{3.18}$$

In testing, we receive points $\{\mathcal{X}(\mathbf{p})\}$ in the global frame, generated by the RGB-D camera from the surface of a previously unseen instance. Assuming known object class, we fix the trained shape parameters $\mathbf{z}^*$, θ^*, ϕ^* and optimize the unknown instance transform $\mathbf{T} \in \mathrm{SIM}(3)$ and shape deformation $\delta\mathbf{z} \in \mathbb{R}^d$ as defined in Definition 2:

$$\min_{\mathbf{T},\delta\mathbf{z}} e_{\theta^*,\phi^*}(\mathbf{T}, \mathbf{z}^*, \delta\mathbf{z}; \{\mathcal{X}(\mathbf{p})\}). \tag{3.19}$$

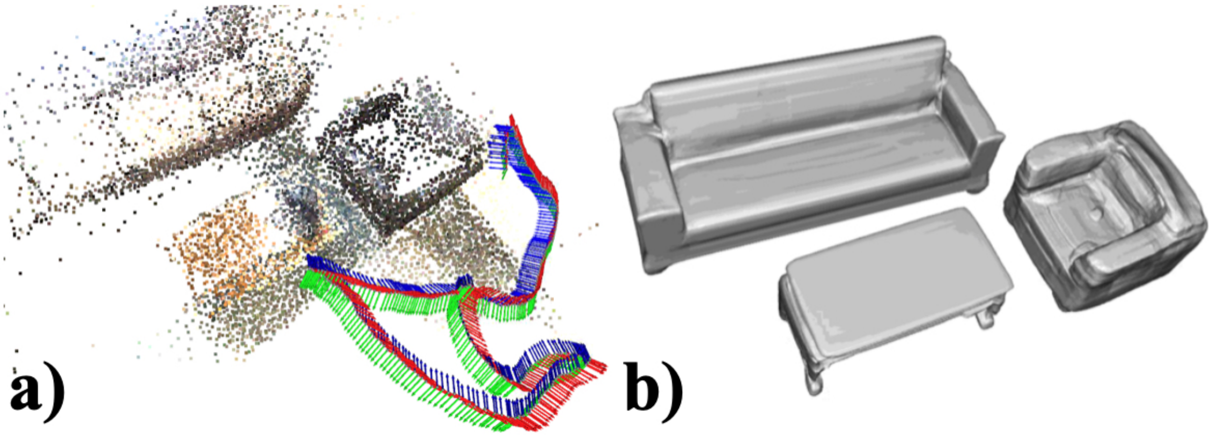

Figure 3.7. Overview of ELLIPSDF: a) Ground-truth scene reconstruction from colored point clouds in ScanNet scene 0087, where the RGB axes show the camera trajectory, b) Reconstructed object meshes in the world frame using the SDF model decoded from a latent code, and the optimized SIM(3) transformation representing object pose.

3.4 ELLIPSDF: Object Pose and Shape Optimization with Bi-level Shape Representation

In this chapter, we proposes ELLIPSDF, an expressive yet compact model of object pose and shape and an associated optimization algorithm to infer an object-level map from multi-view RGB-D camera observations, as shown in Fig. 3.7. ELLIPSDF is expressive because it captures the identity, scale, position, orientation, and shape of objects in the environment. It is compact because it relies on a low-dimensional latent encoding of the signed distance function (SDF) to an object's surface, allowing onboard storage of large multi-category object maps. We first present the model and define the error functions for its parameter optimization. Then we describe how a trained ELLIPSDF model is used at test time for multi-view joint optimization of object pose and shape. An overview is shown in Fig. 3.8.

3.4.1 Bi-level Shape Representation

We choose to use the autodecoder [13, 138] to model the object shape. The autodecoder takes in a latent code and outputs the desired shape, depending on the shape representation. During training, we simultaneously optimize the latent code assigned to each object and the

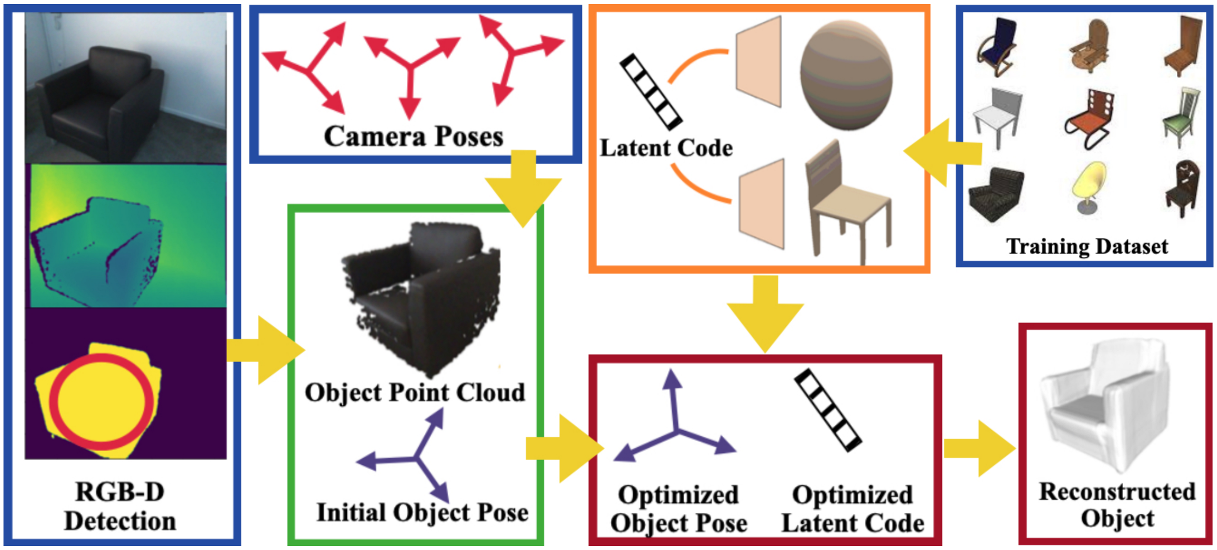

Figure 3.8. ELLIPSDF Overview: A point cloud and initial pose (*green*) are obtained from RGB-D detections of a chair instance from known camera poses (*blue*). A bi-level category shape description, consisting of a latent shape code, a coarse shape decoder, and a fine shape decoder (*orange*), is trained offline using a dataset of mesh models. Given the observed point cloud, the pose and shape deformation of the newly seen instance are optimized jointly online, achieving shape reconstruction in the global frame (*red*).

decoder weights. During inference, an optimal latent code is searched by optimization to minimize the error w.r.t. the observation with fixed decoder parameters. The ELLIPSDF shape model consists of two autodecoders $g_\phi(\mathbf{z})$ and $f_\theta(\mathbf{x},\mathbf{z})$, using a shared latent code $\mathbf{z} \in \mathbb{R}^d$. The first autodecoder $g_\phi(\mathbf{z})$ provides a *coarse* shape representation as an axis-aligned ellipsoid $\mathcal{E}_\mathbf{u}$ in a canonical coordinate frame with semi-axis lengths $\mathbf{u} = g_\phi(\mathbf{z}) \in \mathbb{R}^3$. The second autoencoder $f_\theta(\mathbf{x},\mathbf{z})$ provides a *fine* shape representation with parameters θ, as an implicit SDF surface $\left\{\mathbf{x} \in \mathbb{R}^3 \mid f_\theta(\mathbf{x},\mathbf{z}) = 0\right\}$ in the same canonical coordinate frame. The reparametrization trick [96] is used to maintain a Gaussian distribution $\mathbf{z} = \mu + \text{diag}(\sigma)\varepsilon$ over the latent code with $\varepsilon \sim \mathcal{N}(\mathbf{0},\mathbf{I})$ a random variable. Thus, at training time, the ELLIPSDF model parameters include the mean $\mu \in \mathbb{R}^d$ and standard deviation $\sigma \in \mathbb{R}^d$ of the latent shape code, the coarse shape autodecoder parameters ϕ, and the fine shape autodecoder parameters θ. The model is visualized in Fig. 3.9.

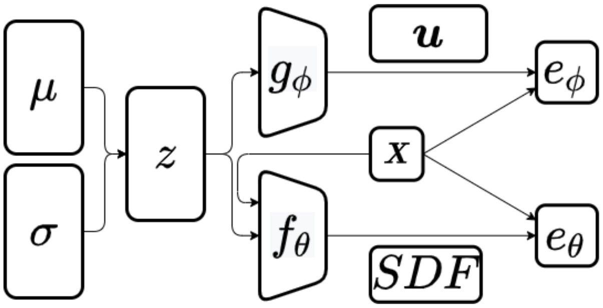

Figure 3.9. ELLIPSDF bi-level object shape decoder model. A latent shape code, **z**, with distribution $\mathcal{N}(\mu, \mathrm{diag}(\sigma)^2)$ is shared by a coarse shape decoder g_ϕ, providing an ellipsoid shape description, and a fine shape decoder f_θ, providing an SDF shape description. During training, the decoder parameters ϕ and θ are optimized by minimizing the errors between the SDF values of the training points **x**, obtained close to the object surface, and the coarse and fine shape models.

3.4.2 Error Functions

We introduce the error functions that play a key role for optimizing the category-level latent code $\mathbf{z}$ and decoder parameters θ, ϕ during the training time. They are also used for optimizing the transformation $\mathbf{T}$ from the global frame to the canonical object frame and the latent code deformation $\delta\mathbf{z}$ of a particular instance during the test time. The training data for an ELLIPSDF model consists of distance-labeled point clouds $\{\mathcal{X}(\mathbf{p})\}_n$ associated with instances n from the same class, as introduced in Sec. 3.3. A different latent code $\mathbf{z}_n$ is optimized for each instance n, while the decoder parameters θ and ϕ are common for all instances of the same class.

The fine-level shape error function e_θ of a point $\mathbf{x}$ in global coordinates with signed distance label d is defined as:

$$e_\theta(\mathbf{T}, \mathbf{z}, \delta\mathbf{z}; \mathbf{x}, d) \triangleq \rho(s f_\theta(\mathbf{P}\mathbf{T}\underline{\mathbf{x}}; \mathbf{z} + \delta\mathbf{z}) - d). \tag{3.20}$$

In the definition above, the point $\mathbf{x}$ is first transformed to the object coordinate frame via $\mathbf{P}\mathbf{T}\underline{\mathbf{x}}$ ($\mathbf{P} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{3\times4}$) and the fine-shape model f_θ is queried with the instance shape code $\mathbf{z} + \delta\mathbf{z}$ to predict the SDF to the object surface. Since SDF values vary proportionally with scaling [1], the returned value is scaled back by s before measuring its discrepancy with the label d. Instead of measuring the difference between sf_θ and d in absolute value, we employ a Huber term [83] to make the error function robust against outliers:

$$\rho(r) \triangleq \begin{cases} \frac{1}{2}r^2 & |r| \leq \delta, \\ \delta(|r| - \frac{1}{2}\delta) & \text{else.} \end{cases} \tag{3.21}$$

Note that the error e_θ relates both the object pose and shape to the SDF residual, which enables their joint optimization.

The coarse-level shape error function e_ϕ is defined similarly, using a signed distance function for the coarse shape. Since the coarse shape decoder, $\mathbf{u} = g_\phi(\mathbf{z})$, provides an explicit ellipsoid description, we first need a conversion to SDF before we can define the error term. An approximation of the SDF of an ellipsoid $\mathcal{E}_\mathbf{u}$ with semi-axis lengths $\mathbf{u}$ [146] can be obtained as:

$$h(\mathbf{x}, \mathbf{u}) = \frac{\left\|\mathbf{U}^{-1}\mathbf{x}\right\|_2 \left(\left\|\mathbf{U}^{-1}\mathbf{x}\right\|_2 - 1\right)}{\left\|\mathbf{U}^{-\mathbf{2}}\mathbf{x}\right\|_2}. \tag{3.22}$$

Then, the coarse-level shape error of a point $\mathbf{x}$ in global coordinates with signed distance label d is defined as:

$$e_\phi(\mathbf{T}, \mathbf{z}, \delta\mathbf{z}; \mathbf{x}, d) \triangleq \rho(sh(\mathbf{PT}\underline{\mathbf{x}}, g_\phi(\mathbf{z} + \delta\mathbf{z})) - d). \tag{3.23}$$

During training, the object transformation is fixed to be the canonical coordinate frame $\mathbf{T} = \mathbf{I}_4$ because the training point-cloud data is collected directly in the object frame. The regularization term $e_r(\delta\mathbf{z})$ in (3.17) is defined as the KL divergence between the distribution of $\delta\mathbf{z}$ and a standard normal distribution [69].

3.4.3 Ellipsoid Pose and Shape Initialization

We follow [155, 58] to initialize the SIM(3) scale and pose of an observed object, relying on its coarse ellipsoid shape representation. We fit ellipses to the pixel-wise segmentation Ω^2 of an object:

$$\left\{\mathbf{q} \in \Omega^2 \mid (\mathbf{q}-\mathbf{c})^\top \mathbf{E}^{-1}(\mathbf{q}-\mathbf{c}) = 1\right\}, \tag{3.24}$$

where the center and symmetric matrix are obtained as $\mathbf{c} = \frac{1}{|\Omega^2|}\sum_{\mathbf{p}\in\Omega^2}\mathbf{p}$ and $\mathbf{E} = \frac{1}{|\Omega^2|}\sum_{\mathbf{p}\in\Omega^2}(\mathbf{p}-\mathbf{c})(\mathbf{p}-\mathbf{c})^\top$. The axes lengths are the eigenvalues λ_0, λ_1 of $\mathbf{E}_k$. The 2D quadric surface corresponding to the ellipse in (3.24) and its dual are defined by the matrix $\mathbf{H}$ and its inverse $\mathbf{H}^*$:

$$\mathbf{H} = \begin{bmatrix} \mathbf{E}^{-1} & -\mathbf{E}^{-1}\mathbf{c} \\ -\mathbf{c}^\top\mathbf{E}^{-1} & \mathbf{c}^\top\mathbf{E}^{-1}\mathbf{c}-1 \end{bmatrix}, \ \mathbf{H}^* = \begin{bmatrix} \mathbf{E}-\mathbf{c}\mathbf{c}^\top & -\mathbf{c} \\ -\mathbf{c}^\top & -1 \end{bmatrix}. \tag{3.25}$$

An ellipsoid in dual quadric form $\mathbf{Q}^*$ in global coordinates and its conic projection $\mathbf{H}^*$ are related by $\beta\mathbf{H}^* = \mathbf{P}\mathbf{C}^{-1}\mathbf{Q}^*\mathbf{C}^{-\top}\mathbf{P}^\top$ defined up to a scale factor β. This equation can be rearranged to $\beta\mathbf{h} = \mathbf{G}\mathbf{v}$, where $\mathbf{h} = \mathrm{vech}(\mathbf{H}^*)$, $\mathbf{h} \in \mathbb{R}^6$, $\mathbf{v} = \mathrm{vech}(\mathbf{Q}^*)$ and $\mathbf{v} \in \mathbb{R}^{10}$. The operator vech serializes the lower triangular part of a symmetric matrix, and $\mathbf{G}$ is a matrix that depends on $\mathbf{P}\mathbf{C}^{-1}$. The explicit form of $\mathbf{G}$ is derived in Eq. (5) in [155]. Next, a least squares system is constructed from the multi-view observations. By stacking all observations, we obtain $\mathbf{M}\mathbf{w} = \mathbf{0}$, where $\mathbf{w} = (\mathbf{v}, \beta_1, \ldots, \beta_k)^\top$, and $\mathbf{M}$ is defined in Eq. (8) in [155]. This leads to a least squares system:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} ||\mathbf{M}\mathbf{w}||_2^2 \quad \text{s.t.} \quad ||\mathbf{w}||_2^2 = 1, \tag{3.26}$$

which can be solved by applying SVD to $\mathbf{M}$, and taking the right singular vector associated to the minimum singular value. The constraint $||\mathbf{w}||_2^2 = 1$ avoids a trivial solution. The first 10 entries of $\hat{\mathbf{w}}$ are $\hat{\mathbf{v}}$, which is a vectorized version of the dual ellipsoid $\hat{\mathbf{Q}}^*$ in the global frame. To avoid degenerate quadrics, a variant of the least squares system in (3.26) is proposed in [58], which constrains the center of the ellipse and the reprojection of the center of the 3D ellipsoid to be close, which is used here.

The object pose $\hat{\mathbf{T}}^{-1}$ can be recovered by relating the estimated ellipsoid $\hat{\mathbf{Q}}^*$ in global coordinates to the ellipsoid $\mathbf{Q}^*_\mathbf{u}$ in the canonical coordinate frame predicted by the coarse shape decoder $\mathbf{u} = g_\phi(\mathbf{z})$ using the average class shape $\mathbf{z}$:

$$\hat{\mathbf{Q}}^* = \hat{\mathbf{T}}^{-1}\mathbf{Q}^*_{\mathbf{u}}\hat{\mathbf{T}}^{-\top} = \begin{bmatrix} \hat{s}^2\hat{\mathbf{R}}\mathbf{U}\mathbf{U}^\top\hat{\mathbf{R}}^\top - \hat{\mathbf{t}}\hat{\mathbf{t}}^\top & -\hat{\mathbf{t}} \\ -\hat{\mathbf{t}}^\top & -1 \end{bmatrix}. \tag{3.27}$$

The translation $\hat{\mathbf{t}}$ can be recovered from the last column of $\hat{\mathbf{Q}}^*$. To recover the rotation, note that $\mathbf{A} \triangleq \mathbf{P}\hat{\mathbf{Q}}^*\mathbf{P}^\top + \hat{\mathbf{t}}\hat{\mathbf{t}}^\top = \hat{s}^2\hat{\mathbf{R}}\mathbf{U}\mathbf{U}^\top\hat{\mathbf{R}}^\top$ is a positive semidefinite matrix. Let its eigen-decomposition be $\mathbf{A} = \mathbf{V}\mathbf{Y}\mathbf{V}^\top$, where $\mathbf{Y}$ is a diagonal matrix containing the eigenvalues of $\mathbf{A}$. Since $\mathbf{U}\mathbf{U}^\top$ is diagonal, it follows that $\hat{\mathbf{R}} = \mathbf{V}$, while the scale $\hat{s}$ is obtained as $\hat{s} = \frac{1}{3}\sqrt{\mathrm{tr}(\mathbf{U}^{-1}\mathbf{Y}\mathbf{U}^{-\top})}$. Note that although the SIM(3) pose could also be recovered from the object point cloud, other outlier rejection methods are required [194] when the point cloud is noisy.

3.4.4 Joint Pose and Shape Optimization with an ELLIPSDF Model

Given the initialized instance transformation $\hat{\mathbf{T}}$ and initial shape deformation $\delta\hat{\mathbf{z}} = \mathbf{0}$, we solve the joint pose and shape optimization in (3.19) via gradient descent. Note that the decoder parameters θ, ϕ and the mean category shape code $\mathbf{z}$ are fixed during online inference. Since $\mathbf{T}$ is an element of the SIM(3) manifold, the gradients and gradient steps need to be computed by projecting to the tangent $\mathfrak{sim}(3)$ vector space and retracting back to SIM(3). We introduce local perturbations $\mathbf{T} = \exp\left(\xi_\times\right)\hat{\mathbf{T}}$, $\delta\mathbf{z} = \delta\tilde{\mathbf{z}} + \delta\hat{\mathbf{z}}$ and derive the Jacobians of the error function in (3.17) with respect to ξ and $\delta\tilde{\mathbf{z}}$.

Proposition 1. *The Jacobian of e_θ in* (3.20) *with respect to the transformation perturbation $\xi \in \mathfrak{sim}(3)$ is:*

$$\begin{aligned} \frac{\partial e_\theta}{\partial \xi} &= \frac{\partial \rho(r)}{\partial r}\left(\hat{s}[\mathbf{0}_6, 1]f_\theta(\mathbf{x}, \delta\hat{\mathbf{z}}) + \hat{s}\nabla_{\mathbf{x}} f_\theta(\mathbf{x}, \delta\hat{\mathbf{z}})^\top \mathbf{P}\left[\hat{\mathbf{T}}\underline{\mathbf{x}}\right]^{\odot}\right) \\ \frac{\partial e_\theta}{\partial \delta\tilde{\mathbf{z}}} &= \frac{\partial \rho(r)}{\partial r}\hat{s}\nabla_{\mathbf{z}} f_\theta(\mathbf{x}, \delta\hat{\mathbf{z}}), \end{aligned} \tag{3.28}$$

where $f_\theta(\mathbf{x}, \delta\hat{\mathbf{z}}) = f_\theta(\mathbf{P}\hat{\mathbf{T}}\underline{\mathbf{x}}; \mathbf{z} + \delta\hat{\mathbf{z}})$ is defined in (3.20) *and $\frac{\partial \rho(r)}{\partial r}$ is the derivative of the Huber* *term in* (3.21) *evaluated at* $r = \hat{s} f_\theta(\mathbf{x}, \delta\hat{\mathbf{z}}) - d$*:*

$$\frac{\partial \rho(r)}{\partial r} = \begin{cases} r & |r| \leq \delta, \\ sign(r)\delta & else. \end{cases} \tag{3.29}$$

The operator $\underline{\mathbf{x}}^{\odot}$ *is defined as:*

$$\underline{\mathbf{x}}^{\odot} \triangleq \begin{bmatrix} \mathbf{I}_3 & -\mathbf{x}_\times & \mathbf{x} \\ \mathbf{0}^\top & \mathbf{0}^\top & 0 \end{bmatrix} \in \mathbb{R}^{4\times 7}. \tag{3.30}$$

Proof. Using the chain rule and the product rule:

$$\frac{\partial e_\theta}{\partial \xi} = \frac{\partial e_\theta}{\partial r}\frac{\partial r}{\partial \xi} = \frac{\partial e_\theta}{\partial r}\left(\frac{\partial s}{\partial \xi} f_\theta(\mathbf{x}, \delta\mathbf{z}) + s\frac{\partial f_\theta}{\partial {}_O\mathbf{x}}\frac{\partial {}_O\mathbf{x}}{\partial \xi}\right), \tag{3.31}$$

where ${}_O\mathbf{x} = \mathbf{P}\mathbf{T}\underline{\mathbf{x}}$ is a point in the object frame. We have $\frac{\partial s}{\partial \xi} = e^\sigma[\mathbf{0}_6, 1] = s[\mathbf{0}_6, 1]$. The term $s\frac{\partial f_\theta}{\partial {}_O\mathbf{x}}$ is the gradient of the fine-level SDF decoder with respect to the input $s\nabla_\mathbf{x} f_\theta(\mathbf{x}, \delta\mathbf{z})$, which could be obtained from auto-differentiation. Finally, we have:

$$\begin{aligned} {}_O\mathbf{x} &= \mathbf{P}\mathbf{T}\underline{\mathbf{x}} \approx \mathbf{P}(\mathbf{I} + \xi_\times)\hat{\mathbf{T}}\underline{\mathbf{x}} \\ &= \mathbf{P}\hat{\mathbf{T}}\underline{\mathbf{x}} + \mathbf{P}\xi_\times\hat{\mathbf{T}}\underline{\mathbf{x}} \\ &= \mathbf{P}\hat{\mathbf{T}}\underline{\mathbf{x}} + \underbrace{\mathbf{P}[\hat{\mathbf{T}}\underline{\mathbf{x}}]^{\odot}}_{\frac{\partial {}_O\mathbf{x}}{\partial \xi}}\xi. \end{aligned} \tag{3.32}$$

□

In the second equality in Prop. 1, the term $\frac{\partial \rho(r)}{\partial r}\hat{s}\nabla_\mathbf{z} f_\theta(\mathbf{x}, \delta\hat{\mathbf{z}})$ is the gradient of the fine-level SDF loss with respect to the input $\mathbf{z}$ and can be obtained via auto-differentiation. The Jacobians of the coarse-level SDF error $\frac{\partial e_\phi}{\partial \xi}$, $\frac{\partial e_\phi}{\partial \delta\tilde{\mathbf{z}}}$ can be obtained in a similar way. After obtaining the Jacobians, the pose and latent shape code can be optimized via:

$$\begin{aligned}\mathbf{T}^{i+1} &\triangleq \exp\left(-\eta_1 \frac{\partial e(\mathbf{T}, \delta\mathbf{z}, \theta^*, \phi^*; \{\mathcal{X}_k(\mathbf{p})\})}{\partial \xi}\right)\mathbf{T}^i, \\ \delta\mathbf{z}^{i+1} &\triangleq \delta\mathbf{z}^i - \eta_2 \left(\frac{\partial e(\mathbf{T}, \delta\mathbf{z}, \theta^*, \phi^*; \{\mathcal{X}_k(\mathbf{p})\})}{\partial \delta\mathbf{z}}\right),\end{aligned} \tag{3.33}$$

where η_1, η_2 are step sizes, $\delta\mathbf{z}^0 = \mathbf{0}$, and $\mathbf{T}^0 = \hat{\mathbf{T}}$ is obtained from the initialization. During optimization, we add regularization $e_r(\delta\mathbf{z}) = \|\delta\mathbf{z}\|_2^2$ to restrict the amount of latent code deformation.

3.5 Experiments

We evaluate the performance of CORSAIR and ELLIPSDF on the synthetic ShapeNet [20] dataset and the real-world ScanNet [34] + Scan2CAD [6] dataset. ShapeNet is a large scale repository for 3D CAD models and we focus on the furniture categories. ScanNet is an RGB-D video dataset containing 2.5M views in 1513 indoor scenes annotated with 3D camera poses, surface reconstructions, and semantic segmentations. Scan2CAD dataset aligns ScanNet indoor scans with CAD models from ShapeNet. With this two datasets, we can evaluate how our proposed algorithms work on both the synthetic, delicate CAD model objects and the real, noisy object observations from depth images or reconstructed mesh. Since we focus on category-specific objects, we assume that the category of any given point cloud is known so that we can train models for different categories separately.

3.5.1 CORSAIR Evaluation Metrics

Given a query point cloud $\mathbf{X}$ and its retrieved point cloud $\mathbf{Y}$, we define the ground truth pose of $\mathbf{X}$ with respect to $\mathbf{Y}$ as $(\mathbf{R}^*, \mathbf{p}^*)$ and the estimated pose as $(\hat{\mathbf{R}}, \hat{\mathbf{p}})$. We compute relative rotation error (RRE) and relative translation error (RTE) as pose estimation metrics:

$$\mathbf{RTE}(\hat{\mathbf{p}}, \mathbf{p}^*) = ||\hat{\mathbf{p}} - \mathbf{p}^*||_2, \tag{3.34}$$

$$\mathbf{RRE}(\hat{\mathbf{R}}, \mathbf{R}^*) = \arccos((\mathbf{tr}(\hat{\mathbf{R}}^\top \mathbf{R}^*) - 1)/2). \tag{3.35}$$

Two different metrics are used to evaluate the retrieval performance on ShapeNet: Precision@M and Top-1 Chamfer distance. Precision@M is defined as:

$$\text{Precision@M} = \frac{1}{M} \sum_{\mathbf{Y} \in \mathcal{R}} \mathbb{1}_{\mathbf{Y} \in \mathcal{P}_\mathcal{G}(\mathbf{X})}, \tag{3.36}$$

where $\mathcal{R} \subset \mathcal{Y}$ is the retrieved set of M CAD models and $\mathcal{P}_\mathcal{G}(\mathbf{X})$ is the ground truth positive set for query object $\mathbf{X}$ as defined in 3.2.3. The Top-1 Chamfer distance metric is calculated by measuring the Chamfer distance (3.7) between the top-1 retrieved CAD model and the ground-truth top-1 similar CAD model.

3.5.2 CORSAIR Retrieval & Registration on ShapeNet

We use point clouds sampled from the ShapeNet CAD models provided by [197]. In the category of chair, we use 4612, 592, 1242 point clouds for training, validation and testing. In the category of table, we use 5744, 778, 1557 point clouds for training, validation and testing. The shape retrieval and pose estimation tasks are performed within each category. We first train the local feature extractor with positive and negative matching pairs as mentioned in Eq. (3.2) for 100 epochs. Then we freeze the parameters and train the embedding network with triplets $(\mathbf{Y}, \mathbf{P}, \mathbf{N})$ for another 100 epochs. Random rotation is applied to each of $\mathbf{Y}$, $\mathbf{P}$ and $\mathbf{N}$ before training and evaluating. The random seed is fixed in all the experiments for fair comparisons.

In ShapeNet, pose registration and the retrieval are considered as two separate tasks. To evaluate the pose registration performance, we estimate the transformation between a CAD model $\mathbf{Y}_i$ and a similar object in $\mathbf{Y}_j \in \mathcal{P}_\mathcal{G}(\mathbf{Y}_i)$ and measure the RRE and RTE. We assume that the symmetry of the $\mathbf{Y}_j$ is known and $\mathbf{Y}_i$ shares the same symmetry. In the evaluation of retrieval, we use the metric defined in 3.5.1. We report Precision@M$= 0.1n$, where n is the size of the test set. In the pose registration task, we compare our learned local feature (based on FCGF) with the hand-crafted FPFH feature [158]. FCGF is the same local feature extractor as ours but it

Table 3.1. Quantitative pose registration results on ShapeNet [20]. Every object is aligned with a randomly selected object from its top-10% similar positive set. RANSAC is applied for all the methods for pose registration. The table shows the percentage of test cases below different error thresholds for different feature extraction and matching strategies.

Category	Method	RRE$\leq 5^{\circ}$	RRE$\leq 15^{\circ}$	RRE$\leq 45^{\circ}$	RTE≤ 0.03	RTE≤ 0.05	RTE≤ 0.10
Chair	FPFH [158]	1.4	7.8	18.5	2.0	10.7	25.8
	FCGF [31]	41.0	86.1	96.1	32.2	83.8	96.0
	CORSAIR (Ours)	62.3	92.1	98.1	48.5	90.2	97.7
Table	FPFH [158]	3.8	18.2	34.4	3.0	14.9	32.5
	FCGF [31]	25.4	62.7	78.4	24.9	63.8	80.5
	CORSAIR (Ours)	54.5	74.8	82.7	46.0	77.0	85.0

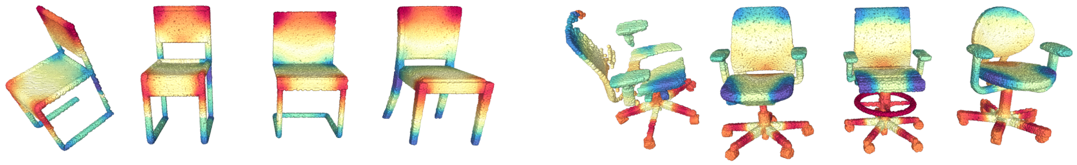

Figure 3.10. Examples of point feature embedded by t-SNE. Column 1: incomplete model pointcloud from single depth scan. Column 2: identical CAD model. Column 3&4: neighbor models.

generates matching pairs without our symmetry-aware method. RANSAC is applied to estimate the pose with given matching pairs in (3.12) with $K = 5$.

First, we want to verify the help of having cross-instance matching with the help of NCC. In Fig. 3.10 we visualize the point cloud features between the partial point cloud and the CAD models by embedding the high dimensional feature vector using t-SNE [186] and coloring the point cloud. The learned features have the ability to generate match across different instances. We compare our model with cross-instance matching with the model trained with only same-instance matching pairs. We visualize some comparing cases on pose estimation in Fig. 3.11. We can see that with cross-instance matching data included for the training, the model can generate more consistent feature vectors across different instance, estimate more inlier matching pairs and also perform better on the pose estimation task.

Then, we want to evaluate the symmetry-aided registration. Fig. 3.12 is a qualitative result showing that our method generates more accurate matching pairs with the aid of symmetry information than the naive nearest-neighbor method. Mismatches caused by symmetry areas are filtered out. The quantitative results are shown in Table 3.1. Our symmetry-aware method outperforms FCGF baseline by 21.3% and 29.1% for chairs and tables, in terms of the ratio of test cases with RRE $\leq 5^{\circ}$. The results show that our symmetry-aware method improves the pose estimation performance by refining matching pairs.

We compare our retrieval module with other prevalent global shape descriptors including 3D ResNet18 [72] and PointNet [143] as well as FCGF without our global embedding network.

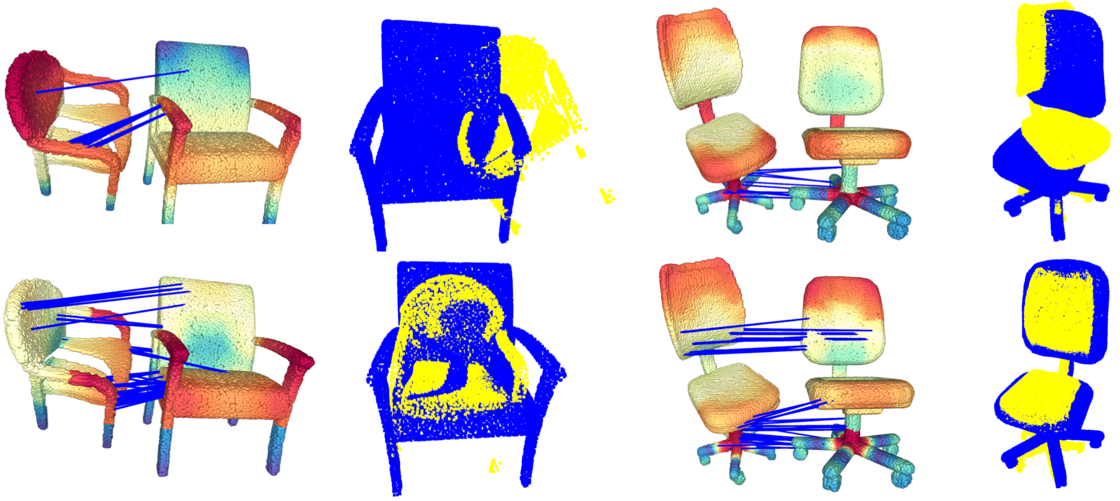

Figure 3.11. Comparing the models trained with (bottom row) and without (top row) cross-instance matching data on ShapeNet [20] synthetic data. Column 1 and 3 show the point feature embedded by t-SNE. The depth scan is on the left and the CAD model is on the right. The blue lines indicate the accurate match among the total 1000 sampled matches. Column 2 and 4 show the final registration results based on 1000 matching pairs. The CAD model is in blue and the depth scan is in yellow. The negative matching pairs are not shown, which cause the pose estimation failure.

Table 3.2. Retrieval quantitative results on ShapeNet [20].

Method	Chair		Table	
	Precision@M	Top-1 CD	Precision@M	Top-1 CD
3D ResNet18 [72]	21.81	0.182	17.49	0.231
PointNet [143]	25.65	0.188	17.76	0.234
FCGF [31]	31.83	0.132	36.19	0.135
CORSAIR (Ours)	51.47	0.115	57.77	0.112

We use the 3D ResNet18 implementation in [30]. Both 3D ResNet18, PointNet and our method generate a 256-D global descriptor for retrieval. The FCGF method directly uses latent vector $\mathbf{Z}(\mathbf{X})$ as the global feature, and the distance is measured by the Chamfer distance (3.7) in 256-D. All the methods are trained with the same loss function as defined in (3.10). Quantitative results are shown in Table 3.2. Our method outperforms the baseline by a large margin in both Precision@M and Top-1 Chamfer distance metrics.

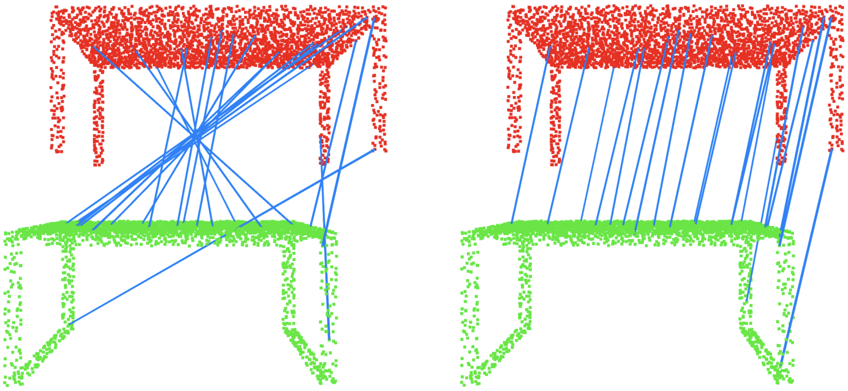

Figure 3.12. Local feature matching between a query point cloud (red) and a retrieved point cloud (green). The blue lines show the pairs between the two point clouds based on nearest neighbor matching of the local features. The left pair of point clouds shows matching without symmetry aid, while the right pair shows matching after detecting the plane of symmetry.

3.5.3 CORSAIR Retrieval & Registration on ScanNet&Scan2CAD

The Scan2CAD dataset [6] provides object-level human-generated annotations on the ScanNet [34] scans.. The annotations includes category label, segmentation, a similar CAD model in ShapeNet, and the corresponding pose. In this dataset, we assume that the segmentation and category are known but the annotated similar CAD model and the pose are unknown. We use the object segmentation labels to segment the object meshes from the scene and sample points on the surfaces to convert them to point clouds. In the chair category, we use 2896, 343, 993 scanned point clouds for training, validation and testing. In the table category, we use 1164, 150, 291 scanned point clouds for training, validation and testing. Since the object distribution in the scenes is not uniform, we split the dataset by scenes instead of objects. Given pretrained parameters from ShapeNet, we train the local feature extractor and the embedding network on the Scan2CAD dataset separately for 100 epochs each. In the training phase, random rotations are applied to both scanned objects and CAD models. In the evaluation phase, we set the CAD models in the canonical pose.

For the pose registration task, we assume the number of symmetry classes for the CAD models are known in advance, and the scanned objects share the same symmetry with retrieved

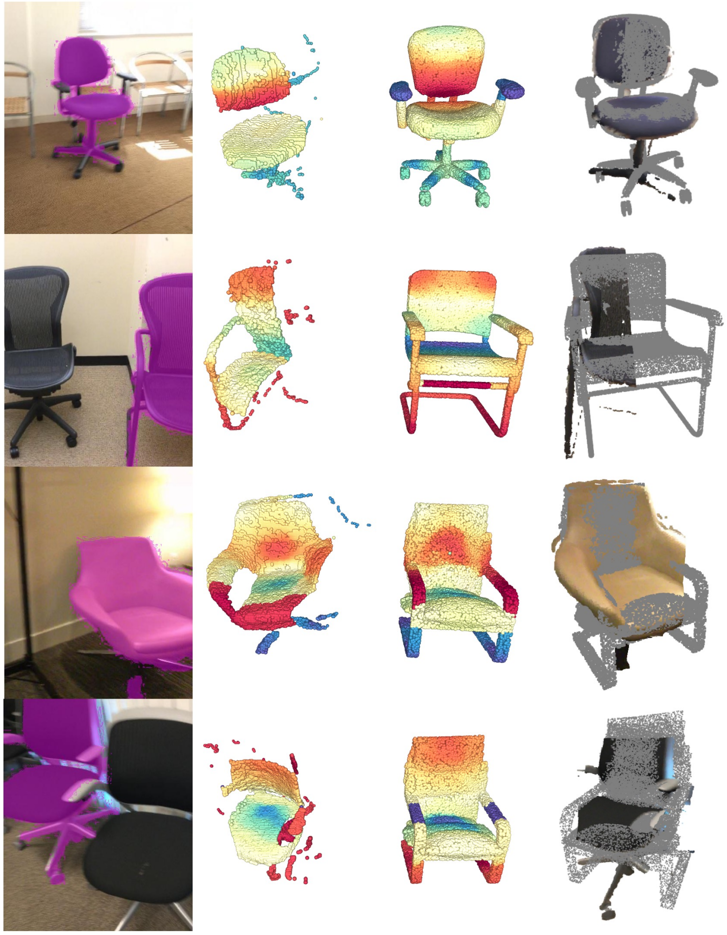

Figure 3.13. Examples on matching the point cloud segmented from a single RGB-D image from ScanNet dataset [34]. Column 1: RGB images and object segmentatin masks in purple. Column 2 & 3: color-coded point cloud features of the observed objects and the CAD models. Column 4: Alignment results with the CAD models painted in grey.

CAD models. For the retrieval task, the database $\mathcal{Y}$ contains CAD models from the Scan2CAD dataset and belong to the same category as the scanned object. The size of the CAD model database is 652 and 830 for the chair and table categories, respectively. Unlike in the ShapeNet experiments, for a scanned object $\mathbf{X}$, we only consider the ground-truth annotated CAD model as positive object $\mathbf{P}$. The negative object $\mathbf{N}$ is randomly sampled from the negative set $\mathcal{N}_{\mathcal{G}}(\mathbf{P})$ with respect to the annotated positive object $\mathbf{P}$, since the similarity between partially scanned objects and a CAD model is not well-defined. The retrieval task is evaluated jointly with the pose estimation task. We report the RRE and RTE as well as the single direction Chamfer distance to

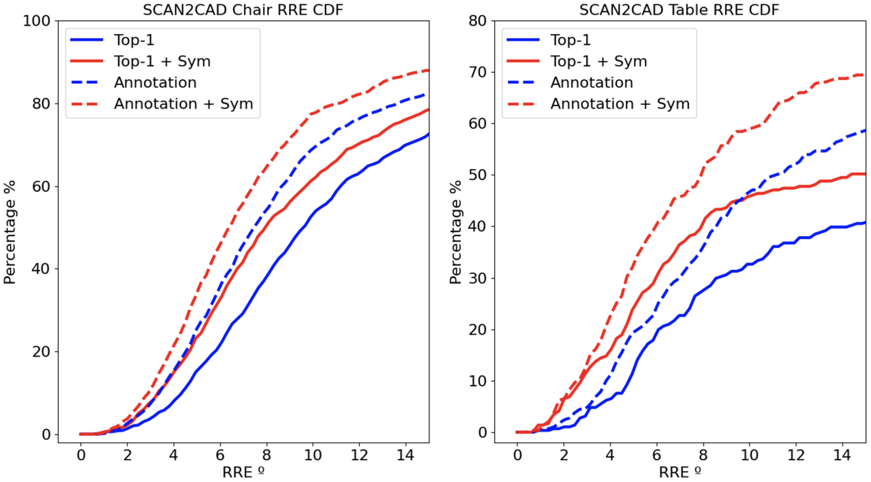

Figure 3.14. Cumulative distribution function of RRE on the Scan2CAD dataset [6]. The solid lines represent RRE when aligning with top-1 retrieved CAD models. The dotted lines represents RRE aligning with annotated CAD models. Blue lines use naive nearest neighbor features + RANSAC for registration, while the red lines use our symmetry-aided nearest neighbor matching + RANSAC.

assess the overall alignment quality.

We first evaluate the pose registration method by aligning scanned objects with the annotated CAD models. We visualize some qualitative result in Fig. 3.13. Here we extract the observed object point cloud from a segmented RGB-D image. The first row shows an example with relatively complete observations. The second example is incomplete due to image truncation. The thrid and the fourth observed objects are sharing the same CAD model for pose estimation. Our model is showing reasonable performance. Table 3.3 and Fig. 3.14 provide more quantitative results. In a real world setting, our pose registration method is still able to estimate accurate poses (RRE $\leq 5°$) for 25.2% of the chairs and 19.2% of the tables. The error for the majority of the test cases is within a reasonable range (RRE $\leq 15°$). With our symmetry-aware method, we can further improve the ratio of accurate estimation by 9.2% and 13.1% for chairs and tables. The symmetry-aware method still generates better pose results in both categories when point clouds are partially observed. Our method works well on approximately complete point clouds, and for severely occluded scans we use the naive nearest neighbors with RANSAC as a back-up

Table 3.3. Ablation evaluation of CORSAIR on Scan2CAD [6]. We compare our top-1 retrieved CAD (Top-1 Retrieval) and the ground-truth CAD annotation (GT annotation). We also compare our symmetry-aware matching method (w/ sym) with naive nearest neighbor (w/o sym). The percentage of test cases lower than different thresholds and the average single-direction Chamfer distance are reported.

Category	CAD model	Registration	RRE $\leq$ 5º	RRE $\leq$ 15º	RRE $\leq$ 45º	RTE$\leq$ 0.05	RTE$\leq$ 0.10	RTE$\leq$ 0.15	$SCD(\times 10^{-2})$
Chair	GT Annotation	w/o sym	25.2	82.2	89.9	21.6	60.4	78.7	5.99
		w/ sym	34.4	87.9	94.0	27.4	68.5	85.2	5.43
	Top-1 Retrieval	w/o sym	15.4	72.5	86.4	8.2	35.7	59.0	7.53
		w/ sym	23.2	78.5	88.9	9.7	40.5	63.5	6.81
Table	GT Annotation	w/o sym	19.2	58.4	72.5	12.7	38.5	58.8	7.20
		w/ sym	32.3	69.4	76.3	26.1	56.7	70.1	5.68
	Top-1 Retrieval	w/o sym	11.7	40.9	52.9	4.8	17.5	28.5	9.06
		w/ sym	24.5	50.2	57.0	10.7	27.8	41.2	7.14

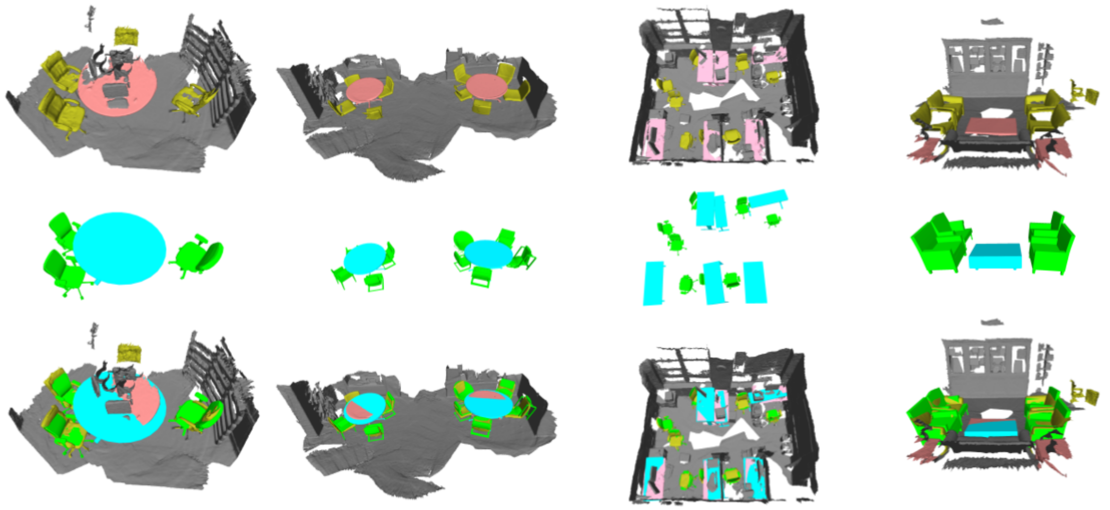

Figure 3.15. Scene-level reconstruction of scenes 0314_00, 0355_00, 0653_00 and 0690_00 from the Scan2CAD dataset [6]. The segmented chairs (yellow) and tables (pink) in the first row are inputs to our model. The second row shows the predicted object map, obtained from aligning retrieved instances to the query point clouds using CORSAIR. The retrieved chairs (green) and tables (blue) are overlaid back into the scene to visualize the reconstruction qualitatively.

to handle asymmetric cases. Then we evaluate both the retrieval and the pose estimation using single direction Chamfer distance in the last column of Table 3.3. Our retrieved CAD models can reach comparable alignment results compared with human-labeled ground-truth CAD models. This indicate that our method is able to retrieve reasonable models for better alignment.

In Fig. 3.15, we visualize the scene-level reconstruction to show qualitative results in real-world scenarios. Our method is able to retrieve CAD models that share the same structure with the scanned objects and align them accurately. Some failed cases are also presented, e.g., the chair on the right of the second scene. Most of the failed cases are caused by severe occlusions. The absence of key structures, like the legs of a table or chair, may lead to multiple solutions. The limitations of raw point cloud measurements make it hard for our method to solve this kind of problems.

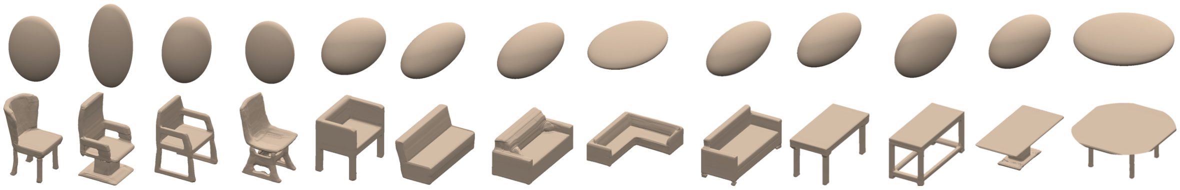

Figure 3.16. Visualization of the object models trained on ShapeNet [20] for chair, sofa and table. Upper row: coarse ellipsoid shapes regressed from g_ϕ and $\mathbf{z}$. Lower row: SDF object model from f_θ and $\mathbf{z}$.

3.5.4 ELLIPSDF Details

The ELLIPSDF decoder model is trained on synthetic CAD models from ShapeNet [20]. Each model's scale is normalized to be inside a unit sphere. We sample points and calculate their SDF values using a uniform distribution in the unit sphere for training the coarse-level shape decoder g_ϕ. Another set of points that are close to the model surface are sampled for training the fine-level shape decoder f_θ. The following setting were used to train the decoder networks and the latent shape code $\mathbf{z}$. We use the Adam optimizer with initial learning rate 5×10^{-4}, 0.5 ratio decay every 300/700 epochs for the coarse and fine level networks separately. The total epoch number is 1500. The latent code dimension is 64, and the network structure follows the model in DualSDF [69].

3.5.5 ELLIPSDF Qualitative Results

We first sample from the trained embedded vector distribution and visualize the decoder reconstruction. Different cagetories of models are trained with the synthetic CAD models from ShapeNet. Fig. 3.16 visualizes the rendering results for some chairs, sofas and tables in the training set. It shows that the scale of the primitive-based representation varies proportionally with the high-resolution representation.

We evaluate ELLIPSDF on the ScanNet dataset [34], which provides 3D scans captured by a RGB-D sensor of indoor scenes with chairs, tables, displays, etc. We segment out objects from scene-level mesh using provided instance labels, and sample points from object meshs to generate point observations. Visualizations of shape optimization for a chair are shown in

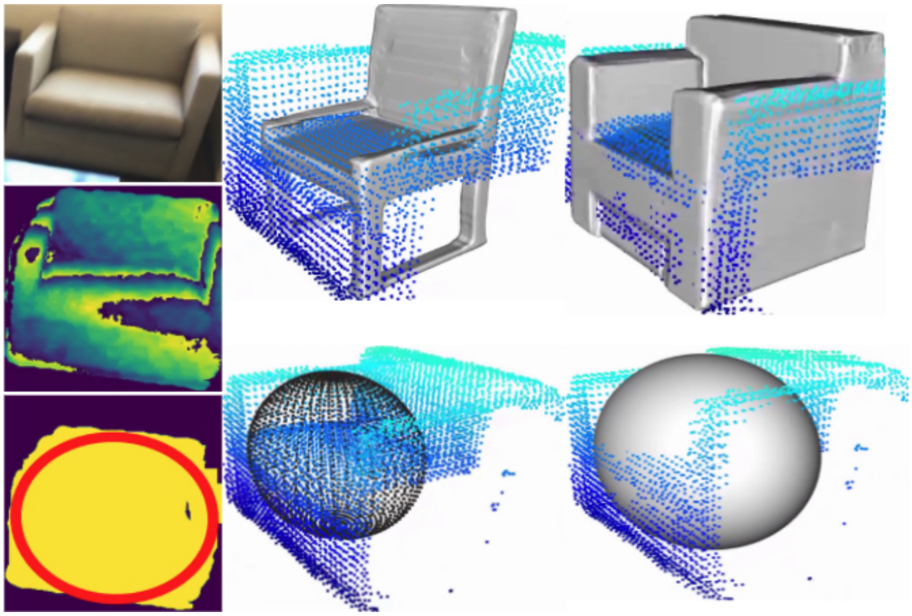

Figure 3.17. Intermediate ELLIPSDF stages. First column: RGB image, depth image, instance segmentation (yellow), fitted ellipse (red) for a chair in ScanNet [34] scene 0461. Second column: mean shape and ellipsoid with initialized pose. Third column: optimized fine-level and coarse-level shapes with optimized pose.

Fig. 3.17. Optimization step improves the scale and shape estimates notably, e.g. by transforming the four-leg mean shape into an armchair. Larger scale qualitative results are shown in Fig. 3.18, demonstrating the effectiveness of joint shape and pose optimization. Optimized poses are closer to the ground-truth, and optimized shapes resemble the objects better than simple primitive shapes such as cuboids or quadrics that lacks fine details. For example, the successful reconstruction of an angle sofa is illustrated in the upper row in Fig. 3.18, which deforms from an initial mean sofa shape that does not have an angle. ELLIPSDF is also able to deal with partial observations as seen in the lower row in Fig. 3.18. Although the observed point clouds of the displays and the chairs are sparse, our approach still reconstructs those objects successfully. Nevertheless, the reconstruction is a square instead of rounded for the table due to a severe occlusion of the observation that only less than half of the table is observed.

Another ScanNet scene with bookshelves and tables are shown in Fig. 3.19, to demonstrate the usefulness of the coarse and fine level residuals. The figure illustrates that the initialized

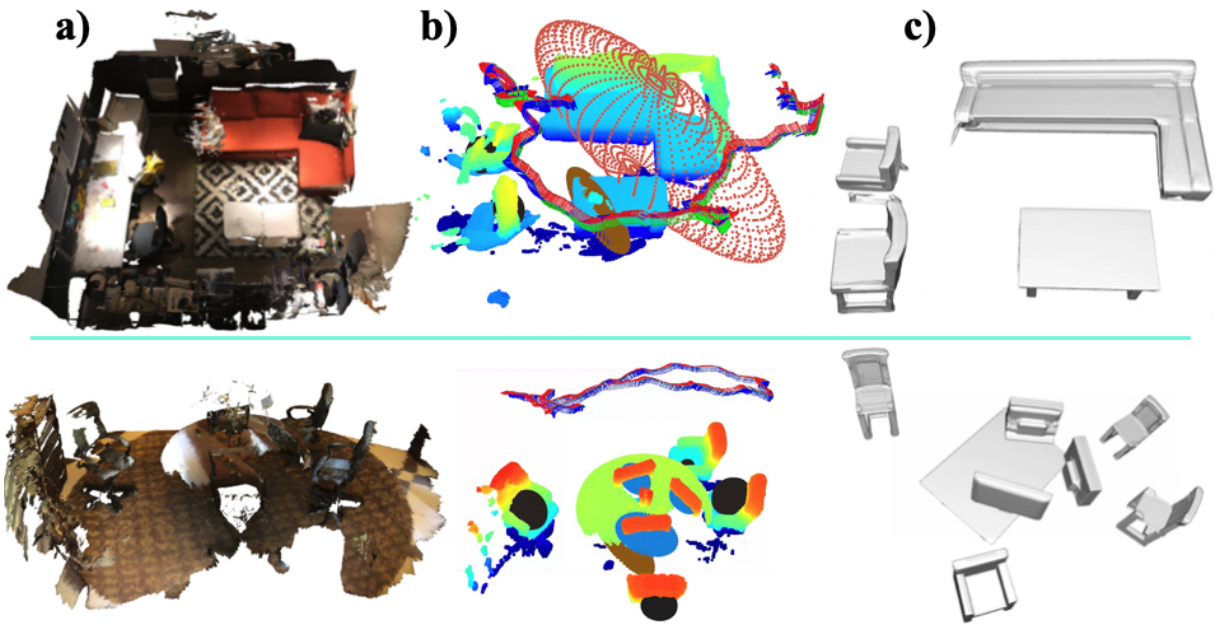

Figure 3.18. Qualitive results. Column a): Ground-truth scene in ScanNet [34] sequence 0518 (upper row) and 0314 (lower row). Column b): The RGB axes are the camera trajectory, point clouds are the ones obtained from RGB-D sensor with added pesudo points, and the ellipsoids (black for chair, red for sofa, blue for monitor, brown for table) are the initialized objects. Column c): Reconstructed meshes using ELLIPSDF, rendered from the optimized latent code and pose.

object pose and shape are different from the actual scene, since the two bookshelves in the center are not parallel and are too small compared to the observation. In contrast, the bookshelves become larger after applying the fine level residual, which is more consistent with the observations. The reconstructions are further improved with both the coarse and fine level residuals, where the bookshelves become parallel. Moreover, the bottom bookshelf and the top right table also become thinner, which agrees more with the observation. This example clearly shows the effectiveness of the proposed bi-level model for joint object pose and shape optimization.

3.5.6 ELLIPSDF Quantitative Results

This section presents quantitative evaluation against other methods regarding both pose and shape estimation accuracy. We also present ablation studies to showcase the improvement of the optimization over initialization-only results, and the bi-level model over a one level model.

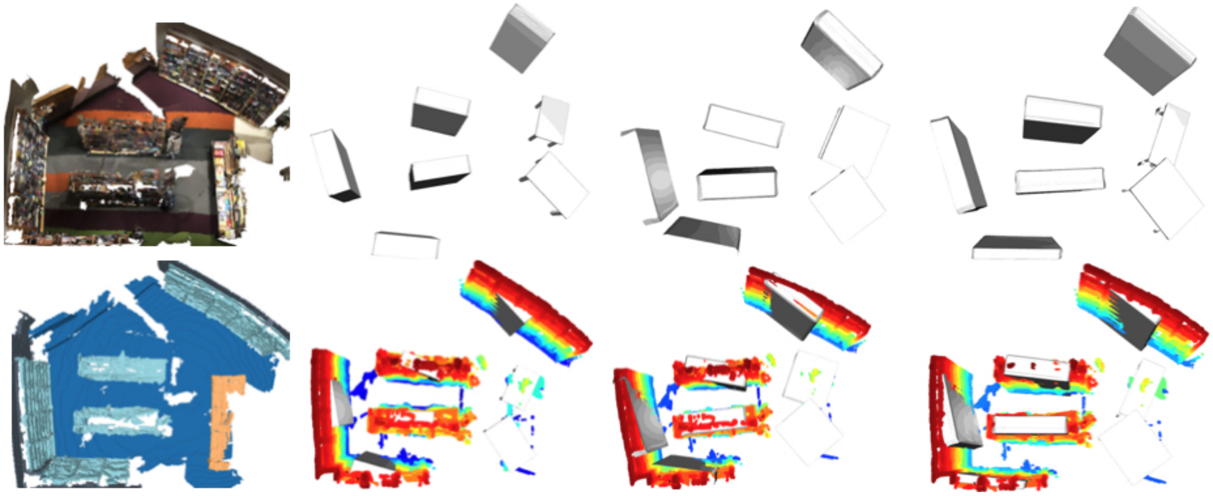

Figure 3.19. Visualization of the original scene and reconstructed objects for ScanNet [34] scene 0208. First row from left to right: original scene, reconstruction using initialized pose and mean categorical object shape, reconstruction using optimized pose and shape with fine level residual only, reconstruction using optimized pose and shape with both coarse and fine level residuals. Second row from left to right: original scene with bookshelves and tables highlighted in light blue and beige, the rest are reconstructions overlaid with object point clouds and added pseudo points.

Table 3.4. Quantitative results for pose estimation on ScanNet [34].

Scan2CAD [6]	Vid2CAD [117]	ELLIPSDF (init)	ELLIPSDF (opt)
31.7	38.3	31.5	**39.6**

Evaluation on Object Pose

We obtain the ground-truth object pose annotations from Scan2CAD [6] and follow the pose evaluation metrics it defines, which decomposes a pose $\mathbf{T} \in \text{SIM}(3)$ into rotation **q**, translation **p** and scale **s**. For an accurate pose estimation, the error thresholds for translation, rotation, and scales are set as 0.2, 20° and 20% respectively with respect to the ground-truth pose. The pose evaluation is presented in Tab. 3.4, in which ELLIPSDF (init) refers to the initialization-only step in Sec. 3.4.4, whereas ELLIPSDF (opt) refers using both the initialization and optimization steps in Sec. 3.4.4. The last two columns in Tab. 3.4 show that adding optimization step using SDF residuals improves the estimation by the initialization-only variant, due to the additional SDF residuals to help estimate pose. Moreover, ELLIPSDF (opt) outperforms

Table 3.5. Quantitative results for shape evlaution on ScanNet [34].

Method # intances	cabinet 132	chair 820	display 209	table 146	avg. 327
ELLIPSDF (fine)	88.4	88.3	90.6	76.2	85.9
ELLIPSDF (coarse+fine)	**91.0**	**90.6**	**96.9**	**77.3**	**89.0**

both Scan2CAD and Vid2CAD, which demonstrates the superiority of ELLIPSDF that employs a primitive ellipsoid shape tailored for pose and scale estimation.

Evaluation on Object Shape

We evaluate ELLIPSDF for shape prediction on ScanNet [34] dataset in Tab. 3.5. Instead of single object evaluation in FroDO [159], we evaluate on multiple objects, which is harder than the single-object-scene due to clustering and partial observations. The large scale evaluation verifies that our method can generalize across different sequences and objects. The object point cloud sampled from the object mesh from [6] is used as the ground truth $\mathcal{S}_{gt}$, and the estimated point cloud $\mathcal{S}_{est}$ is generated from the optimized latent code $\mathbf{z}+\delta\mathbf{z}$. Given the ground-truth point cloud $\mathcal{S}_{gt}$ and ELLIPSDF point cloud $\mathcal{S}_{est}$ for an object, the fitting rate with inlier ratio is

$$\begin{aligned} fit(\mathcal{S}_{est},\mathcal{S}_{gt}) &= \frac{|\mathcal{S}_{close}|}{|\mathcal{S}_{est}|}, \\ \mathcal{S}_{close} &= \{\mathbf{v}\in\mathcal{S}_{est} : d_f(\mathbf{v},\mathcal{S}_{gt}) < \lambda\}, \end{aligned} \tag{3.37}$$

where $\lambda = 0.2(m)$. A distance function $d_f(\cdot,\cdot)$ is utilized to measure the distance between a point $\mathbf{v}$ and a point cloud $\mathcal{S}$, which is the distance from the closest point $\mathbf{u}\in\mathcal{S}$ to the point $\mathbf{v}$. In CAD-Deform [86], the distance function is set to be L1 distance, while we use L2 distance.

We run ELLIPSDF (fine) and ELLIPSDF (coarse+fine) on 150 validation sequences on ScanNet [34], where ELLIPSDF (fine) means only the fine level SDF residual is used by setting $\gamma = 0$ in (3.17), and ELLIPSDF (coarse+fine) means the bi-level SDF residuals are used. For each optimized object, we calculate the fitting rate and then average across all instances. In Tab. 3.5, we show the number of instances and average fitting rates for 4 object classes. ELLIPSDF

Table 3.6. Comparison of 3D detection results on ScanNet [34].

mAP @ IoU=0.5	Chair	Table	Display
FroDO [159]	0.32	0.06	0.04
MOLTR [104]	0.39	0.06	0.10
ELLIPSDF (fine)	0.42	0.26	0.25
ELLIPSDF (coarse+fine)	**0.43**	**0.27**	**0.31**

(coarse+fine) achieves better results than ELLIPSDF (fine) across all classes, demonstrating an average 3% boost of fitting rate with the assistance of coarse model, reaching nearly 90% accuracy. The results indicate the effectiveness of the coarse level error function for improving the scale estimation.

Evaluation on 3D IoU

For a quantitative evaluation on pose estimation, our approach is compared with FroDO [159] and MOLTR [104] on ScanNet. The ground-truth object poses and shapes are from Scan2CAD, whereas the estimated 3D bounding box is generated from the estimated point cloud. The evaluation metric is same as [104], i.e. mean Average Precision (mAP), and the IoU threshold is 0.5. The results are shown in Tab. 3.6. First, we compare the bi-level model against the one-level model. From the last two rows in Tab. 3.6, ELLIPSDF (coarse+fine) is superior than ELLIPSDF (fine) in terms of 3D IoU, and thus demonstrates that the bi-level model is beneficial by providing additional cues to constrain the pose and shape. The improvement is more significant for smaller objects, e.g. the displays. This may be explained by the fact that the initialization error is relatively larger for smaller objects, and thus requires a coarse shape residual to confine its pose. Moreover, ELLIPSDF outperforms both FroDO and MOLTR by a large margin for two probably reasons. Firstly, 3D point clouds are used in the observation for ELLIPSDF, while the other two only rely on 2D observations. Secondly, ELLIPSDF computes coarse level SDF residuals using a primitive shape to aid the estimation of pose and shape scale, whereas the other methods use SDF residuals computed from fine shape details.

3.6 Summary

This chapter introduces CORSAIR, an approach for category-level retrieval and registration. CORSAIR is a fully convolutional model for point-cloud processing which jointly generates local point-wise geometric features and a global rotation-invariant shape feature. The global feature allows retrieval of similar object instances from the same category, while the local features, aided by symmetry class labeling, provide matching pairs for pose registration between the retrieved and query objects. The symmetry-aware method proposed in CORSAIR refines the matching pair based on the naive nearest neighbor method and leads to considerable improvement on pose registration. This chapter also introduces ELLIPSDF, a shared latent representation for a category-level bi-level object model to achieve joint pose and shape optimization. Two shape decoders are used to decode the compact latent shape vector to the coarse level ellipsoid representation as well as the fine level SDF representation. Both the pose and the shape of the object get optimized to align with the 3D observations. Enhanced with such technique, the robot can build a meaningful object-level map with detailed object semantic labels and pose and shape information, for all kinds of challenging tasks like semantic navigation or object manipulation.

Acknowledgements

Chapter 3, in part, is a reprint of the material as it appears in Q. Feng and N. Atanasov, ”Fully Convolutional Geometric Features for Category-level Object Alignment,” IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 8492-8498, 2020, and in T. Zhao, Q. Feng, S. Jadhav and N. Atanasov, ”CORSAIR: Convolutional Object Retrieval and Symmetry-AIded Registration,” IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 47-54, 2021, and in M. Shan, Q. Feng, Y. -Y. Jau and N. Atanasov, ”ELLIPSDF: Joint Object Pose and Shape Optimization with a Bi-level Ellipsoid and Signed Distance Function Description,” IEEE/CVF International Conference on Computer Vision (ICCV), pp. 5926-5935, 2021. The dissertation author was the primary author of these papers.

Chapter 4

Object Mapping from 2D Images

In Chapter 3, we discussed the object-level mapping from the 3D measurements. Although 3D sensors are getting more and more popular, a sensor suite of monocular or stereo cameras with inertial measurement unit (IMU) is still the more affordable option for many robots and mobile devices. A visual-inertial sensor suite can provide an accurate odometry at a high frequency as the sensors compensate each other. However, less attention has been dedicated to object-level mapping and 3D reconstruction with semantic information. In this chapter, we work on the object pose and shape estimation for object-level mapping using the 2D sequential images, and combining this with the localization task. Again we assume the semantic category label of the object is known.

The foundations of robotics perception lie in the twin technologies of inferring geometry (e.g., occupancy mapping) and semantic content (e.g., object and scene recognition). Visual-inertial odometry (VIO) [121, 103, 52, 144, 60] and Simultaneous Localization And Mapping (SLAM) [37, 122, 18, 123, 19] are approaches capable of tracking the pose of a robotic system while simultaneously reconstructing a sparse [42, 19] or dense [127, 126, 193, 35] geometric representation of the environment. Current VIO and SLAM techniques achieve impressive performance, yet most rely on low-level geometric features such as points [111, 156] and planes [90, 77] and are unable to extract semantic content. Computer vision techniques based on deep learning recently emerge as a potentially revolutionary way for context comprehension.

A major research challenge today is to exploit information provided by deep learning, such as category-specific object keypoints [128, 139], semantic edges [22, 202], and segmentation masks [71], in VIO and SLAM algorithms to build rich models of the shape, structure, and function of objects and other semantic information. Researchers utilize both spatial and semantic information include [56, 102, 32, 187, 141] but the spatial and semantic states are estimated independently and merged later. Works that consider joint metric-semantic mapping utilize sparse object-level [161, 68, 14, 4, 199, 129] or dense scene-level [100, 66, 152, 213, 3] representations.

In an early approach [32], objects are inserted in the map based on matching of feature descriptors to the models in a database, constructed offline using structure from motion. The camera trajectory provides multi-view information for object pose estimation but the object detections are not used as constraints to optimize the camera trajectory. Recent works often consider the optimization of object and camera poses simultaneously. SLAM++ [161] optimizes the camera and object poses jointly using a factor graph and moreover reconstructs dense surface mesh models of pre-defined object categories. A limitation of this work is that the estimated object shapes are pre-defined and rigid instead of being optimized to match the specific instances detected online. The popularity of joint optimization of camera and object poses keeps increasing with the advent of robust 2-D object detectors based on structured models [46] and deep neural networks [72, 71]. The stacked hourglass model [128] is used by several works [139, 4] to extract mid-level object parts and, in turn, perform factor graph inference to recover the global positions and orientations of objects detected from a monocular camera. In [45], a deep network object detector is used to generate object hypotheses, which are subsequently validated using geometric and semantic cues and optimized via nonlinear filtering. Some of these approaches [14, 4] use inertial measurements and probabilistic data association among detections and objects as additional constraints in the optimization. While most approaches focus on static objects, [105] uses a stereo camera to track ego-motion and dynamic 3-D objects in urban driving scenarios. The authors use bundle adjustment to fuse 3-D object measurements obtained from detection and viewpoint classification. However, the object shape remain fixed.

Methods that optimize object shape online have employed cuboids [7, 199], spheres [53, 134], or ellipsoids [40, 77, 155, 129, 133, 194] as shape models. SSFM [7] uses the tightest bounding cube enclosing an object to parameterize the object location and pose. The object state is optimized via maximum likelihood estimation using bounding box and object pose measurements obtained from a 3D object detector [163]. CubeSLAM [199] generates and refines 3D cuboid proposals using multi-view bundle adjustment without relying on prior models. QuadricSLAM [129] uses ellipsoids and defines a bounding-box detection model based on the camera-frame conic projection of an ellipsoid. Structural constraints based on supporting and tangent planes, commonly observed under in a Manhattan world, have also been introduced [78]. Using generic symmetric shapes, however, makes the orientation of object instances potentially irrecoverable. For instance, the front and back of an object modeled as an ellipsoid become indistinguishable. Recent work has considered deformable mesh models [86] and neural network latent space representations [176] to optimize object shape.

This chapter is based on the papers [50, 166]. In this chapter, we focus on the object mapping from 2D images as well as the coupling between the object mapping and the camera localization. There are two forms of object representations we consider here. One is the 3D triangle mesh. We generate a category-level object mesh and optimize the object pose and shape based on the 2D semantic segmentation and keypoints. The camera pose can also be optimized. The other is the 3D ellipsoid. We triangulate the object pose and scale from the multiple-frame semantic bounding boxes and keypoints detections, and update the camera poses under the Multi-State Constraint Kalman Filter (MSCKF) framework [121]. In summary, the **contributions** of this chapter are summarized as follows.

- We develop an instance-specific object shape model based on a triangular mesh and differentiable functions that measure the discrepancy in the image plane between projections of the model and detected semantic information.
- We define residuals relating ellipsoid object states and IMU-camera states to object seman-

tic bounding box and keypoint detections, and explicitly derive their Jacobians.

- We develop an extension of the multi-state constraint Kalman filter (MSCKF) [121] to enable online tightly coupled estimation of object and IMU-camera states using semantic residuals. Our algorithm is suitable for real-time incremental odometry and object mapping.

4.1 Problem: Localization and Mapping with Mesh Objects

We consider the problem of detecting, localizing, and estimating the shape of object instances present in the scene, and estimating the pose of a camera over time. The states we are interested in estimating are the camera poses $\mathcal{C} \triangleq \{\mathbf{c}_t\}_{t=1}^{T}$ with $\mathbf{c}_t \in SE(3)$ and the object shapes and poses $\mathcal{O} \triangleq \{\mathbf{o}_n\}_{n=1}^{N}$. More precisely, a camera pose $\mathbf{c} := (\mathbf{R}_c, \mathbf{p}_c)$ is determined by its orientation $\mathbf{R}_c \in SO(3)$ and position $\mathbf{p}_c \in \mathbb{R}^3$, while an object state $\mathbf{o} = (\mu, \mathbf{R}_o, \mathbf{p}_o)$ consists of a pose $\mathbf{R}_o \in SO(3)$, $\mathbf{p}_o \in \mathbb{R}^3$ and shape μ, specified as a 3-D triangular mesh $\mu = (\mathbf{V}, \mathbf{F})$ in the object canonical frame with vertices $\mathbf{V} \in \mathbb{R}^{3\times|\mathbf{V}|}$ and faces $\mathbf{F} \in \mathbb{R}^{3\times|\mathbf{F}|}$. Here we define $|\cdot|$ as the number of columns of a matrix. Each column of $\mathbf{F}$ contains the indices of 3 vertices that form a triangular face. A subset of the mesh vertices are designated as keypoints – distinguishing locations on an object's surface (e.g., car door, windshield, or tires) that may be detected using a camera sensor. We define a keypoint association matrix $\mathbf{A} \in \mathbb{R}^{|\mathbf{V}|\times|\mathbf{K}|}$ that generates $|\mathbf{K}|$ keypoints $\mathbf{K} = \mathbf{V}\mathbf{A}$ from all mesh vertices.

Suppose that a sequence $\mathcal{I} \triangleq \{\mathbf{I}_t\}_{t=1}^{T}$ of T images $\mathbf{I}_t \in \mathbb{R}^{W\times H}$, collected from the corresponding camera poses $\{\mathbf{c}_t\}_{t=1}^{T}$, are available for the estimation task. From each image $\mathbf{I}_t$, we extract a set of object observations $\mathcal{Z}_t \triangleq \{z_{lt} = (\xi_{lt}, \mathbf{S}_{lt}, \mathbf{Y}_{lt})\}_{l=1}^{L_t}$, consisting of a detected object's category $\xi_{lt} \in \Xi$, a segmentation masks $\mathbf{S}_{lt} \in \{0,1\}^{W\times H}$ and the pixel coordiantes of detected keypoints $\mathbf{Y}_{lt} \in \mathbb{R}^{2\times|\mathbf{K}_{lt}|}$. We suppose that Ξ is a finite set of pre-defined detectable object categories and that the data association $n = \pi_t(l)$ of observations to object instances is known (we describe an object tracking approach in Sec. 4.2.1 but global data association can also be performed [125, 91, 14]). See Fig. 4.1 for example object observations.

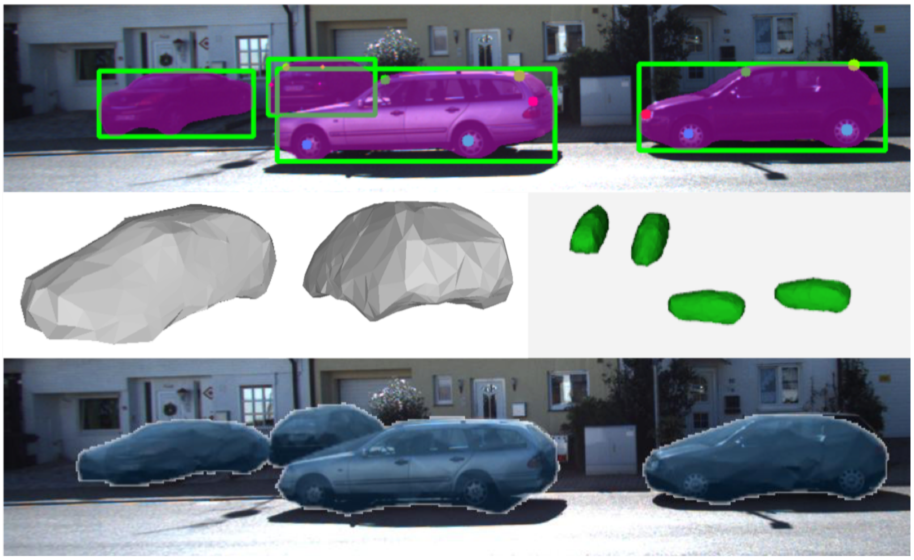

Figure 4.1. Our objective is to build detailed environment maps incorporating object poses and shapes. The figure from KITTI [59] in the top row shows the kind of information that our method relies on: bounding boxes (green), segmentation masks (magenta) and semantic keypoints (multiple colors). The middle row includes the reconstructed mesh models and 3D configuration. The last row shows the projection result.

For a given estimate of the camera pose $\hat{\mathbf{c}}_t$ and the object state $\hat{\mathbf{o}}_n$, we can predict expected semantic mask $\hat{\mathbf{S}}_{lt}$ and semantic keypoint observations $\hat{\mathbf{Y}}_{lt}$ using a perspective projection model:

$$\begin{aligned} \hat{\mathbf{S}}_{lt} &= \mathcal{R}_{\text{mask}}(\hat{\mathbf{c}}_t, \hat{\mathbf{o}}_n) \\ \hat{\mathbf{Y}}_{lt} &= \mathcal{R}_{\text{kps}}(\hat{\mathbf{c}}_t, \hat{\mathbf{o}}_n, A_n) \end{aligned} \tag{4.1}$$

where the mask and keypoint projection functions $\mathcal{R}_{\text{mask}}$, $\mathcal{R}_{\text{kps}}$ will be defined precisely in Sec. 4.2.2. The camera and object estimates can be optimized by reducing the error between the predicted $\hat{\mathcal{Z}}_{1:T}$ and the actual $\mathcal{Z}_{1:T}$ observations. We define loss functions $\mathcal{L}_{\text{mask}}$, measuring discrepancy between semantic masks, and $\mathcal{L}_{\text{kps}}$, measuring discrepancy among semantic keypoints. The loss functions details are in Sec. 4.2.3.

Problem. *Given object observations $\mathcal{Z}_{1:T}$, determine the camera poses $\mathcal{C}$ and object states $\mathcal{O}$*

that minimize the mask and keypoint losses:

$$\begin{aligned}\min_{\mathcal{C},\mathcal{O}} \sum_{t=1}^{T}\sum_{l=1}^{L_t} \big(& w_{mask}\mathcal{L}_{mask}(\mathbf{S}_{lt}, \mathcal{R}_{mask}(\mathbf{c}_t, \mathbf{o}_{\pi_t(l)})) \\ & + w_{kps}\mathcal{L}_{kps}\left(\mathbf{Y}_{lt}, \mathcal{R}_{kps}(\mathbf{c}_t, \mathbf{o}_{\pi_t(l)}, \mathbf{A}_{\pi_t(l)})\right)\big)\end{aligned} \tag{4.2}$$

where w_{mask}, w_{kps} are scalar weight parameters specifying the relative importance of the mask and keypoint loss functions.

4.2 Estimating Mesh Object and Camera Pose from 2D Semantic Segmentation and Keypoint

We begin by describing how the object observations $\mathcal{Z}_t$ are obtained. Next, we provide a rigorous definition of the perspective projection models in (4.1), which, in turn, define the loss functions in (4.8) precisely. Finally, in order to perform the optimization in (4.2), we derive the gradients of $\mathcal{L}_{\text{mask}}$ and $\mathcal{L}_{\text{kps}}$ with respect to $\mathbf{c}$ and $\mathbf{o}$.

4.2.1 Semantic Perception

We extract both category-level (object category ξ_{lt} and keypoints $\mathbf{Y}_{lt}$) and instance-level (segmentation masks $\mathbf{S}_{lt}$) semantic information from the camera images. For each frame, we first use pre-trained Mask R-CNN model [71] implemented in [195] to get object detection results represented with bounding boxes and instance segmentations inside the boxes. Each object is assigned to one of the class labels in Ξ. Then we extract semantic keypoints $\mathbf{Y}_{lt}$ within the bounding box of each detected object using the pre-trained stacked hourglass model of [211], which is widely used for human-joint and object-keypoint detector. The l-th detection result at time t contains the object category $\xi_{lt} \in \Xi$, keypoints $\mathbf{Y}_{lt} \in \mathbb{R}^{2\times|\mathbf{K}_{lt}|}$, mask $\mathbf{S}_{lt} \in \{0,1\}^{W\times H}$, bounding box $\beta_{lt} \in \mathbb{R}^4$ (2-D center location, width, and height), object detection confidence $u_{lt} \in \mathbb{R}$ and keypoint detection confidences $q_{lt} \in \mathbb{R}^{|\mathbf{K}_{lt}|}$ as shown in Fig. 4.1.

We develop an object tracking approach in order to associate the semantic observations obtained over time with the object instance that generated them. We extend the KLT-based ORB-feature-tracking method of [178] to track semantic features $\mathbf{Y}_{lt}$ by accounting for their individual labels (e.g., car wheel, car door) and share object category ξ_{lt}. In detail, let z_{lt} be a semantic observation from a newly detected object at time t. The objective is to determine if z_{lt} matches any of the semantic observations $z_{m,t+1} \in \mathcal{Z}_{t+1}$ at time $t+1$ given that both have the same category label, i.e., $\xi_{lt} = \xi_{m,t+1}$. We apply the KLT optical flow algorithm [112] to estimate the locations $\mathbf{Y}_{l,t+1}$ of the semantic features $\mathbf{Y}_{lt}$ in the image plane at time $t+1$. We use the segmentation mask $\mathbf{S}_{m,t+1}$ of the m-th semantic observation to determine if $\mathbf{Y}_{l,t+1}$ are inliers (i.e., if the segmentation mask $\mathbf{S}_{m,t+1}$ is 1 at image location $\mathbf{Y}_{l,t+1}$) with respect to observation m. Let $in(\mathbf{Y}_{l,t+1}, \mathbf{S}_{m,t+1}) \in \{0,1\}^{|\mathbf{K}_{lt}|}$ return a binary vector indicating whether each keypoint is an inlier or not. We repeat the process in reverse to determine if the backpropagated keypoints $\mathbf{Y}_{m,t}$ of observation m are inliers with respect to observation l. Eventually, we compute a matching score based on the inliers and their detection confidences:

$$M_{lm} = \sum_{k=1}^{\mathbf{K}_{lt}} in(\mathbf{Y}_{l,t+1}^{(k)}, \mathbf{S}_{m,t+1}^{(k)}) \cdot in(\mathbf{Y}_{m,t}^{(k)}, \mathbf{S}_{l,t}^{(k)}) \cdot q_{lt}^{(k)} \tag{4.3}$$

where $q_{lt}^{(k)}$ is the k-th element of keypoint detection confidences q_{lt}. Finally, we match observation l to the observation at time $t+1$ that maximizes the score, i.e., $m^* = \arg\max_k M_{lm}$. If the object bounding boxes β_{lt} and $\beta_{m^*,t+1}$ have compatible width and height, we declare that object l has been successfully tracked to time $t+1$. Otherwise, we declare that object track l has been lost.

4.2.2 Mesh Renderer as an Observation Model

Next, we develop the observation models $\mathcal{R}_{\text{mask}}$ and $\mathcal{R}_{\text{kps}}$ in Eq. (4.1) that specify how a semantic observations $z = (\xi, \mathbf{S}, \mathbf{Y})$ is generated by a camera with pose $(\mathbf{R}_c, \mathbf{p}_c) \in SE(3)$ observing an object of class $\xi \in \Xi$ with pose $(\mathbf{R}_o, \mathbf{p}_o) \in SE(3)$ and mesh shape $\mu = (\mathbf{V}, \mathbf{F})$ with keypoint association matrix $\mathbf{A}$. Let K be the intrinsic matrix of the camera. Let $\mathbf{x} := \mathbf{V}\mathbf{A}\mathbf{e}_i \in \mathbb{R}^3$ be the coordinates of the i-th object keypoint in the object frame, where $\mathbf{e}_i$ is a standard basis vector. The projection of $\mathbf{x}$ onto the image frame can be determined by first projecting it from the object frame to the camera frame using $(\mathbf{R}_o, \mathbf{p}_o)$ and $(\mathbf{R}_c, \mathbf{p}_c)$ and then the perspective projection $\pi : \mathbb{R}^3 \rightarrow \mathbb{R}^3$, and the linear transformation K. In detail, this sequence of transformations leads to the pixel coordinates of $\mathbf{x}$ projection as follows:

$$\mathbf{y}^{(i)} = K\pi(\mathbf{R}_c^T(\mathbf{R}_o\mathbf{x} + \mathbf{p}_o - \mathbf{p}_c)) \in \mathbb{R}^2 \tag{4.4}$$

where the standard perspective projection function is:

$$\pi(x) = \begin{bmatrix} x_1/x_3 & x_2/x_3 & x_3/x_3 \end{bmatrix}^T \tag{4.5}$$

Applying the same transformation to all object keypoints $\mathbf{VA}$ simultaneously leads to the keypoint projection model:

$$\begin{aligned} \mathcal{R}_{\text{cam}}(\mathbf{c}, \mathbf{o}, \mathbf{A}) &:= \mathbf{R}_c^T(\mathbf{R}_o\mathbf{VA} + (\mathbf{p}_o - \mathbf{p}_c)\mathbf{1}^T) \\ \mathcal{R}_{\text{kps}}(\mathbf{c}, \mathbf{o}, \mathbf{A}) &:= K\pi\mathcal{R}_{\text{cam}}(\mathbf{c}, \mathbf{o}, \mathbf{A}) \end{aligned} \tag{4.6}$$

where $\mathbf{1}$ is a vector whose elements are all equal to 1.

To define $\mathcal{R}_{\text{mask}}$, we need an extra rasterization step, which projects the object faces $\mathbf{F}$ to the image frame. A rasterization function, $Raster(\cdot)$, can be defined using the standard method in [84], which assumes that if multiple faces are present, only the frontmost one is drawn at each pixel. Kato et al. [93] also show how to obtain an approximate gradient for the rasterization function. Relying on [84] and [93] for $Raster(\cdot)$, we can define the mask projection model:

$$\mathcal{R}_{\text{mask}}(\mathbf{c}, \mathbf{o}) := Raster(\mathcal{R}_{\text{cam}}(\mathbf{c}, \mathbf{o}, \mathbf{I}), \mathbf{F}) \tag{4.7}$$

where $\mathbf{I}$ is the identity matrix.

4.2.3 Loss Function and Gradient

The loss functions $\mathcal{L}_{\text{mask}}$ and $\mathcal{L}_{\text{kps}}$ are defined as follows:

$$\begin{aligned}
\mathcal{L}_{\text{mask}}(\mathbf{S}, \hat{\mathbf{S}}) &= -\frac{\|\mathbf{S} \odot \hat{\mathbf{S}}\|_1}{\|\mathbf{S} + \hat{\mathbf{S}} - \mathbf{S} \odot \hat{\mathbf{S}}\|_1} \\
\mathcal{L}_{\text{kps}}(\mathbf{Y}, \hat{\mathbf{Y}}) &= \|\mathbf{Y} - \hat{\mathbf{Y}} \cdot vis(\hat{\mathbf{Y}})\|_F^2
\end{aligned} \tag{4.8}$$

where $\odot$ is an element-wise product and $vis(\hat{\mathbf{Y}}_{lt}) \in \{0,1\}^{|\mathbf{K}_n| \times |\mathbf{K}_{lt}|}$ is a binary selection matrix that discards unobservable object keypoints. Now that the projection models (4.1) and hence the loss functions (4.8) have been well defined, the final step needed to perform the optimization in (4.2) is to derive their gradients. We assume that the connectivity $\mathbf{F}$ of the object mesh is fixed and the mesh is deformed only by changing the locations of the vertices $\mathbf{V}$. We use the results of [93] for the gradient $\nabla_V Raster(\mathbf{V}, \mathbf{F})$. Since $\mathcal{R}_{\text{mask}}$ is a function of $\mathcal{R}_{\text{cam}}$ according to (4.7), we only need to derive the following:

$$\begin{aligned}
&\nabla_{\hat{\mathbf{S}}} \mathcal{L}_{\text{mask}}(\mathbf{S}, \hat{\mathbf{S}}), \quad \nabla_{\hat{\mathbf{Y}}} \mathcal{L}_{\text{kps}}(\mathbf{Y}, \hat{\mathbf{Y}}), \\
&\nabla_{\mathbf{c}} \mathcal{R}_{\text{kps}}(\mathbf{c}, \mathbf{o}, \mathbf{A}), \quad \nabla_{\mathbf{o}} \mathcal{R}_{\text{kps}}(\mathbf{c}, \mathbf{o}, \mathbf{A}).
\end{aligned} \tag{4.9}$$

Our results are summarized in the following propositions.

Proposition 2. *The gradients of the loss functions* $\mathcal{L}_{mask}(\mathbf{S}, \hat{\mathbf{S}})$ *and* $\mathcal{L}_{kps}(\mathbf{Y}, \hat{\mathbf{Y}})$ *in* (4.8) *with respect to the estimated mask* $\hat{\mathbf{S}} \in \{0,1\}^{W \times H}$ *and keypoint pixel coordinates* $\hat{\mathbf{Y}} \in \mathbb{R}^{2 \times K}$ *are:*

$$\nabla_{\hat{\mathbf{Y}}} \mathcal{L}_{kps}(\mathbf{Y}, \hat{\mathbf{Y}}) = 2\,(\hat{\mathbf{Y}} \cdot vis(\hat{\mathbf{Y}}) - \mathbf{Y})\, vis(\hat{\mathbf{Y}})^T \tag{4.10}$$

$$\nabla_{\hat{\mathbf{S}}} \mathcal{L}_{mask}(\mathbf{S}, \hat{\mathbf{S}}) = -\frac{1}{u(\mathbf{S}, \hat{\mathbf{S}})} \cdot \mathbf{S} + \frac{i(\mathbf{S}, \hat{\mathbf{S}})}{u^2(\mathbf{S}, \hat{\mathbf{S}})} \cdot (\mathbf{1}\mathbf{1}^T - \mathbf{S})$$

where $i(\mathbf{S}, \hat{\mathbf{S}}) := \|\mathbf{S} \odot \hat{\mathbf{S}}\|_1$ *and* $u(\mathbf{S}, \hat{\mathbf{S}}) := \|\mathbf{S} + \hat{\mathbf{S}} - \mathbf{S} \odot \hat{\mathbf{S}}\|_1$.

Proposition 3. *Let* $\mathbf{y}^{(i)} = \mathcal{R}_{kps}(\mathbf{c}, \mathbf{o}, \mathbf{I}) \in \mathbb{R}^2$ *be the pixel coordinates of the i-th vertex* $\mathbf{v}_i := \mathbf{V}\mathbf{e}_i$ *of an object with pose* $(\mathbf{R}_o, \mathbf{p}_o) \in SE(3)$ *obtained by projecting* $\mathbf{v}_i$ *onto the image plane of a camera* *with pose* $(\mathbf{R}_c, \mathbf{p}_c) \in SE(3)$*, calibration matrix* $K \in \mathbb{R}^{2\times 3}$*. Let* $\theta_c, \theta_o \in \mathbb{R}^3$ *be the axis-angle representations of* $\mathbf{R}_c$ *and* $\mathbf{R}_o$*, respectively, so that* $\mathbf{R}_c = \exp(\lfloor \theta_c \times \rfloor)$ *and* $\mathbf{R}_o = \exp(\lfloor \theta_o \times \rfloor)$ *and* $\lfloor \cdot \times \rfloor$ *is the hat map. Then, the derivative of* $y^{(i)}$ *with respect to* $\alpha \in \{\theta_c, p_c, \theta_o, p_o, v_i\}$ *is:*

$$\frac{\partial \mathbf{y}^{(i)}}{\partial \alpha} = K \frac{\partial \pi}{\partial x}(\gamma) \frac{\partial \gamma}{\partial \alpha} \tag{4.11}$$

where:

$$\begin{aligned}
\frac{\partial \pi}{\partial \mathbf{x}}(\mathbf{x}) &= \frac{1}{x_3} \begin{bmatrix} 1 & 0 & -x_1/x_3 \\ 0 & 1 & -x_2/x_3 \\ 0 & 0 & 0 \end{bmatrix} \\
\gamma &= \mathbf{R}_c^T (\mathbf{R}_o \mathbf{v}_i + \mathbf{p}_o - \mathbf{p}_c) \qquad (4.12)\\
\frac{\partial \gamma}{\partial \mathbf{p}_c} &= -\mathbf{R}_c^T \quad \frac{\partial \gamma}{\partial \mathbf{p}_o} = \mathbf{R}_c^T \quad \frac{\partial \gamma}{\partial \mathbf{v}_i} = \mathbf{R}_c^T \mathbf{R}_o \\
\frac{\partial \gamma}{\partial \theta_c} &= \mathbf{R}_c^T \lfloor (\mathbf{R}_o \mathbf{v}_i + \mathbf{p}_o - \mathbf{p}_c) \times \rfloor J_{rSO(3)}(-\theta_c) \\
\frac{\partial \gamma}{\partial \theta_o} &= -\mathbf{R}_c^T \mathbf{R}_o \lfloor \mathbf{v}_i \times \rfloor J_{rSO(3)}(\theta_o)
\end{aligned}$$

and $J_{rSO(3)}(\theta)$ *is the right Jacobian of* $SO(3)$*, which is necessary because the gradient needs to be projected from the tangent space to the* $SO(3)$ *manifold [28, Ch. 10.6], and can be computed in closed form:*

$$J_{rSO(3)}(\theta) = \mathbf{I}_3 - \frac{1 - \cos\|\theta\|}{\|\theta\|^2} \lfloor \theta \times \rfloor + \frac{\|\theta\| - \sin\|\theta\|}{\|\theta\|^3} \lfloor \theta \times \rfloor^2. \tag{4.13}$$

Proof. By definition (4.6), $\mathbf{y}^{(i)} = K\pi(\gamma)$ so most steps follow by the chain rule. We only discuss the relationship between the axis-angle vectors θ_c, θ_o and the orientations $\mathbf{R}_c$, $\mathbf{R}_o$. Any rotation matrix $\mathbf{R} \in SO(3)$ can be associated with a vector $\theta \in \mathbb{R}^3$ specifying it as a rotation about a fixed axis $\frac{\theta}{\|\theta\|_2}$ through an angle $\|\theta\|_2$. The axis-angle representation θ is related to $\mathbf{R}$ through the exponential and logarithm maps:

$$
\begin{aligned}
\mathbf{R} &= \exp(\lfloor\theta\times\rfloor) = I + \left(\frac{\sin\|\theta\|}{\|\theta\|}\right)\lfloor\theta\times\rfloor + \left(\frac{1-\cos\|\theta\|}{\|\theta\|^2}\right)\lfloor\theta\times\rfloor^2 \\
\lfloor\theta\times\rfloor &= \log(\mathbf{R}) = \frac{\|\theta\|}{2\sin\|\theta\|}(\mathbf{R}-\mathbf{R}^T)
\end{aligned}
$$

See [116] and [28] for details. Consider the derivative of γ with respect to θ_o. The right Jacobian of $SO(3)$ satisfies the following for small $\delta\theta$:

$$
\exp(\lfloor(\theta+\delta\theta)\times\rfloor) \approx \exp(\lfloor\theta\times\rfloor)\exp(\lfloor(J_{rSO(3)}(\theta)\delta\theta)\times\rfloor)
$$

Using this and $\mathbf{R}_o = \exp(\lfloor\theta_o\times\rfloor)$, we can compute:

$$
\begin{aligned}
\frac{\partial\gamma}{\partial\theta_o} &= \mathbf{R}_c^T\frac{\partial\exp(\lfloor\theta_o\times\rfloor)v_i}{\partial\theta_o} \\
&= \mathbf{R}_c^T\mathbf{R}_o\frac{\partial}{\partial\delta\theta_o}\lfloor(J_{rSO(3)}(\theta_o)\delta_o\theta)\times\rfloor\mathbf{v}_i \\
&= -\mathbf{R}_c^T\mathbf{R}_o\lfloor\mathbf{v}_i\times\rfloor J_{rSO(3)}(\theta_o)\frac{\partial\delta\theta_o}{\partial\delta\theta_o}
\end{aligned}
\tag{4.14}
$$

The derivative of γ with respect to θ_c can be obtained using the same approach. □

In conclusion, we derived explicit definitions for the observation models $\mathcal{R}_{\text{kps}}$, $\mathcal{R}_{\text{mask}}$, the loss functions $\mathcal{L}_{\text{mask}}$, $\mathcal{L}_{\text{kps}}$, and their gradients directly taking the $SO(3)$ constraints into account via the axis-angle parameterization. As a result, we can treat (4.2) as an unconstrained optimization problem and solve it using gradient descent. The explicit gradient equations in Prop. 3 allow solving an object mapping-only problem by optimizing with respect to $\mathcal{O}$, a camera localization-only problem by optimizing with respect to $\mathcal{C}$, or a simultaneous localization and mapping problem.

4.2.4 Optimization Initialization

We implemented the localization and mapping tasks separately. In the localization task, we initialize the camera pose using inertial odometry obtained from integration of IMU measurements [121]. The camera pose is optimized sequentially between every two images via (4.2), leading to an object-level visual-inertial odometry algorithm.

To initialize the object model in the mapping task, we collect high-quality keypoints (according to q_{lt} defined in Sec. 4.2.1) from multiple frames until an object track is lost. The 3-D positions of these keypoints are estimated by optimizing $\mathcal{L}_{\text{kps}}$ only using the Levenberg-Marquardt algorithm. Using a predefined category-level mesh model(mean model) with known keypoints, we apply the Kabsch algorithm [89] to initialize the object pose (i.e., the transformation from the detected 3-D keypoints to the category-level model keypoints). After initialization, we take two steps to optimize the object states. First, we fix the mesh vertices and optimize the pose based on the combined loss function in (4.2). Next, we fix the object pose, and optimize the mesh vertices using only the mask loss because the keypoint loss affects only few vertices. To improve the deformation optimization and obtain a smooth mesh model, we add regularization using the mean mesh curvature. The curvature is computed using a discretization of the continuous Laplace-Beltrami operator [172]. Constraints from symmetric object categories can be enforced by directly defining the mesh shape model to be symmetric.

4.3 Problem: Localization and Mapping with Ellipsoid Objects

We denote the IMU, camera, object, and global reference frames as $\{I\}$, $\{C\}$, $\{O\}$, $\{G\}$, respectively. The transformation from frame $\{A\}$ to $\{B\}$ is specified by a 4×4 matrix:

$$ {}_{A}^{B}\mathbf{T} \triangleq \begin{bmatrix} {}_{A}^{B}\mathbf{R} & {}_{A}^{B}\mathbf{p} \\ \mathbf{0}^{\top} & 1 \end{bmatrix} \in SE(3) \tag{4.15} $$

where ${}^B_A\mathbf{R} \in SO(3)$ is a rotation matrix and ${}^B_A\mathbf{p} \in \mathbb{R}^3$ is a translation vector. To simplify the notation, we will not explicitly indicate the global frame when specifying transformations. For example, the pose of the IMU frame $\{I\}$ in $\{G\}$ at time t_k is specified by ${}_I\mathbf{T}_k$. ${}^B_A\bar{q}$ is the unit quaternion parameterizing the rotation $\mathbf{R}({}^B_A\bar{q}) = {}^B_A\mathbf{R}$. Let $\underline{\mathbf{x}} = \begin{bmatrix}\mathbf{x}^\top & 1\end{bmatrix}^\top$ be the homogeneous coordinates of a vector $\mathbf{x}$. Notice the last element can also be 0. A *quadric shape* [70] is a set $\left\{\mathbf{x} \mid \underline{\mathbf{x}}^\top \mathbf{Q}\underline{\mathbf{x}} = 0\right\}$, where $\mathbf{Q}$ is a 4×4 symmetric matrix. Consider an axis-aligned ellipsoid centered at $\mathbf{0}$, $\mathcal{E}_\mathbf{u} \triangleq \left\{\mathbf{x} \mid \mathbf{x}^\top \mathbf{U}^{-\top}\mathbf{U}^{-1}\mathbf{x} = 1\right\}$, where $\mathbf{U} \triangleq \operatorname{diag}(\mathbf{u})$ and the elements of the vector $\mathbf{u}$ are the lengths of the semi-axes of $\mathcal{E}_\mathbf{u}$. In homogeneous coordinates, $\mathcal{E}_\mathbf{u}$ is a special case of a quadric shape $\left\{\mathbf{x} \mid \underline{\mathbf{x}}^\top \mathbf{Q}_\mathbf{u}\underline{\mathbf{x}} = 0\right\}$ with $\mathbf{Q}_\mathbf{u} \triangleq \mathbf{diag}(\mathrm{U}^{-2}, -1)$. A quadric shape can also be defined in dual form, as the set of planes $\underline{\boldsymbol{\pi}} = \mathbf{Q}\underline{\mathbf{x}}$ that are tangent to the shape surface at each $\mathbf{x}$. A *dual quadric surface* is defined as $\left\{\pi \mid \underline{\boldsymbol{\pi}}^\top \mathbf{Q}^* \underline{\boldsymbol{\pi}} = 0\right\}$, where $\mathbf{Q}^* = \operatorname{adj}(\mathbf{Q})$. A dual quadric surface $\mathbf{Q}^*$ can be transformed by $\mathbf{T} \in SE(3)$ to another reference frame as $\mathbf{T}\mathbf{Q}^*\mathbf{T}^\top$. Similarly, it can be projected to a lower-dimensional space by a projection matrix $\mathbf{P} \triangleq \begin{bmatrix}\mathbf{I} & \mathbf{0}\end{bmatrix}$ as $\mathbf{P}\mathbf{Q}^*\mathbf{P}^\top$.

Consider a system equipped with an IMU-camera sensor. Let the IMU state at time t_k be ${}_I\mathbf{x}_k \triangleq \left({}^I_G\mathbf{q}_k,\ {}_I\mathbf{v}_k,\ {}_I\mathbf{p}_k,\ \mathbf{b}_{g,k},\ \mathbf{b}_{a,k}\right)$, consisting of unit quaternion representing the rotation from global to IMU frame ${}^I_G\mathbf{q}_k$, velocity ${}_I\mathbf{v}_k \in \mathbb{R}^3$, position ${}_I\mathbf{p}_k \in \mathbb{R}^3$, gyroscope bias $\mathbf{b}_{g,k} \in \mathbb{R}^3$, and accelerometer bias $\mathbf{b}_{a,k} \in \mathbb{R}^3$. The quaternion follows JPL convention as it is used in MSCKF [121]. Assume that the camera is rigidly attached to the IMU with relative transformation ${}^I_C\mathbf{T} \in SE(3)$, known from extrinsic calibration. To facilitate the use of multi-frame camera information, define an augmented state $\mathbf{x}_k \triangleq \left({}_I\mathbf{x}_k,\ {}^I_G\mathbf{q}_{k-1}, {}_I\mathbf{p}_{k-1}\ \ldots,\ {}^I_G\mathbf{q}_{k-W}, {}_I\mathbf{p}_{k-W}\right)$, containing a sliding window of W past IMU poses in addition to the current IMU state ${}_I\mathbf{x}_k$. The system state over time is $\mathcal{X} \triangleq \{\mathbf{x}_k\}_k$.

Suppose that the system evolves in an unknown environment, containing *geometric landmarks* $\mathcal{L} \triangleq \{\ell_m\}_m$ and *objects* $\mathcal{O} \triangleq \{\mathbf{o}_i\}_i$, represented in a global frame $\{G\}$. A geometric landmark is a static point $\ell_m \in \mathbb{R}^3$, detectable via image keypoint algorithms, such as FAST [153]. An object $\mathbf{o}_i = (\mathbf{c}_i, \mathbf{i}_i)$ is an instance $\mathbf{i}_i$ of a semantic class $\mathbf{c}_i$, detectable via object recognition algorithms, such as YOLO [150]. The precise definitions of object class and instance follow.

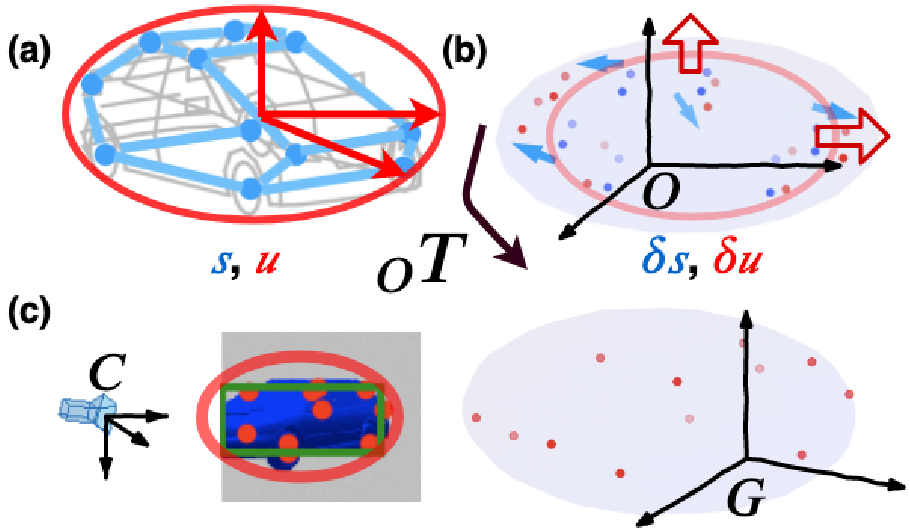

Figure 4.2. (a) An object class is defined by a semantic class σ and average shape specified by semantic landmarks $\mathbf{s}$ (blue) and an ellipsoid with semi-axes lengths $\mathbf{u}$ (red). (b) A specific instance has landmark and ellipsoid deformations parameterized by $\delta\mathbf{s}$ (blue arrows) and $\delta\mathbf{u}$ (red arrows). (c) The landmarks and ellipsoid are transformed from the object frame $\{O\}$ to the global frame $\{G\}$ via the instance pose ${}_O\mathbf{T}$.

Definition 3. *An* object class *is a tuple* $\mathbf{c} \triangleq (\sigma, \mathbf{u}, \mathbf{s})$*, where* $\sigma \in \mathbb{N}$ *specifies a semantic type (e.g., chair, table, monitor),* $\mathbf{u} \in \mathbb{R}^3$ *and* $\mathbf{s} \in \mathbb{R}^{3\times N_s}$ *specify the average class shape. The class shape is modeled by an axis-aligned ellipsoid* $\mathcal{E}_{\mathbf{u}}$ *and a set of* semantic landmarks $\mathbf{s}_l \in \mathbb{R}^3$ *corresponding to the columns of* $\mathbf{s}$*. The semantic landmarks* $\mathbf{s}_l$ *define the 3D positions of object parts (such as the front wheel of a car) in the object canonical frame* $\{O\}$*.*

Definition 4. *An* object instance *of class* $\mathbf{c}$ *is a tuple* $\mathbf{i} \triangleq ({}_O\mathbf{T}, \delta\mathbf{s}, \delta\mathbf{u})$*, where* ${}_O\mathbf{T} \in SE(3)$ *is the instance pose and* $\delta\mathbf{s} \in \mathbb{R}^{3\times N_s}$*,* $\delta\mathbf{u} \in \mathbb{R}^3$ *are deformations of the average class-level semantic landmarks* $\mathbf{s}$ *and ellipsoid semi-axes lengths* $\mathbf{u}$*.*

The shape of an object $\mathbf{o}_i$ in the global frame $\{G\}$ is obtained by deforming and transforming the semantic landmark positions, ${}_O\mathbf{T}(\underline{\mathbf{s}}_l + \underline{\delta\mathbf{s}}_l)$, and the dual ellipsoid, ${}_O\mathbf{T}\mathbf{Q}^*_{(\mathbf{u}+\delta\mathbf{u})}{}_O\mathbf{T}^\top$, using the instance pose ${}_O\mathbf{T}$ and deformations $\delta\mathbf{s}$, $\delta\mathbf{u}$. Fig. 4.2 shows an illustration of a car model with 12 semantic landmarks.

The IMU-camera sensor provides inertial measurements ${}^i\mathbf{z}_k$, geometric keypoint mea-

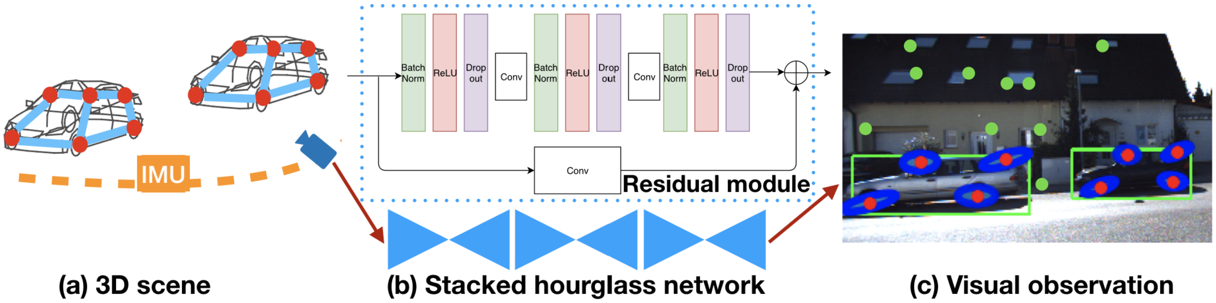

Figure 4.3. OrcVIO utilizes visual-inertial information to optimize the sensor trajectory and the shapes and poses of objects. The observations include geometric keypoints (FAST keypoints indicated by green dots in (c)), semantic keypoints (car parts indicated by red dots with blue ellipse covariances in (c)), bounding-box lines (green lines in (c)), and inertial data (orange dotted lines in (a)). The semantic keypoints and their covariances are obtained from a stacked hourglass CNN in (b) composed of residual modules (blue dotted rectangle in (b)), including convolution, ReLU, batch normalization, and dropout layers. The dropout layers are used to sample different weight realizations and estimate the semantic keypoint covariances.

surements ${}^{g}\mathbf{z}_{k,n}$, and semantic measurements, containing object class ${}^{c}\mathbf{z}_{k,j}$, bounding-box ${}^{b}\mathbf{z}_{k,l,j}$, and semantic keypoint ${}^{s}\mathbf{z}_{k,l,j}$ detections, illustrated in Fig. 4.3. The inertial measurements ${}^{i}\mathbf{z}_k \triangleq ({}^{i}\omega_k, {}^{i}\mathbf{a}_k) \in \mathbb{R}^6$ are the IMU's body-frame angular velocity ${}^{i}\omega_k$ and linear acceleration ${}^{i}\mathbf{a}_k$ at time t_k. The geometric keypoint measurements are noisy detections ${}^{g}\mathbf{z}_{k,n} \in \mathbb{R}^2$ in normalized pixel coordinates of the image projections of the subset of geometric landmarks $\mathcal{L}$ visible to the camera at time t_k. To obtain semantic observations, an object detection algorithm is applied to the image at time t_k, followed by semantic keypoint extraction within each detected bounding-box (see Sec. 4.4.1 for details). The j-th detected object provides the object class ${}^{c}\mathbf{z}_{k,j} \in \mathbb{N}$, bounding box $\beta_{k,j} = (u_0, v_0, u_1, v_1) \in \mathbb{R}^4$ with top left corner (u_0, v_0) and bottom right corner (u_1, v_1) are normalized pixel coordinates, and semantic keypoints ${}^{s}\mathbf{z}_{k,l,j} \in \mathbb{R}^2$ in normalized pixel coordinates associated with the $l = 1, \ldots, N_s$ semantic landmarks. Given a bounding box $\beta_{k,j}$, we consider the four bounding-box line segments ${}^{b}\mathbf{z}_{k,l,j} \in \mathbb{R}^2$ in normalized pixel coordinates, defined as:

$$\begin{aligned} {}^{b}\mathbf{z}_{k,1,j} &= [0, -1/v_0]^\top, \quad {}^{b}\mathbf{z}_{k,2,j} = [-1/u_1, 0]^\top, \\ {}^{b}\mathbf{z}_{k,3,j} &= [0, -1/v_1]^\top, \quad {}^{b}\mathbf{z}_{k,4,j} = [-1/u_0, 0]^\top, \end{aligned} \tag{4.16}$$

as separate measurements. Let $\mathbb{1}_{k,m,n} \in \{0,1\}$ indicate whether the n-th geometric keypoint observed at time t_k is associated with the m-th geometric landmark. Similarly, let $\mathbb{1}_{k,i,j} \in \{0,1\}$ indicate whether the j-th object detection at time t_k is associated with the i-th object instance. This data association information can be obtained by keypoint and object tracking as described in Sec. 4.4.1. Given the associations, we introduce error functions:

$$\begin{aligned} {}^{i}\mathbf{e}_{k,k+1} &\triangleq {}^{i}\mathbf{e}\left(\mathbf{x}_k, \mathbf{x}_{k+1}, {}^{i}\mathbf{z}_k\right) \quad & {}^{g}\mathbf{e}_{k,m,n} &\triangleq {}^{g}\mathbf{e}\left(\mathbf{x}_k, \ell_m, {}^{g}\mathbf{z}_{k,n}\right) \\ {}^{s}\mathbf{e}_{k,i,l,j} &\triangleq {}^{s}\mathbf{e}\left(\mathbf{x}_k, \mathbf{o}_i, {}^{s}\mathbf{z}_{k,l,j}\right) \quad & {}^{b}\mathbf{e}_{k,i,l,j} &\triangleq {}^{b}\mathbf{e}\left(\mathbf{x}_k, \mathbf{o}_i, {}^{b}\mathbf{z}_{k,l,j}\right) \end{aligned}$$

for the inertial, geometric, semantic keypoint and bounding-box measurements, respectively, defined precisely in Sec. 4.4.2. We also introduce an object shape regularization error term ${}^{r}\mathbf{e}_i \triangleq {}^{r}\mathbf{e}(\mathbf{o}_i)$ to ensure that the instance deformations $(\delta\mathbf{s}_i, \delta\mathbf{u}_i)$ remain small, and consider the following problem.

Problem. *Determine the sensor trajectory $\mathcal{X}^*$, geometric landmarks $\mathcal{L}^*$, and object states $\mathcal{O}^*$ that minimize the sum of squared errors:*

$$\begin{aligned} \min_{\mathcal{X},\mathcal{L},\mathcal{O}} & \sum_k \|{}^{i}\mathbf{e}_{k,k+1}\|^2_{{}^{i}\mathbf{V}} + \sum_{k,m,n} \mathbb{1}_{k,m,n} \|{}^{g}\mathbf{e}_{k,m,n}\|^2_{{}^{g}\mathbf{V}} \\ & + \sum_{k,i,l,j} \mathbb{1}_{k,i,j} \|{}^{s}\mathbf{e}_{k,i,l,j}\|^2_{{}^{s}\mathbf{V}} + \sum_{k,i,l,j} \mathbb{1}_{k,i,j} \|{}^{b}\mathbf{e}_{k,i,l,j}\|^2_{{}^{b}\mathbf{V}} + \sum_i \|{}^{r}\mathbf{e}_i\|^2_{{}^{r}\mathbf{V}} \end{aligned} \tag{4.17}$$

where ${}^{*}\mathbf{V}$ are positive-definite matrices specifying the covariances of the inertial, geometric, semantic, bounding-box, and shape regularization errors. An error covariance $\mathbf{V}$ defines a quadratic (Mahalanobis) norm $\|\mathbf{e}\|^2_{\mathbf{V}} \triangleq \mathbf{e}^\top \mathbf{V}^{-1} \mathbf{e}$.

4.4 Estimating Ellipsoid Object and Camera Pose from 2D Semantic Detection and Keypoint

Inspired by the MSCKF algorithm [121], we decouple the optimization in Eq. (4.17) over $\mathcal{L}$ and $\mathcal{O}$ from that over $\mathcal{X}$ to design an efficient real-time algorithm. When a geometric-keypoint or object track is lost, we perform multi-view iterative optimization over the corresponding geometric landmark ℓ_m or object $\mathbf{o}_i$ based on the estimate of the latest IMU-camera state $\mathbf{x}_k$. The IMU-camera state is propagated using the inertial observations and updated using the optimized geometric landmark and object states and the geometric and semantic observations. This decoupling leads to potentially lower accuracy but higher efficiency compared to window or batch keyframe optimization techniques [14]. Our approach offers tight coupling between semantic information and geometric structure in visual-inertial odometry. In Sec. 4.4.1, we describe the detection and tracking of the geometric and semantic observation in the front end. The error functions of the geometric and semantic observations and Jacobians w.r.t. the poses and the objects are introduced in Sec. 4.4.2. In Sec. 4.4.3 we describe the initialization of the object and how it is used to update the tracked IMU pose. Our method is named *Object residual constrained Visual Inertial Odometry* (OrcVIO) to emphasize the role of the semantic residuals in the localization and mapping optimization process. OrcVIO is capable of producing object maps and estimating sensor trajectories, as shown in Fig. 4.4

4.4.1 Keypoint and Object Tracking

This subsection discusses the detection and the tracking of geometric keypoint ${}^{g}\mathbf{z}_{k,n}$, object class ${}^{c}\mathbf{z}_{k,j}$, bounding-box ${}^{b}\mathbf{z}_{k,l,j}$, and semantic keypoint ${}^{s}\mathbf{z}_{k,l,j}$ measurements in the camera images in the algorithm front-end.

Geometric keypoints ${}^{g}\mathbf{z}_{t,n}$ are detected using the FAST detector [153] and are tracked temporally using the Lucas-Kanade (LK) algorithm [112]. Keypoint tracking has lower accuracy but higher efficiency than descriptor-matching methods, allowing our method to use a high frame-rate camera. Outliers are eliminated by estimating the essential matrix between two consecutive views and removing those keypoints that do not fit the estimated model. Assuming that the time between two consecutive images is short, the relative orientation is obtained by integrating the gyroscope measurements ${}^{i}\omega_k$ and only the unit translation vector is estimated using two-point RANSAC [99].

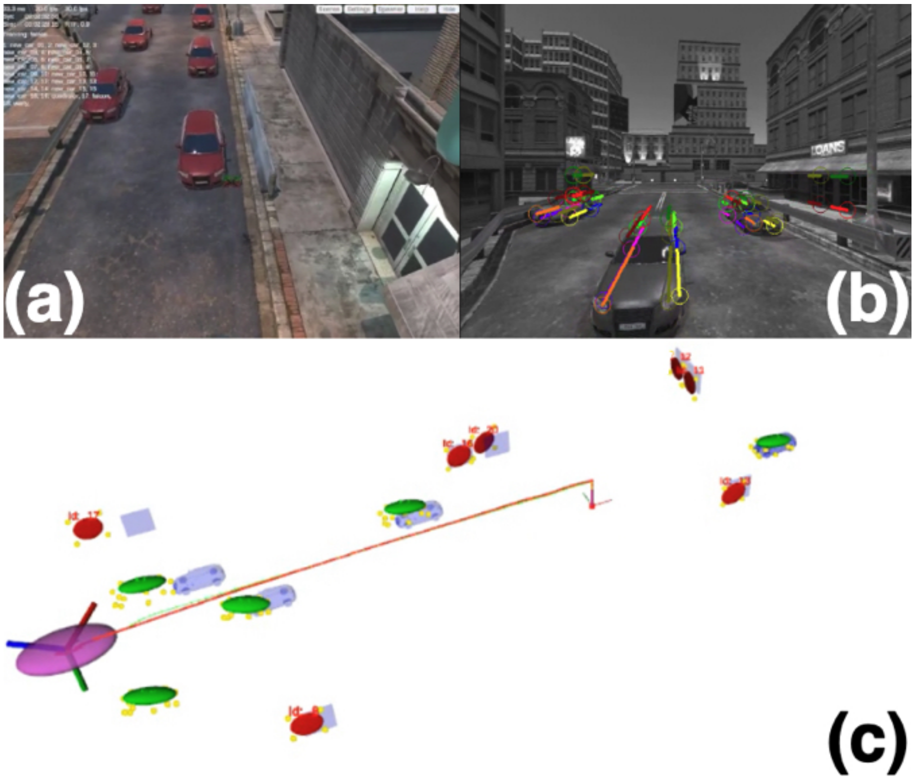

Figure 4.4. This work develops a tightly coupled visual-inertial odometry and object state estimation algorithm. The figure shows: (a) a quadrotor robot navigating in a simulated Unity environment, containing cars and doors, (b) color-coded semantic keypoints detected and tracked on the cars and doors, and (c) ground-truth (red) and estimated (green) robot trajectory as well as ground-truth cars and doors (blue meshes) and estimated cars (green ellipsoids) and doors (red ellipsoids) along with their semantic keypoints (yellow points).

The YOLOv4 detector [12] is used to detect object classes ${}^{c}\mathbf{z}_{k,j}$ and bounding-box lines ${}^{b}\mathbf{z}_{k,l,j}$. Semantic keypoints ${}^{s}\mathbf{z}_{k,l,j}$ are extracted within each bounding box using the StarMap stacked hourglass convolutional neural network [211]. We augment the original StarMap network with dropout layers as shown in Fig. 4.3(b). We perform several stochastic forward passes using Monte Carlo dropout [55] offline on a validation image dataset to obtain semantic keypoint covariances ${}^{s}\mathbf{V}$, illustrated in Fig. 4.3(c). The matrices ${}^{s}\mathbf{V}$ are fixed for the actual test-time algorithm deployment to ensure real-time operation.

The bounding boxes are tracked temporally using the SORT algorithm [10]. At each frame, the new object detections from YOLO are filtered to prune those with low confidence. A similarity score between each valid new detection and each existing track is computed using

Intersection-over-Union (IoU), and then the Hungarian algorithm is used to match the detections with the tracks whose scores are above a threshold. After a successful match, the bounding-box is updated using a Kalman filter update step and a constant velocity model is used to predict its motion to the next frame. Each unmatched new detection is initialized as a new track, and the tracks with no match for a few frames are removed. There are several cases where tracking may fail. When the relative motion between the object and the camera is large, e.g., the camera is moving fast, tracking may fail because the constant velocity model does not predict the bounding-box trajectory reliably. Another possible failure case happens when the object is occluded and the IoU score is smaller than the threshold. Those fail cases decrease the number of objects in the map produced by OrcVIO. To mitigate tracking failures, we fine tune the parameters such that few valid matches are needed to establish a track. We also prune objects that are too close to each other in 3D to prevent duplicate object mapping due to spurious tracks. Another solution is to only consider objects within a certain distance from the camera since objects far away have smaller bounding boxes and a higher chance to introduce spurious tracks. The semantic keypoints ${}^{s}\mathbf{z}_{k,l,j}$ within each bounding box are tracked via a Kalman filter, which uses the LK algorithm for prediction and the StarMap keypoint detections for update.

4.4.2 Landmark and Object Error Functions and Jacobians

Next, we define the geometric-keypoint ${}^{g}\mathbf{e}_{k,m,n}$, semantic-keypoint ${}^{s}\mathbf{e}_{k,i,l,j}$, bounding-box ${}^{b}\mathbf{e}_{k,i,l,j}$, and regularization ${}^{r}\mathbf{e}_{i}$ error terms in Eq. (4.17) and derive their Jacobians. The error functions are linearized around estimates of the IMU-camera state $\hat{\mathbf{x}}_k$, geometric landmarks $\hat{\ell}_m$, and objects $\hat{\mathbf{o}}_i$.

We define the geometric-keypoint error as the difference between the image projection of a geometric landmark ℓ in camera frame and its associated keypoint observation ${}^{g}\mathbf{z}$:

$$
{}^{g}\mathbf{e}\left(\mathbf{x}, \ell, {}^{g}\mathbf{z}\right) \triangleq \pi\left({}^{C}_{G}h\left(\ell\right)\right) - {}^{g}\mathbf{z}, \tag{4.18}
$$

where

$$ {}^C_G h(\ell) ={}^C_G \mathbf{R}(\ell -{}_C\,\mathbf{p}) ={}^C_I \mathbf{R}{}^I_G\mathbf{R}(\ell -{}_I\,\mathbf{p}) +{}^C_I \mathbf{p}, \tag{4.19} $$

is the Euclidean transformation from global frame to camera frame and

$$ \pi(\mathbf{s}) \triangleq \begin{bmatrix} s_1/s_3 & s_2/s_3 \end{bmatrix}^\top \in \mathbb{R}^2, \tag{4.20} $$

is the perspective projection function. The Jacobian of ${}^g\mathbf{e}$ with respect to the landmark ℓ is

$$ \begin{aligned} \frac{\partial {}^g\mathbf{e}}{\partial \ell} &= \frac{\partial \pi}{\partial \mathbf{s}} \frac{\partial {}^C_G h}{\partial \ell}, && (4.21) \\ \frac{\partial \pi}{\partial \mathbf{s}} &= \begin{bmatrix} 1/s_3 & 0 & -s_1/s_3^2 \\ 0 & 1/s_2 & -s_2/s_3^2 \end{bmatrix}, && (4.22) \\ \frac{\partial {}^C_G h}{\partial \ell} &={}^C_I \mathbf{R}{}^I_G\mathbf{R}. && (4.23) \end{aligned} $$

The Jacobian of ${}^g\mathbf{e}$ with respect to the IMU pose is

$$ \begin{aligned} \frac{\partial {}^g\mathbf{e}}{\partial {}^I_G\mathbf{R}} &= \frac{\partial \pi}{\partial \mathbf{s}} \frac{\partial {}^C_G h}{\partial {}^I_G\mathbf{R}}, \\ \frac{\partial {}^C_G h}{\partial {}^I_G\mathbf{R}} &={}^C_I \mathbf{R}\lfloor {}^I_G\mathbf{R}(\ell -{}_I\,\mathbf{p})\times\rfloor, \\ \frac{\partial {}^g\mathbf{e}}{\partial {}_I\mathbf{p}} &= \frac{\partial \pi}{\partial \mathbf{s}} \frac{\partial {}^C_G h}{\partial {}_I\mathbf{p}}, \\ \frac{\partial {}^C_G h}{\partial {}_I\mathbf{p}} &= -{}^C_I\mathbf{R}{}^I_G\mathbf{R}. \end{aligned} \tag{4.24} $$

The semantic keypoint error is defined as the difference between the projection of a semantic landmark $\mathbf{s}_l + \delta\mathbf{s}_l$ from the object frame to the image plane, using instance pose ${}_O\mathbf{T}$

and camera pose ${}_C\mathbf{T}$, and its corresponding semantic-keypoint observation ${}^s\mathbf{z}_l$:

$$\begin{aligned} {}^s\mathbf{e}_l(\mathbf{x},\mathbf{o},{}^s\mathbf{z}) &\triangleq \pi\left({}^C_O h\left(\mathbf{s}_l+\delta\mathbf{s}_l\right)\right)-{}^s\mathbf{z}_l, && (4.25)\\ {}^C_O h(\mathbf{s}) &=^C_I \mathbf{R}^I_G\mathbf{R}(({}_O\mathbf{R}\mathbf{s}+_O\mathbf{p})-_I\mathbf{p})+^C_I\mathbf{p} && (4.26) \end{aligned}$$

The Jacobians w.r.t. the semantic keypoint and the IMU pose are similar to Eq. (4.21) and Eq. (4.24). The Jacobian w.r.t. the object pose is similar to Eq. (4.24) but with different Jacobian on the Euclidean transformation equation ${}^C_O h(\mathbf{s})$,

$$\begin{aligned} \frac{\partial {}^C_O h}{\partial {}_O\mathbf{R}} &=^C_I \mathbf{R}^I_G\mathbf{R}\lfloor {}_O\mathbf{R}(\mathbf{s}_l+\delta\mathbf{s}_l)\times\rfloor,\\ \frac{\partial {}^C_O h}{\partial {}_O\mathbf{p}} &=^C_I \mathbf{R}^I_G\mathbf{R}. \end{aligned} \qquad (4.27)$$

We define the bounding-box error as the distance between the hyperplane induced by projecting a bounding-box line ${}^b\mathbf{z}$ in Eq. (4.16) to the object frame and the closest hyperplane that is tangent to the quadric surface $\mathbf{Q}^*_{(\mathbf{u}+\delta\mathbf{u})}$ of object $\mathbf{o}$. To derive the hyperplanes defined by the camera center and the bounding box lines, we perform cross product over the four corner vector.

$$\begin{aligned} {}^b\underline{\mathbf{z}}_{k,1,j} &= [(u_0,v_0,1)\times(u_1,v_0,1),0]^\top = [0,-1/v_0,1,0]^\top,\\ {}^b\underline{\mathbf{z}}_{k,2,j} &= [(u_1,v_0,1)\times(u_1,v_1,1),0]^\top = [-1/u_1,0,1,0]^\top,\\ {}^b\underline{\mathbf{z}}_{k,3,j} &= [(u_1,v_1,1)\times(u_0,v_1,1),0]^\top = [0,-1/v_1,1,0]^\top,\\ {}^b\underline{\mathbf{z}}_{k,4,j} &= [(u_0,v_1,1)\times(u_0,v_0,1),0]^\top = [-1/u_0,0,1,0]^\top, \end{aligned} \qquad (4.28)$$

Without loss of generality, we can normalize the bounding planes and write them as ${}^b\underline{\mathbf{z}} = \begin{bmatrix} {}^b\mathbf{z}^\top & 0\end{bmatrix}^\top$ where $\|{}^b\mathbf{z}\| = 1$ is a normalized 3D vector. We are going to compute the distance from this ellipsoid tangent plane to the bounding box hyperplane in the object frame. Given an initial plane in the camera frame ${}^b\underline{\mathbf{z}} \in \mathbb{R}^4$, we can convert it into a plane in the object frame

$\underline{\mathbf{b}} = \begin{bmatrix}\mathbf{b}^\top & \lambda\end{bmatrix}^\top \in \mathbb{R}^4.$

$$\underline{\mathbf{b}} = ({}_O\mathbf{T}^{-1}{}_C\mathbf{T})^{-\top\,b}\underline{\mathbf{z}} = {}_O\mathbf{T}^\top{}_C\mathbf{T}^{-\top\,b}\underline{\mathbf{z}} \tag{4.29}$$

where ${}_C\mathbf{T}$ is the camera pose and ${}_O\mathbf{T}$ is the object pose.

$$\begin{bmatrix}\mathbf{b}\\ \lambda\end{bmatrix} = \begin{bmatrix}{}_O\mathbf{R}^\top & 0\\ {}_O\mathbf{p}^\top & 1\end{bmatrix}\begin{bmatrix}{}_C\mathbf{R} & 0\\ -{}_C\mathbf{p}^\top{}_C\mathbf{R} & 1\end{bmatrix}\begin{bmatrix}{}^b\mathbf{z}\\ 0\end{bmatrix} = \begin{bmatrix}{}_O\mathbf{R}^\top{}_C\mathbf{R}^b\mathbf{z}\\ {}_O\mathbf{p}^\top{}_C\mathbf{R}^b\mathbf{z} - {}_C\mathbf{p}^\top{}_C\mathbf{R}\mathbf{z}\end{bmatrix} \tag{4.30}$$

$$\mathbf{b} = {}_O\mathbf{R}^\top{}_C\mathbf{R}^b\mathbf{z} \tag{4.31}$$

$$\lambda = ({}_O\mathbf{p} - {}_C\mathbf{p})^\top{}_C\mathbf{R}^b\mathbf{z} \tag{4.32}$$

where $||\mathbf{b}|| = 1$ as it is derived from rotating normalized vector ${}^b\mathbf{z}$. The camera pose is derived from the IMU pose and the IMU-camera extrinsic.

$$\begin{aligned}{}_C\mathbf{R} &= {}^I_G\mathbf{R}^\top{}^C_I\mathbf{R}^\top\\ {}_C\mathbf{p} &= {}^I_G\mathbf{R}^\top{}^I_C\mathbf{p} + {}_I\mathbf{p} = -{}^I_G\mathbf{R}^\top{}^C_I\mathbf{R}^\top{}^C_I\mathbf{p} + {}_I\mathbf{p}\end{aligned} \tag{4.33}$$

Given a hyperplane $\underline{\mathbf{b}} = \begin{bmatrix}\mathbf{b}^\top & \lambda\end{bmatrix}^\top$ and an origin-centered axis-aligned ellipsoid as a dual quadric $\mathbf{Q}^* = \mathrm{diag}(\mathbf{U}^2, -1)$. There should be another plane $\underline{\mathbf{b}}^* = \begin{bmatrix}\mathbf{b}^\top & \lambda^*\end{bmatrix}^\top$ that shares the same normal as $\underline{\mathbf{b}}$ and is tangent to $\mathbf{Q}^*$. To derive this plane, we have

$$\begin{aligned}&\underline{\mathbf{b}}^{*\top}\mathbf{Q}^*\underline{\mathbf{b}}^* = 0\\ &= \begin{bmatrix}\mathbf{b}^\top & \lambda^*\end{bmatrix}\begin{bmatrix}\mathbf{U}^2 & \mathbf{0_{3\times 1}}\\ \mathbf{0_{1\times 3}} & -1\end{bmatrix}\begin{bmatrix}\mathbf{b}\\ \lambda^*\end{bmatrix}\\ &= \mathbf{b}^\top\mathbf{U}^2\mathbf{b} - \lambda^{*2} = 0\\ \lambda^* &= \pm\sqrt{\mathbf{b}^\top\mathbf{U}^2\mathbf{b}}\end{aligned} \tag{4.34}$$

There are two tangent planes of the ellipsoid that share the same normal direction. Assuming the observed bounding plane is close to the tangent plane, we can choose the tangent plane that shares the same sign at the last element as the bounding plane. Then the distance between the two parallel planes, as the observation residual, is

$$ {}^{b}\mathbf{e}(\mathbf{x}, \mathbf{o}, {}^{b}\mathbf{z}) = \operatorname{sgn}(\lambda)\sqrt{\mathbf{b}^{\top}\mathbf{U}^{2}\mathbf{b}} - \lambda \tag{4.35} $$

where $\operatorname{sgn}(x) = \frac{\partial |x|}{\partial x}$ is the signum function and $\mathbf{b}, \lambda$ are as defined in Eq. (4.30).

The Jacobian of ${}^{b}\mathbf{e}$ with respect to the object pose is

$$ \frac{\partial {}^{b}\mathbf{e}}{\partial {}_{O}\mathbf{R}} = \frac{\partial {}^{b}\mathbf{e}}{\partial \underline{\mathbf{b}}} \frac{\partial \underline{\mathbf{b}}}{\partial {}_{O}\mathbf{R}}, \tag{4.36} $$

$$ \frac{\partial {}^{b}\mathbf{e}}{\partial {}_{O}\mathbf{p}} = \frac{\partial {}^{b}\mathbf{e}}{\partial \underline{\mathbf{b}}} \frac{\partial \underline{\mathbf{b}}}{\partial {}_{O}\mathbf{p}}, \tag{4.37} $$

$$ \frac{\partial {}^{b}\mathbf{e}}{\partial \underline{\mathbf{b}}} = \operatorname{sgn}(\lambda) \frac{\begin{bmatrix} \mathbf{b}^{\top}\mathbf{U}^{2} & 0 \end{bmatrix}}{\sqrt{\mathbf{b}^{\top}\mathbf{U}^{2}\mathbf{b}}} - [0, 0, 0, 1], \tag{4.38} $$

$$ \frac{\partial \underline{\mathbf{b}}}{\partial {}_{O}\mathbf{R}} = \begin{bmatrix} -{}_{O}\mathbf{R}^{\top} \lfloor {}_{C}\mathbf{R}^{b}\mathbf{z} \times \rfloor \\ \mathbf{0}_{1\times 3} \end{bmatrix}, \tag{4.39} $$

$$ \frac{\partial \underline{\mathbf{b}}}{\partial {}_{O}\mathbf{p}} = \begin{bmatrix} \mathbf{0}_{3\times 3} \cdot \\ {}^{b}\mathbf{z}^{\top} {}_{C}\mathbf{R}^{\top} \end{bmatrix} \tag{4.40} $$

We treat $\operatorname{sgn}(\lambda)$ as a constant and ignore its Jacobian w.r.t. λ. Notice that ${}_{C}\mathbf{R} = ({}_{I}^{C}\mathbf{R}{}_{G}^{I}\mathbf{R})^{\top}$ is the rotation from camera frame to global frame. The Jacobian of ${}^{b}\mathbf{e}$ with respect to the object scale $\mathbf{u}$ is

$$ \frac{\partial {}^{b}\mathbf{e}}{\partial \mathbf{u}} = \operatorname{sgn}(\lambda) \frac{\mathbf{b} \circ \mathbf{b} \circ \mathbf{u}}{\sqrt{\mathbf{b}^{\top}\mathbf{U}^{2}\mathbf{b}}} \tag{4.41} $$

where $\circ$ is the Hadamard product or element-wise product. The Jacobian of ${}^{b}\mathbf{e}$ with respect to the IMU pose is similar to those above with the difference

$$\frac{\partial \underline{\mathbf{b}}}{\partial {}^I_G\mathbf{R}} = \begin{bmatrix} -{}_O\mathbf{R}^\top {}^I_G\mathbf{R}^\top \lfloor {}^C_I\mathbf{R}^\top {}^b\mathbf{z}\times \rfloor \\ -({}^I_G\mathbf{R}^\top \lfloor {}^C_I\mathbf{R}^\top {}^C_I\mathbf{p}\times \rfloor)^\top {}^I_G\mathbf{R}^\top {}^C_I\mathbf{R}^\top - ({}_O\mathbf{p} + {}^I_G\mathbf{R}^\top {}^C_I\mathbf{R}^\top {}^C_I\mathbf{p} - {}_I\mathbf{p})^\top {}^I_G\mathbf{R}^\top \lfloor {}^C_I\mathbf{R}^\top {}^b\mathbf{z}\times \rfloor \end{bmatrix}, \tag{4.42}$$

$$\frac{\partial \underline{\mathbf{b}}}{\partial {}_I\mathbf{p}} = \begin{bmatrix} \mathbf{0}_{3\times 3}\cdot \\ {}^b\mathbf{z}^\top {}_C\mathbf{R}^\top \end{bmatrix} \tag{4.43}$$

4.4.3 OrcVIO Algorithm

We return to the problem of joint IMU-camera, geometric-landmark, and object optimization and describe the Object residual constrained Visual Inertial Odometry (OrcVIO) algorithm. OrcVIO is an extension of MSCKF [121, 60], with an extension of adding the object and using the semantic observation residuals to update the tracked pose. The IMU-camera state $\mathbf{x}_k$ is tracked using an extended Kalman filter. Prediction step is performed using the inertial measurements ${}^i\mathbf{z}_k$.

When a geometric-keypoint or object track is lost, initialization and iterative optimization is performed over $\hat{\ell}_m$ and $\hat{\mathbf{o}}_i$. For this step we keep the estimated IMU poses fixed. The geometric landmarks $\hat{\ell}_m$ are initialized by solving ${}^g\hat{\mathbf{e}}_{k,m,n} = \mathbf{0}$ via the linear system of equations:

$$\mathbf{P}_C \hat{\mathbf{T}}_k^{-1} \underline{\hat{\ell}}_m - \lambda_{k,n} {}^g\mathbf{z}_{k,n} = \mathbf{0} \tag{4.44}$$

for all k, m, n such that $\mathbb{1}_{k,m,n} = 1$, where the unknowns are $\hat{\ell}_m$ and the keypoint depths $\lambda_{k,n}$. The deformations of an object instance $\hat{\mathbf{o}}$ are initialized as $\delta\hat{\mathbf{s}} = \mathbf{0}$ and $\delta\hat{\mathbf{u}} = \mathbf{0}$. The instance pose is determined from the system of equations consisting of semantic keypoint and bounding-box line residuals:

$$\begin{aligned} \mathbf{P}_C \hat{\mathbf{T}}_k^{-1} {}_O\hat{\mathbf{T}} \underline{\mathbf{s}}_l - \lambda_{k,l,j} {}^s\mathbf{z}_{k,l,j} = \mathbf{0} \\ {}^b\underline{\mathbf{z}}_{k,l,j}^\top \mathbf{P}_C \hat{\mathbf{T}}_k^{-1} {}_O\hat{\mathbf{T}} \mathbf{Q}_{\mathbf{u}}^* {}_O\hat{\mathbf{T}}^\top {}_C\hat{\mathbf{T}}_k^{-\top} \mathbf{P}^\top {}^b\underline{\mathbf{z}}_{k,l,j} = 0 \end{aligned} \tag{4.45}$$

for all l and all k, j such that $\mathbb{1}_{k,i,j} = 1$, where the unknowns are object pose ${}_O\hat{\mathbf{T}}$ and the semantic keypoint depths $\lambda_{k,l,j}$. The least squares problem for semantic keypoints is a generalization of the pose from n point correspondences (PnP) problem [74]. While this system may be solved using polynomial equations [198], we perform a more efficient initialization by defining $\zeta_l \triangleq {}_O\hat{\mathbf{R}}\mathbf{s}_l + {}_O\hat{\mathbf{p}}$ and solving the first set of (now linear) equations in (4.45) for ζ_l and $\lambda_{k,l,j}$. We recover ${}_O\hat{\mathbf{T}}$ via the Kabsch algorithm [89] between $\{\zeta_l\}$ and $\{\mathbf{s}_l\}$. This approach works well as long as there is a sufficient number of semantic keypoints ${}^s\mathbf{z}_{k,l,j}$ (at least two per landmark across time for at least three semantic landmarks $\mathbf{s}_l$) associated with the object. If fewer semantic keypoints are available, ${}_O\hat{\mathbf{T}}$ and $\delta\hat{\mathbf{u}}$ can be initialized using the LfD approach [155]. LfD fits ellipses, described by dual quadrics $\mathbf{C}_k^* \in \mathbb{R}^{3\times3}$, to the bounding-box detections ${}^b\underline{\mathbf{z}}_{k,l,j}$ with $\mathbb{1}_{k,i,j} = 1$ and vectorizes the equations $\mathbf{C}_k \propto \mathbf{P}_C\hat{\mathbf{T}}_k^{-1}\hat{\mathbf{Q}}^*{}_C\hat{\mathbf{T}}_k^{-\top}\mathbf{P}^\top$ to solve for $\hat{\mathbf{Q}}^* \triangleq {}_O\hat{\mathbf{T}}\mathbf{Q}^*_{\mathbf{u}+\delta\hat{\mathbf{u}}}{}_O\hat{\mathbf{T}}^\top$. The object pose ${}_O\hat{\mathbf{T}}$ and ellipsoid deformation $\delta\hat{\mathbf{u}}$ can be recovered from $\hat{\mathbf{Q}}^*$ by relating the estimated ellipsoid $\hat{\mathbf{Q}}^*$ in global coordinates to the ellipsoid $\mathbf{Q}^*_{\mathbf{u}+\delta\hat{\mathbf{u}}}$ in the object frame:

$$\hat{\mathbf{Q}}^* = {}_O\hat{\mathbf{T}}\mathbf{Q}^*_{\mathbf{u}+\delta\hat{\mathbf{u}}}{}_O\hat{\mathbf{T}}^\top = \begin{bmatrix} {}_O\hat{\mathbf{R}}\hat{\mathbf{U}}\hat{\mathbf{U}}^\top{}_O\hat{\mathbf{R}}^\top - {}_O\hat{\mathbf{p}}{}_O\hat{\mathbf{p}}^\top & -{}_O\hat{\mathbf{p}} \\ -{}_O\hat{\mathbf{p}}^\top & -1 \end{bmatrix},$$

where $\hat{\mathbf{U}} = \operatorname{diag}(\mathbf{u} + \delta\hat{\mathbf{u}})$. The translation ${}_O\hat{\mathbf{p}}$ can be recovered from the last column of $\hat{\mathbf{Q}}^*$. To recover the rotation and deformation, note that $\mathbf{A} \triangleq \mathbf{P}\hat{\mathbf{Q}}^*\mathbf{P}^\top + {}_O\hat{\mathbf{p}}{}_O\hat{\mathbf{p}}^\top = {}_O\hat{\mathbf{R}}\hat{\mathbf{U}}\hat{\mathbf{U}}^\top{}_O\hat{\mathbf{R}}^\top$ is a positive semidefinite matrix. Let its eigen-decomposition be $\mathbf{A} = \mathbf{V}\mathbf{Y}\mathbf{V}^\top$ so that ${}_O\hat{\mathbf{R}} = \mathbf{V}$ and $\hat{\mathbf{U}}\hat{\mathbf{U}}^\top = \mathbf{Y}$.

The optimized geometric landmark and object estimates are used to update the IMU-camera pose. Consider the nonlinear measurement function

$$\mathbf{z}_k = h(\mathbf{x}_k, \ell, \mathbf{o}) + \mathbf{n}_k \tag{4.46}$$

where we have the measurement noise $\mathbf{n}_{m,k} \sim \mathcal{N}(0, \mathbf{R}_k)$ and the measurement functions are introduced in 4.4.2. We linearize Eq. (4.46) with respect to the current zero-mean error state $\tilde{\mathbf{x}} = \mathbf{x} - \hat{\mathbf{x}}$ and get

$$\tilde{\mathbf{z}}_k = \mathbf{H}_k^x \tilde{\mathbf{x}}_{k|k-1} + \mathbf{H}_k^f \begin{bmatrix} \ell & \mathbf{o} \end{bmatrix} + \mathbf{n}_k \tag{4.47}$$

where $\mathbf{H}_k^x$ is the stacked measurement Jacobian w.r.t. the IMU poses and $\mathbf{H}_k^f$ is the stacked measurement Jacobian w.r.t. the geometric landmarks and semantic objects, also introduced in 4.4.2. By performing left nullspace projection, we can remove the term related to the landmarks and objects to have

$$\tilde{\mathbf{z}}_k^x = \mathbf{H}_k \tilde{\mathbf{x}}_{k|k-1} + \mathbf{n}_k \tag{4.48}$$

Using the linearized measurement model, we can now perform the standard EKF update on the state

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} \boxplus \mathbf{K}_k(\mathbf{z}_k - h(\hat{\mathbf{x}}_{k|k-1}, \ell, \mathbf{o})), \tag{4.49}$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k \mathbf{H}_k \mathbf{P}_{k|k-1}, \tag{4.50}$$

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^\top (\mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^\top + \mathbf{R}_k)^{-1}. \tag{4.51}$$

We implement the OrcVIO based on the OpenVINS version of MSCKF [60].

4.5 Experiments

We evaluate the ability of the proposed localization and mapping technique to optimize the camera trajectory and reconstruct object poses and shapes using both simulated and real data. Our experiments use images from monocular or stereo cameras and inertial odometry information. The real data including the open-source dataset KITTI [59] and our own collected data.

We use two standard metrics for quantitative VIO evaluation: position Root Mean Square Error (RMSE) [175] (also referred to as position Absolute Trajectory Error (ATE) [207]) and KITTI's translation error (TE) metric [59]. Let Trans($\mathbf{T}$) return the position component of a pose

$\mathbf{T} \in SE(3)$. Let ${}_I\mathbf{T}_k$ be the ground-truth pose trajectory and ${}_I\hat{\mathbf{T}}'_k$ be the estimated pose trajectory. To measure error, the estimated trajectory is first aligned to the initial frame of the ground-truth trajectory via ${}_I\hat{\mathbf{T}}_k = {}_I\mathbf{T}_0 {}_I\hat{\mathbf{T}}_0^{-1} {}_I\hat{\mathbf{T}}'_k$. After alignment, RMSE (m) and TE (%) are measured as:

$$\begin{aligned} \text{RMSE} &\triangleq \left(\frac{1}{K} \sum_{k=0}^{K-1} \left\| \text{Trans}\left({}_I\mathbf{T}_k^{-1} {}_I\hat{\mathbf{T}}_k \right) \right\|_2^2 \right)^{1/2}, \qquad (4.52) \\ \text{TE} &\triangleq \frac{1}{|\mathcal{F}|} \sum_{(i,j)\in\mathcal{F}} \frac{\left\| \text{Trans}\left(\left({}_I\hat{\mathbf{T}}_j^{-1} {}_I\hat{\mathbf{T}}_i \right)^{-1} \left({}_I\mathbf{T}_j^{-1} {}_I\mathbf{T}_i \right) \right) \right\|_2}{\text{length}(i,j)}, \end{aligned}$$

where $\mathcal{F}$ is a set of frames with fixed distances length(i,j) over specific values based on the total trajectory length.

The object estimates are evaluated using 3D Intersection over Union (IoU). A 3D bounding box $\hat{\beta}_i$ is obtained from each estimated object $\hat{\mathbf{o}}_i$, and IoU is defined as the ratio of the intersection volume over the union volume with respect to the bounding box β_i of the closest ground truth object:

$$\text{IoU}(\hat{\beta}_i, \beta_i) \triangleq \sum_i \frac{\text{Volume of Intersection}(\hat{\beta}_i, \beta_i)}{\text{Volume of Union}(\hat{\beta}_i, \beta_i)}. \qquad (4.53)$$

To understand the distribution of the object orientation and translation errors, we define an estimate as *true positive* if the closest ground-truth object pose is within a specific rotation or translation error threshold. Specifically, a rotation error of α° means $\|\log({}_O\hat{\mathbf{R}}^\top {}_O\mathbf{R})^\vee\|_2 \leq \alpha^\circ$, and translation error of $\beta\ m$ means $\|{}_O\mathbf{p} - {}_O\hat{\mathbf{p}}\|_2 \leq \beta\ m.$ We define *precision* as the fraction of true positives over all estimated objects and *recall* is the fraction of true positives over all ground-truth objects.

A few video demos of the ellipsoid object experiments can be found in the http://erl.ucsd.edu/pages/orcvio.html.

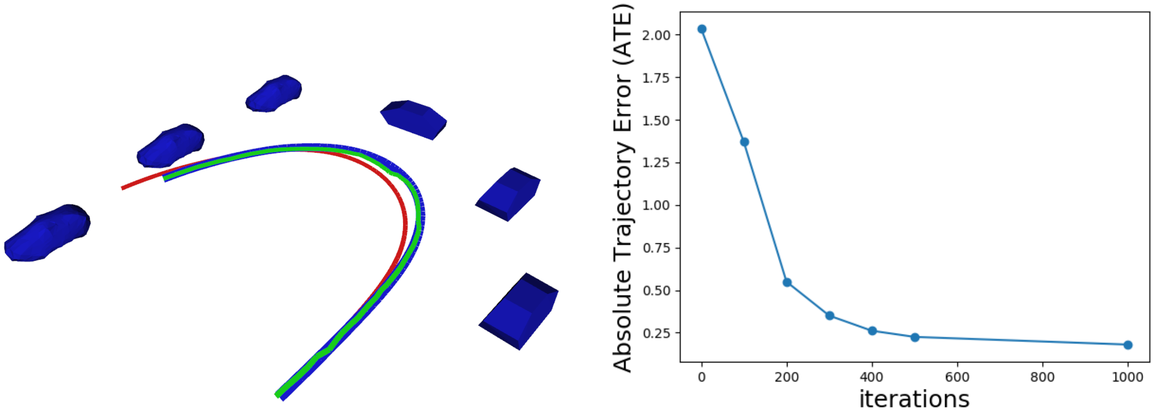

Figure 4.5. Left: Localization results from a simulated dataset, showing car poses (blue), the ground truth camera trajectory (blue), the inertial odometry used for initialization (red), and the optimized camera trajectory (green). Right: Change of Absolute Trajectory Error versus number of optimization iterations.

4.5.1 Mesh Object on Simulation Dataset

For the mesh representation, we focus on the car objects. We represent cars using a symmetric mesh model with 642 vertices and 1280 faces.

To model the real mechanism of IMU, we chose a sub-sequence IMU measurement and associated groundtruth pose from synchronized KITTI odometry dataset. We collected camera images following the groundtruth pose in a simulated Gazebo environment populated with car mesh models, so that we simulated a real camera-IMU sensor (see Fig. 4.5). The car models were annotated with keypoints and both the car surface and the keypoints were colored in contrasting colors to simplify the semantic segmentation and keypoint detection tasks. The simulated experiments used ground-truth data association among the observations. We evaluated both the localization and the mapping tasks.

For the localization task, we used a sequence with 70 frames and synchronized IMU measurements and 6 known cars were placed around. We initialized the estimation by predicting the transformation between two camera poses based on the IMU odometry. Then, we optimized the predicted camera pose by solving problem (4.2) and used the IMU to predict the next pose. An example camera trajectory and the associated localization results are shown in Fig. 4.5. We

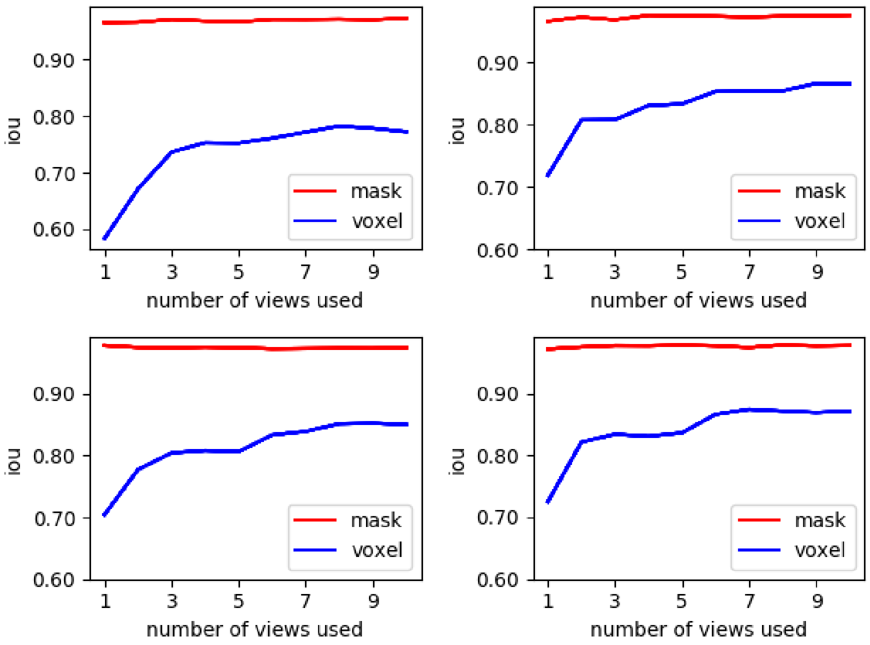

Figure 4.6. Mask and 3-D voxel intersection over union (IoU) results obtained with different numbers of object views for four different object instances (see Fig. 4.7). Notice that IoU of masks only take available masks (in x-axis) into account.

can see that our optimization successfully reduced the error accumulated from IMU integration.

The mapping performance was evaluated on a sequence of images obtained from different views of a single object (see Fig. 4.8). The optimization was initialized using a generic category-level car mesh and its vertices were optimized based on the detected keypoints and segmentation masks. The mapping quality is evaluated qualitatively using the Intersection over Union (IoU) ratio between the predicted and groundtruth car masks volumes. In detail, the mask IoU compares the area differences between predicted binary car masks, while the voxel IoU compares the voxelized volume of the predicted and groundtruth car models. Fig. 4.6 shows the dependence of the mapping accuracy on the number of different views used. The optimized car meshes are shown in Fig. 4.7. The differences among car models are clearly visible in the reconstructed meshes and their shapes are very close to the corresponding groundtruth shapes. Using only a few views, the optimization process is able to deform the mesh vertices to fit the segmentation masks but not necessarily align the estimated model with the real 3-D shape. As more observations

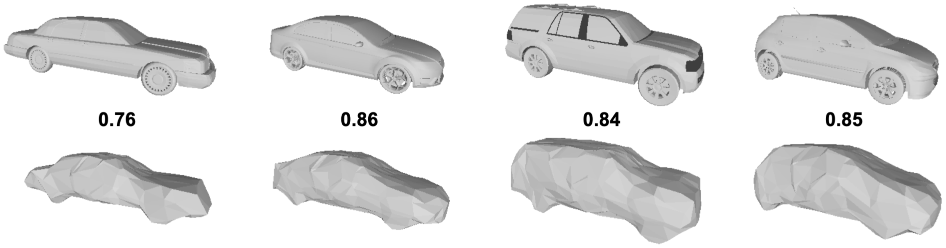

Figure 4.7. Qualitative comparison between estimated car shapes (bottom row) obtained from a simulation sequence and ground truth object meshes (top row). The numbers indicate 3D IoU.

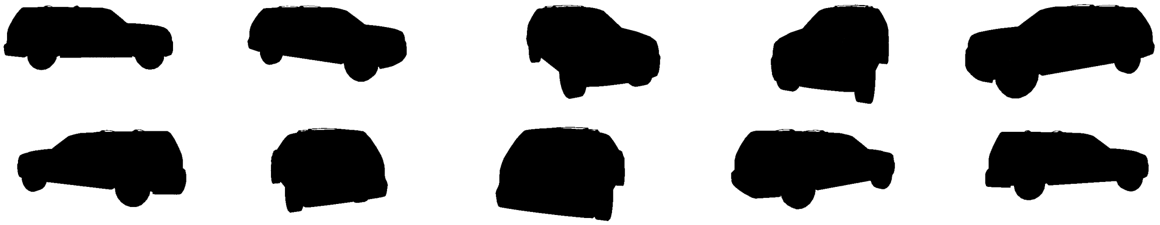

Figure 4.8. Views used to evaluate the object-level mapping approach in simulation.

become available, the 3D IoU increases, which makes sense since different views can provide information about additional instance-level characteristics. Based only on 3 views, the IoU reaches over 0.8, while the generic category-level mesh has an average IoU of 0.63 with respect to the different object instances.

4.5.2 Mesh Object on KITTI Dataset

Experiments with real observations were carried out using the KITTI dataset [59]. We choose three sequences with different lengths. The experiments used the ground-truth camera poses and evaluated only the mapping task. The object detector, semantic segmentation [195] and the keypoint detector [211] algorithms used pre-trained weights. Semantic observations were collected as described in Sec. 4.2.1. The poses and shapes of the detected cars were initialized and optimized as described in Sec. 4.2.4. Fig. 4.9 shows a bird-eye view of the estimated car poses and compares the results with the ground truth car positions provided in [5]. The poses and shapes of 56 out of 62 marked cars were reconstructed, with an average position error across all cars of 1.9 meters. Fig. 4.10 shows some estimated 3-D car mesh models projected back

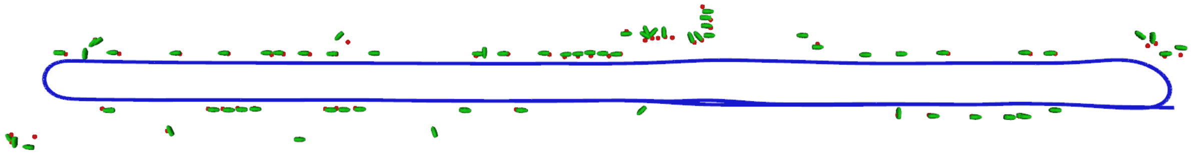

Figure 4.9. Qualitative results showing the accuracy of the estimated car positions (green) on sequence 06 of the KITTI odometry dataset in comparison with hand-labeled ground truth (red) obtained from [5]. The camera trajectory (blue) is shown as well.

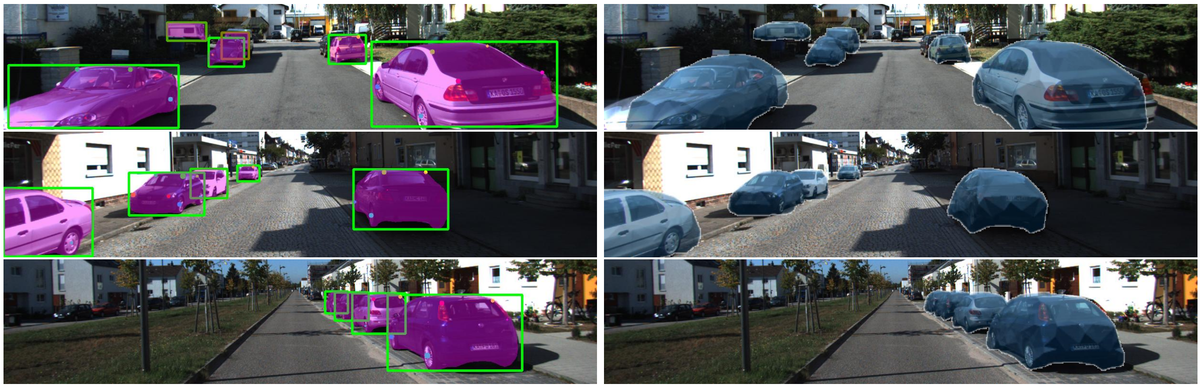

Figure 4.10. Left: the semantic observations. Right: the projection of reconstructed mesh models.

onto the camera images. Fig. 4.11 compares the differences between the category-level and instance-specific models. Fig. 4.12 shows the estimated model shapes and poses in 3D.

Since groundtruth object shapes are not available, we evaluate the quality of shape reconstruction based on the 2D IoU compared with the observed instance segmentation masks. We trained a single-image mesh predictor [92] on car data from the PASCAL3D+ dataset [196] and calculated its mean IoU for individual objects over multiple frames. Table 4.1 shows that our multi-view optimization method improves the IoU by leveraging semantic information from multiple images. The reconstruction quality on the real dataset is limited by the accuracy of the semantic information because the optimization objective is to align the predicted car shapes with the observed semantic masks and keypoints. The viewpoint changes on the real dataset are smaller, making the reconstruction task harder than in simulation. The pose estimation relies heavily on the keypoint detections, which in some cases are not robust enough. Nevertheless,

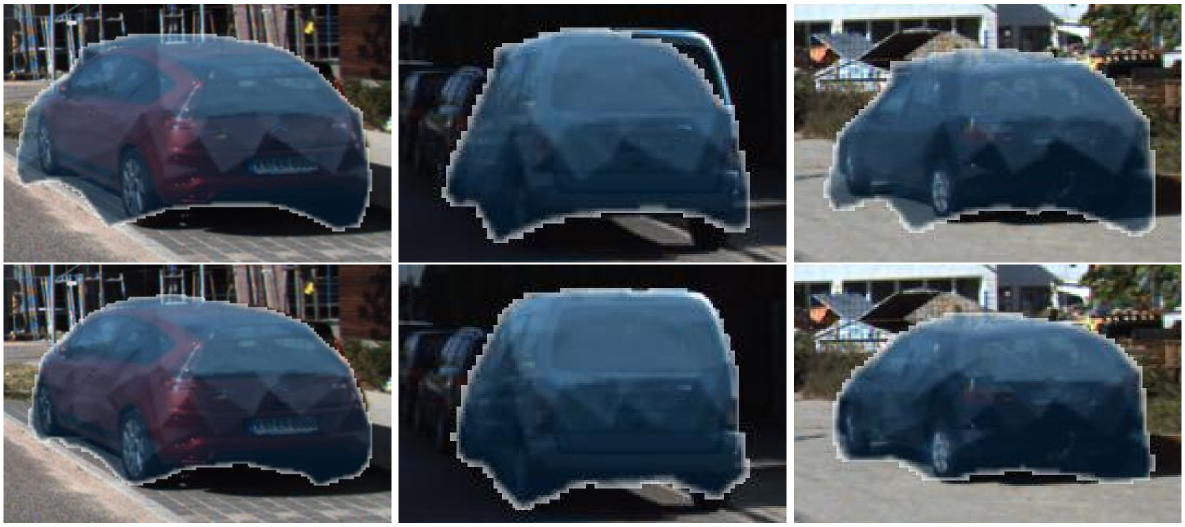

Figure 4.11. Top: category-level model before shape optimization. Bottom: instance-level model after shape optimization.

Table 4.1. 2D projection mIoU with respect to object segmentation on three KITTI sequence.

Dataset	09_26 0048	09_26 0035	09_30 0020
Frames	22	131	1101
Detected objects	6	28	77
Single image mesh prediction [92]	0.692	0.642	0.641
With pose estimation	0.735	0.656	0.689
With pose and shape estimation	**0.778**	**0.675**	**0.725**

our approach is able to generate accurate instance-specific mesh models in an environment containing occlusion and different lighting conditions.

4.5.3 Ellipsoid Object on Simulation Dataset

We first evaluate OrcVIO on a customized simulation. Given a trajectory, we use SE(3) B-spline interpolation to generate simulated IMU measurements [60]. The objects are modeled as ellipsoids and put along the trajectory. We project the ellipsoids onto the groundtruth camera poses. Then the projected 2D ellipses on the image can be computed and we use their minimum and maximum value along the UV axis on the image to draw the axis-aligned bounding boxes. The Gaussian noise can be added to the top-left and bottom-right cornmer of the bounding

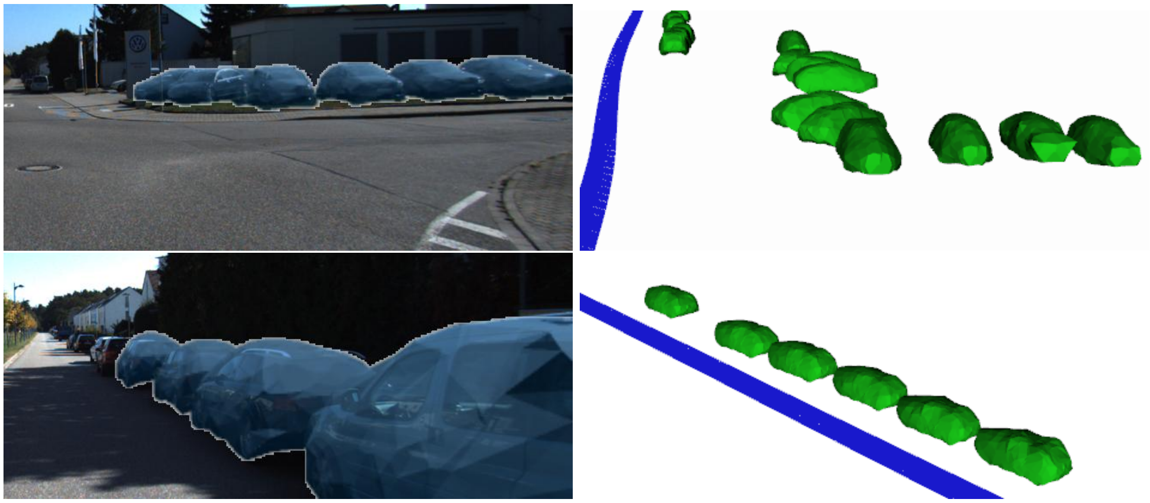

Figure 4.12. Left: 2D projections of mesh models. Right: corresponding 3D configuration. Trajectory in blue.

Table 4.2. Comparing localization accuracy with or without object update on the simulation dataset.

Methods	RMSE	TE(100m)
w/o Object Update	0.294	0.472%
w/ Object Update	0.220	0.361%

box. The bounding boxes are tagged with the object IDs so the groundtruth data association is provided. We constrain the object visible range between 2m and 30m. The qualitative result is shown in Fig. 4.13 and the quantitative result is reported in Table 4.2. Our method can get a relative good object estimation when having enough observations that span a large baseline. And introducing the object observations in the update step can improve the localization accuracy.

OrcVIO is evaluated in a Unity simulation [183] containing 50 car, road barrier, and door object instances. A ROS bridge between Unity and Gazebo is used to simulate a quadrotor robot, navigating in the environment and providing IMU and camera measurements. The object map reconstructed by OrcVIO is shown in Fig. 4.14. The estimated objects are generally quite close to the ground-truth ones. The object poses near the starting position are less accurate due to insufficient motion parallax, since the quadrotor performs a pure rotation in the beginning. The trajectory RMSE and TE are 0.97m and 0.40%, respectively. The odometry drift is mainly due

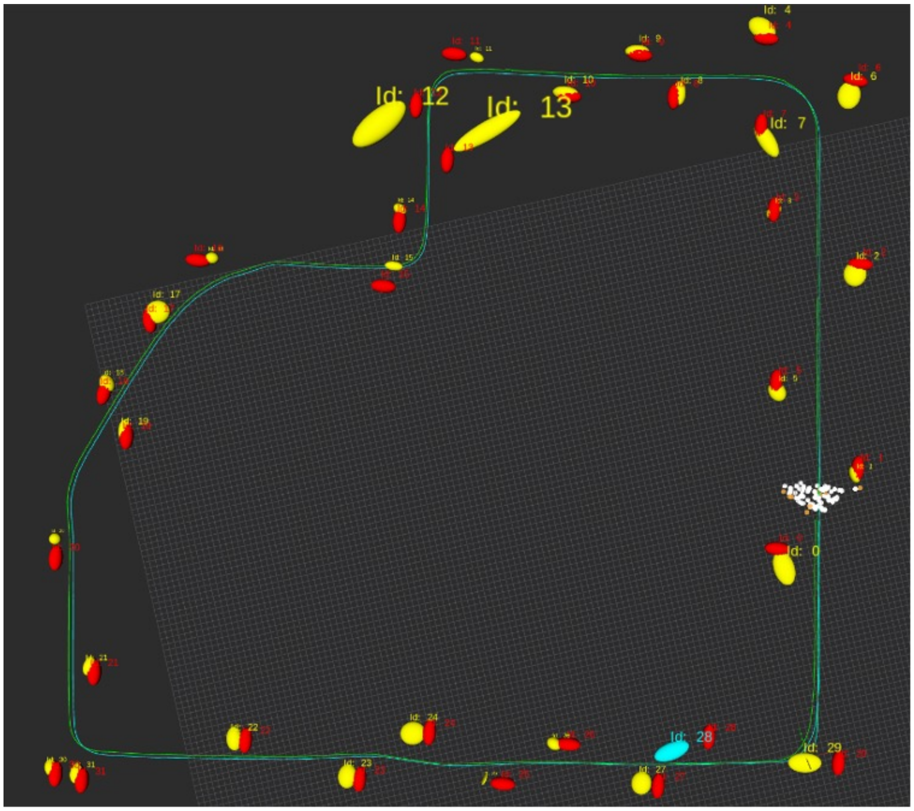

Figure 4.13. OrcVIO on simulation dataset. The red ellipsoids are the groundtruth objects. The yellow ellipsoids are the estimated objects. The cyan ellipsoids are the estimated outlier objects. The cyan line is the groundtruth trajectory. The green line is the estimated trajectory.

to pure rotation maneuvers at planned path corners executed by the quadrotor controller.

The 3D IoU of the object estimates is 0.49. The Precision and Recall of the object reconstruction is shown in Table 4.3. Despite that doors and barriers have a thin structure, causing even small pose estimation drift to reduce the overlap with the ground-truth object instances, OrcVIO is able to produce an accurate object map with good 3D IoU.

4.5.4 Ellipsoid Object on Real Dataset

We also evaluated OrcVIO using real data collected with two commercial VIO sensors on UCSD's campus. The results are qualitative due to the lack of ground truth information.

First, we used Intel RealSense D435i with image frequency of 30 Hz, image resolution of 640×480, IMU frequency of 200 Hz in an indoor lab scene with chairs and monitors as shown

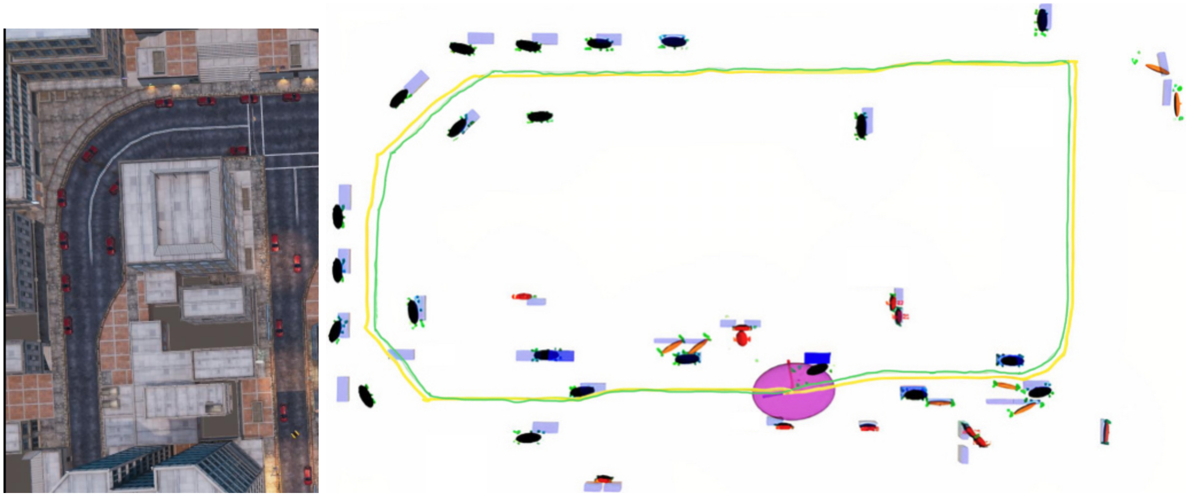

Figure 4.14. Left: Urban simulation in the Unity engine with car, door, and barrier object instances. Right: Object map reconstructed by OrcVIO on the Unity dataset. The figure shows the ground-truth trajectory (green curve), estimated trajectory (yellow curve), the covariance of the last pose (purple ellipsoid), the ground-truth objects (blue cuboids), estimated object ellipsoids (cars are black, doors are red, barriers are orange), as well as the reconstructed semantic landmarks (green dots).

Table 4.3. Precision-Recall Evaluation on the Unity Dataset

Translation error →	≤ 0.5 m		≤ 1.0 m		≤ 1.5 m	
Rotation error	Precision	Recall	Precision	Recall	Precision	Recall
≤ 30°	0.05	0.05	0.18	0.20	0.22	0.25
≤ 45°	0.09	0.10	0.27	0.30	0.45	0.50
–	0.14	0.15	0.41	0.45	0.64	0.70

in Fig. 4.15. The estimated sensor trajectory and reconstructed object map by OrcVIO are shown in the figure. The results demonstrate that OrcVIO can map object instances from different categories and operates at real-time speed in a cluttered indoor scene. Since OrcVIO does not currently have a loop-closing mechanism for object re-identification, objects reappearing after getting lost will be mapped twice. Thus, there are more reconstructed chairs in Fig. 4.15 than in the reality.

Semantic keypoint detection is challenging due to occlusion, viewpoint change, and the lack of distinctive features on the monitors as shown in Fig. 4.15 (b). To handle the monitor class succesfully, we decreased the weight of the semantic keypoint residual in the object Levenberg-

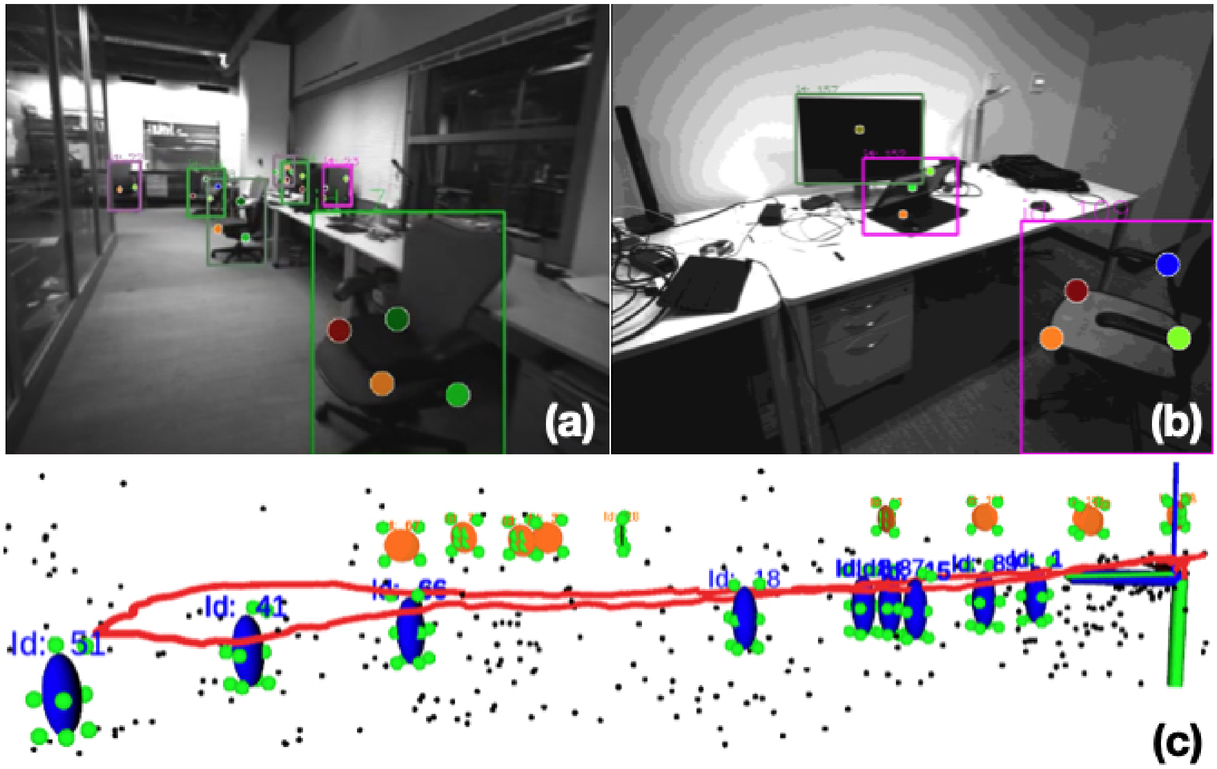

Figure 4.15. OrcVIO localization and object mapping in an indoor scene with chairs and monitors. Bounding-box and semantic-keypoint detections are shown in (a) and (b). The estimated sensor trajectory (red curve), geometric landmarks (black dots), semantic landmarks (green dots), and object ellipsoids (blue for chairs, orange for monitors) obtained by OrcVIO are shown in (c).

Marquardt optimization. Although removing the reliance on semantic keypoints leads to worse object orientation estimation, it allows OrcVIO to work with bounding-box detections only. This simplifies the front-end to an object detector and tracker and makes the algorithm more efficient and easier to deploy on resource-constrained robots.

Finally, we used an INDEMIND Binocular Visual-Inertial Camera to run OrcVIO outdoors with images at 25 Hz with resolution of 640×400 and IMU measurements at 200 Hz. The sensor initially observes bikes and chairs, and then makes a transition into a parking structure, as shown in Fig. 4.16 (b), (c). The resulting object map is shown in Fig. 4.16 (a), demonstrating that OrcVIO is able to estimate object states from different categories in both indoor and outdoor scenes. This experiment also shows that OrcVIO can handle large illumination changes,

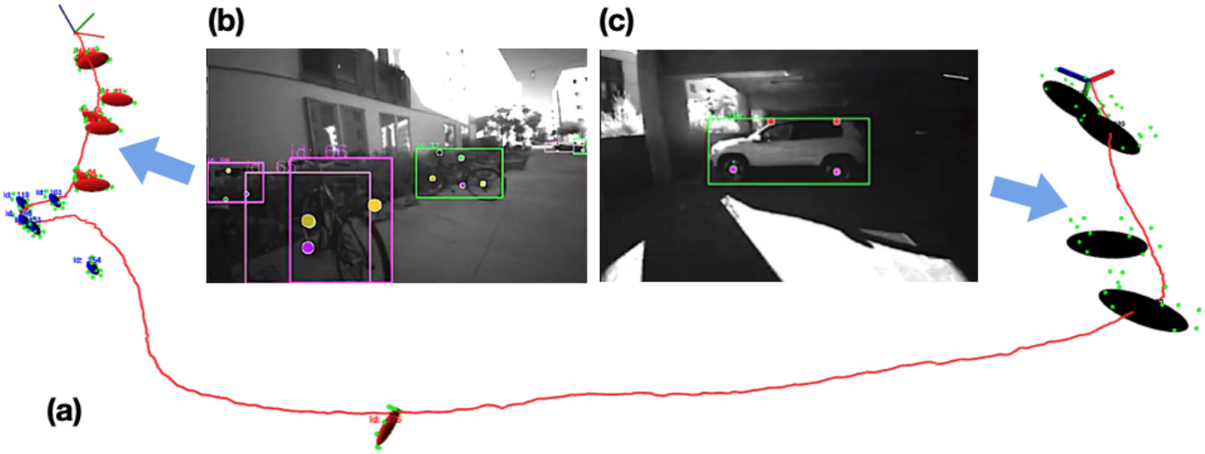

Figure 4.16. OrcVIO localization and object mapping in an outdoor scene with chairs, bikes and cars. The estimated sensor trajectory (red curve), semantic landmarks (green dots), and object ellipsoids (blue for chairs, red for bikes, and black for cars) are shown in (a). Bounding-box and semantic-keypoint detections are shown for bicycle and car instances in (b) and (c).

transitioning from direct outdoor sunlight to dim lighting inside the parking structure.

In this experiment, we have demonstrated several experiments where multiple object categories are involved during the mapping process. The extension to full object categories in YOLO and StarMap is quite easy as well, since only the categorical mean shapes are required. For any new object class, we only need to obtain the average positions of the 3D semantic keypoints from the dataset annotations.

4.6 Summary

This chapter introduces the technique of estimating the object pose and shape from the semantic observations extracted from the 2D RGB image and the coupling with the camera pose estimation. We first develop the object mesh model and use the differentiable semantic keypoint and segmentation mask projection models to optimize the object pose and shape as well as the camera pose. We then present an Object residual constrained Visual Inertial Odometry (OrcVIO) algorithm that model the objects as ellipsoids. It tightly couple 3D object pose and shape estimation with online sensor localization. OrcVIO initializes and optimizes the pose and shape of detected and tracked objects by differentiating through two new optimization terms capturing object-feature and bounding-box measurements. The estimated object poses and shapes aid in real-time incremental multi-state constraint Kalman filtering (MSCKF) over the visual-inertial sensor states.

Acknowledgements

Chapter 4, in part, is a reprint of the material as it appears in Q. Feng, Y. Meng, M. Shan and N. Atanasov, ”Localization and Mapping using Instance-specific Mesh Models,” IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4985-4991, 2019, and in M. Shan, Q. Feng and N. Atanasov, ”OrcVIO: Object residual constrained Visual-Inertial Odometry,” IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5104-5111, 2020. Chapter 4, in part, is currently being prepared for submission for publication of the material as it may appear in M. Shan, Q. Feng, J. Li, A. Yin, V. Dhiman and N. Atanasov, ”OrcVIO: Object residual constrained Visual Inertial Odometry.” The dissertation author was the primary author of these papers.

Chapter 5

Conclusions and Future Work

Semantic reconstruction and understanding is undoubtedly an important task, which can expand the robot's potential to execute numerous applications. The semantic map should be accurate, meaningful as well as efficient, scalable to fit the need of mobile robots. This dissertation discusses the methods of dense semantic map and object-level semantic maps that boost the robot's understanding of the environment.

In Chapter 2, we develop TerrainMesh, a metric-semantic mesh that reconstruct the terrain from the aerial RGB images and sparse depth measurements. The 2-step, initialization and refinement strategy is proposed for mesh reconstruction and we develop joint 2D-3D feature extractor and 2D-3D geometric-semantic loss function for the learning-based reconstruction algorithm. TerrainMesh is an effcient and compact dense semantic mapping algorithm that has the potential to enhance the aerial UAV ability of the environment monitoring.

In Chapter 3, we develop the object-level semantic map using 3D measurements. We introduce CORSAIR, a fully Convolutional Object Retrieval and Symmetry-AIded Registration algorithm. We investigate a 3D sparse fully convolutional neural network to extract both global and local features for the object point cloud. The global features are used for retrieving the similar shape from the object database, and the local features are used to generate correspondence for pose registration with the help of object symmetry. We also introduce ELLIPSDF, a bi-level object model including a coarse level of ellipsoid and a detailed level of continuous signed distance function. We develop the initialization and optimization method for object pose and shape recovery. The detailed 3D object shape can improve the fidelity of the 3D environment modeling comparing to the raw 3D measurements and more information like object semantic category and pose are included.

In Chapter 4, we develop the object-level semantic map from 2D images. We introduce the object mesh model and its observations model that generates semantic instance mask and semantic keypoints. We optimize both the object states and the camera poses from the observation model. We also introduce OrcVIO, Object residual constrained Visual Inertial Odometry. The object is modeled as an ellipsoid with keypoints and the observation model of the semantic object bounding box and semantic keypoint is developed. We use the Multi-State Constraint Kalman Filter (MSCKF) framework to implement the online tightly-coupled object states and camera poses estimation. This

There are several directions to further improve the semantic mapping works in the dissertation, which are summarized below.

- **Multi-agent collaborative semantic mapping:** In this dissertation we develop the semantic mapping method running on a single agent. When the space is large, it is common to have a team of heterogeneous robot to collaborate for exploration and mapping [213, 181]. The multi-agent semantic mapping tackles the challenges of efficient communication of semantic information exchange and global consistency between different agents' observations.

- **Hierarchical semantic mapping:** We discuss the dense mesh semantic map and object-level semantic map. These two forms of semantic map can be combined with meaningful hierarchical structure. The hierarchical semantic map [152, 203] embeds more relational information between different objects at different abstract levels which can be useful for robotics task specification and decomposition.

- **Different form of 3D representations for semantic reconstruction:** We discuss the implicit surface representation of SDF in the dissertation. Another popular implicit 3D representation is Neural Radiance Field (NeRF) [120] that encode the color and density field of the 3D scene through the neural network. Extensions of NeRF can include the semantic labels as well [209, 210, 107]. The NeRF representation have the potential to model the environment in great details.

- **Semantic map with large language model (LLM):** Since the release of ChatGPT [136] by OpenAI, the large language model (LLM) such as GPT series [147, 137] has been applied to every new domain of task because its astonishing ability to provide useful semantic information and to generalize to all different areas. Robotics researchers apply the LLM for tasks like manipulation and planning and search [82, 148, 17]. LLM is capable of inferring semantic relationships between categories and objects, and some of the LLM models can ground the semantic concept in the visual images to connect the language instruction with the visual observations. This provides valuable support for the semantic mapping and understanding for the robots.

With the better semantic scene understanding and reconstruction technique, the robots will get closer to be the more powerful assistants in all aspects of the life.

Bibliography

[1] Oladapo Afolabi, Allen Y. Yang, and S. Shankar Sastry. Extending DeepSDF for automatic 3D shape retrieval and similarity transform estimation. *arXiv:2004.09048*, 2020.

[2] Marc Vigo Anglada. An improved incremental algorithm for constructing restricted Delaunay triangulations. *Computers & Graphics*, 21(2):215–223, 1997.

[3] Arash Asgharivaskasi and Nikolay Atanasov. Semantic OcTree Mapping and Shannon Mutual Information Computation for Robot Exploration. *IEEE Transactions on Robotics*, 39(3):1910–1928, 2023.

[4] Nikolay Atanasov, Sean L. Bowman, Kostas Daniilidis, and George J. Pappas. A Unifying View of Geometry, Semantics, and Data Association in SLAM. In *International Joint Conference on Artificial Intelligence*, pages 5204–5208, 2018.

[5] Nikolay Atanasov, Menglong Zhu, Kostas Daniilidis, and George J. Pappas. Localization from semantic observations via the matrix permanent. *The International Journal of Robotics Research*, 35(1-3):73–99, 2016.

[6] Armen Avetisyan, Manuel Dahnert, Angela Dai, Manolis Savva, Angel X. Chang, and Matthias Nießner. Scan2CAD: Learning CAD Model Alignment in RGB-D Scans. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2609–2618, 2019.

[7] Sid Yingze Bao and Silvio Savarese. Semantic Structure from Motion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2025–2032, 2011.

[8] H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf. Parametric Correspondence and Chamfer Matching: Two New Techniques for Image Matching. In *International Joint Conference on Artificial Intelligence*, page 659–663, 1977.

[9] Matthew Berger, Andrea Tagliasacchi, Lee M. Seversky, Pierre Alliez, Gaël Guennebaud, Joshua A. Levine, Andrei Sharf, and Claudio T. Silva. A Survey of Surface Reconstruction from Point Clouds. *Computer Graphics Forum*, 36(1):301–329, 2017.

[10] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple Online and Realtime Tracking. In *IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468, 2016.

[11] Michael Bloesch, Tristan Laidlow, Ronald Clark, Stefan Leutenegger, and Andrew Davison. Learning Meshes for Dense Visual SLAM. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5854–5863, 2019.

[12] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv:2004.10934*, 2020.

[13] Piotr Bojanowski, Armand Joulin, David Lopez-Pas, and Arthur Szlam. Optimizing the Latent Space of Generative Networks. In *International Conference on Machine Learning*, pages 600–609, 2018.

[14] Sean L. Bowman, Nikolay Atanasov, Kostas Daniilidis, and George J. Pappas. Probabilistic Data Association for Semantic SLAM. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1722–1729, 2017.

[15] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6D Object Pose Estimation Using 3D Object Coordinates. In *Computer Vision – ECCV*, 2014.

[16] John Bridle. Training Stochastic Model Recognition Algorithms as Networks can Lead to Maximum Mutual Information Estimation of Parameters. In *Advances in Neural Information Processing Systems*, 1989.

[17] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. In *Conference on Robot Learning*, 2023.

[18] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J. Leonard. Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[19] Carlos Campos, Richard Elvira, Juan J. Gómez Rodríguez, José M. M. Montiel, and Juan D. Tardós. ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021.

[20] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi,

and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. *arXiv:1512.03012*, 2015.

[21] Dengsheng Chen, Jun Li, Zheng Wang, and Kai Xu. Learning Canonical Shape Space for Category-Level 6D Object Pose and Size Estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11970–11979, 2020.

[22] Liang-Chieh Chen, Jonathan T. Barron, George Papandreou, Kevin Murphy, and Alan L. Yuille. Semantic Image Segmentation with Task-Specific Edge Detection Using CNNs and a Discriminatively Trained Domain Transform. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4545–4554, 2016.

[23] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2018.

[24] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking Atrous Convolution for Semantic Image Segmentation. *arXiv:1706.05587*, 2017.

[25] Meida Chen, Qingyong Hu, Zifan Yu, Hugues THOMAS, Andrew Feng, Yu Hou, Kyle McCullough, Fengbo Ren, and Lucio Soibelman. STPLS3D: A Large-Scale Synthetic and Real Aerial Photogrammetry 3D Point Cloud Dataset. In *British Machine Vision Conference*, 2022.

[26] Zhao Chen, Vijay Badrinarayanan, Gilad Drozdov, and Andrew Rabinovich. Estimating Depth from RGB and Sparse Sensing. In *Computer Vision – ECCV*, 2018.

[27] Ian Cherabier, Johannes L. Schönberger, Martin R. Oswald, Marc Pollefeys, and Andreas Geiger. Learning Priors for Semantic 3D Reconstruction. In *Computer Vision – ECCV*, 2018.

[28] Gregory S. Chirikjian. *Stochastic Models, Information Theory, and Lie Groups, Volume 2: Analytic Methods and Modern Applications*, volume 2. Birkhäuser Boston, first edition, 2011.

[29] Changhyun Choi and Henrik I. Christensen. RGB-D object pose estimation in unstructured environments. *Robotics and Autonomous Systems*, 75:595–613, 2016.

[30] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3070–3079, 2019.

[31] Christopher Choy, Jaesik Park, and Vladlen Koltun. Fully Convolutional Geometric Features. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8957–8965, 2019.

[32] Javier Civera, Dorian Gálvez-López, L. Riazuelo, Juan D. Tardós, and J. M. M. Montiel. Towards semantic SLAM using a monocular camera. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1277–1284, 2011.

[33] Manuel Dahnert, Angela Dai, Leonidas Guibas, and Matthias Niessner. Joint Embedding of 3D Scan and CAD Objects. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8748–8757, 2019.

[34] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-Annotated 3D Reconstructions of Indoor Scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2432–2443, 2017.

[35] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. BundleFusion: Real-Time Globally Consistent 3D Reconstruction Using On-the-Fly Surface Reintegration. *ACM Transactions on Graphics*, 36(3), 2017.

[36] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable Convolutional Networks. In *IEEE International Conference on Computer Vision (ICCV)*, pages 764–773, 2017.

[37] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. MonoSLAM: Real-Time Single Camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007.

[38] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009.

[39] Xinke Deng, Arsalan Mousavian, Yu Xiang, Fei Xia, Timothy Bretl, and Dieter Fox. PoseRBPF: A Rao–Blackwellized Particle Filter for 6-D Object Pose Tracking. *IEEE Transactions on Robotics*, 37(5):1328–1342, 2021.

[40] Vikas Dhiman, Quoc-Huy Tran, Jason J. Corso, and Manmohan Chandraker. A Continuous Occlusion Model for Road Scene Understanding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4331–4339, 2016.

[41] Lee R. Dice. Measures of the Amount of Ecologic Association Between Species. *Ecology*, 26(3):297–302, 1945.

[42] Jakob Engel, Thomas Schöps, and Daniel Cremers. LSD-SLAM: Large-Scale Direct Monocular SLAM. In *Computer Vision – ECCV*, 2014.

[43] Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. Learning SO(3) Equivariant Representations with Spherical CNNs. In *Computer Vision – ECCV*, 2018.

[44] Baofu Fang, Gaofei Mei, Xiaohui Yuan, Le Wang, Zaijun Wang, and Junyang Wang. Visual SLAM for robot navigation in healthcare facility. *Pattern Recognition*, 113:107822, 2021.

[45] Xiaohan Fei and Stefano Soatto. Visual-Inertial Object Detection and Mapping. In *Computer Vision – ECCV*, 2018.

[46] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object Detection with Discriminatively Trained Part-Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[47] Qiaojun Feng and Nikolay Atanasov. Fully Convolutional Geometric Features for Category-level Object Alignment. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8492–8498, 2020.

[48] Qiaojun Feng and Nikolay Atanasov. Mesh Reconstruction from Aerial Images for Outdoor Terrain Mapping Using Joint 2D-3D Learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5208–5214, 2021.

[49] Qiaojun Feng and Nikolay Atanasov. TerrainMesh: Metric-Semantic Terrain Reconstruction from Aerial Images Using Joint 2D-3D Learning. *arXiv:2204.10993*, 2022.

[50] Qiaojun Feng, Yue Meng, Mo Shan, and Nikolay Atanasov. Localization and Mapping using Instance-specific Mesh Models. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4985–4991, 2019.

[51] Martin A. Fischler and Robert C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. In *Readings in Computer Vision*, page 726–740. Morgan Kaufmann, 1987.

[52] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. On-Manifold Preintegration for Real-Time Visual–Inertial Odometry. *IEEE Transactions on Robotics*, 33(1):1–21, 2017.

[53] Duncan Frost, Victor Prisacariu, and David Murray. Recovering Stable Scale in Monocular SLAM Using Object-Supplemented Bundle Adjustment. *IEEE Transactions on Robotics*, 34(3):736–747, 2018.

[54] Yasutaka Furukawa and Carlos Hernández. Multi-View Stereo: A Tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 9(1-2):1–148, 2015.

[55] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *International Conference on Machine Learning*, pages 1050–1059, 2016.

[56] C. Galindo, A. Saffiotti, S. Coradeschi, P. Buschka, J.A. Fernandez-Madrigal, and J. Gonzalez. Multi-Hierarchical Semantic Maps for Mobile Robotics. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2278–2283, 2005.

[57] Sourav Garg, Niko Sünderhauf, Feras Dayoub, Douglas Morrison, Akansel Cosgun, Gustavo Carneiro, Qi Wu, Tat-Jun Chin, Ian Reid, Stephen Gould, Peter Corke, and Michael Milford. Semantics for Robotic Mapping, Perception and Interaction: A Survey. *Foundations and Trends® in Robotics*, 8(1–2):1–224, 2020.

[58] Paul Gay, James Stuart, and Alessio Del Bue. Visual Graphs from Motion (VGfM): Scene Understanding with Object Geometry Reasoning. In *Computer Vision – ACCV*, 2018.

[59] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[60] Patrick Geneva, Kevin Eckenhoff, Woosik Lee, Yulin Yang, and Guoquan Huang. OpenVINS: A Research Platform for Visual-Inertial Estimation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4666–4672, 2020.

[61] Georgia Gkioxari, Justin Johnson, and Jitendra Malik. Mesh R-CNN. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9784–9794, 2019.

[62] Zan Gojcic, Caifa Zhou, Jan D. Wegner, and Andreas Wieser. The Perfect Match: 3D Point Cloud Matching With Smoothed Densities. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5540–5549, 2019.

[63] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[64] Alexander Grabner, Peter M. Roth, and Vincent Lepetit. 3D Pose Estimation and 3D Model Retrieval for Objects in the Wild. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3022–3031, 2018.

[65] W. Nicholas Greene and Nicholas Roy. FLaME: Fast Lightweight Mesh Estimation Using Variational Smoothing on Delaunay Graphs. In *IEEE International Conference on Computer Vision (ICCV)*, pages 4696–4704, 2017.

[66] Margarita Grinvald, Fadri Furrer, Tonci Novkovic, Jen Jen Chung, Cesar Cadena, Roland Siegwart, and Juan Nieto. Volumetric Instance-Aware Semantic Mapping and 3D Object Discovery. *IEEE Robotics and Automation Letters*, 4(3):3037–3044, 2019.

[67] Haoyu Guo, Sida Peng, Haotong Lin, Qianqian Wang, Guofeng Zhang, Hujun Bao, and Xiaowei Zhou. Neural 3D Scene Reconstruction with the Manhattan-world Assumption. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5501–5510, 2022.

[68] Dorian Gálvez-López, Marta Salas, Juan D. Tardós, and J.M.M. Montiel. Real-time monocular object SLAM. *Robotics and Autonomous Systems*, 75:435–449, 2016.

[69] Zekun Hao, Hadar Averbuch-Elor, Noah Snavely, and Serge Belongie. DualSDF: Semantic Shape Manipulation Using a Two-Level Representation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7628–7638, 2020.

[70] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[71] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):386–397, 2020.

[72] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[73] Xinwei He, Yang Zhou, Zhichao Zhou, Song Bai, and Xiang Bai. Triplet-Center Loss for Multi-view 3D Object Retrieval. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1945–1954, 2018.

[74] Joel A. Hesch and Stergios I. Roumeliotis. A Direct Least-Squares (DLS) method for PnP. In *International Conference on Computer Vision (ICCV)*, pages 383–390, 2011.

[75] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes. In *Computer Vision – ACCV*, 2012.

[76] Tomáš Hodan, Pavel Haluza, Štepán Obdržálek, Jirí Matas, Manolis Lourakis, and Xenophon Zabulis. T-LESS: An RGB-D Dataset for 6D Pose Estimation of Texture-Less Objects. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 880–888, 2017.

[77] Mehdi Hosseinzadeh, Yasir Latif, Trung Pham, Niko Suenderhauf, and Ian Reid. Structure Aware SLAM Using Quadrics and Planes. In *Computer Vision – ACCV*, 2018.

[78] Mehdi Hosseinzadeh, Kejie Li, Yasir Latif, and Ian Reid. Real-Time Monocular Object-Model Aware Sparse SLAM. In *International Conference on Robotics and Automation (ICRA)*, pages 7123–7129, 2019.

[79] Qingyong Hu, Bo Yang, Sheikh Khalid, Wen Xiao, Niki Trigoni, and Andrew Markham. SensatUrban: Learning Semantics from Urban-Scale Photogrammetric Point Clouds. *International Journal of Computer Vision*, 130(2):316–343, 2022.

[80] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11105–11114, 2020.

[81] Zeyu Hu, Xuyang Bai, Jiaxiang Shang, Runze Zhang, Jiayu Dong, Xin Wang, Guangyuan Sun, Hongbo Fu, and Chiew-Lan Tai. Voxel-Mesh Network for Geodesic-Aware 3D Semantic Segmentation of Indoor Scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–12, 2022.

[82] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. VoxPoser: Composable 3D Value Maps for Robotic Manipulation with Language Models. In *Conference on Robot Learning*, 2023.

[83] Peter J. Huber. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964.

[84] John F. Hughes, Andries van Dam, Morgan McGuire, David F. Sklar, James D. Foley, Steven K. Feiner, and Kurt Akeley. *Computer Graphics: Principles and Practice*. Addison-Wesley Professional, third edition, 2013.

[85] Christian Häne, Christopher Zach, Andrea Cohen, and Marc Pollefeys. Dense Semantic 3D Reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(9):1730–1743, 2017.

[86] Vladislav Ishimtsev, Alexey Bokhovkin, Alexey Artemov, Savva Ignatyev, Matthias Niessner, Denis Zorin, and Evgeny Burnaev. CAD-Deform: Deformable Fitting of CAD Models to 3D Scans. In *Computer Vision – ECCV*, 2020.

[87] Paul Jaccard. The Distribution Of The Flora In The Alpine Zone. *New Phytologist*, 11(2):37–50, 1912.

[88] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433–449, 1999.

[89] Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A*, 32(5):922–923, 1976.

[90] Michael Kaess. Simultaneous localization and mapping with infinite planes. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4605–4611, 2015.

[91] Michael Kaess and Frank Dellaert. Covariance recovery from a square root information matrix for data association. *Robotics and Autonomous Systems*, 57(12):1198–1210, 2009.

[92] Angjoo Kanazawa, Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. Learning Category-Specific Mesh Reconstruction from Image Collections. In *Computer Vision – ECCV*, 2018.

[93] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3D Mesh Renderer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3907–3916, 2018.

[94] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson Surface Reconstruction. In *Eurographics Symposium on Geometry Processing*, page 61–70, 2006.

[95] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2014.

[96] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

[97] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[98] Ioannis Kostavelis and Antonios Gasteratos. Semantic mapping for mobile robotics tasks: A survey. *Robotics and Autonomous Systems*, 66:86–103, 2015.

[99] Dimitrios G. Kottas, Kejian J. Wu, and Stergios I. Roumeliotis. Detecting and dealing with hovering maneuvers in vision-aided inertial navigation systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3172–3179, 2013.

[100] Abhijit Kundu, Yin Li, Frank Dellaert, Fuxin Li, and James M. Rehg. Joint Semantic Segmentation and 3D Reconstruction from Monocular Video. In *Computer Vision – ECCV*, 2014.

[101] Patrick Labatut, Jean-Philippe Pons, and Renaud Keriven. Robust and Efficient Surface Reconstruction From Range Data. *Computer Graphics Forum*, 28(8):2275–2290, 2009.

[102] Bastian Leibe, Nico Cornelis, Kurt Cornelis, and Luc Van Gool. Dynamic 3D Scene Analysis from a Moving Vehicle. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2007.

[103] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334, 2015.

[104] Kejie Li, Hamid Rezatofighi, and Ian Reid. MOLTR: Multiple Object Localization, Tracking and Reconstruction From Monocular RGB Videos. *IEEE Robotics and Automation Letters*, 6(2):3341–3348, 2021.

[105] Peiliang Li, Tong Qin, and Shaojie Shen. Stereo Vision-Based Semantic 3D Object and Ego-Motion Tracking for Autonomous Driving. In *Computer Vision – ECCV*, 2018.

[106] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal Loss for Dense Object Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):318–327, 2020.

[107] Fangfu Liu, Chubin Zhang, Yu Zheng, and Yueqi Duan. Semantic Ray: Learning a Generalizable Semantic Field with Cross-Reprojection Attention. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17386–17396, 2023.

[108] Jin Liu and Shunping Ji. A Novel Recurrent Encoder-Decoder Structure for Large-Scale Multi-View Stereo Reconstruction From an Open Aerial Dataset. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6049–6058, 2020.

[109] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. A General Differentiable Mesh Renderer for Image-Based 3D Reasoning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(1):50–62, 2022.

[110] William E. Lorensen and Harvey E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *ACM SIGGRAPH Conference Proceedings*, 21(4):163–169, 1987.

[111] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):1573–1405, 1994.

[112] Bruce D. Lucas and Takeo Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *International Joint Conference on Artificial Intelligence*, page 674–679, 1981.

[113] Ren C. Luo and Michael Chiou. Hierarchical Semantic Mapping Using Convolutional Neural Networks for Intelligent Service Robotics. *IEEE Access*, 6:61287–61294, 2018.

[114] Fangchang Ma, Guilherme Venturelli Cavalheiro, and Sertac Karaman. Self-Supervised Sparse-to-Dense: Self-Supervised Depth Completion from LiDAR and Monocular Camera. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3288–3295, 2019.

[115] Fangchang Ma and Sertac Karaman. Sparse-to-Dense: Depth Prediction from Sparse Depth Samples and a Single Image. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4796–4803, 2018.

[116] Yi Ma, Stefano Soatto, Jana Košecká, and S. Shankar Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer New York, first edition, 2004.

[117] Kevis-Kokitsi Maninis, Stefan Popov, Matthias Nießner, and Vittorio Ferrari. Vid2CAD: CAD Model Alignment Using Multi-View Constraints From Videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):1320–1327, 2023.

[118] Mapillary. OpenSfM. https://github.com/mapillary/OpenSfM.

[119] Shigemichi Matsuzaki, Hiroaki Masuzawa, Jun Miura, and Shuji Oishi. 3D Semantic Mapping in Greenhouses for Agricultural Mobile Robots with Robust Object Recognition Using Robots' Trajectory. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 357–362, 2018.

[120] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *Computer Vision – ECCV*, 2020.

[121] Anastasios I. Mourikis and Stergios I. Roumeliotis. A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3565–3572, 2007.

[122] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

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

[124] Andriy Myronenko and Xubo Song. Point Set Registration: Coherent Point Drift. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(12):2262–2275, 2010.

[125] José Neira and Juan D. Tardós. Data Association in Stochastic Mapping Using the Joint Compatibility Test. *IEEE Transactions on Robotics and Automation*, 17(6):890–897, 2001.

[126] Richard A. Newcombe, Dieter Fox, and Steven M. Seitz. DynamicFusion: Reconstruction and tracking of non-rigid scenes in real-time. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 343–352, 2015.

[127] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136, 2011.

[128] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked Hourglass Networks for Human Pose Estimation. In *Computer Vision – ECCV*, 2016.

[129] Lachlan Nicholson, Michael Milford, and Niko Sünderhauf. QuadricSLAM: Dual Quadrics From Object Detections as Landmarks in Object-Oriented SLAM. *IEEE Robotics and Automation Letters*, 4(1):1–8, 2019.

[130] Feiping Nie, Heng Huang, Xiao Cai, and Chris Ding. Efficient and Robust Feature Selection via Joint $\ell_{2,1}$-Norms Minimization. In *Advances in Neural Information Processing Systems*, 2010.

[131] Yinyu Nie, Xiaoguang Han, Shihui Guo, Yujian Zheng, Jian Chang, and Jian Jun Zhang. Total3DUnderstanding: Joint Layout, Object Pose and Mesh Reconstruction for Indoor Scenes From a Single Image. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 52–61, 2020.

[132] Jean Oh, Martial Hebert, Hae-Gon Jeon, Xavier Perez, Chia Dai, and Yeeho Song. Explainable Semantic Mapping for First Responders. In *NeurIPS 2019 Artificial Intelligence for Humanitarian Assistance and Disaster Response Workshop*, 2019.

[133] Kyel Ok, Katherine Liu, Kris Frey, Jonathan P. How, and Nicholas Roy. Robust Object-based SLAM for High-speed Autonomous Navigation. In *International Conference on Robotics and Automation (ICRA)*, pages 669–675, 2019.

[134] Kyel Ok, Katherine Liu, and Nicholas Roy. Hierarchical Object Map Estimation for Efficient and Robust Navigation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1132–1139, 2021.

[135] Helen Oleynikova, Zachary Taylor, Marius Fehr, Roland Siegwart, and Juan Nieto. Voxblox: Incremental 3D Euclidean Signed Distance Fields for on-board MAV planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1366–1373, 2017.

[136] OpenAI. ChatGPT. https://chat.openai.com/.

[137] OpenAI. GPT-4 Technical Report. *arXiv:2303.08774*, 2023.

[138] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 165–174, 2019.

[139] Georgios Pavlakos, Xiaowei Zhou, Aaron Chan, Konstantinos G. Derpanis, and Kostas Daniilidis. 6-DoF object pose from semantic keypoints. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2011–2018, 2017.

[140] David Paz, Hengyuan Zhang, Qinru Li, Hao Xiang, and Henrik I. Christensen. Probabilistic Semantic Mapping for Urban Autonomous Driving Applications. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2059–2064, 2020.

[141] Sudeep Pillai and John Leonard. Monocular SLAM Supported Object Recognition. In *Robotics: Science and Systems*, 2015.

[142] Xavier Puig, Eric Undersander, Andrew Szot, Mikael Dallaire Cote, Tsung-Yen Yang, Ruslan Partsey, Ruta Desai, Alexander William Clegg, Michal Hlavac, So Yeon Min, Vladimír Vondruš, Theophile Gervet, Vincent-Pierre Berges, John M. Turner, Oleksandr Maksymets, Zsolt Kira, Mrinal Kalakrishnan, Jitendra Malik, Devendra Singh Chaplot, Unnat Jain, Dhruv Batra, Akshara Rai, and Roozbeh Mottaghi. Habitat 3.0: A Co-Habitat for Humans, Avatars and Robots. *arXiv:2310.13724*, 2023.

[143] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017.

[144] Tong Qin, Peiliang Li, and Shaojie Shen. VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018.

[145] Tong Qin, Yuxin Zheng, Tongqing Chen, Yilun Chen, and Qing Su. A Light-Weight Semantic Map for Visual Localization towards Autonomous Driving. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 11248–11254, 2021.

[146] Inigo Quilez. Ellipsoid SDF. https://iquilezles.org/articles/ellipsoids/.

[147] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving Language Understanding by Generative Pre-Training, 2018.

[148] Krishan Rana, Jesse Haviland, Sourav Garg, Jad Abou-Chakra, Ian Reid, and Niko Suenderhauf. SayPlan: Grounding Large Language Models using 3D Scene Graphs for Scalable Robot Task Planning. In *Conference on Robot Learning*, 2023.

[149] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3D Deep Learning with PyTorch3D. *arXiv:2007.08501*, 2020.

[150] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[151] I.D. Reid and J.M. Brady. Recognition of object classes from range data. *Artificial Intelligence*, 78(1):289–326, 1995.

[152] Antoni Rosinol, Andrew Violette, Marcus Abate, Nathan Hughes, Yun Chang, Jingnan Shi, Arjun Gupta, and Luca Carlone. Kimera: From SLAM to spatial perception with 3D dynamic scene graphs. *The International Journal of Robotics Research*, 40(12-14):1510–1546, 2021.

[153] Edward Rosten, Reid Porter, and Tom Drummond. Faster and Better: A Machine Learning Approach to Corner Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1):105–119, 2010.

[154] Tianshu Ruan, Hao Wang, Rustam Stolkin, and Manolis Chiou. A Taxonomy of Semantic Information in Robot-Assisted Disaster Response. In *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 285–292, 2022.

[155] Cosimo Rubino, Marco Crocco, and Alessio Del Bue. 3D Object Localisation from Multi-View Image Detections. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(6):1281–1294, 2018.

[156] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2564–2571, 2011.

[157] J.R. Ruiz-Sarmiento, C. Galindo, and J. Gonzalez-Jimenez. Robot@Home, a robotic dataset for semantic mapping of home environments. *The International Journal of Robotics Research*, 36(2):131–141, 2017.

[158] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast Point Feature Histograms (FPFH) for 3D registration. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3212–3217, 2009.

[159] Martin Rünz, Kejie Li, Meng Tang, Lingni Ma, Chen Kong, Tanner Schmidt, Ian Reid, Lourdes Agapito, Julian Straub, Steven Lovegrove, and Richard Newcombe. FroDO: From Detections to 3D Objects. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14708–14717, 2020.

[160] Inkyu Sa, Marija Popović, Raghav Khanna, Zetao Chen, Philipp Lottes, Frank Liebisch, Juan Nieto, Cyrill Stachniss, Achim Walter, and Roland Siegwart. WeedMap: A Large-Scale Semantic Weed Mapping Framework Using Aerial Multispectral Imaging and Deep Neural Network for Precision Farming. *Remote Sensing*, 10(9), 2018.

[161] Renato F. Salas-Moreno, Richard A. Newcombe, Hauke Strasdat, Paul H.J. Kelly, and Andrew J. Davison. SLAM++: Simultaneous Localisation and Mapping at the Level of Objects. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1352–1359, 2013.

[162] Samuele Salti, Federico Tombari, and Luigi Di Stefano. SHOT: Unique signatures of histograms for surface and texture description. *Computer Vision and Image Understanding*, 125:251–264, 2014.

[163] Silvio Savarese and Li Fei-Fei. 3D generic object categorization, localization and pose estimation. In *IEEE International Conference on Computer Vision (ICCV)*, 2007.

[164] Mohamed Sayed, John Gibson, Jamie Watson, Victor Prisacariu, Michael Firman, and Clément Godard. SimpleRecon: 3D Reconstruction Without 3D Convolutions. In *Computer Vision – ECCV*, 2022.

[165] Johannes L. Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise View Selection for Unstructured Multi-View Stereo. In *Computer Vision – ECCV*, 2016.

[166] Mo Shan, Qiaojun Feng, and Nikolay Atanasov. OrcVIO: Object residual constrained Visual-Inertial Odometry. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5104–5111, 2020.

[167] Mo Shan, Qiaojun Feng, You-Yi Jau, and Nikolay Atanasov. ELLIPSDF: Joint Object Pose and Shape Optimization with a Bi-level Ellipsoid and Signed Distance Function Description. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5926–5935, 2021.

[168] Tixiao Shan, Brendan Englot, Drew Meyers, Wei Wang, Carlo Ratti, and Daniela Rus. LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5135–5142, 2020.

[169] Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):640–651, 2017.

[170] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. PointRCNN: 3D Object Proposal Generation and Detection From Point Cloud. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–779, 2019.

[171] Alessandro Simoni, Stefano Pini, Roberto Vezzani, and Rita Cucchiara. Multi-Category Mesh Reconstruction From Image Collections. In *International Conference on 3D Vision (3DV)*, pages 1321–1330, 2021.

[172] Olga Sorkine. Differential Representations for Mesh Processing. *Computer Graphics Forum*, 25(4):789–807, 2006.

[173] Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, and Hans-Peter Seidel. Laplacian Surface Editing. In *Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, page 175–184, 2004.

[174] Noah Stier, Alexander Rich, Pradeep Sen, and Tobias Höllerer. VoRTX: Volumetric 3D Reconstruction With Transformers for Voxelwise View Selection and Fusion. In *International Conference on 3D Vision (3DV)*, pages 320–330, 2021.

[175] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 573–580, 2012.

[176] Edgar Sucar, Kentaro Wada, and Andrew Davison. NodeSLAM: Neural Object Descriptors for Multi-View Shape Reconstruction. In *International Conference on 3D Vision (3DV)*, pages 949–958, 2020.

[177] Jiaming Sun, Yiming Xie, Linghao Chen, Xiaowei Zhou, and Hujun Bao. NeuralRecon: Real-Time Coherent 3D Reconstruction from Monocular Video. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15593–15602, 2021.

[178] Ke Sun, Kartik Mohta, Bernd Pfrommer, Michael Watterson, Sikang Liu, Yash Mulgaonkar, Camillo J. Taylor, and Vijay Kumar. Robust Stereo Visual Inertial Odometry for Fast Autonomous Flight. *IEEE Robotics and Automation Letters*, 3(2):965–972, 2018.

[179] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David Mcallister, Justin Kerr, and Angjoo Kanazawa. Nerfstudio: A Modular Framework for Neural Radiance Field Development. In *ACM SIGGRAPH Conference Proceedings*, 2023.

[180] Marvin Teichmann, Michael Weber, Marius Zöllner, Roberto Cipolla, and Raquel Urtasun. MultiNet: Real-time Joint Semantic Reasoning for Autonomous Driving. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 1013–1020, 2018.

[181] Yulun Tian, Yun Chang, Fernando Herrera Arias, Carlos Nieto-Granda, Jonathan P. How, and Luca Carlone. Kimera-Multi: Robust, Distributed, Dense Metric-Semantic SLAM for Multi-Robot Systems. *IEEE Transactions on Robotics*, 38(4):2022–2038, 2022.

[182] S. Umeyama. Parameterized Point Pattern Matching and Its Application to Recognition of Object Families. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(2):136–144, 1993.

[183] Unity. Unity Real-Time Development Platform — 3D, 2D, VR & AR Engine. https://unity.com/.

[184] Mikaela Angelina Uy, Jingwei Huang, Minhyuk Sung, Tolga Birdal, and Leonidas Guibas. Deformation-Aware 3D Model Embedding and Retrieval. In *Computer Vision – ECCV*, 2020.

[185] Julien P.C. Valentin, Sunando Sengupta, Jonathan Warrell, Ali Shahrokni, and Philip H.S. Torr. Mesh Based Semantic Modelling for Indoor and Outdoor Scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2067–2074, 2013.

[186] Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[187] Vibhav Vineet, Ondrej Miksik, Morten Lidegaard, Matthias Nießner, Stuart Golodetz, Victor A. Prisacariu, Olaf Kähler, David W. Murray, Shahram Izadi, Patrick Pérez, and Philip H. S. Torr. Incremental Dense Semantic Stereo Fusion for Large-Scale Semantic Scene Reconstruction. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 75–82, 2015.

[188] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3338–3347, 2019.

[189] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J. Guibas. Normalized Object Coordinate Space for Category-Level 6D Object Pose and Size Estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2637–2646, 2019.

[190] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images. In *Computer Vision – ECCV*, 2018.

[191] Wei Wang, Yong Zhao, Pengcheng Han, Pengcheng Zhao, and Shuhui Bu. TerrainFusion: Real-time Digital Surface Model Reconstruction based on Monocular SLAM. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7895–7902, 2019.

[192] Zhenzhen Weng and Serena Yeung. Holistic 3D Human and Scene Mesh Estimation from Single View Images. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 334–343, 2021.

[193] Thomas Whelan, Stefan Leutenegger, Renato Salas Moreno, Ben Glocker, and Andrew Davison. ElasticFusion: Dense SLAM Without A Pose Graph. In *Robotics: Science and Systems*, 2015.

[194] Yanmin Wu, Yunzhou Zhang, Delong Zhu, Yonghui Feng, Sonya Coleman, and Dermot Kerr. EAO-SLAM: Monocular Semi-Dense Object SLAM Based on Ensemble Data Association. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4966–4973, 2020.

[195] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

[196] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. Beyond PASCAL: A benchmark for 3D object detection in the wild. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 75–82, 2014.

[197] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. PointFlow: 3D Point Cloud Generation With Continuous Normalizing Flows. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4540–4549, 2019.

[198] Heng Yang, Jingnan Shi, and Luca Carlone. TEASER: Fast and Certifiable Point Cloud Registration. *IEEE Transactions on Robotics*, 37(2):314–333, 2021.

[199] Shichao Yang and Sebastian Scherer. CubeSLAM: Monocular 3-D Object SLAM. *IEEE Transactions on Robotics*, 35(4):925–938, 2019.

[200] Zi Jian Yew and Gim Hee Lee. 3DFeat-Net: Weakly Supervised Local 3D Features for Point Cloud Registration. In *Computer Vision – ECCV*, 2018.

[201] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3D Object Detection and Tracking. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11779–11788, 2021.

[202] Zhiding Yu, Chen Feng, Ming-Yu Liu, and Srikumar Ramalingam. CASENet: Deep Category-Aware Semantic Edge Detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1761–1770, 2017.

[203] Yufeng Yue, Chunyang Zhao, Ruilin Li, Chule Yang, Jun Zhang, Mingxing Wen, Yuanzhe Wang, and Danwei Wang. A Hierarchical Framework for Collaborative Probabilistic Semantic Mapping. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 9659–9665, 2020.

[204] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3DMatch: Learning Local Geometric Descriptors from RGB-D Reconstructions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 199–208, 2017.

[205] Ji Zhang and Sanjiv Singh. LOAM: Lidar Odometry and Mapping in Real-time. In *Robotics: Science and Systems*, 2014.

[206] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. FairMOT: On the Fairness of Detection and Re-identification in Multiple Object Tracking. *International Journal of Computer Vision*, 129(11):3069–3087, 2021.

[207] Zichao Zhang and Davide Scaramuzza. A Tutorial on Quantitative Trajectory Evaluation for Visual(-Inertial) Odometry. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7244–7251, 2018.

[208] Tianyu Zhao, Qiaojun Feng, Sai Jadhav, and Nikolay Atanasov. CORSAIR: Convolutional Object Retrieval and Symmetry-AIded Registration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 47–54, 2021.

[209] Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew J. Davison. In-Place Scene Labelling and Understanding with Implicit Scene Representation. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15818–15827, 2021.

[210] Shuaifeng Zhi, Edgar Sucar, Andre Mouton, Iain Haughton, Tristan Laidlow, and Andrew J. Davison. iLabel: Revealing Objects in Neural Fields. *IEEE Robotics and Automation Letters*, 8(2):832–839, 2023.

[211] Xingyi Zhou, Arjun Karpur, Linjie Luo, and Qixing Huang. StarMap for Category-Agnostic Keypoint and Viewpoint Estimation. In *Computer Vision – ECCV*, 2018.

[212] Yin Zhou and Oncel Tuzel. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4490–4499, 2018.

[213] Ehsan Zobeidi, Alec Koppel, and Nikolay Atanasov. Dense Incremental Metric-Semantic Mapping for Multiagent Systems via Sparse Gaussian Process Regression. *IEEE Transactions on Robotics*, 38(5):3133–3153, 2022.

[214] Xingxing Zuo, Nathaniel Merrill, Wei Li, Yong Liu, Marc Pollefeys, and Guoquan Huang. CodeVIO: Visual-Inertial Odometry with Learned Optimizable Dense Depth. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 14382–14388, 2021.