Long-term Object-based SLAM in Low-dynamic Environments

by

Jiahui Fu

B.E., Zhejiang University (2017)
S.M., University of Michigan–Ann Arbor (2018)

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2024

Authored by: Jiahui Fu
Department of Mechanical Engineering
September 21, 2023

Certified by: John J. Leonard
Samuel C. Collins Professor of Mechanical and Ocean Engineering
Thesis Supervisor

Accepted by: Nicolas Hadjiconstantinou
Chairman, Department Committee on Graduate Theses

Long-term Object-based SLAM in Low-dynamic Environments

by

Jiahui Fu

Submitted to the Department of Mechanical Engineering
on September 21, 2023, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Mechanical Engineering

Abstract

Simultaneous Localization and Mapping (SLAM) is fundamental for autonomous agents to understand their surroundings. Moreover, for advanced robotic tasks, engaging in consistent object-level reasoning is critical, especially for activities involving repetitive traversal within the same environment, such as household cleaning and object retrieval. In a changing world, robots should always locate themselves and their targets while maintaining an updated environment map. Traditional SLAM relies on static geometric primitives from observations, lacking semantic understanding. These unordered sets of points, lines, or planes struggle with object-level interpretation, leading to false estimation against scene changes.

As the world functions and evolves under the minimal unit of objects, object-aided SLAM is a logical option. This thesis revolves around long-term object-based SLAM within low-dynamic environments to bridge the communication gap between SLAM techniques and high-level robotic applications and enhance SLAM compatibility with object-level perception. It presents three contributions:

First, we propose a multi-hypothesis approach for the ambiguity-aware adoption of object poses in object-based SLAM. This approach accommodates the inherent ambiguity arising from occlusion or symmetrical object shapes. We design a multi-hypothesis object pose estimator front end in a mixture-of-expert fashion and utilize a max-mixture-based back end to infer globally consistent camera and object poses from a sequence of pose hypothesis sets.

Second, we develop two change detection approaches for offline and online applications, with two novel scene and object representations, PlaneSDF and shape-consistent neural descriptor fields, respectively. Regarding long-term operation, we account for inevitable scene changes over extended periods and the efficiency and scalability of the chosen map representations. Furthermore, we explore cluster- and object-level change detection, following a "divide-and-conquer" strategy to enable more accurate and flexible change detection through local scene differencing.

Last, we propose a neural SE(3)-equivariant object embedding (NeuSE) for long-term consistent spatial understanding in object-based SLAM. NeuSE is trained to serve as a compact point cloud surrogate for complete object models. Our NeuSE-based object SLAM paradigm directly derives SE(3) camera pose constraints compatible with general SLAM pose graph optimization. This realizes object-assisted localization and a lightweight object-centric map with change-aware mapping ability, ultimately achieving robust scene understanding despite temporal environment changes.

Thesis Supervisor: John J. Leonard
Title: Samuel C. Collins Professor of Mechanical and Ocean Engineering

Acknowledgments

To everyone who helped make this ride fun and unforgettable, these words are for you.

First and foremost, I would like to thank my advisor, Prof. John Leonard, for all the unwavering guidance, encouragement, and support along the way. Thank you for taking me to Marine Robotics Group and guiding me through SLAM, as well as a world of state estimation and mapping beyond. I am genuinely thankful for your forever positive energy and optimism whenever I confront research puzzles or uncertainty about the path ahead.

I would also like to thank my committee members, Prof. Alberto Rodriguez and Prof. Faez Ahmed, for contributing to watching me gear up and grow through this course. Alberto, thank you for your insights into robotics research and career development. Faez, I am so grateful for your knack for identifying intriguing avenues for exploration. Whether it was SLAM, manipulation, or design, the feedback and comments from you two could always enrich my mind.

I was fortunate to work with Steve Banzaert and Prof. Jonathan How as the TA for 2.678: Electronics for Mechanical Systems and 16.002: Unified Engineering: Signals and Systems. I will always remember the fun of playing with electronic circuits, the "good old math" from control classes, and the energetic sophomores. Thank you two for showing me your passion for teaching, perfect work ethic, and all the small and cheerful moments that glimpse back into my undergraduate days.

To the folks of the Marine Robotics Group, past and present, your camaraderie and encouragement have been unparalleled. Research-wise, thanks to Dehann Fourie for guiding and bearing with me through complex mathematical concepts when I was still a newbie, Kevin Doherty for all the insightful discussions on research and juggling, and Kurran Singh for all the time we spent (and wasted) messing up with mugs and bottles in Holodeck. To each and every one of you - Alan Papalia, Brendan O'Neil, Chad (Qiangqiang) Huang, David Baxter, David Rosen, Megan Flynn, Tim Magoun, Tonio Teran, Violet Killy, Yihao Zhang, and Ziqi Lu, I will remember all the joyful conversations, hotpot laughter, and CBC gatherings. I would also like to thank Nira Manokharan for taking care of all the administrative stuff and keeping our lives running smoothly around the lab.

I also had the pleasure of spending two wonderful summers at Meta Reality Labs, working with Dr. Yuichi Taguchi and Dr. Shichao Yang on exploring the future of AR/VR technologies. Thank you for all the enriching discussions, enlightening whiteboard sessions, and invaluable career advice you generously shared with me. I thank my colleagues, Chengyuan Lin, Andrea Cohen, Yifu Zhang, Yipu Zhao, and Meng Tang, for our frequent research exchanges, collaborative problem-solving, and casual conversations.

Acknowledgments also go to Leslie Regan, Saana McDaniel, and Una Sheehan for their logistical support within MechE. I am grateful to the MIT Asian Dance Team for injecting vibrancy into my early Ph.D. days. To the Sidney-Pacific graduate community, thank you for the sense of belonging. The pandemic made me complete a significant portion of the thesis work in my SidPac dorm. I will miss being a resident/a hall councilor, the vibrant community events I was part of, and all the sunrises and sunsets over the Charles encountered when I was working by the window.

For all the friends at MIT and beyond, thank you for being a constant source of support in research and life. Thanks to Yichen Zhan, Michelle Xu, Cheng Zheng, Francesca Pentimalli, Wajeeha Ahmad, Lingbo Ji, Wenhui Tang, Can Pu, Jingwen Cai, Xingyao Wu, and Kuangye Lu (the list can just go longer) for all the help and good memories.

Last but not least, I want to thank Rong Wu and Nianxian Fu for being the best parents in the world. Thank you for letting me pursue what I want and always being my back, no matter what happens. I want to express my gratitude to my grandparents, who looked after me when I was little. And especially to my grandma, Shengfen Zhou, who first introduced me to the world of “mapping” with a collection of tools handed down from her mapper days in the ’70s. Beyond that, special thanks to Yilun Du. Be it my collaborator, birthday buddy, or forever listener, I am just so lucky to have shared and continue to share my journey with you.

Contents

1 Introduction **15**

1.1 Motivation 15

1.2 Overview 17

1.3 Problems of Interest and Contributions 19

1.3.1 Problems of Interest 19

1.3.2 Contributions and Thesis Outline 20

2 SLAM Preliminaries **23**

2.1 Representations and Optimization for SLAM 23

2.1.1 Graphical Models for SLAM 23

2.1.2 Optimization for SLAM: Lie Algebra 25

2.2 Visual SLAM Formulation 26

2.2.1 Camera Projection Model 26

2.2.2 Geometric Cost 27

2.2.3 Photometric Cost 28

2.2.4 Relative Pose Cost 28

2.3 Related Work 29

2.3.1 Visual SLAM 29

2.3.2 Object-based SLAM 30

2.4 Bridging Objects and SLAM 31

3 A Multi-hypothesis Approach for Object Pose Estimation **34**

3.1 Introduction 35

3.2 Related Work 37
3.2.1 6D Object Pose Estimation 37
3.2.2 Object-based SLAM 38
3.2.3 Uncertainty-aware SLAM Back End 38
3.3 Front End: Multi-hypothesis 6D Object Pose Estimator (MHPE) . . 39
3.3.1 MHPE Network 39
3.3.2 Learning Objective for MHPE Outputs 40
3.4 Back End: Max-mixture-based Multi-hypothesis Modeling and Optimization 41
3.5 Experiments and Results 43
3.5.1 Datasets 44
3.5.2 Metrics 44
3.5.3 Implementation Details 45
3.5.4 Results on YCB-Video Dataset 46
3.5.5 SLAM Results for the Simulated Video 49
3.6 Conclusion 52

4 PlaneSDF-based Change Detection for Long-term Dense Mapping 54
4.1 Introduction 55
4.2 Related Work 57
4.3 Method Overview 59
4.3.1 PlaneSDF Instantiation 59
4.3.2 PlaneSDF Registration 61
4.3.3 Height Map Comparison 62
4.3.4 3D Voxel Validation 63
4.4 Experiments and Results 67
4.4.1 Datasets 68
4.4.2 Evaluation Metrics 68
4.4.3 Implementation Details 69
4.4.4 Results on the Synthetic Tabletop Dataset 69

4.4.5 Results on the Object Change Detection Dataset 71
4.5 Conclusion 76

5 Robust Change Detection with Neural Descriptor Fields 77
5.1 Introduction 78
5.2 Related Work 81
5.2.1 Change Detection 81
5.2.2 Neural Implicit Representations 81
5.3 Category-level Object Representation for Partial Observations 82
5.3.1 Neural Descriptor Fields 82
5.3.2 Shape Consistency 83
5.3.3 Training in Simulation 85
5.3.4 Object Representation 85
5.4 NDF-based Object Change Detection 86
5.4.1 Spatial Object Tree Construction 86
5.4.2 NDF-based Change Detection 88
5.5 Experiments and Results 91
5.5.1 Datasets 92
5.5.2 Metrics 93
5.5.3 Implementation Details 93
5.5.4 Generalization to Unseen Instances 94
5.5.5 Results on Change Detection 96
5.6 Conclusion 101

6 NeuSE: Neural SE(3)-equivariant Embedding for Consistent Spatial Understanding with Objects 102
6.1 Introduction 103
6.2 Related Work 107
6.3 Category-level Neural SE(3)-equivariant Embedding (NeuSE) for Objects 108
6.3.1 Learning SE(3)-equivariance across Viewing Angles 110
6.3.2 Dealing with Pose Ambiguity 111

6.3.3 Shape Consistency across Viewing Angles 114
6.3.4 Training in Simulation 114
6.4 NeuSE-based Object SLAM with Long-term Scene Inconsistency . . . 115
6.4.1 System Formulation and Update 116
6.4.2 Data Association 116
6.4.3 Pose Graph Optimization 117
6.4.4 Change-aware Object-centric Mapping 119
6.5 Experiments and Results 121
6.5.1 Datasets 121
6.5.2 Experimental Details 122
6.5.3 Localization with Temporal Scene Inconsistency 125
6.5.4 Change-aware Object-centric Mapping 136
6.6 Conclusion 138

7 Conclusion 140
7.1 Contributions 140
7.2 Lessons Learned 142
7.3 Limitations 144
7.4 Future Work 145

List of Figures

1-1 Object as the minimal unit for world operation and changes. 16
1-2 Various map representations. 17

2-1 A typical scenario for SLAM. 24

3-1 Approach overview. 36
3-2 Hypothesis diversity from the MHPE front end on the YCB object clamp. 41
3-3 Max-mixtures for processing multiple pose hypotheses. 42
3-4 Simulated video top view and YCB object layout. 45
3-5 Qualitative results of object pose estimation. 47
3-6 Running average of the estimation error for rotation and translation across the course of the robot motion. 49
3-7 Comparison of camera trajectories and object position estimates. . . . 50
3-8 Landmark pose estimation errors. 52

4-1 System Overview. 56
4-2 Change detection results for a complete indoor scene. 60
4-3 Height map comparison. 63
4-4 Key voxel distribution and corresponding similarity score distribution of planes with and without changes. 64
4-5 Sample change detection results on the synthetic tabletop dataset. . . 70

4-6 Illustration of key voxel distribution and detected false positive points from the per point FPFH feature matching baseline for two unchanged sub-scenes. 73
4-7 Qualitative examples of the change detection results for the four scenes in the object change detection dataset. 74

5-1 Approach overview. 79
5-2 Training sample generation. 86
5-3 Neighborhood comparison as object graph matching. 89
5-4 Synthetic scene layout and camera trajectories. 91
5-5 Real-world scene layout and camera trajectories. 92
5-6 3-view shape reconstruction of five unseen mugs from the synthetic and real-world testing sequences. 95
5-7 Robustness to little observation overlap. 97
5-8 Reconstruction results. 100

6-1 Schematic of consistent spatial understanding with NeuSE. 104
6-2 System overview. 106
6-3 (a) Breaking pose ambiguity with covisible ambiguous objects. (b)Latent symmetry. 112
6-4 Pose graph optimization. 118
6-5 Object layout comparison through graph matching. 119
6-6 Evaluation data overview. 122
6-7 Column 1-2: Comparison of estimated and ground truth trajectories on synthetic sequences. Column 3: Ambiguous objects for inter-frame transform prediction. 128
6-8 Distribution of translational RPE along synthetic sequences. 129
6-9 Column 1-2: Visualization of the estimated trajectories: (a) CubeSLAM and (b) EM-Fusion. Column 3: Trajectory/object cuboid estimation drifts of the two selected object SLAM baselines. 131
6-10 Visualization of estimated trajectories against ground truth. 132

6-11 Complete object reconstruction of synthetic sequences for the planar and non-planar object layouts. 136
6-12 Results of change-aware mapping for the real-world sequences. 139

List of Tables

3.1 Quantitative ADD and ADD-S AUC results on YCB-Video Dataset. . 47

4.1 Result comparison of the proposed approach with three baselines. . . 72

5.1 Change detection results on the synthetic sequences. 98

5.2 Change detection results on the real-world sequences. 99

6.1 Encoder architecture. 125

6.2 Decoder architecture. 125

6.3 RMSE of ATE and Translational RPE on synthetic sequences. 126

6.4 RMSE of ATE on real-world sequences. 130

6.5 Change detection results on the synthetic (planar-layout) and real-world sequences. 135

Chapter 1

Introduction

1.1 Motivation

Simultaneous Localization and Mapping (SLAM), as its name suggests, refers to the task of an autonomous agent localizing itself as well as building a map of the surroundings when exploring a new environment. This is a fundamental capability for robots, serving as the perception basis for higher-level robotic tasks, such as object retrieval and household cleaning.

As the world operates and evolves on the minimal unit of discrete objects, these sophisticated robotic tasks often emulate the cognitive processes of humans. In this context, the assessment of the degree of "objectness" in the environment becomes a crucial facet (see Fig. 1-1). When humans issue a directive, they instinctively embed their own comprehension of "objects" into the task's framework. For instance, a simple command like "Kindly retrieve that azure coffee mug for me." inherently integrates our conceptualization of a "blue coffee mug". To execute a similar command, a household robot must embark on a sequential journey: grasping the essence of a "blue coffee mug", identifying and pinpointing the target's location on the map, and ultimately planning a path to procure the item based on its spatial relationship with the target's position.

The perceptual discrepancy between the languages employed by robots and humans introduces additional challenges in achieving effective localization and mapping.

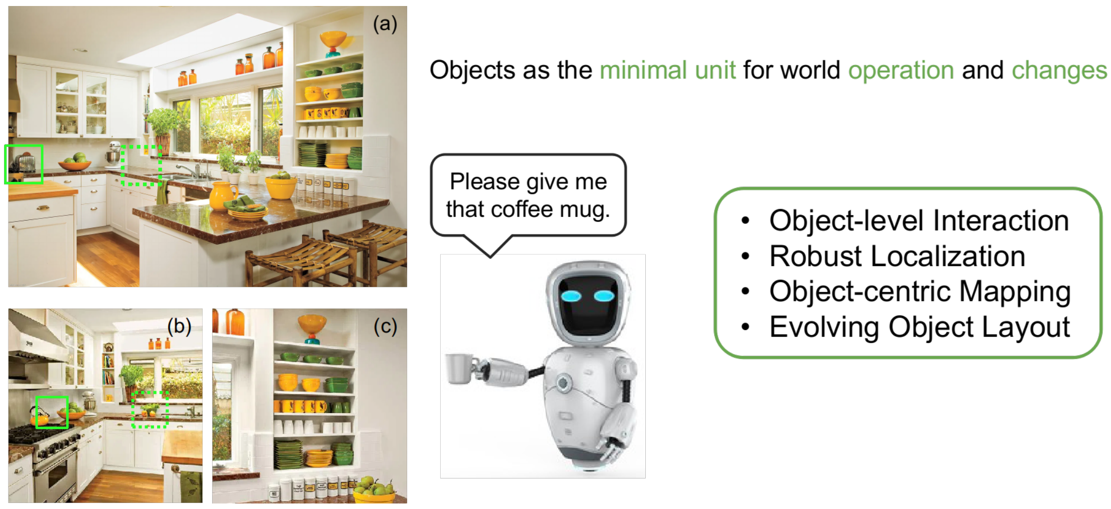

(a)
Objects as the minimal unit for world operation and changes
Please give me that coffee mug.
• Object-level Interaction
• Robust Localization
• Object-centric Mapping
• Evolving Object Layout
(b)
(c)

Figure 1-1: Object-level reasoning is a desirable capability for advanced robotic tasks where traditional SLAM approaches are not a perfect fit. For a robot operating in a household environment (a) with an object retrieval task, it is expected to develop a sense of "objectness" in the environment, e.g., coffee mugs on the shelf (c), and keep up with the evolving object layout of the room (b).

Traditional SLAM methods [27, 11, 29] often rely on pixel-level photometric differences or manually engineered lower-level geometric features to infer relative motion between frames. Lacking semantic insights, these localization approaches and resultant mapped outputs remain devoid of compatibility with object-oriented reasoning. Consequently, they may struggle to seamlessly incorporate instructions framed at the object level, as mandated by high-level tasks. This situation imposes extra computational burdens and complexities when attempting real-time communication with the robot during task execution.

In extended operating scenarios, the challenge of this non-smooth communication is magnified, exacerbated by ongoing changes. Long-term operation robustness stands as a key requirement for robots. Yet, encountering changes in the environment is inevitable, particularly in tasks involving repetitive traversal within the same space, e.g., mobile manipulation. Tables may shift locations periodically, and commonplace items like coffee mugs or water bottles may frequently change places. A capable SLAM pipeline must effectively adapt to environment changes, continuously updating its map of the surroundings.

Conversely, typical SLAM methods often function on the assumption of a static

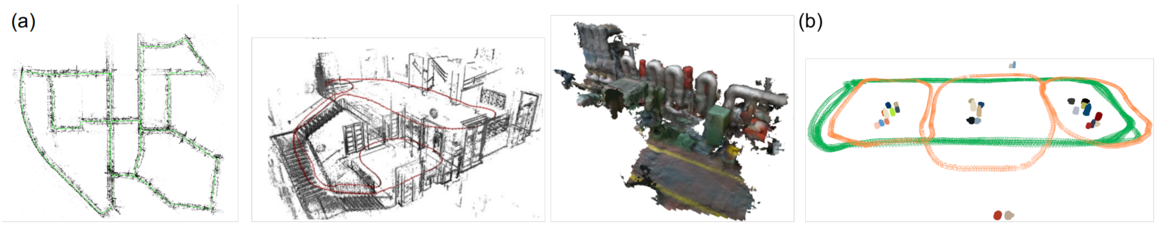


Figure 1-2: (a) Map representations of conventional SLAM systems. From left to right: A sparse point cloud map generated by ORB-SLAM2 [83], a semi-dense map produced by DSO [28], and a dense TSDF reconstruction map obtained through C-blox [79]. (b) A dense, object-centric map along with robot trajectory estimate created by our own system.

environment, rendering them susceptible to false feature correspondences resulting from changed objects. Additionally, as inference processes mainly operate on pixel-wise or point-wise data, the lack of "object" level comprehension hampers the handling of environment changes and map updates, leading to cumbersome and error-prone update procedures (see Fig. 1-2).

Hence, given the limited compatibility with object-level reasoning in advanced robotic tasks and the challenge of addressing environmental object changes, enhancing common SLAM pipelines with object-level understanding emerges as an attractive proposition. This, in essence, outlines the core goal of this thesis:

Develop a general SLAM paradigm that can operate on object-level information for long-term spatial understanding against low-dynamic temporal scene inconsistency.

Here, we target the common RGB-D data modality, and a *low-dynamic* environment pertains to scenarios with object layout changes, but whose moving processes are not directly captured by the camera.

1.2 Overview

To address the aforementioned challenges, this thesis delves into the potential of harnessing object-level information to enhance the localization and mapping processes of SLAM. Specifically, the focus is on enabling the robot's localization to align with downstream tasks necessitating object-level perception, while effectively dealing with long-term scene changes. Consequently, the outcome of the localization process naturally leads to the establishment of change-aware, object-centric mapping capability, thereby offering more support to the execution of advanced robotic tasks.

Embracing the notion that the world's perception hinges upon objects, objects emerge as a logical avenue for aiding localization, and an object-centric map stands as a lightweight and adaptable reflection of the evolving environment layout. Taking advantage of the object shape invariance and the robot-landmark geometric constraints compactly encoded within 6D object poses, we identify object shapes and poses as pivotal bridges that link object observations and the SLAM reasoning process.

Our consideration primarily orbits around two realms: the agent itself and the objects, although these two are interrelated. On the one hand, within the context of robot motion, pose ambiguity can emerge due to occlusion or inherent geometric symmetry, while inconsistent shape observations might arise from various occlusion patterns and viewing angles. On the other hand, for objects, the challenge lies in efficient representation. While conventional approaches employ unordered point, line, or voxel collections to characterize objects via point clouds or Truncated Signed Distance Function (TSDF), the question arises: Do these representations sustain efficiency during prolonged operation? Moreover, as objects diminish in size and lie closer, do we still require individual representations for each object, or would a single representation per cluster suffice?

In this spirit, the efforts of this thesis unfold across three core themes: (1) Addressing object ambiguity in the environment; (2) Devising object- and cluster-level representations for change detection; and (3) Establishing a foundation for long-term spatial understanding amidst temporal inconsistency. By delving into these three topics, we establish a framework showcasing our endeavors to achieve the overarching objective: the realization of robust, long-term SLAM capabilities in low-dynamic environments that seamlessly aligns with object-oriented advanced robotic tasks.

1.3 Problems of Interest and Contributions

1.3.1 Problems of Interest

In the pursuit of creating an object-based SLAM system that attains robust spatial understanding over extended periods, this thesis endeavors to tackle three pivotal challenges that have not been comprehensively addressed in previous research.

Developing Representations. The initial challenge revolves around developing effective and scalable representations for objects and scenes. In the context of mobile robotics, issues such as disparities in occlusion and viewing angles often result in erroneous data associations. Furthermore, long-term operation typically involves a gradual increase in the number of objects and the size of observed environments. As a consequence, these issues prompt an exploration of object and scene representations that can compactly yet effectively capture pose and geometric details. Traditional representations used in SLAM systems, such as point clouds and TSDF, while suitable for short-term scenarios, may prove memory-inefficient as operational time increases and may only capture partial details from varying perspectives. To facilitate data association across different angles and occlusion patterns during prolonged operation, we aim to devise object and scene representations that encode object geometric attributes with reduced dependence on viewing angles and can scale seamlessly for extended durations.

Handling Environmental Changes. The second challenge pertains to addressing environment changes during long-term operation. Existing research has primarily focused on short-term and highly dynamic scenes, where solutions often involve excluding or predicting the potential impacts of object dynamics on the scene. These methods typically encompass actions like removing movable object classes and developing motion models. However, the situation turns harder when dealing with low-dynamic scenes over an extended period. As the motion of objects is not directly captured by the camera, the environment may appear misleadingly static with undesirable motion blur or dynamics not explicitly manifesting in sensor images. In such cases, objects merely shift to new locations with altered orientations, leading to confusion in data association. Consequently, a critical aspect involves developing change detection techniques that continuously reflect the most up-to-date scene layout, thus providing a clear and conflict-free understanding of the current operational environment.

Utilizing Object-Level Information. The last challenge involves smoothly and effectively integrating object-level information into the SLAM pipeline. Current approaches often obtain inter-frame camera pose constraints from exterior object pose estimators or engage in joint optimization of object representations alongside other measurements. This requires the availability of external pose estimators and does not allow for the direct integration of object-level data into existing SLAM pipelines. Furthermore, the common issue of object pose ambiguity is rarely addressed adequately while maintaining a balance between computational efficiency and speed. To tackle the ambiguity issue and explore convenient ways to incorporate object poses into common SLAM pipelines, we seek to investigate methods for leveraging objects most accurately and seamlessly within the SLAM system in either a coupled or decoupled fashion.

1.3.2 Contributions and Thesis Outline

Following our structured approach, we proceed to present our contributions in the ensuing chapters. The outline for the rest of the thesis is as follows:

- Chapter 2: **Technical Foundamentals.** In this chapter, we lay the groundwork by providing essential insights into visual and, in particular, object-based SLAM formulation, and the broader context of visual and object-based SLAM research.

- Chapter 3: **Multi-hypothesis Approach to Object Pose Ambiguity.** In this chapter, we leverage multi-hypothesis object pose estimation and adopt max-mixtures during optimization to recover a set of globally consistent estimates of robot and landmark poses.

- Chapter 4: **PlaneSDF Representations and Cluster-level Change Detection.** Here, we propose a novel PlaneSDF representation for mapping and design a cluster-level PlaneSDF-based change detection scheme. This approach targets scenarios with both source and target maps available, catering to offline operations.

- Chapter 5: **Object-level Neural Descriptor Fields (NDFs) and Change Detection.** In this chapter, we introduce the concept of shape-consistent Neural Descriptor Fields (NDFs) and outline an NDF-based change detection approach. Designed for online operations without pre-alignment tools, this technique addresses scene changes with extra robustness to localization errors.

- Chapter 6: **Neural SE(3)-equivariant Embeddings (NeuSE) for Long-term Spatial Understanding.** This chapter presents Neural SE(3)-equivariant Embeddings (NeuSE) for objects and illustrates its full potential in supporting object SLAM, fostering consistent spatial comprehension in the presence of long-term scene changes.

- Chapter 7: **Conclusion and Future Directions.** Our concluding chapter synthesizes our contributions and thoughts, acknowledges limitations, and outlines three potential avenues for future exploration.

The publications involved in chapter 3-6 are as follows:

- Chapter 3: J. Fu, Q. Huang, K. Doherty, Y. Wang, and J. J. Leonard. A Multi-Hypothesis Approach to Pose Ambiguity in Object-Based SLAM. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021.

- Chapter 4: J. Fu, C. Lin, Y. Taguchi, A. Cohen, Y. Zhang, S. Mylabathula, and J. J. Leonard. PlaneSDF-Based Change Detection for Long-Term Dense Mapping. IEEE Robotics and Automation Letters (RA-L) with IROS option, 2022.

- Chapter 5: J. Fu, Y. Du, K. Singh, J. B. Tenenbaum and J. J. Leonard. Robust Change Detection Based on Neural Descriptor Fields. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2022.

- Chapter 6: J. Fu, Y. Du, K. Singh, J. B. Tenenbaum and J. J. Leonard. NeuSE: Neural SE (3)-Equivariant Embedding for Consistent Spatial Understanding with Objects. Robotics: Science and Systems (RSS), 2023.

Chapter 2

SLAM Preliminaries

In this chapter, we first cover some fundamentals of SLAM formulation and optimization techniques. We then introduce camera geometry widely employed in visual SLAM, followed by multiple cost functions commonly seen in object-based SLAM. We later review some existing approaches in visual SLAM and object-based SLAM. Lastly, we discuss how we should address the research gap between current progress and realizing a robust object-based SLAM system for long-term operation.

2.1 Representations and Optimization for SLAM

A typical SLAM system consists of two main components: a *front end* and a *back end* that work together to estimate camera poses and construct the environment map. The front end handles sensor measurement processing and generates odometry constraints, while the back end processes these inter-frame pose constraints to perform inference. Here, we introduce factor graphs and Lie Algebra, which comprise the foundation of SLAM problem representation and optimization.

2.1.1 Graphical Models for SLAM

A typical SLAM problem (see Fig. 2-1) can be defined as follows: Suppose an agent (e.g., a racecar) is moving among several landmarks (e.g., teddy bears). Given a set

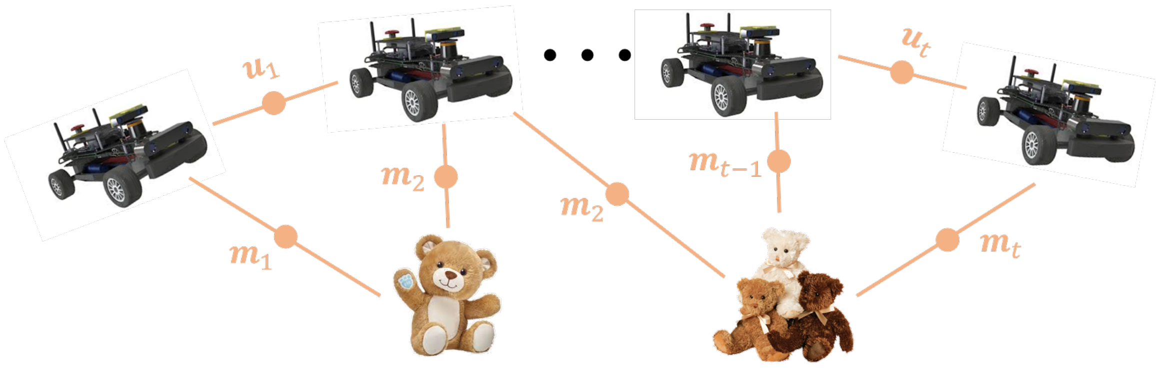


Figure 2-1: A typical scenario for SLAM. A racecar is moving around two teddy bears (landmarks) from time 0 to t, obtaining odometry measurements $\mathbf{u}_i$s and landmark observations $\mathbf{m}_i$s.

of (1) odometry measurements of adjacent camera frames, $\{\mathbf{u}_i\}_{t=1}^{t=T}$, and (2) landmark measurements, $\{\mathbf{m}_i\}_{t=1}^{t=T}$, from on-board sensors such as RGB-D cameras, lidars, and IMUs, the SLAM problem seeks to solve for the camera poses, $\{\mathbf{T}i\}_{t=1}^{t=T}$, and the landmark poses, $\{\mathbf{P}_i\}_{i=1}^{i=N}$, at all time steps.

Treating unknown variables, i.e., camera and landmark poses, as variable nodes, and the measurements of the odometry and landmarks as factor nodes, *factor graphs* can then serve as a general graphical model for SLAM. In particular, we assume the value of each node follows some certain distributions determined by the sensor *observation model*, and the optimization problem for SLAM can usually be formulated as a maximum *a posteriori* (MAP), or if no prior knowledge $p(\boldsymbol{\theta})$ about the variable value is available, a maximum likelihood estimation (MLE) inference problem. That is, we want to find a set of values for variable nodes, $\boldsymbol{\theta} = (\{\mathbf{T}_i\}_{t=1}^{t=T}, \{\mathbf{P}_i\}_{i=1}^{i=N})$, that maximizes the posterior probability of $\boldsymbol{\theta}$ given measurements $\mathbf{Z} = (\{\mathbf{u}_i\}_{t=1}^{t=T}, \{\mathbf{m}_i\}_{t=1}^{t=T})$ from all sensors:

$$\begin{aligned}\boldsymbol{\theta}^* &= \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\mathbf{Z}) \\ &\equiv \arg\max_{\boldsymbol{\theta}} p(\mathbf{Z}|\boldsymbol{\theta})p(\boldsymbol{\theta}) \quad (MAP) \\ &\equiv \arg\max_{\boldsymbol{\theta}} p(\mathbf{Z}|\boldsymbol{\theta})\cancel{p(\boldsymbol{\theta})} \quad (MLE).\end{aligned} \tag{2.1}$$

The optimization process is typically conducted using common optimization methods, such as the Gauss-Newton or Levenberg-Marquardt algorithm, and forms of cost functions can vary according to sensor types.

2.1.2 Optimization for SLAM: Lie Algebra

Typical optimization methods involve a gradual variable update step $\mathbf{x} \leftarrow \mathbf{x} + \delta\mathbf{x}$ guided by the cost function. This rule works well for variables belonging to the Euclidean space. However, for 6D poses, which are in the three-dimensional Special Euclidean Group, $\mathbf{T} = (\mathbf{R} \in \mathbb{R}^{3\times3}, \mathbf{t} \in \mathbb{R}^3) \in \mathrm{SE}(3)$, the simple "add-for-update" rule no longer works for the rotation part, which belongs to the three-dimensional Special Orthogonal Group, $\mathfrak{so}(3)$. To perform optimization with a valid addition operation for the elements in $\mathfrak{so}(3)$ without additional constraints, we introduce Lie Algebra.

For each rotation matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$ in the three-dimensional space, its associated Lie Algebra $\mathfrak{so}(3) = \{\boldsymbol{\phi} \in \mathbb{R}^3, \boldsymbol{\Phi} = \boldsymbol{\phi}^\wedge \in \mathbb{R}^{3\times3}\}$ is defined as follows:

$$\boldsymbol{\Phi} = \boldsymbol{\phi}^\wedge = \begin{bmatrix} 0 & -\phi_3 & \phi_2 \\ \phi_3 & 0 & -\phi_1 \\ -\phi_2 & \phi_1 & 0 \end{bmatrix}, \tag{2.2}$$

where the original rotation matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$ can be recovered by the exponential mapping as:

$$\begin{aligned} \mathbf{R} &= exp(\boldsymbol{\phi}^\wedge) \\ \boldsymbol{\phi} &= ln(\mathbf{R})^\vee. \end{aligned} \tag{2.3}$$

Similarly, for 3D Special Euclidean Group, SE(3), the corresponding Lie Algebra $\mathfrak{se}(3)$ is defined as:

$$\mathfrak{se}(3) = \{\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{\phi} \end{bmatrix} \in \mathbb{R}^6, \boldsymbol{\rho} \in \mathbb{R}^3, \boldsymbol{\phi} \in \mathfrak{so}(3), \boldsymbol{\xi}^\wedge = \begin{bmatrix} \boldsymbol{\phi}^\wedge & \boldsymbol{\rho} \\ \mathbf{0}^T & 0 \end{bmatrix} \in \mathbb{R}^{4\times4}\}, \tag{2.4}$$

where we have

$$\begin{aligned} \mathbf{T} &= exp(\boldsymbol{\xi}^{\wedge}) \\ \boldsymbol{\xi} &= ln(\mathbf{T})^{\vee}. \end{aligned} \tag{2.5}$$

Suppose we have a point $\mathbf{p} \in \mathbb{R}^3$ that is rotated by $\mathbf{R} \in \mathbb{R}^{3\times3}$ and the associated Lie Algebra $\boldsymbol{\phi}$ of $\mathbf{R}$. If we apply a small left disturbance $\delta\mathbf{R}$ with corresponding Lie Algebra $\delta\boldsymbol{\phi}$, we have

$$\begin{aligned} \frac{\partial \mathbf{Rp}}{\partial \delta\boldsymbol{\phi}} &= lim_{\delta\boldsymbol{\phi}\to\mathbf{0}} \frac{exp(\delta\boldsymbol{\phi}^{\wedge})exp(\boldsymbol{\phi}^{\wedge})\mathbf{p} - exp(\boldsymbol{\phi}^{\wedge})\mathbf{p}}{\delta\boldsymbol{\phi}} \\ &= (-\mathbf{Rp})^{\wedge}. \end{aligned} \tag{2.6}$$

In a similar vein, for the same point $\mathbf{p}$ and the transform $\mathbf{T} = (\mathbf{R}, \mathbf{t})$, if we give $\mathbf{T}$ a small left disturbance $\Delta\mathbf{T} = exp(\delta\boldsymbol{\xi}^{\wedge})$, we have

$$\begin{aligned} \frac{\partial \mathbf{Tp}}{\partial \delta\boldsymbol{\xi}} &= lim_{\delta\boldsymbol{\xi}\to\mathbf{0}} \frac{exp(\delta\boldsymbol{\xi}^{\wedge})exp(\boldsymbol{\xi}^{\wedge})\mathbf{p} - exp(\boldsymbol{\xi}^{\wedge})\mathbf{p}}{\delta\boldsymbol{\xi}} \\ &= \begin{bmatrix} \mathbf{I} & (-\mathbf{Rp} + \mathbf{t})^{\wedge} \\ \mathbf{0}^T & \mathbf{0}^T \end{bmatrix}. \end{aligned} \tag{2.7}$$

2.2 Visual SLAM Formulation

In this thesis, we limit our scope to "Visual SLAM", namely, SLAM systems that operate on visual measurements from RGB and depth cameras. In this section, we will accordingly introduce the cost functions that have been adopted in visual SLAM.

2.2.1 Camera Projection Model

One cost used to guide the camera and landmark optimization process involves comparing the estimated 2D/3D locations of the feature points with those obtained from the measurements.

Here, we assume a pinhole camera model with intrinsic matrix K (in image pixel unit):

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.8}$$

where f_x and f_y are the focal lengths in the x and y direction of the image coordinate, respectively. c_x and c_y are the coordinates of the camera's principal point.

Given a point $\mathbf{p}^w = (x, y, z) \in \mathbb{R}^3$ in the world coordinate frame and the current camera pose $\mathbf{T} = (\mathbf{R} \in \mathbb{R}^{3\times3}, \mathbf{t} \in \mathbb{R}^3)$, then the projection process will first transform the point in the world frame, $\mathbf{p}^w$, to the local camera frame, $\mathbf{p}^c$, and then onto the image frame to its pixel coordinate (u, v):

$$z \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \mathbf{K}\mathbf{p}^c = \mathbf{K}\mathbf{T}\mathbf{p}^w \overset{def}{=} \pi(\mathbf{T}\mathbf{p}^w), \tag{2.9}$$

where we define π to be the projection process from the 3D camera frame to the 2D image frame.

Recently, neural implicit representations have been newly introduced into SLAM research due to their ability to perform object/scene completion and novel view synthesis. When neural implicit representations are employed for map representation, the camera projection process is parametrized as a neural network, typically a multi-layer perceptron (MLP), instead of adhering to the previously mentioned pinhole camera model. We will explore how to integrate them into the SLAM pipeline in Chapter 5 and Chapter 6.

2.2.2 Geometric Cost

One commonly used cost function in visual SLAM is the geometric cost. Given a point in the map maintained by the SLAM system, $\mathbf{p}^w = (x, y, z)$, this cost computes the reprojection error, i.e., the pixel coordinate difference between the estimated projected pixel location using the current camera pose estimate, $\mathbf{T}$, and the actual corresponding pixel location $\mathbf{u}$:

$$\mathbf{e}_{reproj} \overset{def}{=} \mathbf{u} - \pi(\mathbf{T}\mathbf{p}^w). \tag{2.10}$$

2.2.3 Photometric Cost

Another widely adopted cost function, the photometric cost, computes the photometric differences of the corresponding pixel locations of the same 3D point in different camera frames. Here, the 3D point is no longer represented by its three-dimensional coordinate (x, y, z). Instead, each point in the map is specified by its pixel location along with the estimated depth, $\mathbf{d}$, i.e., $\mathbf{p}^c = (\mathbf{u}, \mathbf{d})$ in a camera frame with estimated frame pose $\mathbf{T}_c$. Then, given two camera frames, I_1 and I_2, with the estimated relative transform from I_1 to I_2 as $\mathbf{T}_{12}$, for a pixel, $(\mathbf{u}, \mathbf{d})$, in the first frame, the photometric cost is computed as:

$$\mathbf{e}_{pmt} \overset{def}{=} I_1(\mathbf{u}) - I_2(\pi(T_{12}\pi^{-1}(\mathbf{u}, \mathbf{d}))). \tag{2.11}$$

2.2.4 Relative Pose Cost

Among SLAM with various sensor data modalities, the most lightweight instantiation of a SLAM problem is PoseSLAM, which estimates solely the camera poses of camera frames without maintaining an actual map of the environment. Here, the graphical representation of SLAM only includes odometry (adjacent) and loop-closing (non-adjacent) relative frame measurements from various sources, and the relative pose error between two camera frames $\mathbf{T}_i$ and $\mathbf{T}_j$ is computed as follows:

$$\mathbf{e}_{pose} \overset{def}{=} \ln(\mathbf{T}_i^{-1}\mathbf{T}_j)^\vee \ln(\mathbf{T}_{ij}^{-1}\mathbf{T}_i^{-1}\mathbf{T}_j)^\vee, \tag{2.12}$$

where $\mathbf{T}_{ij} = \mathbf{T}_i^{-1}\mathbf{T}_j$ is the relative transform between the two poses.

2.3 Related Work

In this section, we first review some relevant works in visual SLAM and then focus specifically on SLAM works that integrate object-level information into the SLAM pipeline.

2.3.1 Visual SLAM

For SLAM systems operating on visual data, i.e., RGB and RGB-D cameras, previous approaches can be roughly divided into two categories: sparse feature-based methods and dense direct methods. The former usually involves combining hand-crafted higher-dimensional features or geometric primitives such as lines and planes with geometric cost for localization and maintains a sparse map consisting of mostly feature points. The latter directly compares pixel-level learning-based object/scene representations for photometric differences, with the resulting map a dense reconstruction of the environment.

For sparse feature-based methods, one of the earliest works, PTAM [61], adopted FAST features for conducting the tracking and mapping tasks. ORBSLAM [11] was built on top of the ORB feature and had four modules: tracking, mapping, relocalization, and loop-closing, which has become a popular paradigm for SLAM system design. Other hand-crafted features, such as SIFT and SURF, have also been seen in works such as Fovis [42] and Libviso [36], respectively. In addition to point-based features, previous works have explored integrating lines [120, 137] and planes [104, 55] for camera localization.

For dense methods, DTAM [86] pioneered in GPU-accelerated real-time direct dense SLAM, which was then adopted in later learning-based SLAM pipelines, such as Demon [124] and Deeptam [143]. Deepfactors [16] employed a set of depth maps as bases so as to reduce the computation cost during optimization. For those CPU-only solutions, LSD-SLAM [27] and DSO [28] compared solely pixel intensity differences only around regions with high gradients, thereby saving some computation to be runnable on CPUs. TANDEM [63] followed the DSO architecture and further integrated multi-view stereo to produce a real-time dense SLAM system.

More related to our works are previous efforts in exploring various environment representations for developing SLAM pipelines. Voxel-based Truncated Signed Distance Fields (TSDF) [85, 8] and surfels [132, 107] are popular representations adopted in dense SLAM, such as KinectFusion [85], Kintinuous [131], ElasticFusion [132], and BundleFusion [17]. DROID-SLAM [122] learned to regress optical flow and recurrently refined camera poses and pixel-wise depth through a Dense Bundle Adjustment layer. Recently, with the development of neural implicit representations, several works have investigated using a latent code to represent objects or the environment, such as CodeSLAM [6], SceneCode [142], and NodeSLAM [115]. These works conducted iterative optimization of the latent code representation for the scene or objects so as to make the learned code align with all sensor (RGB-D) measurements.

2.3.2 Object-based SLAM

SLAM++ [103] introduced object-based SLAM by incorporating camera-object constraints with objects from a predefined model database. Attempts [75, 76, 100, 136] have been made to leverage semantic segmentation for instance-level dense reconstructions. Furthermore, simple parameterized geometry, e.g., ellipsoids adopted by Nicholson et al. [87] and Hosseinzadeh et al. [49] and cuboids by Yang and Scherer [138], have been explored to guide the joint optimization of the object shape parameters and camera poses. For environments with moving objects, Strecke and Stueckler [114] proposed an object-level SLAM approach that utilized local Signed Distance Function (SDF) object volumes for tracking moving objects and performing camera localization. Recently, following the aforementioned trend in integrating neural implicit representations into SLAM, neural shape priors for objects have been utilized in object SLAM pipelines for their shape completion capability. NodeSLAM [115] adopted a class-level optimizable object shape descriptor and used RGB-D images for joint estimation of object shapes, poses, and camera trajectory through iterative probabilistic rendering optimization. DSP-SLAM [128], on the other hand, used DeepSDF [95] for object representation and optimized the object code, camera poses, and sparse landmark points all together through a similar rendering loss in RGB, stereo, or stereo+LiDAR modalities.

2.4 Bridging Objects and SLAM

Considering the gaps in existing visual, particularly object-based SLAM research, we here discuss our solutions to narrow these gaps as presented in the following chapters and how they are related to achieving the ultimate goal of a robust object-based SLAM system for long-term spatial understanding against scene inconsistency.

Effective and Scalable Object and Scene Representations. Our first topic addresses the challenge of devising object and scene representations that are both effective and scalable. Traditional SLAM methodologies, whether relying on sparse or dense map representations, often fall short of providing sufficient semantic information for advanced robotic tasks. Additionally, they can be vulnerable to false correspondence matches when confronted with long-term scene changes. These extracted representations merely reflect sensor observations, which may exhibit inconsistency across different viewing perspectives and not readily scale to encompass larger scenes.

To address these limitations, we explore the idea of composing multiple low-level geometric primitives, thus interpreting the environment structure as a collection of planes, each accompanied by a Truncated Signed Distance Function (TSDF) volume. This approach allows us to describe local geometric details. By doing so, we decompose global operations into multiple local operations constrained within the current camera viewing frustum, enhancing robustness and consistency across viewing angles and occlusion patterns.

In parallel, we investigate emerging neural implicit representations for their shape completion capabilities. Through the art of embedding geometric attributes into a compact code, these representations encode not only observable elements but also hidden details along the current viewing direction. This strategy enables us to attain a certain level of three-dimensional consistency, particularly regarding object shapes across varying viewing angles. Consequently, our approach best mimics the behavior of shape invariance exhibited by objects in the three-dimensional physical space.

Treatment for Long-Term Low-Dynamic Scene Changes. The second topic addresses the challenge of long-term low-dynamic scene changes. While considerable efforts have been made to mitigate the impact of high dynamics on the SLAM reasoning pipeline, treatments for long-term, low-dynamic scenarios have received comparatively less attention.

Recognizing that the motion of vehicles or pedestrians can be identified through motion blur in camera observations and subsequently addressed through semantic classification or motion models, long-term dynamics present a more intricate challenge. These dynamics cannot be independently determined from a few consecutive frames and may interact with the SLAM system. Therefore, change detection emerges as a critical component in achieving a robust SLAM system capable of sustained long-term operation. Moreover, by embracing novel object/scene representations, our SLAM system naturally acquires object-level reasoning capabilities, facilitating the detection of changes, which always occur at the object level.

Mechanism for Smooth Integration of Objects. The final topic delves into the mechanism of seamlessly integrating object-level information into the SLAM pipeline, focusing on the aspects of *ambiguity* and *compatibility*. Present-day object-based SLAM systems are typically not self-contained, often requiring external object pose estimators to generate object pose measurements. They then rely on optimizable object code embeddings coupled with backend optimization to estimate camera poses. These approaches can suffer from object pose ambiguity and may not be readily integrable to off-the-shelf SLAM systems.

Recognizing the remarkable performance of traditional SLAM approaches in static environments and taking into account the occasional absence of objects within the camera's viewing frustum, we decide to enhance the SLAM system with object-level reasoning capabilities. This enhancement can be accomplished through the adoption of highly compatible object integration mechanisms. Such integration can be realized either through the design of a standalone object-based SLAM system or by seamlessly transforming a traditional SLAM system into an object-assisted hybrid variant using the proposed object integration strategy.

Consequently, we first tackle the ambiguity issue through a multi-hypothesis approach. Subsequently, we explore the design of an SE(3)-equivariant object representation to derive SE(3) inter-frame camera pose constraints that are fully compatible with general pose graph optimization.

Therefore, our whole line of efforts revolves around two kinds of "effectiveness", i.e., the effectiveness in object representation and then in object integration, so that object-level information can be smoothly and correctly digested by a SLAM system to achieve both robust and long-term operation.

Chapter 3

A Multi-hypothesis Approach for Object Pose Estimation

We begin our story by looking at one of the frequently adopted object measurements in object-based SLAM, 6D object poses. By encoding robot-landmark geometric constraints in a compact form, 6D object pose is a desirable type of object measurement to be leveraged. However, measurement ambiguity then arises as objects may possess complete or partial object shape symmetry (e.g., due to occlusion), making it difficult or impossible to generate a consistent object pose estimate. One idea is to generate multiple pose candidates to counteract measurement ambiguity.

In this chapter, we present a novel approach that enables the object-based SLAM system to reason about multiple pose hypotheses of an object and synthesize this locally ambiguous information into a globally consistent robot and landmark pose estimation formulation[1]. In particular, we (1) design a learned pose estimation network that provides multiple hypotheses about an object's pose; (2) by treating the output of our network as components of a mixture model, we incorporate pose predictions into a SLAM system which, over successive observations, recovers a globally consistent set of robot and object (landmark) pose estimates. We evaluate our approach on the popular YCB-Video Dataset and a simulated video featuring YCB objects. Experiments demonstrate that our approach effectively improves the robustness of

[1]Supplementary video: https://youtu.be/E4sheabxWBI

object-based SLAM in the face of pose ambiguity.

3.1 Introduction

In object-based Simultaneous Localization and Mapping (SLAM), by encoding abundant robot-landmark geometric constraints in a compact form, 6D object poses tend to be a desirable type of object measurements to be leveraged [103]. By perceiving the world in terms of objects, robots could readily integrate these 6D object pose measurements into the joint recovery of robot and landmark poses. However, when there exists environment ambiguity due to complete or partial object (landmark) shape symmetry (e.g., from occlusion), the front end may be misled to make false positive pose measurements. For instance, the front end may throw out a random measurement of the mug's orientation when its handle is obscured. Such a pose measurement may incur considerable estimation errors in the back end if used to recover robot and landmark poses.

One way to counteract the effect of ambiguity is to generate multiple hypotheses each time, hoping that the correct hypothesis could be covered. As a result, rather than conducting estimation based on one single frontend measurement at a time, as most current object-based SLAM systems do, we choose to enable the object-based SLAM system to be aware of the potential environment ambiguity by generating multiple possible object pose hypotheses. In this way, by considering multiple hypotheses across many observations, an object-based SLAM system could then employ past evidence to *disambiguate* the true pose of the landmark (object) and recover a globally consistent set of robot and object pose estimates.

This idea motivates the design of our approach. The goal here is to have an object-based SLAM system capable of reasoning about multiple pose hypotheses of an object, and by synthesizing these sets of locally ambiguous information into a joint multi-hypothesis robot/landmark pose estimation setting, the system could solve for a globally consistent set of object and robot poses. Therefore, we instantiate our approach as follows (see Fig. 3-1):

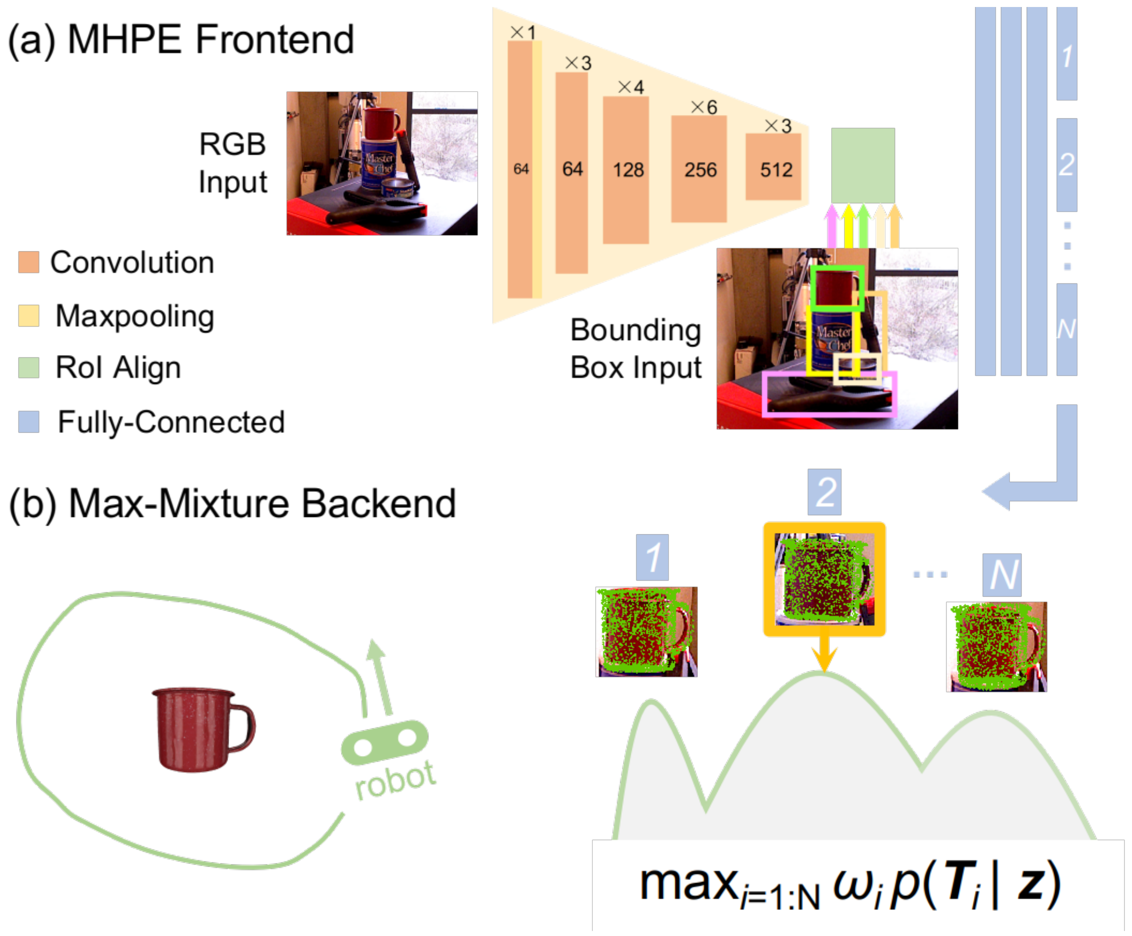


Figure 3-1: Approach overview. The system takes in an RGB image along with its bounding box detection and outputs the optimized robot/landmark pose estimates. (a) The front end's multi-hypothesis 6D object pose estimator (MHPE) comprises three components: feature extraction, region-of-interest (RoI) Align, and a multilayer perceptron. The output is N 6D pose hypotheses for each object, coming from N separate branches in the last layer. (b) The max-mixture back end models the multi-modality embedded in the multi-hypothesis output. With the MHPE covering the correct object pose (orange) in most of the observed frames, max-mixtures could gradually converge to the dominant mode, corresponding to the correct estimates of robot/landmark poses.

- To receive ambiguity-aware measurements from the front end, we develop a deep-learning-based multi-hypothesis 6D object pose estimator (MHPE) trained with known object CAD models to explore the potential pose hypothesis space.
- To solve the joint pose optimization with the multi-hypothesis formulation in the back end, we treat the multiple measurements as components of a mixture model and exploit nonlinear least square optimization to solve the max-mixture formulation [91] of pose hypotheses.

Experiments are conducted on the real YCB-Video Dataset and a synthetic video sequence featuring YCB objects in a virtual environment. Results demonstrate our approach's effectiveness in improving the robustness of the object-based SLAM system in the face of ambiguous measurements.

3.2 Related Work

We first walk through relevant approaches at the time of publication on 6D object pose estimation and object-based SLAM.

3.2.1 6D Object Pose Estimation

ICP [5] is the most commonly used method for 6D object pose estimation. However, it is prone to local minima due to non-convexity. [52, 2, 47] used probabilistic approaches to improve resilience to uncertain data.

For learning-based methods, SegICP [133] integrated semantic segmentation and pose estimation using neural networks, which achieved robust point cloud registration and per-pixel segmentation at the same time. The Deep Closest Point (DCP) [129] and Partial Registration Network (PRNet) [130] methods incorporated descriptor learning into the ICP pipeline so that the model could learn matching priors from the data.

PoseCNN [135] and SSD-6D [59], as two pioneering methods for estimating object poses in cluttered scenes, used neural networks to perform object detection and pose estimation at the same time. DOPE [123] employed only synthetic training data for robot manipulation tasks, and the network showed good generalization to novel environments. Densefusion [127] fused pixel-wise dense features from RGB and depth images and achieved strong pose estimation results.

While all these above methods achieve good pose estimation accuracy, they either need to explicitly address the object shape ambiguity or are too slow to produce object pose measurements as a SLAM front end.

3.2.2 Object-based SLAM

SLAM++ [103] used pixel-wise reprojection error and added 6D object poses as camera-object pose constraints to the pose-graph optimization. However, it required object CAD models at runtime and was limited to few categories of pre-selected objects. QuadricSLAM [87] managed to estimate camera poses and quadric parameters together based on odometry and bounding box detections. CubeSLAM [138] conducted single-view 3D cuboid object detection and incorporated its refinement into the joint optimization of camera poses and objects. While these two works improved the camera pose estimation by including constraints from novel representations of the objects, they did not explicitly consider the impact of shape ambiguity on these novel representations. PoseRBPF [20] used a Rao-Blackwellized particle filter for object tracking, providing robustness to shape symmetry with sufficient particles, but at the cost of computation speed.

3.2.3 Uncertainty-aware SLAM Back End

Uncertainty in SLAM involves many types of problems, such as loop closure and data association, where different backend methods have been developed. FastSLAM [81] tackled unknown data association in a particle-filter-based fashion, using sampling to cover the overall probability space. Sünderhauf et al. [117, 118] developed methods to change the graph topology and achieved improved robustness to outliers with switchable constraints. The max-mixture method developed by Olson and Agrawal [91] represented discrete hypotheses as components of max-mixtures; this approach has recently been leveraged for unknown data association in the context of semantic SLAM by Doherty et al. [26].

With object (landmark) poses as measurements, our work enhances the robustness of object-based SLAM in ambiguous environments by effectively generating multiple pose hypotheses from a learned pose estimation network and using max-mixtures to efficiently solve for the set of optimal robot/landmark poses.

3.3 Front End: Multi-hypothesis 6D Object Pose Estimator (MHPE)

The MHPE front end takes in an RGB image together with its object bounding boxes (which could come from any object detectors) and outputs N hypotheses for the 6D pose of each object detected in the scene.

The object pose is represented by rotation $\mathbf{q} = (w, x, y, z)$ as a unit quaternion and 3D translation $\mathbf{t} = (t_x, t_y, t_z)$, which are defined w.r.t. the current camera frame. As $\mathbf{q} = (w, x, y, z)$ and $\mathbf{q} = (-w, -x, -y, -z)$ are equivalent, we enforce $\mathbf{q}$ to have only non-negative real part w.

3.3.1 MHPE Network

The network processes the input data in three stages: feature extraction, region-of-interest alignment (RoI Align), and a multilayer perceptron (MLP) for final pose estimation (see Fig. 3-1).

We use ResNet-34 [40] without the last average pooling layer to get pixel-level feature embeddings. The feature maps, as well as corresponding bounding boxes, are fed into the RoI Align layer. First introduced by MaskRCNN [41], this layer performs bilinear interpolation on the image feature embeddings and produces feature maps of the same shape for each bounding box region. The last MLP component is divided into two parts, one for rotation estimation and the other for translation. For N pose hypotheses for each object, the output size of the last fully connected layer is $4N$ for rotation (w, x, y, z) and $3N$ for translation (t_x, t_y, t_z). We apply Softplus to ensure non-negativity for "w" values. The final unit quaternion is obtained by normalizing the four outputs from the rotation branch.

3.3.2 Learning Objective for MHPE Outputs

The MHPE network aims to minimize the average distance (ADD loss [46]) between the two object point sets, transformed by the predicted and ground truth poses, respectively. The ADD loss is defined as:

$$l = \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x} \in \mathcal{X}} \|(\hat{\mathbf{R}}\mathbf{x} + \hat{\mathbf{t}}) - (\mathbf{R}\mathbf{x} + \mathbf{t})\|, \tag{3.1}$$

where $\mathbf{R} \in \mathrm{SO}(3)$, $\mathbf{t} \in \mathrm{R}^3$, $(\mathbf{R}|\mathbf{t})$ and $(\hat{\mathbf{R}}|\hat{\mathbf{t}})$ are the ground truth and estimated object poses, and $\mathbf{x}$ is the 3D point in the object point sets $\mathcal{X}$ from the known CAD model. This metric performs well for the common single-hypothesis network. However, it needs adaption for the MHPE case, as different treatments of the individual loss in the hypothesis set may lead to distinct optimization directions.

Here, our pose estimation network is designed in a "mixture-of-expert" fashion [38], where the N hypotheses are made independently based on a shared set of feature maps. Thus, it is ideal to have each output branch acquire some specialty in certain output domains, helping improve the estimation accuracy and shed extra light on possible causes of pose ambiguity.

In this spirit, we choose to apply Eq. (3.1) in a winner-takes-all fashion, which is formulated as:

$$\begin{aligned} \hat{\mathbf{P}}_i &= \left\langle \hat{\mathbf{P}}_i^{(1)}, \ldots, \hat{\mathbf{P}}_i^{(N)} \right\rangle \\ L_i\left(\hat{\mathbf{P}}_i\right) &= \underset{j \in [1,N]}{\operatorname{argmin}}\ l_i(\hat{\mathbf{P}}_i^{(j)}), \end{aligned} \tag{3.2}$$

where $\hat{\mathbf{P}}_i^{(j)} = (\mathbf{R}(\mathbf{q}_i^{(j)})|\mathbf{t}_i^{(j)})$ is the j^{th} pose estimate in the output ensemble for the i^{th} input and $\mathbf{R}$ is expressed as the rotation matrix from quaternion $\mathbf{q}$.

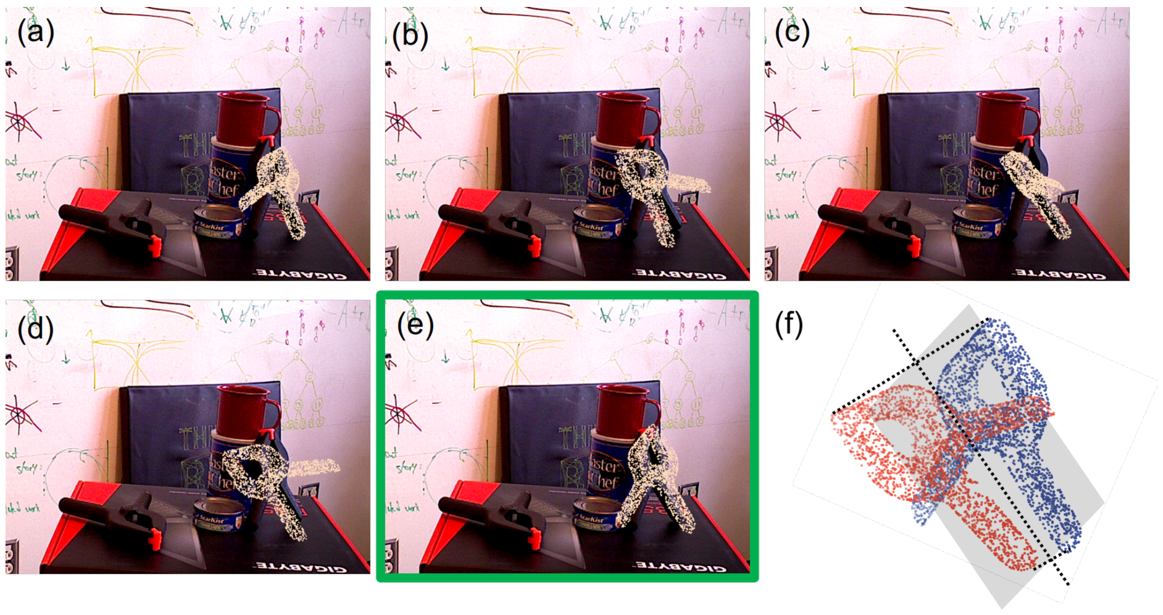


Figure 3-2: Hypothesis diversity from the MHPE front end on the YCB object clamp: Hypotheses (a) & (d) and (b) & (e) are two quasi mirror images; hypothesis (c) is spurious, and the green-boxed hypothesis (e) gives a relatively good estimate. (f) illustrates that the 3D point clouds of the mirror image pair could incur similar distance loss w.r.t. the grey mirror plane.

The defined loss function, L_i for input i, only penalizes the most accurate estimation, regardless of "how bad" the rest of the hypotheses are. Compared to averaging, this treatment prevents other worse estimations from vanishing and adds some randomness, hence diversity, to the multi-branch estimator. Those branches with an initialization too far away from the minimum could thus slowly evolve to different domains of the output space, hedging their bets while refraining from paying the price of making the wrong tentative decision. Fig. 3-2 shows one of our diverse pose estimates of a symmetrical clamp from the YCB-Video Dataset, illustrating the benefit of encouraging multiple hypotheses in different domains.

3.4 Back End: Max-mixture-based Multi-hypothesis Modeling and Optimization

Compared to its common single-hypothesis counterpart, the MHPE front end can better tackle the ambiguity within the current observation by giving multiple pose

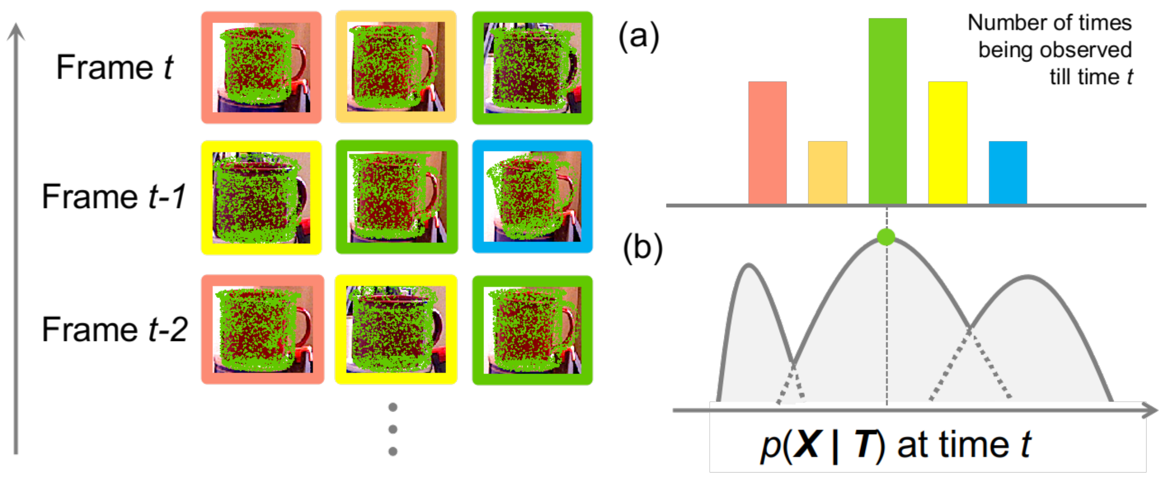

Frame t
Frame t-1
Frame t-2
(a)
Number of times being observed till time t
(b)
$p(X \mid T)$ at time t

Figure 3-3: Max-mixtures for processing multiple pose hypotheses. Left half: current pool of pose hypotheses (T_i) for the landmark "mug" with the same hypothesis in the same boundary color till time t. (a) Number of times each T_i has been observed so far, which is positively correlated to the posterior distribution of robot/landmark poses, $p(\mathbf{X}|\mathbf{T})$, where $\mathbf{T}$ denotes all multi-hypothesis measurements. Hence the significance of *consistent inclusion* of the *correct pose* within the hypothesis set. Otherwise, max-mixtures may converge to a sub-optimal hypothesis. (b) Schematic probability distribution of three pose hypotheses for $\mathbf{X}$.

candidates. Nevertheless, this leads to exponential computation complexity for picking a pose hypothesis combination to produce a sequence of optimal pose estimates.

To efficiently solve the optimization problem under the multi-hypothesis formulation, we here adapt max-mixtures to model multi-modal measurements and implicitly seek out consistent landmark pose hypotheses via optimization. Previously, the max-mixture method has been leveraged to solve data association problems in SLAM [91, 26] for its effectiveness in addressing a large number of outliers in the least squares SLAM formulation. Thus, by relying on max-mixtures' ability to distinguish the better-behaved hypothesis, the back end can recover a globally consistent set of robot/landmark poses out of an exponentially increasing number of ambiguous measurement combinations (see Fig. 3-3).

Generally, estimating poses from a sequence of observations can be formulated as a maximum likelihood estimation (MLE) problem:

$$\hat{\mathbf{X}} = \arg\max_{\mathbf{X}} \prod_i \phi(\mathbf{z}_i \mid \mathbf{X}_i), \tag{3.3}$$

where $\mathbf{X}_i$ denotes the latent variables involved in the i^{th} input, $\mathbf{z}_i$ denotes the corresponding observation, and $\phi(\cdot)$ is the likelihood function for this observation. The set of all latent variables, $\mathbf{X}$, is the camera and object pose sequence being jointly optimized in the back end. For each observation, the additive measurement noise is modeled as the canonical Gaussian noise. With N hypotheses, a natural extension of the noise model is thus the sum of Gaussian mixtures:

$$\phi(\mathbf{z}_i \mid \mathbf{X}_i) = \sum_{j=1}^{N} \omega_{ij} \mathcal{N}(\mathbf{z}_{ij}; h(\mathbf{X}_i), \mathbf{\Sigma}), \tag{3.4}$$

in which each component corresponds to a hypothesis. With an assumption of equally probable hypothesis, the weight ω_{ij} for component j is $\frac{1}{N}$. The measurement of interest here is the relative pose of the object w.r.t. the camera frame: ${}^{c}_{o}\mathbf{P} = h(\mathbf{X}_i) = h({}^{w}_{o}\mathbf{P},\ {}^{w}_{c}\mathbf{P}) = {}^{c}_{w}\mathbf{P}{}^{w}_{o}\mathbf{P}$, which is a function of camera pose ${}^{w}_{c}\mathbf{P}$ and object pose ${}^{w}_{o}\mathbf{P}$ in the world frame.

Since Gaussian mixtures fall outside the common nonlinear least-squares optimization approaches to SLAM, we here harness the max-mixture [91] factor to model multiple pose hypotheses, which writes as:

$$\phi(\mathbf{z}_i|\mathbf{X}_i) = \max_{j,i=1:N} \omega_{ij} \mathcal{N}(\mathbf{z}_{ij}; h(\mathbf{X}_i), \mathbf{\Sigma}). \tag{3.5}$$

Max-mixtures switches to the "max" of all probability, retaining the better-behaved hypothesis while keeping the problem still within the realm of Gaussian distribution optimization. By optimizing Eq. (3.3) for $\mathbf{X}$, we evaluate all the Gaussian components in Eq. (3.5) and identify the most likely one as the hypothesis contributing to the estimate in effect, whose value will then be incorporated for future joint optimization.

3.5 Experiments and Results

In this section, to improve object-based SLAM results in the ambiguous environment setting, we would like to answer two questions: (1) Can MHPE predict a set of hypotheses that do contain the close-to-ground-truth object pose? (2) Can max-mixtures recover a globally consistent set of robot/landmark poses from MHPE measurements? To answer (1), we evaluate our approach on the YCB-Video benchmark regarding distance loss and processing speed. For (2), we test our approach on a simulated video made with YCB objects for SLAM results, where the camera executes a much larger trajectory.

3.5.1 Datasets

YCB-Video Dataset. The YCB-Video Dataset [135] is built on 21 YCB objects [9] placed in various indoor scenes and consists of 92 video sequences with 133,827 images in all. Each sequence provides RGB-D images, object segmentation masks, and annotated object poses. Following [135], we divide the whole dataset into 80 videos for training and select 2,949 keyframes from the rest of the 12 videos for testing. An extra set of 80,000 synthetic images created by [135] is also included in the training set.

AirSim Simulated Video. Since the camera motion in the YCB-Video Dataset is relatively small compared to a typical SLAM-application scenario and its environment does not involve much observation ambiguity, we create a virtual environment of five YCB objects with some shape symmetry placed around a cuboid in Unreal Engine [30] (see Fig. 3-4). AirSim [108] car simulator is adopted to simulate a robot-mounted camera circling around the objects, thus making observations in an ambiguous environment setting. The observations provide RGB images, object segmentation masks, and automatically annotated 6D poses of the five objects.

3.5.2 Metrics

To facilitate comparison within the YCB-Video Dataset benchmark, we follow two metrics used in [135] and [20], i.e., ADD (introduced in Section 3.3.2) and ADD-S. The ADD-S metric, i.e., the average distance between the point in the estimation-transformed point set $\mathcal{X}_1$ and its closest point in the ground truth-transformed point

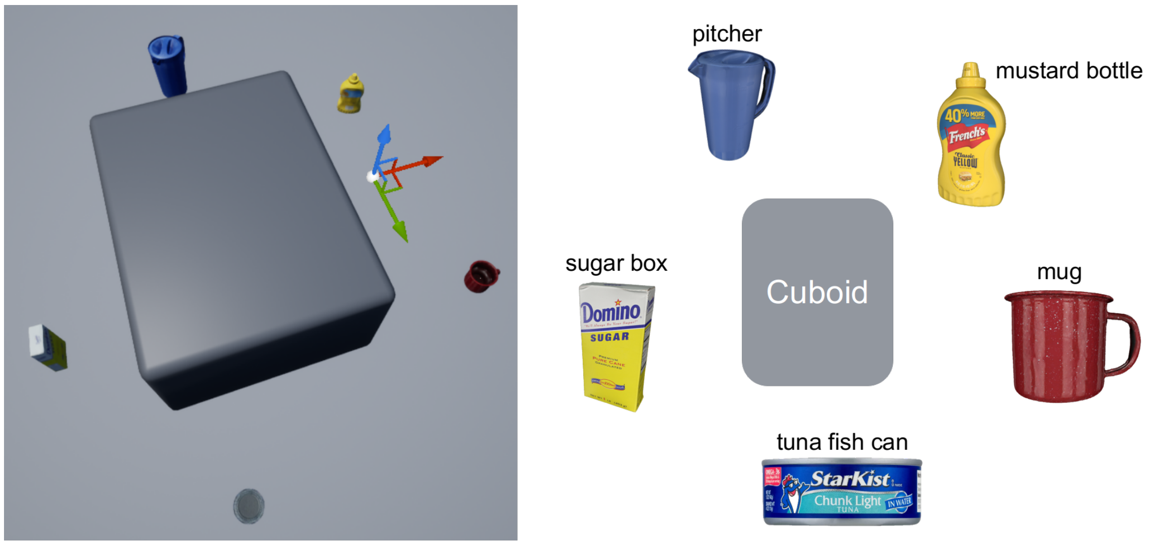


Figure 3-4: Simulated video top view and YCB object layout. Some YCB objects are deliberately chosen to bear more or less *shape symmetry* (the mug and the tuna fish can), thus contributing to an *ambiguous environment setting* for ground-level observations. Note that objects are also scaled up to 50 times their original size to match the observer's size, i.e., the AirSim car simulator, in the video.

set $\mathcal{X}_2$, is defined as:

$$l = \frac{1}{|\mathcal{X}_1|} \sum_{\mathbf{x}_1 \in \mathcal{X}_1} \min_{\mathbf{x}_2 \in \mathcal{X}_2} \|(\hat{\mathbf{R}}\mathbf{x}_1 + \hat{\mathbf{t}}) - (\mathbf{R}\mathbf{x}_2 + \mathbf{t})\|, \tag{3.6}$$

where $\mathcal{X}_1$ and $\mathcal{X}_2$ are transformed from the object model point set.

The ADD metric reflects pose estimation accuracy, while ADD-S focuses on shape similarity and is thus appropriate for evaluating symmetrical objects. Following [135], we set the distance threshold to 10 cm and compute the area under the ADD and ADD-S curves (AUC).

3.5.3 Implementation Details

The MHPE network is implemented in PyTorch. We initialize ResNet-34 with weights pretrained on ImageNet. The final MLP module comprises three fully connected layers of size 512, 512, and 256. The number of output hypotheses is set as $N = 5$. We use a batch size of 64 for training and train 100 epochs with the Adam optimizer. The initial learning rate is 0.0001 with a decay rate of 0.1 at epoch 50. The backend max-mixture algorithm is implemented in C++ using the Robot Operating System [96] and the iSAM2 [57] implementation from the GTSAM library [18]. All training and testing are conducted on a laptop with an Intel Core i7-9750H CPU and an Nvidia GeForce RTX 2070 GPU.

3.5.4 Results on YCB-Video Dataset

We report our quantitative and qualitative object pose estimation results on the YCB-Video Dataset.

Quantitatively, we compare our MHPE results to those of PoseCNN and the more recent PoseRBPF [20], all with RGB inputs (see Table 3.1). We choose the PoseRBPF-50-particle variant, which is of similar processing speed to ours and is therefore suitable for SLAM applications.

In particular, for MHPE results, we compare our hypothesis with the lowest ADD loss in each set (best hypothesis) against the single output from PoseCNN and PoseRBPF, as in our problem, it is the estimation accuracy of the best hypothesis, rather than the average quality of the multi-hypothesis set, that indicates the existence of a consistent hypothesis mode for max-mixtures to switch to, and hence the possibility of recovering the true robot/landmark poses.

Qualitatively, we display some sample visualizations of the object pose estimate in Fig. 3-5 for comparison between PoseCNN and the best hypothesis of our MHPE results. Since PoseRBPF does not provide multi-object pose estimation results on the YCB-Video Dataset, we do not include them in Fig. 3-5.

Overall Performance. Table 3.1 presents the ADD and ADD-S AUC values of all the 21 objects in the YCB-Video Dataset and the rough processing speed of each approach. We can conclude that with similar processing speed (ours: 23.0 FPS & poseRBPF: 20.6 FPS), the best hypothesis from the MHPE front end outperforms PoseCNN and PoseRBPF by 18.6%, 11.6% on ADD, and 9.4%, 11.1% on ADD-S, respectively. This demonstrates the effectiveness of our multi-hypothesis strategy in improving pose estimation accuracy and the reliability of providing a valid hypothesis set for supporting the backend optimization of robot/landmark poses.

Table 3.1: Quantitative ADD and ADD-S AUC results on YCB-Video Dataset. Symmetrical objects are in bold. The best ADD and ADD-S AUC values for each object are in bold (excluding ICP-refined results).

	RGB						RGB + Depth Image (for ICP)	
	PoseCNN		PoseRBPF 50 particles		MHPE Best Hypothesis		MHPE Best Hypothesis + ICP	
Frame Rate (FPS)	5.9		20.6		32.3		2.1 (CPU-only)	
Objects	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S
002_master_chef_can	50.9	84.0	56.1	75.6	**67.9**	**93.8**	74.3	90.0
003_cracker_box	51.7	76.9	**73.4**	**85.2**	67.8	82.9	86.8	91.6
004_sugar_box	68.6	84.3	73.9	86.5	**83.1**	**91.3**	97.4	98.2
005_tomato_soup_can	66.0	80.9	71.1	82.0	**79.5**	**92.2**	38.1*	66.9*
006_mustard_bottle	79.9	90.2	80.0	90.1	**81.6**	**90.8**	98.0	98.7
007_tuna_fish_can	70.4	87.9	56.1	73.8	**78.0**	**92.5**	87.0	95.8
008_pudding_box	**62.9**	**79.0**	54.8	69.2	45.4	71.5	83.7	92.7
009_gelatin_box	75.2	87.1	**83.1**	**89.7**	76.1	87.8	97.2	98.4
010_potted_meat_can	59.6	78.5	47.0	61.3	**69.1**	**85.5**	66.4	78.4
011_banana	72.3	85.9	22.8	64.1	**87.7**	**93.7**	93.8	97.3
019_pitcher_base	52.5	76.8	74.0	87.5	**76.8**	**88.8**	96.9	98.1
021_bleach_cleanser	50.5	**71.9**	**51.6**	66.7	47.7	70.3	88.8	94.1
024_bowl	6.5	69.7	26.4	**88.2**	**40.2**	80.1	73.6	95.8
025_mug	57.7	78.0	**67.3**	**83.7**	40.6	72.8	69.3	91.1
035_power_drill	55.1	72.8	**64.4**	**80.6**	39.5	71.2	61.1	81.7
036_wood_block	31.8	65.8	0.0	0.0	**64.6**	**85.5**	85.2	93.0
037_scissors	35.8	56.2	20.6	30.9	**64.5**	**88.9**	63.5	88.9
040_large_marker	58.0	71.4	45.7	54.1	**81.1**	**90.6**	89.5	96.2
051_large_clamp	25.0	49.9	27.0	**73.2**	**49.2**	70.7	31.5	51.8
052_extra_large_clamp	15.8	47.0	**50.4**	**68.7**	8.6	47.4	9.7	31.3
061_foam_brick	40.4	87.8	**75.8**	88.4	75.1	**92.6**	90.5	97.2
All	53.7	75.9	57.1	74.8	**63.7**	**83.1**	72.6	85.2

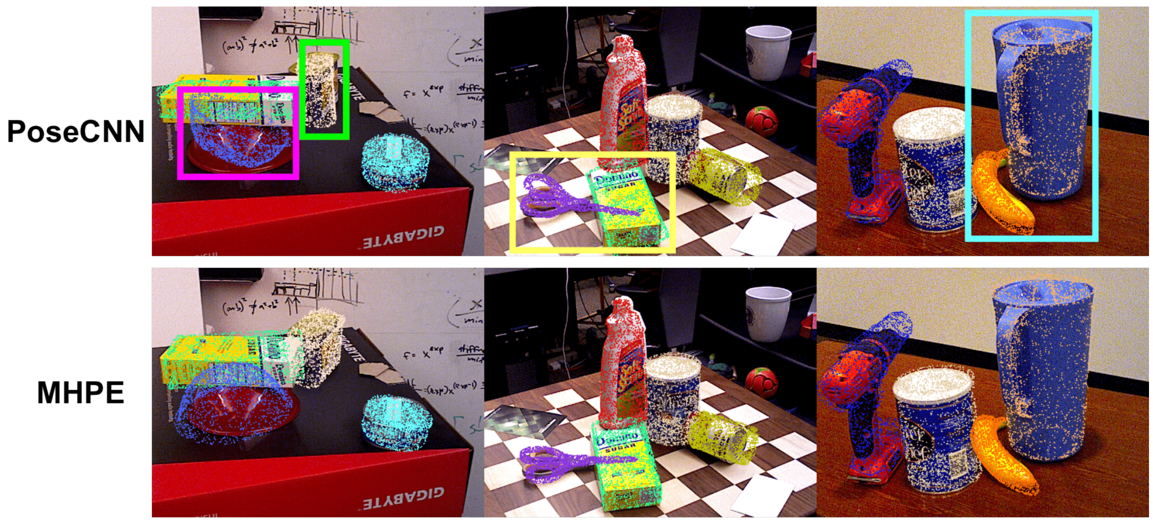


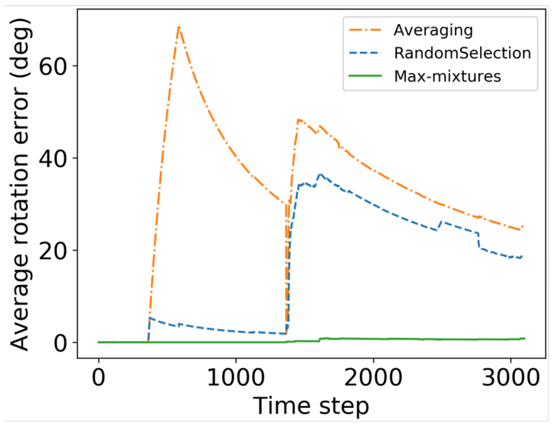
Figure 3-5: Qualitative results of object pose estimation. Each object pose is annotated with the 2D image frame projection of a colored point cloud transformed by the pose estimate. Colored bounding boxes mark improved areas in the first PoseCNN results row.

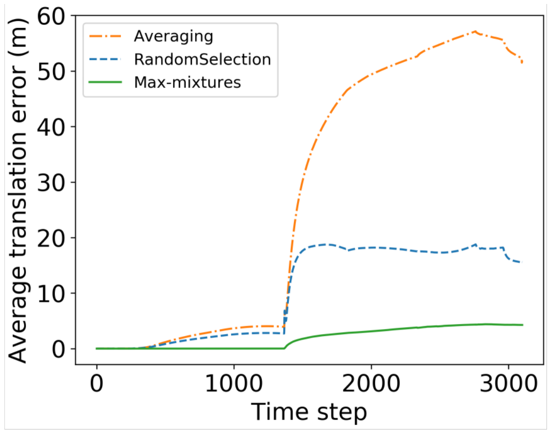
As shown by Fig. 3-5, when compared to PoseCNN, the best hypotheses from MHPE give more accurate pose estimations for the upside-down bowl in the first scene, the scissors and sugar box in the second, and the banana and the blue pitcher in the last scene.

Dealing with Ambiguity. From Table 3.1, it is observed that the MHPE network performs better on most of the symmetrical objects (marked in bold). This proves the effectiveness of our object pose estimation strategy in dealing with shape ambiguity, as it is easier for symmetrical objects to have ambiguous poses, which may induce the network to generate random false positive hypotheses.

We further explain this with the "clamp" example in Fig. 3-2. The five pose hypotheses rendered here are MHPE's estimates for the clamp pose, which include the correct estimate, hypothesis (e). Considering the diversity of the hypothesis set, which consists of seemingly inaccurate and mirror-image pair hypotheses, a single-output network may throw out a random estimate as any of (a) & (d) or (b) & (e) since these two mirror-image pairs tend to incur similar ADD loss during training w.r.t. the grey mirror plane (shown in (f)). Therefore, with each output branch evolving towards different domains of the solution space, MHPE can first produce a set of possible pose candidates and then let the back end pick one based on all of the accumulated observations. The generally poor performance on the extra large clamp may result from the given bounding box quality as it is hard for PoseCNN to distinguish between "large clamp" and "extra large clamp".

ICP Post-Processing. We also provide AUC results with ICP post-processing, where the refined poses achieve significant accuracy improvements. While we are not comparing them to the non-post-processing PoseCNN and PoseRBPF results here, considering the significant improvement in AUC values after ICP, we argue that our MHPE-predicted hypotheses are on average not far from the ground truth and thus effectively prevent ICP from local-minimum failure. The only exception is for the "tomato soup can" (marked with an asterisk), as in some frames, the lack of enough visible points from heavy occlusion greatly impairs the registration quality.





(a) Rotation Error (b) Translation Error

Figure 3-6: Running average of the estimation error for rotation and translation across the course of the robot motion. The ground truth solution is obtained by solving with object poses extracted from the simulator.

3.5.5 SLAM Results for the Simulated Video

We apply our approach to the simulated video and test its effectiveness in the scenario with larger camera motion. For evaluation purposes, we compare the robot/landmark pose sequence optimized from our MHPE+max-mixtures approach against those obtained by two practical approaches when dealing with multiple hypotheses: averaging and random selection. Additionally, the random selection approach also resembles the behavior of the traditional single-hypothesis approach under ambiguous settings, as the front end tends to throw out a random time-inconsistent measurement.

In addition to the YCB-Video data, the MHPE network is trained on an extra 18,523 images of a robot circumnavigating each object at different ranges to better adapt to the simulated environment. The testing is then run on a 3100-frame sequence in a similar environment, with the camera moving first in the inner and then the outer circle of the area enclosed by the five objects. The magnitude of the distance between the camera and objects is set to be around 50 times the object size.

Robot Pose Estimation. Quantitatively, in Fig. 3-6, we plot the running average of the estimation error for rotation and translation, respectively.

As shown in Fig. 3-6, we can generally conclude that max-mixtures performs the

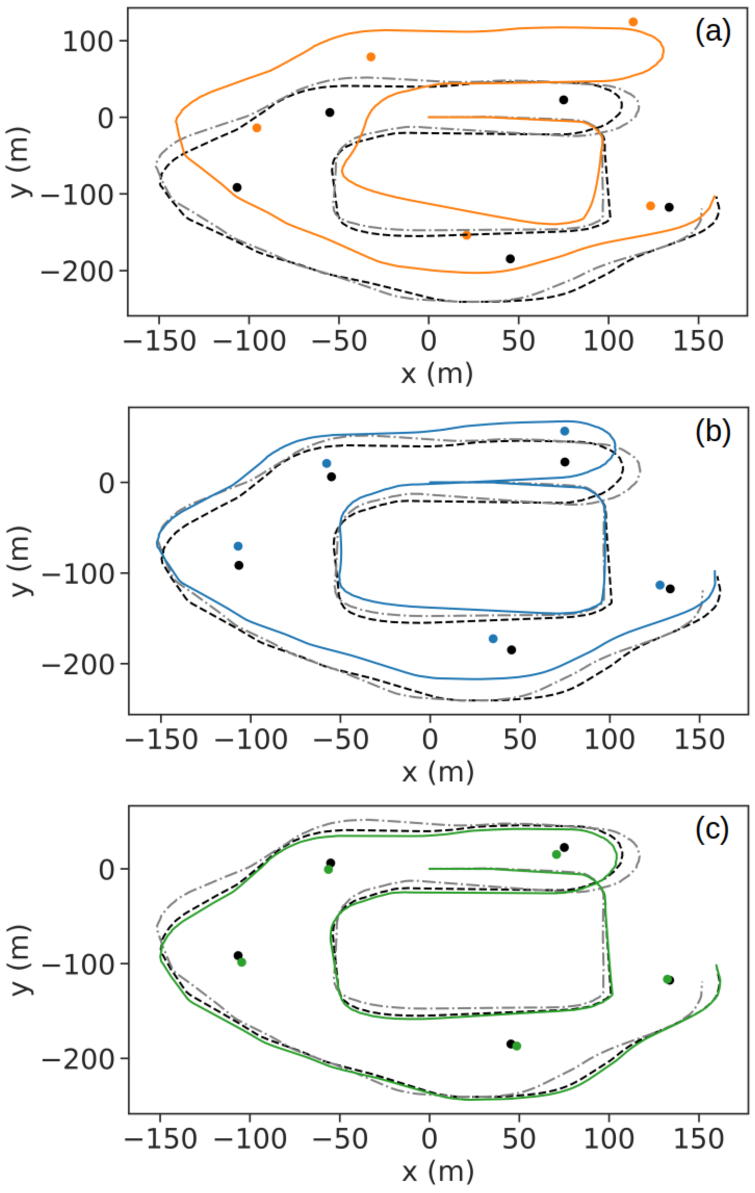


Figure 3-7: Comparison of camera trajectories and object position estimates. We use evo [37] for trajectory evaluation. Objects are shown as dots. Ground truth is shown in black. (a) Averaging; (b) Random Selection; (c) Max-mixtures.

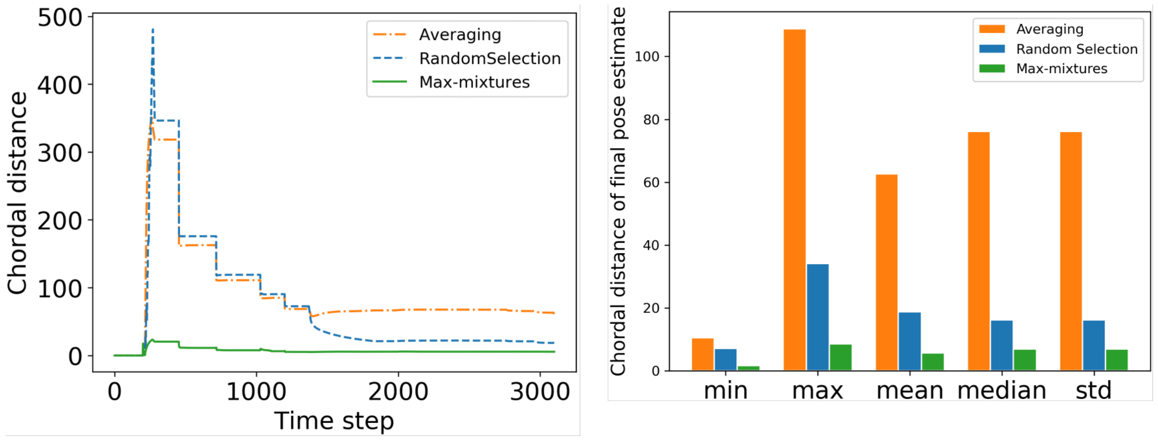
best while averaging performs worst. The discontinuities in the plot correspond to the emergence of new objects. Considering the ambiguous measurements from the deliberately selected symmetrical objects, with a continuously lower rotation/translation error, our approach shows robustness to the disturbance of the false-positive pose hypotheses. It outperforms random selection, which to some extent also indicates how an ordinary, single-hypothesis SLAM system will behave. In Fig. 3-6(b), the sudden climb in the translation error at about halfway through the course can be attributed to the reappearance of the mug in the video, and averaging obviously suffers from considerable drift due to ambiguous measurements at this loop closure.

Qualitatively, it can be observed from Fig. 3-7 that our approach outperforms averaging and random selection by a large margin in terms of estimation accuracy. We argue that by leveraging the accumulated observation from previous frames, our approach is able to recover consistent estimates from the ambiguous pose measurements, as opposed to its single-hypothesis-based counterparts.

This is especially useful for treating shape-ambiguous objects and loop closures. For example, the camera could obtain a better pose estimate of the mug by taking into account the last time the mug handle was visible, and this will, in turn, benefit loop closure detection. Furthermore, the result also implies that averaging over the candidate pool is the least effective approach, as the result of averaging over rotation matrices from the hypothesis ensemble can deviate from the actual rotation of any member hypothesis [39].

Landmark Pose Estimation. Here, we choose landmark pose estimation from the MHPE as the frontend measurements to help recover robot poses. After joint optimization for comparison, we compute the chordal distance between the ground truth and estimated landmark pose (see Fig. 3-8).

As shown in Fig. 3-8(a), we could observe a step-wise increase in error each time a new landmark is observed, followed by a gradual error decrease as optimization proceeds. Compared to the other two approaches, our approach exhibits a much less steep surge between "stairs" as though with no prior knowledge about the object, it could employ past knowledge to help gradually switch to the better-behaved mode.



(a) Running Average (b) Statistics of Final Estimate

Figure 3-8: Landmark pose estimation errors.

Around time step 1400, when the robot finishes its first circle and returns back to the mug, while averaging merges ambiguous measurements at the loop closure and hence renders the optimization off track (corresponds to the trajectory in Fig. 3-7(a)), max-mixtures manages to maintain a lower average error.

The above behaviors are also reflected in Fig. 3-8(b). As for the final estimate of object poses, max-mixtures proves its superiority in all statistics descriptions, indicating a constantly higher estimation precision amongst all three methods.

3.6 Conclusion

In this chapter, we develop a novel approach that allows the robot to reason about multiple hypotheses for an object's pose and synthesize this locally ambiguous information into a globally consistent object and robot trajectory estimation formulation. We instantiate this approach by designing a deep-learning-based front end for producing multiple object pose hypotheses and a max-mixture-based back end for selecting pose candidates and conducting joint robot/landmark pose optimization. Experiments on the real YCB-Video Dataset and synthetic YCB object video demonstrate the effectiveness of our approach to performing object pose and robot trajectory estimation in the ambiguous environment setting. Future work may involve providing extra clues to facilitate the max-mixtures optimization, such as by regressing weights for each pose hypothesis from the MHPE front end.

Chapter 4

PlaneSDF-based Change Detection for Long-term Dense Mapping

After investigating the prevalent issue of scene ambiguity, we turn to look at another key aspect of the thesis: long-term robust operation and the accompanying low-dynamic object changes over time. As world changes always happen on the object scale, object-based SLAM has a natural advantage over traditional SLAM pipelines in conducting change detection and thus achieving long-term operation.

In this chapter, we begin this exploration by first looking at *offline* change detection, a common demand for processing environment maps across multiple sessions over extended periods of time. Specifically, it is desirable for autonomous agents to detect changes amongst maps of different sessions so as to gain a conflict-free understanding of the current environment.

Here, we build our change detection scheme on top of a novel map representation, dubbed Plane Signed Distance Fields (PlaneSDF), where dense maps are represented as a collection of planes and their associated geometric components in SDF volumes[1]. Given the point clouds of the source and target scenes, the proposed three-step PlaneSDF-based change detection approach works as follows: (1) PlaneSDF volumes are instantiated within each scene and registered across scenes using plane poses; 2D height maps and object maps are extracted per volume via height projection and

[1]Supplementary video: https://youtu.be/oh-MQPWTwZI

connected component analysis. (2) Height maps are compared and intersected with the object map to produce a 2D change location mask for changed object candidates in the source scene. (3) 3D geometric validation is performed using SDF-derived features per object candidate for change mask refinement. We evaluate our approach on both synthetic and real-world datasets and demonstrate its effectiveness via the task of changed object detection.

4.1 Introduction

The ability to perform robust long-term operations is critical in many robotics and AR/VR applications, such as household cleaning and AR/VR environment scanning. Agents accumulate a more holistic understanding of their working environments through multiple traverses of the same place. However, in the long-term setting, the working environment is prone to changes over time, e.g., the removal of a coffee mug. Conflicts may then arise when agents try to synthesize scans from different sessions. Therefore, agents are expected first to capture these changes and then obtain the up-to-date 3D reconstruction of the scene after all change conflicts have been resolved.

An intuitive way to detect changes is through scene differencing between the two reconstructions of interest. Previous works on change detection leverage scene representations such as point clouds [33, 4, 44, 103] or Signed Distance Fields (SDF) [84, 76, 98, 32] and perform point- or voxel-wise comparison *globally* between the two scenes. To ensure that comparison is carried out at the corresponding locations of the two observations, these methods demand consistent and precisely aligned reconstructions, which are susceptible to sensor noises and localization errors.

We observe that most scene changes occur at the object level and that man-made environments can often be modeled as a set of planes with objects attached to them, as opposed to a cluster of unordered points or voxels with no geometric structure. Therefore, we choose to represent the whole scene as a set of planes, each having an associated SDF volume that describes the geometric details of the objects attached to it, which we term the PlaneSDF representation. Similar to the idea

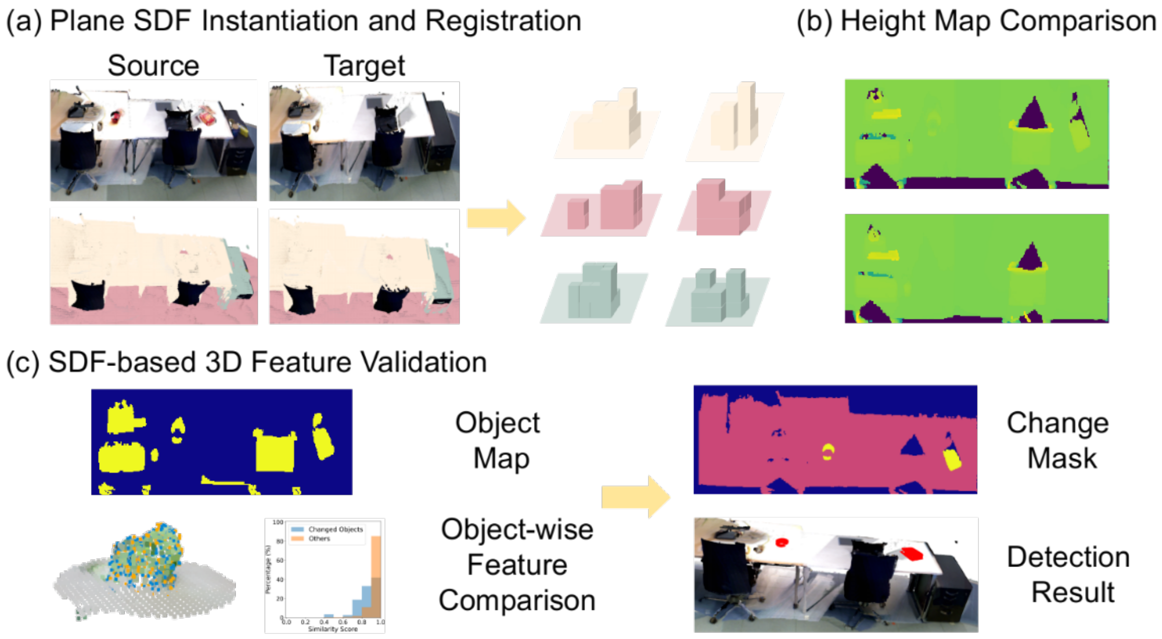

(a) Plane SDF Instantiation and Registration
Source
Target
(b) Height Map Comparison
(c) SDF-based 3D Feature Validation
Object Map
Object-wise Feature Comparison
Change Mask
Detection Result
Changed Objects
Others
Similarity Score
Percentage (%)

Figure 4-1: System overview. Input: point clouds of the source and target scene. Output: voxels of objects detected as changes between the two scenes. (a): For the two input point clouds, PlaneSDF volumes are fused and registered using poses of major planes (e.g., desk, cabinet, and the floor, as indicated in different colors). A 2D height map and an associated object map are obtained for each plane through projection and connected component analysis. (b): Height values for corresponding planes are compared, which yields a preliminary 2D change mask for the source plane w.r.t. the target plane. (c): The intersection of the current change mask and the source object map is found to determine changed object candidates. Each of these objects has its SDF-based features extracted and compared against the corresponding one in the target for change mask refinement.

of dividing the whole environment into submaps, e.g., based on time intervals [98] or objects [76, 105], agents could maintain multiple PlaneSDF volumes of scalable sizes in lieu of a single chunk of global SDF while saving update and memory reload time by updating volumes only in the current viewing frustum. Furthermore, this representation is also more robust to localization drift as local, regional correction can be performed patch by patch each time two planes from different traverses are registered via plane pose.

Taking advantage of the PlaneSDF representation, we propose a change detection algorithm given a source and a target scene that decomposes the original global comparison in a local plane-wise fashion. Treating each plane as a separator, local change detection is performed plane-wise and on the object level. The global localization drift issue between two scenes is alleviated during plane-pose registration. Through the projection of SDF voxel height values onto the plane, the obtained height map and its value connectivity offer a solid indication of the potential object candidates along with their projected 2D contours, making it possible to conduct 3D geometric validation only on SDF voxels belonging to the potentially changed objects. Our main contributions are as follows:

- PlaneSDF is proposed as a novel representation for indoor scene reconstruction.
- A change detection algorithm, consisting of 2D height map comparison and 3D geometric validation, is developed leveraging the PlaneSDF data structure.
- The effectiveness of the proposed algorithm is demonstrated on both synthetic and real-world datasets of indoor scenes.

4.2 Related Work

Change detection, as widely discussed in research concerning long-term robotic operations, can be roughly divided into two categories: geometric and probabilistic approaches. We here list some relevant works at the time of publication.

Geometric Approaches. Geometric approaches are usually based on comparing geometric features extracted from various environment representations. Walcott-Bryant et al. [125] developed Dynamic Pose Graph SLAM, where change detection was performed on the 2D occupancy grid to edit and update the pose graph. Classical 2D feature descriptors, e.g., SURF, ORB, and BRISK [22, 23], were extracted from the greyscale input images and the visual database, respectively. Next, the Euclidean distance between the two features was computed to determine if changes occurred. There are also many works in the literature that use 3D representations. Finman et al. [33] performed scene differencing on depth data among multiple maps and learned segmentation models with surface normals and color edges to discover new objects in the scene. Ambrus et al. [4] computed a meta-room reference map of the environment from the collected point cloud and employed spatial clustering based on global descriptors to discover new objects in the scene. Fehr et al. [32] adapted volumetric differencing onto a multi-layer SDF grid and showed its effectiveness in object discovery and class recognition. Kunze et al. [65] built and updated a hierarchical map of the environment by comparing object positions between observations and corresponding map contents. Schmid et al. [105] proposed a panoptic map representation using multiple Truncated Signed Distance Fields for each panoptic entity to detect long-term object-level scene changes on the fly. Langer et al. [67] combined semantic as well as supporting plane information and conducted local verification (LV) to discover objects newly introduced into the scene. The proposed method outperforms several global point- and voxel-based approaches and is selected as the baseline here for comparison.

Probability-based Approaches. Previous works in this category tend to develop statistical models to describe sensor measurements or environment dynamics. Krajnik et al. [64] modeled the environment's spatiotemporal dynamics by its frequency spectrum, while [44, 72] exploited probabilistic measurement models to indicate how likely it is for each surface element in the scene to have moved between two scenes. Bore et al. [7] proposed a model for object movement describing both local moves and long-distance global motion. Katsura et al. [58] converted point clouds and measured data into ND (Normal Distribution) voxels using the Normal Distribution Transform (NDT) and compared voxel-wise distribution similarity.

There are also learning-based change detection approaches [3, 126] that learn geometric features through neural networks trained on pre-registered images or SDF pairs. Considering the potential challenges of the availability of training data and generalization to unseen changes, we focus only on non-learning-based methods.

Despite all the results reported, the global point- or voxel-wise geometric comparisons are susceptible to sensor noises and localization errors, and the results of probabilistic approaches may not be readily applicable to scene mapping tasks. Hence, in this work, we consider 2D as well as 3D information on the voxel and object level with the proposed PlaneSDF structure and achieve robust change detection on both synthetic and real-world datasets.

4.3 Method Overview

Our method (see Fig. 4-1) leverages the plane-to-object supporting structure through the PlaneSDF representation, thereby enabling us to first perform local pairwise plane pose alignment against global reconstruction errors. We then obtain change detection results via efficient and effective local scene comparison on a 2D height map and 3D object surface geometry informed by the SDF volume.

4.3.1 PlaneSDF Instantiation

We first generate the PlaneSDF representation for each scene, i.e., representing the input 3D point cloud for the scene as a set of planes and their associated SDF volumes.

For plane detection, when given sequential point cloud streams, we extract planes from each frame with RANSAC and merge them when a new frame arrives, as how SLAM systems commonly proceed when using planes as pose estimation constraints [119, 74, 50]. When a point cloud for the complete scene is available, we run a spatial clustering algorithm [113] to detect a set of planes out of the cloud.

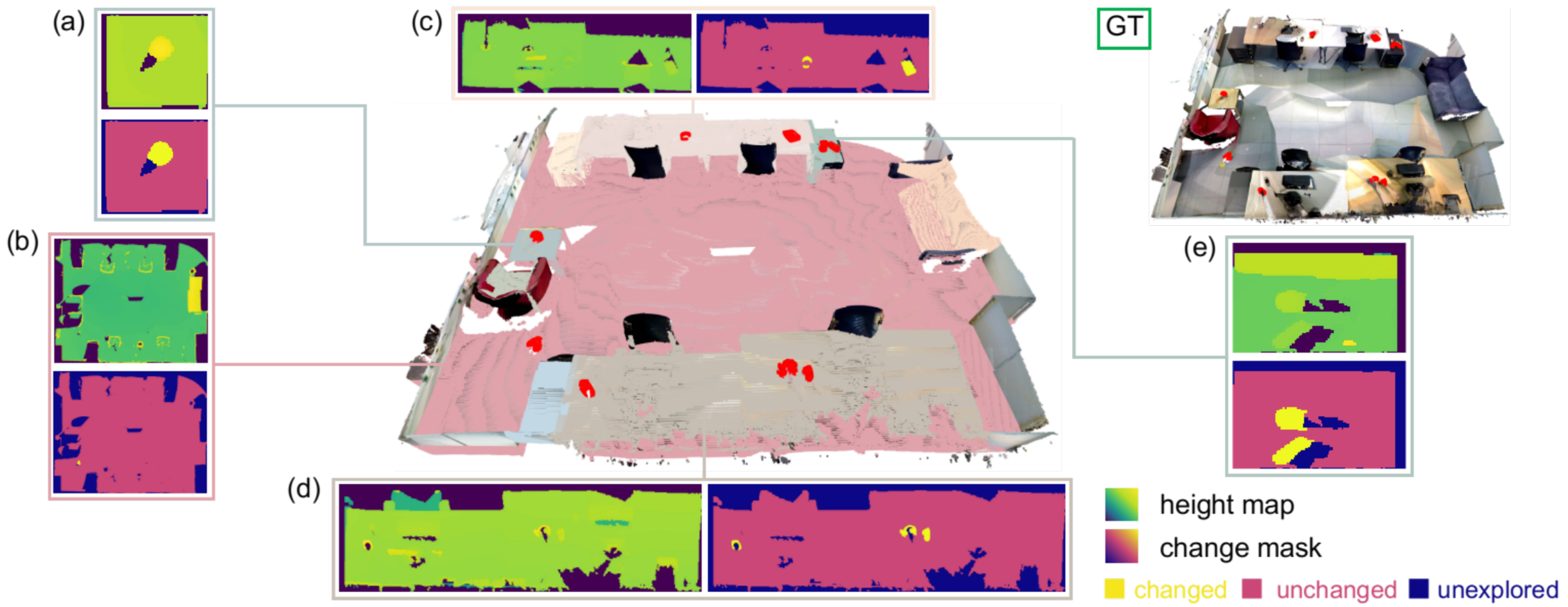


Figure 4-2: Change detection results for a complete indoor scene from the object change detection dataset [67]. The whole scene is spatially subdivided into multiple PlaneSDF instances (marked by distinct colors). Note that there could be some overlap among certain SDF volumes (e.g., the sofa's seating area in the upper right of the scene is also fused into the floor volume). For each plane of interest, i.e., planes with objects newly introduced onto them, the associated height map and the final change mask are shown. The detected object changes are colored in red, while the ground truth (GT) changes are rendered in the upper right corner of the figure.

For each plane detected, we fuse an SDF volume using all the points within a predefined distance to the plane in the hope that the obtained SDF will record the free space and object geometry solely from objects directly supported by the plane, e.g., the drawings hanging on the vertical wall or the soda can placed on the table. Note that when two detected planes are less than the defined fusing distance away from each other, or there are bigger objects supported by multiple planes, a point could be fused into multiple PlaneSDF instances, e.g., the color overlap of the sofa and the floor instances in Fig. 4-2). We also limit our detection of planes to only horizontal and vertical ones, as they constitute most of the "plane-supporting-objects" cases we encounter daily.

Furthermore, the local 2D height grid map evaluated w.r.t. the plane is computed, where each grid stores the maximum voxel-to-plane distance in the height direction at the current plane location. The height map is non-zero for plane locations occupied by objects, zero for flat unoccupied locations, and -1 for unobserved regions. Building on top of this, as non-zero regions are disconnected from each other by the plane zero-level set, we could quickly obtain an "object" or "object cluster" (for multiple small objects close to each other) map (Fig. 4-1(d)) preserving relatively accurate object contours through connected component labeling on the height map.

Given two PlaneSDF volumes, a source and a target, instantiated from the two scenes respectively, we define the 2D change mask of the pair as a ternary mask of the same size as the *source* height map, indicating all changed plane locations in the *source* w.r.t. the *target* (Fig. 4-2).

4.3.2 PlaneSDF Registration

Before scene differencing is conducted, PlaneSDF volumes of the two scenes are first registered so that the comparison is guaranteed to be carried out on two observations of the same plane. With the assumption that input point clouds from different sessions share the same world coordinate frame, PlaneSDF volumes are registered through plane poses to alleviate the effect of localization drift among reconstructions of the same plane. For each pair of PlaneSDFs, we determine if they belong to the same plane according to the orientation cosine similarity and offset difference of the two plane poses:

$$\begin{aligned} &\mathbf{n}^{\mathbf{T}}\mathbf{n}' \geq \delta_{\mathbf{n}} \\ &||d - d'|| \leq \delta_d, \end{aligned} \tag{4.1}$$

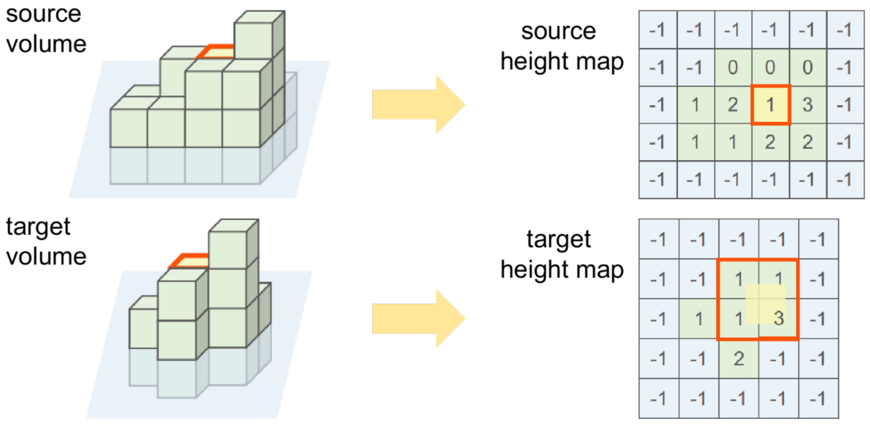
where $(\mathbf{n}, d)$ and $(\mathbf{n}', d')$ are the plane surface normals and offsets from the origin of the source and target PlaneSDF volumes, respectively. $\delta_{\mathbf{n}}$ and δ_d are the minimum cosine similarity and maximum offset distance for two planes to be regarded as the same plane. In this way, via associating plane detections of similar orientations and offsets in the pair of reconstructions, small localization drift of the same plane can be mitigated by applying the relative transform between plane poses, from which we are then ready for change detection on each registered PlaneSDF pair.

4.3.3 Height Map Comparison

As floating objects are rare in daily scenes, height value discrepancy at the same plane location in different observations can offer informative speculation about the changes on this plane, e.g., when objects are newly removed or added, drastic changes between zero and non-zero height values will occur. In this spirit, we project each location, (x, y), of the source height map H onto the target height map H' using the relative plane pose. If the height value variation is above a threshold δ_h, we mark this plane location as *changed* (see Fig. 4-3). Usually, the projected location, (x', y'), will not land exactly onto a grid center in the target map, so comparisons are drawn between the source height and those of the four nearest neighbors of (x', y'):

$$\begin{aligned} &\sum_{i=0,1; j=0,1} \mathbb{1}(|H'(\lfloor x' \rfloor + i, \lfloor y' \rfloor + j) - H(x, y)| \leq \delta_h) \\ &\qquad = \begin{cases} 0, & \text{changed} \\ \geq 1, & \text{unchanged.} \end{cases} \end{aligned} \tag{4.2}$$

In most cases, due to measurement noises, the change mask obtained after direct comparison is usually corrupted by small false positive clusters scattered around the



-1	-1	-1	-1	-1	-1
-1	-1	0	0	0	-1
-1	1	2	1	3	-1
-1	1	1	2	2	-1
-1	-1	-1	-1	-1	-1

-1	-1	-1	-1	-1
-1	-1	1	1	-1
-1	1	1	3	-1
-1	-1	2	-1	-1
-1	-1	-1	-1	-1

Figure 4-3: Height map comparison. For the registered source and target PlaneSDF pairs, each grid in the source height map is projected onto the target height map, with its height value compared against those of its closest 2×2 neighborhood. If all four neighbors have a height difference above a threshold, this grid (plane location) is preliminarily marked as changed.

map. Therefore, a round of connected component filtering followed by dilation is applied to remove the noise.

4.3.4 3D Voxel Validation

Comparing height values for changes works well when (1) objects are removed or added, inducing significant variation in height values, or (2) camera trajectories have a high observation overlap of the unchanged objects between two runs. However, height implications can fail easily when old objects are replaced with new ones in the same place or different parts of the same unchanged object are observed due to disparate viewing angles.

Therefore, 3D validation on the SDF of potentially changed source plane locations is introduced to correct false positives indicated by the change mask. For the overlapping space of two observations, if the same object persists, the local surface geometry and free space description should be similar, or the target SDF will otherwise be remarkably different from that of the source.

Here, for the sake of selecting key voxels and obtaining corresponding descriptive geometry characterization around the selected locations, the curvature-derived description of the SDF is adopted for its capability to characterize the geometry of both

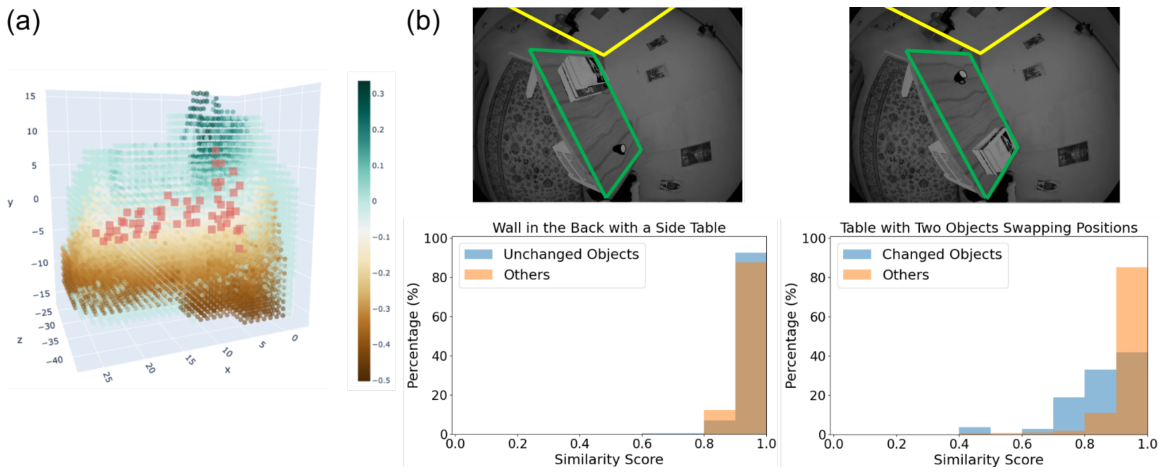


Figure 4-4: Key voxel distribution and corresponding similarity score distribution of planes with and without changes. (a) Key voxel (red square dots) distribution within a voxel blob (round dots with colors indicating the SDF value). (b) Key voxels within the same PlaneSDF volume are classified as either "part of an object" or "others" as all remaining background elements. Left (PlaneSDF of the yellow plane): Both the side table (object) and the wall (others) are unchanged. Hence, all similarity scores are biased towards higher-valued bins. Right (PlaneSDF of the green plane): The book stack and the coffee mug swap their positions on the table. Their shape distinction leads to scattered distribution of voxel similarity scores at the same 3D position, while the "other" unchanged voxels around the tabletop plane still share high similarity.

object surfaces and the unoccupied space in between. In addition to indicating the planarity, convexity, or concavity of the object surface, the trend of SDF variation amid object surfaces can reflect inter-surface spatial relations, e.g., the sudden drop of an increasing SDF value along a ray direction can imply the switch of the nearest reference surface for SDF value calculation as the ray marches through surfaces. In contrast, the raw SDF value description and its gradient-derived counterpart are less suitable for the unified goal of key voxel selection and local geometry description. The former, due to the unavailability of ground truth surfaces during point fusion, is prone to slight inconsistency when constructed from different camera trajectories, while the latter returns an indistinguishable magnitude of one by construction in most places.

Additionally, to make the comparison more robust to measurement noises and reconstruction errors, the SDF voxels of interest are extracted and compared in the minimal unit of an object (cluster). This is achieved by selecting voxel blobs in each source PlaneSDF as those whose 2D projected clusters from the change mask have high overlap with the connected clusters in the object map, i.e., the intersection of the change mask and the object map. Through per-blob 3D geometry validation, the final change mask not only preserves a more detailed object contour in cases of adding/removing an object to/from a free space but also self-corrects false per-voxel height variation induced by sensor noises in a clean way.

Key Voxel Selection. Key voxels are selected per object blob to offer a more compact and robust characterization of the overall blob shape. Inspired by [80], we select voxels around regions of high curvature as key voxels, implying neighborhoods of significant shape variations (see Fig. 4-4(a)). We adopt the measure of local extrema of the determinant of Hessian (DoH), $det(Hess(\mathbf{v}))$, and calculate the Hessian matrix within a complete $3 \times 3 \times 3$ neighborhood $\mathcal{N}$:

$$\begin{aligned} Hess(\mathbf{v}) &= \begin{bmatrix} s_{xx} & s_{xy} & s_{xz} \\ s_{yx} & s_{yy} & s_{yz} \\ s_{zx} & s_{zy} & s_{zz} \end{bmatrix} \\ s_{ij} &= (\mathbf{G}_j * \mathbf{G}_i)(\Phi(\mathbf{v})) \quad i, j = x, y, z, \end{aligned} \tag{4.3}$$

where each element s_{ij} in the Hessian matrix of $\mathbf{v}$ is obtained via convolution of $\Phi(\mathbf{v})$, the $3 \times 3 \times 3$ SDF neighborhood at $\mathbf{v}$, with the 3D Sobel filter $\mathbf{G}$ in turn in the i and j direction.

Per-voxel Shape Description. For each key voxel $\mathbf{v}_0$ in the object blob $\mathcal{O}$, the three eigenpairs of the Hessian matrix, $\mathbf{p}_i = (\lambda_i, \mathbf{e}_i)$, $i = x, y, z$, are computed and represent the three principal curvatures (λ_is) and their directions ($\mathbf{e}_i$s) at $\mathbf{v}$, respectively. This operation is then repeated for each voxel in $\mathcal{N}$ and its corresponding neighborhood $\mathcal{N}'$ in the target map (determined by its projected location $\mathbf{v}'$ in the target map). The three eigenvalues are normalized for numerical stability, and each principal direction vector $\mathbf{e}_i$ is converted into spherical coordinate (θ_i, ϕ_i).

We then construct eigenpair histograms, $\mathcal{H}$ and $\mathcal{H}'$, for the corresponding neighborhood $\mathcal{N}$ and $\mathcal{N}'$. For neighborhood $\mathcal{N}$, we compute three sub-histograms, h_is, for all the eigenpairs $\mathbf{p}_{j_i}$ in the i direction, where $i = x, y, z$:

$$\begin{aligned} \mathbf{p}_{j_i} &= [\theta_i, \phi_i, \lambda_i], \mathbf{p}_j \in \mathcal{N} \\ &\Rightarrow h_i \in \mathbb{R}^{N_\theta \times N_\phi \times N_\lambda} \\ \theta_{h_i} &= [0, 180°], \phi_{h_i} = [-90°, 90°] \\ \lambda_{h_i} &= [\min_{j\in\mathcal{N}}(\lambda_{j_i}), \max_{j\in\mathcal{N}}(\lambda_{j_i})], \end{aligned} \tag{4.4}$$

where N_θ,N_ϕ, and N_λ are the number of bins, and θ_{h_i},ϕ_{h_i}, and λ_{h_i} are the bin threshold in the θ, ϕ, and λ directions for h_i, respectively. With each h_i of dimension $N_\theta \times N_\phi \times N_\lambda$, we then concatenate the three to form the final histogram,

$$\mathcal{H} = [h_1||h_2||h_3], \tag{4.5}$$

describing the local shape distribution around this key voxel in the source. The corresponding $\mathcal{H}'$ for the target neighborhood is computed in the same fashion while sharing all the histogram thresholds with those of $\mathcal{H}$.

To further enhance its ability to characterize local shapes, we append the final histogram with a weighted signed distance value s of the neighborhood. The weights are assigned with a Gaussian filter centered at $\mathbf{v_0}$ with deviation of $\sigma = 2$, and the weighted SDF s is computed as follows:

$$\begin{aligned} w_i &= \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{(\mathbf{v}_i - \mathbf{v}_0)^2}{2\sigma^2}}, \mathbf{v}_i \in \mathcal{N}(\mathbf{v}_0) \\ s &= \frac{\sum w_i \Phi(\mathbf{v}_i)}{\sum w_i}. \end{aligned} \tag{4.6}$$

Thus, the ultimate feature vector for the key voxel in the source is $f(\mathbf{v}_0) = [\mathcal{H}, s]$, which is of dimension $3 \times N_\theta \times N_\phi \times N_\lambda + 1$. We define a similarity score, $sim \in (0, 1)$, at this key voxel between the two features, f and f', of the source and target map, respectively, as:

$$sim(f, f') = 1/(1 + \alpha\|f - f'\|_2), \tag{4.7}$$

where $f'(\mathbf{v}_0) = [\mathcal{H}', s']$ and α is a coefficient for adjusting the contribution of the

Euclidean distance between f and f', $\|f - f'\|_2$, to the similarity score.

Per-object Shape Comparison. The distribution of the similarity scores for all key voxels in the current object blob then makes it possible to determine if the space is occupied by the same object across two sessions. We argue that for an *unchanged* space occupied with the same object blob, the similarity scores, as an indication of the local shape, should be concentrated around higher values. In contrast, for a space with objects later removed, added, or replaced by another object, they should either be low (removed or added) or distributed more evenly around a wider range of bins (replaced) (see Fig. 4-4(b)). Therefore, we construct the similarity score histogram for the object blob and compute the histogram mean to determine if the object has changed:

$$\begin{aligned} &H_{avg} = \sum m_i n_i / N \\ &isChanged(\mathcal{O}) = \mathbb{1}(H_{avg} < \delta_{blob}), \end{aligned} \tag{4.8}$$

where m_i and n_i are the midpoint value and frequency of each bin i, and N is the total number of key voxels in this object blob. The object is then validated as changed if H_{avg} is below a similarity threshold, δ_{blob}, or false positives from 2D comparison can be corrected based on the relatively high H_{avg} value.

Following the plane locations marked as changed in the change mask, all the corresponding voxels along the height direction are extracted, which are the changed part of the source scene w.r.t. the target.

4.4 Experiments and Results

In this section, we evaluate our approach on both synthetic and real-world indoor datasets and demonstrate its strength via tasks revolving around object-level change detection.

4.4.1 Datasets

Synthetic Tabletop Dataset. We generate synthetic indoor sequences with known object models on a tabletop for evaluations under controlled environments. We first scan a static, furnished room with a Lidar scanner to obtain a ground-truth 3D point cloud of the room. A few synthetic daily objects, e.g., mug and book stack, are then arbitrarily placed on a synthetic table in the scene, which are added, removed, or moved across multiple sequences, thus creating the desired changes to be detected. The scenes are rendered by simulating cameras on the Oculus Quest 2 headset moving in a preset trajectory around the table, from which per-frame 3D point cloud observations are generated and used as the input to our algorithm.

Object Change Detection Dataset. The object change detection dataset [67] is recorded with an Asus Xtion PRO Live RGB-D camera mounted on an HSR robot, consisting of multiple complete or partial point clouds of five scenes: big room, small room, kitchen, office, and living room. Each scene consists of a reference reconstruction and 5 to 6 other reconstructions obtained using Voxblox [90], accompanied by various levels of permanent structure misalignment and noisy boundaries due to localization and reconstruction errors. Ground truth annotation of 3 to 18 newly introduced YCB [10] objects to the scene is provided.

4.4.2 Evaluation Metrics

We adopt the commonly used precision and recall rates as the metrics for change detection evaluation.

For the object change detection dataset, following the measures in [67], we compute precision, recall rate, and F1 score at the *point* level based on the ground truth changed point annotation and our detection results. *Precision* is computed as the proportion of the total number of detected points corresponding to the ground truth, and *recall rate* is defined as the proportion of ground truth points incorporated in the detection points. The F1 score provides the harmonic mean of the two metrics. Two other metrics, the number of missing objects (changed objects with no points detected as changed) and wrongly detected clusters (clusters generated by the method that do not overlap with any changed objects), are also reported to better manifest the approach's performance on the object/cluster level.

4.4.3 Implementation Details

We follow the procedures described in Section 4.3.1 for generating PlaneSDF instances, with the RANSAC-based approach for data streams of the synthetic tabletop dataset and the clustering-based approach for scene point clouds of the object change detection dataset. We set the fusing threshold to include points within 0.3 m from the plane, hoping to cover most of the easy-to-move daily objects supported by a plane. The SDF voxel grid resolution is set as 7 mm to best preserve the scene geometry, especially for smaller objects.

For PlaneSDF registration, the minimum cosine similarity and maximum offset distance are set as $\delta_{\mathbf{n}} = 0.95$ and $\delta_d = 0.2$ m.

For change detection, the height map difference threshold is set as $\delta_h = 0.02$ m so as to avoid missing smaller objects. To construct the 3D feature histogram for each area of interest, we set the number of bins along each dimension to be $N_\phi = 5$, $N_\theta = 5$, $N_\lambda = 6$. The α and the threshold δ_{blob} are set as $(\alpha, \delta_{blob}) = (2, 0.9)$ for the synthetic dataset and further moderately tuned for the object change detection dataset to accommodate certain dataset-defined cases where some slightly moved planes are not marked as changed.

4.4.4 Results on the Synthetic Tabletop Dataset

The tabletop dataset captures a relatively complete surrounding view of the various objects on a tabletop, which provides a simple yet effective scene for the initial evaluation of the proposed algorithm. The experiments are run on 20 arbitrarily selected source-target sequence pairs, with objects on the tabletop ranging from, e.g., a coffee mug (5- cm in height), a toy car (10 cm in height), and a 3-layer book stack (30+ cm in height). The output is the 2D change mask of the same size as the height map

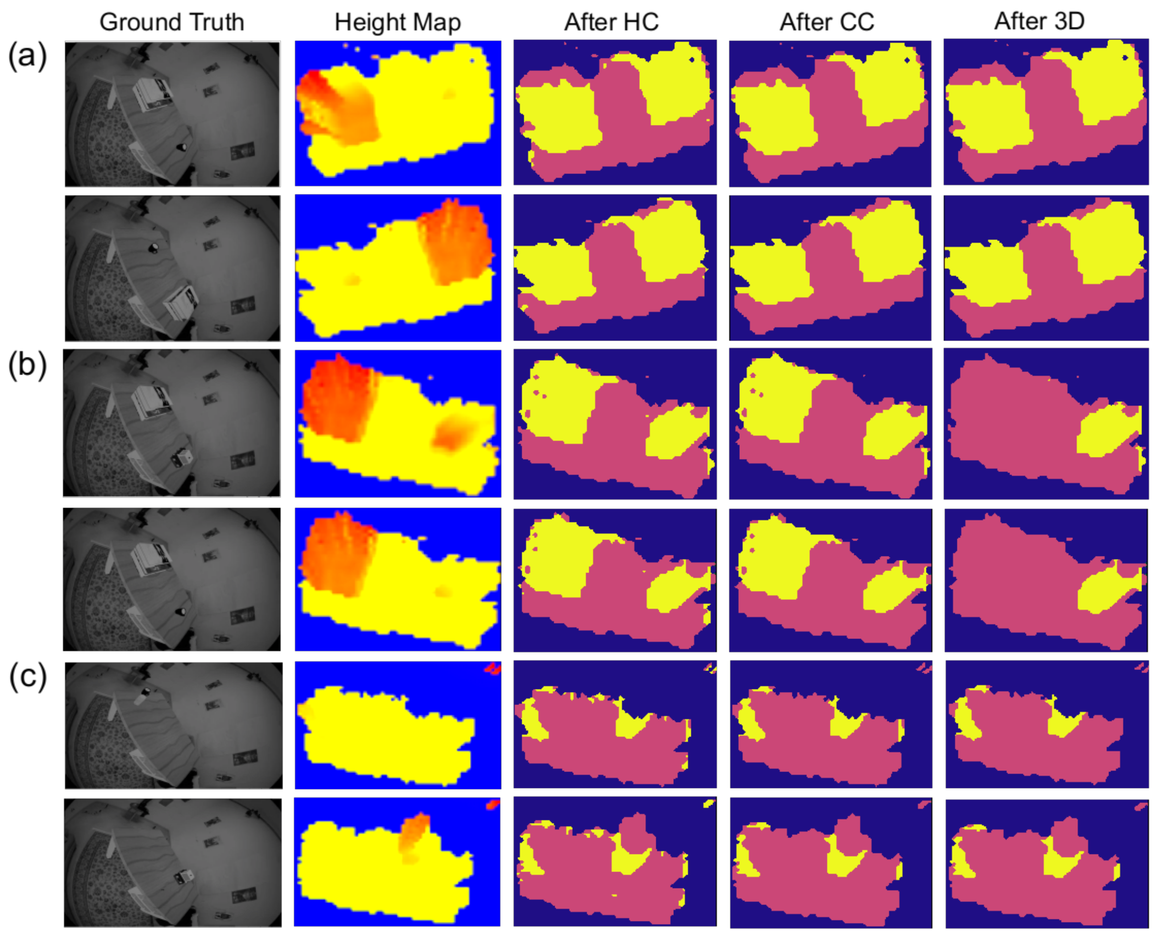


Figure 4-5: Sample change detection results on the synthetic tabletop dataset. Each mask showcases the change detection result of treating the sequence in the same row as the source. Here, we include snapshots of the actual scene in the first column, the associated height map in the second column, and the evolution of the change mask out of each stage of our approach in the last three columns: (1) height map comparison (HC) (2) connected component filtering and dilation (CC) (3) 3D geometric validation (3D).

of the source PlaneSDF volume, indicating all the changed locations on the source plane w.r.t. the target. To prove the robustness of our algorithm, we also run all the experiments in a bi-directional fashion, i.e., detecting changes source-to-target as well as target-to-source.

With relatively complete observation of all the tabletop objects, for the 20 pairs we have tested, the algorithm is able to achieve 100% recall and 80% precision rate for detecting changed objects without 3D geometric validation. The precision rate further rises to 100% after incorporating 3D validation, where false positive height differences are corrected by verifying the shape similarity in the SDF field (as for the case of the book stack shown in Fig. 4-5(b)).

Fig. 4-5 shows examples of the evolution of change masks out of each stage in the proposed method for three common object-changing scenarios: (a) Two objects swap places. (b) One object changes, and one remains. (c) Objects are added/removed to/from a free space. We can see that the masks out of height map comparison (3^{rd} column) still contain noisy false positive (FP) clusters as a consequence of reconstruction errors. The smaller FP clusters are then partially removed by connected-component filtering and dilation, as shown in the 4^{th} column. However, bigger FP patches still persist, such as the book stack on the left side of the tabletop in scenario (b). The 3D validation here then plays a significant role in comparing the 3D geometric similarity of all the possible patches and effectively reverting the FP book stack back to unchanged (5^{th} column in (b)). The results also demonstrate bi-directional robustness as the change masks have similar patterns within each source-target pair.

4.4.5 Results on the Object Change Detection Dataset

In addition to the synthetic tabletop dataset, we further evaluate our algorithm on the more challenging real-world object change detection dataset, which offers scene settings with object changes of more diverse sizes and layouts.

Quantitatively, Table 4.1 compares the results of our approach in terms of the five metrics against those of the volumetric/point-based approaches Octomap [66] and Meta-room [4], and the best results of the approach proposed by [67]. The

Table 4.1: Result comparison of the proposed approach with three baselines provided by the object change detection dataset. The best values are marked in bold (Pr = precision, Re = recall, F1 = F1 score, M = missed objects, W = wrongly detected clusters).

	Small Room					Big Room				
	Pr	Re	F1	M	W	Pr	Re	F1	M	W
Octomap [66]	0.11±0.05	0.61±0.18	0.19±0.08	15	176	0.07±0.04	0.42±0.15	0.12±0.07	42	434
Meta-room [4]	0.04±0.03	0.44±0.08	0.07±0.04	24	276	0.24±0.30	0.55±0.05	0.25±0.27	31	464
Best of [67]	**0.55±0.36**	0.66±0.17	0.57±0.22	**6**	28	0.78±0.13	0.78±0.04	0.69±0.10	**2**	50
FPFH	0.13±0.14	0.12±0.05	0.11±0.08	32	-	0.13±0.12	0.39±0.12	0.18±0.14	19	-
Ours	0.50±0.24	**0.83±0.14**	**0.59±0.21**	10	**18**	**0.78±0.03**	**0.85±0.15**	**0.81±0.09**	8	**15**
	Living Room (partial)					Office (partial)				
	Pr	Re	F1	M	W	Pr	Re	F1	M	W
Octomap [66]	0.11±0.08	0.50±0.08	0.17±0.10	19	74	0.18±0.07	0.77±0.13	0.28±0.10	8	73
Meta-room [4]	0.13±0.18	0.42±0.10	0.14±0.14	15	122	0.17±0.25	0.39±0.20	0.17±0.18	12	146
Best of [67]	**0.83±0.29**	0.69±0.11	0.72±0.17	**4**	**13**	0.49±0.27	0.83±0.06	0.54±0.20	**0**	16
FPFH	0.11±0.11	0.31±0.14	0.15±0.14	12	-	0.21±0.13	0.50±0.19	0.27±0.13	5	-
Ours	0.80±0.05	**0.87±0.10**	**0.83±0.05**	**4**	**13**	**0.72±0.10**	**0.94±0.08**	**0.79±0.06**	**0**	**4**
	Kitchen (partial)					Average				
	Pr	Re	F1	M	W	Pr	Re	F1	M	W
Octomap [66]	0.43±0.08	0.41±0.08	0.41±0.07	9	40	0.18±0.14	0.54±0.18	0.23±0.13	18.6	159.4
Meta-room [4]	0.56±0.17	0.35±0.12	0.44±0.14	9	70	0.23±0.26	0.43±0.13	0.21±0.20	18.2	215.4
Best of [67]	0.62±0.21	**0.92±0.07**	0.55±0.11	**0**	55	0.64±0.27	0.74±0.14	0.61±0.16	**2.8**	34.2
FPFH	0.57±0.16	0.62±0.11	0.59±0.14	4	-	0.22±0.21	0.38±0.21	0.26±0.21	14.6	-
Ours	**0.77±0.015**	0.85±0.05	**0.81±0.03**	2	**3**	**0.72±0.16**	**0.86±0.12**	**0.76±0.13**	4.8	**10.6**

results are computed by projecting the ground truth point clouds into SDF voxels and determining each point's change state according to its corresponding voxel indicated by the 2D change mask from our approach. Note that following the dataset definition, we manually exclude all detected changed points resulting from moved furniture and decoration from evaluation.

Moreover, to demonstrate the effectiveness of our blob-level curvature-based SDF description for robust change detection, we provide another baseline (*FPFH* in Table 4.1) with a point-wise variant of the proposed method by replacing the 3D voxel validation step in Section 4.3.4 with the point-based FPFH [101] feature matching using the Open3D [145] implementation. As our selected key voxels are *not* located on object surfaces, where off-the-shelf point feature extractors cannot be directly applied, FPFH features are extracted for every point in the original point cloud that contributes to the fusion of the SDF. A point is marked as changed if its source FPFH feature cannot be matched in its target neighborhood.

Here, Fig. 4-6 illustrates the key voxel distribution and the false positive points detected by FPFH matching for two unchanged sub-scenes: a single green object and

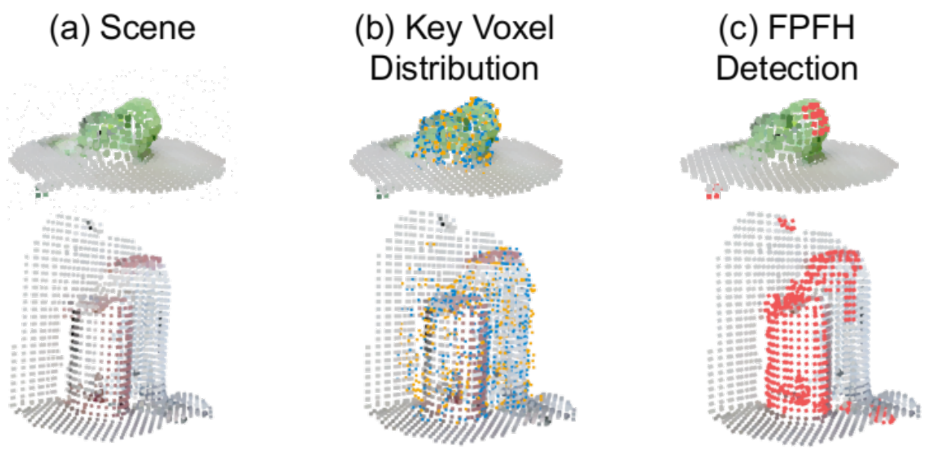


Figure 4-6: Illustration of key voxel distribution and detected false positive points from the per point FPFH feature matching baseline for two unchanged sub-scenes. (a) Scene rendering. Above: an isolated green object in the center of a tabletop in the "small room" scene. Below: two bottles standing together against a wall in the "kitchen" scene. (b) The scene point clouds are rendered in bigger colored squares, and the key voxels are in smaller squares with blue ones as those near object surfaces and orange ones farther away in the unoccupied space amid object surfaces. (c) Falsely detected changed points from the FPFH feature matching baseline are rendered in red.

two bottles standing closely against a wall. In (b), near-surface key voxels (within 1.5 SDF voxel size to an object point, shown in blue) are distributed around the object surface, giving a good characterization of the object geometry. In contrast, key voxels farther away from the surface are more frequently witnessed in spaces amid surfaces, e.g., the area around the top of the shorter bottle and the left gap between the bottles and the wall, acting to unravel the spatial relations of these adjacent surfaces. The effectiveness of considering both object surfaces and inter-surface regions is then demonstrated by (c). While our method correctly recognizes the two scenes as unchanged, FPFH shows a small ratio of false positive points for the less noisy, single-object scenario but induces considerable amounts of false positives for the two-bottle case given a partial and warped reconstruction of the shorter bottle and the wall.

From Table 4.1, we see that our approach achieves the highest values regarding the five metrics mentioned above in most scenes. The point-wise FPFH matching baseline, while not eligible for wrongly detected cluster measurements as no cluster-

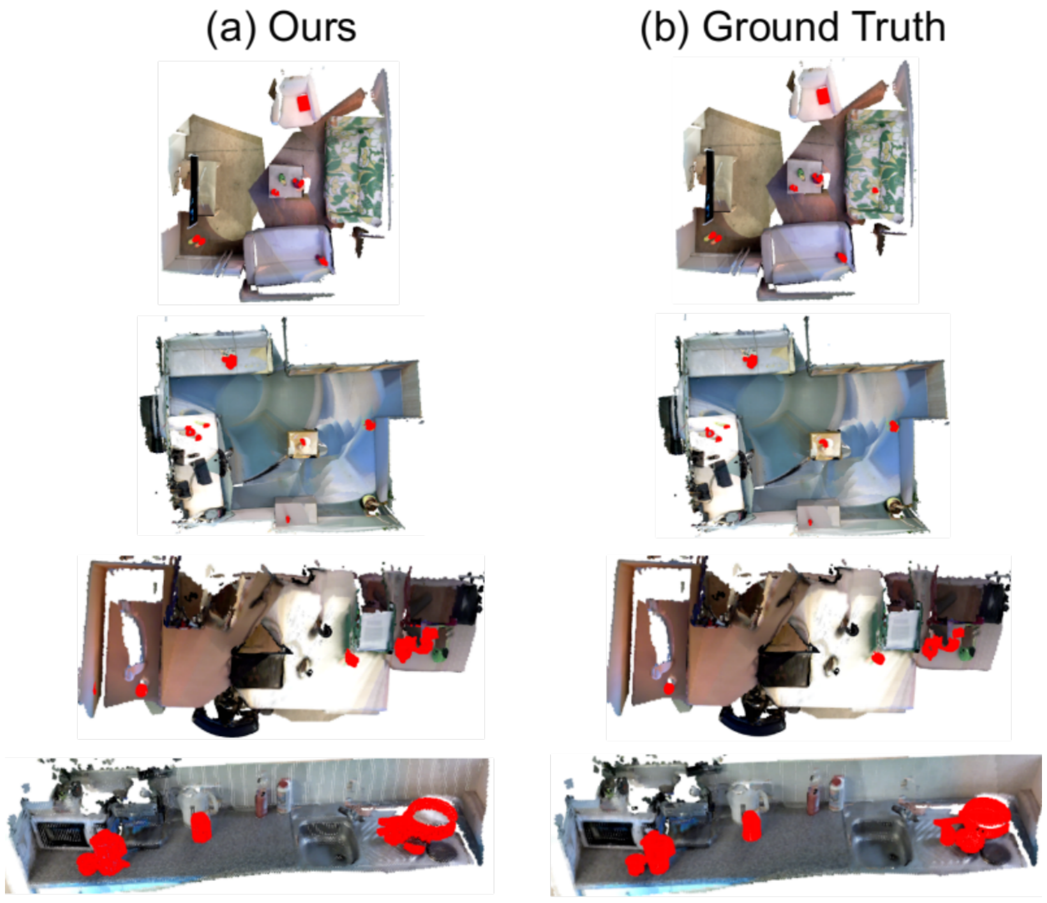


Figure 4-7: Qualitative examples of the change detection results (red) for the four scenes in the object change detection dataset, from top to bottom: living room (partial), small room, office (partial), kitchen (partial). (a): Detected objects from our algorithm. (b) Ground truth.

level operations are involved, results in worse performance in the rest of the four metrics. This can be ascribed to its sensitivity to reconstruction noises, e.g., residual points or warpings that are prevalent around boundaries.

Compared to the baseline approaches, our better performance could be attributed to the more distinct object contours and more robust neighborhood geometry verification enabled by the PlaneSDF representation. First, finding intersections between the preliminary change mask and the object map ensures that most of the voxels extracted for 3D validation belong to *part of* an object and *all* voxels of the potentially changed objects are selected for 3D validation, hence unaffected by common artifacts, e.g., noisy and incomplete object boundaries, in 3D clustering and segmentation in [67]. Second, local geometry verification, as opposed to point-wise nearest neighbor searching, offers additional robustness for detecting smaller objects and rejecting false positives, especially in the face of undesired point cloud residuals, such as when reconstruction quality is poor, and objects are close to fixed structures such as walls.

Qualitatively, Fig. 4-2 and Fig. 4-7 display examples of qualitative change detection results of each of the five scenes. From Fig. 4-7, we can see that the proposed algorithm is able to extract point clouds belonging to most of the newly introduced objects, with some points missing from the planar parts that are attached to the plane, such as the bottom of the skillet in the kitchen scene (the last row of Fig. 4-7).

While the proposed algorithm has proved to be effective in object change detection both quantitatively and qualitatively, we point out the failure case as when the height discrepancy between the object and the plane is ambiguous. Two typical examples within the dataset are: (1) The new object is partially occluded by a fixed structure in the height direction, e.g., the baseball placed under the table is missing from detection as its height is not correctly reflected in the height map. (2) The object is close to some noisy plane boundaries such as those caused by non-rigid deformation, e.g., missing object detection on the sofa (first row in Fig. 4-7) and our lower precision scores on the “small room” and “living room” scenes with new objects on the sofa.

4.5 Conclusion

In this chapter, we have presented a new approach for change detection based on the newly proposed PlaneSDF representation. By making the most of the plane-supporting-object structure, our approach decomposes the typical noise-sensitive global scene differencing scheme in a local plane-wise and object-wise manner, demonstrating enhanced robustness to measurement noises and reconstruction errors on both synthetic and real-world datasets.

Chapter 5

Robust Change Detection with Neural Descriptor Fields

Diving deeper into change detection, in this chapter, we turn our attention to *online* change detection. This is a common scenario in mobile robotics, where agents are expected to capture changes *during operation* so that actions can be followed to ensure a smooth progression of the working session. Different from its offline counterpart, where all observations are readily available, in the online setting, varying viewing angles and accumulated localization errors make it easy for robots to falsely detect changes in the surrounding world due to low observation overlap and drifted object associations.

In this chapter, based on the recently proposed category-level Neural Descriptor Fields (NDFs), we develop an object-level online change detection approach robust to partially overlapping observations and noisy localization results[1]. Utilizing the consistent shape completion capability of NDFs, we represent objects with compact shape codes encoding *complete* object shapes from partial observations. The objects are then organized in a spatial tree structure based on object centers recovered from NDFs for fast queries of object neighborhoods. By associating objects via shape code similarity and comparing local object-neighbor spatial layouts, our proposed approach demonstrates robustness to low observation overlap and localization noises.

[1]Project page: http://yilundu.github.io/ndf_change

We conduct experiments on both synthetic and real-world sequences and achieve improved change detection results compared to multiple baseline methods.

5.1 Introduction

The ability to perform robust long-term operations is critical for many robotics applications, such as room scanning and household cleaning. As these tasks usually involve frequent visits to the same environment over extended periods of time, during which the environment may experience changes, robots are expected to understand these newly-emerged scene differences, e.g., the introduction and removal of a coffee mug, as they may impact the proper subsequent actions to be taken *during* operation.

An intuitive way to conduct change detection is to perform scene differencing between current observations and scenes reconstructed from previous sessions. Using various scene representations such as point clouds and Truncated Signed Distance Fields (TSDF), previous works detect changed areas in the environment through global point-wise or voxel-wise differencing on two scenes reconstructed and pre-aligned *post hoc* from sequential data [33, 4, 32]. These methods, which treat the environment as an unordered collection of points or voxels, demand high inter-session viewpoint overlap and are susceptible to noisy sensor data and localization errors. Considering the fact that changes take place at the object level, recent works [67, 106] explore the use of semantic consistency for local object-level verification on top of the common global point- or voxel-wise scene comparison scheme. Having demonstrated improved robustness to localization errors, they are nevertheless prone to failure with noisy reconstruction input and little observation overlap, both of which are frequently encountered during online change detection tasks.

It is therefore highly desirable to seek approaches that yield consistent representations of the object surmounting viewing angle limitations, where we then adopt the recently proposed Neural Descriptor Fields (NDFs) [110]. NDFs have been developed as a category-level, SE(3)-equivariant object representation for object manipulation tasks. By encoding an object as a continuous SE(3)-equivariant function, NDFs are

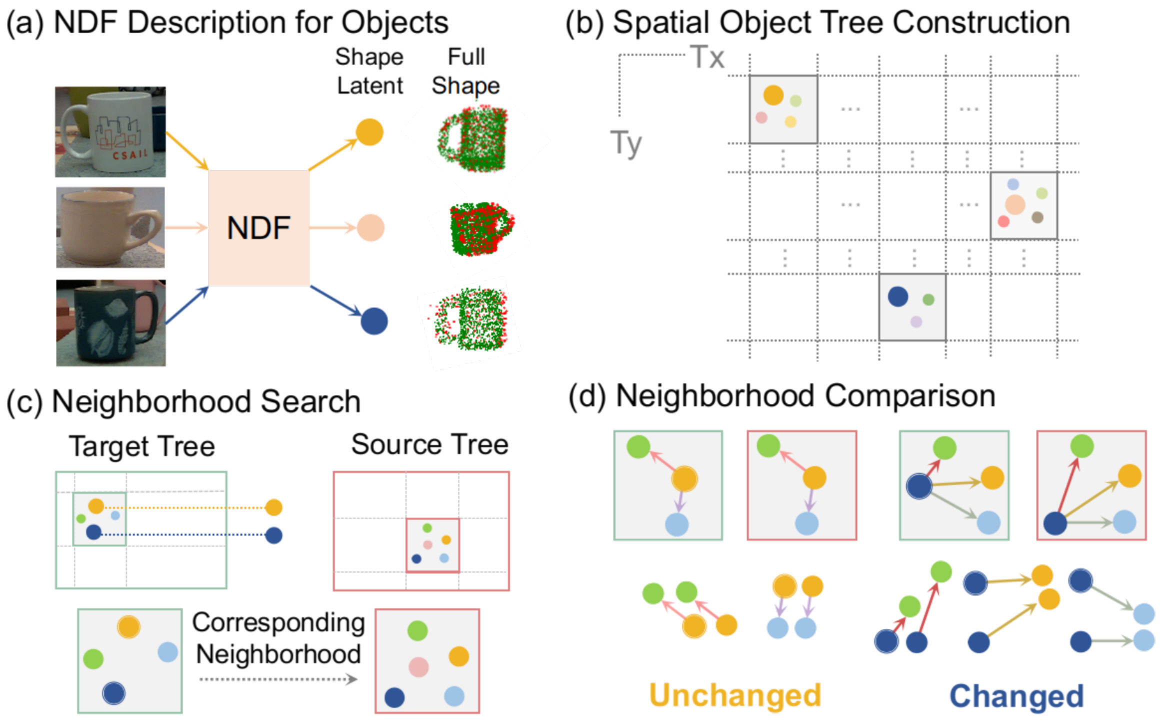

(a) NDF Description for Objects
Shape Latent
Full Shape
NDF
(b) Spatial Object Tree Construction
Tx
Ty
(c) Neighborhood Search
Target Tree
Source Tree
Corresponding Neighborhood
(d) Neighborhood Comparison
Unchanged
Changed

Figure 5-1: Approach overview. Given a sequence of observations as the source and the target, during the streaming of the target sequence, the system takes in partial object point clouds from depth sensors and outputs changed objects on the fly in the target w.r.t the source. (a) Given each partial object point cloud, Neural Descriptor Fields (NDFs) represent the object as a shape code encoding the full object shape and recover the object center from full shape reconstruction. (b) Based on recovered object centers, observed objects are organized in a spatial object tree consisting of two coordinate interval trees (T_x, T_y), allowing for fast query of neighboring objects. (c)-(d): Corresponding neighborhoods of the current object are found from the source and the target object tree using object locations, where objects of similar shape codes are matched. For each matched object pair, object graphs of the neighborhood are constructed and compared to determine if the local object layouts are different, which can imply changed objects.

able to reconstruct the full object shape through a compact shape encoding and further recover object centers (translation). Considering that NDFs are formulated only to ensure identical shape codes given rotated point clouds that are otherwise identical, we augment its shape completion capability by enforcing a new shape similarity loss with partial object observations as inputs, making it encode consistent full object shapes and positions across different viewing angles.

Following the object-level interpretation of the world via NDFs-derived object representations, here, we propose an object change detection approach (see Fig. 5-1) for mobile robots, targeting the most common online operating scenarios with high viewing angle variation, potential localization errors, and no pre-alignment tools available. Based on the consistent category-level shape completion ability of NDFs, we represent the object with a compact *full* shape code obtained from partial observations and organize the objects in a spatial tree structure in terms of object centers recovered from NDFs for fast query of object neighborhoods. By associating objects through shape code similarity and decomposing the global differencing scheme into local object-neighbor layout comparison, our approach shows improved robustness to little observation overlap and localization errors. Our main contributions are as follows:

- First, we explore the use of NDFs for category-generalizable shape-consistent object representation across different viewing angles.
- Next, we propose an online change detection approach based on NDF-derived object representations and local object layout comparison.
- Finally, we demonstrate the effectiveness of the proposed approach on both synthetic and real-world testing sequences featuring novel same-category object instances not seen during training time.

5.2 Related Work

In this section, we quickly go over previous efforts in change detection and neural implicit representations at the time of publication. For a more detailed review of the two topics, please refer to Section 4.2 and Section 6.2.

5.2.1 Change Detection

Previous works conduct change detection based on inputs in various 2D and 3D environment representations. Given 2D image inputs, Derner et al. [24] built a visual database and compared classical 2D feature descriptors extracted from greyscale images for query and reference feature matching. For works with 3D representations, Finman et al. [33] discovered new objects as changed parts of multiple depth point clouds, while Herbst et al. detected changes through the movement of surfaces [43]. Ambrus et al. [4] computed a meta-room static reference map based on point clouds and discovered new objects from changes via spatial clustering. Fehr et al. [32] proposed a multi-layer TSDF grid structure and performed volumetric differencing for object discovery and class recognition. Langer et al. [67] combined semantic information and supporting plane information to discover objects newly introduced to the scene. Schmid et al. [106] proposed a multi-TSDF panoptic mapping approach and conducted online change detection based on TSDF value comparison.

Most previous works focus on point-wise or voxel-wise differencing on pre-aligned reconstructions in an offline fashion, demanding high observation overlap and decent localization results. They, therefore, only sometimes satisfy the need for online change detection under various viewing angles, as commonly encountered during mobile robot operation.

5.2.2 Neural Implicit Representations

Neural implicit representations have been proposed as a continuous, differentiable, and parameterized representation of 3D geometry [88, 95], appearance [78], auditory [73], and tactile properties [35] of both objects and scenes. Neural implicit representations represent shapes as continuous functions, enabling the principled incorporation of symmetries, such as SE(3) equivariance [19, 110], as well as the construction of latent spaces that encode class information as well as 3D correspondences [21]. Owing to their continuous parameterized nature, several works have applied [110, 115] them to robotics tasks as intermediate object representations directly inferable from raw perception. Simenov et al. [110] showed that symmetries incorporated from such a representation could generalize demonstrations of objects to novel poses, while [115, 71, 51] demonstrated their ability to be integrated with online robotic mapping tasks.

In summary, we explore how to enforce shape consistency on NDFs [110] to represent objects from partial observations, thus allowing for robust online change detection with disparate viewing angles and localization noises.

5.3 Category-level Object Representation for Partial Observations

We base our object representation on the recently proposed Neural Descriptor Fields (NDFs) [110], a function f_θ that encodes both object shapes and poses through a category-level SE(3)-equivariant latent representation.

5.3.1 Neural Descriptor Fields

NDFs consist of an encoder function $f_\theta(\mathbf{P}) = \mathbf{z}$, which maps a partial object point cloud $\mathbf{P}$ into a global latent code $\mathbf{z}$, and a decoder function, $\Phi(\mathbf{x}, \mathbf{f}_\theta(\mathbf{P}))$, which maps an input point $\mathbf{x}$ to its predicted occupancy value:

$$\begin{aligned} f_\theta(\mathbf{P}) &= \mathbf{z} : \mathbb{R}^{n\times 3} \to \mathbb{R}^{k\times 3} \\ \Phi(\mathbf{x}, \mathbf{f}_\theta(\mathbf{P})) &= \Phi(\mathbf{x}, \mathbf{z}) : \mathbb{R}^3 \times \mathbb{R}^{k\times 3} \to [0, 1]. \end{aligned} \tag{5.1}$$

The encoder function $f_\theta(\cdot)$ is constructed such that by zero-centering the input $\mathbf{P}$, the inferred global latent code $\mathbf{z}$ is represented as a vector of points that is equivariant with respect to SO(3) rotations of the input point cloud $\mathbf{P}$. This means that if a point cloud is rotated by $\mathbf{R}$, the inferred latent will be equivalently rotated by $\mathbf{R}$, as ensured via the Vector Neuron encoder layers [19]. By subtracting from $\mathbf{x}$ the point cloud center of $\mathbf{P}$ as the translation, we then make $f_\theta(\cdot)$ an SE(3)-equivariant shape occupancy predictor for different points $\mathbf{x}$ on and off $\mathbf{P}$.

NDFs are trained with partial object point clouds recovered from semantic-segmented RGB-D images and corresponding 3D occupancy voxel grids of objects' complete geometry. During training, the full model $[f_\theta, \Phi]$ is trained to predict the complete 3D occupancy of an object using the standard cross-entropy classification loss L_{occ}:

$$L_{occ} = \mathcal{L}(\Phi(\mathbf{p}, f_\theta(\mathbf{P}), v), \tag{5.2}$$

where $\mathbf{p}$ is a point sampled from the object occupancy grid and v is the ground truth occupancy value at point $\mathbf{p}$.

By feeding $\Phi(\cdot,\cdot)$ with a query point cloud $\mathcal{X}$, which is obtained via uniform sampling within a large bounding box centered around $\mathbf{P}$, the full shape point cloud $\mathcal{S}$ of the object can be reconstructed in terms of the predicted occupancy values:

$$\mathcal{S} = \{\mathbf{x} | \Phi(\mathbf{x}, f_\theta(\mathbf{x}|\mathbf{P})) > v_0, \mathbf{x} \in \mathcal{X}\}, \tag{5.3}$$

where v_0 is the occupancy threshold to mark a point location as occupied.

5.3.2 Shape Consistency

Thanks to the SO(3)-equivariance of $\mathbf{z}$, a shape code invariant of view directions, $\mathbf{s} \in \mathbb{R}^k$, can then be extracted from its rotation invariant portion:

$$s_i = ||\mathbf{z}_i||_2, i = 1, 2, ..., k. \tag{5.4}$$

Ideally, the shape code, serving as a compact representation of the full object shape, should be consistently close among partial observations of the same shape from various viewing perspectives while discriminatively far apart across observations of different shapes. While NDFs enable identical shape codes given identical point clouds that are rotated, shape consistency across partial inputs of the same shape is not inherently guaranteed.

To enforce $\mathbf{s}$ to be a discriminative yet consistent representation for shapes seen from various viewing angles, as commonly seen during mobile robot operation, we further formulate a shape similarity loss, L_{shape}, fashioned after the idea of the triplet loss with [*anchor, positives, negatives*] for person re-identification tasks [45]. Taking a partial point cloud from object shape i as the anchor $\mathbf{A}_i$, we assign it with a positive sample $\mathbf{P}_i$ as the point cloud obtained from another perspective of the same object, and any other partial observations of a different object as the negative sample $\mathbf{N}_i$. With the distance metric $D(\cdot,\cdot)$ chosen as the cosine similarity, $L_{shape}(\mathbf{A}_i, \mathbf{P}_i, \mathbf{N}_i)$ then tends to pull the shape codes of $\mathbf{A}_i$ and $\mathbf{P}_i$ closer, while pushing those of $\mathbf{A}_i$ and $\mathbf{N}_i$ further apart:

$$L_{shape} = -D(\mathbf{s}_{\mathbf{A}_i}, \mathbf{s}_{\mathbf{P}_i}) + D(\mathbf{s}_{\mathbf{A}_i}, \mathbf{s}_{\mathbf{N}_i}). \tag{5.5}$$

Moreover, to ensure that L_{shape} always finds the more informative $(\mathbf{A}, \mathbf{P}, \mathbf{N})$ triplet, we adopt the *batch hard* way for triplet forming [45], i.e., using the most dissimilar $(\mathbf{A}, \mathbf{P})$ and the most similar $(\mathbf{A}, \mathbf{N})$ within each batch to guide training.

We hence populate each training batch B with randomly generated observations of (o_A, o_P) pairs of different (but likely repetitive) object shapes, which ensures that at least two observations exist for each object within the batch. Every sample within the batch is then treated as an anchor and paired batch-wise for the most dissimilar positive and the most similar negative samples. Hence, the final batch-hard shape similarity loss is formulated as:

$$\begin{aligned} L_{b\_shape} = \frac{1}{|B|} \sum_{i=1}^{N} \sum_{j=1}^{N_i} \Big(& - \min_{k \in [1, N_i]} D(o_{ij}, o_{ik}) \\ & + \max_{m \neq i} D(o_{ij}, o_{mn}) \Big), \end{aligned} \tag{5.6}$$

where N is the number of objects whose observations are included in the current batch, N_i the number of samples of object i and o_{ij} the jth observation of object i within the batch.

The final training objective is therefore formulated as the weighted combination of the cross entropy loss and the shape similarity loss as:

$$L = L_{occ} + \alpha L_{b\_shape}, \tag{5.7}$$

where α is the weighting coefficient set as $\alpha = 0.01$.

5.3.3 Training in Simulation

To overcome the data availability issue, we train our category-level shape-consistent NDFs entirely in simulation using depth images rendered with Pybullet [15]. To consider the commonly encountered variation in viewing angles, observation distances, and occlusion during operation, for each training sample, we place a randomly posed object on the table, along with 2-4 randomly drawn objects around it to create occlusion. The camera locations are sampled in a hollow rectangular region with the table at the center, thus accounting for both nearby and longer distance observations seen in real-world operations (see Fig. 5-2).

5.3.4 Object Representation

We hence base our object representation on the shape code in Eq. (5.4) and the object center (translation) $\mathbf{t}$ recovered from NDFs as $(\mathbf{s}, \mathbf{t})$, where $\mathbf{t}$ is simply the predicted center of the reconstructed full point cloud $\mathcal{S}$ in Eq. (5.3):

$$\mathbf{t} = \frac{1}{|\mathcal{S}|} \sum_{\mathbf{x} \in \mathcal{S}} \mathbf{x}. \tag{5.8}$$

By converting $\mathbf{t}$ into the world frame with the given camera pose, the object is characterized by $\mathbf{s}$ for its full shape and $\mathbf{t}$ for its global location. With the shape completion ability of NDFs, the current object representation is expected to provide a compact as well as robust way to distinguish object instances, even under varying viewing angles during robot motion.

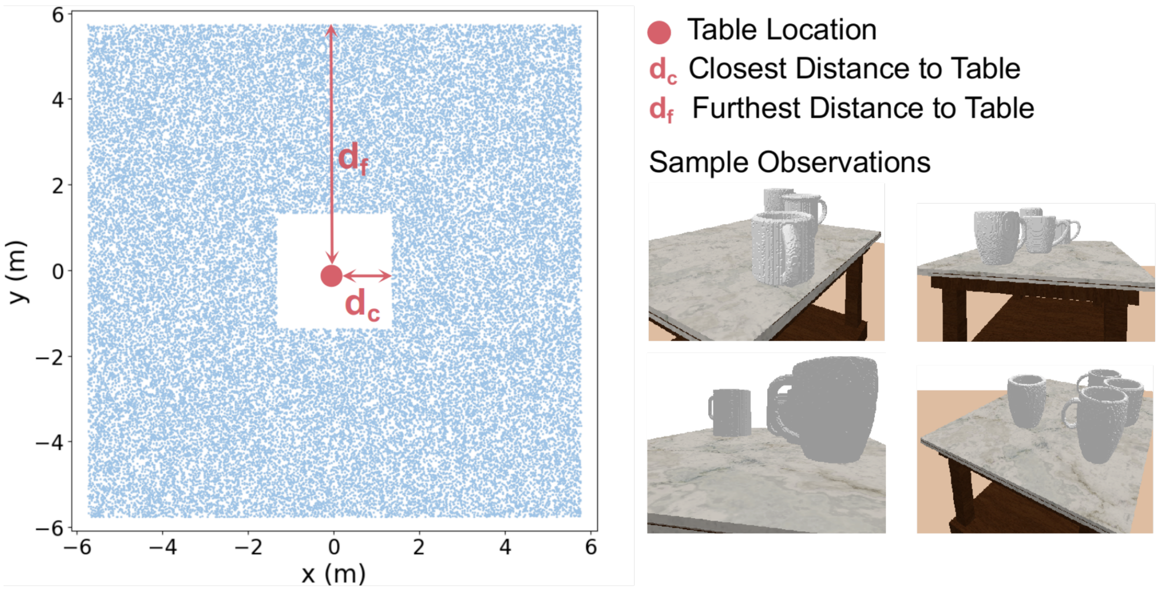


Figure 5-2: Training sample generation. Snapshots of the rendered scenes are presented on the right. Along with a height uniformly varying within 0.1 m from the tabletop, 2D camera locations (blue dots) are drawn uniformly from the hollow rectangular regions centered around the table. The width of the region is determined by the closest and furthest camera-to-table distance d_c and d_f. $2 - 4$ geometric object models are randomly selected and placed on the table to simulate potential occlusion.

5.4 NDF-based Object Change Detection

Inspired by the idea that local relative comparison is less sensitive to global localization drift than its global counterpart [67], we organize the observed object instances in a spatial object tree based on the predicted object centers, allowing for the convenient query of object neighborhoods. By comparing local neighboring object layouts, we improve the method's robustness to localization drift by avoiding direct comparison on the absolute values of localization results.

5.4.1 Spatial Object Tree Construction

To enable the online execution of our approach, we construct and maintain a spatial object tree in the world coordinate to organize the incoming data stream.

Built on top of the coordinate interval tree proposed in [141], our object tree comprises two translation interval trees, $T = (T_x, T_y)$, in the x and y direction, respectively, which are initialized and updated simultaneously in the same fashion each time a new object measurement $(\mathbf{s}, \mathbf{t})$ arrives. Considering that adjacent objects are located mostly on the same plane, for our case, it suffices to maintain trees only in the x and y direction for neighborhood query (while an extension to the z direction should be straightforward).

Tree Construction. Consider T_x for instance. T_x divides the $x - y$ plane into several equi-distant interval slices in the x direction (Fig. 5-1(b)), where each instantiated interval $[x_{min}, x_{max}]$ of fixed length $l = x_{max} - x_{min}$ is represented by a node n. Each node stores a set of object instances whose translation component $\mathbf{t}$ follows $t_x \in [x_{min}, x_{max}]$. The x_{min} and x_{max} are determined by the first t_x that initializes this new node as $[t_x - l/2, t_x + l/2]$. The new node is then inserted in the tree such that the upper interval limit of a left child n_l is always smaller than the lower interval limit of its parent node n_p, i.e., $x_{l,max} < x_{p,min}$ and similarly for the right child n_r, we have $x_{r,min} > x_{p,max}$.

Intra-tree Update. We associate the incoming object measurement $\mathbf{m} = (\mathbf{s}, \mathbf{t})$ with existing object instances in the tree through spatial proximity and the shape cosine similarity used in Eq. (5.5). We first traverse through T_x and T_y to locate the corresponding coordinate interval nodes n_x and n_y that $\mathbf{t}_x$ and $\mathbf{t}_y$ land in, respectively. By finding the intersection of the objects within n_x and n_y, we obtain the neighborhood $\mathcal{N}$ that $\mathbf{m}$ should be adjacent to as $\mathcal{N} = \{o | o \in n_x \wedge o \in n_y\}$. The measurement is then successfully associated with an object instance $o_0 = (\mathbf{s}_0, \mathbf{t}_0)$ in the neighborhood when:

$$||\mathbf{t}_0 - \mathbf{t}|| < \delta_d \wedge D(\mathbf{s}_0, \mathbf{s}) > \delta_s, \tag{5.9}$$

where δ_d and δ_s are the maximum distance threshold and minimum shape similarity threshold for valid association. We then replace the old shape code and the recovered object center of the object with $\mathbf{m}$ as an update if the average occupancy value of the reconstructed full shape point cloud for $\mathbf{m}$ (calculated in Eq. (5.3)) is higher than that of o_0.

5.4.2 NDF-based Change Detection

Given two sequences as the source and the target, we construct the source object tree T_s in advance and perform object change detection while constructing the target tree T_t.

Sequence Registration. We wish to apply our approach in an online mobile operating scenario where post-alignment tools or motion capture systems are not always readily available. Therefore, we first conduct a rough alignment between the target and the source sequence at the beginning of the target traverse, in the hope that target objects can correctly locate the corresponding neighborhood patch in the source so as to enable *valid* neighboring object comparison.

With all the shape codes of the source objects easily obtainable from T_s, after intra-tree update for T_t, we determine each corresponding source object o' for the current target object o as the one sharing the highest shape code similarity among all the objects (if exists) whose shape similarity with o is above δ_s (similar to Eq. (5.9) amid intra-tree measurement-object association).

After N pairs of object correspondences (o', o) have been accumulated, we apply Single Value Decomposition (SVD) with RANSAC onto the N object center pairs $(\mathbf{t'}_i, \mathbf{t}_i)$ and obtain the relative transform T_{rel} between the two sequences. This considers the potential inclusion of changed object instances while assuming that they only take up a small portion of the environment to allow for alignment when the target sequence starts. Here, we set $N = 6$ to account for transform accuracy versus timing to start change detection.

Change Detection. Rather than examining object-wise correspondences through spatial proximity and shape similarity (as in Eq. (5.9)), we draw support from neighboring objects and detect changes through the relative spatial layout consistency of the object within its target neighborhood and that of its corresponding neighborhood (if exists) in the source.

When an object measurement $\mathbf{m} = (\mathbf{s}, \mathbf{t})$ arrives, we update the T_t by finding/instantiating the associated object instance O. Considering the online operation

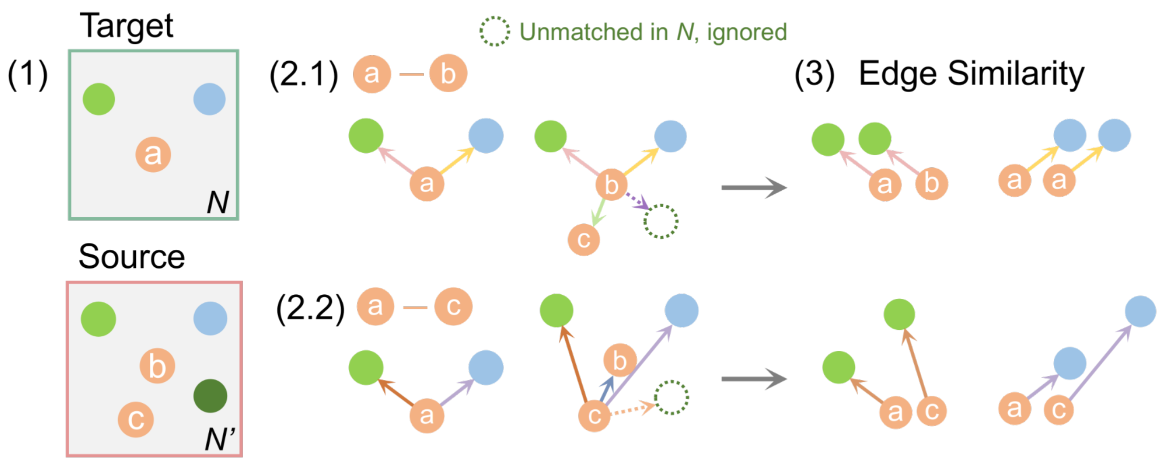


Figure 5-3: Neighborhood comparison as object graph matching. (1) Shape a in the target neighborhood $\mathcal{N}$ forms two shape-similar pairs (a, b) and (a, c) from the corresponding source neighborhood $\mathcal{N}'$ (dots colored in orange). (2) Object graphs centered at a, b, and c are constructed respectively, demonstrating local inter-object layout. Graphs of each pair are compared as shown in (2.1) for (a, b) and (2.2) for (a, c). Each object graph consists of vertices as all the objects in the neighborhood (colored dots) and directed edges pointing from the object to the neighbor with length as the object center difference (colored arrows, e.g., $\mathbf{e}_{cb} = \mathbf{t}_b - \mathbf{t}_c$). (3): Graph (neighborhood layout) comparison is conducted edge-wise, where edge similarity between vertex-matched edges (shown in arrows of the same color) is measured by the Euclidean distance between the two edge vectors.

setting, change detection is then performed on $O = (\mathbf{s}_0, \mathbf{t}_0)$ that has *not* been marked as changed before.

Given $O = (\mathbf{s}_0, \mathbf{t}_0)$ along with its neighborhood $\mathcal{N}_0$ in the target, we find the projected location in T_s with T_{rel} and query the corresponding source neighborhood $\mathcal{N}' = \{o'_i = (\mathbf{s}_i, \mathbf{t}_i)\}$. Potential object matches are found based on shape code similarity between $\mathbf{s}_0$ and each $\mathbf{s}_i$ in $\mathcal{N}'$.

The most straightforward change cases are that either $\mathcal{N}'$ is empty or no matched o_i is found in the source neighborhood, as both indicate newly added objects to new locations or else new shapes.

Then, with object matches found within $\mathcal{N}'$, we conduct local neighborhood comparison for each matched pair (O, o'_i) (see Fig. 5-3 for a concrete example). This also serves as a validation step for rejecting false positives in dealing with noise-corrupted localization. When object position changes at a scale similar to cross-session localization errors, it can be hard to determine if the absolute object center difference between (O, o'_i) is brought by actual changes (e.g., objects move within the plane) or merely localization/projection drift. At the same time, the latter can be validated through the relatively more stable inter-object spatial layout.

For each of the matched pairs, neighborhood comparison is accomplished via object graph matching. A local object graph $G_o = (V, E)$ for object o in neighborhood $\mathcal{N}_0$ is constructed as a directed graph with vertices as all the objects in $\mathcal{N}_0$, $V = \{o_i | o_i \in \mathcal{N}_0\}$, and directed edges as the object center difference pointing from o to its neighbors, $E = \{\mathbf{e}_{oi} | \mathbf{e}_{oi} = \mathbf{t}_i - \mathbf{t}_o, \forall i \in \mathcal{N}_0, i \neq o\}$. Considering the limited number of objects within a neighborhood (size set similar to a tabletop), the relatively small graph size makes it possible for edge-wise matching. We find corresponding edges $\mathbf{e}_{oj}$ and $\mathbf{e}_{o'j'}$ through vertex shape matching. Similar edges, indicating unchanged inter-object layout, are determined by the Euclidean distance between them:

$$\sum_j \mathbb{1}(||\mathbf{e}_{oj} - \mathbf{e}_{o'j'}||_2 \leq \delta_e) = \begin{cases} 0, & \text{changed} \\ \geq 1, & \text{unchanged.} \end{cases} \tag{5.10}$$

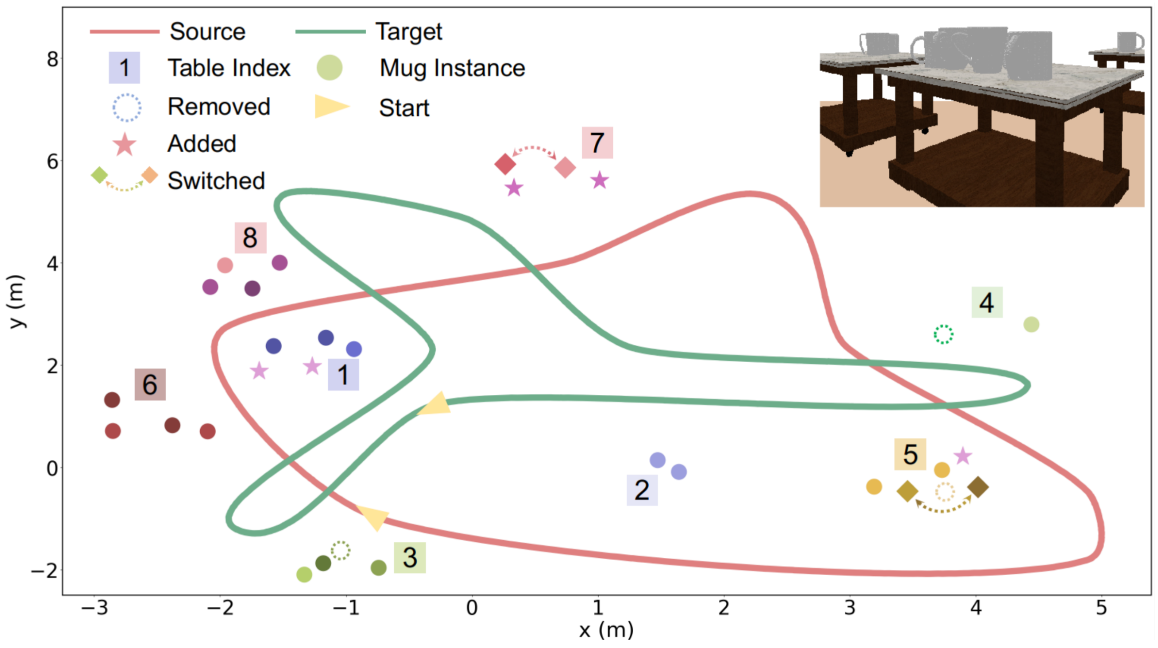


Figure 5-4: Synthetic scene layout and camera trajectories. Mug instances are shown in colored dots around eight tables as numbered squares. Changes of different types are shown with different icons. A snapshot of the rendered scene is shown in the upper right corner. The trajectory covers an area of $8 \times 8\ m^2$.

If at least one pair of edges is found to be closer than δ_e, i.e., at least one out of the few neighboring objects remains the same, the local object layout is marked as consistent. This implies an unchanged object and vice versa. Note that in the rarer case, when either O or o'_i is the only object in the neighborhood, we revert back to object-wise comparison in terms of spatial proximity and shape similarity.

Lastly, for detecting objects removed from the source, during target streaming, we label each source object as *observed* if it ever participates in shape matching with any target objects, then *removed* objects can be found after the target sequence finishes as those *observed* but never matched with a target object.

5.5 Experiments and Results

Targeting robust online change detection in the face of little observation overlap and localization errors, we would like to answer two questions: (1) Can we use the shape-consistent NDF-based representation as a valid category-level object representation

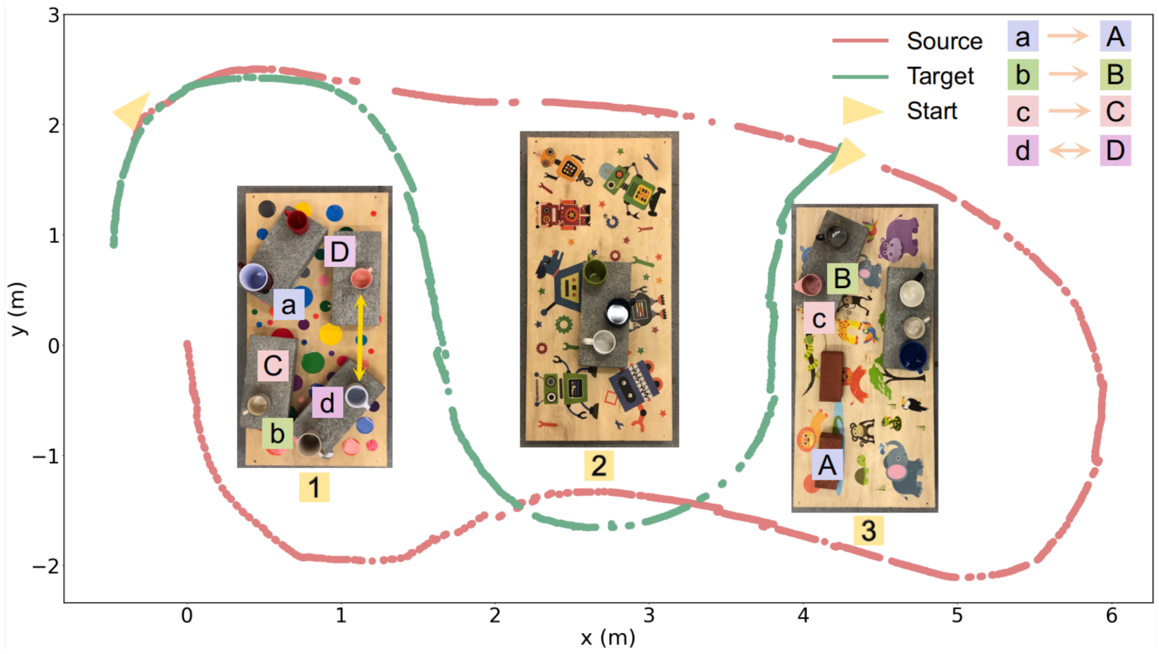


Figure 5-5: Real-world scene layout and camera trajectories. Changes take place in tables 1 and 3 and are indicated by the colored letter block highlighted in the top view pictures of the three tables.

that robustly generalizes to unseen object instances under partial observations? (2) Can our change detection approach perform well on sequences with little observation overlap and demonstrate robustness to localization errors? We evaluate our approach on both synthetic and real-world sequences consisting of various mug instances, where mugs are added, removed, and switched places between sequences.

5.5.1 Datasets

To better examine the effectiveness of our *category-level* object representation and the change detection approach built on top of it, we choose to have our testing sequences composed of objects from the same category. We hence design two pairs of testing sequences, in simulation and in the real world, respectively, featuring coffee mugs of diverse shapes observed from distinct viewing angles. Considering the extensive multi-category results reported in relevant works [110, 77, 89], we argue that the effectiveness of our approach should be extendable to the multi-category case by incorporating more object categories into the training data.

Synthetic Sequences. We set up an environment in Pybullet with 35-40 instances of 20 ShapeNet [12] mugs models scattered around eight tables and obtain RGB-D images along with segmentation masks from two preset camera trajectories. The mug models are unseen by NDFs during training. Each table supports 3-5 mugs with mugs newly added, removed, or switched locations, creating a total of 12 changes between the two sessions. The two camera trajectories are designed such that the camera always faces towards the nearest table, prompting less observation overlap between sequences around table 1,2, and 5 (see Fig. 5-4).

Real-world Sequences. We collect two sequences featuring 14 mugs of diverse shapes randomly placed on three rectangular tables. We mount an Intel RealSense L515 camera onto a Jackal robot close to the mugs' height and collect RGB-D data with the trajectories shown in Fig. 5-5. The camera trajectories are recovered using ORB-SLAM3 [11]. Since the camera is mounted facing sideways, while the camera circles around table 1, 2, and 3 in the source, it only weaves through table 1 and 3 in the target, thereby creating observation disparity on mugs along the inner side of table 1 and 3.

5.5.2 Metrics

We adopt the commonly used precision and recall rate at the *object* level, i.e., the number of objects, for evaluation. We report the number of correctly detected changes (true positives, TP), falsely detected changes (false positives, FP), and undetected changes (false negatives, FN). Precision and recall rate are computed as $Precision = \frac{TP}{TP+FP}$ and $Recall = \frac{TP}{TP+FN}$.

5.5.3 Implementation Details

To train NDFs using an occupancy network, we set the camera-table distances as $d_c = 0.2$ m and $d_f = 5$ m and generate 50,000 RGB-D partial observations with 94 ShapeNet [12] mug models following the sample generation strategy in Section 5.3.3. Partial object point clouds are obtained by extracting corresponding depth points in the camera frame indicated by the segmentation masks on RGB images. We build our model upon the network implementation provided by [110] and train it on two NVIDIA RTX 3090 GPUs using a learning rate of 6×10^{-4} with the Adam optimizer. The length of the interval tree l is set to be 1.2 m and 1.6 m for the synthetic and real-world sequence, respectively, intending to group most objects on the same table plane. For parameters in Section 5.4, we set $\delta_s = 0.9$, δ_d =0.02 m, and $\delta_e = 0.03$ m.

5.5.4 Generalization to Unseen Instances

To demonstrate the robust generalization of the shape-consistent NDFs to unseen object instances under various viewing angles, in Fig. 5-6, we render the reconstructed 3D shapes of five mugs drawn from the synthetic and real-world testing sequences. The three views are selected in the hope of representing some of the most typical perspectives when observing a mug, e.g., handles visible in different directions or obscured. Despite the variety of the selected mug shapes, we can conclude that the shape-consistent NDFs still demonstrate satisfactory performance in encoding and completing the full shape under varying viewing angles, justifying the adoption of NDF-derived object representation in our change detection task.

This can be attributed to the fact that by learning purely from geometric structures embedded in the partial point clouds, NDFs are able to transfer this structural knowledge to unseen instances regardless of object color and texture. We also include cases with occlusion and shape ambiguity. For occluded observations, as shown in the first two columns of “View 2” and “View 3”, we can still see decent reconstruction results by virtue of the occlusion scenes incorporated in the training data. For the case of shape ambiguity, i.e., the handle is obscured in the second and last column of “View 2”, the major body parts of the mugs are still reconstructed reasonably well. While the handle location was incorrectly predicted, such mistakes only result in negligible errors in object center recovery.

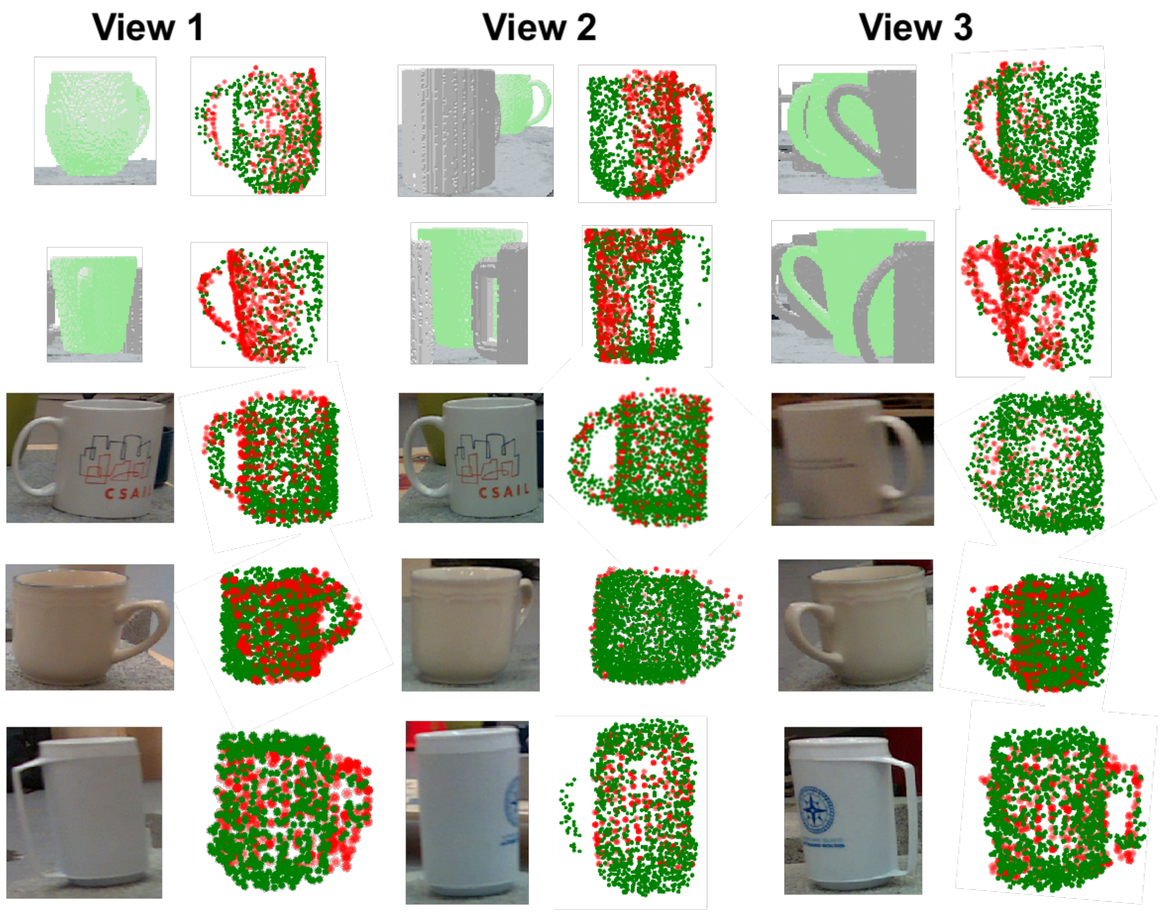


Figure 5-6: 3-view shape reconstruction of five unseen mugs (seen in each row) from the synthetic and real-world testing sequences. For each view, Left: Partial observation. Right: Partial point clouds (red) and full shape reconstruction (green). For the first two rows of synthetic mugs, the two target mugs for reconstruction are highlighted in green. The point clouds are rendered in an orientation that showcases the boundary between partial observation and the predicted shape completion, whose shape fitness between the two colored point clouds demonstrates NDFs' effectiveness in completing partial shapes.

5.5.5 Results on Change Detection

We further present the change detection results on the two sets of testing sequences based on our proposed approach and compare them to two baselines.

The first baseline is the typical nearest neighboring point search (NN) commonly used for offline scene differencing [67, 4]. Given two spatially aligned observations S and T, the change point set C of T w.r.t S is found as

$$C = \{p | p \in T, \forall s \in S, ||p - s||_2 > d\}. \tag{5.11}$$

We adapt it to the object level and define that an object is *changed* if a certain proportion r of its target point cloud finds no neighbors in the source. To make the results interpretable to online object-level change detection evaluation, we mark an object as *changed* the first time during data streaming it is detected as *changed*, as during real-world operation, corresponding actions will be taken right after changes are detected, and usually no correction can be made for false positives.

The second baseline is the recently proposed panoptic multi-TSDFs (PMT) mapping method by Schimd et al. [106], which represented panoptic entities as TSDF submaps and captured online object-level scene changes based on voxel-wise weight counting informed by TSDF value differences. After tuning the default parameters for better performance, we count the total number of changed objects based on the number of conflicting submaps determined by the baseline.

Since both baselines require pre-aligned sequences, we feed the baselines with ground truth camera poses from Pybullet and those aligned by ORB-SLAM3, respectively. For NN, each source object's reference partial point cloud is fused from the whole sequence of the source depth images. Data associations between the target partial mug observations and the source point clouds are determined by the instance ID in Pybullet for synthetic sequences and manual labeling based on the panoptic masks produced by Detectron2 [134] (used in PMT) for real-world sequences.

All results are obtained on a laptop with an Intel Core i7-9750H CPU and an Nvidia GeForce RTX 2070 GPU. The network inference speed for NDF is around

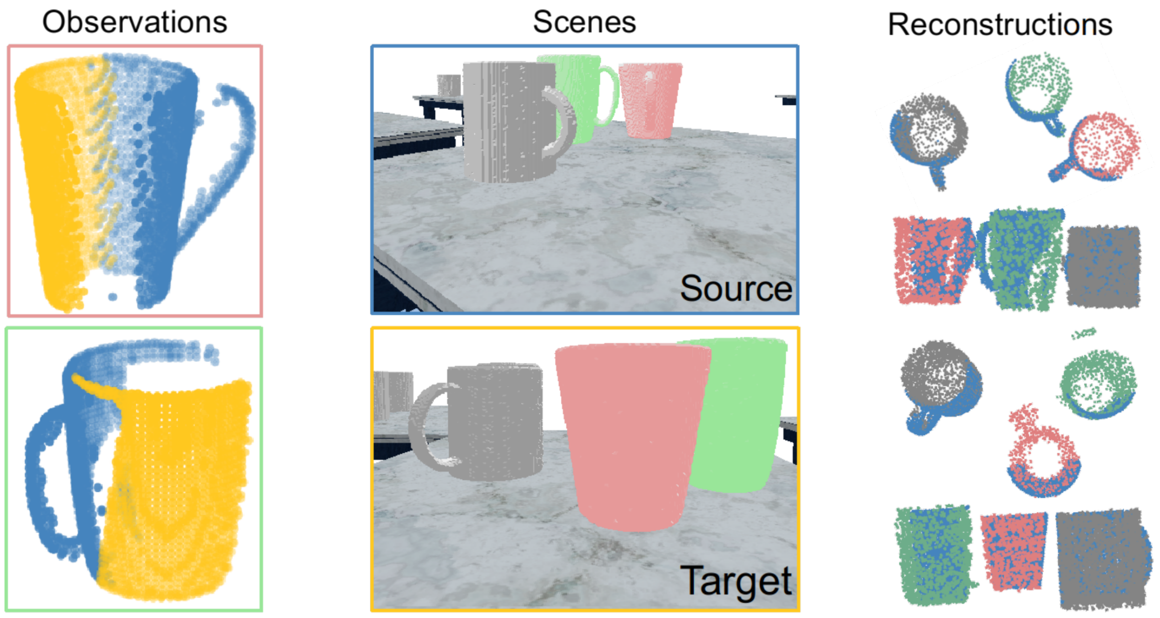


Figure 5-7: Robustness to little observation overlap. The source and target sequences share little observation overlap for the green and red mugs. Despite the distinct viewing angles and occlusion (blue: from source, yellow: from target), our approach can still recover the shape and accurate object center from the NDFs representation, thereby successfully associating the target point clouds to the correct source mugs.

0.019 s per frame. The change detection operation with an expanding spatial tree executes at an average speed of 0.023 s per frame given the magnitude of the object number in our experiment setting (up to 40 mugs), resulting in an overall of 24 FPS for the complete pipeline, thus showing no apparent latency for online operation.

Results on Synthetic Sequences. We present the change detection results of the three methods in Table 5.1. Here, we set $d = 0.002$ m and $r = 0.3$ for NN. We apply a 6 DoF random transform to the target camera poses when running our approach so as to simulate the unavailability of motion capture systems during common mobile robot operations.

We can see that our method accurately detects all the changes without any false positives. On the one hand, NN and PMT respectively ignore four and one changes for the two pairs of mugs switching positions on table 5 and 7, as the similar viewing angles of these mugs during the source and target traverse induce high overlap between the target and source point clouds, thereby confounding the baselines by the existence of neighboring points and similar local geometry.

Table 5.1: Change Detection Results on the Synthetic Sequences. The best results are marked in bold.

Approach	TP	FP	FN	Precision	Recall
NN	8	4	4	66.7%	66.7%
PMT	11	4	1	73.3%	91.7%
Ours	**12**	**0**	**0**	**100%**	**100%**

On the other, both baselines falsely mark the four mugs on table 1 and 2 as *changed*, which is due to the remarkable viewing angle difference on these four mugs between the source and target trajectory. As explained in Fig. 5-7, the two mugs highlighted in red and green are observed in almost opposite directions, leading to little/no overlap between the two partial point clouds (blue and yellow). Therefore, few neighboring points can be found between the source and target observations, while at the same time, leading to drastic differences in the resulted TSDF values. In contrast, our method is capable of encoding the full mug shape with a compact shape code from the partial observations and hence produces no false positives.

Results on Real-world Sequences. We report change detection results of the three methods on the real sequence in Table 5.2 and present the reconstructions obtained from PMT and our method (for mugs) in Fig. 5-8. For a fair comparison, we obtain the mug point clouds for all three methods using the panoptic masks adopted in PMT, which are generated by Detectron2 [134] and preprocessed by DBSCAN [31]. Here, the parameters of NN are set as $d = 0.002$ m and $r = 0.4$, considering the noises in the real-world point clouds. Due to localization drift between two sessions, for PMT, the final number of *changed* objects are counted after the manual merging of the few submaps instantiated for the same object. In order to mimic the real operating scenarios, the camera poses to our approach are obtained through running ORB-SLAM3 separately on the two traverses.

When confronted with noisy point cloud inputs and localization results, despite the reconstruction failure of the beige mug on table 1 and the blue mug on table 2 (as shown Fig. 5-8(a)) due to poor point cloud and panoptic mask quality, our method still maintains better detection efficacy in terms of both precision and recall

Table 5.2: Change Detection Results on the Real-world Sequences. The best results are marked in bold.

Approach	TP	FP	FN	Precision	Recall
NN	3	3	2	50%	60%
PMT	**5**	3	**0**	62.5%	**100%**
Ours	**5**	**1**	**0**	**83.3%**	**100%**

rate. From Fig. 5-8(b), we see that the confusion brought by disparate viewing angles persists, as both NN and PMT wrongly mark the black and blue mug on table 3 (highlighted in green) as *changed*. This matches the PMT reconstruction results shown in the second column, as the two mugs have just one side partially reconstructed during each session. Furthermore, another false detection emerges as the dark blue mug on table 3 (highlighted in yellow in Fig. 5-8(a)), which results from the more significant localization error at the starting point of the target trajectory. Our method reduces its reliance on the absolute accuracy of camera pose estimation by conducting local neighborhood layout matching, i.e., the relative spatial relationship among neighboring mugs, thereby successfully recognizing the mug through its unchanged relative positions with its neighbors.

Limitation and Extension. The proposed approach works well under the assumption that the changed objects do not take up a predominant portion of all the objects in the scene. This is a common case for performing frequent visits to the same environment whose changes occur incrementally. For scenes with drastic changes between the two scans, our approach may not work well in the absence of a consistent local layout to refer to. Our approach can be further extended to detect objects that only rotate but do not move by utilizing the SO(3) equivariance of NDFs, i.e., computing an extra SO(3) transform between the two $\mathbf{z}$'s after the shape-similarity-based registration step. This situation was not included in the experiment as changing daily objects rarely have a pure rotation but no translation.

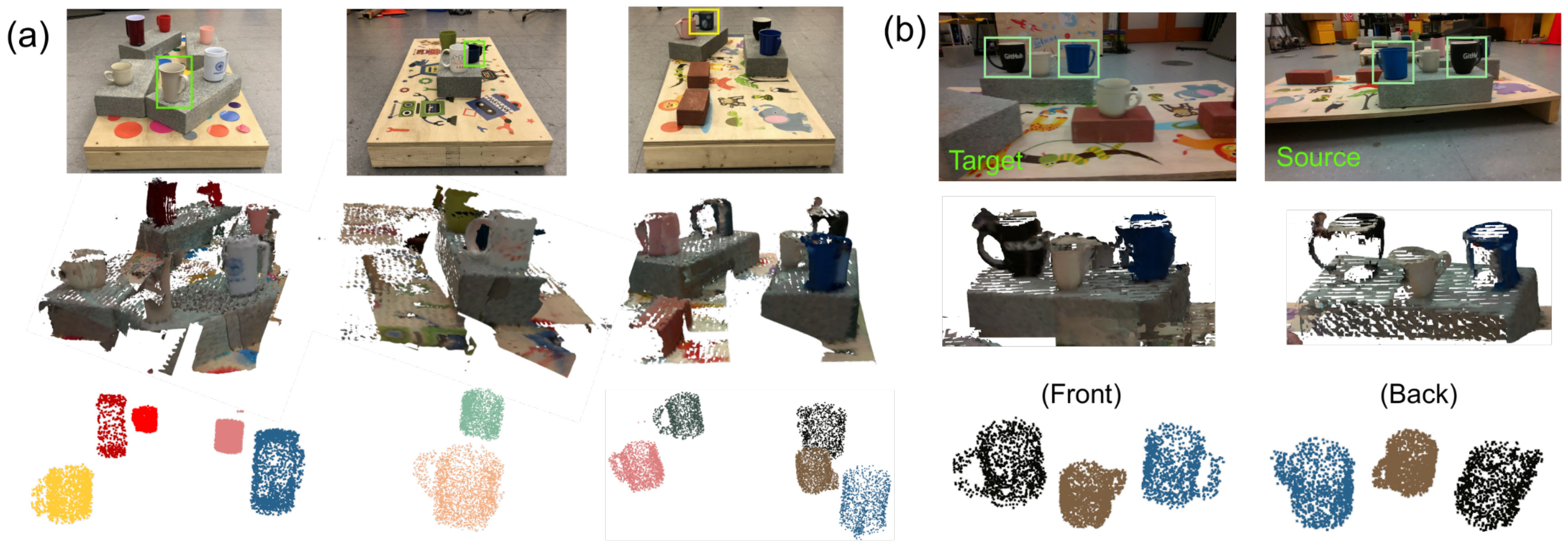


Figure 5-8: Reconstruction Results. Top to bottom: real scenes, results from PMT, and results from NDFs. (a) Full reconstruction of the source real-world sequence. Front views of the three tables (first row) are included for mug layout illustration. Fully trained in simulation, NDFs are able to reconstruct 12 out of the 14 unseen mugs given partial observations, and the two missing mugs are highlighted in green in the front view pictures. (b) PMT and NDFs reconstruction results for the two mugs falsely detected as changes. As shown in the first row, two mugs (highlighted in green) are observed in almost opposite views in the source and target sequence. As shown in the second row, the missing part in the back of the PMT reconstruction misleads PMT to mark these two mugs as changed, while our NDF-based approach correctly recognizes the mugs by virtue of the robust recovery of the full mug shape.

5.6 Conclusion

In this chapter, we propose an online object-level change detection approach based on an NDF-derived object representation, demonstrating improved robustness in viewing angle disparity and localization drift. For future work, we would like to test the approach's scalability to larger scenes if incorporated as part of an object-level SLAM system targeting long-term operation and further explore the potential of using NDFs for providing object pose constraints to help improve camera localization.

Chapter 6

NeuSE: Neural SE(3)-equivariant Embedding for Consistent Spatial Understanding with Objects

Recalling our major claim in Chapter 1 that object-based SLAM would increase SLAM's compatibility with advanced robotic tasks through object-level reasoning, in this chapter, we conclude our line of efforts by taking into account all the aforementioned concerns on object ambiguity and change detection, and present here our ultimate object SLAM paradigm to achieve robust spatial understanding against scene changes.

Therefore, we develop NeuSE, a novel **Neu**ral **S**E(3)-Equivariant **E**mbedding for objects, and illustrate how it supports object SLAM for consistent spatial understanding with long-term scene changes[1]. NeuSE is a set of latent object embeddings created from partial object observations. It serves as a compact point cloud surrogate for complete object models, encoding full shape information while transforming SE(3)-equivariantly in tandem with the object in the physical world. With NeuSE, relative frame transforms can be *directly* derived from inferred latent codes. Our proposed SLAM paradigm, using NeuSE for object shape and pose characterization, can operate independently or in conjunction with typical SLAM systems. It di-

[1]Project page: https://neuse-slam.github.io/neuse/

rectly infers SE(3) camera pose constraints that are compatible with general SLAM pose graph optimization while also maintaining a lightweight object-centric map that adapts to real-world changes. Our approach is evaluated on synthetic and real-world sequences featuring changed objects and shows improved localization accuracy and change-aware mapping capability, when working either standalone or jointly with a common SLAM pipeline.

6.1 Introduction

The ability to conduct consistent object-level reasoning is crucial for many high-level robotic tasks, especially those involving repetitive traversal in the same environment, such as household cleaning and object retrieval. In a constantly evolving world, robots are expected to accurately locate themselves and their target while keeping an updated map of the environment, ensuring that a specific "blue coffee mug" can always be retrieved regardless of its location since its last use.

Traditional Simultaneous Localization and Mapping (SLAM) approaches [11, 27, 62] see the world through a static set of low-level geometric primitives extracted from observations, making themselves less amenable to human-like reasoning about the world. In the absence of semantic information, these unordered collections of points, lines, or planes are not completely compatible with object-level interpretation, making them susceptible to false correspondence matches when faced with scene changes over time.

As the world changes and operates under the minimal unit of objects, objects serve as an intuitive source for assisting localization, and an object-centric map can act as a lightweight and flexible reflection of the latest environment layout. To bridge the communication between objects and typical SLAM systems, previous works have experimented with various object representations to guide back-end optimization, ranging from pre-defined object model libraries [103, 121], semantic segmentation masks [75, 100, 76, 136], to parameterized geometry [87, 49, 138]. However, they are confined to either a limited number of objects or a loss of geometric details due to

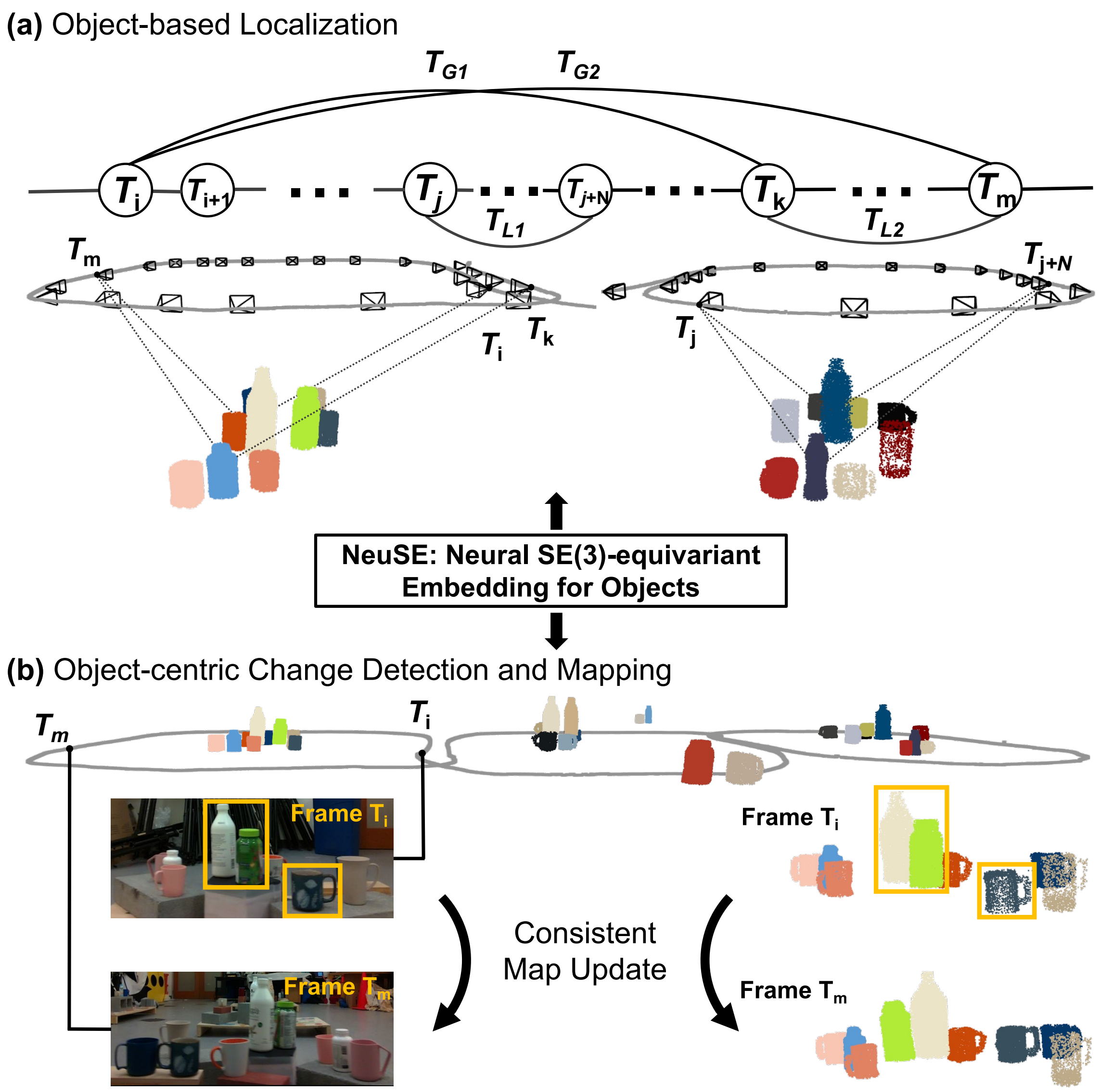

(a) Object-based Localization
T_{G1}
T_{G2}
T_i
T_{i+1}
T_j
T_{j+N}
T_k
T_m
T_{L1}
T_{L2}
NeuSE: Neural SE(3)-equivariant Embedding for Objects
(b) Object-centric Change Detection and Mapping
Frame T_i
Frame T_m
Consistent Map Update

Figure 6-1: Schematic of consistent spatial understanding with NeuSE. An object-centric map of mugs and bottles constructed from a real-world experiment is shown for illustration. (a) NeuSE acts as a compact point cloud surrogate for objects, encoding full object shapes and transforming SE(3)-equivariantly with the objects. Latent codes of bottles and mugs from different frames can be effectively associated (dashed line) for *direct* computation of inter-frame transforms, which are then added to constrain camera pose (T_i) optimization both locally (T_{Li}) and globally (T_{Gi}). (b) The system performs change-aware object-level mapping, where changed objects (highlighted in orange) are updated alongside unchanged ones with full shape reconstructions in the object-centric map.

partial reconstruction or simplification of object shapes.

Recently, neural implicit representations have been introduced [115, 128, 147, 116] to SLAM as object or scene representations, working with probabilistic rendering loss to help constrain camera localization. However, the rendering process is parameterized as a neural network with no physical meaning, thus requiring iterative optimization with a good initialization to gradually reflect the correct SE(3) camera pose constraint embedded within the observation. This incurs extra training and computation overhead and thus makes the integration of neural representations a cumbersome process.

In order to leverage the shape description power of neural representations while bypassing the undesirable iteration, we, therefore, break with the dominant *"render-optimize"* convention in previous works by explicitly imposing SE(3)-equivariance onto the vanilla representation.

Hence, we introduce NeuSE, a novel category-level **Neu**ral **S**E(3)-Equivariant **E**mbedding for objects. NeuSE learns a latent canonical point cloud from partial object observations, encoding the full object shape while transforming SE(3)-equivariantly as the object transforms in the physical world. Consequently, relative frame transforms can be *directly* computed from the corresponding latent codes of an object when it is observed in different frames. To account for pose ambiguity arising from symmetrical geometry, we further train NeuSE's behaviors to conform to object geometric ambiguity. In this way, working with NeuSE is akin to working with the full object model, only with operations applied to a compact latent point cloud surrogate with known correspondences.

In this chapter, we present NeuSE and further demonstrate how it supports object SLAM targeting spatial understanding with long-term scene inconsistency (see Fig. 6-1). By using NeuSE for object shape and pose characterization, we unify the representations of major SLAM modules, e.g., data association, pose constraint derivation, etc., around one versatile latent code. Our proposed approach can either work standalone or complement common SLAM systems by *directly* inferring SE(3) camera pose constraints compatible with general SLAM pose graph optimization and

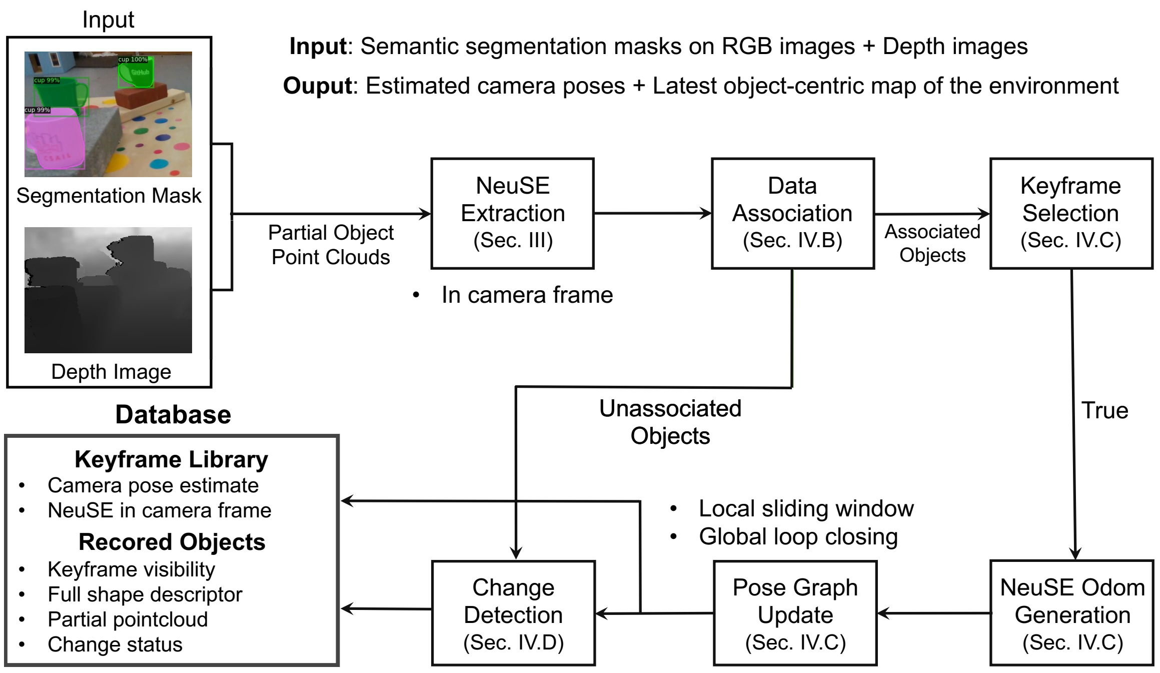

Input
Segmentation Mask
Depth Image
Input: Semantic segmentation masks on RGB images + Depth images
Ouput: Estimated camera poses + Latest object-centric map of the environment
Partial Object Point Clouds
NeuSE Extraction (Sec. III)
• In camera frame
Data Association (Sec. IV.B)
Associated Objects
Keyframe Selection (Sec. IV.C)
Unassociated Objects
True
Database
Keyframe Library
• Camera pose estimate
• NeuSE in camera frame
Recored Objects
• Keyframe visibility
• Full shape descriptor
• Partial pointcloud
• Change status
• Local sliding window
• Global loop closing
Change Detection (Sec. IV.D)
Pose Graph Update (Sec. IV.C)
NeuSE Odom Generation (Sec. IV.C)

Figure 6-2: System overview. We propose a NeuSE-based object SLAM approach targeting consistent spatial understanding with long-term scene changes.

maintaining a lightweight object-centric map with change-aware mapping ability (see Fig. 6-2). Our main contributions are as follows:

- We introduce NeuSE, a neural SE(3)-equivariant embedding for objects, encoding the full object shape and transforming SE(3)-equivariantly with the real-world object.
- We propose a NeuSE-based object SLAM paradigm targeting long-term scene inconsistency, enabling NeuSE-predicted object-level localization and change-aware mapping.
- We evaluate our approach on both synthetic and real-world sequences and demonstrate improved localization performance and flexible mapping capability when working standalone or jointly with a common SLAM pipeline.

6.2 Related Work

In this section, we discuss a list of robotic applications of neural implicit representations and, in particular, compare the pros and cons of various efforts in adopting them into the SLAM pipeline.

Neural implicit representations have emerged as a promising tool to encode the underlying 3D geometry of objects and scenes [95, 77, 93]. Different works have explored how neural implicit representations can be used in various fields, including change detection [34], localization [1, 82], SLAM [142, 14, 116, 147, 99], and manipulation [54, 53, 140, 13, 112, 70, 60, 109, 102, 68].

Notably, some works extend the original representation by integrating SO(3) or SE(3) equivariance for tasks such as reconstruction [19], point cloud registration [146, 69], and manipulation [111]. Zhu et al. [146] learned SO(3)-equivariant features to perform correspondence-free point cloud registration, while Lin et al. [69] used SE(3)-equivariant representations to obtain and refine the registration result globally and locally. Simeonov et al. [111] learned SE(3)-equivariant object representations for manipulation and estimated relative transforms through optimization. These methods focused on point clouds known to be associated with the same object. They could suffer from performance degradation for partially overlapped point clouds [146, 69] or required iterative refinement to recover the desired relative transform [111].

In the context of SLAM, most works, other than the object-based methods listed in the previous section, utilize scene-level neural implicit representations to be jointly optimized with camera poses. iNeRF [139] is the first to demonstrate that camera poses can be derived from a neural radiance field (NeRF) representation of the scene. iMap [116] showed that a multilayer perceptron (MLP) could serve as the scene representation for real-time RGB-D SLAM. NICE-SLAM [147], built on top of iMap, further introduced a hierarchical grid-based neural encoding, enabling RGB-D SLAM on a larger scale. In terms of monocular SLAM, recently, Orbeez-SLAM [14] adopted traditional feature-based SLAM for pose initialization and leveraged NeRF to obtain a hierarchical volumetric map of the environment. NeRF-SLAM [99] relied on an indirect loss for pose estimation and produced higher-quality reconstructions by supervising the radiance field with depth information.

As the rendering process is parameterized as a neural network with no interpretable meaning, these methods require iterative optimization with photometric or depth loss, as well as a proper initialization to obtain the SE(3) transform constraint that aligns with the real-world observation. This results in added training and computational expenses, making the adoption of neural representations a complex process and hard to adapt to changes with the scene represented as one single code.

Our NeuSE-based SLAM paradigm distinguishes itself from prior SLAM works with neural representations by further explicitly imposing SE(3)-equivariance onto the vanilla neural object representations. To handle unknown data associations, in contrast to the previous works on point cloud registration or manipulation with equivariant representations, we take a step beyond to enforce shape code consistency across viewing angles. This allows partial point clouds to be matched regardless of viewing angle differences. With additional regularization on objects with pose ambiguity, we ultimately achieve *direct* inference of SE(3) camera pose constraints from partial object representations. This eliminates the need for the computationally expensive "render-optimize" process and offers a lightweight as well as flexible solution to object SLAM problems with long-term changes.

6.3 Category-level Neural SE(3)-equivariant Embedding (NeuSE) for Objects

We propose to represent each object in a scene by using a corresponding SE(3)-equivariant latent embedding. Precisely, given a point cloud $\mathbf{P} \in \mathbb{R}^{N\times 3}$, we represent it with a lower dimensional latent embedding ("a canonical latent point cloud") $\mathbf{z} \in \mathbb{R}^{D\times 3}$, inferred using a neural network encoder f so that $\mathbf{z} = f(\mathbf{P})$. The underlying latent embedding is equivariant, so that for any SE(3) transform $\mathbf{T}$:

$$\mathbf{T}\mathbf{z} = f(\mathbf{T}\mathbf{P}), \tag{6.1}$$

i.e. the latent embedding $\mathbf{z}$ transforms equivariantly with respect to the point cloud $\mathbf{P}$.

By representing objects using this equivariant embedding, we obtain the following three benefits:

Latent Pose Constraints. The underlying latent embedding space operates under the same SE(3) action as point clouds. Thus, we may express pose constraints between matched objects directly in the latent space as opposed to the full point cloud space of objects. As the latent space is both low dimensional and canonical, pose constraints may be more efficiently computed with the closed-form solution developed by Horn [48].

Implicit Pose Representation. The object latent code implicitly captures the underlying SE(3) transform of an object. This circumvents the need to explicitly specify 6DOF poses of objects when computing pose constraints, which may not always be accessible and can be ill-defined for objects with symmetrical ambiguity.

Implicit Shape Representation. The object latent code richly encodes both the underlying shape and features of an object, which then allows for robust data association against viewing angle disparity.

To infer SE(3)-equivariant latent codes, NeuSE uses a SO(3)-equivariant encoder function [19] $f_\theta(\mathbf{P}) = \mathbf{z}$ that maps a partial object point cloud $\mathbf{P}$ into a global latent point cloud $\mathbf{z}$, and a decoder function $\Phi(\mathbf{x}, \mathbf{f}_\theta(\mathbf{P}))$ that maps an input query point $\mathbf{x}$ to its predicted occupancy value according to $\mathbf{z}$:

$$\begin{aligned} f_\theta(\mathbf{P}) &= \mathbf{z} : \mathbb{R}^{n\times 3} \to \mathbb{R}^{k\times 3} \\ \Phi(\mathbf{x}, \mathbf{f}_\theta(\mathbf{P})) &= \Phi(\mathbf{x}, \mathbf{z}) : \mathbb{R}^3 \times \mathbb{R}^{k\times 3} \to [0, 1]. \end{aligned} \tag{6.2}$$

By feeding $\Phi(\cdot,\cdot)$ with a point cloud $\mathcal{X}$ obtained via uniform sampling within a large bounding box centered around $\mathbf{P}$, the full shape point cloud $\mathcal{S}$ of the object can be reconstructed in terms of the predicted occupancy values with $\mathcal{S} = \{\mathbf{x} | \Phi(\mathbf{x}, f_\theta(\mathbf{x}|\mathbf{P})) > v_0, \mathbf{x} \in \mathcal{X}\}$, where v_0 is the threshold to mark whether a point location is occupied.

6.3.1 Learning SE(3)-equivariance across Viewing Angles

We construct SE(3)-equivariance separately through rotation and translation equivariance.

For rotation equivariance, as our encoder is rotation equivariant, when a point cloud is rotated by $\mathbf{R}$, the inferred latent code will be equivalently rotated by $\mathbf{R}$:

$$f_\theta(\mathbf{RP}) = \mathbf{Rz}, \mathbf{R} \in \mathbb{SO}(3). \tag{6.3}$$

Since $\mathbf{P}$ is a partial observation of the complete object geometry, we treat this partial center $\overline{\mathbf{P}}$ as an initial estimate of the actual object translation so as to learn an approximate translation equivariant latent $\mathbf{z}$. We first infer $\mathbf{z}_0$ for the zero-centered partial point cloud $\mathbf{P} - \overline{\mathbf{P}}$. The final latent $\mathbf{z}$ for point cloud $\mathbf{P}$ is obtained by adding back the partial center $\mathbf{z} = \overline{\mathbf{P}} + \mathbf{z}_0$. Hence, to infer an SE(3)-equivariant $\mathbf{z}$, the final formulation of Eq. (6.2) is accordingly written as:

$$\begin{aligned} & f_\theta(\mathbf{P} - \overline{\mathbf{P}}) = \mathbf{z}_0 : \mathbb{R}^{n\times 3} \to \mathbb{R}^{k\times 3} \\ & \mathbf{z} = \mathbf{z}_0 + \overline{\mathbf{P}}, \mathbf{z}' = \mathbf{z} - \overline{\mathbf{z}} \\ & \Phi(\mathbf{x}, \mathbf{f}_\theta(\mathbf{P})) = \Phi(\mathbf{x} - \overline{\mathbf{z}}, \mathbf{z}') : \mathbb{R}^3 \times \mathbb{R}^{k\times 3} \to [0, 1], \end{aligned} \tag{6.4}$$

where $\overline{\mathbf{z}}$ is the center of $\mathbf{z}$. The translational equivariance on $\mathbf{z}$ is imposed by training the center of $\mathbf{z}_0$ to learn the offset between $\overline{\mathbf{P}}$ and the true object center (translation). Ultimately, for the same object observed partially with camera view $\mathbf{T}_1$ and $\mathbf{T}_2$, the SE(3)-transform $\mathbf{T}_{1,2} = (\mathbf{R}, \mathbf{t})$ between the two latent point clouds $\mathbf{z}_1$ and $\mathbf{z}_2$, which is expected to be close to $\mathbf{T}_2^{-1}\mathbf{T}_1$, can be obtained by:

$$\mathbf{T}_{1,2} = (\mathbf{R}, \mathbf{t}) = \Psi(\mathbf{z}_1, \mathbf{z}_2), \tag{6.5}$$

where $\Psi(\cdot,\cdot)$ is Horn's method [48] with the closed-form solution of the relative SE(3)-transform between two point clouds with known correspondence.

6.3.2 Dealing with Pose Ambiguity

SE(3)-equivariance is desirable for unraveling the relative transform between the two frames where the same object is observed. However, shape symmetry can result in ambiguity in the inferred transform, causing our latent code to be fallible when the transform selected is one of many possibilities instead of the correct one. To make our representations applicable to a broader range of objects, we, therefore, propose separate training objectives for object shapes with and without ambiguity w.r.t. the camera viewing frustum.

Unambiguous Objects. For objects without pose ambiguity (e.g., mugs with a handle), the transform $(\mathbf{R}, \mathbf{t})$ obtained from Eq. (6.5) should be unique and thus approximate the true inter-frame camera transform. We therefore simply minimize the L_2 distance between the estimated transform $(\hat{\mathbf{R}}_{3\times3}, \hat{\mathbf{t}}_3)$ and the ground truth $(\mathbf{R}_{3\times3}, \mathbf{t}_3)$:

$$L_{transform}^{uab} = ||(\hat{\mathbf{R}}\mathbf{R}^T) - \mathbf{I}_{3\times3}||_F^2 + ||\hat{\mathbf{t}} - \mathbf{t}||_2^2, \tag{6.6}$$

where $||\cdot||_F^2$ is the Frobenius norm.

Ambiguous Objects. We limit "ambiguous objects" to objects with pose ambiguity from their shapes (e.g., upright wine bottles), but not the ones that may appear ambiguous due to occlusion (e.g., mugs with their handles obscured).

Since ambiguous objects have multiple or infinite possible transforms that can meet the current observation, the exact single correct transform can never be learned. We instead wish that the derived transform will always lead to similar object shapes when transforming the object's point cloud from one frame to another. In a nutshell, we require the latent code $\mathbf{z}$ to implicitly learn the distribution of the possible transforms.

Hence, given the full object point clouds in two frame coordinates, $\mathbf{P}_{o1}$ and $\mathbf{P}_{o2}$ (readily available as we train fully in simulation), we enforce that the Chamfer dis-

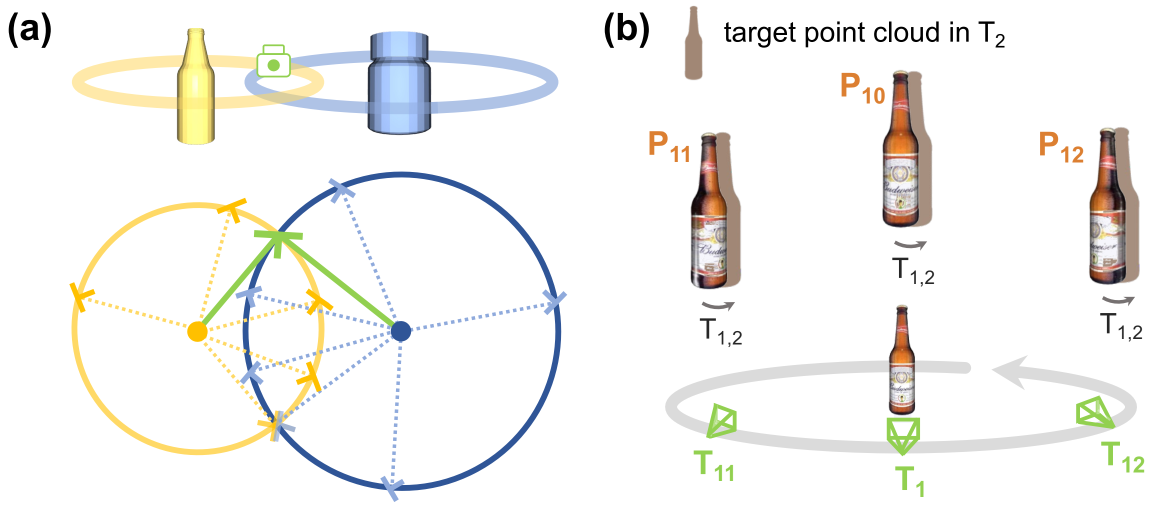


Figure 6-3: (a) Breaking pose ambiguity with covisible ambiguous objects. Motions around a bottle's axis of symmetry result in seemingly identical observations, making it impossible to determine inter-frame transformations. However, with two covisible bottles, the intersection (green) of their camera pose distributions (yellow and blue) for the current observation reveals the true camera pose, where inter-frame transforms can then be determined without ambiguity. (b) Latent symmetry. The canonicalized latent embedding should be invariant with camera motion ($\mathbf{T}_{1i}$) around the object's axis of symmetry, inducing consistently small Chamfer distance between the transformed bottle ($\mathbf{T}_{1,2}\mathbf{P}_{1i}$) and the target point cloud.

tance between the two point clouds should be small after aligning them with the predicted transform:

$$\begin{aligned} L_{amb} =& CD(\mathbf{T}_{1,2}\mathbf{P}_{o,1}, \mathbf{P}_{o,2}) \\ CD(\mathbf{P}_1, \mathbf{P}_2) =& \frac{1}{|\mathbf{P}_1|} \sum_{\mathbf{x}\in\mathbf{P}_1} \min_{\mathbf{y}\in\mathbf{P_2}} ||\mathbf{x}-\mathbf{y}||_2^2 + \\ & \frac{1}{|\mathbf{P}_2|} \sum_{\mathbf{y}\in\mathbf{P}_2} \min_{\mathbf{x}\in\mathbf{P_1}} ||\mathbf{x}-\mathbf{y}||_2^2. \end{aligned} \tag{6.7}$$

We can recover the exact transform that simultaneously justifies all current object observations by intersecting the distributions of possible transforms for multiple ambiguous objects (see Fig. 6-3(a) for the reasoning of the base 2-object case concerning two bottles), or further refine the predicted one when working together with unambiguous objects. Note here we do not account for the rare degenerate case of colinear axes of symmetry for all visible objects.

Furthermore, to facilitate the learning of the underlying distribution, we further augment the original ($\mathbf{P}_{10}$, $\mathbf{P}_{20}$) pair to include extra samples in the distribution. Given camera view $\mathbf{T}_1$ and $\mathbf{T}_2$, we fix $\mathbf{T}_2$ and generate N random transforms $\mathbf{T}_{1i}$s that allow for camera movement around the object's axis of symmetry with seemingly identical observations as that from $\mathbf{T}_1$ (Fig. 6-3(b)). The resulting N object point clouds in corresponding camera frames, $\mathbf{P}_{1i}$s, should retain similar shapes to $\mathbf{P}_2$ using the predicted transform. Hence, the ultimate training objective for ambiguous objects is:

$$L^{amb}_{transform} = \sum_{i=0}^{N} CD(\mathbf{T}_{1,2}\mathbf{P}_{1i}, \mathbf{P}_{20}), \tag{6.8}$$

where N and values of $\mathbf{T}_i$ are determined by the type, e.g., cylindrical (360°) or cubical (180°), of object ambiguity. Here in our experiment, we set $N = 180$ and draw transforms from $[0°, 360°]$ circulation around the cylindrical bottles.

Finally, the target inter-frame transform can be similarly obtained using Eq. (6.5), with the two latent code $\mathbf{z}$s formed by concatenating all corresponding $\mathbf{z}_i$s of covisible objects in each frame.

6.3.3 Shape Consistency across Viewing Angles

Since $\mathbf{z}_0$ is SO(3)-equivariant, its rotation invariant part, $\mathbf{s} \in \mathbb{R}^k$, encoding *full object shapes*, can then be extracted as $\mathbf{s} = \{s_i\}_{i=1}^{i=k} = ||(\mathbf{z}_0)_i||_2$, where we term $\mathbf{s}$ as the shape descriptor.

Following [34], we adopt the batch-hard shape similarity loss L_{b_shape}, enforcing $\mathbf{s}$ to be consistently similar across viewing angles of the same object while discriminatively far apart for different objects.

L_{b_shape} takes the form of the triplet loss as [*anchor, positives, negatives*]. To allow for a variety of viewing angle combinations during training, we populate each training batch B with M partial observations for each of the N randomly drawn objects. Samples of the same object instance serve as mutual *anchors* and *positives*, $(\mathbf{A}_i, \mathbf{P}_i)$, with samples not from the current shape instance being the *negatives*, $\mathbf{N}_i$. L_{b_shape} is calculated in a "batch-hard" fashion, i.e., it only uses the most dissimilar $(\mathbf{A}, \mathbf{P})$ and the most similar $(\mathbf{A}, \mathbf{N})$ for each anchor to guide the training. With $D(\cdot, \cdot)$ as the cosine similarity, the final batch-hard shape similarity loss is formulated as:

$$\begin{aligned} L_{b\_shape} = \frac{1}{|B|} \sum_{i=1}^{N} \sum_{j=1}^{M} (&- \min_{k \in [1,M]} D(o_{ij}, o_{ik}) \\ &+ \max_{m \neq i} D(o_{ij}, o_{mn})), \end{aligned} \tag{6.9}$$

where o_{ij} is the jth observation of object i within the batch.

6.3.4 Training in Simulation

Training Objective. NeuSE is trained with partial object point clouds and corresponding 3D occupancy voxel grids of objects' complete geometry. The full model $[f_\theta, \Phi]$ predicts the complete 3D occupancy values at query object locations, which is then evaluated by the standard cross-entropy classification loss $L_{occ} = \mathcal{L}(\Phi(\mathbf{p}, f_\theta(\mathbf{P}), v))$ with sampled query location $\mathbf{p}$ and its corresponding true occupancy value v.

The ambiguous and unambiguous object categories are trained separately, with respective $L_{transform}$ and shared L_{occ} and L_{shape}. The final training objective is the weighted sum of the three losses

$$L = L_{occ} + \beta_1 L_{transform} + \beta_2 L_{b\_shape}, \tag{6.10}$$

where β_1 and β_2 are constants set to balance the order of magnitude of the three losses. The training samples are organized following L_{b_shape}'s formulation, where L_{occ} is evaluated for each sample in B and $L_{transform}$ for any two observations of the same object. With this composition of the training data, the model is expected to see various pairs of viewing angles and learn to predict the relative transform between two frames within a certain range apart.

Data Generation. NeuSE is trained fully in simulation with RGB-D images rendered with Pybullet [15]. We place a randomly posed principal object on the table, along with 2-4 (for unambiguous objects) and 1-2 (for ambiguous objects) objects arbitrarily selected from the trained categories to simulate a typical cluttered environment. In light of the viewing angle variety, for each multi-object layout, we uniformly sample a fixed number of camera locations over the hollow cubical space centered around the table. The cubical space is set to be $[d_n, d_f]$ away from the table within the table plane and $[d_l, d_h]$ away from the table in the vertical direction, thus accounting for observations from near, far, low, and high locations.

6.4 NeuSE-based Object SLAM with Long-term Scene Inconsistency

NeuSE enables robust data association across viewing angles and further serves as a lightweight, alternative "sensor" for providing cross-frame camera pose constraints. We propose a NeuSE-based localization strategy in tandem with a change-aware object-centric mapping procedure to enable robust robotic operation in scenes with long-term changes.

6.4.1 System Formulation and Update

Our object-based SLAM problem is formulated as a pose graph consisting of only keyframe camera pose vertices, where an edge exists to constrain the two vertices if there are inter-frame transform measurements available from NeuSE or any other sources. The measurement error between vertex i and j for each edge is defined as $\mathbf{e}_{ij} = log(\mathbf{Z}_{ij}\hat{\mathbf{T}}_j^{-1}\hat{\mathbf{T}}_i)^{\vee}$, where $\mathbf{Z}_{ij}$ is the odometry measurement from arbitrary sources between frame i and j, and $\hat{\mathbf{T}}$ is the current estimate of $\mathbf{T}$.

The system maintains a library of keyframes with the latest camera pose estimates obtained via pose graph updates, as well as NeuSE latent codes of the observed objects in the frame coordinate. The camera pose of the current frame is recovered as the smoothed estimate of pose constraints from associated objects and external sources between the frame itself and the nearest keyframe.

The objects in the system are recorded by their per-keyframe visibility, change status, a partial point cloud from their last keyframe observation (for query points generation during rendering), and the latest shape descriptor from initialization or mapping updates.

For localization, the system works only with latent codes in the local camera frame, while their world-frame counterparts are used for mapping operations. When an object is first observed, its world-frame latent code is initialized and then updated as needed by averaging the back-projected latent codes of the same object using the latest camera pose estimates recorded in the keyframe library.

6.4.2 Data Association

NeuSE-predicted inter-frame transforms are only valid if computed from latent codes belonging to the same object. Our data association scheme exploits both full shape similarity and spatial proximity so as to allow pose constraint generation only between latent codes with reliable object association.

Shape Similarity. For each object in the current frame, we extract the shape descriptor from the latent code and calculate its cosine shape similarity (as adopted in Eq. (6.9)) with all objects in the library. Objects with a shape similarity score greater than δ_{shape} are considered potential data association candidates $\mathcal{O}_c$. If no similarity scores exceed δ_{shape}, a new object instance is initialized and added to the object library.

Spatial Proximity. Spatial proximity involves examining the Euclidean distance between the partial point cloud center of the current object and its candidates in $\mathcal{O}_c$, where the current partial center is projected to the latest keyframe its candidate is last seen. The transform for projection is calculated using Horn's method (Eq. (6.5)) between corresponding latent codes. Candidate with the smallest distance while below δ_{prox} is deemed a successful match to be included in $\mathcal{O}_{matched}$ for further pose constraint generation. Otherwise, the current object is unassociated and grouped into $\mathcal{O}_{unmatched}$.

The procedure is performed first on unambiguous objects and later on ambiguous objects, differing only in the acquisition of inter-frame transforms. For unambiguous objects, we compute the transform directly using Horn's method. For ambiguous objects, we utilize the transform from associated unambiguous objects if available. If not, we conduct an exhaustive search of all paired combinations of covisible object candidates in previous keyframes and obtain the inter-frame transform from the concatenated object latent codes.

We hence divide all covisible objects $\mathcal{O}$ in one frame into three groups: (1) $\mathcal{O}_{matched}$, which has objects with shape and spatial consistency and is adopted for pose constraint generation, (2) $\mathcal{O}_{unmatched}$, which consists of scene changes or temporally ambiguous observations, and is processed by change detection, and (3) new objects never seen before.

6.4.3 Pose Graph Optimization

With objects successfully associated across frames, we compute NeuSE-predicted transforms among frames so as to constrain the pose graph both locally and globally (see Fig. 6-4).

Keyframe Selection. Keyframes are selected based on the presence of new

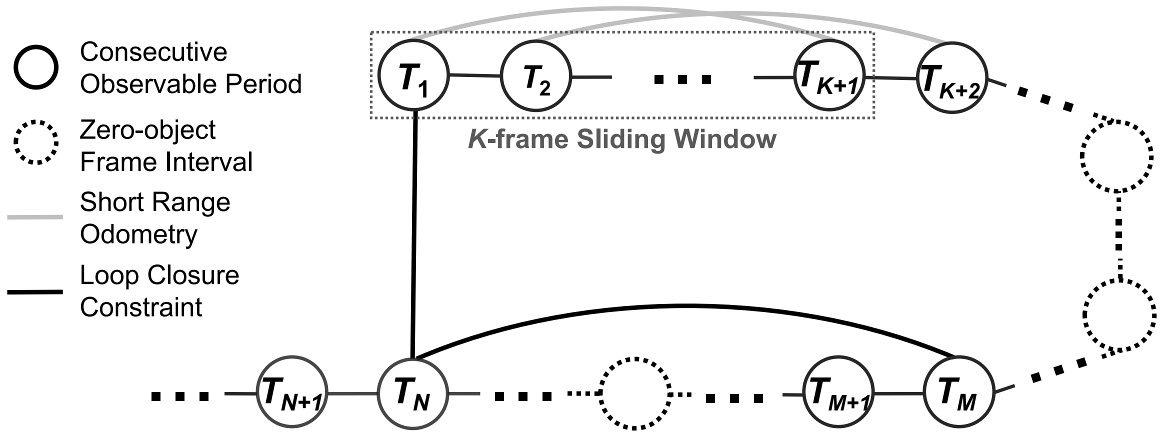


Figure 6-4: Pose graph optimization. With objects observed in periods of consecutive frames, we derive from corresponding latent codes (1) short-range odometry constraints (grey) within a local K-frame sliding window, and (2) global loop closure constraints (black) between the current ($\mathbf{T}_N$) and the first frame of each of its previous consecutive observable periods ($\mathbf{T}_1$ and $\mathbf{T}_M$), which are working jointly to constrain the pose graph optimization.

objects and proximity to previous keyframes. New objects trigger the selection of a frame as a keyframe, and frames located at least 0.04 m away from the previous keyframe based on accumulated odometry are also chosen. Additional keyframes may be added after change detection for frames with changes.

Short-range Odometry. To reduce local drift in frames with persistently observed objects, short-range NeuSE-predicted pose constraints are applied to a sliding window optimization of K keyframes. For each newly added keyframe, we search its preceding $K-1$ keyframes and identify the common objects observed between the current and previous frames. The inter-frame transform constraint is computed based on the concatenated latent codes of the shared objects (or a single unambiguous object) and then added as an edge to the pose graph.

Long-range Loop Closing. Global loop closing is activated when an object is detected again in a frame after its last consecutive observable period. The common objects between the current frame and the initial frames of all its previous observation periods are identified, and relative transform constraints are derived from the concatenated NeuSE latent codes. These constraints are then added to the pose graph, which initiates a global optimization process using the latest pose estimates from the

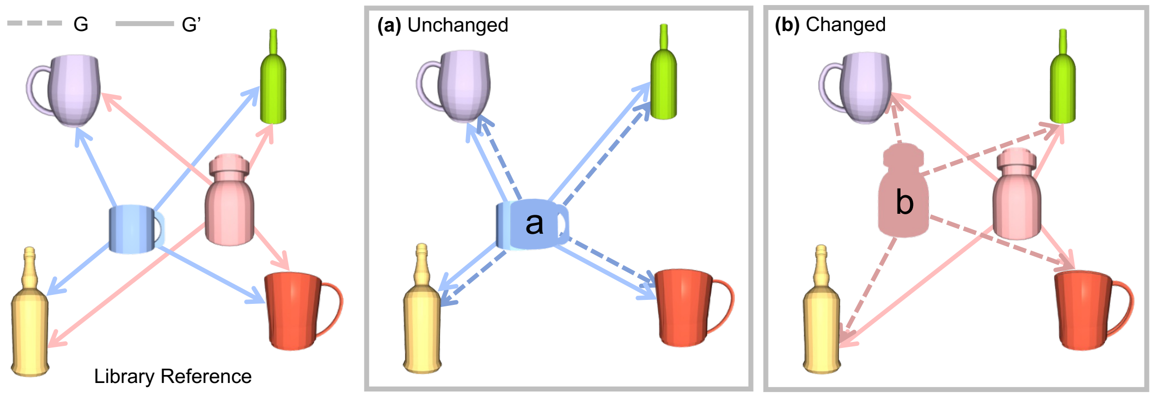


Figure 6-5: Object layout comparison through graph matching. Object graphs are constructed for the current frame (G) and the library (G'). For object a and b, which are similar in shape to the blue mug and pink bottle in the library, respectively, the inter-object distance between them and the anchor objects in the four corners are computed and compared. (a) All corresponding edges (dashed and solid lines) with anchor objects have similar oriented lengths, indicating that the mug is unchanged but was seen with an occluded handle, leading to a false, ambiguous transform by the latent code. (b) There are no similar edges, indicating a different layout with the bottle moved.

local sliding-window optimization as the starting point.

6.4.4 Change-aware Object-centric Mapping

Change detection is performed frame-by-frame on objects in $O_{unmatched}$ that match in shape but are identified as spatially apart based on latent codes, providing a foundation for consistent long-term mapping.

As changes are often gradual and occupy a small portion of the object clutter in long-term scenes, here, change detection is done by comparing the relative layout of the query unmatched object $o_{ui} \in \mathcal{O}_{unmatched}$ with all objects $o_{mi} \in \mathcal{O}_{matched}$ in the matched set serving as anchors. We argue that the relative object position disparity is more robust to camera pose drift compared to the absolute position difference, as all objects observed will be drifting concurrently in the world frame.

We represent the local layout with a directed object graph G constructed with $\mathcal{O}_{unmatched}$ and $\mathcal{O}_{matched}$. Each vertex of G represents an object o with its shape descriptor and the true object center as $(\mathbf{s}, \mathbf{c})$. The center $\mathbf{c}$ is computed from the full object reconstruction using the decoding steps in Eq. (6.4) and back-projected to the world frame using the latest camera pose estimate. Edges are established between objects $o_{ui} \in \mathcal{O}_{unmatched}$ and all anchor objects $o_{mj} \in \mathcal{O}_{matched}$, indicating the oriented distance between their centers $E = \{\mathbf{e}_{ij} | \mathbf{e}_{ij} = \mathbf{c}_{ui} - \mathbf{c}_{mj}, \forall o_{ui} \in \mathcal{O}_{unmatched}, o_{mj} \in \mathcal{O}_{matched}\}$.

We build the local and reference object graph, G and G', respectively, for $\mathcal{O}$ in the current frame and their associated or shape-similar counterparts in the system library (see Fig. 6-5). After a quick alignment of the two graphs using the centers of anchor objects, for each pair of edges $(\mathbf{e}_{ij}, \mathbf{e}_{i'j'})$ connecting vertices of similar shapes (determined by $\mathbf{s}$ from data association), we compare their edge value disparity to assess if this is a changed layout:

$$\sum_j \mathbb{1}(|\mathbf{e}_{ij} - \mathbf{e}_{i'j'}| \leq \delta_e) = \begin{cases} 0, & \text{changed} \\ \geq 1, & \text{unchanged.} \end{cases} \tag{6.11}$$

An object o_i is marked as *unchanged* if at least one pair of edges is found to be closer than a threshold δ_e. This indicates that its inter-spatial relationship with at least one of the anchor objects is consistent. If no edges are found to be close, the object is marked as *changed*, and its change status and partial point cloud are updated in the object library. Here, we define an object to be "removed" from the scene if it has never been shape-matched in frame periods with global loop closure.

Therefore, we are able to maintain a lightweight, object-centric map that accurately reflects the full object reconstructions from NeuSE predictions. By using objects as the basic building blocks of the map, we can update changes seamlessly by replacing the old latent code with the new one during the decoding stage, avoiding the cumbersome and artifact-prone point- or voxel-wise modifications commonly used in traditional low-level geometric maps.

6.5 Experiments and Results

We aim to assess the efficacy of NeuSE for object shape/pose characterization and robot spatial understanding. Specifically, we would like to answer two questions: (1) Can NeuSE-based object SLAM perform reliable localization on its own or improve existing results when combined with other SLAM measurements, especially in the presence of temporal scene inconsistency? (2) Can the proposed approach build a consistent object-centric environment map with timely updates to reflect scene changes? We train NeuSE fully in simulation and evaluate the proposed algorithm directly on both synthetic and real-world sequences consisting of unseen objects of the trained categories, where objects are added, removed, and switched places to simulate long-term environment changes.

6.5.1 Datasets

Given the limited availability of object model collections for training and the scarcity of public data with appropriate object-level scene changes, we created our own synthetic and real-world sequences. The collected data feature mugs and bottles in various cluttered arrangements, with diverse occlusion patterns, various viewing angles, and gradual object changes. We chose mugs and bottles as the representative object categories due to their common use and distinct unique (mugs) or ambiguous cylindrical (bottles) shapes for localization, which allow us to evaluate the effectiveness of our latent code design. Following past work [111, 77, 95], our approach should be extendable to even more categories by incorporating related objects into training.

Synthetic Sequences. An environment is rendered in Pybullet with 50 previously unseen ShapeNet [12] mugs and bottles scattered onto ten tables in a 10×15 m^2 area (Fig. 6-6(a)). To fully examine the proposed SE(3)-equivariance of NeuSE, two object layouts are generated: (1) a roughly planar layout with all upright objects, and (2) a non-planar hilly layout with nearly half of the objects laid down and arbitrarily oriented on tabletops. The camera follows a preset closed-loop trajectory and records RGB-D images and segmentation masks of both layouts, respectively. This leads to

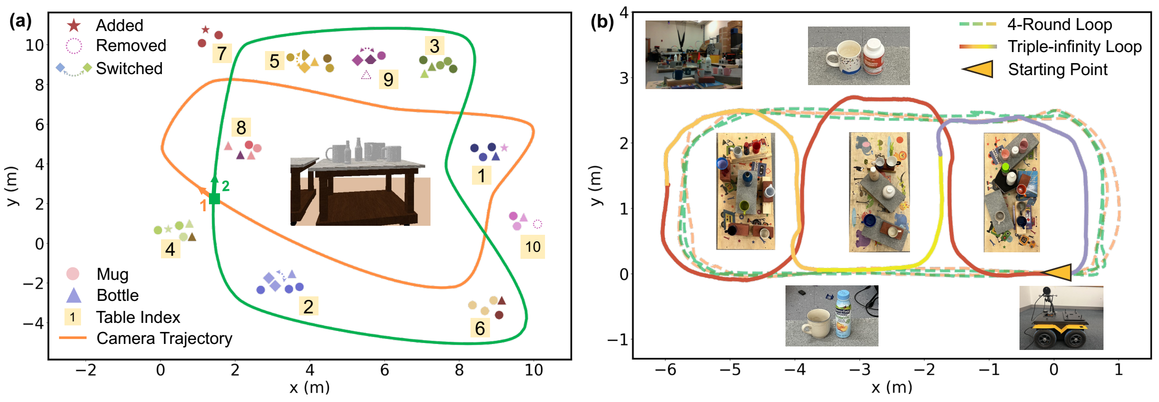


Figure 6-6: Evaluation data overview. Object changes happen at each joint of the colored trajectory segments. (a) Table layout with object changes and the ground truth camera trajectory of the two synthetic sequences. (b) Real-world setup with ground truth camera trajectories.

two sequences with *uninterrupted* object observation among the ten tables, where objects are revisited on most tables (excluding table 4, 7, and 10) from approximately opposite views. For each sequence, objects are added, removed, or moved to different locations, resulting in a total of nine changes within the trajectory.

Real-world Sequences. 28 mugs and bottles of various shapes and sizes are densely located on five tables in a $6{\times}3\ m^2$ space (Fig. 6-6(b)), among which ten objects are added, removed, or switched locations to create two sets of object arrangements. A RealSense D515 camera mounted on a Clearpath Jackal robot records RGB-D data along two preset trajectories: (1) A four-round peripheral loop around three central tables, with the first two rounds captured with one object arrangement and the latter two with the other arrangement, in total having nine changed objects. (2) A more challenging triple-infinity loop where the camera moves through four central and side tables, with seven object changes along the way. The ground truth camera trajectories are recovered from a Vicon motion capture system. The object segmentation masks are obtained from Detectron2 [134].

6.5.2 Experimental Details

To train NeuSE's occupancy network, we generate two sets of training samples using 94 mug models and 242 cylindrical bottle models from ShapeNet. The sets are, respectively, unambiguous (mugs) and ambiguous (bottles) objects, each containing 60,000 RGB-D partial observations with segmentation masks. We follow the sample generation strategy in Section 6.3.4: 2,000 object layout mixing bottles and mugs are created in Pybullet, from each of which 30 views are uniformly sampled with $[d_n, d_f] = [0.3, 5]$ (m) and $[d_l, d_h] = [-0.2, 0.2]$ (m). We train our approach on two NVIDIA RTX 3090 GPUs using a learning rate of 5×10^{-4} with the Adam optimizer. The latent code size is $k = 512$, and the occupancy threshold for reconstruction is $v_0 = 0.5$. We set the weight coefficients in Eq. (6.10) to be $(\beta_1, \beta_2) = (0.1, 0.1)$ for unambiguous objects, and $(\beta_1, \beta_2) = (1, 0.1)$ for ambiguous objects. The training batch is populated with eight object shapes, each with 15 partial observations, by setting $M = 15$ and $N = 8$.

For the object SLAM system, we have $\delta_{shape} = 0.95$, $(\delta_{prox}, \delta_e) = (0.03, 0.02)$ (m) for the synthetic sequence, and $(\delta_{prox}, \delta_e) = (0.04, 0.03)$ (m) for real-world sequences for data association and change detection. We set the sliding window size as $K = 10$ and adopt the factor graph representation for SLAM pose graph optimization. The local sliding-window optimization is solved with a Levenberg–Marquardt fixed-lag smoother, and the global pose graph is solved with iSAM2 [56], both using implementations from GTSAM [18].

Model Details. We provide the network architecture of our encoder and decoder in Table 6.1 and Table 6.2, respectively, which are adopted from Neural Descriptor Fields [111] using the VNNLinear, VNNResnetBlock, VNLeakyReLU blocks introduced in Vector Neurons [19]. Here, z_{dim} refers to the size of the NeuSE latent code, $\mathbf{z} \in \mathbb{R}^{z_{dim} \times 3}$.

Training Details. During training, we randomly draw 500 points from the observed partial point clouds for each sample to be fed into the encoder network. For training with L_{occ}, the query point size is 750, consisting of half object points and half off-the-object points from the given model.

We set the dimension of the latent code $\mathbf{z}$ to be $z_{dim} = 512$ and the weight coefficients $(\beta_1, \beta_2) = (0.1, 0.1)$ for unambiguous objects and $(\beta_1, \beta_2) = (1, 0.1)$ for ambiguous objects so as to balance the order of magnitude difference among L_{occ},

$L_{transform}$, and L_{shape} (with L_{occ} as the reference). In terms of our choice of the latent code dimension, we further find that, as opposed to training the three losses jointly using a single 512-dimensional latent code, we may also enforce SE(3)-equivariance and cross-viewing-angle shape consistency separately on two lower dimensional latent codes/networks. Here, when applying $L_{occ} + L_{transform}$ and $L_{occ} + L_{shape}$ individually on two network models, each with a latent size of 128, we obtain almost on-par transform and shape characterization power from the combination of two 128-dimensional latent codes compared to that of the vanilla 512-dimensional code. Hence, this can serve as a memory-efficient alternative to our original training approach for more lightweight training and memory-critical application scenarios.

Reconstruction and Update of the Object-centric Map. We here elaborate on our choices and procedures in building and maintaining the object-centric map, which we adopt to deal with noisy real-world data.

Our proposed approach depends on correct object segmentation masks to produce effective latent codes for objects. To avoid potential failures from false object latent codes, considering the uncertainty of off-the-shelf object detectors and depth cameras, we only initialize a new object instance if it has been recognized robustly by the depth camera and the object detector, e.g., an object close enough to the camera with an abundant number of points in the observed point cloud. Here in our experiment for object instantiation, we only consider objects that are within 2 m away from the depth camera and with their pixel-level segmentation mask size above 4,000. Ultimately, a new object is instantiated after it has been regarded as a "new" object three times in a row during data association.

For map maintenance and update, we conduct the removal of "residual" object instances based on the bounding box of each full object reconstruction. This ensures less redundant object instantiation from partial observations and no overlapping object reconstructions for detected changed objects whose new positions were previously occupied. Thanks to NeuSE, we can always get a reasonable shape prediction out of partial observations, e.g., an upright bottle with its bottom half obscured. We are, therefore, able to obtain the 3D bounding box of each observed object and conduct

VNLinear(128,256)
VNLinear(256,128)
VNLinear(256,128)
VNLinear(256,128)
VNLinear(256,128)
VNLinear(256,128)
Meanpool
VNLinear(128,z_{dim})
$\mathbf{z} \leftarrow$ Encode

Table 6.1: Encoder architecture.

VNLinear(z_{dim},z_{dim})
Linear(2*z_{dim}+1,128)
ResnetBlockFC(128)
ResnetBlockFC(128)
ResnetBlockFC(128)
ResnetBlockFC(128)
ResnetBlockFC(128)
Linear(128,1)
Sigmoid

Table 6.2: Decoder architecture.

the object removal procedure as follows: (1) If a small object's bounding box has a high overlap with a large one (0.95 in our case), this small object is deemed as a partial instance belonging to the large one and will be removed. (2) A changed object's bounding box intersects (we set it as 20% of its bounding box volume) with an older object, meaning the older object should no longer be in its original place. We hence remove the older object from the map and update the changed object to its new position. In this way, we are able to maintain a consistent object map while avoiding overlapping reconstructions such as the artifacts shown in Fig. 6-12(b) and (c).

6.5.3 Localization with Temporal Scene Inconsistency

All results are obtained on a laptop with an Intel Core i7-9750H CPU and an Nvidia GeForce RTX 2070 GPU. NeuSE network inference takes 6 ms per object, with inter-frame pose constraint calculation taking 1 ms. One-time rendering for object-centric map construction costs 30 ms per object with 20,000 query points. With data association included, the speed is approximately 28 FPS for generating object-level inter-frame pose constraints with our NeuSE-based front-end, making it promising for NeuSE to be integrated as an external "constraint sensor" with real-time operating speed. The final overall localization speed of our change-aware SLAM system is

Table 6.3: RMSE of ATE and Translational RPE on synthetic sequences. Gains (Δ) are computed based on results from Mug-only. The best results are marked in bold.

	Planar			**Non-planar**		
	Mug-only	All-object	Δ (%)	Mug-only	All-object	Δ (%)
RMSE of ATE (m)						
Synthetic: 1st traversal	0.072	**0.043**	**40.3%**	0.058	**0.045**	**22.4%**
Synthetic: 2nd traversal	0.096	**0.071**	**26.0%**	0.077	**0.033**	**57.1%**
Synthetic: Full	0.116	**0.065**	**44.0%**	0.091	**0.053**	**41.8%**
RMSE of Trans RPE (m/f)						
Synthetic: Full	0.026	**0.017**	**34.6%**	0.024	**0.016**	**33.3%**

11 FPS for the current experiment setting, with no software optimization or major tuning of the back-end iSAM2 solver. All following localization results are reported as the median of five runs.

Synthetic Sequences. The consecutive observations of objects in the synthetic data allow for uninterrupted operation of the proposed SLAM strategy, enabling an independent evaluation of NeuSE's capabilities for conducting change-aware localization and mapping.

Therefore, we report quantitatively in Table 6.3 the Root Mean Squared Error (RMSE) of both the translational Relative Pose Error (RPE) and the Absolute Trajectory Error (ATE) of the estimated camera poses for the two testing sequences, showcasing consistent NeuSE's performance both locally and globally. We further visualize the RPE and ATE error distribution along the way in Fig. 6-8 and Fig. 6-7, respectively.

To justify our treatment of the inclusion of ambiguous objects, we run two variants of the system as (1) Localizing with mugs only (Mug-only) and (2) Localizing with all objects of interest, i.e., mugs and bottles (All-object). For the few frames with no objects for data association or pose generation, we maintain system operation with odometry measurements corrupted from ground truth by a zero-mean Gaussian noise with $\sigma = 0.003$ rad for rotation and $\sigma = 0.05$ m for translation.

The RPE and ATE values in Table 6.3 show that (1) NeuSE is a reliable "constraint sensor" for producing consistent short- and long-range camera pose constraints, and (2) our system is capable of producing a globally consistent trajectory, despite various occlusion patterns, viewing angle disparities, and object changes between the two traversals. The smooth distribution of RPE throughout the sequence, as shown in Fig. 6-8, also demonstrates the robustness of our localization strategy against temporal scene changes, which is attributed to the effectiveness of our proposed data association and change detection in distinguishing objects in the second traversal.

Specifically, we observe from Table 6.3 that the proposed object SLAM approach performs better on the non-planar object layout, fully showing the efficacy of our SE(3)-equivariant representations in handling randomly oriented objects. This can be attributed to our training data generation strategy, which includes various views and occlusion patterns to learn robust geometric features of object shapes across viewing angles. Further, the lying-down mugs in the sequence help reduce shape ambiguity by providing more valid observations for generating camera pose constraints, as their handles are more frequently visible when pointing upwards than in the usual sideways direction. With the SE(3)-equivariant property of NeuSE, our approach can learn from upright observations to benefit the processing of laid down objects, thus enabling generalization to new scenarios with various object orientations.

Our attempt for the incorporation of ambiguous objects for pose constraint generation is validated by (1) the consistent improvement of All-object over Mug-only throughout the two traversals in Table 6.3, and (2) the lower dispersion of RPE values for All-object in Fig. 6-8. Besides, in Fig. 6-7(c)-(d), with object point clouds in (c) transformed from the upper (orange) to the lower (green) frame using transforms derived from only the pink mug and together with green bottles, the better point cloud alignment in (d) of All-object over Mug-only demonstrates the viability of leveraging covisible ambiguous objects for improving transform estimation accuracy.

Real-world Sequences. It is common for objects to be out of sight during real-world robot motion. Hence, in this section, we validate the feasibility and benefit of our strategy in complementing other SLAM measurements and promoting loop closing for a globally consistent estimated trajectory.

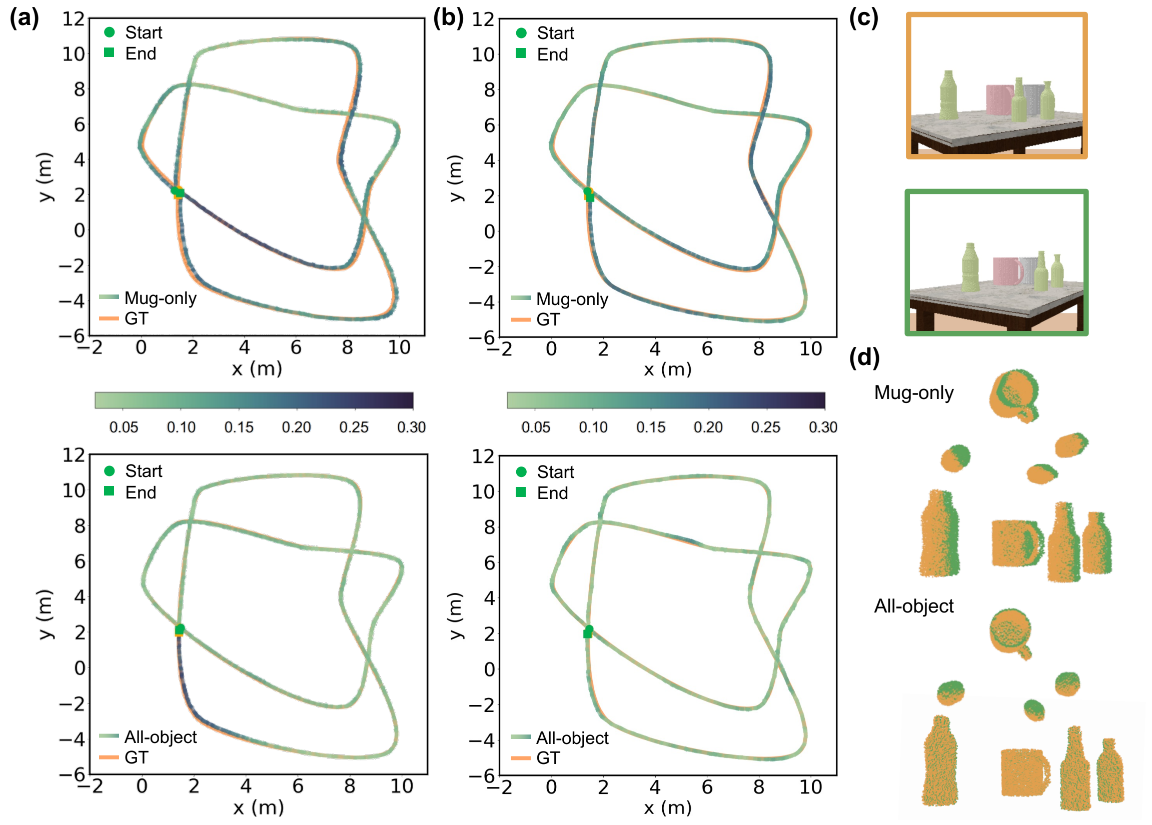


Figure 6-7: Column 1-2: Comparison of estimated and ground truth trajectories (GT) on synthetic sequences. (a) Planar and (b) Non-planar object layout. Color variation implies ATE value distribution along the path. All-object leads to better estimation accuracy than Mug-only, as shown by the evenly lighter trajectory color with lower ATE values. Column 3: Ambiguous objects for inter-frame transform prediction. With object point clouds in (c) transformed from the orange frame to the green frame using transforms derived from merely pink mugs and together with green bottles, the better point cloud alignment in (d) of All-object over Mug-only demonstrates the effectiveness of using covisible ambiguous objects to improve transform prediction accuracy.

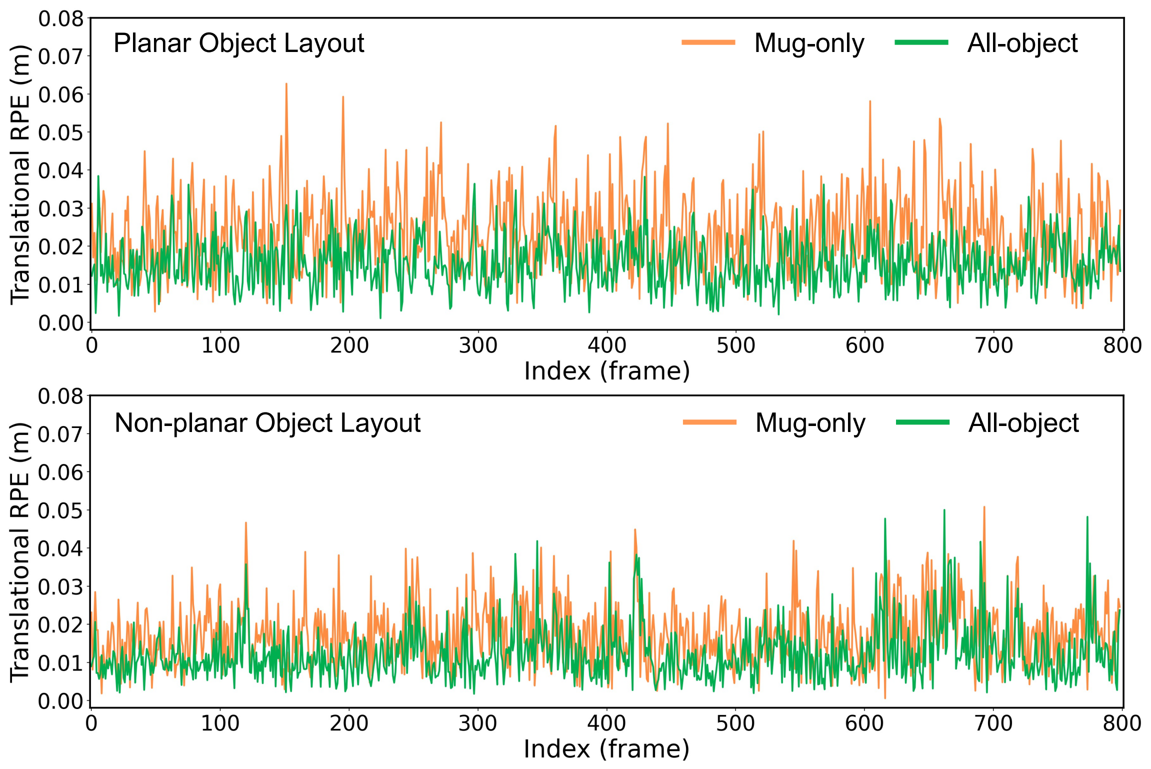


Figure 6-8: Distribution of translational RPE along synthetic sequences. The green lines in both layouts reveal lower RPE dispersion, indicating consistently lower local drift when using all objects of interest (mugs and bottles), as opposed to using only unambiguous ones (mugs).

Table 6.4: RMSE (m) of ATE on real-world sequences. The best results for each trajectory are marked in bold.

	CubeSLAM [138]	**EM-Fusion** [114]	**Obj-only**			**ORB3-NS**		**ORB3-PW**	
	All Objects Detected	All Objects Detected	Raw Odometry	Mug-only	All-object	Base	+ Ours	Base	+ Ours
4-Round: 1st – 2nd round	0.108	0.162	1.22	0.122	0.112	0.101	0.096	0.102	**0.084**
4-Round: 2nd – 3rd round	0.114	0.174	1.85	0.124	0.114	0.126	0.090	0.102	**0.083**
4-Round: 3rd – 4th round	0.128	0.127	2.07	0.123	0.090	0.119	0.085	0.086	**0.076**
4-Round: Full	0.131	0.154	3.51	0.134	0.111	0.118	0.092	0.093	**0.079**
Triple-infinity	0.147	0.193	1.12	0.137	0.106	0.101	**0.082**	0.160	0.083

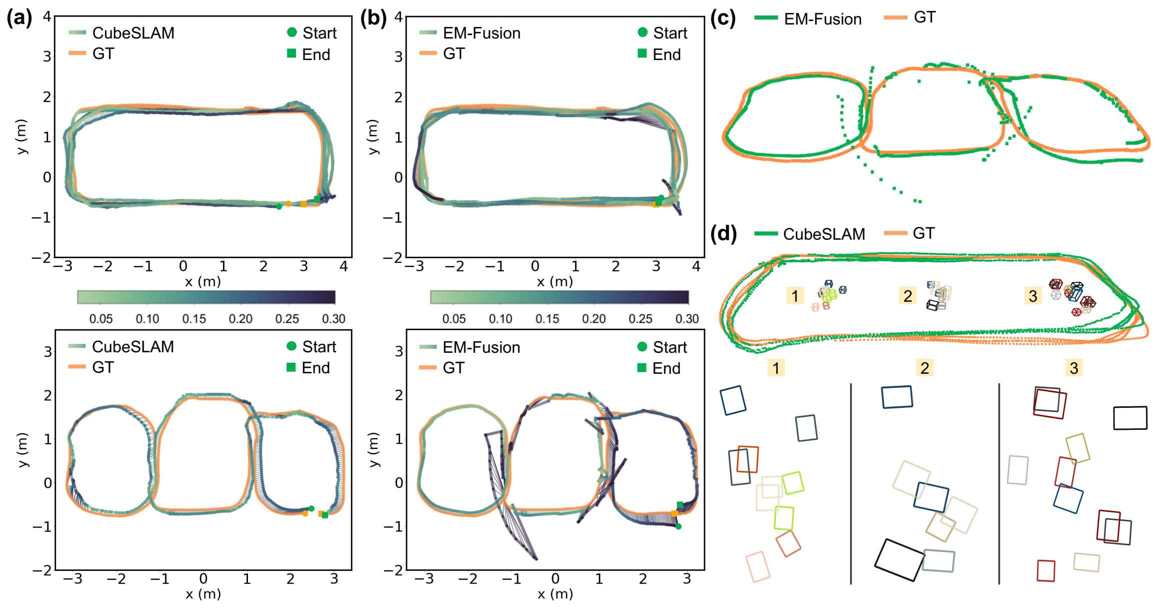
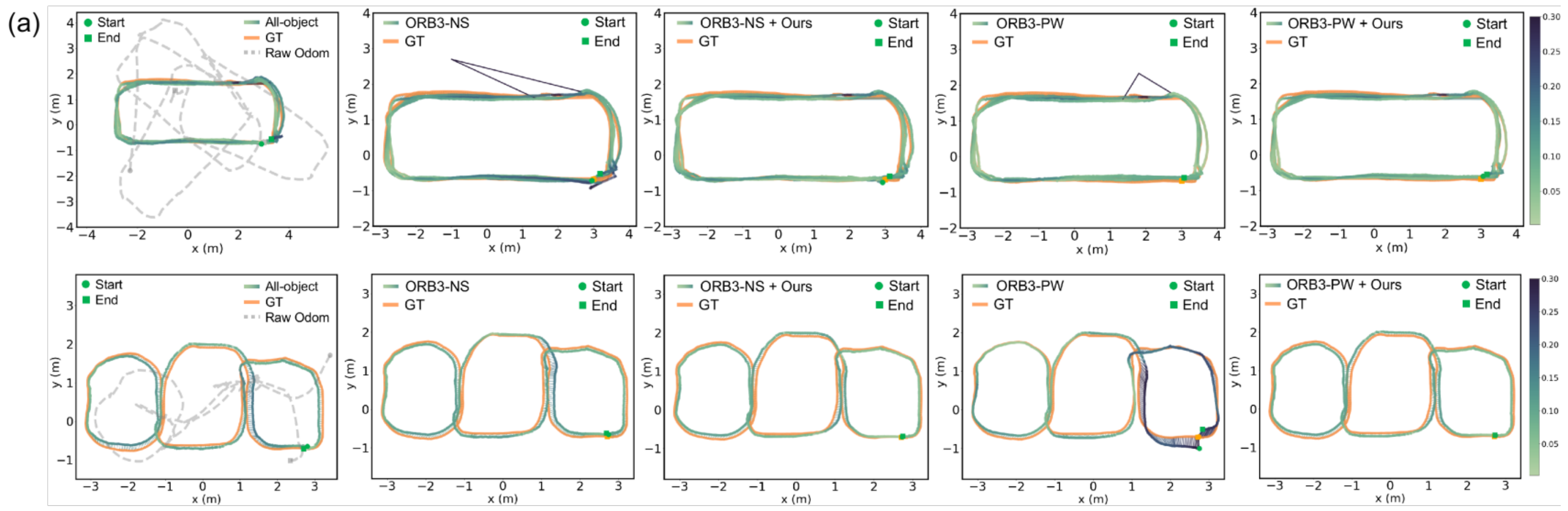


Figure 6-9: Column 1-2: Visualization of the estimated trajectories: (a) CubeSLAM and (b) EM-Fusion. Color variation of the line indicates ATE value distribution along the trajectory. Column 3: Trajectory/object cuboid estimation drifts of the two selected object SLAM baselines. (c): EM-Fusion undergoes heavy out-of-plane drift in the Triple-infinity loop due to faster rotations around the corners. (d): The top-down view (bottom row) displays the cuboid estimates of mugs and bottles in the 4-Round loop. CubeSLAM struggles to handle object changes, which causes inaccuracies in data association. This results in multiple missed, drifted, and falsely overlapped cuboid detections and affects the joint optimization of cuboid estimates and camera trajectory.



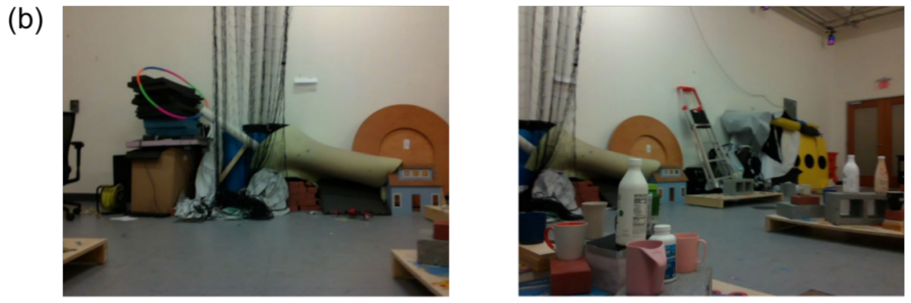


Figure 6-10: Visualization of estimated trajectories against ground truth (GT). Color variation (color bar on the right) of the line indicates ATE value distribution along the trajectory. (a) Above: Estimated trajectories of the 4-Round loop. The integration of our strategy (column 3 and 5) helps prevent the tracking failure, as shown by the two spikes in the second and fourth column. Below: Estimated trajectories of the Triple-infinity loop. Our strategy (column 5) successfully eliminates the start and end point drift for ORB3-PW (column 4), resulting in improved trajectory estimate when revisiting the rightmost table, as indicated by the lighter color of ATE values along the trajectory. (b) Vast viewing angle variance from auto-exposure malfunction between two frames leads to tracking failure in ORB-SLAM3.

In this spirit, we adopt ATE as the metric and compare our approach to two directly deployable object-based SLAM strategies, CubeSLAM [138] and EM-Fusion [114], as well as the popular and state-of-the-art ORB-SLAM3 [11] pipeline. CubeSLAM assumes a static operating environment (or objects with known motion models, which is not applicable here), and EM-Fusion can handle moving objects in the scene. They serve as baselines to evaluate object SLAM performance and the potential influence of object changes in the scene. For CubeSLAM, the implementation of its integration with ORB-SLAM is chosen. As ORB-SLAM3 does not address temporal scene inconsistency, to explore the effect of object changes on localization performance, we generate two sets of ORB-SLAM3 odometry measurements as baselines by running it (1) non-stop (ORB3-NS) for the whole trajectory, and (2) piecewise (ORB3-PW) for each trajectory segment with consistent object layout (as shown in Fig. 6-6(b)).

In addition, to verify NeuSE's transferability from simulation to reality, we follow the object-only experiments for synthetic data and run Mug-only and All-object on the two real-world sequences. Raw Odometry, generated using Open3D [144] based on photometric and geometric loss [94], is adopted to sustain system operation when no objects are in sight or associated to generate a pose constraint.

We present in Table 6.4 the RMSE of ATE for all estimated trajectories and visualize them in Fig. 6-9 and Fig. 6-10. The tracking failures of the 4-Round loop (the two spikes in the first row of Fig. 6-10) are excluded from RMSE calculation to better reflect the global localization performance of the trajectory.

The transferability of NeuSE from simulation to the real world is verified by its fair performance in terms of RMSE values and remarkable correction of the accumulated drift from Raw Odometry, as seen in the first column of Fig. 6-10. This confirms NeuSE's full functionality when applied to real data, which shows our design advantage of learning geometric relationships from point clouds in simulations. In this context, the acquired object geometry smoothly translates to the real world without being affected by the common but undesirable differences in lighting and appearance between simulations and reality.

This is explained by learning geometric relations from point clouds in simulation, where the learned object geometry applies smoothly to the real world and is not affected by the undesirable yet common photometric differences between simulation and the real world.

In comparison to the two selected object SLAM baselines that use all detected objects in the scene, from Table 6.4 and Fig. 6-9, our proposed approach outperforms CubeSLAM and EM-Fusion on both the 4-Round and Triple-infinity loop with using all objects of our interest (mugs + bottles), showing the advantage of NeuSE for facilitating lightweight and robust localization in real-world sequences with scene inconsistency.

Notably, CubeSLAM produces shrinking camera trajectory estimates in Fig. 6-9(a) around the right table with object changes in the Triple-infinity loop, and multiple missed, drifted, and falsely overlapped cuboid estimates from the top-down view (bottom row) in Fig. 6-9(d). Assuming a static environment, CubeSLAM struggles to address object changes within the two sequences, inducing errors in cuboid association and estimation among old and new objects in neighboring areas. This yields false camera-cuboid geometric constraints and ultimately affects the joint optimization of object cuboids and camera trajectory.

Meanwhile, EM-Fusion, as shown in Fig. 6-9(b) and (c), gives subpar bumpy and drifted trajectory estimates. While it can handle scene layout changes at sequence segment intersections, EM-Fusion suffers from lower tracking accuracy due to accumulated drift from less object overlap. Besides, originally tested on tabletop scenes, EM-Fusion requires a coarser SDF background volume resolution so as to avoid memory exhaustion here in our larger multi-table scenario, leading to a further loss of accuracy in camera tracking.

As to working jointly with other SLAM measurements, in Table 6.4, we observe consistent improvement in terms of RMSE values when integrating our proposed strategy (using all objects) with the vanilla ORB-SLAM3 measurements. NeuSE enables robust data association and manages to prevent the occurrence of tracking failure (the spikes in the second and fourth column of Fig. 6-10) for the 4-Round trajectory.

Table 6.5: Change detection results on the synthetic and real-world sequences. The best results are marked in bold.

		TP	FP	FN	Pr	Re
Synthetic						
	PMT	7	2	2	77.8%	77.8%
	Ours	9	0	0	**100.0%**	**100.0%**
4-Round						
	PMT	7	0	2	**100.0%**	77.8%
	Ours	9	0	0	**100.0%**	**100.0%**
Triple-Infinity						
	PMT	5	2	2	71.4%	71.4%
	Ours	7	1	0	**87.5%**	**100.0%**

The greatest RMSE improvement in Table 6.4 is observed from ORB3-PW + Ours on the Triple-infinity trajectory. Our proposed strategy helps decrease the RMSE by 48.1% from 0.16 m to 0.083 m. In this way, ORB3-PW + Ours outperforms ORB3-NS (0.101 m) despite receiving less global loop closing constraints from ORB3-PW than ORB3-NS, while aligning the start and end point with better trajectory accuracy when revisiting the rightmost table. Considering the little scene overlap within each of the four trajectory segments, this notable improvement highlights the critical role of our strategy in constraining pose estimates in short and longer ranges, especially when insufficient loop closing (e.g., throughout ORB3-PW) is performed by the external SLAM system.

Our strategy also demonstrates robustness in handling scene changes, despite the less significant improvement in the 4-Round loop that is with abundant loop closure from ORB-SLAM3. The fourth column of Table 6.4 presents the RMSE values of ORB3-NS on different parts of the 4-Round loop, as the sequence proceeds with the object layout transition. Note that ORB3-PW does not run between the second and third round, with the corresponding value listed only for comparison purposes. When object changes happen at the intersection of the second and third round, ORB3-NS is clearly affected and shows an RMSE jump from 0.101 m to 0.126 m. On the contrary, our effective data association based on full object shape

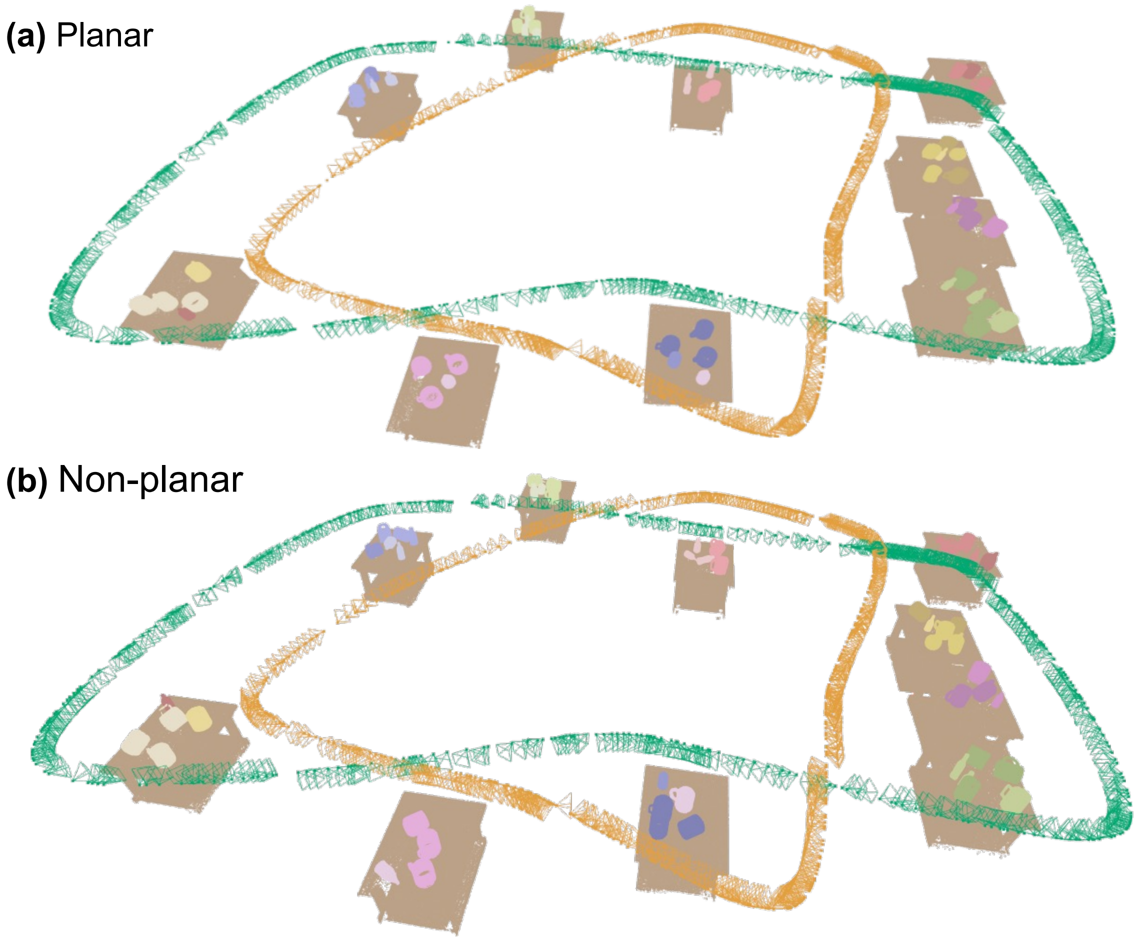


Figure 6-11: Complete object reconstruction of synthetic sequences for the two object layouts. (a): Planar layout and (b): Non-planar layout. Tables are rendered for visual clarity, whose points are back-projected to the world using camera pose estimates from NeuSE-predicted constraints, demonstrating the effectiveness of our localization strategy.

similarity and spatial proximity allows ORB3-NS + Ours to maintain a steady yet gradually improving RMSE (around 0.09 m) during object changes, bringing ORB3-NS almost on-par performance with ORB3-PW (free from object changes) for the entire trajectory.

6.5.4 Change-aware Object-centric Mapping

Built on top of the decoding steps in Eq. (6.4) for full object reconstruction, we demonstrate the ability to maintain a consistent map of objects of interest in the environment, with always timely updates of the latest changes.

Since there are no suitable SLAM pipelines for direct comparison of mapping with temporal scene changes, we use the recent object-level mapping method with online change detection, panoptic multi-TSDFs (PMT) by Schmid et al. [106], as our baseline. We feed PMT with our trajectory estimates that have the lowest RMSE of ATE values and compare the change detection results for synthetic and real-world sequences.

We quantify the performance of our system and PMT in Table 6.5 by comparing the number of correctly detected changes (true positives, TP), falsely detected changes (false positives, FP), and undetected changes (false negatives, FN). We further calculate precision (Pr) and recall (Re) rates based on these numbers. The results show that our system correctly detects most of the changes for both synthetic and real-world data, while PMT produces several false positives and false negatives due to localization errors and the inability to reason holistically from partial observations.

Qualitatively, we present in Fig. 6-11 reconstructions of all objects that have appeared in the synthetic planar and non-planar layouts, respectively. Fig. 6-12 displays the map evolution of our method and PMT before and after changes for each table in the real-world sequences. Our approach generates a lightweight, object-centric map that precisely captures changes (see Fig. 6-12(d)). In contrast, PMT, being a traditional TSDF-based mapping technique, fails to deliver accurate change detection results and produces reconstructions with various defects. PMT struggles to distinguish between switched objects of the same category due to its inability to perform full object shape comparison as NeuSE does. This is shown by the overlapping reconstructions of the white and green bottles (object 2 and 3 of table 3) and the red and black mugs (object 8 and 9 of table 5) in Fig. 6-12(b). In addition, Fig. 6-12(c) highlights PMT's susceptibility to localization errors, where it mistakenly marks the green mug on Table 5 as newly added when the other side of the mug, which is observed later, drifts to be misaligned with the original volume.

6.6 Conclusion

In this chapter, we present NeuSE, a category-level neural SE(3)-equivariant embedding for objects, and demonstrate how it supports object SLAM for consistent spatial understanding with long-term scene inconsistency. NeuSE differs itself from prior neural representations adopted in SLAM through its ability to *directly* obtain camera pose constraints from SE(3)-equivariance and its flexible map representation that easily accommodates long-term scene changes. Our evaluation results on both synthetic and real-world data showcase the feasibility of our approach for change-aware localization and mapping when working standalone or as a complement to traditional SLAM pipelines.

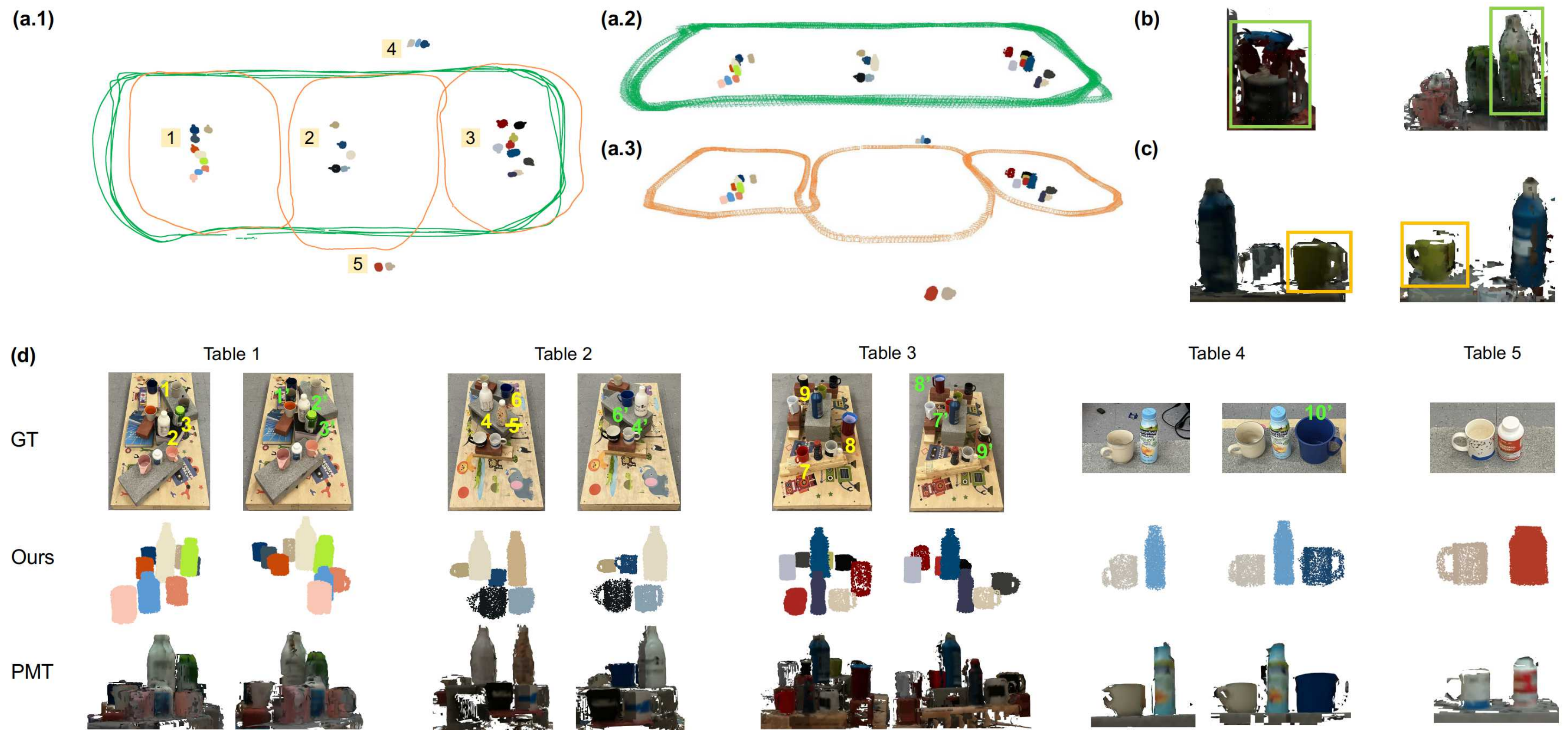


Figure 6-12: Results of change-aware mapping for the real-world sequences. (a) Comparison of our object-centric map to ground truth trajectories, displaying qualitative spatial consistency. (d) shows the evolution of the reconstructed object layout before and after changes, with ground truth scenes (GT), object-centric maps from our approach (Ours), and PMT reconstructions (PMT) from top to bottom. Changed objects are numbered as n, with n' representing their correspondence after changes or newly added objects, and ~~n~~ indicating objects removed from the scene. (b) Reconstruction artifacts of overlapping bottles (left) and mugs (right) from PMT's change detection failure. (c) False positive changed mug marked by PMT due to imperfect localization, where little overlap exists between the two sides of the green mug when viewed from different frames.

Chapter 7

Conclusion

7.1 Contributions

In this thesis, we dive into the problems of realizing object-based SLAM, with the ultimate goal of robust spatial reasoning in a long-term and low-dynamic environment. Evolving around the keywords of *object* and *long-term+low-dynamic*, we progressively explore three topics: (1) Object pose ambiguity for smooth integration into the SLAM pipeline; (2) Scalable scene and object representations for change detection; And (3) 3D-consistent neural implicit object embeddings for robust spatial understanding against temporal inconsistency. In this vein, we summarize our contributions as follows:

- **On ambiguity-aware integration of object poses into the SLAM pipeline**, in Chapter 3, we propose a multi-hypothesis approach for the smooth adoption of object poses in object-based SLAM. This approach accommodates the inherent ambiguity arising from occlusion or symmetrical object shapes. We design a multi-hypothesis object pose estimator front end in a mixture-of-expert fashion and a max-mixture-based back end to infer a globally consistent set of camera and object poses from a sequence of pose hypothesis sets.

- **On robust change detection with scalable and 3D-consistent representations**, in Chapter 4 and 5, we develop two change detection methodologies, each for online and offline applications, with two novel scene and object representations, PlaneSDF and shape-consistent neural descriptor fields. Aiming for long-term operation, we account for (1) inevitable scene changes over extended time spans and (2) the efficiency and scalability of the chosen map representations. On top of the proposed representations, we explore cluster- and object-level change detection, following a "divide-and-conquer" strategy to enable more accurate and flexible change detection through local scene differencing.

- **On long-term spatial reasoning against temporal scene inconsistency**, in Chapter 6, we propose a neural SE(3)-equivariant object embedding (NeuSE) and demonstrate its use in object-based SLAM for long-term consistent spatial understanding. Considering the earlier ambiguity and scene inconsistency concerns, NeuSE are designed to serve as a compact point cloud surrogate for complete object models. Our NeuSE-based object SLAM paradigm directly derives SE(3) camera pose constraints compatible with general SLAM pose graph optimization. As a result, it can conduct object-assisted localization and maintain a lightweight object-centric map with change-aware mapping capability, ultimately achieving robust scene understanding in the face of low-dynamic scene changes.

In sum, this thesis demonstrates the potential of harnessing object-level information to support a SLAM system that exhibits comparable performance in contrast to its feature-based or direct SLAM counterparts. At the same time, our approach introduces a heightened level of adaptability and scalability, adeptly addressing the complexities inherent in long-term operations and low-dynamic scene changes. We believe that object-based SLAM pipelines, as opposed to traditional SLAM approaches, adhere more naturally to the concept of objects as elemental units of the world. As a result, our paradigm seamlessly navigates object changes with greater ease and effectively aligns with the demands of today's high-level robotic tasks.

7.2 Lessons Learned

In this section, we reflect on the insights gained from our journey in developing a line of work aimed at building an *object-based* SLAM system, or more broadly, an *object-assisted* SLAM system by employing object information into common SLAM pipelines.

The Power of Object Integration. Our exploration has demonstrated the remarkable advantages of incorporating objects into the SLAM pipeline, particularly in low-dynamic environments. Representing the world in terms of objects closely mirrors human-like reasoning. By introducing the concept of "objects" into the SLAM workflow, we inherently enhance its compatibility with advanced robotic tasks that rely on the notion of "objectness". We showcase our design to employ object-based SLAM in intriguing low-dynamic scenarios, defining low-dynamics as object changes occurring over time without direct camera observation. Such scenarios are inevitable in long-term operations, especially when object-level interactions come into play. Without appropriate measures, these changes can lead to false feature correspondence matches and erroneous map reconstructions, potentially misguiding robotic tasks. In this spirit, we argue that object-based SLAM pipelines bear superiority over their traditional counterparts, as their object-level interpretation naturally accommodates the assimilation and update of static and changing objects. This results in more robust localization and change-aware object-centric mapping, particularly for objects of interest in the current task.

Metric Selection for Evaluation. We scrutinize the metrics chosen for evaluating trajectory and object pose estimates. From all the metrics adopted in the thesis, we distinguish between two categories: correspondence-based and nearest-neighbor-based. Metrics requiring strict correspondences, such as average distance (ADD) and all metrics for trajectory evaluation, excel in reflecting the absolute accuracy of the estimation results. Conversely, metrics computing nearest-neighbor point distances, such as ADD-S and Chamfer distance, prove helpful in representing and capturing the multi-modality inherent in the data. This becomes particularly significant when addressing ambiguity arising from object shapes and poses, as demonstrated in Chapter 3 and Chapter 6.

Transferability with Learning from Simulation. In Chapter 5 and 6, as we aim to develop efficient and effective object representations, we argue that our choice to exclusively utilize simulated point clouds helps us overcome the challenges posed by the Sim2Real gap when learning from simulations. While addressing the limitations of real-world training data, particularly in terms of achieving diverse object shapes, the simulation environment provides us with a unique advantage. Here, we can generate as much data as needed, ensuring that the knowledge gained about object geometry effortlessly applies to the real world. This adaptation occurs without being affected by the common, yet undesirable, differences in lighting and visual characteristics between simulations and reality. As a result, our approach not only improves the transferability of knowledge from simulation to reality but also demonstrates remarkable versatility, both within object categories of varying shapes and even across categories that share common or similar shapes.

Scalability through Modular Operation. Our solution to long-term and larger-scale operation centers on decomposing global and scene-wise operations into local and object- or cluster-wise operations. This approach enhances memory efficiency and reduces susceptibility to global absolute errors stemming from localization or alignment using a single registration transform. This concept is manifest in our proposals of NeuSE and PlaneSDF for object/scene representation (Chapter 4-6), as well as our two change detection schemes. Furthermore, this modular operation framework facilitates the continuous update of environmental reconstructions, adding scalability and flexibility to our system operation.

Bridging Frontend and Backend Divide. Lastly, we revisit the boundary between common SLAM concepts of "front end" and "back end". Traditionally, the front end handles sensor data processing, while the back end processes constraints generated in the front end to estimate camera and landmark poses. The division is often shaped by the distinct tools respectively employed for sensor data processing and optimization rather than being an inherent and irremovable distinction. This division corresponds to the two ways of leveraging object information: the coupled and decoupled fashion. Current SLAM systems using neural implicit representations predominantly follow the coupled approach with gradient descent optimization running on optimizable object codes and camera poses. However, we contend that going in the decoupled way would not only allow for still accurate object-derived inter-frame camera pose constraints but also greater flexibility in integrating and benefiting from the power of off-the-shelf SLAM systems. Moreover, we accomplish this by further enforcing SE(3)-equivariance onto the adopted object representation.

7.3 Limitations

Performance Gap with Traditional Methods. Drawing from prior research and our experimental findings, it is evident that current object-based SLAM systems still exhibit a certain performance gap when compared to traditional SLAM pipelines. This discrepancy can be attributed to the nature of obtaining odometry or loop-closure constraints through learning-based methods. The performance of these methods can be influenced by factors such as motion blur, sensor noise (e.g., depth measurements and camera auto-exposure settings), or variations in viewing angles that lie outside the training data distribution. Despite our dedicated efforts to emphasize learning from geometric point clouds rather than RGB data (Chapter 5 and 6) and aiming for smooth knowledge transfer from synthetic data to the real world, there remains a challenge. The complex motion patterns encountered during actual mobile operations can prove challenging to fully incorporate into our training dataset.

Limited Number of Object Categories for Employment. This limitation stems from the finite training data available. We address the issue to some extent in Chapter 6 by introducing NeuSE as a category-level object representation. This approach allows us to handle shape variations within specific object categories. While one straightforward solution would be providing extensive training data, this often demands prior object knowledge and can become costly when continuously accommodating new object categories in the environment over time.

Growing Complexity with Long-term Operation. As we aim for robustness in the long term, the system records a growing number of objects, subsequently expanding the array of object pose hypotheses. This influx adds complexity to the backend optimization process. A potential solution involves quantifying the confidence for each object hypothesis. This enables the pruning of less confident hypotheses or object measurements, thereby reducing computational overhead. Additionally, a confidence score can be employed to assign varying weights to hypotheses, facilitating faster convergence.

7.4 Future Work

Reflecting on the limitations mentioned in the preceding section, we see the necessity of more system integration efforts on building an actual SLAM system for further detailed real-world experiments. As we chart our course towards achieving more refined, human-like object-level reasoning about the world, we list the following exciting directions for potential further exploration.

Better Generalization across Wider Range of Objects. As previously discussed, the constrained processing capacity when addressing a fixed array of object categories and diverse motion patterns can significantly impede the performance of learning-based object SLAM systems. To support the generalizability of the employed object representations, two potential approaches stand out. The first is adopting a category-agnostic fashion, involving compositional representations (as exemplified by [13]) that capture the common geometric structures of objects – such as rectangular handles or cylindrical bodies – rather than detailing each object's shape as a whole. The second avenue leverages the emerging visual foundation models trained on internet-scale data, like BERT [25], CLIP [97], and GPT [92]. A subsequent question would then be how to harness 3D reconstructed maps from SLAM to train for 3D-consistent representations that incorporate 2D features from foundation models.

Multi-sensor Fusion. Currently, most object-based SLAM systems rely heavily on RGB(-D) data, which impart a limitation on their robustness when confronted with diverse motion patterns. One solution emerges from utilizing different data modalities, including Inertial Measurement Units (IMUs) and RGB-D cameras, in a synergistic manner. This collaboration proves particularly advantageous when specific sensors, such as cameras during sharp turns, encounter limitations. Despite the challenge of defining a clear "object" concept for non-visual sensors, a compelling direction to explore involves developing efficient strategies to integrate information from these non-visual measurements with object-based data seamlessly. A fundamental query then arises: Should we opt to *decouple* non-visual measurements from their visual counterparts, mirroring the approach undertaken in Chapter 6? If so, what form – for instance, odometry measurements – and structure – like pose graphs – would best facilitate their holistic incorporation? Alternatively, the potential exists for novel neural network architectures that *couple* diverse measurement channels from varying sources, engendering comprehensive inter- and intra-modal learning.

Uncertainty Quantization. Quantifying the uncertainty inherent in object-derived information offers valuable advantages, facilitating calibrated processing and prioritization of measurements based on confidence levels. The representation of uncertainty can take diverse forms, ranging from covariance matrices – offering insights into uncertainty across each dimension – to weight coefficients that better inform the optimization back end, especially as the number of measurements increases. A pertinent extension of this endeavor involves distinguishing outlier measurements and mitigating potential negative effects stemming from the inadvertent inclusion of erroneous constraints during optimization. This challenge assumes critical importance in object-based SLAM pipelines, where the number of landmarks and, consequently, camera-landmark measurements are significantly lower than those in traditional SLAM methods. This heightened vulnerability thus requires robust measures to counter erroneous data association or correspondence matches.

Bibliography

[1] Michal Adamkiewicz, Timothy Chen, Adam Caccavale, Rachel Gardner, Preston Culbertson, Jeannette Bohg, and Mac Schwager. Vision-only robot navigation in a neural radiance world. In RA-L, 2022.

[2] Gabriel Agamennoni, Simone Fontana, Roland Siegwart, and Domenico Sorrenti. Point clouds registration with probabilistic data association. In Proceedings of The International Conference on Intelligent Robots and Systems (IROS), 2016.

[3] Pablo F Alcantarilla, Simon Stent, German Ros, Roberto Arroyo, and Riccardo Gherardi. Street-view change detection with deconvolutional networks. Autonomous Robots, 42(7):1301–1322, 2018.

[4] Rareş Ambruş, Nils Bore, John Folkesson, and Patric Jensfelt. Meta-rooms: Building and maintaining long term spatial models in a dynamic world. In 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1854–1861. IEEE, 2014.

[5] Paul J. Besl and Neil D. McKay. A method for registration of 3-D shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence, 14(2):239–256, February 1992. ISSN 0162-8828. doi: 10.1109/34.121791. URL http://dx.doi.org/10.1109/34.121791.

[6] Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, and Andrew J Davison. Codeslam—learning a compact, optimisable representation for dense visual slam. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2560–2568, 2018.

[7] Nils Bore, Johan Ekekrantz, Patric Jensfelt, and John Folkesson. Detection and tracking of general movable objects in large three-dimensional maps. IEEE Transactions on Robotics, 35(1):231–247, 2018.

[8] Erik Bylow, Jürgen Sturm, Christian Kerl, Fredrik Kahl, and Daniel Cremers. Real-time camera tracking and 3d reconstruction using signed distance functions. In Robotics: Science and Systems, volume 2, page 2, 2013.

[9] Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. The YCB object and model set: Towards common bench-

marks for manipulation research. In 2015 international conference on advanced robotics (ICAR), pages 510–517. IEEE, 2015.

[10] Berk Calli, Arjun Singh, James Bruce, Aaron Walsman, Kurt Konolige, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. Yale-cmu-berkeley dataset for robotic manipulation research. The International Journal of Robotics Research, 36(3):261–268, 2017.

[11] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM 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.

[12] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012, 2015.

[13] Ethan Chun, Yilun Du, Anthony Simeonov, Tomas Lozano-Perez, and Leslie Kaelbling. Local neural descriptor fields: Locally conditioned object representations for manipulation. arXiv preprint arXiv:2302.03573, 2023.

[14] Chi-Ming Chung, Yang-Che Tseng, Ya-Ching Hsu, Xiang-Qian Shi, Yun-Hung Hua, Jia-Fong Yeh, Wen-Chin Chen, Yi-Ting Chen, and Winston H Hsu. Orbeez-SLAM: A real-time monocular visual SLAM with ORB features and NeRF-realized mapping. arXiv preprint arXiv:2209.13274, 2022.

[15] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. GitHub repository, 2016.

[16] Jan Czarnowski, Tristan Laidlow, Ronald Clark, and Andrew J Davison. Deepfactors: Real-time probabilistic dense monocular slam. IEEE Robotics and Automation Letters, 5(2):721–728, 2020.

[17] 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 (ToG), 36(4): 1, 2017.

[18] Frank Dellaert. Factor graphs and gtsam: A hands-on introduction. Technical report, Georgia Institute of Technology, 2012.

[19] Congyue Deng, Or Litany, Yueqi Duan, Adrien Poulenard, Andrea Tagliasacchi, and Leonidas J. Guibas. Vector neurons: A general framework for so(3)-equivariant networks. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pages 12200–12209, 2021.

[20] Xinke Deng, Arsalan Mousavian, Yu Xiang, Fei Xia, Timothy Bretl, and Dieter Fox. Poserbpf: A rao-blackwellized particle filter for 6d object pose tracking. arXiv preprint arXiv:1905.09304, 2019.

[21] Yu Deng, Jiaolong Yang, and Xin Tong. Deformed implicit field: Modeling 3d shapes with learned dense correspondence. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 10286–10296, 2021.

[22] Erik Derner, Clara Gomez, Alejandra C Hernandez, Ramon Barber, and Robert Babuška. Towards life-long autonomy of mobile robots through feature-based change detection. In 2019 European Conference on Mobile Robots (ECMR), pages 1–6. IEEE, 2019.

[23] Erik Derner, Clara Gomez, Alejandra C Hernandez, Ramon Barber, and Robert Babuška. Change detection using weighted features for image-based localization. Robotics and Autonomous Systems, 135:103676, 2021.

[24] Erik Derner, Clara Gomez, Alejandra C. Hernandez, Ramon Barber, and Robert Babuška. Change detection using weighted features for image-based localization. Robotics and Autonomous Systems, 135:103676, 2021. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2020.103676. URL https://www.sciencedirect.com/science/article/pii/S0921889020305169.

[25] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.

[26] Kevin J Doherty, David P Baxter, Edward Schneeweiss, and John J Leonard. Probabilistic data association via mixture models for robust semantic SLAM. In 2020 IEEE International Conference on Robotics and Automation (ICRA), pages 1098–1104. IEEE, 2020.

[27] Jakob Engel, Thomas Schöps, and Daniel Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part II 13, pages 834–849. Springer, 2014.

[28] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. IEEE transactions on pattern analysis and machine intelligence, 40(3):611–625, 2017.

[29] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. IEEE Transactions on Pattern Analysis and Machine Intelligence, 40(3):611–625, 2018. doi: 10.1109/TPAMI.2017.2658577.

[30] Epic Games. Unreal engine. URL https://www.unrealengine.com.

[31] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In kdd, volume 96, pages 226–231, 1996.

[32] Marius Fehr, Fadri Furrer, Ivan Dryanovski, Jürgen Sturm, Igor Gilitschenski, Roland Siegwart, and Cesar Cadena. Tsdf-based change detection for consistent long-term dense reconstruction and dynamic object discovery. In 2017 IEEE International Conference on Robotics and automation (ICRA), pages 5237–5244. IEEE, 2017.

[33] Ross Finman, Thomas Whelan, Michael Kaess, and John J Leonard. Toward lifelong object segmentation from change detection in dense rgb-d maps. In 2013 European Conference on Mobile Robots, pages 178–185. IEEE, 2013.

[34] Jiahui Fu, Yilun Du, Kurran Singh, Joshua B. Tenenbaum, and John J. Leonard. Robust change detection based on neural descriptor fields. In 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 2817–2824, 2022.

[35] Ruohan Gao, Yen-Yu Chang, Shivani Mall, Li Fei-Fei, and Jiajun Wu. Objectfolder: A dataset of objects with implicit visual, auditory, and tactile representations. In CoRL, 2021.

[36] Andreas Geiger, Julius Ziegler, and Christoph Stiller. Stereoscan: Dense 3d reconstruction in real-time. In 2011 IEEE intelligent vehicles symposium (IV), pages 963–968. Ieee, 2011.

[37] Michael Grupp. evo: Python package for the evaluation of odometry and SLAM. https://github.com/MichaelGrupp/evo, 2017.

[38] Abner Guzman-Rivera, Dhruv Batra, and Pushmeet Kohli. Multiple choice learning: Learning to produce multiple structured outputs. In Advances in Neural Information Processing Systems, pages 1799–1807, 2012.

[39] Richard Hartley, Jochen Trumpf, Yuchao Dai, and Hongdong Li. Rotation averaging. International journal of computer vision, 103(3):267–305, 2013.

[40] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

[41] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In Proceedings of the IEEE international conference on computer vision, pages 2961–2969, 2017.

[42] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. The international journal of Robotics Research, 31(5): 647–663, 2012.

[43] Evan Herbst, Peter Henry, Xiaofeng Ren, and Dieter Fox. Toward object discovery and modeling via 3-d scene comparison. In 2011 IEEE International Conference on Robotics and Automation, pages 2623–2629, 2011. doi: 10.1109/ICRA.2011.5980542.

[44] Evan Herbst, Peter Henry, and Dieter Fox. Toward online 3-d object segmentation and mapping. In 2014 IEEE International Conference on Robotics and Automation (ICRA), pages 3193–3200. IEEE, 2014.

[45] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. arXiv preprint arXiv:1703.07737, 2017.

[46] 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 Asian conference on computer vision, pages 548–562. Springer, 2012.

[47] Timo Hinzmann, Thomas Stastny, Gianpaolo Conte, Patrick Doherty, Piotr Rudol, Marius Wzorek, Enric Galceran, Roland Siegwart, and Igor Gilitschenski. Collaborative 3D reconstruction using heterogeneous uavs: System and experiments. In International Symposium on Experimental Robotics, 2016.

[48] Berthold KP Horn. Closed-form solution of absolute orientation using unit quaternions. Josa a, 4(4):629–642, 1987.

[49] Mehdi Hosseinzadeh, Kejie Li, Yasir Latif, and Ian Reid. Real-time monocular object-model aware sparse SLAM. In 2019 International Conference on Robotics and Automation (ICRA), pages 7123–7129. IEEE, 2019.

[50] Ming Hsiao, Eric Westman, Guofeng Zhang, and Michael Kaess. Keyframe-based dense planar slam. In 2017 IEEE International Conference on Robotics and Automation (ICRA), pages 5110–5117. IEEE, 2017.

[51] Jiahui Huang, Shi-Sheng Huang, Haoxuan Song, and Shi-Min Hu. Di-fusion: Online implicit 3d reconstruction with deep priors. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 8932–8941, 2021.

[52] Dirk Hähnel and Wolfram Burgard. Probabilistic matching for 3D scan registration. In Proceedings of the VDI Conference, 2002.

[53] Jeffrey Ichnowski*, Yahav Avigal*, Justin Kerr, and Ken Goldberg. Dex-NeRF: Using a neural radiance field to grasp transparent objects. In CoRL, 2020.

[54] Zhenyu Jiang, Yifeng Zhu, Maxwell Svetlik, Kuan Fang, and Yuke Zhu. Synergies between affordance and geometry: 6-dof grasp detection via implicit representations. In RSS, 2021.

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

[56] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John Leonard, and Frank Dellaert. isam2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering. In 2011 IEEE International Conference on Robotics and Automation, pages 3281–3288, 2011. doi: 10.1109/ICRA.2011.5979641.

[57] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, and Frank Dellaert. isam2: Incremental smoothing and mapping using the bayes tree. The International Journal of Robotics Research, 31(2): 216–235, 2012.

[58] Ukyo Katsura, Kohei Matsumoto, Akihiro Kawamura, Tomohide Ishigami, Tsukasa Okada, and Ryo Kurazume. Spatial change detection using voxel classification by normal distributions transform. In 2019 International Conference on Robotics and Automation (ICRA), pages 2953–2959. IEEE, 2019.

[59] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[60] Justin Kerr, Letian Fu, Huang Huang, Yahav Avigal, Matthew Tancik, Jeffrey Ichnowski, Angjoo Kanazawa, and Ken Goldberg. Evo-nerf: Evolving nerf for sequential robot grasping of transparent objects. In 6th Annual Conference on Robot Learning.

[61] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In 2007 6th IEEE and ACM international symposium on mixed and augmented reality, pages 225–234. IEEE, 2007.

[62] Georg Klein and David Murray. Parallel tracking and mapping on a camera phone. In 2009 8th IEEE International Symposium on Mixed and Augmented Reality, pages 83–86, 2009. doi: 10.1109/ISMAR.2009.5336495.

[63] Lukas Koestler, Nan Yang, Niclas Zeller, and Daniel Cremers. Tandem: Tracking and dense mapping in real-time using deep multi-view stereo. In Conference on Robot Learning, pages 34–45. PMLR, 2022.

[64] Tomas Krajnik, Jaime Pulido Fentanes, Grzegorz Cielniak, Christian Dondrup, and Tom Duckett. Spectral analysis for long-term robotic mapping. In 2014 IEEE International Conference on Robotics and Automation (ICRA), pages 3706–3711. IEEE, 2014.

[65] L Kunze, H Karaoguz, J Young, F Jovan, J Folkesson, P Jensfelt, and N Hawes. Soma: a framework for understanding change in everyday environments using semantic object maps. pages 47–54.

[66] Edith Langer, Bram Ridder, Michael Cashmore, Daniele Magazzeni, Michael Zillich, and Markus Vincze. On-the-fly detection of novel objects in indoor environments. In 2017 IEEE International Conference on Robotics and Biomimetics (ROBIO), pages 900–907. IEEE, 2017.

[67] Edith Langer, Timothy Patten, and Markus Vincze. Robust and efficient object change detection by combining global semantic information and local geometric verification. In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 8453–8460. IEEE, 2020.

[68] Yunzhu Li, Shuang Li, Vincent Sitzmann, Pulkit Agrawal, and Antonio Torralba. 3d neural scene representations for visuomotor control. In Conference on Robot Learning, pages 112–123. PMLR, 2022.

[69] Cheng-Wei Lin, Tung-I Chen, Hsin-Ying Lee, Wen-Chin Chen, and Winston H Hsu. Coarse-to-fine point cloud registration with se (3)-equivariant representations. arXiv preprint arXiv:2210.02045, 2022.

[70] Yen-Chen Lin, Pete Florence, Andy Zeng, Jonathan T Barron, Yilun Du, Wei-Chiu Ma, Anthony Simeonov, Alberto Rodriguez Garcia, and Phillip Isola. Mira: Mental imagery for robotic affordances. In 6th Annual Conference on Robot Learning, 2022.

[71] Stefan Lionar, Lukas Schmid, Cesar Cadena, Roland Siegwart, and Andrei Cramariuc. Neuralblox: Real-time neural representation fusion for robust volumetric mapping. In 2021 International Conference on 3D Vision (3DV), pages 1279–1289. IEEE, 2021.

[72] Lukas Luft, Alexander Schaefer, Tobias Schubert, and Wolfram Burgard. Detecting changes in the environment based on full posterior distributions over real-valued grid maps. IEEE Robotics and Automation Letters, 3(2):1299–1305, 2018.

[73] Andrew Luo, Yilun Du, Michael J Tarr, Joshua B Tenenbaum, Antonio Torralba, and Chuang Gan. Learning neural acoustic fields. arXiv preprint arXiv:2204.00628, 2022.

[74] Lingni Ma, Christian Kerl, Jörg Stückler, and Daniel Cremers. Cpa-slam: Consistent plane-model alignment for direct rgb-d slam. In 2016 IEEE International Conference on Robotics and Automation (ICRA), pages 1285–1291. IEEE, 2016.

[75] John McCormac, Ankur Handa, Andrew Davison, and Stefan Leutenegger. Semanticfusion: Dense 3d semantic mapping with convolutional neural networks. In 2017 IEEE International Conference on Robotics and automation (ICRA), pages 4628–4635. IEEE, 2017.

[76] John McCormac, Ronald Clark, Michael Bloesch, Andrew Davison, and Stefan Leutenegger. Fusion++: Volumetric object-level SLAM. In 2018 international conference on 3D vision (3DV), pages 32–41. IEEE, 2018.

[77] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In Proc. CVPR, 2019.

[78] 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 Proc. ECCV, 2020.

[79] Alexander Millane, Zachary Taylor, Helen Oleynikova, Juan Nieto, Roland Siegwart, and César Cadena. C-blox: A scalable and consistent tsdf-based dense mapping approach. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 995–1002. IEEE, 2018.

[80] Alexander James Millane, Helen Oleynikova, Christian Lanegger, Jeffrey Delmerico, Juan Nieto, Roland Siegwart, Marc Pollefeys, and Cesar Cadena Lerma. Freetures: Localization in signed distance function maps. IEEE Robotics and Automation Letters, 2021.

[81] M. Montemerlo and S. Thrun. Simultaneous localization and mapping with unknown data association using FastSLAM. In 2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422), volume 2, pages 1985–1991 vol.2, 2003. doi: 10.1109/ROBOT.2003.1241885.

[82] Arthur Moreau, Nathan Piasco, Dzmitry Tsishkou, Bogdan Stanciulescu, and Arnaud de La Fortelle. Lens: Localization enhanced by nerf synthesis. In Conference on Robot Learning, 2022.

[83] Raul 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.

[84] 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 2011 10th IEEE International Symposium on Mixed and Augmented Reality, pages 127–136, 2011. doi: 10.1109/ISMAR.2011.6092378.

[85] 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 2011 10th IEEE international symposium on mixed and augmented reality, pages 127–136. IEEE, 2011.

[86] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In 2011 international conference on computer vision, pages 2320–2327. IEEE, 2011.

[87] 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, 2018.

[88] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In Proc. ICCV, 2019.

[89] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In Proceedings of the IEEE International Conference on Computer Vision, pages 5379–5389, 2019.

[90] Helen Oleynikova, Zachary Taylor, Marius Fehr, Roland Siegwart, and Juan Nieto. Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning. In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1366–1373. IEEE, 2017.

[91] Edwin Olson and Pratik Agarwal. Inference on networks of mixtures for robust robot mapping. The International Journal of Robotics Research, 32(7):826–840, 2013.

[92] OpenAI. Gpt-4 technical report, 2023.

[93] Joseph Ortiz, Alexander Clegg, Jing Dong, Edgar Sucar, David Novotny, Michael Zollhoefer, and Mustafa Mukadam. isdf: Real-time neural signed distance fields for robot perception. In RSS, 2022.

[94] Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Colored point cloud registration revisited. In 2017 IEEE International Conference on Computer Vision (ICCV), pages 143–152, 2017. doi: 10.1109/ICCV.2017.25.

[95] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In Proc. CVPR, 2019.

[96] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. ROS: an open-source robot operating system. In ICRA workshop on open source software, volume 3, page 5. Kobe, Japan, 2009.

[97] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In International conference on machine learning, pages 8748–8763. PMLR, 2021.

[98] Victor Reijgwart, Alexander Millane, Helen Oleynikova, Roland Siegwart, Cesar Cadena, and Juan Nieto. Voxgraph: Globally consistent, volumetric mapping using signed distance function submaps. IEEE Robotics and Automation Letters, 5(1):227–234, 2019.

[99] Antoni Rosinol, John J Leonard, and Luca Carlone. NeRF-SLAM: Real-time dense monocular SLAM with neural radiance fields. arXiv preprint arXiv:2210.13641, 2022.

[100] Martin Runz, Maud Buffier, and Lourdes Agapito. Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects. In 2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pages 10–20. IEEE, 2018.

[101] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In 2009 IEEE international conference on robotics and automation, pages 3212–3217. IEEE, 2009.

[102] Hyunwoo Ryu, Jeong-Hoon Lee, Hong-in Lee, and Jongeun Choi. Equivariant descriptor fields: Se (3)-equivariant energy-based models for end-to-end visual robotic manipulation learning. arXiv preprint arXiv:2206.08321, 2022.

[103] Renato F Salas-Moreno, Richard A Newcombe, Hauke Strasdat, Paul HJ Kelly, and Andrew J Davison. SLAM++: Simultaneous localisation and mapping at the level of objects. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1352–1359, 2013.

[104] Renato F Salas-Moreno, Ben Glocken, Paul HJ Kelly, and Andrew J Davison. Dense planar slam. In 2014 IEEE international symposium on mixed and augmented reality (ISMAR), pages 157–164. IEEE, 2014.

[105] Lukas Schmid, Jeffrey Delmerico, Johannes Schönberger, Juan Nieto, Marc Pollefeys, Roland Siegwart, and Cesar Cadena. Panoptic multi-tsdfs: a flexible representation for online multi-resolution volumetric mapping and long-term dynamic scene consistency. arXiv preprint arXiv:2109.10165, 2021.

[106] Lukas Schmid, Jeffrey Delmerico, Johannes Schönberger, Juan Nieto, Marc Pollefeys, Roland Siegwart, and Cesar Cadena. Panoptic multi-tsdfs: a flexible representation for online multi-resolution volumetric mapping and long-term dynamic scene consistency. In 2022 IEEE International Conference on Robotics and Automation (ICRA), 2022.

[107] Thomas Schops, Torsten Sattler, and Marc Pollefeys. Bad slam: Bundle adjusted direct rgb-d slam. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 134–144, 2019.

[108] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. AirSim: High-fidelity visual and physical simulation for autonomous vehicles. In Field and service robotics, pages 621–635. Springer, 2018.

[109] Bokui Shen, Zhenyu Jiang, Christopher Choy, Leonidas J Guibas, Silvio Savarese, Anima Anandkumar, and Yuke Zhu. Acid: Action-conditional implicit visual dynamics for deformable object manipulation. arXiv preprint arXiv:2203.06856, 2022.

[110] Anthony Simeonov, Yilun Du, Andrea Tagliasacchi, Joshua B Tenenbaum, Alberto Rodriguez, Pulkit Agrawal, and Vincent Sitzmann. Neural descriptor fields: Se (3)-equivariant object representations for manipulation. arXiv preprint arXiv:2112.05124, 2021.

[111] Anthony Simeonov, Yilun Du, Andrea Tagliasacchi, Joshua B. Tenenbaum, Alberto Rodriguez, Pulkit Agrawal, and Vincent Sitzmann. Neural descriptor fields: Se(3)-equivariant object representations for manipulation. In 2022 International Conference on Robotics and Automation (ICRA), pages 6394–6400, 2022. doi: 10.1109/ICRA46639.2022.9812146.

[112] Anthony Simeonov, Yilun Du, Lin Yen-Chen, Alberto Rodriguez, Leslie Pack Kaelbling, Tomas Lozano-Perez, and Pulkit Agrawal. Se (3)-equivariant relational rearrangement with neural descriptor fields. arXiv preprint arXiv:2211.09786, 2022.

[113] Julian Straub, Trevor Campbell, Jonathan P How, and John W Fisher. Small-variance nonparametric clustering on the hypersphere. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 334–342, 2015.

[114] Michael Strecke and Joerg Stueckler. EM-Fusion: Dynamic object-level SLAM with probabilistic data association. In 2019 IEEE/CVF International Conference on Computer Vision (ICCV). IEEE, oct 2019. doi: 10.1109/iccv.2019.00596.

[115] Edgar Sucar, Kentaro Wada, and Andrew Davison. NodeSLAM: Neural object descriptors for multi-view shape reconstruction. In 2020 International Conference on 3D Vision (3DV), pages 949–958. IEEE, 2020.

[116] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J Davison. imap: Implicit mapping and positioning in real-time. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 6229–6238, 2021.

[117] Niko Sünderhauf and Peter Protzel. Towards a robust back-end for pose graph SLAM. In 2012 IEEE international conference on robotics and automation, pages 1254–1261. IEEE, 2012.

[118] N. Sünderhauf and P. Protzel. Switchable constraints for robust pose graph SLAM. In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1879–1884, 2012. doi: 10.1109/IROS.2012.6385590.

[119] Yuichi Taguchi, Yong-Dian Jian, Srikumar Ramalingam, and Chen Feng. Point-plane slam for hand-held 3d sensors. In 2013 IEEE International Conference on Robotics and Automation (ICRA), pages 5182–5189. IEEE, 2013.

[120] Juan Jose Tarrio and Sol Pedre. Realtime edge-based visual odometry for a monocular camera. In Proceedings of the IEEE International Conference on Computer Vision, pages 702–710, 2015.

[121] Keisuke Tateno, Federico Tombari, and Nassir Navab. When 2.5d is not enough: Simultaneous reconstruction, segmentation and recognition on dense SLAM. In 2016 IEEE International Conference on Robotics and Automation (ICRA), pages 2295–2302, 2016. doi: 10.1109/ICRA.2016.7487378.

[122] Zachary Teed and Jia Deng. Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. Advances in neural information processing systems, 34:16558–16569, 2021.

[123] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. arXiv preprint arXiv:1809.10790, 2018.

[124] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. Demon: Depth and motion network for learning monocular stereo. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 5038–5047, 2017.

[125] Aisha Walcott-Bryant, Michael Kaess, Hordur Johannsson, and John J Leonard. Dynamic pose graph slam: Long-term mapping in low dynamic environments. In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1871–1878. IEEE, 2012.

[126] Johanna Wald, Armen Avetisyan, Nassir Navab, Federico Tombari, and Matthias Nießner. Rio: 3d object instance re-localization in changing indoor environments. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 7658–7667, 2019.

[127] 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 Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3343–3352, 2019.

[128] Jingwen Wang, Martin Rünz, and Lourdes Agapito. DSP-SLAM: Object oriented SLAM with deep shape priors. In 2021 International Conference on 3D Vision (3DV), pages 1362–1371. IEEE, 2021.

[129] Yue Wang and Justin M. Solomon. Deep closest point: Learning representations for point cloud registration. In The IEEE International Conference on Computer Vision (ICCV), October 2019.

[130] Yue Wang and Justin M. Solomon. PRNet: Self-supervised learning for partial-to-partial registration. In 33rd Conference on Neural Information Processing Systems, 2019.

[131] Thomas Whelan, John McDonald, Michael Kaess, Maurice Fallon, Hordur Johannsson, and John J. Leonard. Kintinuous: Spatially extended kinectfusion, July 2012.

[132] Thomas Whelan, Stefan Leutenegger, R Salas-Moreno, Ben Glocker, and Andrew Davison. Elasticfusion: Dense slam without a pose graph. Robotics: Science and Systems, 2015.

[133] Jay M. Wong, Vincent Kee, Tiffany Le, Syler Wagner, Gian-Luca Mariottini, Abraham Schneider, Lei Hamilton, Rahul Chipalkatty, Mitchell Hebert, David M.S. Johnson, and et al. SegICP: Integrated deep semantic segmentation and pose estimation. 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Sep 2017. doi: 10.1109/iros.2017.8206470. URL http://dx.doi.org/10.1109/IROS.2017.8206470.

[134] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2, 2019.

[135] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. PoseCNN: A convolutional neural network for 6Dobject pose estimation in cluttered scenes. arXiv preprint arXiv:1711.00199, 2017.

[136] Binbin Xu, Wenbin Li, Dimos Tzoumanikas, Michael Bloesch, Andrew Davison, and Stefan Leutenegger. Mid-fusion: Octree-based object-level multi-instance dynamic SLAM. In 2019 International Conference on Robotics and Automation (ICRA), pages 5231–5237. IEEE, 2019.

[137] Shichao Yang and Sebastian Scherer. Direct monocular odometry using points and lines. In 2017 IEEE International Conference on Robotics and Automation (ICRA), pages 3871–3877. IEEE, 2017.

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

[139] Lin Yen-Chen, Pete Florence, Jonathan T. Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. iNeRF: Inverting neural radiance fields for pose estimation. In IROS, 2021.

[140] Lin Yen-Chen, Pete Florence, Jonathan T. Barron, Tsung-Yi Lin, Alberto Rodriguez, and Phillip Isola. NeRF-Supervision: Learning dense object descriptors from neural radiance fields. In ICRA, 2022.

[141] Jiazhao Zhang, Chenyang Zhu, Lintao Zheng, and Kai Xu. Fusion-aware point convolution for online semantic 3d scene segmentation. In Proceedings of the

IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 4534–4543, 2020.

[142] Shuaifeng Zhi, Michael Bloesch, Stefan Leutenegger, and Andrew J Davison. Scenecode: Monocular dense semantic reconstruction using learned encoded scene representations. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 11776–11785, 2019.

[143] Huizhong Zhou, Benjamin Ummenhofer, and Thomas Brox. Deeptam: Deep tracking and mapping. In Proceedings of the European conference on computer vision (ECCV), pages 822–838, 2018.

[144] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. arXiv:1801.09847, 2018.

[145] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing. arXiv preprint arXiv:1801.09847, 2018.

[146] Minghan Zhu, Maani Ghaffari, and Huei Peng. Correspondence-free point cloud registration with so (3)-equivariant implicit shape representations. In Conference on Robot Learning, pages 1412–1422. PMLR, 2022.

[147] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R Oswald, and Marc Pollefeys. Nice-SLAM: Neural implicit scalable encoding for SLAM. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 12786–12796, 2022.