DISSERTATION

Object Change Detection for Autonomous Indoor Robots in Open-World Settings

conducted in partial fulfillment of the requirements for the degree of a

Doktor der technischen Wissenschaften (Dr. techn.)

supervised by

Ao.Univ. Prof. Dipl.-Ing. Dr. techn. Markus Vincze
E376 Automation and Control Institute

submitted at the

TU Wien
Faculty of Electrical Engineering and Information Technology

by

Dipl.-Ing. Edith Langer
DOB 11.09.1988
Matr. Nr.: 00725806

Vienna, March 2023 Edith Langer

Acknowledgment

I like to thank Markus Vincze for accepting me in the Vision4Robotics group and for his professional but also personal support during my Ph.D. journey. At its beginning, Michael Zillich supported me in finding my research direction. Thank you!

I'm grateful that Timothy Patten joined the group. With his kind and caring way and his endless support in discussing ideas and writing papers, he has contributed a great deal to the success of this work. Thank you Tim for proofreading my thesis in your spare time!

I'm glad to have met all the great people working together with me in the V4R group. Special thanks to Georg Halmetschlager-Funek and Markus Suchi for collecting one or the other ECTS as a group, Jean-Baptiste Weibel for fruitful discussions, Simon for making the time at university more fun, and Matthias for being the heart of the lab and gluing it together and for the pre-pandemic tea breaks to cheer me up when I was close to giving up. I also like to thank Markus Leitner, Christian Eder, and Kevin Wolfram for their help in setting up robot experiments and annotating data.

I am deeply indebted to Ursula Hufnagl, one of the most selfless people I know. Thank you for your time, your compassion, and for never giving up on me.

I am grateful for the friends I met during my time in the student dorm, at VrVis, and university. A special mention deserves the MIST - although we don't manage to meet regularly anymore, I always enjoy the time we spend together.

Andrea and Maria, I know you now for more than half of my life. Thanks for listening and for the encouraging words when I doubted research and myself. You are always there for me! I hope we will continue testing and eating a lot of ice cream together in the future!

A heartful thank you to my family, especially my parents, who made it possible for me to study in the first place and always encouraged me on my way.

Last but not least I like to express my special gratitude to Harri who supported me in every possible way and it is no exaggeration to say that without him I would not have been able to complete this journey.

This work was funded by the European Community's Seventh Framework Programme under grant agreement no. 610532 (SQUIRREL), the Vienna Science and Technology Fund (WWTF) under grant agreement No.ICT15-045 (RALLI), the Austrian Science Fund (FWF) under grant agreement I3967-N30 (BURG), and the Austrian Research Promotion Agency (FFG) under grant agreement 879878 (K4R).

Abstract

Deploying robots in homes to fulfill advanced tasks such as tidying up or filling the dishwasher, requires the capability to understand and adapt to changing environments. Detecting task-relevant objects is a key aspect for further steps such as object recognition and manipulation. Available learning-based object detectors achieve reasonable results when training and test data stem from the same distribution and all objects are known in advance. However, in an open-world setting, it is not enough to handle known objects, the robot also needs to detect objects it has never seen before. Another important aspect of object detection is the question of where in the environment it makes sense for a robot to look for objects. The proposed methods in this thesis not only support the robot in exploring the environment continuously but also perform tasks such as open-world object change detection. Due to the interest in not only table-top scenes but larger entities such as full rooms, 3D reconstructions are a common choice to represent environments. Reconstructions are used to store a specific state of the environment as a reference for future comparisons to the current state. This reveals changes in the environment depicting potentially interesting objects.While the first presented method uses voxel-based differencing to detect novel objects, the advantage compared to other works is that the current recording of the environment is created on-the-fly while the robot moves around to fulfill an unrelated task. In order to make differencing robust against insignificant changes in the environment, such as small movements of furniture, the second approach exploits semantic segmentation allowing the division of the environment into meaningful parts. This concept builds the basis for the third approach where a point pair feature descriptor is used to match detected objects from different timestamps and categorize them either into static, moved, removed, or novel.

For qualitative evaluation, and as part of the contribution of this thesis, we create the new dataset *ObChange*, which is specifically designed for detecting changes on the object-level. The dataset provides real-world environment reconstructions from different timestamps acquired with a robot and where small objects that change are annotated. Until now no other publicly available dataset combines all of these characteristics. We use *ObChange* to show that our method using semantic segmentation to partition an environment achieves superior object detection results compared to global differencing methods. Furthermore, we use the dataset to compare our object detection and matching method against an adaption of recently published learning-based work. Even when only considering objects presented in the training set, the baseline falls behind our approach when considering real-world scenarios. Additionally, real-world robot experiments demonstrate the applicability of the proposed methods.

Kurzzusammenfassung

Roboter sind längst in der Industrie, aber auch im alltäglichen Leben angekommen. Obwohl Aufgaben wie Staubsaugen gut bewältigbar sind, stoßen autonome Roboter selbst bei scheinbar einfachen Aufgaben wie etwa Zimmer aufräumen oder das Befüllen eines Geschirrspülers an ihre Grenzen. Ein Grund dafür ist die dynamische Umgebung und das notwendige, stets aktualisierte Verständnis darüber. Dabei ist die Detektion aufgabenrelevanter Objekte essentiell für weitere Schritte wie etwa Objektwiedererkennung und -manipulation.

Verfügbare lernbasierte Systeme zur Objektdetektierung erzielen gute Ergebnisse, wenn Trainings- und Testdaten aus der gleichen Verteilung stammen und alle Objekte im Voraus bekannt sind. Allerdings treffen diese Voraussetzungen für viele Umgebungen und Aufgaben nicht zu. Es ist daher essentiell, dass Roboter sich auch in nicht abgeschlossenen Umgebungen (*open world*) zurechtfinden und Objekte detektieren können, die sie noch nie zuvor gesehen haben. Eine wichtige zu klärende Frage ist, wo in der Umgebung ein Roboter sinnvollerweise nach Objekten suchen soll. Die in dieser Arbeit vorgestellten Methoden ermöglichen dem Roboter sowohl die kontinuierliche Erkundung der *open-world* Umgebung als auch die Detektierung der darin befindlichen Objekte, sollten diese neu sein oder ihre Position verändert haben. Um die Methoden nicht nur auf Tischszenen zu beschränken, sondern auch auf größere Umgebungen anzuwenden, wie zum Beispiel ganze Räume, sind 3D-Rekonstruktionen eine gängige Wahl. Rekonstruktionen werden verwendet, um einen bestimmten Zustand der Umgebung als Referenz zu speichern und diesen mit zukünftigen Zuständen zu vergleichen. Auf diese Weise werden Veränderungen in der Umgebung sichtbar, die potenziell interessante Objekte darstellen.

Die erste vorgestellte Methode detektiert Objekte, indem sie zu unterschiedlichen Zeitpunkten aufgenommene, voxelbasierte Rekonstruktionen einer Umgebung miteinander vergleicht. Der Vorteil gegenüber anderen Arbeiten besteht darin, dass dies gänzlich nebenbei erfolgen kann, während der Roboter sich im Raum bewegt und eine andere Aufgabe durchführt. Um die Unterschiede in einer Umgebung und damit die Objektdetektierung robust gegenüber unbedeutenden Veränderungen zu machen, wie zum Beispiel geringfügige Bewegungen von Möbeln, unterteilt der zweite Ansatz in dieser Arbeit die Umgebung in semantisch sinnvolle Einheiten, welche dann unabhängig abgearbeitet werden. Dieses Konzept bildet die Grundlage für die dritte Methode, bei der ein Merkmalsdeskriptor verwendet wird, um zu unterschiedlichen Zeitpunkten detektierte Objekte zu vergleichen und sie entweder als statisch, verschoben, entfernt oder neu zu kategorisieren.

Ein wichtiger Beitrag dieser Arbeit ist der neue Datensatz *ObChange*. Er eignet sich besonders zur quantitativen Auswertung von Methoden, welche Objekte durch den Vergleich zweier Umgebungszustände detektieren. Der Datensatz besteht aus Rekonstruktionen von realen Umgebungen zu verschiedenen Zeitpunkten, die mit einem Roboter aufgenommen wurden und in denen kleine Objekte annotiert sind. Bislang gibt es keinen anderen veröffentlichten Datensatz, der all diese Eigenschaften vereint. Basierend auf *ObChange* erzielt die vorgestellte Objektdetektierungsmethode, welche semantische Information zur Partitionierung einer Umgebung verwendet, bessere Ergebnisse als globale Vergleichssmethoden. Darüber hinaus verwenden wir den Datensatz, um unsere Methode, welche nicht nur Objekte detektiert sondern diese auch zwischen zwei Zeitpunkten abgleicht und kategorisiert, zu evaluieren. Die Ergebnisse werden mit einer leicht abgeänderten Form einer kürzlich vorgestellten Arbeit, welche lernbasierte Objektdetektoren verwendet, verglichen. Selbst wenn nur die im Trainingsset verwendeten Objekte berücksichtigt werden, schneidet der lernbasierte Ansatz schlechter ab. Zusätzlich zu den erzielten Ergebnissen der vorgestellten Methoden auf *ObChange*, zeigen wir mit durchgeführten Roboterexperimenten, dass die Ansätze auch in der echten Welt anwendbar sind.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Human expectations of robot capabilities are influenced by movies and convincing marketing and advertisement. While research in the field of robotics has made huge progress over the last decade, there is still a large gap between what humans expect from robots and what they are really capable of doing robustly in real-world settings. Let us think about robots beyond research prototypes that are currently operating in the world. Industrial robots in factories and robot vacuum cleaners at home are the first examples that immediately come to mind. Not so common, but already in use, are delivery robots in restaurants bringing food and drinks to tables as well as robots as museum guides or in hotels to inform tourists about nearby attractions, showing them the elevator or the dining room. Figure 1.1 shows four of the previously mentioned robots as examples of robots deployed in the real world.

All these available robots have one thing in common: their main goal is to plan collision-free trajectories in a known and pre-mapped environment. Additionally, some of them transfer information via determined sentences, simple dialogues, or shown on an attached screen. While the aforementioned service robots, which assist humans, may get in contact with unknown objects, they do not require further information about them. For example, it is irrelevant if the object blocking their way is a backpack or a potted plant as long as they are able to navigate around it. Robots, which are designed to manipulate objects, usually apply predefined actions to known objects, such as an industry robot assembling a product.

The mentioned applications for commercially available robots show that, at the moment, robots demonstrate their strength in structured environments performing predictable actions, which do not require learning about new objects. Dealing with changing and dynamic environments is therefore an active research area [1]–[3]. Also, the fact that many tasks of a service robot require the ability to work in open-world settings, where not every object is known by the robot in advance, recently gained attention in research [4]–[7]. To unlock the potential of robots, future approaches need to enable robots to adapt to and semantically understand the environment and the objects within as well as to learn about new objects.

With these observations in mind, the goal of the developed and presented methods in this thesis is to improve a robot's ability to autonomously detect objects in open-world environments. The overarching main idea of the techniques introduces in this thesis

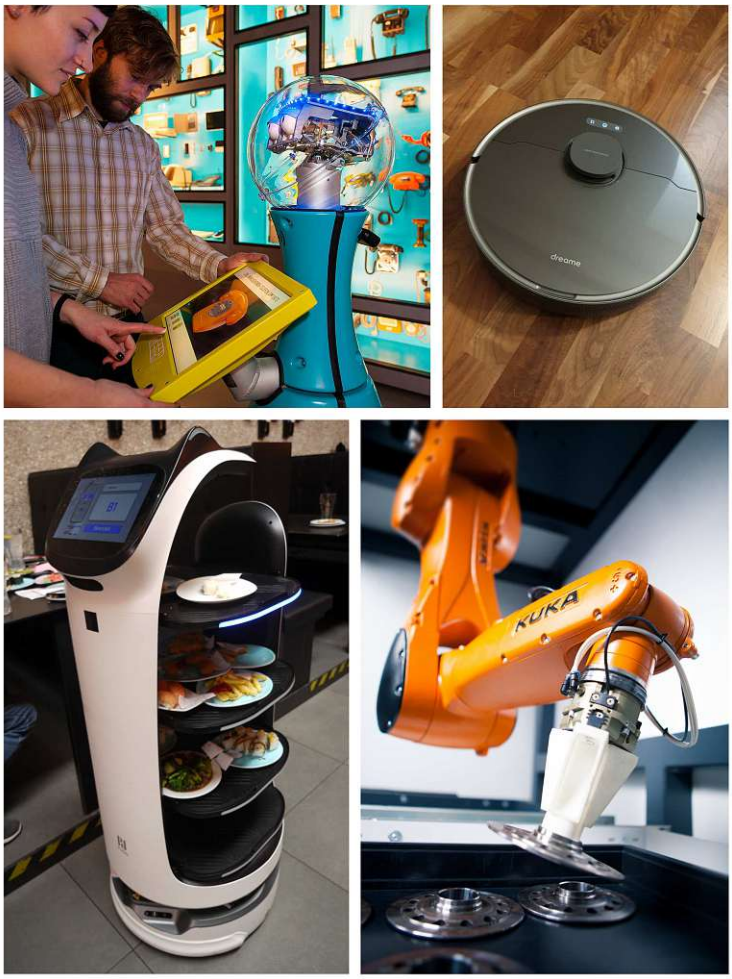


Figure 1.1: Examples of commercially available robots deployed outside of research labs. From top to bottom and left to right: robot guide by MetraLabs in the German Museum of Technology (©SDTB/C. Kirchner), robot vacuum cleaner by Dreame, food delivery robot by Pudu, industrial robot by Kuka (©KUKA AG).

is to re-use as much information as possible. For example, when considering scene understanding, instead of repeatedly analyzing the whole environment all over again to capture and understand the current situation, we propose to focus on relevant regions by utilizing change detection and concentrating the analysis on the identified changes. An example where the concept of change detection is highly effective is a fetch-and-carry robot for the elderly [8], [9], which first needs to identify where the required object is placed. Another example is a surveillance robot [10]. It needs to detect if objects are out of place compared to the normal state of an environment. One way to obtain the locations of such objects is via change detection by finding differences between the current situation and the reference. This is also the initially required step for a robot with the task of tidying up an office or apartment[1]. The identified locations serve as

[1] http://www.squirrel-project.eu

the basis for subsequent processing steps. The robot navigates there, recognizes or classifies the object, derives the correct storage location, and brings the object to the target location.

The complexity of these applications increases significantly when the robot has to deal with real-world conditions such as dynamic environments, changing accessibility of viewpoints, cluttered scenes, and objects it may not have seen before. The latter is particularly relevant because a robot rarely has the chance to learn about all objects it encounters in advance. Generally, the successful accomplishment of the mentioned tasks requires the robot to detect known as well as unknown objects and further differentiate between static, moved, removed, and novel objects.

1.1 Motivation

To fulfill tasks such as fetching objects, patrolling, or tidying up, a mobile robot needs to detect objects robustly and efficiently. As a first step, it is important for the robot to know where to look for objects of interest and focus its attention on these locations. This is crucial to achieve satisfying results for subsequently needed computer vision algorithms dealing with object classification or recognition where the actual viewpoint of the camera heavily influences the accuracy. Loghmani et al. [11] show in their work that most of the datasets are created under constrained and artificial settings where objects are nicely placed in the center of the image. This leads to the fact that methods achieve excellent results when tested on the dataset, but underperform in the real world, particularly when deployed on a robot. Loghmani et al. extract two situations that are especially challenging for object classification methods. First, the classification accuracy depends on the object size in the image. Thus small objects and objects far away from the robot are difficult to correctly classify. Therefore, the smaller an object, the closer a robot should try to get. Second, occluded objects are found to be challenging. Distinctive parts may be hidden by other objects or they are cut-off because the object is not fully in the field of view. With these challenges in mind, it is essential that a mobile robot has the necessary knowledge of an object of interest, like position and size, in order to approach it in such a way that it is perfectly in the field of view for subsequently applied computer vision methods.

1.2 Problem Statement

A common approach to tackle the problem of where to focus the view is to extract regions of interest via change detection as employed by [12]–[17]. The basic idea is to compare the state of an environment at two different timestamps and to consider the differences as dynamic objects, which should gain the robot's attention. In contrast to a different research area where live motion is used for object discovery [18]–[21], change detection is based on the more relaxed assumption that the robot operates in a semi-static environment. To perform change detection it is not necessary that the robot actively observes when objects move or are moved. It has to be noted that methods, which utilize geometric change between two timestamps to detect objects,

do not directly provide information about the semantics of the objects and are not able to tell if discovered differences represent one object or multiple objects forming a heap. Therefore, when considering more complex tasks, such as tidying up a room autonomously, more information about objects is needed. Robust object detection, however, is an important prerequisite for follow-up computer vision techniques, such as segmentation or classification.

To perceive the environment, technologically advanced robots are typically equipped with cameras. Low-cost RGB-D cameras using time-of-flight or structured light technology, e. g. Microsoft Kinect, Asus Xtion, or Intel RealSense, are a common choice. Besides an RGB image, these cameras also provide aligned dense depth measurements, which supplement computer vision algorithms with an additional and highly relevant dimension. Since these types of cameras are sensitive to sunlight, we focus on indoor applications and leverage the fact that indoor environments are always man-made. Therefore, it is a fair and common assumption that objects sit on nearly horizontal planes considering physical plausibility [22], [23]. Tables, shelves, sofas, chairs, and even the floor are examples of possible supporting planes for objects.

The kind of objects we want to detect using change as a cue is potentially detachable from the supporting plane and of a size manageable by a mobile robot, while other works usually focus on bigger objects such as furniture [16], [24]. In general, objects detected by comparing the same environment at two different timestamps, t_0 and t_1, can be categorized into static, moved, removed, and novel. These four categories represent all possibilities and, therefore, a detected object has to be assigned to one of them. The definition of the categories is given in Table 1.1 and Figure 1.2 shows an example of object change detection by comparing a kitchen scene recorded at two different timestamps. Previous works, such as [12], [14], [25], and the methods described in Chapters 4 and 5, are only interested in the fact that something has changed between t_0 and t_1, meaning if an object is new at respectively removed from its current location. This is in contrast to the general definition given in Table 1.1, which takes the whole environment into account and not only a certain location. To be able to differentiate between truly novel respectively removed and moved objects in an environment, the detected objects need to be matched between the two timestamps t_0 and t_1. In Chapter 6 we present an object detection method with a subsequent matching step. Therefore, this method is able to categorize the objects into the four defined categories.

Table 1.1: Definition of the different categories an object gets assigned to when comparing an environment at two different timestamps.

Category	**Description**
Static	Object moved less than a distance d at time t_1 compared to t_0
Moved	Object is detected at time t_0 and at time t_1, but at different locations
Removed	Object is detected at time t_0 but not at time t_1
Novel	Object is detected at time t_1 but not at time t_0

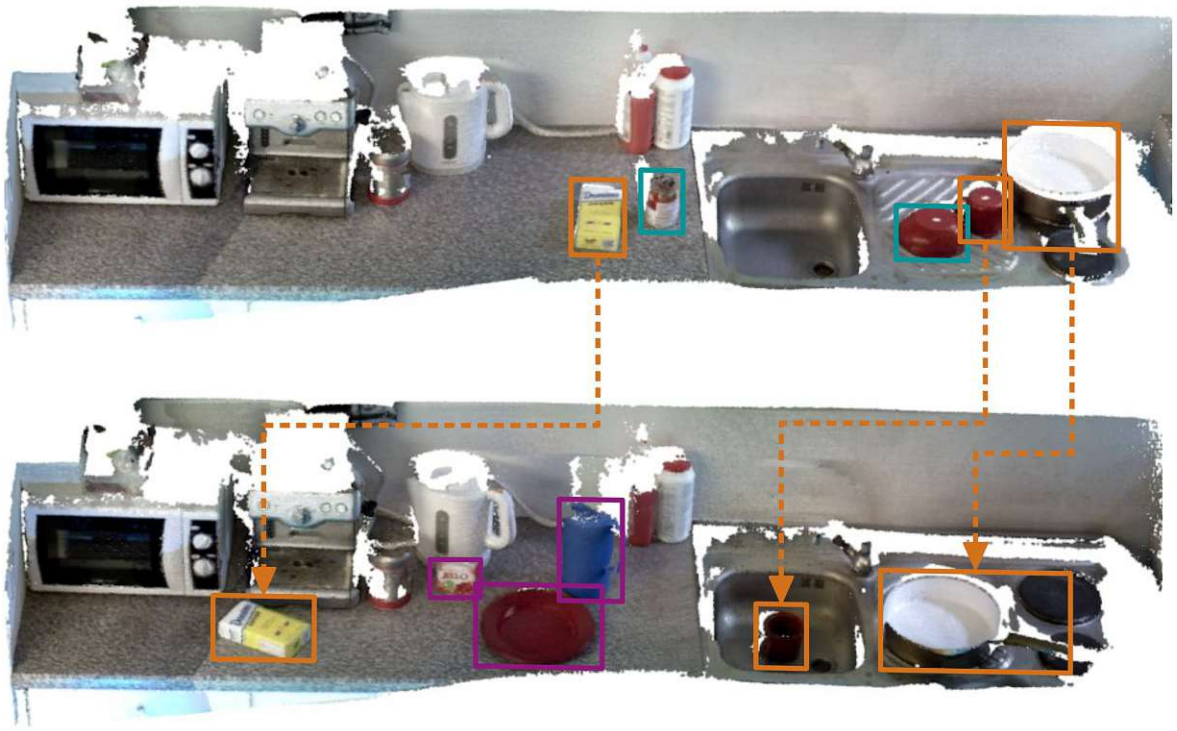


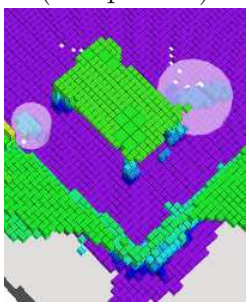
Figure 1.2: A kitchen scene at two different timestamps where objects of interest, i. e. changes, are marked with a bounding box. Orange: object moved, turquoise: object was removed, violet: object is novel.

1.3 Contributions and Outline

In this thesis, we present methods to discover objects based on the comparison of an indoor environment represented as 3D reconstruction at two different timestamps. The main contributions of the thesis are as follows:

- an annotated dataset consisting of five different indoor environments, which is suitable to evaluate object change detection and object matching methods (Chapter 3),
- unsupervised object change detection and object matching methods, which are independent of training data and consequently, there is no need for tedious object modeling (Chapters 4, 5, and 6),
- efficient methods, which are applicable in open-world settings because dynamic objects do not need to be known in advance (Chapters 4, 5, and 6),
- leveraging semantic information to systematically focus the change detection methods on promising regions (Chapters 5 and 6), and
- additional experiments performed with real robots showing the practicality of the proposed methods (Chapters 4 and 6).

Table 1.2: Overview of the methods presented in this thesis. Differences are pointed out by three main aspects.

	On-the-Fly (Chapter 4)	**Local Verification** (Chapter 5)	**Object Mapping** (Chapter 6)
			
Reconstruction Method	OctoMap	Voxblox	Voxblox, ElasticFusion
Object Locations	Floor	Floor, horizontal planes	Horizontal planes
Object Categories	Novel	Novel	Static, moved, removed, novel

While the three methods presented in this thesis have a slightly different focus, commonly for all of them, their output contains the location and the spatial extent of all detected objects. Depending on the reconstruction method used to map the environment, the output may consist of a more detailed depiction of the objects, such as a better geometric representation or the incorporation of color. In summary, the developed methods differ in the following main aspects:

- The **reconstruction method** used to represent the environment – *OctoMap* [26], *voxblox* [27], or *ElasticFusion* [28]
- The **locations** at which objects are discovered – On the floor and/or placed on other horizontal planes
- The object **categories**, which are distinguished when comparing the current recording to a reference – Static, moved, removed, and/or novel objects.

Table 1.2 outlines how the three object change detection methods presented in this thesis relate to these aspects, while the following subsections give a compact description. It has to be noted that the first two methods automatically categorize detected objects as novel, in the sense of novel at this specific location compared to novel in the environment, which is the case for the third method.

The following subsections outline the ideas of the change detection methods discussed in this thesis.

1.3.1 On-the-Fly Detection of Novel Objects using *OctoMaps* – Chapter 4

Identifying objects in an environment, which were not at this location before, is an important capability for a robot, which conducts surveillance or maintains order in homes or industrial settings. In previous works, the robot navigates in the environment using pre-defined waypoints and analyzes the parts of the scene visible from these waypoints. Without knowing where to find novel objects, this process is time-consuming and prone to detecting false positives. To overcome these limitations we propose a method that combines navigation and attention in order to rapidly detect objects, which are new at that location. We exploit the *OctoMap*, which maps the environment as a voxel grid and is generated as a by-product while the robot moves around to fulfill its (unrelated) task. The voxel size of an *OctoMap* has to be considered as a trade-off between the degree of detail depicted in the map and the immanent noise of the integrated depth measurements. We use the current *OctoMap* of an environment to compare it to a reference. The reference is previously generated and depicts the target state of the environment. Computing the difference between the reference map and the map of the current state indicates regions of interest, which consist of one or more objects. These regions are attention cues where subsequent steps, like segmentation, should be performed. The focus of this work is on detecting objects on the floor, which are novel and therefore were not present at this location when the reference map was generated. We show the benefits of using object change detection in simulated as well as real-world experiments and present the duration and number of segmentation actions required to discover all novel objects. Figure 1.3 shows a simplified flowchart of the method. A more detailed description of the contributions, the approach, and the results are given in Chapter 4 and have been published in the scientific paper [Langer, ROBIO 2017].

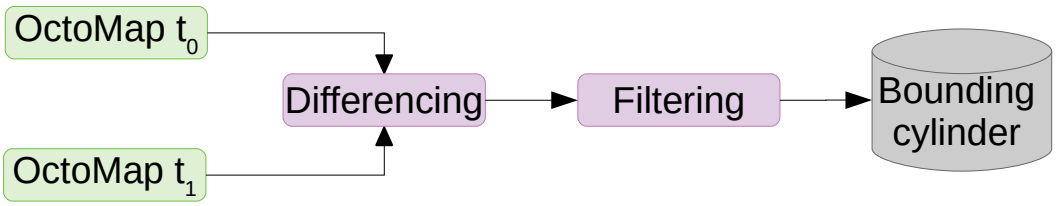


Figure 1.3: Overview of on-the-fly object change detection using *OctoMaps*.

1.3.2 Change Detection by Combining Global Semantic Information and Local Geometric Verification – Chapter 5

Previous object change detection approaches, such as [13], [14], [29], do not distinguish between novel objects or simple scene readjustments nor do they sufficiently deal with localization error and sensor noise. To overcome these limitations, we combine the strengths of global and local methods for the efficient detection of novel objects compared to a reference of the environment. In this work, the 3D reconstruction method

used to map the full environment is based on the *voxblox* library. Global structure in the reconstruction, determined from 3D semantic information, is exploited not only to directly establish object candidates but also to efficiently extract horizontal planes. These planes, including the floor, help to discover additional object candidates, which were missed by the semantic segmentation method because they are, e. g., too small. The set of object candidates is then locally verified by comparing isolated geometry to a reference reconstruction provided by the task. This is a two-step approach. First, for each object, the supporting plane gets aligned between the two timestamps to compensate for mapping inaccuracies. Second, the object itself gets aligned and the overlap indicates if the object in the current map is novel or not. We evaluate our approach on a real-world robotic dataset containing different types of environments with a total of 31 recordings and 260 annotated objects (see Section 3.2). Figure 1.4 shows a simplified flowchart of the method. A more detailed description of the contributions, the approach, and the results are given in Chapter 5 and have been published in the scientific paper [Langer, IROS 2020].

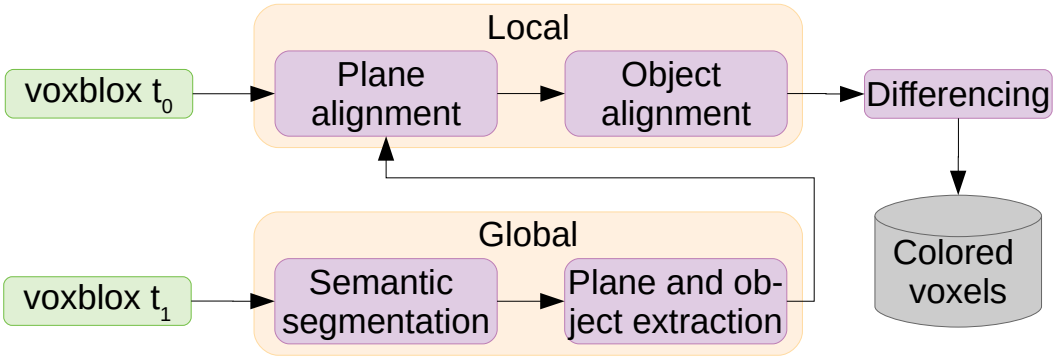


Figure 1.4: Overview of using global structures and local verification to detect novel objects.

1.3.3 Autonomous Object Mapping in Open-World Settings – Chapter 6

For numerous applications in robotics, it is not enough to detect objects, which are new at a location compared to a reference like in the two previously mentioned methods. In order to release the full potential of a service robot, it is necessary to distinguish between truly novel objects in the environment and objects that are out of place and have moved to a new location. With this knowledge available the robot is, e. g., able to bring back moved objects to their original position. The problem of categorizing detected objects into static, moved, removed, and novel is particularly challenging in open-world scenarios without a predefined set of objects the robot is going to encounter in the environment.

The main idea of the presented method is to create a full reconstruction of the environment using *voxblox* once and extract interesting surfaces that the robot should visit repeatably. For each surface of interest, a detailed reconstruction with *ElasticFusion*

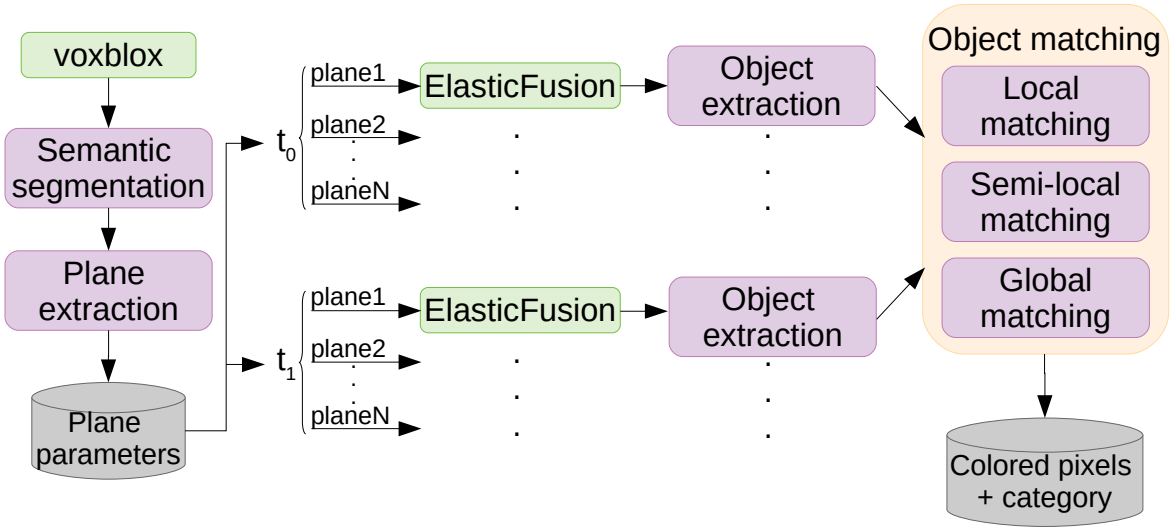


Figure 1.5: Overview of detecting and matching objects between two timestamps.

is generated. Based on these partial reconstructions we identify planes, consider clusters on top as objects, and compute their point-pair-feature descriptors. These feature descriptors are used to match potential objects between a reference and the current state and categorize them robustly into static, moved, removed, and novel objects even in the presence of partial object reconstructions and clutter. Our approach dissolves heaps of objects without specific object knowledge, but only with the information acquired from change detection. The evaluation is performed on real-world data, which include challenges affecting the quality of the reconstruction as a result of noisy input data. We perform a quantitative evaluation and compare our method against a baseline using learning-based object detection. The results show that, even in the case of a closed-world assumption where the training set of the baseline contains all objects occurring in the test scenes, our approach outperforms the baseline for most test cases. Lastly, we also deploy our method on a robot and demonstrate its effectiveness with real-world experiments. Figure 1.5 shows a simplified flowchart of the method. A more detailed description of the contributions, the approach, and the results are given in Chapter 6 and have been published in the scientific paper [Langer, Frontiers 2022].

1.4 List of Publications

Parts of the content presented in this dissertation have been previously published in the following papers:

- *Edith Langer, Timothy Patten, and Markus Vincze.* **Where Does It Belong? Autonomous Object Mapping in Open-World Settings.** Frontiers in Robotics and AI, 2022. DOI: 10.3389/frobt.2022.828732

- *Edith Langer, Timothy Patten, and Markus Vincze.* **Robust and Efficient Object Change Detection by Combining Global Semantic Information and**

Local Geometric Verification. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020. DOI: 10.1109/IROS45743.2020.9341664

- *Edith Langer, Bram Ridder, Michael Cashmore, Daniele Magazzeni, Michael Zillich, and Markus Vincze.* **On-the-Fly Detection of Novel Objects in Indoor Environments.** IEEE International Conference on Robotics and Biomimetics (ROBIO), 2017. DOI: 10.1109/ROBIO.2017.8324532

- *Edith Langer, Michael Zillich, and Markus Vincze.* **On-The-Fly Detection of Regions of Interest to Find Dynamic Objects in Indoor Environments**. Poster at Austrian Robotics Workshop (ARW), 2016

Chapter 2

Related Work

This chapter gives an overview of available reconstruction methods using RGB-D data as input, reviews literature related to the work presented in this thesis dealing with unsupervised object discovery by exploiting changes in an environment at different timestamps, and discusses learning-based object detection approaches tackling the open-world challenge.

2.1 3D Representation of Environments

An accurate 3D representation of an environment requires the fusion of many camera frames. However, it is not expedient to fuse the raw data, which usually results in noisy reconstructions. In this section, we discuss reconstruction methods, which combine frames from RGB-D cameras in an intelligent way in real time. The list of methods is not comprehensive but covers the most popular ones. Available methods are divided according to their underlying geometric representation. In general, methods are either based on points or voxels, but exceptions exist, e. g. *ScalableFusion* [30] uses triangle meshes.

Two **voxel-based** mapping methods, which require camera poses relative to the world as input, are *OctoMap* [26] and *voxblox* [27]. Both are designed to run on a CPU only. While the former stores occupancy probabilities in octrees, the latter works with Truncated Signed Distance Fields (TSDF). To generate TSDFs each voxel stores the distance to the closest surface point up to a defined truncated distance t. Voxels behind a surface have a negative value and voxels in front have a positive one. The zero-crossing of the TSDF defines the surface. Usually, new input frames are integrated using ray casting. TSDF is a very popular choice for 3D reconstructions and is suitable to smooth out sensor noise when several frames of a region are given. All the following methods use TSDF, perform simultaneous localization and mapping (therefore do not require camera poses as input), and need a GPU for processing. *KinectFusion* [31] made TSDF popular for dense real-time reconstruction purposes. Whelan et al. [32] extended this approach to not only work with table-sized scenes but large environments. They also integrated loop closure and, in a further extension, color. Loop closure applies when coming back to a region, which was already mapped. It accounts for drift in pose estimation and subsequent distortion of the map. *InfiniTAM* [33] divides the

scene into overlapping submaps and then optimizes transformations between them to achieve better and more efficient loop closure results. *BundleFusion* [34] also works with submaps but in a hierarchical manner. First, consecutive frames are pooled into chunks and their poses are optimized. This intra-chunk optimization is followed by an inter-chunk optimization leading to a globally consistent reconstruction.

The advantage of **point-based** methods is that the camera input can be directly used without the need to compute an implicit function and convert it to a grid structure. The most prominent way to represent point-based maps is an unordered list of surfels. A surfel is a point in 3D space with additionally assigned attributes like normal, color, radius, and confidence. The radius of a surfel adapts to the density of the map, while the confidence value allows the differentiation between stable and unstable points. Confidence increases if data from several frames are associated with a point. Keller et al. [35] leverage the confidence value to deal with dynamic objects in the scene and exclude them from the reconstruction. *ElasticFusion* [28] adopts ideas from [35] and includes loop closure using non-rigid deformations. Unfortunately, the ability to reconstruct dynamic scenes is lost.

2.2 Unsupervised Change Detection for Indoor Environments

Object detection in RGB images, as well as 3D data, is an active field in computer vision and robotics. While most of the single frame, learning-based detectors are limited by their training dataset, some methods working on 3D data use only geometric properties [36], [37] or additional semantics [38]. Although the mentioned 3D methods are useful in open-world settings, this section reviews work related to object detection by comparing an environment at two different timestamps. This is closer to the work presented in this thesis as it limits the set of detected objects to the ones that changed, which are often relevant in robotic applications.

Available approaches performing object change detection can be divided into those computing the difference between (1) two single camera frames [13], [39], (2) two (partial) 3D reconstructions [14], [16], [17], [25], [29], [40]–[42] and (3) a reconstruction and a single frame [12], [15]. As for (1), the environment is usually captured from fixed waypoints, which the robot sequentially visits. This is not very flexible in terms of view capturing if, e. g., the pre-defined waypoint cannot be approached due to an obstacle. Furthermore, acquiring only one single view of an area is insufficient to capture it to its full extent and, especially in cluttered scenes, suffers from occlusion. However, the apparent extension of taking several views and performing change detection for each of them separately generates the problem of ultimately fusing the results, which may be ambiguous. This problem does not exist for methods falling into category (2), which directly work on 3D reconstructions to detect changes in an environment. Typically the 3D reconstructions used in this context are composed of geometric primitives such as points, surfels, voxels, or meshes. The drawback of reconstructions, when used to map larger areas, is the possibility of warping, smearing, and inaccuracies. This leads to a significant number of wrongly detected objects. Existing methods propose several

strategies to reduce their occurrences by filtering detected objects of small size [40], [41] or planar shape [13], [14], applying morphological operations [16], [41], or use a very restrictive camera trajectory [14].

Computing the difference between two recordings of an environment inherently leads to the distinction between static background and dynamic objects. Therefore, several methods exploit this information not only to discover objects but also to model the static parts of an environment and refine them with every visit [12], [14], [16], [25], [39].

2.2.1 Frame-to-Frame Comparison

Comparing two frames picturing the same area of an environment at different timestamps received attention shortly after the rise of affordable RGB-D cameras. Mason and Marthi [13] leverage the supporting surface assumption and detect horizontal planes in RGB-D frames while the robot moves around. Clusters on top of planes are referred to as objects. They store simple properties for each object, such as color, size, and shape. However, when the robot revisits the location of a known object, they only compare the two-dimensional convex hull to detect changes. This very limited change detection approach assumes a global reference frame and does not rely on registered frames. Alimi et al. [39] compare RGB-D frames from table-top scenes captured from pre-defined locations. The frames are registered using ICP and afterward downsampled to make differencing more robust against sensor noise. For each remaining cluster after differencing a feature vector based on geometric and visual appearance is computed and stored in a database. Feature vectors from previously stored objects are compared to classify the newly detected clusters as novel, moved, or removed.

Recently, Adam et al. [43] proposed an approach to detect added, moved, and removed objects in a reconstructed environment at two different timestamps. The initial change detection is obtained by computing the difference between two aligned and rendered depth images. Based on this difference, projected to the 3D space, supervoxels are built and further used to propagate the changes also to the parts of the object that were not detected in the first place. This leads to geometric transformation consistency. To compute the dominant transformations of objects, learned features (trained on a different task, therefore still an unsupervised method) for every point are extracted and then matched by nearest neighbor search. RANSAC is applied to extract 3D rigid transformations. For each transformation, a graph optimization problem is solved to segment the corresponding object. Finally, they perform a connected component analysis to resolve over-segmentation. This leads to the effect that in some cases it is not possible anymore to differentiate single objects. The different change categories (added, moved, removed) are not considered in the evaluation, which uses the *3RScan* dataset [24]. Similar to the designed baseline in Chapter 6 they use *Mask R-CNN* trained on the COCO dataset for their baseline to generate semantic object labels. These labels are used instead of the geometric consistency term to propagate changes.

In contrast to aligning images taken at a specific location, pairing frames from a robot's camera stream without any pose information is a non-trivial task and reduces the performance of change detection approaches as shown by Park et al. [44] and Weihs et al. [45].

2.2.2 Map-to-Map Comparison

Creating a 3D reconstruction of an environment and comparing it to a 3D reconstruction of the same environment (or parts of it) at a later point in time is a very common approach for unsupervised object change detection. Herbst et al. [25] use a probabilistic sensor model for change detection taking color, depth, and surface orientation into account. They use surface patches to describe the environment and determine the probability that the surface patches move between two aligned RGB-D maps. The approach is demonstrated with table-top scenes where just the differencing step alone takes 180 seconds. This indicates that the approach does not scale. Finman et al. [40] discover objects through differencing of reconstructions. They propose a free-space filtering technique, which prevents the method from detecting a change in an area that was unseen in one map but visible in the other one because of different viewing poses. Differencing requires an accurate alignment of the two reconstructions, therefore a bounding cube is manually set to identify the overlapping regions. The overlapping regions are then aligned using ICP, which tends to fail if big parts have changed. The focus of their work is on learning segmentation methods to re-discover objects in future visits. The best method assigned to an object is then used to segment the whole environment at the next visit. This result is used to find segments with features similar to the query object. The main goal of Ambrus et al. [14] is to create and maintain the static structure of a room, the so-called meta-room, which is subsequently used to extract dynamic objects. At each visit, a map is built from several registered point clouds observed from one location using different camera angles limiting the application to convex rooms. According to the clustered differences between two RGB-D scans of the environment, the meta-room is updated to model previously occupied parts. This method deals with structural room changes by detecting spatially and visually similar clusters over past observations, which are then added to the meta-room. In a follow-up work, Ambrus et al. [46] extend the approach to take sensor noise into account. Based on the meta-room approach Bore et al. [47] detect changed objects and additionally track the movement of these objects over time by defining a two-stage movement model, which is limited to a closed world. With the same goal in mind as Ambrus et al. [14], Fehr et al. [16] propose an approach to create a dense 3D reconstruction of the static structure of an environment using TSDF. The TSDF value is used to identify changes between the static map and the current observation. Dynamic objects are not integrated into the static map, but segmented and stored in a database. They apply the approach of Furrer et al. [36] to incrementally update and refine the stored objects.

In Chapter 5 and Chapter 6 we introduce and build on the idea of dividing an environment into meaningful regions to perform a more robust object change detection, which was recently adopted by Fu et al. [29]. They propose to model the environment as a set of SDF volumes, so-called PlaneSDFs, each of them representing a plane and its supported objects. For each PlaneSDF a 2D height map is computed and the thresholded difference between two of them results in a change mask. To filter wrongly detected objects they compare the 3D local surface geometry. For each key voxel of an object and its counterpart in the target environment, a shape descriptor and their similarity is computed. The distribution of all similarity scores is used to determine if

an object has changed. The quantitative evaluation using our newly created dataset (see Chapter 3) shows improved results compared to [17], with the constraint that their method also detects slightly moved furniture or objects, which were manually excluded.

2.2.3 Frame-to-Map Comparison

Similar to their previous work [25] Herbst et al. [12] detect changes by using a probabilistic sensor noise model with the difference that during the reconstruction process a single RGB-D frame is compared to a stored 3D map. While processing incoming frames, the background map is updated as well as the 3D model for each detected object using a connected component algorithm. Once spatially close objects are merged, it is not possible to split them even when they are seen separated in the future.

Song et al. [15] reason that a robot usually stays in the same environment. Their goal is to determine global instance-based labeling and to further recognize individual objects. The method requires a high-quality reconstruction, which is based on the output of an RGB-D camera array. Based on the full semantic labeling generated by a crowd-sourcing marketplace, objects are either classified as movable or non-movable. Non-movable objects are considered as background and are used to align new frames with the reconstruction. For the remaining environment parts, a SIFT descriptor is computed and used for object matching. Objects that cannot be matched with already known ones are considered novel. Although manual labeling is required to achieve refined alignment, we categorize this method as unsupervised because the change detection process itself does not rely on specific training data. Furthermore, given recent advances in automatic semantic labeling such as, e. g., *SparseConvNet* [48] or *MinkowskiEngine* [49], the manual step could be simplified or replaced.

2.3 Learning-Based Object Detection Covering Unknown Objects

While comparing reconstructions of an environment from two different timestamps allows to discover unknown objects in 3D, learning-based approaches face the challenge of detecting objects not seen during the training phase. Based on this requirement, extracting 3D bounding boxes of objects including unknown ones recently gained attention in research. Category-level methods [50]–[53] are trained on pre-defined categories and applied to unseen instances from these categories. This is a first step in the right direction. However, the result depends on the selected training categories. Kollar et al. [54] and Kriegler et al. [55] train their models on synthetic stereo image datasets and apply them to real-world data. Both utilize only 3D geometric primitives and are therefore independent of 3D object models and semantic information of objects. Kriegler et al. state promising results for cuboid-shaped and cylindrical objects on table top-scenes, but the method needs improvement for objects that are not compact and can not be described with simple primitives, such as a pair of scissors.

The term `object detection` is not used consistently in the research community. For example, the famous architectures *YOLO* [56] and *Mask R-CNN* [57] are considered as

object detectors. Although, their aim is not only to find all objects and their boundaries in a 2D image but also to assign a class to each detected object. Considering 3D use cases the class label is a good indication when associating the 2D detections from several input images. Because *Mask R-CNN* is part of the baseline in Chapter 6, we give an overview of available learning-based object detectors including classification and how they extend to open-world settings. *YOLO* and *Mask R-CNN* [57] are very popular, yet limited to a closed world, and show weakness by assigning unknown objects mistakenly a learned class with high confidence [58]. While learning-based open-world object classification is an emerging research field [6], [7], [59], [60], the extension to open-world object detection is only recently defined by Joseph et al. [4] and consists of two tasks: (1) detect known objects as well as objects, which were not explicitly used for training and (2) incrementally learn these unknown objects. The first point, which is the aim of open-set object detection, is hard to achieve because unknown objects are categorized as background during training. Concerning the second step, incrementally extending the knowledge of a trained detector leads to the problem of catastrophic forgetting [61], [62], which describes the challenge of maintaining robust performance on known classes as new classes are learned. Before Joseph et al. defined the problem of open-world object detection, only two works existed that deal with estimating uncertainty and therefore being able to distinguish unknown objects [63], [64]. However, they are not capable of gradually extending their knowledge when new classes emerge, which is essential to be useful in real-world applications. Recently, triggered by the formal definition, open-set [5], [65]–[67] and open-world object detection [4], [68] are perceived as important, though challenging, tasks.

Kim et al. [5] present a neural network, which generates object proposals for unknown objects by comparing the location and shape of proposed regions with available annotated data. Compared to state-of-the-art object detection algorithms, performance for known objects dropped marginally. In the work of Li et al. [65] region proposals, generated with a Region Proposal Network (RPN), together with their corresponding feature maps are the input to a mask segmentation module on pixel-level predicting also the confidence. In a second step proposed object masks with a high uncertainty for all trained classes are classified as unknown otherwise as known. Applying this method, they reduce false positive detections generated by *Mask R-CNN* [57]. Du et al. [66] suggest exploiting spatial and temporal information from video streams to generate proposals for unknown objects. This is achieved by computing the dissimilarity of feature vectors from known objects and object candidates from different frames. They jointly train an object detector together with an uncertainty regularization branch. Cen et al. [67] transfer the problem of open-set object detection to the 3D space and apply their method to LIDAR data. Based on the fact that object detectors often wrongly assign known classes to unknown objects they extract objects with high uncertainty utilizing a metric-learning framework and Euclidean distance sum in metric space. For each of these unknown objects, a more precise bounding box is acquired by unsupervised clustering of points.

The open-world object detector proposed by Joseph et al. [4] uses an RPN to auto-label unknown objects by selecting regions with a high objectness provided that they do not overlap with an annotated ground truth object. Additionally, they use a binary

energy-based classifier on contrastive clustered data from the feature space. To be able to learn new classes without forgetting existing ones and without retraining from scratch, they utilize example replay and store a balanced set of examples to fine-tune the model. Gupta et al. [68] adapt an approach, which uses an end-to-end object detection model with transformers. They use an attention-driven approach to detect potential unknown objects. To be able to distinguish unknown objects from the background, they introduce an additional class besides the known classes, namely novel. Finally, they implement a foreground objectness branch to transfer knowledge from known to unknown objects and help to learn the model to differentiate between foreground and background. They use the same approach as [4] to countermeasure the problem of catastrophic forgetting. In their experiments they outperform the method of [4], however, the recall for unknown objects is very low with a maximum of 7.5.

Chapter 3

Datatsets

In this chapter, we list and describe available indoor datasets in the context of change detection (see Section 3.1). After careful consideration, we acquired a new robotic dataset fulfilling the requirements needed to evaluate the performance of the object change detection methods proposed in this thesis. A detailed description of this dataset is given in Section 3.2.

3.1 Existing Indoor Datasets

Only a few datasets are available that capture not only table-top scenes but full indoor environments at different timestamps. The characteristics of these datasets, which are applicable for change detection, are summarized and compared in Table 3.1.

Table 3.1: Comparison of indoor object discovery and change detection datasets (O=Office, H=Household, W=Warehouse).

	#Environments	#Recordings	Type	Real world	Robot	Reference map	Open world	Small objects	Annotation
Mason et al. [13] [1]	1 [2]	67	O	✓	✓	–	✓	✓	–
Ambrus et al. [46]	1	88	O	✓	✓	–	✓	✓	–[3]
Fehr et al. [16]	3	23	H/O	✓	–	✓	✓	–	–
Qian et al. [69]	1	18	W	✓	✓	✓	–	–	-
Halber et al. [70] - *Rescan*	13	45	H	✓	–	–	✓	–	✓
Wald et al. [24] - *3RScan*	478	1482	H	✓	–	–	✓	–	✓
Park et al. [44] - *ChangeSim*	10	80	W	–	✓	✓	✓	–	✓
Weihs et al. [45] - *RoomR*	120	6000	H	–	✓	✓	–	✓	✓
Ours - *ObChange*	5	31	H/O	✓	✓	✓	✓	✓	✓

[1] Rosbags available on request.
[2] Includes several rooms.
[3] Only partially and inconsistently annotated.

The dataset described in [13] is captured over a long time period using a mobile robot. The robot autonomously gathers data during frequent visits to an office environment consisting of several rooms, where change is naturally induced by the employees. Similarly, the dataset of Ambrus et al. [46] consists of data captured with a robot in a regular office, so it includes a large variety of objects and even people. However, the dataset only considers one room with the robot placed in the middle and not moving during acquisition. Only the RGB-D sensor is rotated on a pan/tilt unit. As a result, interesting regions and objects are far away from the camera, thus the views to create the reconstruction of the room are constrained. Fehr et al. [16] provide raw recordings from a handheld Google Tango for three different environments. The main focus of the work is not on object detection but on static environment recovery. Therefore, the dataset mainly contains furniture that moved between successive observations. Recently, Qian et al. [69] created a robotic warehouse dataset where fences and boxes move. It is intended for the evaluation of semi-static reconstruction methods. The datasets provided by Halber et al. [70] and Wald et al. [24] are captured with a handheld device. Each object in a recording has an instance ID to enable the inference of the movement between two timestamps. While the former provides voxel annotations the latter also provides the 3D transformations for moved objects. The dataset from Park et al.[44] consists of warehouse environments created by simulating a drone flight using a game engine.

Publicly available datasets differ not only in the type of environment they present, like office, household, or warehouse, but also in the number of different environments. While, e. g., Ambrus et al. [14] capture data from a single room only, Wald et al. [24] provide almost 500 different environments. As a downside, however, the latter is acquired with a handheld device and therefore leads to trajectories that would not be possible with a robot. In our opinion, only data captured by a robot is suitable to reliably test approaches developed specifically for robotic tasks.

In order to perform a quantitative evaluation of change detection approaches an according annotation of the test data is needed. Only the datasets introduced in [70], [24], [44], and [45] provide the necessary information to be suitable for evaluation purposes. Halber et al. [70] and Wald et al. [24] add semantic instance segmentation to 3D reconstructions to support tracking of objects between different timestamps. Since object matching can be ambiguous, both provide the necessary information to allow equally likely solutions. The dataset of Wald et al. [24] additionally includes 3D transformation for each moved object. *Rescan* [70] and *3RScan* [24] are both open-world datasets, meaning that objects are not only moved but also new ones are introduced or existing ones are removed between timestamps. In contrast, the data in *ChangeSim* from Park et al. [44] is annotated frame-wise. It is curated to support online detection approaches, which work directly on frames. For change detection, they define the following categories: new, removed, replace, rotated, or static. The most important category for object matching, moved, is not defined. This makes it impossible to differentiate between a removed or a moved object in an environment. Weihs et al. [45] introduce the new dataset *RoomR* with object rearrangements. It is based on the virtual and interactive environment AI2-THOR [71]. Assuming a closed world, they provide the start and goal poses of all objects while images are generated by AI2-THOR on-the-fly.

Datasets created in a simulated environment, like *ChangeSim* and *RoomR*, have the big advantage of automatically gathering precise annotations of a large number of environments as well as the possibility to control the amount of sensor noise, shadow, and other challenging factors. However, since closing the sim-to-real gap is an ongoing research area [72]–[74], we believe it is not enough to train and evaluate methods only on simulated data.

Unfortunately, except for *RoomR*, all annotated datasets focus on bigger entities like furniture and therefore lack the annotation of smaller objects like a mug. On the one hand, this is very cumbersome for real-world data, on the other hand, available tools are not suitable for fine-grained annotations. Wald et al. [24], for example, use the annotation tool provided by Dai et al. [75], where a pre-segmentation step of the environment usually results in under-segmentation making it impossible to annotate smaller objects.

The highlighted problems with existing datasets were the reason for us to create a new robotic indoor dataset that also includes the annotation of small objects. The following section explains the dataset in more detail.

3.2 Object Change Detection Dataset for Indoor Environments

For different robotic applications, it is highly relevant to collect knowledge about small objects, such as a mug or a hammer, as well as track them over time as they naturally move more frequently than, for example, a couch. However, as visible in Table 3.1 the aspect of annotated small objects is underrepresented in the available change detection datasets and not present at all in datasets captured in the real world. Therefore, we provide a new dataset focusing on small objects, which is gathered in an open-world environment with a real robot. This dataset is suitable for quantitatively evaluating object change detection methods as well as methods developed for object matching, which categorize objects into static, moved, removed, or novel when comparing an environment at two different timestamps.

To be able to provide real robotic data, we used the Human Support Robot from Toyota [76]. It is equipped with an Asus RGB-D camera mounted on the head. Figure 3.1 shows the robot observing objects on a table. The camera stream, consisting of RGB and depth data, is stored together with the transformation matrices between coordinate frames.

To build a dataset focusing on object change detection it is necessary to visit and explore an environment at several different timestamps. A set of waypoints is defined so that the robot explores the environment exhaustively when navigating from one waypoint to the next. Between the visits, the general structure of the environment remains the same, but objects may move or disappear or new ones are added. Furthermore, furniture and permanent background objects may be slightly rearranged. These rearrangements are small so that the moved furniture does not interfere with the robot's navigation and that the moved objects are considered irrelevant in a tidy-up task.

The dataset comprises five different environments, namely a big room, a small room,

Figure 3.1: Human Support Robot from Toyota with the RGB-D camera highlighted.

a living area, an office desk, and a kitchen counter. Each environment is captured in a clean and tidied-up state, which we refer to as the reference. Figure 3.3 and Figure 3.4 show these references for all five environments as reconstructions created with *voxblox* [27]. Additionally, the robot visits each environment five or six times while between each visit a subset of objects from the YCB dataset [77] is rearranged. Objects of diverse sizes ranging from small, such as a screwdriver, to large, such as a plastic water pitcher, are selected. The collection of objects used in the dataset is shown in Figure 3.2. Every time before an environment is explored by the robot, between 3 and 18 objects are selected and placed at various locations. In summary, the dataset consists of five different environments, 31 observations (including the reference), and 260 arranged YCB objects in total.

Based on the recorded data, we generated two datasets consisting of 3D reconstructions with the YCB objects labeled. The main distinctions between the two datasets are the different reconstruction methods and the scale of the reconstructions.

The dataset used in Chapter 5 comprises full-environment reconstructions. *Voxblox* was the reconstruction method of choice. We found in an empirical evaluation that

Figure 3.2: Overview of YCB objects used for our dataset.

this is the only method that is able to generate realistic reconstructions using the camera data from the robot together with the transformation matrices. One drawback of the generated full-environment reconstructions is the low level of detail. This dataset is publicly available at `https://www.acin.tuwien.ac.at/en/vision-for-robotics/software-tools/object-change-detection-dataset-of-indoor-environments/`.

The second dataset is a collection of reconstructions of interesting areas. Areas in an environment are selected if they contain a potential object support plane (excluding the floor). To generate high-quality reconstructions we extend *ElasticFusion* [28] by integrating odometry data. Section 6.3.1.1 gives more details regarding how the areas of interest are selected and reconstructed. The dataset, which we call *ObChange* (*Ob*ject *Change*), is available at `https://doi.org/10.48436/y3ggy-hxp10`.

Both datasets are annotated using the same approach comprising automatic and manual labeling. First, the publicly available 3D object models of the YCB objects are roughly aligned in the reconstruction. Next, points within a small distance threshold to the model points are automatically selected using a k-d tree. As a final step, single points are manually added to or removed from the object masks. This is necessary because most objects are usually not reconstructed precisely.

A subset of reconstructions from both datasets is visualized in the appendix.

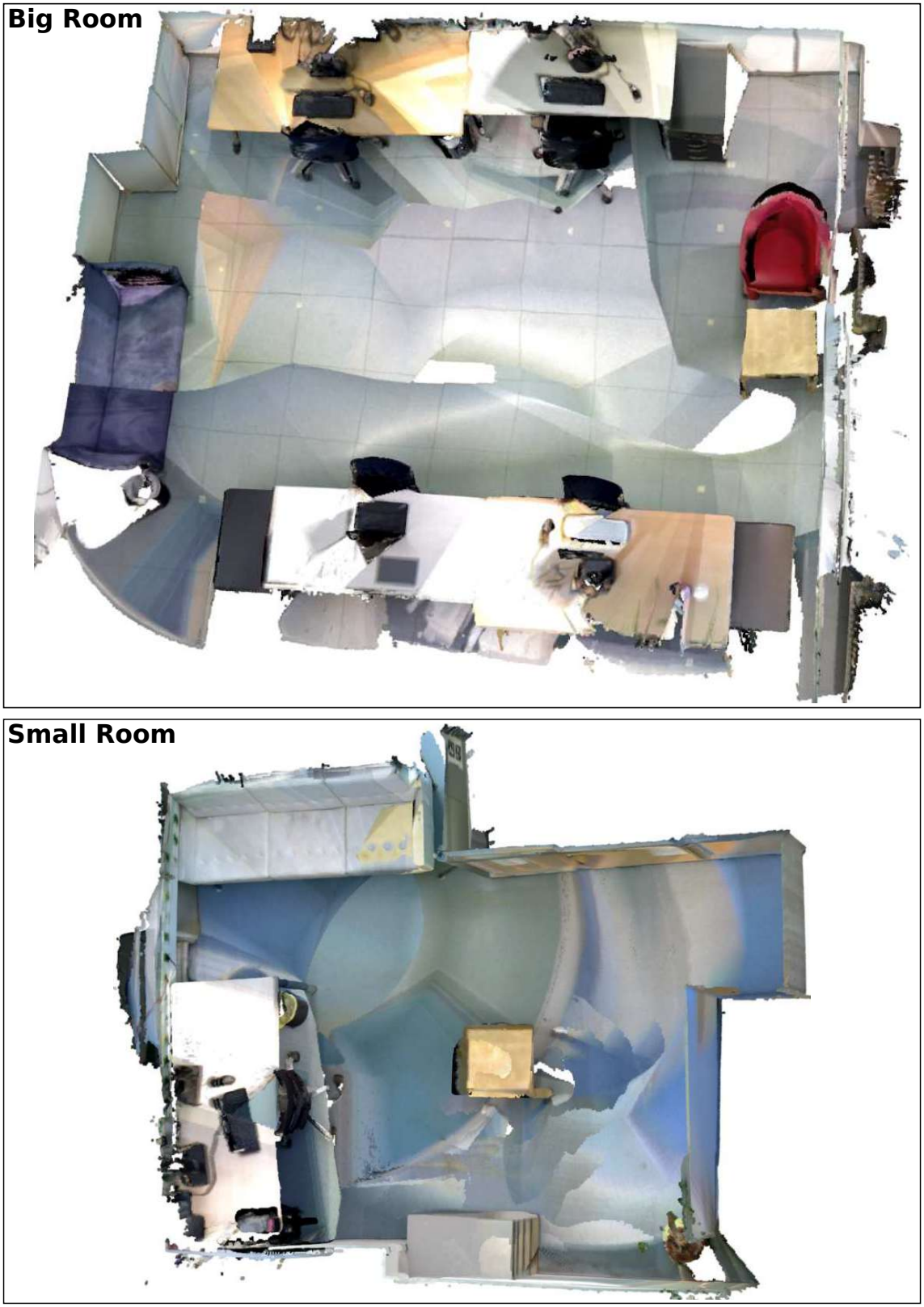


Figure 3.3: Reconstructions for the big and small room showing the reference state.

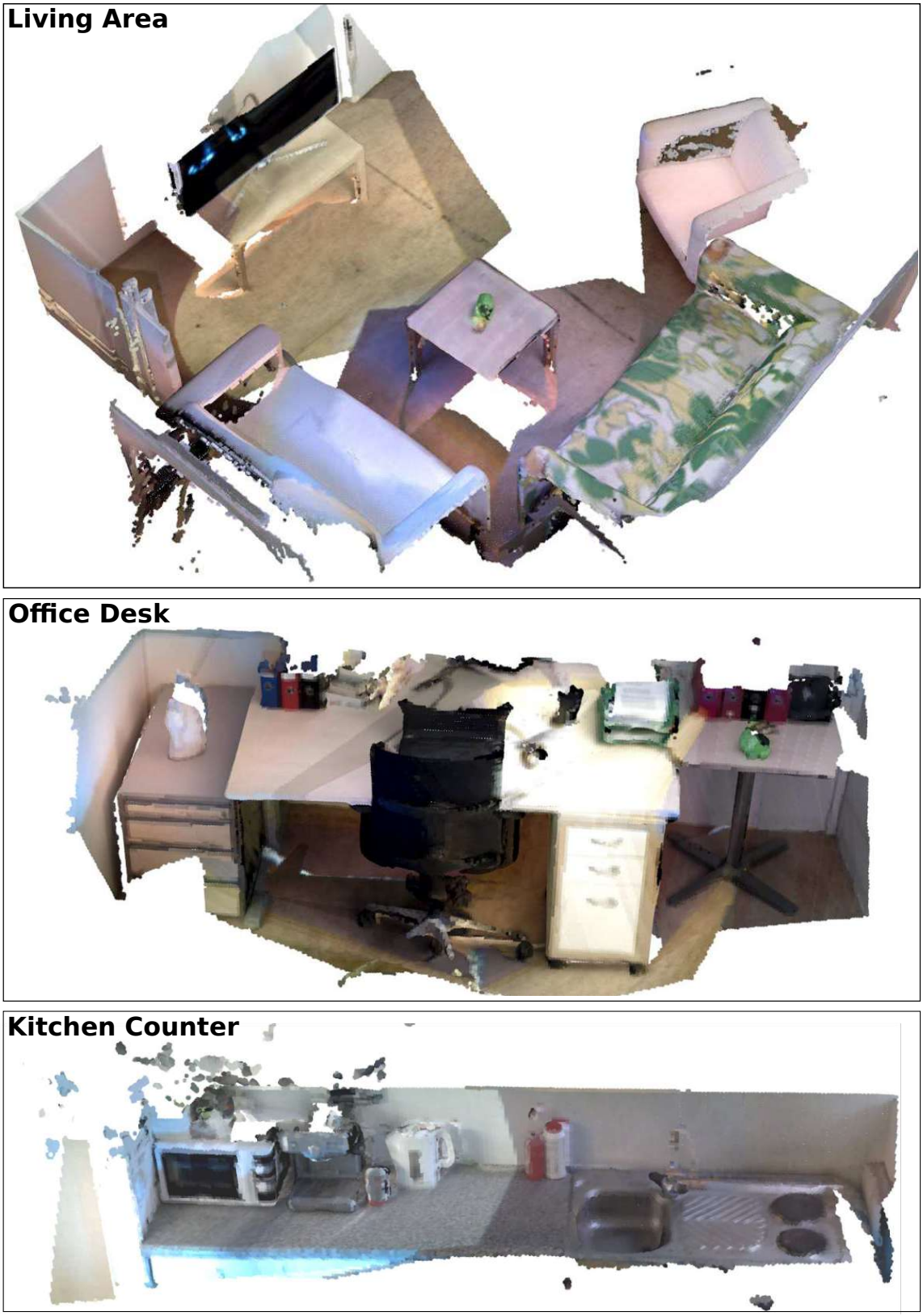


Figure 3.4: Reconstructions for the living area, office desk, and kitchen counter showing the reference state.

Chapter 4

On-the-Fly Detection of Novel Objects on the Floor

4.1 Motivation

Consider the scenario where a robot is given the task to tidy a child's room. The robot must find items lying strewn on the floor, recognize these items and then put them in the places where they belong. This task is also of interest in other applications where floors should be free of objects. For example, in elderly care, objects on the floor may cause a person to fall. Therefore, it is important to detect obstacles and remove them in order to reduce the risk of falling. In these scenarios, the first step is to search for and detect objects on the floor. An example of such a scenario of cluttered toys is shown in Figure 4.1. In this work, we focus on detecting novel objects, which lie on the floor. However, the approach is easy to adapt so that it is applicable for other tasks, which involve, e.g., objects on tables as well (see Chapter 5 and[43]).

We propose a method to detect objects, which are at a location that was unoccupied in a reference and call them novel objects. The method compares two *OctoMaps* depicting the same environment but captured at different points in time. An *OctoMap* [26] is a 3D voxel grid representing the environment and is used primarily for navigation. It is updated while the robot is moving around utilizing depth data from the sensor. While other works [13], [39] find differences between the current and a previous state for a

Figure 4.1: Typical scenes for a robot that has to clear away toys.

specific camera pose, this method aims to find novel objects in parts of the environment that the robot has previously visited and independent of the robot's current position. This means that while comparing *OctoMaps*, the robot is not required to be near a potential object location to be able to detect it. We refer to these areas of change as *regions of interest*, which represent one or several object candidates. Consequently, the aim of this work is not to find specific objects, but any object that is newly introduced at a previously unoccupied location. Objects and their appearance are not required to be known by the robot in advance. The proposed method is independent of pre-learned object models and applicable to open-world settings.

This approach for on-the-fly detection of regions of interest during navigation allows a robot to be more aware of changes in its surroundings and respond faster to these changes. The robot detects regions of interest during an ongoing task and returns to them when requested by the planning module. Knowing the position of potentially interesting regions enables a robot to reliably find and segment objects more efficiently. Having a region of interest in the field of view, it is not necessary to segment the whole frame, but use the identified region as an attention cue. This is more efficient than explicitly visiting waypoints and segmenting the whole environment regardless of how likely it is that objects are present.

To evaluate the performance of the proposed approach we use a mobile robot platform with an omnidirectional base, a navigation system based on a laser sensor, and an active pan-tilt head with an RGB-D sensor (Asus Xtion Pro LIVE). We show that our approach outperforms a fixed waypoint setup in terms of runtime and the number of segmentation actions required to find all novel objects in the environment. Experiments are performed in simulation as well as with a real robot. Different experimental setups are performed ten times each to capture realistic variation. In total, results from 160 (80 simulated, 80 real) experiments are analyzed.

To summarize, the contributions of this work include

- an efficient approach to search for novel objects by exploiting *OctoMaps*,
- on-the-fly detection of object candidates while the robot is carrying out a task and navigates through the environment,
- *OctoMap* comparisons at any time, regardless of the robot's current position, as object candidates are inherently stored in the current *OctoMap*,
- an attention cue to achieve robust and fast segmentation of detected objects in the environment.

The following sections describe the novel object detection approach in more detail, specify the conducted experiments, and discuss consequent results. The content of this chapter follows previously published work in [41].

4.2 Method

Figure 4.2 gives an overview of the proposed method to detect novel objects. In this work, we describe the method on the basis of the task of exhaustively exploring an

environment. The robot stores a reference *OctoMap* depicting the static parts of the environment, which is created during a one-time mapping run. To find novel objects not contained in the reference, a set of view triangles, which covers most of the environment floor is generated. Subsequently, a plan is created to visit these view triangles and the robot executes this plan. During execution, the current *OctoMap* is updated while the robot moves around. Whenever the planner requests a comparison, the reference *OctoMap* is subtracted from the current one. The remaining voxels are checked against the dilated 2D occupancy grid used for navigation to overcome small misalignments. Voxels are then clustered into regions of interest, which are filtered and the remaining ones are stored in a database. The planning system receives these regions and uses them to improve its current plan. Details of these steps are given in the following sections.

4.2.1 Generation of Reference *OctoMap*

As an initial step, we build an *OctoMap*, which captures the static parts of the environment. Subsequently, it is used as a reference, against which the current state of the environment is compared. It is important that the reference *OctoMap* depicts the environment very precisely. *OctoMaps* are generated by using measurements from an RGB-D sensor, which is known to be noisy [78]. Spurious measurements from the acquired *OctoMap* are manually deleted using *octovis*, a viewer that is provided by the developers of the *OctoMap* library. To make our approach more robust against discretization effects, the *OctoMap* is dilated by iterating over its voxels and setting each neighbor of an occupied voxel to occupied. We refer to the dilated *OctoMap* as the reference *OctoMap*. This map is stored and used for subsequent searches. An example of a dilated *OctoMap* is depicted in Figure 4.2 (top left) and in Figure 4.4a. Note, that the generation of the reference *OctoMap* has to be performed only once as long as the static parts of the environment do not change.

4.2.2 Generation of Environment Exploration Poses

To explore an environment completely the RGB-D sensor must sense every part of it. This task shows the characteristics of a set cover problem. Given a set of V possible sensing locations, each with a fixed sensing region, we want to select a small subset $V' \subseteq V$ that maximizes the observed area. The problem presented in this paper satisfies submodularity, which is the property of diminishing returns. When a small portion of the area has been observed a new observation can yield more information, however making a new observation when most of the environment has already been observed will yield less information.

We approximate the area of the floor, which is seen by the robot's camera at a specific pose, as a triangle called the *view triangle*. The area covered is a function of the field of view of the RGB-D sensor (tr_{angle}) and a cut-off distance (tr_{height}) beyond which the measurement becomes too noisy. Additionally, the 2D occupancy map used for navigation is exploited to check if obstacles potentially block the view. Given that the environment exploration problem is submodular, we use a greedy algorithm to determine where to sense and find a near-optimal solution [79]. However, we have not

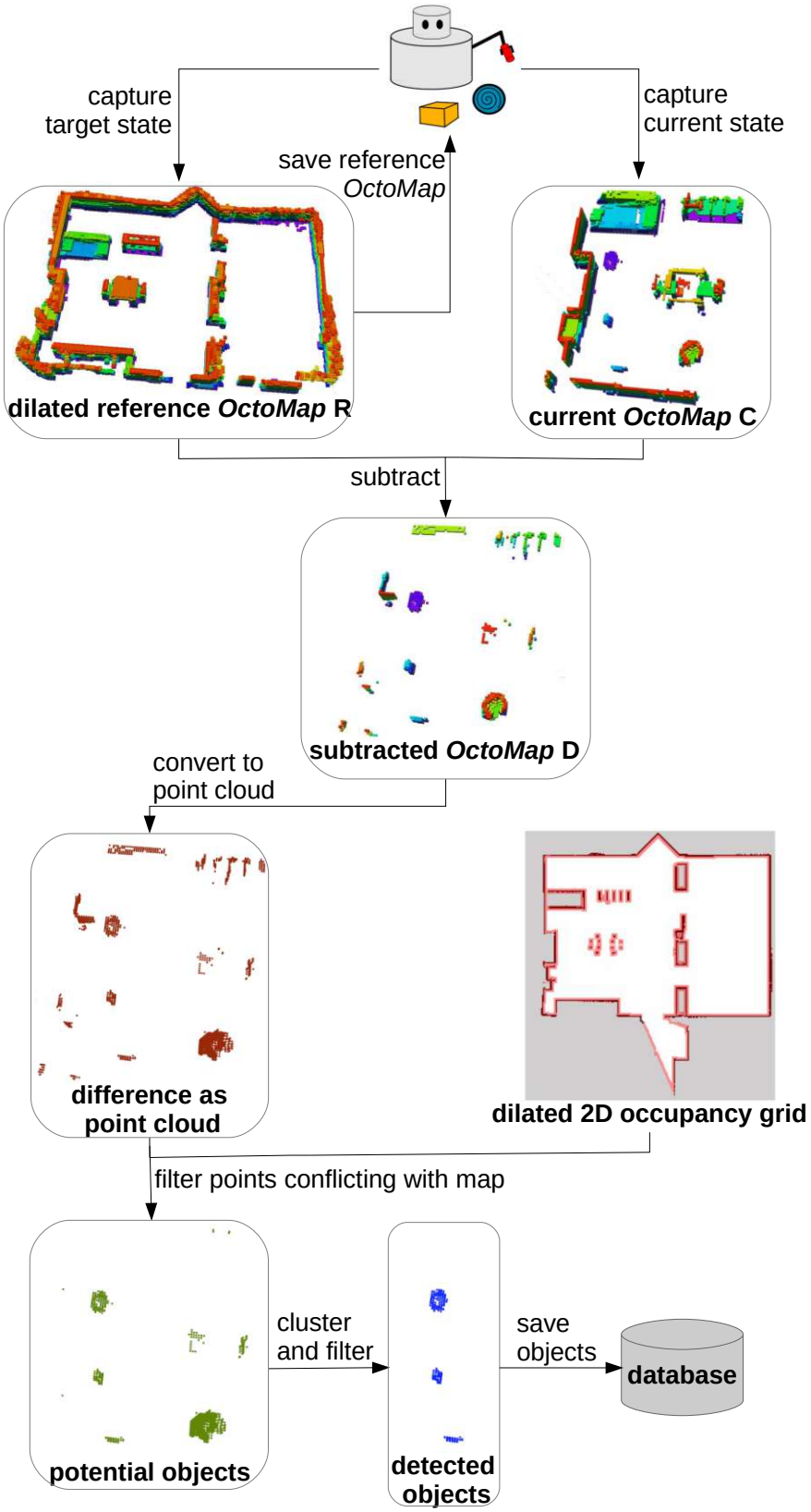


Figure 4.2: Flowchart showing the method parts described in Section 4.2.1, 4.2.4 and 4.2.5.

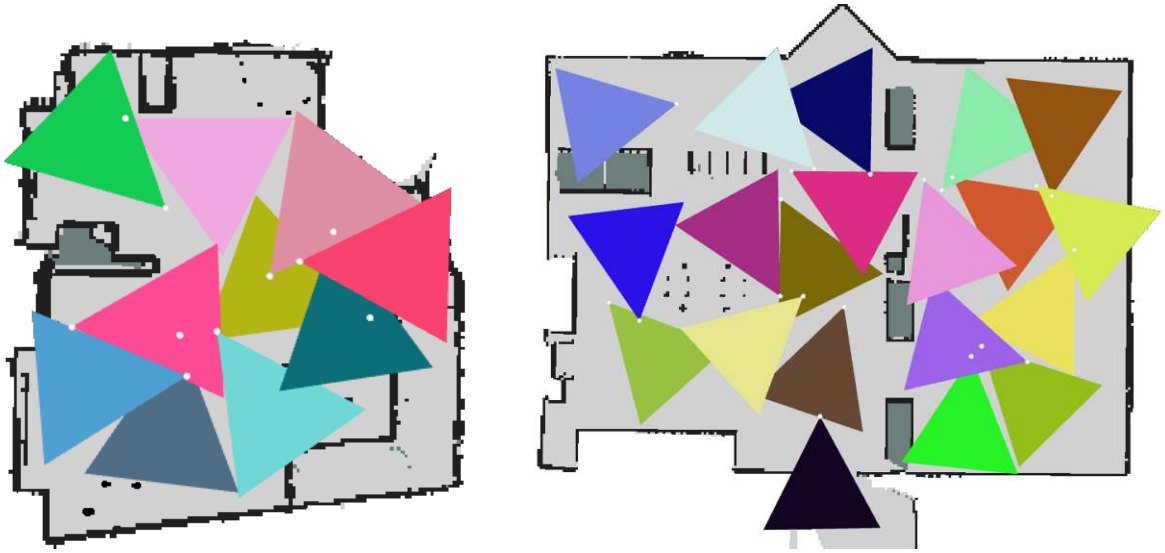


Figure 4.3: Examples of the result of the view triangle generation. Left: Environment used for experiments with real robot. Right: Environment used for experiments in the simulator.

yet identified where the robot can sense. Given that it is an unstructured area we use a random sampling approach. We randomly sample 1000 view triangles and use a greedy algorithm to find a set of exploration poses that covers most of the floor of the environment.

We want the set of view triangles to be small as finding an optimal route to visit all these view triangles is a traveling salesman problem which is NP-hard. While we use view triangles to navigate through the space, it is important to note that we do not perform segmentation there. Other object detection approaches in contrast could use each of these view triangles as a pose to carry out object segmentation actions. As mentioned before this may take an unnecessarily long time, as most of them will not contain an object.

We store the view triangles in the *Knowledge Base* (KB) so that the planning system (see next section) has the necessary information to find an initial search plan. Figure 4.3 shows examples of environments, which are used for our experiments, covered by view triangles. The angle of the view triangles corresponds to the horizontal viewing angle of the camera and the height of the view triangles depicts the viewing distance of the tilted camera.

4.2.3 Robot Motion Planning

We use a planning system to solve the problem of exploring the environment, finding regions of interest, and ultimately inspecting all found regions of interest. We use the domain-independent planner FF [80] to solve these problems. Plans are generated by providing a Planning Domain Definition Language (PDDL) [81] domain and problem file. The domain contains actions that can be performed to affect the world. In our case it contains the actions the robot can perform: goto_waypoint, explore_waypoint,

etc. Each action has a set of preconditions that must be satisfied before the action can be executed and an effect that is applied when the action is executed. For example, the *(goto_waypoint robot wp1 wp2)* action requires the robot to be at *wp1* and when the action is executed the robot is no longer at *wp1* but is at *wp2*. The problem file contains the initial state and goal we want to reach. The initial state contains the current location of the robot, the locations of the view triangles, identified regions of interest, and the connectivity between them. The goal is to have visited all the view triangles and inspected all regions of interest.

We use the ROS module ROSPlan [82] to construct the PDDL files, find a plan, and use this plan to autonomously control the robot. ROSPlan contains a KB that 1) stores data in a database and 2) maps symbols to data. Plans are generated by providing a PDDL domain and populating the KB with predicates that describe the world. ROSPlan generates a problem file and solves the resulting planning problem. This plan is converted into an Esterel program [83] and dispatched.

Dispatched actions are handled by ROS modules that listen to the dispatched actions. For example, ROSPlan contains a ROS module that is an interface for MoveBase. It listens to 'goto_waypoint' actions and maps the action parameters to 3D coordinates stored in the KB. It then calls the MoveBase action server with these coordinates. The feedback of this action server is communicated to ROSPlan to monitor the execution of its plan. If an action fails (e.g., MoveBase cannot reach the desired location) it triggers a replan and dispatches the resulting new plan.

Initially, we take the set of generated view triangles (see 4.2.2) and create a plan to visit them all. During the execution of this plan, we discard view triangles covering an area, which was already mapped on-the-fly. Subsequently, we create a new plan to inspect all regions of interest found from *OctoMap* differencing.

This planning process is an iterative process. When we have inspected all regions of interest we replan by generating a new set of view triangles and find a plan to visit these and subsequently inspect newly found regions of interest. This process continues until a predefined termination condition is satisfied. This could be having found a certain number of objects, a certain percentage of area explored, or a time limit.

4.2.4 *OctoMap* Differencing and Region Detection

While navigating between view triangles an *OctoMap* is built and subsequently used to find regions of interest by comparing it to the reference *OctoMap*. We define the difference D, between the current *OctoMap* C and the reference *OctoMap* R as

$$D = \{C \setminus R\} = \{c \in C \mid c \notin R\}, \tag{4.1}$$

in other words, D contains only voxels that are in C but not in R. The two *OctoMaps*, C and R, are built based on the robot's localization and are therefore in the same coordinate frame and automatically aligned.

Despite generally good alignment between the 2D occupancy grid (constructed using a laser ranger scanner) and the *OctoMap*, we noticed that the laser-based map is more accurate. This is probably due to the increase in depth measurement errors of the

RGB-D sensor at large distances. Furthermore, the *OctoMap* accumulates spurious measurements when localization temporarily fails and as a result, for example, new voxels appear behind a wall. This cannot be handled with the dilated *OctoMap*. Therefore we compare D to the 2D occupancy grid to reduce the effects of noisy measurements. We dilate the occupied pixels of the 2D occupancy grid by a small, 5×5 pixels, rectangle to be more robust. To be able to compare the *OctoMap* D with the 2D occupancy grid, each voxel is projected vertically to the x-y plane. This approach removes voxels that lie outside the map or have the same position as an occupied grid cell in the 2D occupancy grid. Figure 4.2 contains an example of a dilated occupancy grid with gray being pixels outside of the map, white being the free space, black depicting the occupied pixels, and red the dilated area.

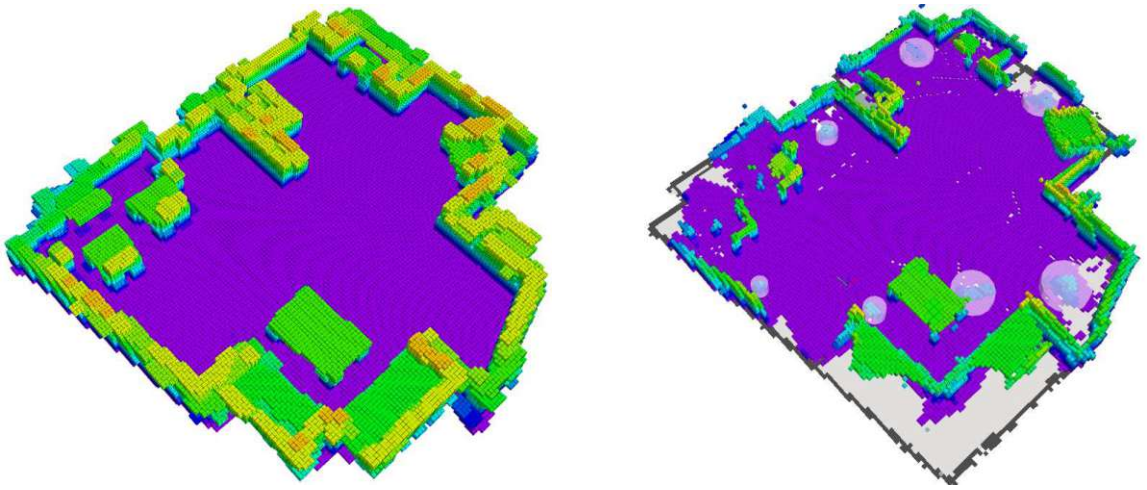
The next step of our algorithm is to cluster the remaining voxels based on their Euclidean distance. This allows us to filter out clusters that are likely to contain objects, which are not relevant to the robot's task. Suppose the task is to find toys scattered on the floor and to eventually pick them up and place them at their respective stow-away location. Clusters that are too small and contain less than c_{min} voxels are considered noise. Very large clusters with more than c_{max} voxels may correspond to moved furniture, which is too heavy for the robot to move. Furthermore, we filter clusters hovering above the floor and clusters that are taller than c_{height}. In Figure 4.2, for example, the cluster on the bottom right was removed because it is too tall and the robot would not be able to manipulate it. Also, the cluster representing the object placed on the table (visible in red in the current *OctoMap*) was removed because it is not reachable by the robot. However, those filter rules can be adjusted according to the robot's task. Objects placed on the table may be of interest to other robots accomplishing a different task. Each remaining cluster fulfilling the requirements represents a potentially interesting region. Figure 4.4b shows the *OctoMap* from one of our experiments with the real robot. This *OctoMap* was compared with the dilated version of the *OctoMap* in Figure 4.4a. The seven white cylinders depict the bounding cylinders of the detected regions of interest.

The resolution *res* of the *OctoMap* determines the size of objects that can be found. Using a high resolution results in a detailed map, but requires a very low-noise sensor and precise mapping. On the other hand, defining a low resolution prevents the method from finding small objects. The points representing these objects would become fused with the supporting plane points into one voxel. Also, the dilation of the *OctoMap* to reduce the effect of measurement noise hinders the algorithm to detect small objects close to static parts of the environment, like walls. A suitable resolution for the *OctoMap* depends therefore on the sensor noise and the desired detail of the *OctoMap*.

4.2.5 Storing Discovered Regions of Interest

The planner needs to know about possible regions of interest. We make this information available by storing identified regions in the KB. For each region we store the following attributes:

- *Timestamp*: This is the time when the respective region was detected by *OctoMap* differencing.



(a) Reference *OctoMap* R depicting the static parts of our test environment.

(b) Current *OctoMap* C with toys lying on the floor.

Figure 4.4: Reference *OctoMap* R and current *OctoMap* C depicting the environment used in our real-world experiments. The white cylinders in C indicate the clusters identified by our method.

- *Position*: We compute the position in map coordinates by extracting the maximum spatial extent in all three major directions and calculating the center.
- *Bounding cylinder*: We use a cylinder to describe the outline of the object.
- *Examined flag*: This flag is set to true once the robot has examined the region.

Every time the planner dispatches the explore_waypoint action the current *OctoMap* is compared with the reference and newly detected regions are checked against those already stored in the KB. New regions are inserted, already stored regions are updated if they have been growing in size (because they are seen again from a different view), and stored regions that are not confirmed are removed.

4.2.6 Examining Regions of Interest

During the execution of the initial plan, which explores the generated view triangles (see Section 4.2.3), all discovered regions of interest are stored in the KB. The next step is to create a plan to examine these regions. For each region of interest, we generate a set of potential viewpoints. To compute these viewpoints, we query the KB. Based on the position and size of the region of interest, we determine viewpoints from where the robot is able to segment it. Depending on the robot's size and whether it has a pan-tilt unit, the optimal distance to view an object varies. We define a minimum distance $exam_{dist}$ between the robot center and the object boundary to ensure that the robot base is not visible in the camera image. Ideally, equally distributed viewpoints with a step size $exam_{step}$ and the same distance to the object are created around it. Our algorithm compares each of these viewpoints against the *OctoMap*. If it is not reachable

or something is in the line of sight between the robot and the object, the algorithm attempts to find a new suitable viewpoint that is nearby, otherwise, it is ruled out. The generated viewpoints consist of a pose in combination with a suitable tilt angle to view the object.

4.2.7 Object Segmentation

After moving to one of the computed viewpoints, the robot has the previously identified region of interest in the field of view. The currently sensed point cloud is cropped according to the dimensions of the bounding cylinder of the respective region of interest including a margin μ in each direction. This has two advantages:

- The time to segment the point cloud decreases.
- Focusing only on the region of interest prevents segmentation to bleed out into static parts of an environment, like a wall or a table leg.

Based on the cropped point cloud, we estimate the ground plane using RANSAC [84] and remove the points associated with the plane. The remaining point cloud is used to extract clusters based on proximity, smoothness (dot product of normals), and the color difference between two points. An example of the proposed segmentation pipeline is shown in Figure 4.5.

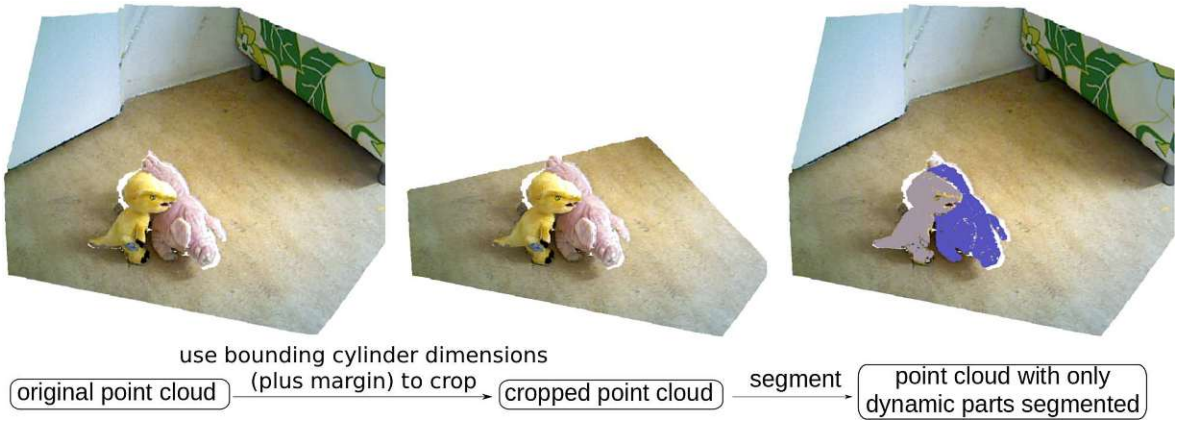


Figure 4.5: Flow diagram of the proposed segmentation approach performed at a computed viewpoint.

4.3 Experimental Results

In order to evaluate the effectiveness of our approach, we integrate our implementation with an existing system running on a mobile robot platform.

The robot's goal in our experiments is to find a given number of objects distributed on the floor in the environment. This means the robot will continue searching (generating regions of interest and performing segmentation) and replanning until all objects are

Table 4.1: Parameters used for the experiments.

Method	Parameter	Value
OctoMap	resolution res	0.05 m
View triangle generation	viewing distance/triangle height tr_{height}	2.0 m
View triangle generation	field of view tr_{angle}	58°
Cluster filtering	min. cluster size c_{min}	4 voxels
Cluster filtering	max. cluster size c_{max}	1000 voxels
Cluster filtering	max. cluster height c_{height}	0.5 m
Examination waypoint	min. dist. between object and robot $exam_{dist}$	0.4 m
Examination waypoint	uniform step size $exam_{step}$	60°
Object segmentation	crop margin μ	0.15 m

found. As we noted in Section 4.2.3, a different termination criterion could be used. We have chosen a fixed number of objects because it allows us to scale the problem by varying the number of objects to find. The location of the segmented objects is then compared to the ground truth positions of the objects.

We compare our approach against segmenting at each created view triangle, which is similar to the methods using fixed waypoints, such as [39] and [13]. We use simple environments with limited clutter to allow repeatable tests because the segmentation method occasionally over-segments cluttered scenes.

The search space for novel objects is restricted to the floor because of the robot's design. The camera is mounted quite low, which makes it impossible to see onto a table and the robot would not be able to grasp things from a table. However, with a different robot, the provided approach is applicable to other setups as well. Novel objects could also be detected on tables or on shelves when adjusting the filtering rules as mentioned in Section 4.2.4. For all the experiments in this work, we use the parameter values listed in Table 4.1.

We perform experiments with a mobile robot in simulation as well as in a real environment. The simulated and real environments have a size of approximately 90 m^2 and 30 m^2. They contain, among other items: sofas, chairs, desks, and cupboards. We perform our tests with one, three, five, and seven novel objects in the environment. 20 view triangles are generated for the simulated environment and ten view triangles for the real environment. We execute each experimental setup ten times. The object locations do not change within one batch, however, the view triangle generation contains a random component and thus returns different results for each test run. In total, we conduct 160 experiments: two methods, two environments, four numbers of objects placed in the environments, and ten runs for each setup.

4.7 % of the detected regions of interest are false positives when running the experiments with the real robot. This is the effect of localization errors and depth measurement noise. Ideally, this gets resolved by the robot moving to the wrongly detected regions and examining them to realize that there is no object of interest. The examined flag gets set to true and the planner moves on. We do not experience any false positives in the simulation, as it has a perfectly calibrated system. In the following discussion, we refer to our proposed approach as *dynamic method* and name the method

that segments at each view triangle *static method*.

4.3.1 Real environment

Figure 4.6 shows the number of segmentation actions needed to find all the distributed objects in the real environment. Our approach almost always needs as many segmentation actions as there are objects because it first finds regions of interest and use them as cue for potential objects. The static method is very fortunate in the experiments with one object. In more than half of the cases, it finds the object with the very first segmentation action. For the other experiments with more objects the static method needs on average three times as many segmentation actions as the number of objects. Furthermore, the variance for the static method is much higher than for our method. Our approach clearly outperforms the static method when it comes to the number of segmentation actions. This is advantageous when segmentation is computationally expensive. Also, the number of segments not corresponding to objects of interest reduces due to only segmenting relevant areas.

Segmenting a frame on the robot takes on average 9 s for the static method compared to 5 s for the dynamic method. The reason for the speedup is that the computed bounding cylinders are used as attention cues, allowing the point cloud to be reduced to contain only relevant parts.

Note, that although the segmentation method is actually quite simple, it takes considerably longer than most reported state-of-the-art methods. The reason for this is the limited computing power on the robot (i5 CPU at 2.4 GHz) in combination with the high load due to the many modules running on the robot. We choose not to tune the processes (turning off computationally costly components etc.), as this is precisely the type of situation encountered by a real robot: limited resources requiring efficient management methods.

Figure 4.7 shows the overall time the robot needs to find all objects. The original times for the static and dynamic methods are shown in light green and pink. Similar to the number of segmentation actions, the measured time needed to find the objects is also very widespread for the static method. The experimental setup with one object is handled slightly better by the static method when comparing the median time. This is because the object is found with the first segmentation action in six out of ten runs. However, on average, the dynamic method completes the task faster. For the other setups where more objects are scattered on the floor, our method achieves superior results.

Since the static method needs to segment more often, the run time of the segmentation process plays an important role. Therefore, we recompute the overall time with a more realistic and state-of-the-art assumption. We assume a real-time performance of 40 ms for segmenting the whole frame as stated by Ückermann et al. [85] and 20 ms for segmenting the cropped frame. The results are shown in Figure 4.7 by the blue and violet boxplots. Still, our method clearly outperforms the static method. For example, finding seven objects in the environment is on average twenty minutes faster.

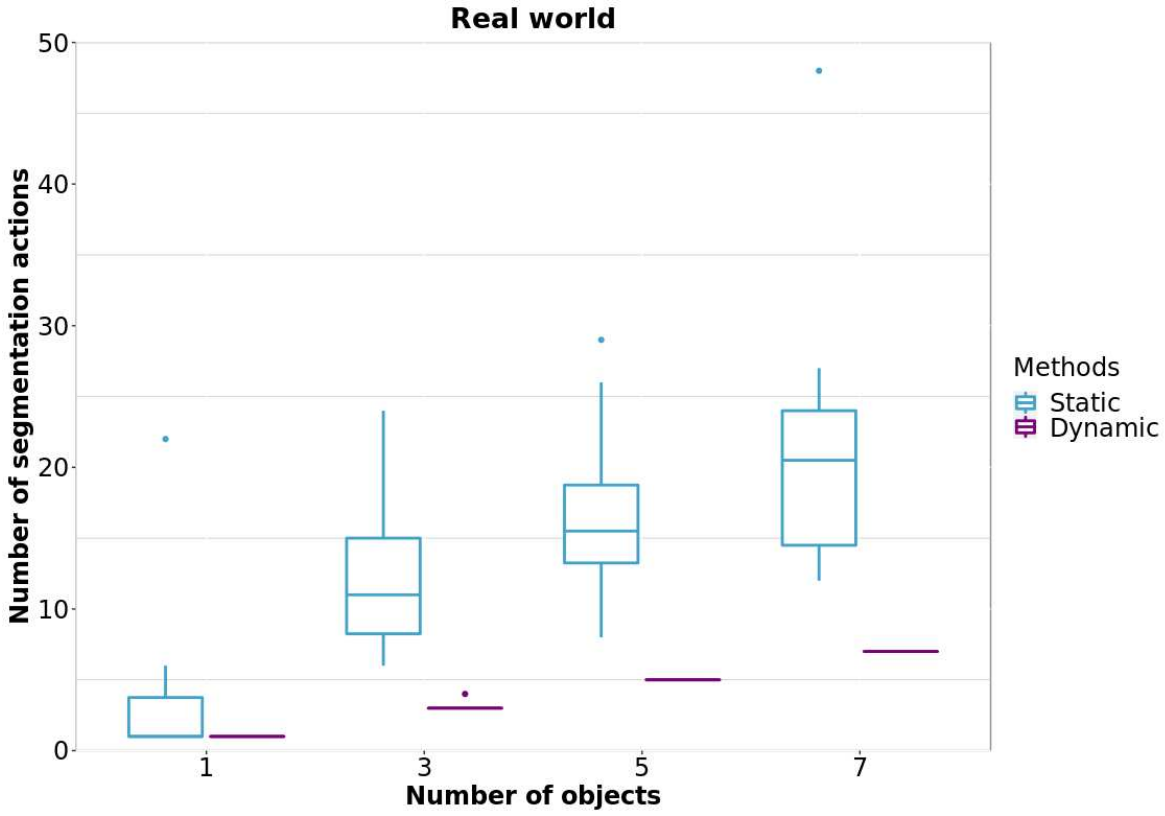


Figure 4.6: Comparison of the number of segmentation actions needed to find all objects in the real environment.

4.3.2 Simulated environment

We also run the experiments in a simulated environment using *Gazebo* as simulator [86]. The area in the simulation is three times larger than the area used for experiments with the real robot. In this larger environment, travel distance plays a more important role. Therefore, we changed the order in which the regions of interest are explored. The robot in the real environment uses the timestamp of the regions. After finding the given number of regions of interest, the robot explores the regions in the order they were found. This is not optimal and increases the travel time of the robot significantly when it travels back and forth between regions in a large environment. Therefore, we reduce the travel time for exploring all regions by implementing a greedy traveling salesman problem solution over the locations of regions. We also take advantage of the fact that the robot knows where it has already inspected the environment. Before moving to a new view triangle we check the area spanned by the view triangle against the current *OctoMap*. If more than 90 % of the view triangle area is already covered by the *OctoMap*, the planner does not add the view triangle to the plan. The robot already explored that part of the environment and most likely would not acquire new information.

Figure 4.8 and Figure 4.9 show that our method again outperforms the static method. The results for the static method present again a high variance for both

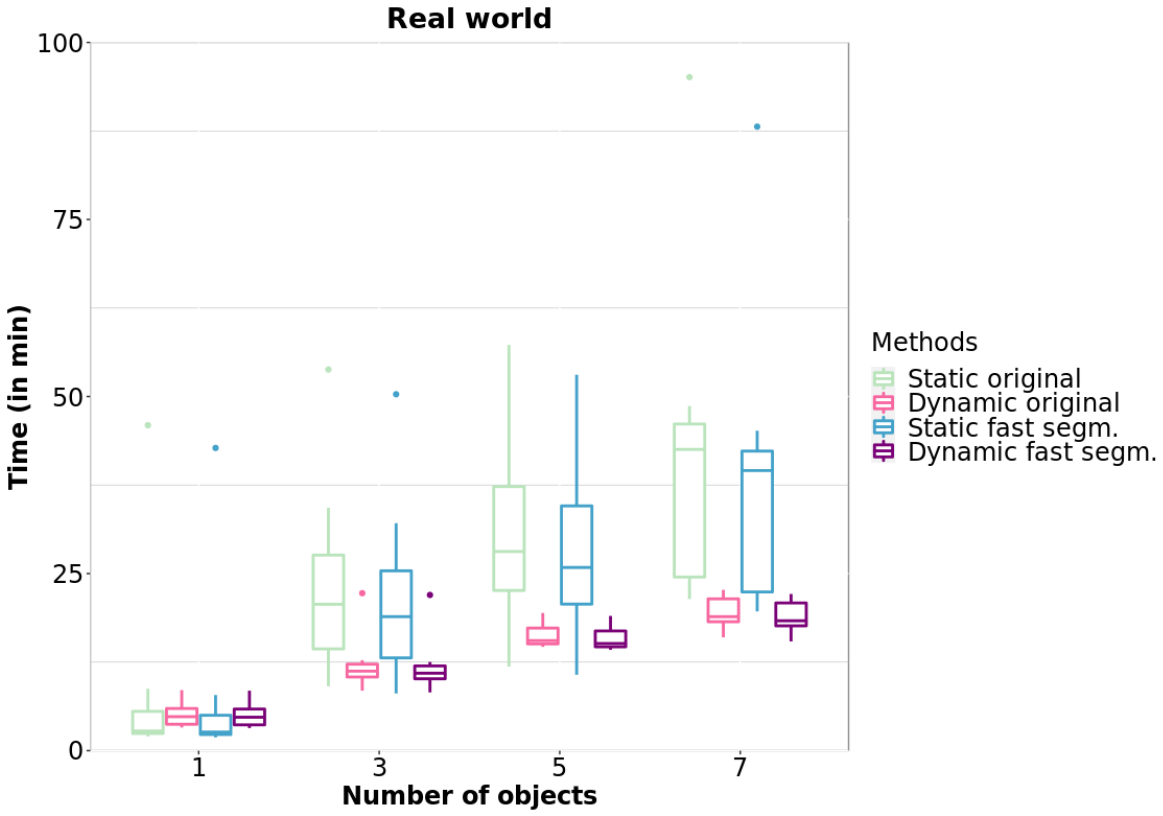


Figure 4.7: Comparison of the time needed to find all objects in the real environment.

measurement criteria. In simulation segmenting the whole frame takes approximately 6 s and segmenting the cropped frame takes approximately 1 s. We again recalculate the overall search times assuming a segmentation time of 40 ms for the whole frame and 20 ms for the cropped one. The recalculated times are shown in blue and violet.

To sum up, the robot's performance greatly benefits from using regions of interest. The time to find regions of interest by *OctoMap* comparisons is negligibly small (less than 800 ms for 90 m^2) compared to the otherwise necessary travel time to visit all the view triangles, which are needed to find all objects as in the static method.

4.4 Conclusion

In this chapter, we proposed a method to detect novel objects on the floor in an indoor environment by exploiting *OctoMaps* initially generated for navigation. Our method does not rely on fixed waypoints but uses arbitrary view triangles to guide the robot through the environment. The view triangle generation may be omitted if the robot moves around while pursuing a plan and therefore maps the environment on-the-fly. By comparing the currently mapped environment to a reference *OctoMap*, the proposed method reliably finds regions of interest, which contain one or several novel objects. Taking the dimensions of these regions into account enables segmentation to perform on cropped point clouds and, consequently, this improves the processing time notably.

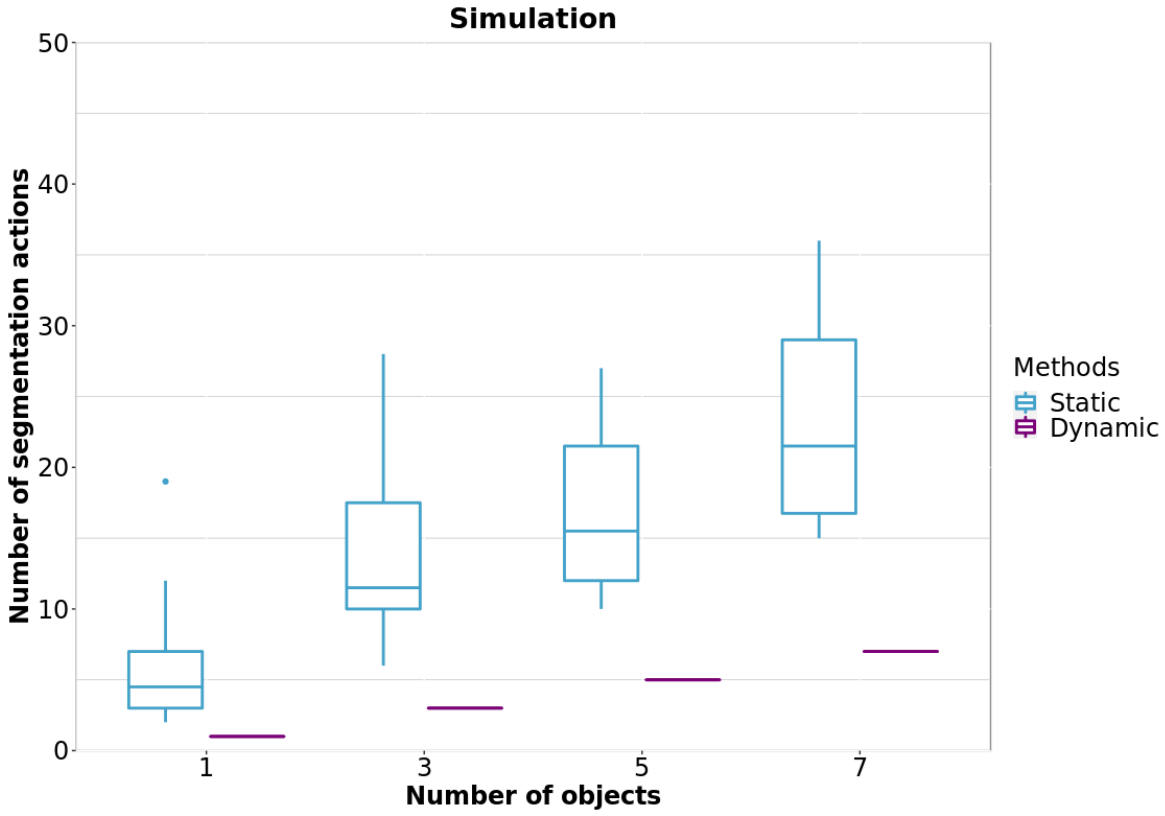


Figure 4.8: Comparison of the number of segmentation actions needed to find all objects in the simulated environment.

Due to the given robot design, objects were only placed on the floor for the experiments. Although detecting objects on the floor is crucial, e. g., to prevent the elderly from falling, it is important to note that the proposed method generalizes also to other locations. Adam et al. [43] adopt this work to use it as a baseline for detecting novel objects in the *3RScan* [24]. They report that our method performs very similar to their approach in terms of intersection over union, but falls behind by 12 % concerning the recall. Additionally, we also use the proposed method as one of the baselines in Chapter 5. The evaluation is performed on our acquired dataset (see Section 3.2) where objects are placed not only on the floor but also, e. g., on tables, sofas, and drawers. While achieving a high recall, the precision is low due to the sensitivity to slightly moved furniture as a result of the spatially global approach. This constraint is the main focus of the work described in Chapter 5.

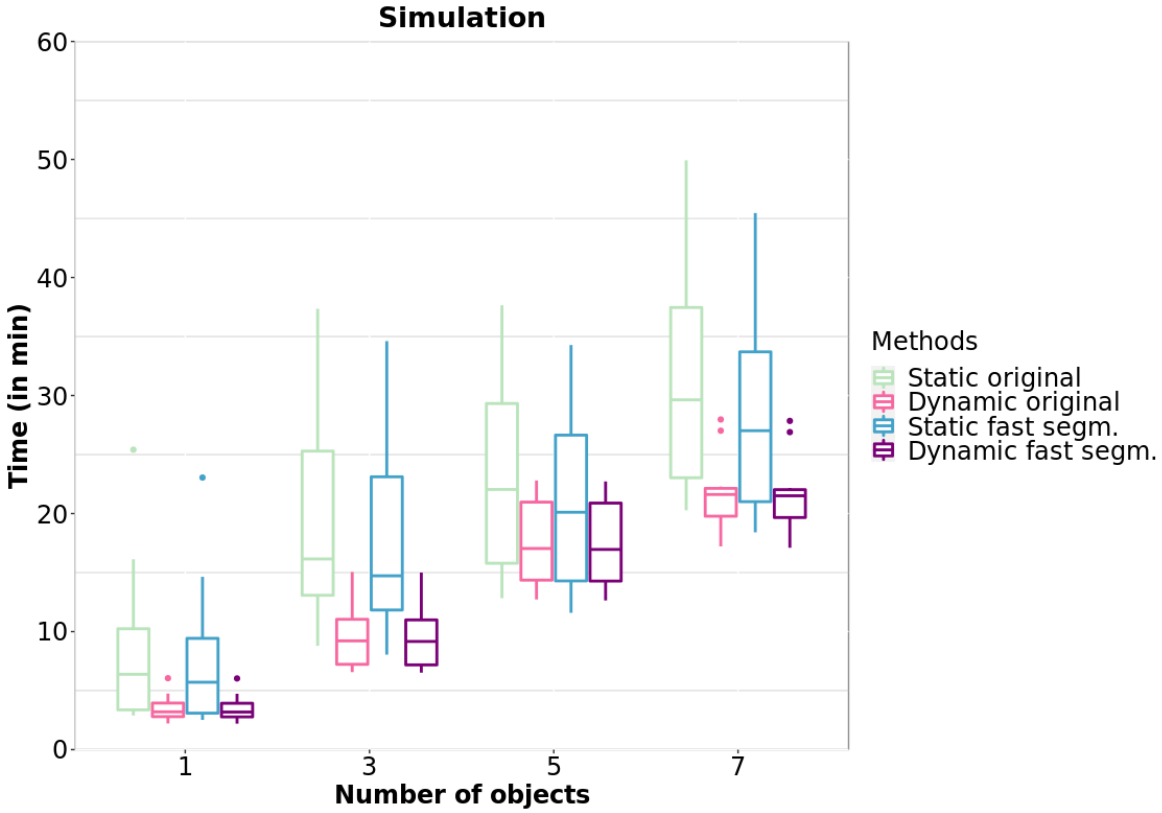


Figure 4.9: Comparison of the time needed to find all objects in the simulated environment.

Chapter 5

Robust and Efficient Object Change Detection by Combining Global Semantic Information and Local Geometric Verification

5.1 Introduction

The ability to detect objects, which are new at a location, in large environments is key for enabling robot tasks such as surveillance, tidying up, or maintaining order in homes or workplaces. These tasks share the commonality of operating in the same environment every day. As such, revisiting a particular environment enables robots to utilize domain knowledge and exploit their memory from previous visits. By storing a reference map of the environment, a robot is able to check for scene consistency and therefore detect changes on the object level. A household robot, for example, uses the cleaned-up version of the environment as a reference map to discover objects it should tidy up (see Figure 5.1). While a surveillance robot knows which objects are expected in its environment and triggers an alarm when the comparison to the current state reveals a missing object. In both cases, the robot is only interested in novel or removed objects, but not in objects that have a permanent place, such as a lamp or computer keyboard, which may move only slightly.

The standard approach to detect inconsistencies in the environment is to compute the difference between a reference and the current situation. This has the advantage over recognition methods, e.g. [87]–[89], since no object models are required, to be suitable for open-world conditions. A disadvantage, however, is that change detection applied at a large scale is sensitive to sensor noise and localization error. Furthermore, the readjustment of uninteresting objects, such as furniture or decorations, cannot be distinguished from novel objects.

This work presents a new approach to detect objects in real-world indoor environments based on reconstructions and overcomes the limitations of existing global scene differencing methods. Our idea is to exploit the strength of different approaches by combing full knowledge about environmental context with local geometry. At a global level, semantic segmentation reveals structures where objects are likely to be located,

Figure 5.1: A household robot tidying up. It compares a previously acquired reference map to the current state of the environment. Although the chair and other permanent objects moved slightly (colored in green), only the mug (colored in pink) should be detected as novel and therefore tidied up.

such as on a table, couch, or floor. In agreement with the real-world fact that objects are mainly placed on surfaces [13], [23], we use relevant structures to identify horizontal planes. The set of object candidates that are extracted from the planes is processed at a local level through geometrical verification against the reference map. In contrast to global scene differencing, local alignment is robust to the effects of sensor noise and localization error.

For the quantitative evaluation of our method, we use our generated full-environment dataset as described in 3.2. Experiments show that our approach significantly outperforms the baseline methods.

In summary, the contributions include

- the exploitation of knowledge from the task domain such as previous visits and the structure of the environment,

- a proposal of a unified approach for 3D open-world novel object detection that combines semantic information, plane extraction, and local verification,

- a significantly improved detection rate of novel objects using the combination of semantic knowledge and perception in comparison to competing approaches.

5.2 Method

This work addresses the problem of detecting novel objects in 3D environments. A novel object refers to an object or to a heap of objects, which is new in the environment at its current location or in close proximity. This differs from permanent objects that are always present at the same location or might have moved only slightly. Our approach, as outlined in Figure 5.2, combines multiple sources of information. Semantic information with horizontal plane detection is used at the environment level to generate an initial set of object candidates. The candidates are then verified through local geometric alignment. The local verification step overcomes the inaccuracies of global differencing because smaller regions suffer less from noise and warping in the reconstruction. The global detection stage is necessary to determine where to apply local verification, which would be time-consuming if performed exhaustively.

This section describes the proposed approach. We first explain the procedure for extracting object candidates from the global environment using semantic information. We then outline the verification procedure using local geometry.

5.2.1 Object Detection from Global Semantic Context

From the global reconstruction, semantic information is exploited to discover novel objects. Semantic segmentation has received the most attention in the computer vision community for pixel-wise classification of images and the rise of deep learning, in particular convolutional neural networks, has drastically improved results [90], [91]. The introduction of the *ScanNet* dataset [75] has enabled the transition to apply semantic segmentation to dense 3D reconstructions of indoor environments. In this work, semantic segmentation generates class labels for all vertices in the 3D reconstruction. We use *SparseConvNet* [48] trained on *ScanNet*, however, other methods and other training datasets could also be applied. Specific details of our implementation are given in Section 5.3.2.2.

We identify objects or parts of objects by searching for protrusions on supporting planes, i.e. horizontal surfaces on which objects may lie. Instead of searching for horizontal planes in the whole environment, the semantic information is leveraged by limiting the search space to relevant classes[1]. The reconstruction vertices corresponding to these classes are clustered. For each cluster, horizontal planes are fit using RANSAC [84] with the distance threshold τ_{r_dist} and the angle threshold τ_{r_angle}. Considering only the semantically relevant subset of vertices not only reduces the number of points that need to be processed but also achieves more accurate plane estimates because there are fewer outliers. Candidate objects are found by segmenting the vertices that lie above

[1] Floor, cabinet, bed, chair, sofa, table, bookshelf, counter, desk, shelves, night stand, other structure, other furniture, and other props.

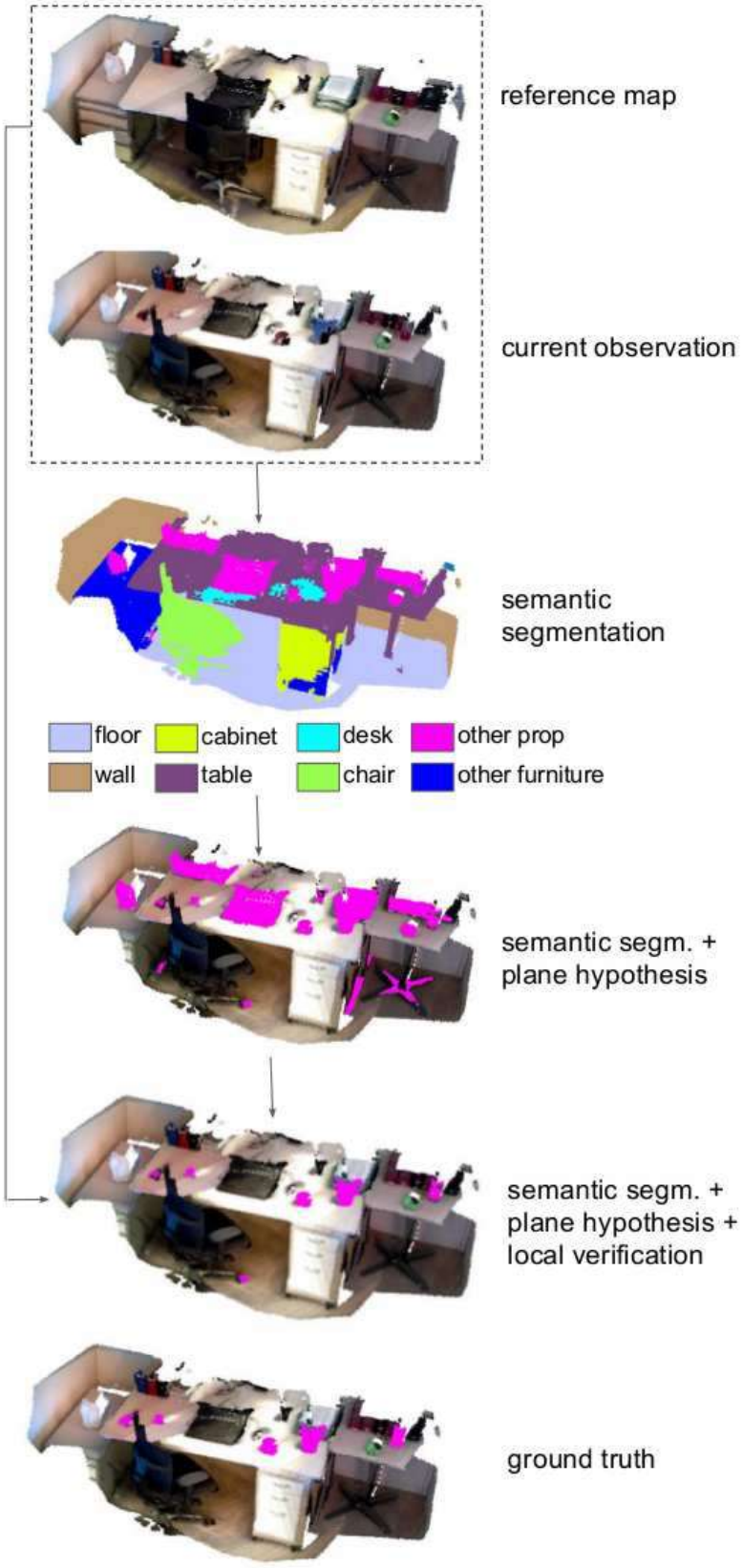


Figure 5.2: Overview of our proposed novel object discovery method showing object detection results for each step. Detected objects are displayed in pink.

Figure 5.3: Examples from *ScanNet* showing annotation inaccuracy. Most small objects are incorrectly labeled, either under-segmented or not separated from the supporting structure. Top: original scan. Bottom: annotation with objects in pink.

the detected supporting planes using Euclidean distance-based clustering [23] with the distance threshold τ_{segm}.

In the *ScanNet* dataset the `otherprop` class is defined and annotated. It is a general label typically associated with small items that do not belong to indoor structures. The vertices with the `otherprop` prediction could be used to directly identify objects, however, they are insufficient for discovering novel objects as we show in our evaluation (see Section 5.3). The main reason is that the ground truth scenes in ScanNet are labeled based on pre-segmented patches, which tend to undersegment the scenes. Therefore, trained models do not exhibit high precision around object boundaries. This is particularly problematic for small objects that either lack precise boundaries or are merged with the supporting structure (see Figure 5.3). Nonetheless, we also include all clusters of the `otherprop` class in the initial set of object candidates. These are then verified using local geometry as described next.

5.2.2 Object Verification with Local Geometry

Change detection is commonly used to identify novel objects in an environment as it requires no prior object information. Typically, a difference is computed between two spatially aligned observations. Points in the result set D are points from observation C that do not have a corresponding point in observation S within a distance d (adapted from [14]):

$$D = \{c | c \in C \wedge \nexists\, s \in S, \|c,s\| < d\}. \tag{5.1}$$

This formulation is highly sensitive to the distance threshold used. Furthermore, the pervasiveness of noise from depth sensors and localization errors reduce detection

accuracy.

Our approach leverages the idea of change detection but applies the operation locally. Since object candidates are already generated from the full environment using semantic information, it is no longer necessary to perform global change detection. It is sufficient to apply the operation in local regions around the initial candidates. This is a two-stage process. First, for each detected object cluster, we extract the supporting plane in its surrounding. It is aligned to the nearest horizontal plane in the reference map (plane-plane alignment) by applying the Iterative Closest Point (ICP) algorithm [92] with a maximum allowed distance d_{pp} and fixed rotation around the x- and y-axes. This initial step makes the verification of the actual object more efficient and robust since the second ICP step permits only transformations in the x- and y-direction and rotation around the z-axis to align the object cluster with the reference map (object-map alignment) allowing a maximum distance of d_{om}. For this operation, we use a crop of the current map with a margin c_{curr} around the detected object and a crop of the reference map with a margin c_{ref} around the potential object location as input. Moreover, the plane points detected in the first step are removed to ensure better alignment. Figure 5.4 exemplifies the steps of local verification.

Given ICP convergence, we transform the object point cloud and use Eq. (5.1) to determine the object's overlap with the reference map. Objects that have a lower overlap than o_{max} are considered as newly introduced into the environment. Details about the parameters used in the local verification stage are provided in Section 5.3.2.3.

Finally, the discovered novel objects are clustered with a distance threshold τ_{filter}. Clusters with less than obj_{min} points are considered as noise and removed from the result set.

Applying local verification has advantages over global differencing. Firstly, it can adapt to subtle changes between the observations, thus accounting for slightly moved objects that are not novel. Secondly, the inaccuracy of warped reconstructions can be absorbed. In contrast, global differencing is anchored to the full environment, which means it is inflexible and requires precisely aligned observations.

In our formulation, permanent objects are identified if they only move within the dimension specified by the reference crop c_{ref} to account for reasonable environment rearrangements. In reality, if a permanent object moves a greater distance, it is in fact out of place and should be detected. If an environment is rearranged, it is necessary to generate a new reference map.

5.3 Experimental Results

This section presents experimental results using our collected full-environment dataset. Before we describe the implementation of the proposed method, in particular, for reconstruction and semantic segmentation, a short summary of the dataset is given. We then introduce the comparison methods and outline the metrics used for evaluation. Lastly, we report quantitative and qualitative results as well as show the generality of our approach by applying it to different reconstruction methods.

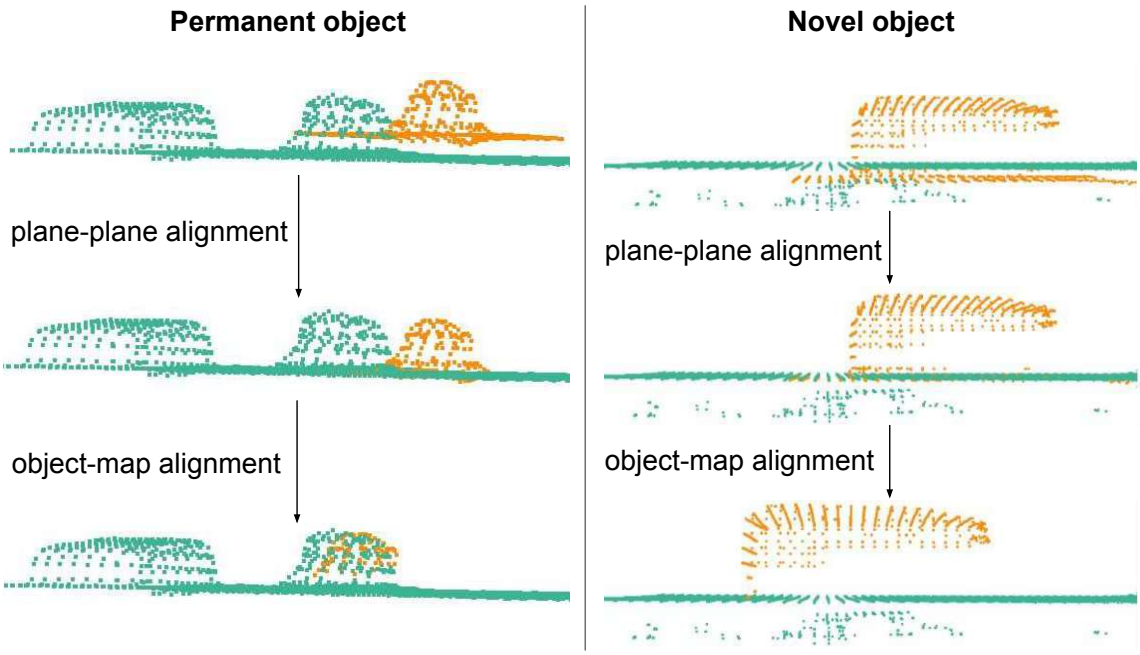


Figure 5.4: The two steps of the local verification are shown for a permanent and a novel object. The reference crop is colored in turquoise, current object and supporting plane in orange. Note in the novel object example, the two-step approach prevents ICP from aligning the object to the reference plane.

5.3.1 Dataset

The quantitative evaluation of the presented method and the comparison to baseline methods are based on the dataset introduced in Section 3.2. While datasets exist for related problems, e.g. [14], [16], none simultaneously fulfill the requirements of comprising different environments, providing annotations for small objects, and are recorded by a mobile robot. The utilized dataset consists of environments of different sizes. For each environment, an object-free recording exists (i.e. reference map) as well as various additional setups containing novel objects, rearranged furniture, and permanent items. Overall, we consider five distinct environments and a total of 31 full reconstructions generated with *voxblox* [27] including 260 annotated objects from the YCB object set.

5.3.2 Implementation Details

5.3.2.1 Reconstruction

The *voxblox* framework [27] is used to generate TSDF-based reconstructions. It was initially developed for planning purposes but is shown to be suitable for other robotic applications, such as incremental scene segmentation [38] and for extracting 3D object models [36]. *Voxblox* creates dense 3D maps based on the TSDF representation. We use the robot pose from the recordings instead of using camera tracking. However, the

Table 5.1: Parameters used for the experiments.

Method	Parameter	Value
Voxblox	TSDF voxel size	0.01 m
Voxblox	TSDF integrator method	simple
Voxblox	max. ray length	2.0 m
RANSAC	distance threshold τ_{r_dist}	0.01 m
RANSAC	angle threshold τ_{r_angle}	5°
Object segmentation	Clustering distance threshold τ_{segm}	0.015 m
LV	ICP max. distance plane-plane alignment d_{pp}	0.05 m
LV	ICP max. distance object-map alignment d_{om}	0.15 m
LV	reference crop margin c_{ref}	0.2 m
LV	current crop margin c_{curr}	0.05 m
LV	max. difference distance d from equation 5.1	0.014 m
LV	max. object overlap o_{max} for novel objects	0.7
Object filtering	Clustering distance threshold τ_{filter}	0.02 m
Object filtering	Minimum number of object points obj_{min}	15

option to refine the pose by aligning the input data to the existing structure with ICP is employed. The *voxblox* framework is very suitable for robotic applications not only because it provides the option to directly integrate the robot's pose, but also because it runs on a CPU only. Details about the parameters are given in Table 5.1.

5.3.2.2 Semantic Segmentation of Dense 3D Maps

SparseConvNet [48] is used to perform semantic segmentation on the full 3D reconstructions. It accepts a set of colored points as input. The output from *voxblox* is converted to this format by taking the centroid and average color of each voxel in the reconstruction. The network is trained on the *ScanNet* dataset [75] with the standard test, validation, and training splits. The annotations follow the second version of the dataset. Data is augmented using the provided tools from *SparseConvNet*. To train the model we used the default settings of *SparseConvNet* except for the following parameters: m=32, residual_blocks=True, scale=50, block_reps=2, batch_size=5. In addition to the 20 classes in the *ScanNet* benchmark, we included `otherprop` to have a total of 21 classes.

5.3.2.3 Parameters

Table 5.1 lists all parameters used in our implementation. This comprises the parameters for creating the reconstructions with *voxblox*, for detecting planes and objects above the supporting planes as well as the various parameters for the local verification (LV) stage and the final filtering of novel objects.

5.3.3 Comparison Methods

We select two baseline methods to compare our method against. Both perform scene differencing to detect dynamic objects. The method proposed by Ambrus et al. [14] (*Meta-room*) creates a reference map from several observations. This point-based volumetric representation is called meta-room and is further used for change detection. After aligning an observation to the meta-room, the difference between them is computed. The remaining points are clustered, then planar and small clusters are removed. Because the focus of our paper is object change detection, we use our object-free reference map created with *voxblox* as the meta-room when evaluating the change detection scheme in [14]. To ensure a fair comparison, we adapted the original parameters to the characteristics of our dataset, which achieves better results.

As another baseline, we compare against the method from Chapter 4, which uses an *OctoMap* [26] (*Octomap*) for representation and differencing. The effect of noisy sensor input is slightly reduced due to the quantization of the points into voxels and also the dilation applied to the reference map. We modified the method to detect all objects, not only those on the floor as in the original approach.

In addition to the two baseline methods, we give results for the different modules of our proposed approach. We use the direct output of semantic segmentation on the object level by considering all vertices labeled with the `otherprop` class (*Semantic segm.*). *Ours (no planes)* states the result when applying semantic segmentation and locally verifying the object candidates, but without including possible objects from supporting planes. We also evaluate our proposed method using only object detection from the global semantic context and without the verification stage of ICP alignment and local differencing (*Ours (no LV)*) to demonstrate the improvements of the final stage of our pipeline (*Ours (full)*).

Note that our method without local verification (*Ours (no LV)*) as well as semantic segmentation (*Semantic segm.*) only propose potential objects but do not exclude objects because they do not incorporate knowledge from a reference map.

5.3.4 Metrics

A number of different metrics are considered for the evaluation. The commonly used metrics of precision and recall are applied at the point level. These measure the accuracy of the object detections by considering all detected points in the environment and all points from the ground truth annotation. Precision measures the proportion of detected points that correspond to the ground truth ($TP/(TP+FP)$) and recall measures the proportion of ground truth points that are in the detection set ($TP/(TP+FN)$). The F1-score is also reported as it provides the harmonic mean of the two quantities.

Since we are concerned about the detection performance of objects, we also report two additional metrics. We measure the number of missed objects by comparing the overlap of the clustered detections with the ground truth objects. If no point of a ground truth object is detected then it is considered missing. To measure overestimation (i.e. false positives), we sum the number of detected clusters that do not overlap with a ground truth object for each recording. While this is not an accurate measure for false positive

detections, it allows an additional comparison of approaches at the object level.

5.3.5 Results

Table 5.2 shows the performance of the evaluated methods on our robotic dataset. The results from the different setups of each environment are averaged for precision, recall, and F1-measure. We also provide the standard deviation for these three metrics. Missed objects and wrongly detected objects are summed. The total average for all environments is also given. Qualitative results of our method for one setup of each environment are shown in Figure 5.5.

The quantitative results show that our approach clearly outperforms the baseline methods. Precision increases drastically for the methods that apply local verification. This shows the benefit of performing differencing only in restricted local areas. *Meta-room* and *Octomap* both report lower recall than our approach because of their post-filtering step, which is used to address the limitations of global differencing.

Semantic segmentation *(Semantic segm.)* performs surprisingly well in terms of recall, given the fact it was trained on a completely different dataset. However, the high number of missed objects indicates that especially smaller objects cannot be detected. Comparing the object candidates proposed by semantic segmentation to the reference map by applying our local verification step (*Ours (no planes)*) results in higher precision and it decreases the number of wrong objects. However, objects are still missing and the recall remains low. This is improved by exploiting the semantic information to explore horizontal planes (*Ours (no LV)*) and find small objects. However, without utilizing the knowledge from a reference map the number of wrongly detected objects explodes. Incorporating both proposed steps, semantic plane detection and local verification (*Ours (full)*), results in the best trade-off. The full method achieved good precision as well as recall and therefore the highest F1-score overall. On average, our method identifies 97.8 % of all novel objects in the dataset while detecting only 14 false positives per recording. This shows the importance of combining global and local procedures for accurate novel object detection.

The most common failure case of our method occurs when spatial clustering is not able to separate novel and permanent objects that are touching. An example of this is shown in Figure 5.5 at the bottom where the yellow-white sugar box is clustered together with the coffee machine. During local verification, the whole cluster is removed because the alignment of the coffee machine leads to a high overlap. In order to deal with this case, a segmentation algorithm could be applied to separate objects. Another failure case is induced by small objects, which get rejected in the course of local verification because it is very likely to find a transformation into the reference map resulting in a high overlap.

5.3.6 Generality to Different Reconstruction Methods

Our approach is applied to the output of different reconstruction methods to show its generality. We consider *Kintinuous* [32], *ElasticFusion* [28], and *ScalableFusion* [30]. Reconstructions of the reference map as well as observations are generated using the

Table 5.2: Comparison of different methods on the robotic dataset. (Pr = precision, Re = recall, F1 = F1-score, M = missed objects, W = wrongly detected objects)

	Pr	Re	F1	M	W
			Big Room		
Octomap[41]	0.07±0.04	0.42±0.15	0.12±0.07	42	434
Meta-room[14]	0.24±0.30	0.55±0.05	0.25±0.27	31	464
Semantic segm.	0.40±0.17	0.63±0.10	0.48±0.13	26	138
Ours (no planes)	**0.78±0.13**	0.62±0.10	**0.69±0.10**	27	**50**
Ours (no LV)	0.28±0.07	**0.78±0.04**	0.40±0.07	**2**	469
Ours (full)	0.60±0.17	0.77±0.04	0.66±0.12	3	168
			Small Room		
Octomap[41]	0.11±0.05	0.61±0.18	0.19±0.08	15	176
Meta-room[14]	0.04±0.03	0.44±0.08	0.07±0.04	24	276
Semantic segm.	0.16±0.09	0.52±0.24	0.24±0.14	21	119
Ours (no planes)	**0.55±0.36**	0.47±0.26	0.51±0.30	26	**28**
Ours (no LV)	0.17±0.06	**0.66±0.17**	0.27±0.10	**6**	236
Ours (full)	0.53±0.25	0.64±0.20	**0.57±0.22**	9	72
			Living Area		
Octomap[41]	0.11±0.08	0.50±0.08	0.17±0.10	19	74
Meta-room[14]	0.13±0.18	0.42±0.10	0.14±0.14	15	122
Semantic segm.	0.25±0.16	0.41±0.12	0.30±0.15	23	49
Ours (no planes)	**0.83±0.29**	0.41±0.12	0.53±0.17	23	**13**
Ours (no LV)	0.26±0.12	**0.69±0.11**	0.37±0.13	**4**	87
Ours (full)	0.79±0.24	0.68±0.11	**0.72±0.17**	5	24
			Office Desk		
Octomap[41]	0.18±0.07	0.77±0.13	0.28±0.10	8	73
Meta-room[14]	0.17±0.25	0.39±0.20	0.17±0.18	12	146
Semantic segm.	0.13±0.05	0.72±0.08	0.22±0.07	5	72
Ours (no planes)	**0.49±0.27**	0.65±0.08	0.52±0.18	8	**16**
Ours (no LV)	0.13±0.04	**0.83±0.06**	0.22±0.07	**0**	147
Ours (full)	0.46±0.23	0.75±0.09	**0.54±0.20**	3	36
			Kitchen Counter		
Octomap[41]	0.43±0.08	0.41±0.08	0.41±0.07	9	**40**
Meta-room[14]	0.56±0.17	0.35±0.12	0.44±0.14	9	70
Semantic segm.	0.19±0.04	**0.92±0.07**	0.31±0.05	**0**	197
Ours (no planes)	0.53±0.17	0.73±0.24	**0.55±0.11**	7	64
Ours (no LV)	0.16±0.05	0.72±0.25	0.26±0.08	2	189
Ours (full)	**0.62±0.21**	0.61±0.23	0.54±0.09	8	55
			Average		
Octomap[41]	0.18±0.14	0.54±0.18	0.23±0.13	18.6	159.4
Meta-room[14]	0.23±0.26	0.43±0.13	0.21±0.20	18.2	215.4
Semantic segm.	0.23±0.15	0.64±0.21	0.32±0.14	15.0	115.0
Ours (no planes)	**0.64±0.27**	0.58±0.19	0.57±0.14	18.2	**34.2**
Ours (no LV)	0.20±0.09	**0.74±0.14**	0.31±0.11	**2.8**	225.6
Ours (full)	0.60±0.23	0.69±0.15	**0.61±0.16**	5.6	71.0

Figure 5.5: Qualitative examples for all five environments from the dataset. Detected/ground truth objects colored in pink.

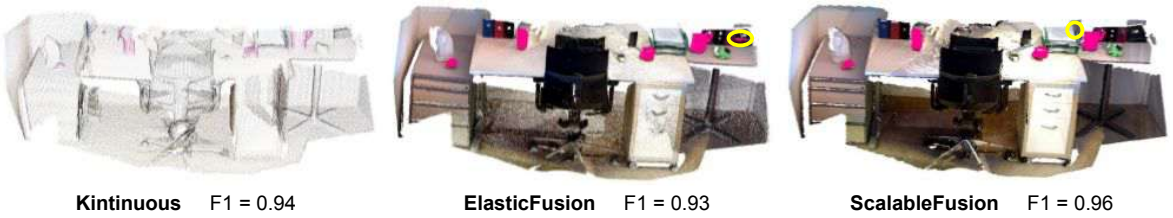


Figure 5.6: Qualitative examples of our approach applied on the same recording for different reconstruction methods. Detected objects are visualized in pink, wrong detections are highlighted with a yellow ellipse. All reconstructions are displayed with original point size.

default implementations of each mentioned method. The outputs of these methods are converted to a point set by taking the centroids (and average color values) of the voxels [32] or surfels [28] or taking the mesh vertices from [30]. For *ElasticFusion* and *ScalableFusion*, the differencing threshold is reduced by half to 0.007 m (d in Eq. (5.1)) due to the higher point density.

Figure 5.6 shows example outputs from the reconstruction methods for a recording of the office environment. The setup includes four novel objects when comparing it to the reference. All mentioned reconstruction methods are able to produce reasonable results, achieving high F1-scores. In all three cases, every novel object was identified. *ElasticFusion* and *ScalableFusion* detected one very small wrong object each.

5.4 Conclusion

This work addressed the problem of detecting novel objects in 3D reconstructions of indoor environments. We presented an approach that analyzes the environment globally and detects objects utilizing semantic information. The semantic context is exploited to extract objects on horizontal planes from relevant structures. The detected objects are then verified by performing change detection with a reference map in a local region. Results achieved on the full-environment dataset (see Section 3.2) show that our combined approach outperforms existing baselines. Our approach correctly detects novel objects while excluding slightly moved furniture and decoration. It is even possible to detect only partially visible objects as well as cluttered novel objects, whereby a post-processing step would be needed to separate them into single objects. A possible solution to dissolve heaps of objects using change detection is presented in Chapter 6. The method described in the following chapter builds on the ideas of the presented work and distinguishes between truly novel objects and moved objects in the environment by utilizing a fast and easy to compute feature descriptor.

Chapter 6

Where Does It Belong? Autonomous Object Mapping in Open-World Settings

6.1 Introduction

When asking people what they wish robots could do at home, tidying up is top-ranked [93], [94]. To work towards the open challenges, several competitions have been started, for example, the ICRA 2018 "Tidy Up My Room" Challenge[1] or the WRS RoboCup@Home [95] tidy-up task.

The tidy-up task is complex because a robot operates in an unstructured and dynamic environment where it needs to localize known as well as unknown objects. To determine where objects belong, the robot needs to have knowledge about the intended storage locations for these objects, e.g., in a knowledge base [96] or another form of reference. The focus of this work is on the perception system for the tidy-up task and related applications, for which we refer to object detection and matching as the *object mapping* task.

Object detection is the prerequisite for finding matches. It reduces the object mapping task to the problem of comparing the locations of objects in the environment at two time instances. This definition is generic and independent of the specific robot task. For example, if the task is tidy up, a comparison is performed between the present situation and a reference map. If the task is patrolling, the observant robot will use all object detections to create a present object map for the new time instance and finds deviations from the normal state. For fetching an object, the knowledge of where this object was last seen, i. e., in the present object map, is used to retrieve it and, if not found, to start a search that may include historic information about where the object has been found before.

Similarly to the object mapping task, the object rearrangement task as introduced by Batra et al. [97] also deals with the goal of transforming the current state of the environment into a target state assuming a closed world. This does not represent the real world, which must consider objects that appear, and are therefore unknown, or disappear. Today most approaches assume a given and fixed set of objects, e.g., [45], [47].

[1] `http://juxi.net/challenge/tidy-up-my-room/`

Towards the goal of open-world object mapping, we present an approach that copes with all possible cases of static, moved, removed, and novel objects in different environments. We partition an environment into local horizontal surfaces, which is motivated by the fact that objects are typically found on furniture such as tables or shelves. Furthermore, it is infeasible for daily use to repeatedly and exhaustively scan the whole environment. Tasks rather need to check if the object is at a specific location or surface. Finally, the concept of local horizontal surfaces is easily extendable to (1) include other structures such as vertical surfaces to locate a broader variety of objects including pictures, switches, or door handles and (2) comprise multiple rooms.

At the core of our approach is a comparison function to match detected objects to previously seen instances. To achieve this we represent surfaces where the objects reside as a 3D point cloud. For autonomous surface discovery, we exploit semantic segmentation. Finally, the concept of local surfaces enables high-quality reconstructions of every individual surface, which enhances the matching of detected objects using state-of-the-art methods such as Point Pair Feature (PPF) descriptors [88]. PPFs are computationally cheap and run on CPU only, which plays a considerable role for approaches deployed on mobile robots.

Our approach is evaluated on the *ObChange* dataset (see Section 3.2). It encompasses multiple recordings of five environments with a total of 219 annotated objects. Taking all possible comparisons of recordings per environment into account, this leads to 961 objects for detection and matching. We report the results achieved on *ObChange* compared to a baseline using a learning-based detection approach as well as highlight possible failures and remaining challenges. Furthermore, we show the performance of our proposed approach using a fully autonomous system working in a real indoor environment.

To summarize, our contributions are

- a procedure that uses semantic segmentation and plane fitting to reliably detect objects and that robustly handles all cases encompassed in an open-world environment, that is, static, moved, removed, and novel objects,
- an object matching approach that does not rely on a trained classifier or pre-defined 3D models and thus, works in an open-world setting by leveraging information extracted from 3D representations,
- the utilization of the *ObChange* dataset to evaluate different detection methods categorizing objects into the four cases,
- an evaluation on a fully autonomous robot that performs experiments in a real environment.

6.2 Object Mapping using Local Surfaces

This section formally defines the problem of object mapping in arbitrary environments in Section 6.2.1. An overview of the perception components for this task is given in Section 6.2.2. Finally, the details for reconstruction, object detection, and object matching are described in Sections 6.2.3 to 6.2.5.

6.2.1 Problem Definition

The goal of this work is to detect objects in an environment and to further assign each object a category depicting its relationship to previous detections. The focus is on objects, which are detachable from the surface they are placed on and can be manipulated by a service robot. To remain task-independent, we compare objects present in an environment at time t_0 and objects detected at a later time t_1. We refer to objects detected at t_0 as models and denote the set of detections as $\mathcal{M}$. Objects detected at t_1 are referred to as candidates and the set is denoted $\mathcal{C}$. Detected objects are matched across the time instances, then categorized into static, moved, removed, and novel; see also Table 1.1. A static object is a candidate $c \in \mathcal{C}$ that has a matching model $m \in \mathcal{M}$ and where the distance between c and m is less than a threshold d. This threshold is selected depending on the uncertainty in robot localization, the reconstruction, object detection, object placement, etc. The value may be different depending on specific applications. We use $d = 20\,\text{cm}$ throughout the chapter. A moved object is a candidate that has a matching model but where the distance between the objects is greater than d. An object is considered removed if it exists in $\mathcal{M}$ but has no matching candidate in $\mathcal{C}$. Novel objects are any candidates in $\mathcal{C}$ that have no matching model in $\mathcal{M}$. The set of models $\mathcal{M}$ is the union of all static, moved, and removed objects at t_0 while the intersection of these must be empty. Likewise, the set of candidates $\mathcal{C}$ is the union of static, moved, and novel objects at t_1 while their intersection must be empty.

6.2.2 System Overview

An overview of the proposed perception system for autonomous object detection and matching is given in Figure 6.1. The approach is composed of two phases: (1) setup of the reference map (blue), which needs to be performed only one time, and (2) a run through the environment to visit all or a subset of known surfaces and match detected objects (green and orange). If the larger structure of the environment or the main surfaces (such as furniture) are significantly moved, phase one must be repeated to generate a new reference map.

This work leverages the concept of surfaces to focus object comparisons, which ultimately leads to improved object detection. Technically, this is achieved by first combining the reconstruction of the environment and semantic segmentation to create a list of relevant surfaces, see Section 6.2.3. Then a more detailed scan of every surface is performed to improve the reconstruction. This high-quality reconstruction of each surface is used to detect objects (see Section 6.2.4), which are stored in a database as the reference object map for future change detection requests. When the robot revisits environments and surfaces for its specific task, for example, tidying up a kitchen counter, a new detailed reconstruction is generated. Objects are extracted and then matched to those in the reference. The matching process is performed in three different ways to handle different cases as outlined in Section 6.2.5.

In this work, we use only change as a cue for segmentation, which is fully applicable to open-world settings. If change occurs multiple times, then a heap of objects may

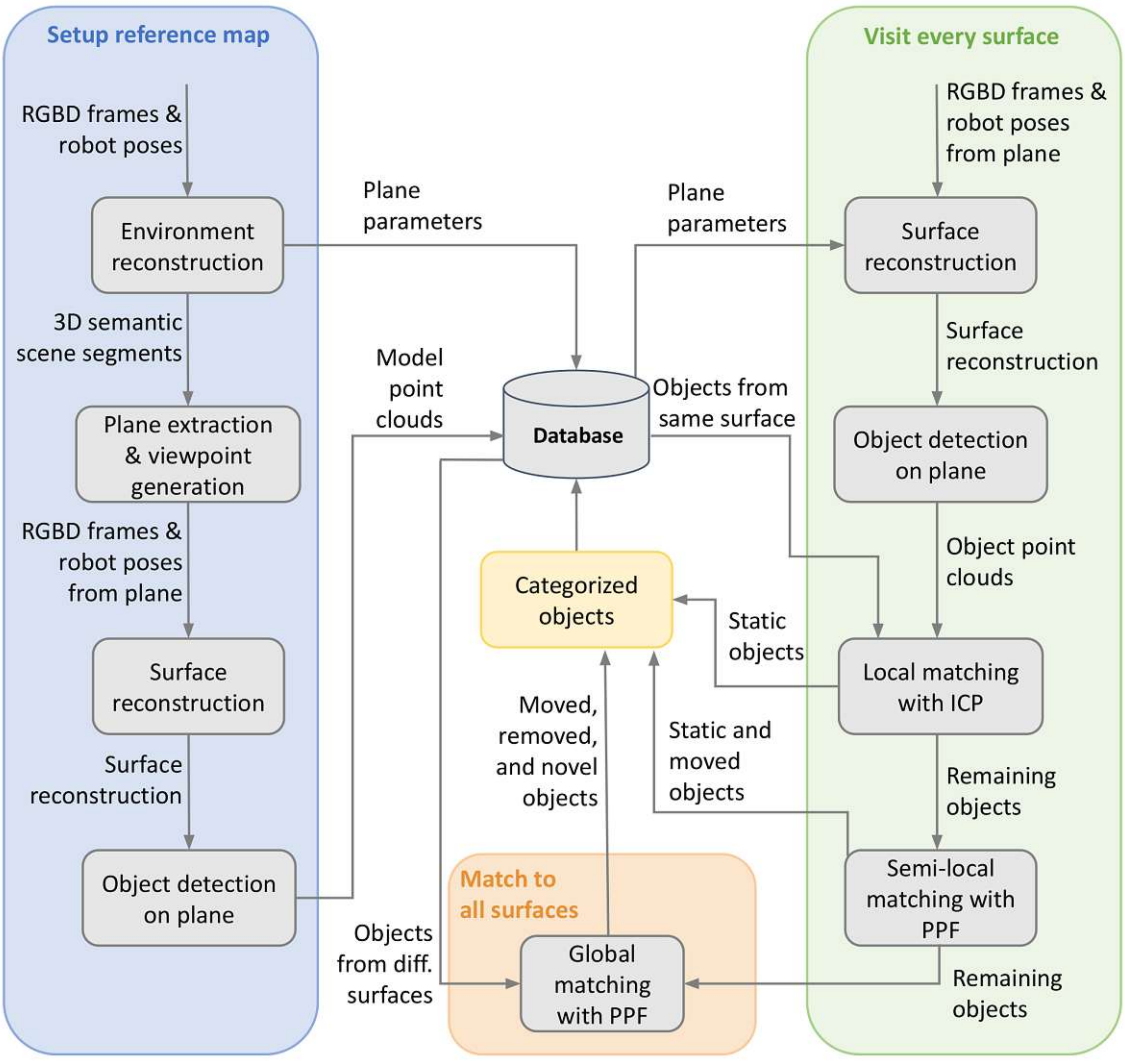


Figure 6.1: System overview of our approach to detect and match objects. The setup of the reference is performed only once.

become disentangled, otherwise, the heap will be considered as a single object. In the following sections, we refer to each detection as an object, both for single-standing items or heaps of objects.

6.2.3 Reconstruction of the Indoor Environment and Plane Extraction

The first step for detecting objects is to identify the regions where objects are commonly located, in other words, the surfaces. Similar to the approach described in Chapter 5, the search space for objects is reduced according to the assumption that objects are typically placed on horizontal planes in home environments [22], [98]. To extract horizontal planes, the environment is reconstructed using *voxblox* [27]. This method

runs on CPU only and is tightly coupled with ROS [99], both great qualities when working with a robot. The coarse reconstruction of the environment, which is a result of the voxelized representation, is used to geometrically search for planes. To do so, the reconstruction is transformed into a point cloud by extracting the centroids of all voxels. In addition, *SparseConvnet* [48] is applied to the reconstruction to retrieve a semantic label for each point and consequently to exclude non-relevant regions for the plane search – all points are removed that are not assigned any of the following classes: cabinet, bed, chair, sofa, table, bookshelf, counter, desk, shelves, nightstand, other props, other structure, and other furniture. Since we focus on horizontal planes, only points with a normal facing upward are retained. Finally, for each semantic class, the remaining points are downsampled and are the input to RANSAC [84] to fit a plane. Each iteration generates one plane and these points are removed from the input to enable further plane fitting. The loop ends when the extracted plane consists of less than a certain number of points.

For each plane, descriptive information such as the plane coefficients, convex hull points, and centroid are stored in a database. Additionally, waypoints for the robot to navigate to when inspecting the plane are computed. Waypoints are equally distributed positions around the plane at a fixed distance to the edge of the convex hull. The pose of each waypoint is described by its position and orientation, which faces the center of the plane. All the surface information is used for subsequent visits.

6.2.4 Reconstruction of Surfaces and Object Detection

Once the global reconstruction of the environment is created, a higher-quality local reconstruction is generated for each surface to enable more precise object detection. In this work, we use *ElasticFusion* [28] for local surface reconstructions. It uses both photometric and geometric pose estimation, which is configured using a relative ICP/RGB tracking weight parameter. While *ElasticFusion* is more precise than *voxblox*, it still suffers specific failure cases, which need to be addressed for robust operation on a mobile robot. Firstly, viewpoints focusing on large planar and low-textured surfaces have too few features to track the camera pose, which results in misalignment (see Figure 6.2 (top)). Secondly, changing lighting conditions resulting in over- and underexposed images is problematic for registration as depicted in Figure 6.2 (middle). Other sources of error are geometric symmetries as well as low depth disparity. Figure 6.2 (bottom) shows an example reconstruction using RGB and ICP registration, which suffers from duplicated and misaligned objects or smeared objects.

To countermeasure those real-world problems we propose a computationally simple solution. Our idea is to assist the tracking method of *ElasticFusion* whenever the estimated trajectory begins to significantly diverge from the reported robot odometry data. Clearly, odometry data has inaccuracies, thus using it directly is not sensible. Whenever the registration fails completely (e.g., Figure 6.2 (top)) the estimated pose and the odometry pose differ significantly. Resetting the estimated pose to the odometry data is not feasible either because this leads to smeared reconstructions. Our approach is to blend the poses from the camera tracking and odometry data to repair drift and misalignment errors in the running reconstruction. For each frame, before running

ElasticFusion's frame processing, we recompute the pose given the last estimated pose and the robot pose from odometry data. To this end, we first compute the difference between the last estimated pose E and the current robot pose C.

More specifically, the poses are represented by transformation matrices:

$$E = [R_E|\mathbf{t}_E]\,, C = [R_C|\mathbf{t}_C] \in \mathbb{SE}_3 \tag{6.1}$$

with rotation components $R_E, R_C \in \mathbb{SO}_3$ and translation components $\mathbf{t}_E, \mathbf{t}_C \in \mathbb{R}^3$. Next, the matrix D representing the difference is computed:

$$D = C^{-1}E = [R_D|\mathbf{t}_D]\,. \tag{6.2}$$

Given this, we compute the angular difference:

$$\phi = |\arccos\left(\frac{\text{trace}(R_D) - 1}{2}\right)|, \tag{6.3}$$

to derive the mixing term:

$$\lambda = \max\left(\min\left(1.0, \left(\frac{\phi}{z_r}\right)^2\right), \min\left(1.0, \left(\frac{|\mathbf{t}_D|}{z_t}\right)^2\right)\right), \tag{6.4}$$

where z_r and z_t are constant scaling factors. This allows the modified E' to be used as a replacement for E through a linear combination of the poses:

$$E' = \lambda C + (1 - \lambda)E. \tag{6.5}$$

The more the two poses disagree (i.e., the estimated pose diverges from the pose measured by the robot) the more the odometry pose is used in the hope of future agreement. Clearly, if odometry is less accurate, the results will degrade. However, it still prevents ElasticFusion from completely failing in difficult scenes. In our experiments, we found good results with $z_r = 0.2$ and $z_t = 0.2$, which is used for all experiments.

Integrating the tracked camera and robot poses generates high-quality reconstructions in real-world scenarios from the robot trajectory. This adaption is used to create a reconstruction for each plane, which is stored in the database. Plane parameters and waypoints derived in the setup stage are queried from the database. Based on this information, the robot navigates around the plane while the camera is directed to the center of the plane. Before *ElasticFusion* transforms the camera stream into a reconstruction, the depth images are pre-processed by cropping them such that only the surface of interest is retained. Cropping the depth input prevents *ElasticFusion* from trying to align the background (e.g., walls) at the expense of reconstruction accuracy of objects placed on the plane. Benefiting from the local surface concept, trajectories to create surface reconstructions are comparably short and result in precise reconstructions. From these generated surface reconstructions, objects are extracted by removing the points of the plane according to the known plane parameters. The parameters α_{max}, to allow deviation from a perfectly horizontal plane, and d_{plane}, which defines the maximum distance of inlier points to the detected plane, have to be chosen to take into account

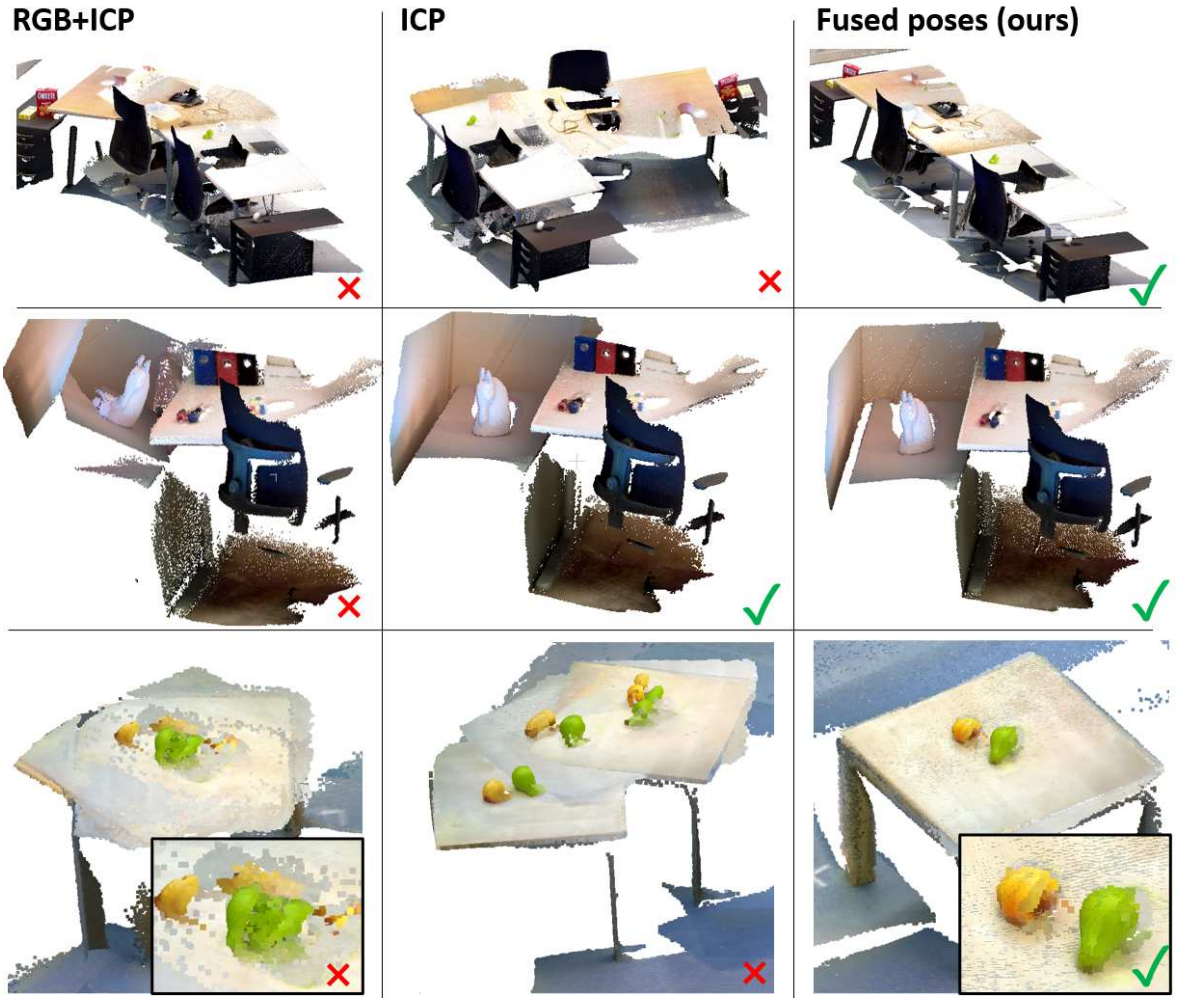


Figure 6.2: Each row shows a surface of one of the environments in our dataset. The first column is the result of *ElasticFusion* using RGB and ICP information to estimate the camera pose, the second row shows results using only ICP and the last column shows the result of fusing the robot poses with the estimated camera poses from *ElasticFusion* (only ICP).

small inaccuracies. The remaining points within the convex hull are clustered using Euclidean distance. All points up to 0.3 m above the plane are considered, however, this value can be chosen depending on the application. We use a minimally shrunk convex hull to reduce the number of false positive detections such as walls or armrests. At this stage, we do not try to separate objects in clutter and treat each cluster as an object.

6.2.5 Object Matching and Categorization

To support high-level robot tasks, objects are assigned one of the four categories, which is an indicator of what action should be performed with the object. For example, in a tidy-up task a static object should be left alone while a moved object should be returned

to its original location. Performing the categorization requires the determination of which objects are present in different visits and additionally finding those that match. The following subsections explain the different stages of matching, which are also depicted in Figure 6.1.

6.2.5.1 Local Matching

For each detected object, a check is performed to determine if it is still approximately located at the same position. If there exists a model and a candidate within a distance less than d, they are considered a potential match. A confidence score for the match is computed by aligning their point clouds with ICP (with fixed rotations around the x- and y-axes), which is suitable in this case as their proximity provides a good initial registration. Two scores are then computed if ICP converges: one for the model S_m and one for the candidate S_c due to the object potentially not being symmetric. The model and the candidate are a match if $\min(S_m, S_c) > \tau_{icp}$ for a given threshold τ_{icp}.

Formally, we consider the model point cloud $\mathcal{P}_m$ and candidate point cloud $\mathcal{P}_c$. For each point $\mathbf{p}_m \in \mathcal{P}_m$, a set of points $\mathcal{Q}_{m,c} \subset \mathcal{P}_c$ is determined as the collection of all corresponding points in $\mathcal{P}_c$ that have a distance to $\mathbf{p}_m$ less than the inlier threshold τ after alignment. A score is then computed for $\mathbf{p}_m$ and each $\mathbf{p}_c \in \mathcal{Q}_{m,c}$, which is composed of the geometric and color similarity. Given the point normals, $\mathbf{n}_m$ and $\mathbf{n}_c$, the geometric score is given by:

$$s_{geo} = \begin{cases} \mathbf{n}_m \cdot \mathbf{n}_c & \text{if } \mathbf{n}_m \cdot \mathbf{n}_c \geq \tau_{geo}, \\ 0 & \text{otherwise.} \end{cases} \tag{6.6}$$

The color score is computed as:

$$s_{col} = \begin{cases} 0 & \text{if } \boldsymbol{\kappa}_m \ominus \boldsymbol{\kappa}_c \geq \tau_{col} \\ 1 - \frac{\boldsymbol{\kappa}_m \ominus \boldsymbol{\kappa}_c}{\tau_{col}} & \text{otherwise,} \end{cases} \tag{6.7}$$

where $\boldsymbol{\kappa}_m$ and $\boldsymbol{\kappa}_c$ are the color values of the points $\mathbf{p}_m$ and $\mathbf{p}_c$ in LAB-space and $\ominus$ is the CIEDE2000 color difference [100]. τ_{geo} and τ_{col} are thresholds. The similarity score between $\mathbf{p}_m$ and $\mathbf{p}_c$ in the correspondence set is the weighted combination of the geometric and color scores:

$$s_{m,c} = \begin{cases} 0 & \text{if } s_{geo} = 0 \vee s_{col} = 0 \\ w s_{geo} + (1 - w) s_{col} & \text{otherwise,} \end{cases} \tag{6.8}$$

where w balances the contribution of the geometric and color similarity.

The overall fitness score for the model is defined as:

$$S_m = \frac{1}{|\mathcal{P}_m|} \sum_{\mathbf{p}_m \in \mathcal{P}_m} s^*_{m,c}, \tag{6.9}$$

where $s^*_{m,c}$ is the best similarity score for $\mathbf{p}_m$ given all the scores computed for the points in its correspondence set $\mathcal{Q}_{m,c}$. Likewise, the overall fitness score for the candidate is computed:

$$S_c = \frac{1}{|\mathcal{P}_c|} \sum_{\mathbf{p}_c \in \mathcal{P}_c} s^*_{c,m}. \tag{6.10}$$

A match is recorded if both scores S_m or S_c are greater than the matching threshold τ_{icp}. If only a subset of the points match, it is necessary to examine if either or both of the model and candidate need to be split into independent objects. A split based on the overlapping points is insufficient due to possible over- or under-segmentation of the objects. Therefore, to generate more precise object boundaries we perform region growing based on color and normal similarity where only points that contributed to the score are seed points. For all seed points, points within a radius r are added if color and normal are similar enough compared to the seed point. The color difference is computed with the CIEDE2000 formula using the color values in the LAB-space of both points and is compared against the maximum allowed color difference rg_{col}. The dot product from the normal vectors is used to check if the angle difference is not greater than rg_{geo}. Points that fulfill both criteria are added and act as new seed points. If no more points can be added, the region growing stops. All points in the result that belong to the model or candidate are categorized as static. The remaining points form a new object in the respective set.

6.2.5.2 Semi-local Matching

Spatial proximity at t_0 and t_1 is the prerequisite to match static objects with ICP. For other objects placed on the same plane, but which are further away or rotated and where the initialization is therefore poor, a more robust matching scheme is required that is independent of the pre-alignment.

We match objects by using a global descriptor based on Point Pair Features (PPFs), which is invariant to rigid transformations. The descriptor is proposed by Drost et al. [88]. It is a simple learning-free descriptor, nevertheless top-ranked in pose estimation challenges [101]. PPFs are represented by a four-dimensional feature vector describing the distance between two points as well as the relationship between their normal vectors. Due to their simplicity PPFs are fast to calculate. The feature vector is computed for all sampled point pairs of a model. Point pairs with similar features are stored together in a hash table, whereas the feature vector is the hash key. The hash table is used for a fast look-up to find point pairs of the model with similar features compared to sampled point pairs of the object. For each match, a transformation to align the model and object point pairs is computed and an efficient Hough voting scheme is applied to retrieve the final transformation.

In our system, the PPF descriptor is computed for each unmatched object and the pipeline returns for each candidate between zero and ten hypotheses for each model. Next, for each hypothesis, we compute a confidence score for the model and the candidate as formulated in (6.9) and (6.10) and then compute the average of S_m and S_c. Only the best-fitting hypothesis from each model is retained per candidate. Note that planar and small objects are filtered before applying PPF matching. All objects are downsampled to achieve a unified point density with a voxel size v and only objects with more than obj_{min} points and where less than τ_{plane} of the points are explained by a plane model are kept. Otherwise, their geometric characteristic is too generic and would result in many false matches.

More concretely, the matching problem is simplified by eliminating objects that match

with high certainty and are therefore assumed to be the same objects or heaps of objects. To unravel the hypothesis we use a bipartite maximum matching graph [102]. The nodes on one side are the models and on the other side are the candidates. A connecting edge exists if the model hypothesis for a candidate fulfills $\min(S_m, S_c) > \tau_{high}$, where τ_{high} is a fixed threshold. The weight of an edge is the minimum of the two scores. The maximum weighted matching of the graph is then computed. For each model-candidate pair in the graph solution, we compute the spatial distance and categorize it either as static or moved. These models and candidates are considered as fully matched and are not processed any further.

With the reduced set of models and candidates, a new graph is built where the condition for an edge is relaxed. An edge is created if $\min(S_m, S_c) > \tau_{min}$ and $\text{avg}(S_m, S_c) > \tau_{low}$. The edge weight is computed as the average of the two scores amplified by the exponential function. This graph is then used to compute the maximum weighted matching. With $\tau_{low} << \tau_{high}$ it is possible to match models and candidates in heaps. In such cases, it is infeasible to achieve very high confidence because of objects that are clustered together and increase the number of points used to normalize the score. This concept also helps to overcome deficits arising from incomplete reconstructions caused, for example, by few viewpoints or occlusions. The extracted matching results from the graph are then processed the same way as described in Section 6.2.5.1: starting from the matched points region growing is performed to extract all points from the reconstruction belonging to the matched model/candidate. These points are then categorized as static or moved depending on their spatial distance. The remaining points are considered as an additional model/candidate.

For all unmatched candidates, new hypotheses with the unmatched models are computed. The matching process restarts by building a graph with the relaxed edge condition. This procedure is repeated until no more matches are found.

6.2.5.3 Global Matching

The final matching procedure considers objects that have been moved between different surfaces. This is performed by collecting all models and candidates from all surfaces that were not matched in the local or semi-local steps. Technically, the same approach as described in Section 6.2.5.2 is applied but now all objects from all surfaces are pooled together to perform global matching. The PPF descriptor is the basis for hypotheses creation and computed confidence scores are used as edge weights for a maximum weight bipartite graph. In the case that there is no match for a candidate or a model in the entire environment, a candidate object is considered novel and a model object is considered removed.

The advantage of our approach is that the robot does not need to learn object models in a cumbersome process, but inherently extends its knowledge through change detection. Figure 6.3 shows an example of the clutter-dissolving capabilities of our system on an example from the *ObChange* dataset. At timestamp t_0 a pitcher and a mug are detected next to each other. They are treated as one object. At timestamp t_1 the pitcher is a single standing object, while the mug belongs to a pile together with the bowl. First, the pitcher is matched and with that information, the mug is separated

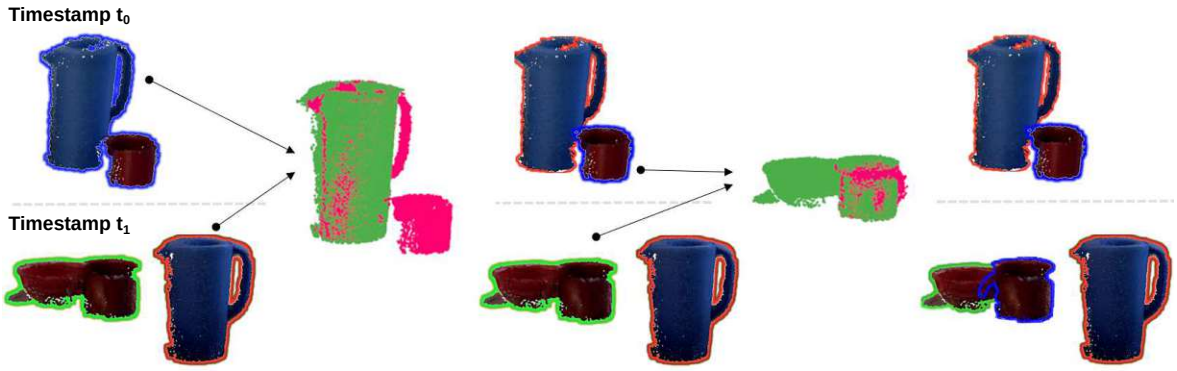


Figure 6.3: Example showing how objects are separated using PPF matching. Objects next to each other having the same shadow color are recognized as one object by the robot. After objects are matched they have the same shadow color in both recordings.

from the pitcher in t_0. Now the separated mug is matched with the mug in t_1. The result of the matching process is that the heaps are disentangled and instead of three object clusters the robot is now aware of five individual objects.

6.3 Experiments and Discussion

This section evaluates the performance of our approach for detecting and categorizing object changes. It is compared against two variants of a learning-based method as baselines and quantitative experiments are conducted with the newly created *ObChange* dataset, consisting of recordings from a real autonomous robot (see Section 3.2 for an overview and Section 6.3.1 for details). Additionally, we qualitatively demonstrate the applicability of our method in real-world scenarios with online experiments using a real robot (see Section 6.3.2). Finally, we discuss the indications of our experimental results and the consequent open research questions in Section 6.3.3. For reproducibility, we give an overview of all parameters used for the dataset and real-world evaluation in Table 6.1.

6.3.1 Evaluation on the Robotic Dataset *ObChange*

Since no suitable dataset exists to evaluate object mapping, we created *ObChange*. Its detailed description is given in Section 6.3.1.1. Section 6.3.1.2 introduces the metrics to measure the performance for the evaluation. In order to analyze our approach quantitatively, we compare it to two learning-based baselines using state-of-the-art methods as outlined in Section 6.3.1.3. Section 6.3.1.4 outlines the evaluation setup and in Section 6.3.1.5 we discuss the results and open challenges in detail.

Table 6.1: Parameters used for evaluation

Method	Parameter	Value
Local plane extraction	Maximum angle α_{max} between plane normal and upward-directed axis	5°
Local plane extraction	Inlier distance threshold d_{plane} for plane model	0.015 m
Matching score	Inlier threshold τ for radius search	0.01
Matching score	Color threshold τ_{col} for point-wise matching	20
Matching score	Dot product threshold τ_{geo} between two normal vectors for point-wise matching	0.95
Matching score	Linear weight factor w for combined score	0.7
Object filtering	Voxel size v for object downsampling	0.005 m
Object filtering	Minimum number of object points obj_{min}	200
Object filtering	Proportion of object points to count as plane τ_{plane}	0.9
Region growing	Point inlier radius r	0.01 m
Region growing	Maximum allowed angle between normal vectors of neighboring points rg_{geo}	5°
Region growing	Maximum allowed color difference between neighboring points rg_{col}	15
Local matching	Minimum score for candidate match τ_{icp}	0.7
Semi-local matching	Low score threshold τ_{low} for graph edge	0.4
Semi-local matching	High score threshold τ_{high} for graph edge	0.8
Semi-local matching	Minimum score for model and candidate τ_{min}	0.2
PPF	Distance sampling rate as defined in [88]	0.025
PPF	Orientation sampling rate as defined in [88]	5

6.3.1.1 The *ObChange* Dataset

For the quantitative evaluation of the object mapping task, we use the *ObChange* dataset created from real camera data collected by a robot. Details about the acquisition, the five different environments, and the experimental setup are given in Section 3.2. For each recording between 3 and 17 objects from the YCB object set are labeled (only excluding those placed on the floor). For the *ObChange* dataset, we use all environments and all different recordings, which include YCB objects – in total 26 recordings.

The *ObChange* dataset extension from the data used in Chapter 5 respectively [17] is necessary to achieve high-quality reconstructions to enable object matching. For each environment, a 3D semantically labeled reconstruction is created, which is used to identify horizontal planes as described in Section 6.2.3. For each detected plane, images from the recorded stream where the surface is visible are extracted, depth images are masked according to the plane parameters and *ElasticFusion* is used to reconstruct the area. Only with the combination of odometry pose and camera tracking pose as given in Equation 6.4, suitable reconstructions for all surfaces are achieved. In this dataset, the robot drives exhaustively through an environment and, therefore, may

Table 6.2: Overview of the dataset used for the quantitative evaluation. Except for the first two columns the numbers state the sum over all possible comparisons per environment.

Environment	#visits	#surfaces	#objects	#static	#moved	#removed	#novel
Big Room	6	12	425	28	260	50	87
Small Room	5	5	208	72	62	25	49
Living Area	5	11	108	0	54	30	24
Office Desk	5	4	100	12	28	24	36
Kitchen Counter	5	2	120	10	66	14	30
	26	**34**	**961**	**122**	**470**	**143**	**226**

see surfaces of interest at several points in time. Unfortunately, *ElasticFusion* cannot handle non-continuous input data. To overcome this problem, we first create individual reconstructions for each appearance of the surface and then merge them using ICP. We visually check the results and adjusted them manually if needed. This manual step is not needed in the real world where the robot moves around the surface only once. The collection of all surface reconstructions together with their point-wise labeling of all YCB objects form the *ObChange* dataset.

Compared to our work in the previous chapters we are not only interested in detecting all objects, but also to assign them to one of the four categories - static, moved, removed, and novel. To create a more meaningful evaluation of the possible categories, we compare each recording with all other recordings of the same environment, leading to 961 objects in total. For each comparison the objects of both recordings are counted, meaning that static and moved objects are counted at t_0 and t_1. Table 6.2 gives an overview of the data for each environment.

6.3.1.2 Metrics

In *ObChange* only YCB objects are annotated. Besides being possibly categorized as static, YCB objects are the only objects that change, i.e., are moved, removed, or novel. All other objects are static or move only slightly. They are considered background and are therefore irrelevant for object change detection. Given that, we apply the following metrics:

- *Detected objects*: The number and percentage of detected YCB objects.
- *Correctly categorized objects*: The number and percentage of YCB objects that are correctly categorized as static, moved, removed, or novel.
- *False positives*: The number of background objects that are not categorized as static plus the number of objects that are correctly classified, but parts of it have an incorrect category assigned.

6.3.1.3 Baseline

To the best of our knowledge, no other work exists to categorize objects into static, moved, removed, and novel by comparing environments at two time instances. Thus, in

Figure 6.4: 3D object map of detected objects using *Mask R-CNN* trained on COCO created with the approach from [103]. Besides some background objects, all YCB objects are detected, except for the gelatin box (marked with cyan rectangle).

order to evaluate our object mapping approach, we extend the recent work of Oliveira et al. [103]. Their proposed method detects objects with *Yolov3* [56] trained on the COCO dataset [104] in each frame. These object detections are used to create a 3D object map by temporal and spatial associations. We utilize this 3D object map to perform object matching. Thus, based on their work we develop two baselines. Instead of *Yolov3* we use *Mask R-CNN* [57], another state-of-the-art object detector because it is not only readily available for the established COCO dataset but also for the YCBV dataset [105]. We refer to the baselines as the *COCO-baseline* respectively the *YCBV-baseline*.

For both baselines, we create 3D object maps for all dataset recordings using the standard parameters mentioned in [103] with the following exceptions: (1) use *Mask R-CNN* instead of *Yolov3*, (2) decrease the distance threshold for the bounding box center distance to 0.0001 (which equals to 30 pixels) and decrease the spatial association to 0.3 m for better results. The RGB images from the recordings and the estimated poses from *ElasticFusion*, which are more accurate than pure robot poses, constitute the input for 3D object maps. For the *COCO-baseline* we do not include detected objects from the following COCO categories in the 3D object map because they cover classes not relevant for indoor object change detection: person, vehicle, outdoor, animal, furniture (except potted plant), and appliance. The example in Figure 6.4 shows the detected objects with their assigned classes overlayed on the reconstruction.

Based on the generated 3D object maps, we compute the changes between two visits to an environment using the assigned labels from the object detector. An object with a class label that exists in the visit at time t_0, but not at time t_1, is considered removed and vice versa as novel. If there is exactly one object of a specific object class in both visits, it is either categorized as static or moved depending on the distance between the object's bounding box centers compared to the threshold d. In the case that several objects from the same class are detected, we find an association by utilizing the feature vector of the second last layer of *Mask R-CNN*. This is inspired by Qiu et al. [106] who extract the features from the last layer as instance-level features. However, in

our experience features from the last layer are already too class-specific, whereas the second last layer is more suitable for instance comparisons. Because an object is usually detected in several frames, the feature vector is extracted for each. To find the best matching instances from the same class within two visits, we compute for an object at t_0 the dot-product between all its feature vectors with all feature vectors of all objects from the same class from t_1 then match the object that achieves the highest value. Depending on the distance between the two matched objects they are either categorized as static or moved.

The baseline using *Mask R-CNN* trained on the YCBV dataset [105] is evaluated to analyze the performance when provided a tailored training set in comparison to the more general COCO dataset. Park et al. [107] provide the weights for *Mask R-CNN*, which they trained for their pipeline to participate in the BOP challenge [101] on the YCBV dataset.

6.3.1.4 Evaluation

Our evaluation is based on manually labeled data. In *ObChange* all YCB objects are point-wise labeled in all surface reconstructions for all the recordings. Besides the object point indices, the object name is stored as ground truth data. The categories (static, moved, removed, novel) are extracted given two recordings and the ground truth: If the object name exists only in one of the two recordings, the object is novel or removed. If it occurs in both recordings, we compute the centroid of both objects and depending on the Euclidean distance assign the category static or moved.

Our method works directly on the surface reconstructions and therefore the resulting object points match directly to the labeled data and no further processing is needed. We consider an object as detected if at least 50 % of the points in the result overlap with the labeled points. Each point in the result has one of the four categories assigned. The object matching stage may erroneously assign different labels to data points from the same object due to imprecise region growing. Therefore, we use the maximum voted category per object and compare it against the ground truth. If the categories match, the object is correctly categorized. Otherwise, it counts as a wrongly categorized object. For the categories static and moved, the correct association of the two involved objects must be given to be counted as correct. For example, it is not enough that an object at time t_0 and t_1 is categorized as moved, also the association that the object at t_0 moved to the location of the object at t_1 must be given. The sum of false positives combines two different failure cases: (1) a YCB object where parts of it are wrongly classified. An object is therefore counted as correctly classified and at the same time, it is a false positive. (2) Points in the result that are categorized as moved, removed, or novel but do not belong to any YCB object are from a static background object. We cluster the points and each cluster is counted as one false positive object. Equally to [17] this metric is an approximation because no ground truth labeling for background objects exists.

In contrast, a point-wise evaluation is not possible for both baselines. Each detected object is described by a single 3D location and a label. Therefore, for each object in the ground truth, the centroid is computed and the closest object in the result is identified.

If the distance is less than ten centimeters we count it as a detected object, otherwise, it is a false positive. Further, we check for each detected object if there is a nearby ground truth object with the same category, which was not already matched. If so, the detected object is correctly categorized. The associations for moved and static detected objects must correspond to the ground truth. For the evaluation of the *YCBV-baseline*, we remove objects from the ground truth that do not appear in the YCBV dataset and are therefore not used for training.

We provide an additional evaluation based on the fact that if one of the two objects of a static or moved model-candidate pair is not detected, a subsequent failure may occur. For example, if a moved mug is detected at t_0, but not at t_1, it can never be categorized correctly as moved. Therefore, we re-evaluate the methods based on an adapted ground truth. Only detected objects are considered when re-computing the categorization for the ground truth objects. The correct categorization for the mug in the previously mentioned example would then be *removed*. This way we evaluate the categorization process stand-alone and independent of the preceding detection performance.

6.3.1.5 Results

A summary of the performance of our proposed method as well as the two baselines on *ObChange* is given in Table 6.3. Our method detects 91.8 % of all the labeled objects in the dataset while achieving the lowest number of false positive objects. Inspecting the detection rates of the two baselines, the result is surprising. Applying *Mask R-CNN* trained on the COCO dataset outperforms *Mask R-CNN* trained on the YCBV dataset by a significant margin although the ground truth objects in *ObChange* are selected from the YCB objects. Recently, Dhamija et al. [58] investigate state-of-the-art object detectors and their performance in open-world settings. They show that all methods detect objects from classes not presented during training with high confidence, despite the fact, that object detectors should only detect objects from known classes. This could explain the good performance of the *COCO-baseline*, although trained on different classes. We conjecture that the detection performance of the *YCBV-baseline* is only about 43 % because of two reasons: (1) although the objects used in *ObChange* are from the official YCB object set, some have a slightly different texture than the objects in the YCBV dataset used for training and (2) the training data is captured from a certain distance range, while the robot did not always get as close to the objects in the dataset. Therefore, we assume that the object detector overfits to the training data, which is a huge problem when applying it to images that are dissimilar to the training dataset. The detection rate could be increased by using a lower confidence threshold, however, this would also increase the number of false positives.

Our approach classifies 64.2 % of the detected objects correctly. This is 1.5× more than with the *COCO-baseline* and 3× more than with the *YCBV-baseline*. The difference between the results evaluated on the original ground truth and the adapted ground truth (*#Correctly categorized in detected objects*) is small for our approach because we achieve a high detection rate. For the *YCBV-baseline*, the difference is significant. Comparing the adapted ground truth, the *YCBV-baseline* slightly outperforms our approach in terms of correctly categorized objects (given that a substantially lower

Table 6.3: Results of the baseline trained on COCO and YCBV compared to the results of our method evaluated on *ObChange* and averaged per environment.

Result using *Mask R-CNN* trained on COCO dataset

Environment	#Total objects	#Detected objects	#Correctly categorized	#Correctly categorized in detected objects	#False positives
Big Room	425	340 (80.0 %)	117 (27.5 %)	130 (38.2 %)	218
Small Room	208	164 (78.8 %)	102 (49.0 %)	106 (64.6 %)	10
Living Area	108	72 (66.7 %)	61 (56.5 %)	65 (90.3 %)	20
Office Desk	100	64 (64.0 %)	50 (50.0 %)	55 (85.9 %)	64
Kitchen Counter	120	88 (73.3 %)	50 (41.7 %)	57 (64.8 %)	22
Overall Performance	961	728 (75.8 %)	380 (39.5 %)	413 (56.7 %)	334

Result using *Mask R-CNN* trained on YCBV dataset

Environment	#Total objects	#Detected objects	#Correctly categorized	#Correctly categorized in detected objects	#False positives
Big Room	215	95 (44.2 %)	48 (22.3 %)	75 (78.9)	219
Small Room	100	32 (32.0 %)	14 (14.0 %)	18 (56.3)	75
Living Area	56	20 (35.7 %)	12 (21.4 %)	18 (90.0)	22
Office Desk	40	24 (60.0 %)	18 (45.0 %)	24 (100)	36
Kitchen Counter	100	48 (48.0 %)	24 (24.0 %)	34 (70.8)	49
Overall Performance	511	219 (42.9 %)	116 (22.7 %)	**169 (77.2 %)**	401

Result of our approach

Environment	#Total objects	#Detected objects	#Correctly categorized	#Correctly categorized in detected objects	#False positives
Big Room	425	419 (98.6 %)	286 (67.3 %)	290 (69.2 %)	66
Small Room	208	183 (88.0 %)	131 (63.0 %)	136 (71.6 %)	13
Living Area	108	92 (85.2 %)	75 (69.4 %)	78 (84.48 %)	10
Office Desk	100	88 (88.0 %)	76 (76.0 %)	78 (88.6 %)	24
Kitchen Counter	120	100 (83.3 %)	49 (40.8 %)	56 (56.0 %)	28
Overall Performance	961	**882 (91.8 %)**	**617 (64.2 %)**	638 (72.3 %)	**141**

number of objects is detected in the first place). However, it has significantly more false positives even though it is specifically trained for the objects in the dataset.

The performance of our approach and the baselines for each YCB object used in the evaluation is shown in Figure 6.5. It shows that the smallest object in the dataset, the large marker, cannot be localized by any method. However, our method was able to detect other small objects such as screwdrivers. It can be seen that our method has difficulties detecting a plate because a significant amount of points is explained by the supporting plane model and are within the distance threshold d_{plane}. Interestingly, for all objects from the YCB dataset, the localization performance of the *COCO-baseline* is superior compared to the *YCBV-baseline*. The lemon is the only object where our approach performs significantly worse in categorizing than the baseline. The reason is that the lemon is small as well as part of a pile in most of the recordings. The wrong categorizations of the master chef can are the result of incomplete object reconstructions and the confusion with the pitcher, which has a very similar appearance.

Figure 6.6 shows for each environment how many of the detected objects are correctly

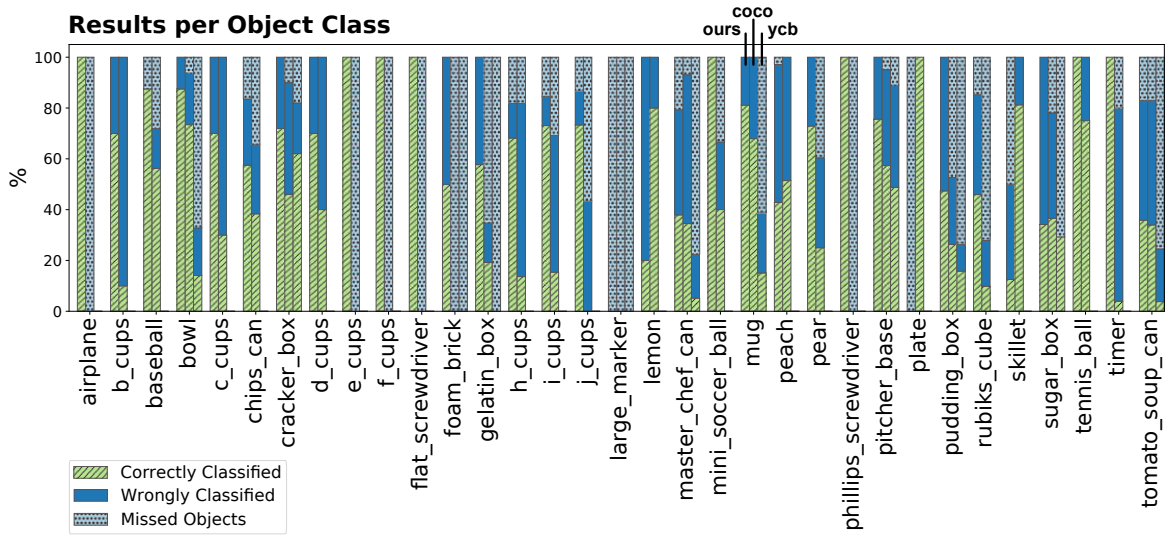


Figure 6.5: Overview of YCB objects used in the dataset. The performance for each object is shown for our approach (first bar), the *COCO-baseline* (second bar), and the *YCBV-baseline* (third bar). If the object is not in the YCBV dataset, no bar for the *YCBV-baseline* is visualized.

classified (#Correctly categorized in detected objects) split into the four categories static, moved, removed, and novel. For each predicted category, the absolute numbers as well as the relative numbers are given. The column *wrong matching* is only relevant for the static and moved categories. It indicates cases where the objects have the correct class assigned, but the association between object pairs is wrong. The correlation matrices show that removed and novel objects are correctly categorized with a high probability. The association of static objects, which is usually found by local matching using ICP, also achieves reasonable performance. Important to notice that for the living area none of the static objects was from the YCB dataset and only YCB objects are annotated in *ObChange* and are therefore applicable for the quantitative evaluation. The below-average performance in the kitchen counter environment, especially for static and moved objects, stems from the more cluttered recordings and the resulting unclear object boundaries in the reconstructions, which poses a problem for object separation using region growing.

6.3.2 Robot Experiments

For the real-world robot experiments, we use a Toyota Human Support Robot in an environment with nine surfaces and conduct three runs with different object changes. Figure 6.7 presents the reconstruction of the environment using *voxblox*. It also shows the planes that are automatically extracted by exploiting semantic labeling as outlined in Section 6.2.3. For each surface, viewing poses for the robot are generated at a distance of 20 cm to the convex hull of the plane, oriented towards the plane center, and evenly distributed every 30 cm around the circumference. To create the reference map

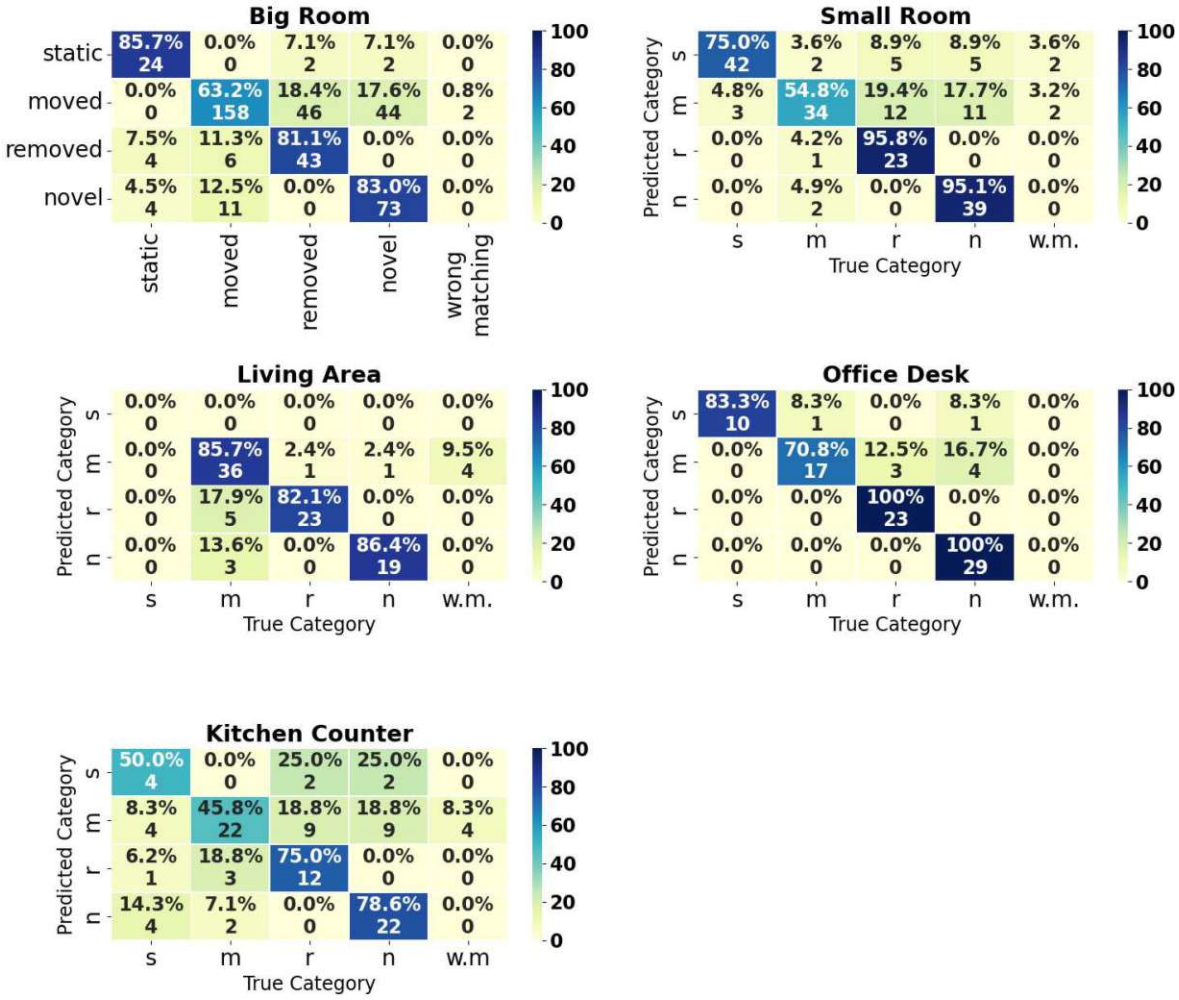


Figure 6.6: Correlation matrices for each environment showing the relative and absolute categorization results.

at t_0 (used for all three comparison runs), the robot visits each surface and stores the detected objects in the database. 20 out of 21 objects placed on the extracted planes are detected. The object set and their locations then partially change for the three runs at times t_1 to t_3.

Table 6.4 presents the results. For each run at t_1, t_2, and t_3 we give the numbers of the objects assigned to the four categories compared to the ground truth. Missed objects are those that were not detected at all and therefore result in incorrectly categorized objects. The table also shows the number of false positive objects as defined in Section 6.3.1.2. In our experiments, the difference in the numbers of the last column between our approach and the ground truth stems from the sum of objects that were not detected (*#Missed)* and from wrongly matched objects. In total 70 objects ($\sum \#Correct(GT) - \sum \#Removed(GT)$) were placed in the three runs, which are compared to the reference run from t_0. In summary, most of the objects are detected (65 of 70) and from the detected ones 61 objects are correctly categorized (61 of 65).

Table 6.4: Results of mapping objects from the robot experiments. The rows with GT refer to ground truth for the respective runs 1, 2, and 3 at t_1, t_2, and t_3.

Run	#Static	#Moved	#Removed	#Novel	#Missed	#FPs	#Correct
1	17	2	1	4	1	0	24
GT	18	2	1	4	0	0	25
2	12	5	2	3	1	0	22
GT	13	6	2	3	0	0	24
3	5	10	1	3	3	2	19
GT	8	12	1	4	0	0	25

Including the removed objects and considering the missed objects as wrongly classified, 65 of 74 objects are correctly mapped (last column in the table). For each run, the detection and matching part took approximately 10 minutes on a standard laptop.

Figure 6.8 gives details about the matches. In the setup phase as well as in each visit of the environment, a wooden box standing in the corner of the shelf was not detected because it was bigger than the shelf depth and therefore not within the convex hull (marked with a red cross in Figure 6.8). In run 1 all detected objects are matched correctly. In run 2, the pringles can is misclassified when comparing the environment at time t_0 and t_2. At t_0 the pringles can is close to a tea box, which is correctly categorized as moved, but region growing fails to stop at the object boundaries. Therefore, the two objects are considered as one object and are assigned the same label. This leads to the error that the pringles can at t_2 is wrongly labeled as novel. At time t_3 the keyboard is partially occluded when the robot moves around the table. Therefore, the reconstruction is only a planar surface, which is not considered an object. As a consequence, the keyboard from time t_0 is matched to the keyboard of a laptop (labeled as M11 in Figure 6.8). The other wrongly categorized objects result from the inability to split the orange and blue objects on the table in the middle of the room. Although for both objects PPF found the correct match and the confidence is the highest compared to other possible matches, it is still too low to accept the match. The reason is the low geometric overlap because the two objects are only partially reconstructed at t_3 but almost a full model exists from t_0.

6.3.3 Discussion

Based on the evaluation results using *ObChange* and the robot experiments, in the following sections, we highlight the findings on remaining open challenges due to factors such as robot localization, covering surfaces with views, the detection of small objects, partially occluded objects, and quality of reconstructions.

6.3.3.1 Robot Localization Error

Precise robot localization is necessary for most change detection applications. It supports the pairing of frames as well as the creation of clean reconstructions. In our experience, the performance of dense visual SLAM methods in environments with areas

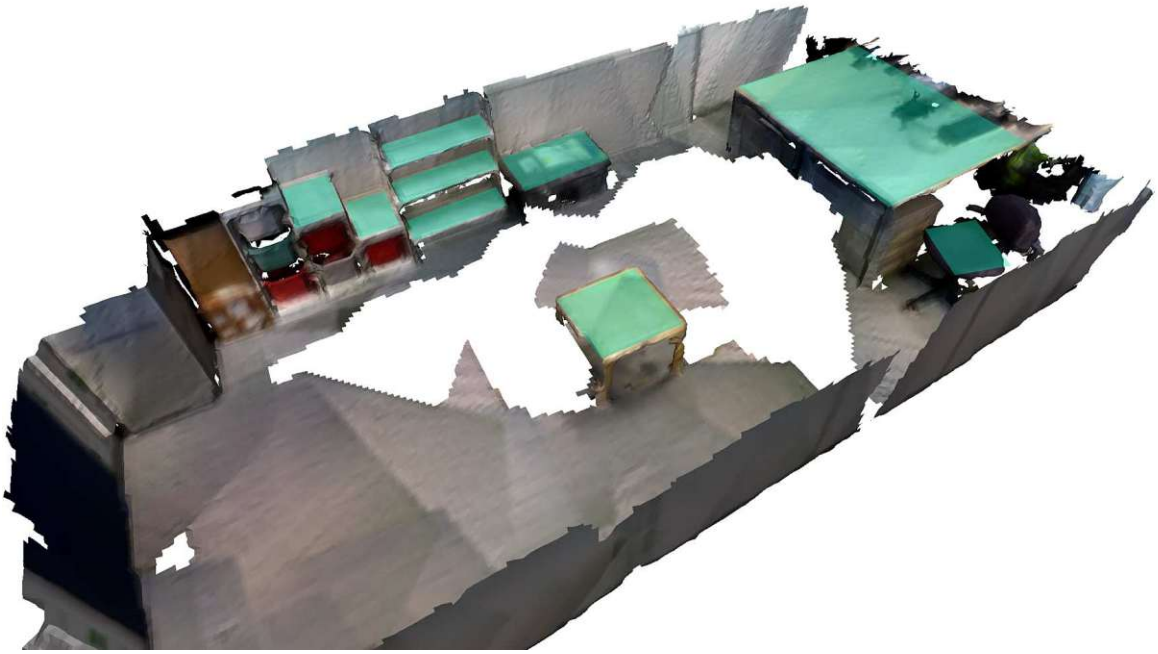


Figure 6.7: Reconstruction of the environment used for the real robot experiments. The detected planes are highlighted in turquoise.

of little visual and geometric features greatly benefits from integrating odometry data assuming state-of-the-art localization. Our solution to integrate odometry data into the *ElasticFusion* framework could reconstruct all surfaces in *ObChange* because the robot drift is insignificant. In case this cannot be guaranteed, the recent work by Houseago et al. [108] shows how to integrate odometry data and dense visual SLAM in the presence of drift. Although their method could be integrated into our system, we opted for a computationally cheap approach. This is important when working with mobile robots, where limited resources have to be shared among different system components.

6.3.3.2 Search Space

Detecting objects is a trade-off between reducing the search space and not missing objects (examples are shown in Figure 6.9). At the same time the chance to detect false positives increases with a less restricted search space. The concept of surfaces reduces the search space significantly but may lead to missed objects if the surface plane is occluded or if it is not nearly planar. We decrease the number of false positive detections by utilizing a shrunk convex hull of the plane but this may cause objects at the border to not be detected.

6.3.3.3 Detection of Small Objects

The detection of small objects is a well-known problem. Especially when working with 3D maps, actions taken to reduce the effect of noise and misalignments are the reason for

filtering small objects. Our evaluation shows that neither the learning-based approaches nor the surface concept approach are capable of reliably detecting small objects.

6.3.3.4 Occlusion

Occlusion poses a significant challenge for object matching because parts of an object are missing. The same accounts for incomplete reconstruction due to view limitations. Generally, ambiguities due to partial reconstructions or truly similar object appearance(see Figure 6.10 first two examples) need to be counter-measured by taking into account as much information as possible – for example, color PPF [109] could provide better a-priori hypotheses at the cost of higher computational efforts.

6.3.3.5 Object Matching Verification

A verification step to check if a match is physically plausible, similar to [36] or [110] could help in some cases to reject inconsistent matches. However, this method of verification does not work in all cases – approaches fail when objects are partially visible or symmetric. For example, a tube-like object standing on a table was detected at t_0 ($tube_0$) and at t_1 ($tube_1$), but only the bottom half was visible in the latter. The object matching and registration is ambiguous and resulted in aligning $tube_1$ to the top of $tube_0$. For verification purposes, projecting $tube_0$ into the t_1-recording results in half of the tube being below the supporting plane, which is physically implausible. This triggers a rejection even though the matched objects are the same. More generally, the fact that detected objects can actually be a heap poses challenges in geometric and semantic verification.

6.3.3.6 Reconstruction Quality

Clearly, the quality of the surface reconstruction and therefore the 3D object models impacts the performance of object matching applications. While *ElasticFusion* tends to create reconstructions with smoothed normals, it also connects spatially close objects with additional points having a continuous color gradient. While this kind of reconstruction is appealing and preferred, for example, in video games, we prefer a more realistic depiction of the environment. This is especially relevant when performing processing steps such as matching or region growing, which favor distinctive geometric features.

6.4 Conclusion

In this work, we tackled the core perception capabilities for open-world operation in the context of the object mapping task. This was defined as comparing the locations of objects in the present visit (present object map) to a previously stored reference map. Without loss of generality, we proposed to create a 3D reference map of the environment. Comparing complete environment reconstructions is impractical for many application scenarios, therefore, we presented a concept where only local surfaces are reconstructed. This has the benefit that local comparisons are performed more rapidly and that local

surfaces are reconstructed more accurately. The latter is critical to enhance object detection and matching results. We showed that semantic segmentation methods are suitable to autonomously provide high-level partitioning into relevant surfaces.

The key step of object mapping is to compare object detections and match objects from two different time instances. The main contribution of this work is the perception of all possible cases of static, moved, removed, and novel objects. For the quantitative evaluation, we used the *ObChange* dataset and compared the proposed approach against two baselines using state-of-the-art learning-based methods. Results indicate that the proposed method significantly improves over the baselines in terms of detected objects as well as the accuracy of categorization. Furthermore, we conducted real-world experiments with an autonomous mobile robot to demonstrate our developed capabilities in a realistic setting.

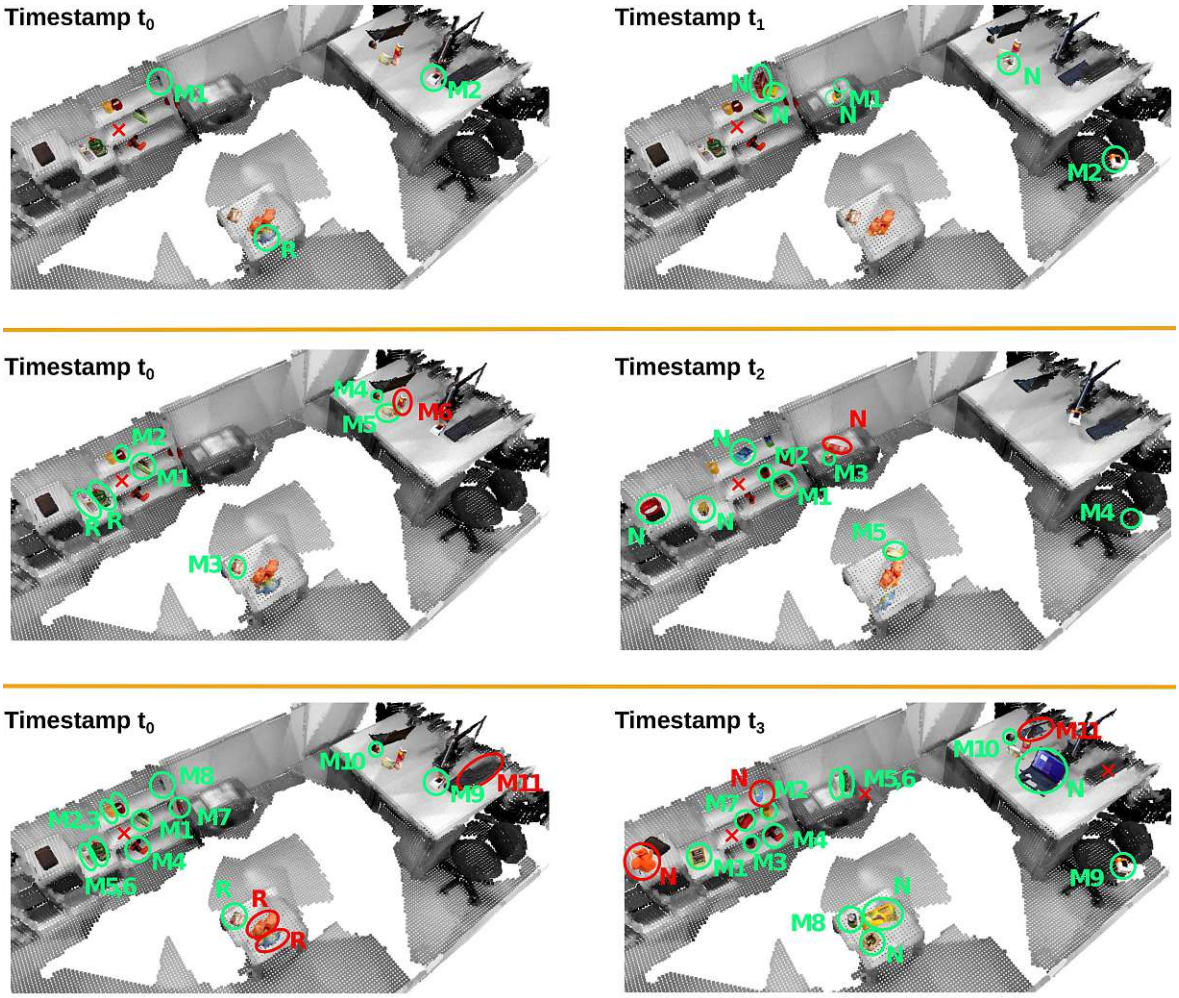


Figure 6.8: Results of real-world robot experiments. Each row shows the comparison of the reference environment (first column) with the state of the same environment at a different time (second column). Each detected object is marked with an ellipse and labeled with M (moved), R (removed), or N (novel). Correct categorizations are colored in green, wrong ones in red. Objects, which got matched, have the same number assigned. For simplicity static objects are not highlighted. Missed objects are marked with a red cross.

Figure 6.9: Examples of missed objects. Extracted planes are visualized in red. First image: two objects in black rectangle. Only the yellow cup is detected because the couch seat is too curved. Second and third image: only few object points are within the convex hull of the plane and therefore classified as noise and rejected. Note, that in the third example the reconstruction is very sparse because the robot did not get close enough to the surface.

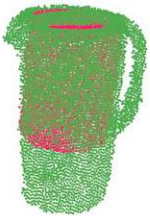
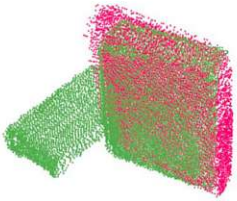
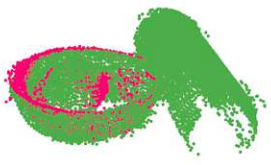
Model	Candidate	Hypothesis	Description
			Wrong match. Color and geometry too similar. Hypothesis was accepted.
			Correctly matched, but model AND object confidence too low because of little geometric overlap. Hypothesis was rejected.
			Correctly matched but *region growing* merges pringles can. Hypothesis was accepted.

Figure 6.10: Examples of hypotheses generated based on PPF.

Chapter 7

Conclusion

Autonomous robots are a particularly good fit for repetitive and tedious tasks. In the future, households will benefit from the co-existence of helper robots undertaking more complex tasks than they do nowadays. According to the survey conducted by Bugmann and Copleston [93], people wish for service robots to fulfill tasks, which require interactions with objects, such as filling and emptying the dishwasher, preparing food, and tidying up.

Although progress is made in this direction, it is still a long way to safely deploy autonomous mobile robots that need to be able to understand their surroundings in the ever-changing real world. To develop their full potential it is necessary that service robots are able to adapt to new situations without the need for tedious manual adjustment or even worse, without the need for a technical expert. For example, a robot needs to recognize when the environment has changed and therefore initiates the process to update the stored maps or it needs to detect new objects in its surroundings and subsequently has to learn their appearance and meaning.

When a robot has to deal with objects, detecting them is the first step. It is usually followed by methods to extract the semantic meaning, estimate the object pose, and manipulate the object. In this thesis, we described methods to lay the foundations for robots to detect objects in open-world scenarios by comparing the current situation to a stored reference. This concept enables a robot to detect unknown objects that it has never seen before. While the first two methods categorize all detected objects as novel, the third method distinguishes between static, moved, removed, and novel. These four cases cover all possible states an object in an environment compared to a reference can have. In this chapter we give a summary of the proposed object change detection methods (Section 7.1) and insights into potential improvement and future research directions (Section 7.2).

7.1 Summary

This thesis encompasses three methods to detect objects using the change in the environment between two timestamps as a cue. The methods differ in the geometric representation of the environment, where objects are detected, and the capability to distinguish between static, moved, removed, and novel objects. Additionally, in

Chapter 3 we present *ObChange*. It is a new dataset, which is suitable for quantitatively evaluating object change detection methods. In contrast to existing datasets, the data were acquired with a robot in real environments and with a focus on small objects. We recorded RGB and depth streams from five different environments where a selection of YCB objects was introduced – in total 26 recordings with 260 YCB objects plus a tidied-up version of each environment. Two versions of the dataset are available with all YCB objects annotated. One version consists of full-environment reconstructions created with *voxblox*. The other version divides an environment into meaningful parts and uses *ElasticFusion* to map them, which leads to a more detailed representation.

The first method to detect objects is described in Chapter 4 and builds on *OctoMaps*, which is a voxel-based map representing the environment. The aim is to find all novel objects placed on the floor. This means that the corresponding location in the reference map is unoccupied. The method compares a stored reference *OctoMap* against the currently available one, which is built on-the-fly while the robot fulfills a task and navigates in the environment. An *OctoMap* inherently stores information if a voxel is unoccupied or hasn't been seen yet. Therefore, the planning system can request a change detection at any time. It is not necessary that the whole environment is mapped before, but obviously, objects are only found in the part, which has been seen already. The detected differences get clustered and filtering rules are applied depending on the task and the robot design. We showed in real-world experiments that the presented approach reliably finds regions of interest, which contain one or several novel objects. Our method significantly outperforms a method based on fixed viewpoints with regard to the number of segmentation actions and the time needed to find and segment all novel objects in the environment. Nevertheless, this method is sensitive to dynamic environments and detects even slightly moved furniture partially as novel objects.

Hence, in Chapter 5 we present and discuss the idea of utilizing semantic information about the environment to perform more robust change detection. Instead of computing a global difference, relevant horizontal planes are taken into account separately. Clusters on top of these planes are considered as potential interesting objects. To detect objects that are new at a location, which was unoccupied compared to a reference map, a two-step approach is applied. First, the corresponding planes from two timestamps are aligned and then for each currently present object cluster placed on the plane we use ICP (with a fixed upward direction) to find a transformation that results in a high overlap with a spatially close cluster in the reference. Object clusters with a low overlap are considered novel at this location. We evaluated the proposed approach and its intermediate steps and compared it to global change detection methods using the dataset described in Chapter 3. The results show that our method outperforms the baselines, especially in terms of precision while also achieving a high recall and detecting most of the novel objects.

In Chapter 6 we adopt the idea of using semantic segmentation to divide the environment into meaningful parts and create a reconstruction for each of them. The short trajectories and combining the robot and camera poses lead to a more precise representation with fewer artifacts. This is crucial to be able to differentiate between static, moved, removed, and novel objects when comparing an environment at two different timestamps. These four cases cover all possible states an object in the envi-

ronment compared to a reference can have. We match detected objects on horizontal planes based on color and geometric similarity. First, spatially close object clusters are compared using ICP. If there is a match, the objects are static. Otherwise, a PPF descriptor is used to match objects across the whole supporting plane and if there is no match all remaining surfaces are checked. If no match is found in the whole environment the object is categorized as removed respectively novel, otherwise as moved. The method enables the disentanglement of heaps if objects are moved or removed at another timestamp. The quantitative evaluation is performed on the *ObChange* dataset and the proposed method is compared against learning-based object detection methods. The results show that it outperforms even the baseline with the tailored training set. Finally, we demonstrate the applicability of the approach with real-world robotic experiments.

7.2 Outlook

To conclude, we discuss remaining open challenges risen from experiments and evaluations (Section 7.2.1). Additionally, the last section outlines possible extensions, which build on the object detection methods described in the thesis (Section 7.2.2).

7.2.1 Improving Object Reconstruction and Detection

Improving the reconstruction quality in terms of accuracy as well as completeness boosts the result of all perception-related tasks. One way to generate a more consistent coloring across a reconstruction of the environment is the integration of the work from Alexandrov et al. [111] with the reconstruction method. It reduces the vignetting effect and therefore leads to better results in subsequent steps, such as object matching where color is used in one part of the matching score equation.

Another open challenge is the detection of small objects and objects that are partially hidden, where in both cases multiple views, as well as close-up views, might be a future solution to work on. Also, active manipulation to separate heaps or generate different views is a promising direction [19].

Deformable objects also pose a challenge for perception algorithms. While the proposed methods detect rigid as well as non-rigid objects, subsequent steps like object matching, recognition, or pose estimation suffer from the different shapes an object can take. A possible solution is to divide objects into meaningful parts and build new approaches based on that.

Finally, another important future research direction includes reflective, refractive, and transparent objects. RGB-D cameras provide invalid or noisy depth data for these kinds of objects. Therefore, it is not possible to map objects in reconstructions and subsequently detect them. Initial approaches exist on how to estimate missing depth data [112].

7.2.2 Expanding on Object Detection Results

Considering an autonomous robot fulfilling a task, such as tidying up or fetch-and-carry as described at the beginning of the thesis, several different components need to be developed and linked together. Navigation, planning, perception, and manipulation are essential parts of such a complex robotic system. In Chapter 6 we integrated some of these components to discuss the performance of the presented method using a robot in the real world. A state machine is responsible for coordinating the robot's navigation and invoking the object detection and matching method. Generally, the object detection methods described in this thesis represent an important initial step at the object perception level. However, more actions are required on top of the presented work to constitute a full working system that handles open-world situations and learns about new objects. The results of the presented methods serve as a starting point for potential future steps. The robot is now able to position itself and the camera in a way that the object is accurately in the field of view. Especially for a novel object in the environment, the robot needs to retrieve more information about it to handle it correctly. There are several options: (1) The robot presents an image of each novel object to a human via a screen where the human then enters characteristics such as object class, storage location, owner, and so on. (2) An image of a novel object is used as a query image to find similar ones on the web where information gets extracted from surrounding text. (3) A classifier does not need a model of each object instance but abstracts general features to assign objects to a class. However, when objects of unknown classes appear the classifier needs to be retrained.

While the thesis focuses on the key aspect of perceiving all object changes, object pose estimation needs to be further integrated with grasping. For these and other subsequent steps like object recognition, full 3D object models are beneficial. Assuming changes occur regularly the burden of manually creating them can be circumvented by merging partially detected objects into a complete 3D model as proposed by Furrer et al. [36].

Another interesting extension, especially for the fetch-and-carry task, is to keep track of where objects have been seen in the past. This knowledge is useful to compute probabilities where objects are likely located. It reduces the amount of time needed to find a requested object when the robot first navigates to locations with a high probability.

Appendix

A.1 Examples from the Object Change Detection Dataset

The following subsections show an exemplary recording of each environment of the novel object change detection datasets described in Section 3.2. In each figure (Figure A.1.1–A.1.6) the full-environment reconstruction is shown on the top. It is created with *voxblox* and used for the evaluation in Chapter 5. At the bottom of each figure surfaces of interest from *ObChange* are arranged. We only present those which include objects from the YCB dataset [77]. *ObChange* is generated with an adapted version of *ElasticFusion* [28] and is used for the quantitative evaluation in Chapter 6. The proposed semantic splitting of the environment into regions containing supporting planes results in shorter camera trajectories. This leads to more accurate reconstructions in *ObChange* compared to the full-environment ones.

A.1.1 Small Room

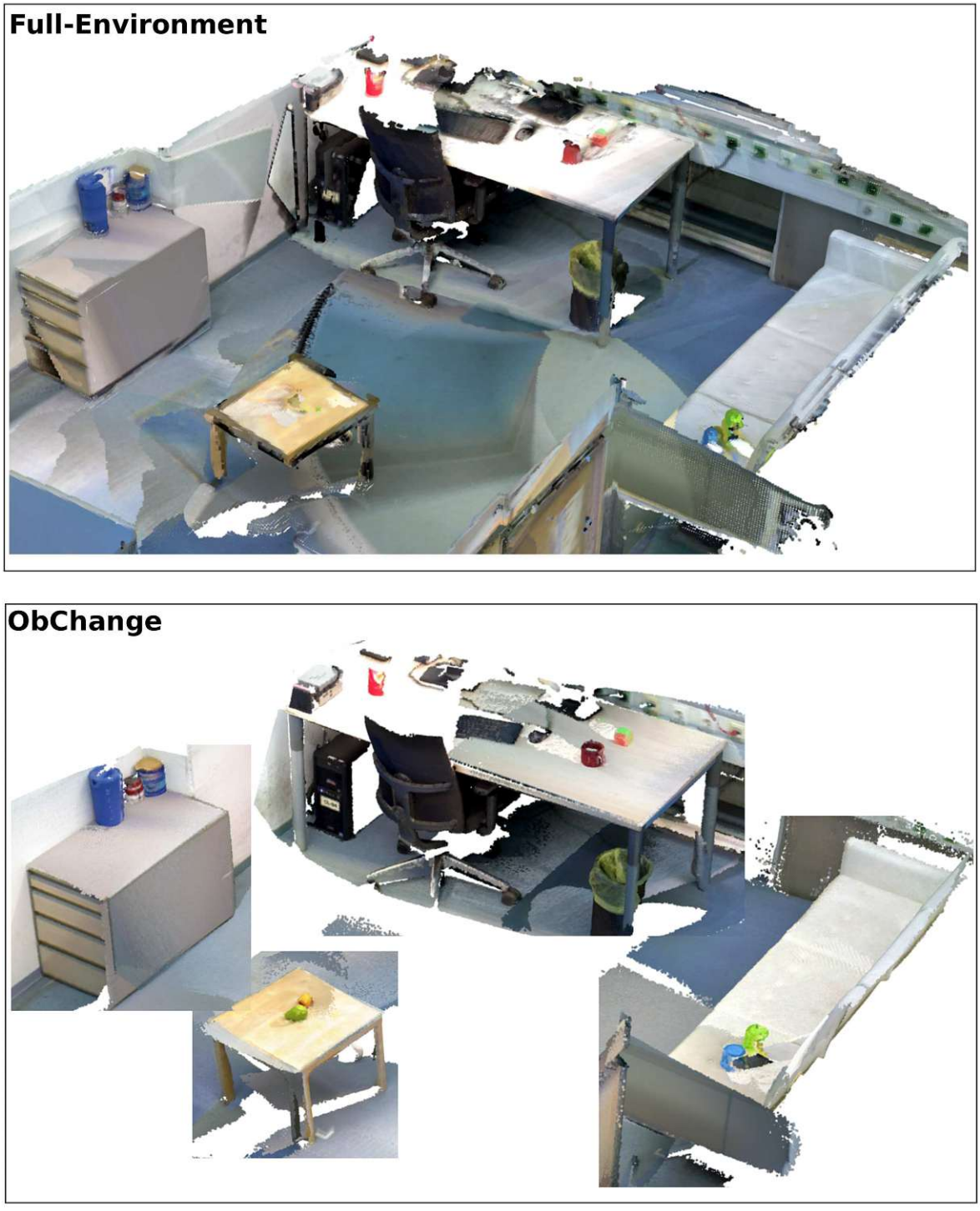


Figure A.1.1: Small room from the object change detection dataset (Section 3.2).
Top: full-environment reconstruction created with *voxblox*.
Bottom: selected surfaces of interest from *ObChange*.

A.1.2 Living Area

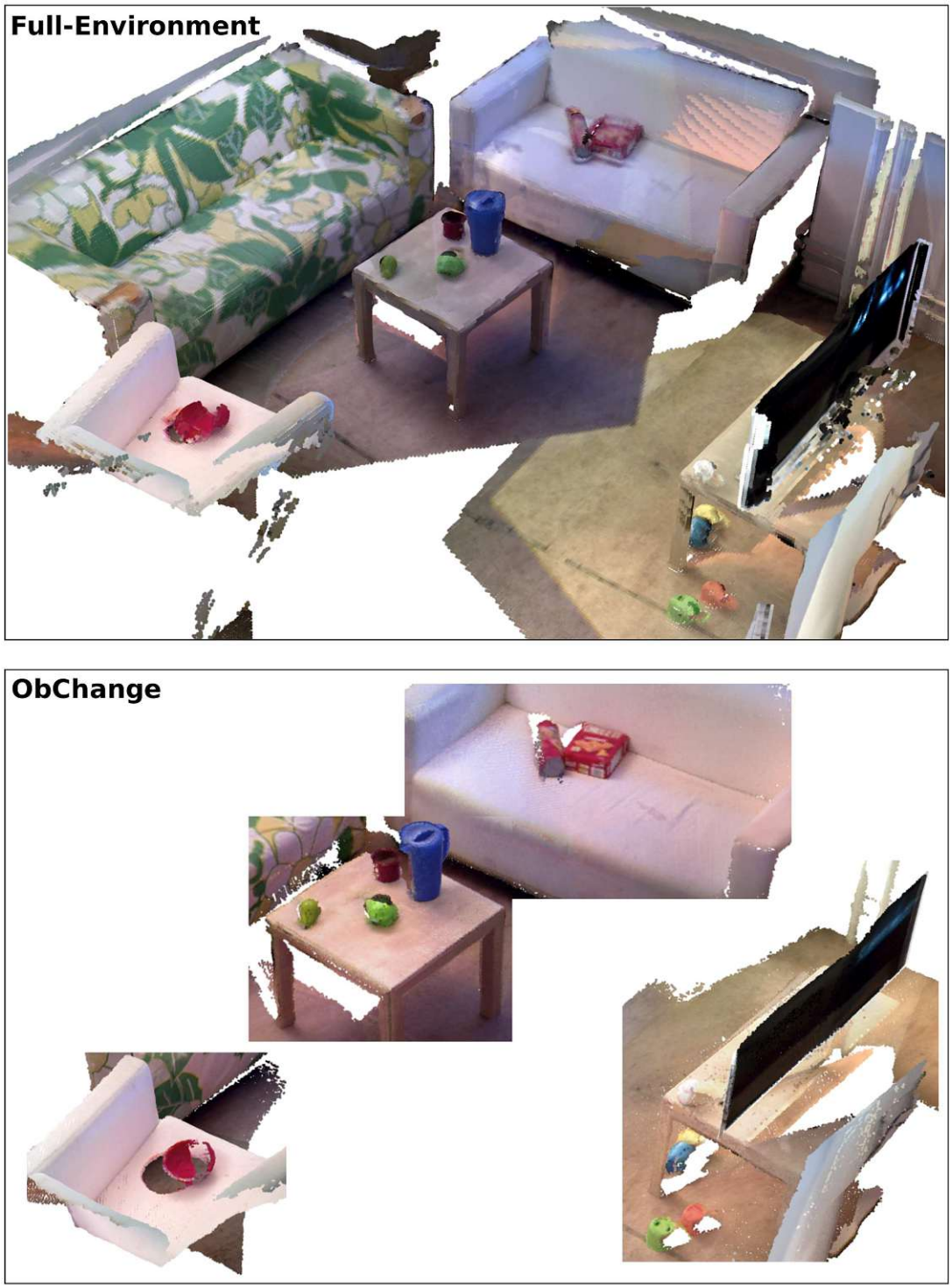


Figure A.1.2: Living area from the object change detection dataset (Section 3.2). Top: full-environment reconstruction created with *voxblox*. Bottom: selected surfaces of interest from *ObChange*.

A.1.3 Office Desk

Figure A.1.3: Office desk from the object change detection dataset (Section 3.2).
Top: full-environment reconstruction created with *voxblox*.
Bottom: selected surfaces of interest from *ObChange*.

A.1.4 Kitchen Counter

Figure A.1.4: Kitchen counter from the object change detection dataset (Section 3.2). Top: full-environment reconstruction created with *voxblox*. Bottom: selected surfaces of interest from *ObChange*.

A.1.5 Big Room - Part 1

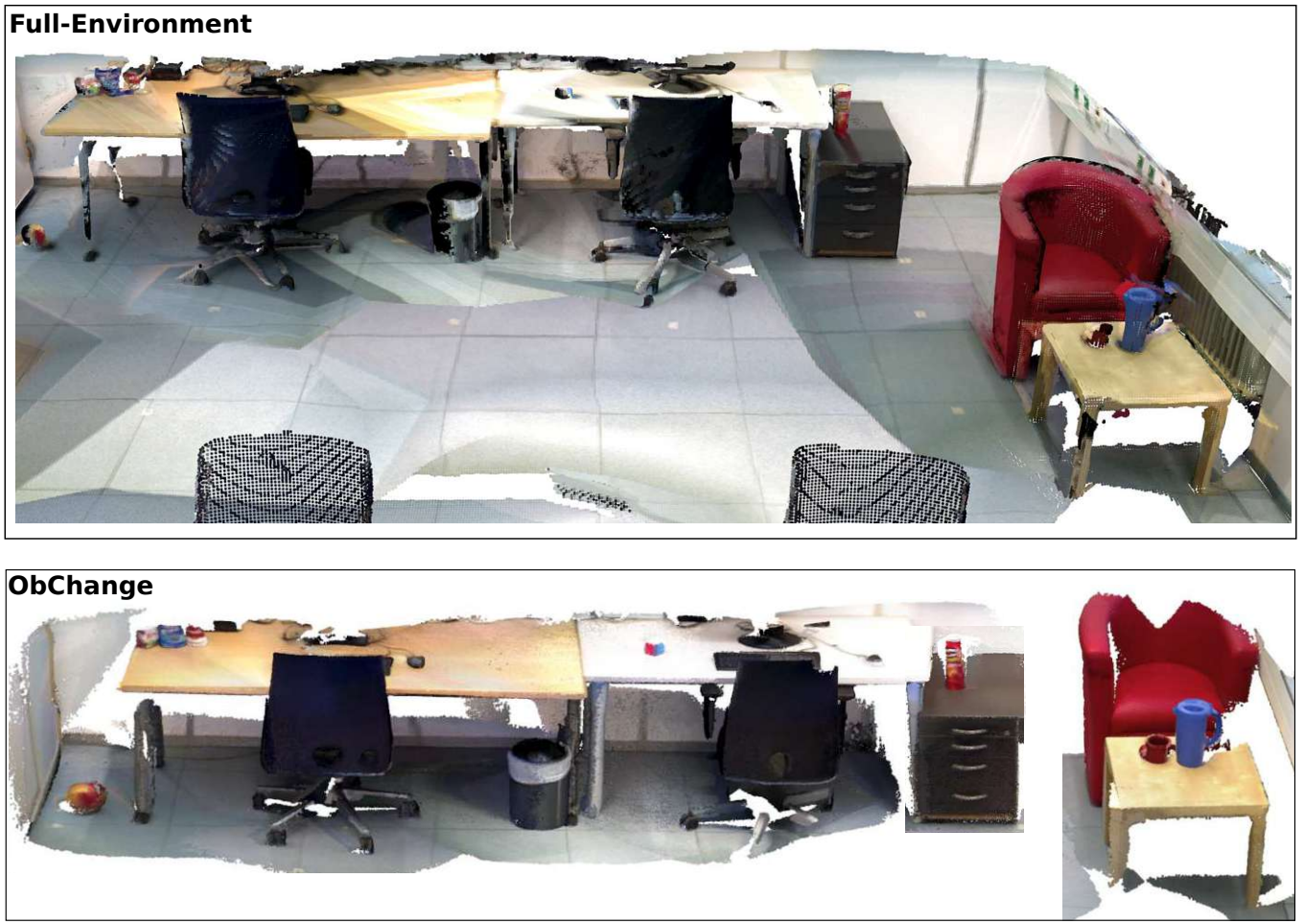


Figure A.1.5: One part of the big room from the object change detection dataset (Section 3.2). See Figure A.1.6 for the other part. Top: full-environment reconstruction created with *voxblox*. Bottom: selected surfaces of interest from *ObChange*.

A.1.6 Big Room - Part 2

Figure A.1.6: One part of the big room from the object change detection dataset (Section 3.2). See Figure A.1.5 for the other part. Top: full-environment reconstruction created with *voxblox*. Bottom: selected surfaces of interest from *ObChange*.

Bibliography

[1] B. Xu, W. Li, D. Tzoumanikas, M. Bloesch, A. Davison, and S. Leutenegger, "MID-fusion: Octree-based object-level multi-instance dynamic SLAM," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 5231–5237 (cit. on p. 1).

[2] C. Gomez, A. C. Hernandez, E. Derner, R. Barber, and R. Babuška, "Object-based pose graph for dynamic indoor environments," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5401–5408, 2020 (cit. on p. 1).

[3] S. Krivic, M. Cashmore, D. Magazzeni, S. Szedmak, and J. Piater, "Using machine learning for decreasing state uncertainty in planning," *Journal of Artificial Intelligence Research*, vol. 69, pp. 765–806, 2020 (cit. on p. 1).

[4] K. Joseph, S. Khan, F. S. Khan, and V. N. Balasubramanian, "Towards open world object detection," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 5830–5840 (cit. on pp. 1, 16, 17).

[5] D. Kim, T.-Y. Lin, A. Angelova, I. S. Kweon, and W. Kuo, "Learning open-world object proposals without learning to classify," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5453–5460, 2022 (cit. on pp. 1, 16).

[6] P. Perera, V. I. Morariu, R. Jain, V. Manjunatha, C. Wigington, V. Ordonez, and V. M. Patel, "Generative-discriminative feature representations for open-set recognition," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 814–11 823 (cit. on pp. 1, 16).

[7] Z. Liu, Z. Miao, X. Zhan, J. Wang, B. Gong, and S. X. Yu, "Large-scale long-tailed recognition in an open world," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 2537–2546 (cit. on pp. 1, 16).

[8] D. Fischinger, P. Einramhof, K. Papoutsakis, W. Wohlkinger, P. Mayer, P. Panek, S. Hofmann, T. Koertner, A. Weiss, A. Argyros, *et al.*, "Hobbit, a care robot supporting independent living at home: First prototype and lessons learned," *Robotics and Autonomous Systems*, vol. 75, pp. 60–78, 2016 (cit. on p. 2).

[9] M. Bajones, D. Wolf, J. Prankl, and M. Vincze, "Where to look first? Behaviour control for fetch-and-carry missions of service robots," in *Proc. of the Austrian Robotics Workshop*, 2014, pp. 500–516 (cit. on p. 2).

[10] N. Hawes, C. Burbridge, F. Jovan, L. Kunze, B. Lacerda, L. Mudrova, J. Young, J. Wyatt, D. Hebesberger, T. Kortner, *et al.*, "The STRANDS project: Long-term autonomy in everyday environments," *IEEE Robotics & Automation Magazine*, vol. 24, no. 3, pp. 146–156, 2017 (cit. on p. 2).

[11] M. R. Loghmani, B. Caputo, and M. Vincze, "Recognizing objects in-the-wild: Where do we stand?" In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 2170–2177 (cit. on p. 3).

[12] E. Herbst, P. Henry, and D. Fox, "Toward online 3-D object segmentation and mapping," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 3193–3200 (cit. on pp. 3, 4, 12, 13, 15).

[13] J. Mason and B. Marthi, "An object-based semantic world model for long-term change detection and semantic querying," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 3851–3858 (cit. on pp. 3, 7, 12, 13, 18, 19, 25, 34, 41).

[14] R. Ambrus, N. Bore, J. Folkesson, and P. Jensfelt, "Meta-rooms: Building and maintaining long term spatial models in a dynamic world," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014, pp. 1854–1861 (cit. on pp. 3, 4, 7, 12–14, 19, 44, 46, 48, 50).

[15] S. Song, L. Zhang, and J. Xiao, "Robot in a room: Toward perfect object recognition in closed environments," *CoRR, abs/1507.02703*, 2015 (cit. on pp. 3, 12, 15).

[16] M. Fehr, F. Furrer, D. Ivan, J. Sturm, I. Gilitschenski, R. Siegwart, and C. Cadena, "TSDF-based change detection for consistent long-term dense reconstruction and dynamic object discovery," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017 (cit. on pp. 3, 4, 12–14, 18, 19, 46).

[17] E. Langer, T. Patten, and M. Vincze, "Robust and efficient object change detection by combining global semantic information and local geometric verification," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 8453–8460 (cit. on pp. 3, 12, 15, 64, 67).

[18] A. Ayvaci and S. Soatto, "Detachable object detection: Segmentation and depth ordering from short-baseline video," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 34, no. 10, pp. 1942–1951, 2011 (cit. on p. 3).

[19] T. Patten, M. Zillich, and M. Vincze, "Action selection for interactive object segmentation in clutter," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 6297–6304 (cit. on pp. 3, 80).

[20] C. Xie, Y. Xiang, Z. Harchaoui, and D. Fox, "Object discovery in videos as foreground motion clustering," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 9994–10 003 (cit. on p. 3).

[21] X. Chen, S. Li, B. Mersch, L. Wiesmann, J. Gall, J. Behley, and C. Stachniss, "Moving object segmentation in 3D LiDAR data: A learning-based approach exploiting sequential data," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6529–6536, 2021 (cit. on p. 3).

[22] M. Björkman and D. Kragic, "Active 3D scene segmentation and detection of unknown objects," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 3114–3120 (cit. on pp. 4, 56).

[23] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, "Close-range scene segmentation and reconstruction of 3D point cloud maps for mobile manipulation in domestic environments," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2009, pp. 1–6 (cit. on pp. 4, 41, 44).

[24] J. Wald, A. Avetisyan, N. Navab, F. Tombari, and M. Nießner, "RIO: 3D object instance re-localization in changing indoor environments," in *Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 7658–7667 (cit. on pp. 4, 13, 18–20, 38).

[25] E. Herbst and D. Fox, "Toward object discovery and modeling via 3-D scene comparison," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 2623–2629 (cit. on pp. 4, 12–15).

[26] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013 (cit. on pp. 6, 11, 25, 48).

[27] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3D Euclidean signed distance fields for on-board MAV planning," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 1366–1373 (cit. on pp. 6, 11, 21, 46, 56).

[28] T. Whelan, S. Leutenegger, R Salas-Moreno, B. Glocker, and A. Davison, "ElasticFusion: Dense SLAM without a pose graph," in *Proc. of Robotics: Science and Systems (RSS)*, 2015 (cit. on pp. 6, 12, 22, 49, 52, 57, 82).

[29] 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*, vol. 7, no. 4, pp. 9667–9674, 2022 (cit. on pp. 7, 12, 14).

[30] S. Schreiberhuber, J. Prankl, T. Patten, and M. Vincze, "ScalableFusion: High-resolution mesh-based real-time 3D reconstruction," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 140–146 (cit. on pp. 11, 49, 52).

[31] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *Proc. of the IEEE International Symposium on Mixed and Augmented Reality*, 2011, pp. 127–136 (cit. on p. 11).

[32] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. J. Leonard, and J. McDonald, "Kintinuous: Spatially extended KinectFusion," in *Proc. of RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, 2012 (cit. on pp. 11, 49, 52).

[33] O. Kähler, V. A. Prisacariu, and D. W. Murray, "Real-time large-scale dense 3D reconstruction with loop closure," in *Proc. of the European Conference on Computer Vision (ECCV)*, 2016, pp. 500–516 (cit. on p. 11).

[34] A. Dai, M. Nießner, M. Zollöfer, S. Izadi, and C. Theobalt, "BundleFusion: Real-time globally consistent 3D reconstruction using on-the-fly surface re-integration," *ACM Transactions on Graphics (TOG)*, 2017 (cit. on p. 12).

[35] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb, "Real-time 3D reconstruction in dynamic scenes using point-based fusion," in *Proc. of the IEEE International Conference on 3D Vision (3DV)*, 2013, pp. 1–8 (cit. on p. 12).

[36] F. Furrer, T. Novkovic, M. Fehr, A. Gawel, M. Grinvald, T. Sattler, R. Siegwart, and J. Nieto, "Incremental object database: Building 3D models from multiple partial observations," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 6835–6842 (cit. on pp. 12, 14, 46, 74, 81).

[37] K. Tateno, F. Tombari, and N. Navab, "Real-time and scalable incremental segmentation on dense slam," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 4465–4472 (cit. on p. 12).

[38] M. Grinvald, F. Furrer, T. Novkovic, J. J. Chung, C. Cadena, R. Siegwart, and J. Nieto, "Volumetric instance-aware semantic mapping and 3D object discovery," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 3037–3044, 2019 (cit. on pp. 12, 46).

[39] P. Alimi, D. Meger, and J. J. Little, "Object persistence in 3D for home robots," in *Proc. of the ICRA Workshop on Semantic Perception, Mapping and Exploration*, 2012 (cit. on pp. 12, 13, 25, 34).

[40] R. Finman, T. Whelan, M. Kaess, and J. J. Leonard, "Toward lifelong object segmentation from change detection in dense RGB-D maps," in *Proc. of the IEEE European Conference on Mobile Robots (ECMR)*, 2013, pp. 178–185 (cit. on pp. 12–14).

[41] E. Langer, B. Ridder, M. Cashmore, D. Magazzeni, M. Zillich, and M. Vincze, "On-the-fly detection of novel objects in indoor environments," in *Proc. of IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2017, pp. 900–907 (cit. on pp. 12, 13, 26, 50).

[42] E. Langer, T. Patten, and M. Vincze, "Where does it belong? Autonomous object mapping in open-world settings," *Frontiers in Robotics and AI*, vol. 9, 2022 (cit. on p. 12).

[43] A. Adam, T. Sattler, K. Karantzalos, and T. Pajdla, "Objects can move: 3D change detection by geometric transformation consistency," in *Proc. of the European Conference on Computer Vision (ECCV)*, Springer, 2022, pp. 108–124 (cit. on pp. 13, 25, 38).

[44] J.-M. Park, J.-H. Jang, S.-M. Yoo, S.-K. Lee, U.-H. Kim, and J.-H. Kim, "ChangeSim: Towards end-to-end online scene change detection in industrial indoor environments," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 8578–8585 (cit. on pp. 13, 18, 19).

[45] L. Weihs, M. Deitke, A. Kembhavi, and R. Mottaghi, "Visual room rearrangement," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 5922–5931 (cit. on pp. 13, 18, 19, 53).

[46] R. Ambrus, J. Folkesson, and P. Jensfelt, "Unsupervised object segmentation through change detection in a long term autonomy scenario," in *Proc. of the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2016, pp. 1181–1187 (cit. on pp. 14, 18, 19).

[47] N. Bore, J. Ekekrantz, P. Jensfelt, and J. Folkesson, "Detection and tracking of general movable objects in large three-dimensional maps," *IEEE Transactions on Robotics*, vol. 35, no. 1, pp. 231–247, 2018 (cit. on pp. 14, 53).

[48] B. Graham, M. Engelcke, and L. van der Maaten, "3D semantic segmentation with submanifold sparse convolutional networks," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 9224–9232 (cit. on pp. 15, 42, 47, 57).

[49] C. Choy, J. Gwak, and S. Savarese, "4D spatio-temporal ConvNets: Minkowski convolutional neural networks," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 3075–3084 (cit. on p. 15).

[50] D. Chen, J. Li, Z. Wang, and K. Xu, "Learning canonical shape space for category-level 6D object pose and size estimation," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 973–11 982 (cit. on p. 15).

[51] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas, "Normalized object coordinate space for category-level 6D object pose and size estimation," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 2642–2651 (cit. on p. 15).

[52] M. Tian, M. H. Ang, and G. H. Lee, "Shape prior deformation for categorical 6D object pose and size estimation," in *Proc. of the European Conference on Computer Vision (ECCV)*, Springer, 2020, pp. 530–546 (cit. on p. 15).

[53] M. Z. Irshad, T. Kollar, M. Laskey, K. Stone, and Z. Kira, "CenterSnap: Single-shot multi-object 3D shape reconstruction and categorical 6D pose and size estimation," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 10 632–10 640 (cit. on p. 15).

[54] T. Kollar, M. Laskey, K. Stone, B. Thananjeyan, and M. Tjersland, "SimNet: Enabling robust unknown object manipulation from pure synthetic data via stereo," in *Conference on Robot Learning*, PMLR, 2022, pp. 938–948 (cit. on p. 15).

[55] A. Kriegler, C. Beleznai, M. Murschitz, K. Göbel, and M. Gelautz, "PrimitivePose: 3D bounding box prediction of unseen objects via synthetic geometric primitives," in *IEEE International Conference on Robotic Computing (IRC)*, 2022, pp. 1–8 (cit. on p. 15).

[56] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018 (cit. on pp. 15, 66).

[57] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017, pp. 2961–2969 (cit. on pp. 15, 16, 66).

[58] A. Dhamija, M. Gunther, J. Ventura, and T. Boult, "The overlooked elephant of object detection: Open set," in *Proc. of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020, pp. 1021–1030 (cit. on pp. 16, 68).

[59] S. Pidhorskyi, R. Almohsen, and G. Doretto, "Generative probabilistic novelty detection with adversarial autoencoders," in *Proc. of the Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, 2018 (cit. on p. 16).

[60] T. Boccato, T. Patten, M. Vincze, and S. Ghidoni, "In the depths of hyponymy: A step towards lifelong learning," in *Proc. of the International Conference on Autonomic and Autonomous Systems*, 2020, pp. 103–109 (cit. on p. 16).

[61] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in cognitive sciences*, vol. 3, no. 4, pp. 128–135, 1999 (cit. on p. 16).

[62] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proc. of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017 (cit. on p. 16).

[63] D. Miller, L. Nicholson, F. Dayoub, and N. Sünderhauf, "Dropout sampling for robust object detection in open-set conditions," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3243–3249 (cit. on p. 16).

[64] D. Miller, F. Dayoub, M. Milford, and N. Sünderhauf, "Evaluating merging strategies for sampling-based uncertainty techniques in object detection," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 2348–2354 (cit. on p. 16).

[65] Y. Li and J. Košecká, "Uncertainty aware proposal segmentation for unknown object detection," in *Proc. of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022, pp. 241–250 (cit. on p. 16).

[66] X. Du, X. Wang, G. Gozum, and Y. Li, "Unknown-aware object detection: Learning what you don't know from videos in the wild," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 13 678–13 688 (cit. on p. 16).

[67] J. Cen, P. Yun, J. Cai, M. Y. Wang, and M. Liu, "Open-set 3D object detection," in *Proc. of the IEEE International Conference on 3D Vision (3DV)*, 2021, pp. 869–878 (cit. on p. 16).

[68] A. Gupta, S. Narayan, K. Joseph, S. Khan, F. S. Khan, and M. Shah, "OW-DETR: Open-world detection transformer," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 9235–9244 (cit. on pp. 16, 17).

[69] J. Qian, V. Chatrath, J. Yang, J. Servos, A. Schoellig, and S. L. Waslander, "POCD: Probabilistic object-level change detection and volumetric mapping in semi-static scenes," in *Proc. of Robotics: Science and Systems (RSS)*, 2022 (cit. on pp. 18, 19).

[70] M. Halber, Y. Shi, K. K. Xu, and T. Funkhouser, "Rescan: Inductive instance segmentation for indoor RGBD scans," in *Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 2541–2550 (cit. on pp. 18, 19).

[71] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi, "AI2-THOR: An interactive 3D environment for visual AI," *arXiv preprint arXiv:1712.05474*, 2017 (cit. on p. 19).

[72] S. Höfer, K. Bekris, A. Handa, J. C. Gamboa, F. Golemo, M. Mozifian, C. Atkeson, D. Fox, K. Goldberg, J. Leonard, *et al.*, "Perspectives on sim2real transfer for robotics: A summary of the r: Ss 2020 workshop," *arXiv preprint arXiv:2012.03806*, 2020 (cit. on p. 20).

[73] S. Thalhammer, M. Leitner, T. Patten, and M. Vincze, "PyraPose: Feature pyramids for fast and accurate object pose estimation under domain shift," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 13 909–13 915 (cit. on p. 20).

[74] J.-B. Weibel, T. Patten, and M. Vincze, "Robust sim2real 3d object classification using graph representations and a deep center voting scheme," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 8028–8035, 2022 (cit. on p. 20).

[75] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3D reconstructions of indoor scenes," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5828–5839 (cit. on pp. 20, 42, 47).

[76] T. Yamamoto, K. Terada, A. Ochiai, F. Saito, Y. Asahara, and K. Murase, "Development of human support robot as the research platform of a domestic mobile manipulator," *ROBOMECH journal*, vol. 6, no. 1, pp. 1–15, 2019 (cit. on p. 20).

[77] B. Calli, A. Singh, J. Bruce, A. Walsman, K. Konolige, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Yale-CMU-Berkeley dataset for robotic manipulation research," *The International Journal of Robotics Research*, vol. 36, no. 3, pp. 261–268, 2017 (cit. on pp. 21, 82).

[78] C. V. Nguyen, S. Izadi, and D. Lovell, "Modeling Kinect sensor noise for improved 3D reconstruction and tracking," in *Proc. of the International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission (3DIMPVT)*, 2012, pp. 524–530 (cit. on p. 27).

[79] G. Nemhauser, L. Wolsey, and M. Fisher., "An analysis of the approximations for maximizing submodular set functions," in *Mathematical Programming*, vol. 14, 1978, pp. 265–294 (cit. on p. 27).

[80] J. Hoffmann, "FF: The fast-forward planning system," *AI magazine*, vol. 22, pp. 57–62, 2001 (cit. on p. 29).

[81] M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. Weld, and D. W. et al., "PDDL - The Planning Domain Definition Language," Yale Center for Computational Vision and Control, Tech. Rep., 1998 (cit. on p. 29).

[82] M. Cashmore, M. Fox, D. Long, D. Magazzeni, B. Ridder, A. Carrera, N. Palomeras, N. Hurtos, and M. Carreras, "ROSPlan: Planning in the robot operating system," in *Proc. of the International Conference on Automated Planning and Scheduling (ICAPS)*, vol. 25, 2015, pp. 333–341 (cit. on p. 30).

[83] G. Berry and G. Gonthier, "The ESTEREL synchronous programming language: Design, semantics, implementation," *Science of Computer Programming*, vol. 19, no. 2, pp. 87–152, 1992 (cit. on p. 30).

[84] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981 (cit. on pp. 33, 42, 57).

[85] A. Ückermann, C. Elbrechter, R. Haschke, and H. Ritter, "Real-time hierarchical scene segmentation and classification," in *Proc. of the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2014 (cit. on p. 35).

[86] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, vol. 3, 2004, pp. 2149–2154 (cit. on p. 36).

[87] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of histograms for local surface description," in *Proc. of the European Conference on Computer Vision (ECCV)*, 2010, pp. 356–369 (cit. on p. 40).

[88] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3D object recognition," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 998–1005 (cit. on pp. 40, 54, 61, 64).

[89] T. Fäulhammer, M. Zillich, J. Prankl, and M. Vincze, "A multi-modal RGB-D object recognizer," in *Proc. of the IEEE International Conference on Pattern Recognition (ICPR)*, 2016, pp. 733–738 (cit. on p. 40).

[90] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2881–2890 (cit. on p. 42).

[91] A. Valada, R. Mohan, and W. Burgard, "Self-supervised model adaptation for multimodal semantic segmentation," *International Journal of Computer Vision*, vol. 128, no. 5, pp. 1239–1285, 2020 (cit. on p. 42).

[92] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 14, no. 2, pp. 239–256, 1992 (cit. on p. 45).

[93] G. Bugmann and S. N. Copleston, "What can a personal robot do for you?" In *Proc. of the Conference Towards Autonomous Robotic Systems*, 2011, pp. 360–371 (cit. on pp. 53, 78).

[94] M. Cakmak and L. Takayama, "Towards a comprehensive chore list for domestic robots," in *Proc. of the ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2013, pp. 93–94 (cit. on p. 53).

[95] H. Okada, T. Inamura, and K. Wada, "What competitions were conducted in the service categories of the world robot summit?" *Advanced Robotics*, vol. 33, no. 17, pp. 900–910, 2019 (cit. on p. 53).

[96] M. Tenorth and M. Beetz, "KnowRob: A knowledge processing infrastructure for cognition-enabled robots," *The International Journal of Robotics Research*, vol. 32, no. 5, pp. 566–590, 2013 (cit. on p. 53).

[97] D. Batra, A. X. Chang, S. Chernova, A. J. Davison, J. Deng, V. Koltun, S. Levine, J. Malik, I. Mordatch, R. Mottaghi, *et al.*, "Rearrangement: A challenge for embodied AI," *arXiv preprint arXiv:2011.01975*, 2020 (cit. on p. 53).

[98] Z.-C. Marton, D. Pangercic, N. Blodow, J. Kleinehellefort, and M. Beetz, "General 3D modelling of novel objects from a single view," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 3700–3705 (cit. on p. 56).

[99] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An open-source Robot Operating System," in *Proc. of the ICRA Workshop on Open Source Software*, vol. 3, 2009 (cit. on p. 57).

[100] M. R. Luo, G. Cui, and B. Rigg, "The development of the CIE 2000 colour-difference formula: CIEDE2000," *Color Research & Application*, vol. 26, no. 5, pp. 340–350, 2001 (cit. on p. 60).

[101] T. Hodaň, F. Michel, E. Brachmann, W. Kehl, A. Glent Buch, D. Kraft, B. Drost, J. Vidal, S. Ihrke, X. Zabulis, C. Sahin, F. Manhardt, F. Tombari, T.-K. Kim, J. Matas, and C. Rother, "BOP: Benchmark for 6D object pose estimation," *Proc. of the European Conference on Computer Vision (ECCV)*, 2018 (cit. on pp. 61, 67).

[102] J. Edmonds, "Maximum matching and a polyhedron with 0, 1-vertices," *Journal of Research of the National Bureau of Standards, Section B, Mathematics and Mathematical Physics*, vol. 69, no. 125-130, pp. 55–56, 1965 (cit. on p. 62).

[103] F. D. de Oliveira, M. R. da Silva, and A. F. Araújo, "Spatio-temporal data association for object-augmented mapping," *Journal of Intelligent & Robotic Systems*, vol. 103, no. 1, pp. 1–19, 2021 (cit. on p. 66).

[104] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Proc. of the European Conference on Computer Vision (ECCV)*, Springer, 2014 (cit. on p. 66).

[105] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A convolutional neural network for 6d object pose estimation in cluttered scenes," in *Proc. of Robotics: Science and Systems (RSS)*, 2018 (cit. on pp. 66, 67).

[106] J. Qiu, Y. Yang, X. Wang, and D. Tao, "Hallucinating visual instances in total absentia," in *Proc. of the European Conference on Computer Vision (ECCV)*, Springer, 2020, pp. 264–282 (cit. on p. 66).

[107] K. Park, T. Patten, and M. Vincze, "Pix2Pose: Pixel-wise coordinate regression of objects for 6D pose estimation," in *Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 7668–7677 (cit. on p. 67).

[108] C. Houseago, M. Bloesch, and S. Leutenegger, "KO-fusion: Dense visual SLAM with tightly-coupled kinematic and odometric tracking," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 4054–4060 (cit. on p. 73).

[109] C. Choi and H. I. Christensen, "3D pose estimation of daily objects using an RGB-D camera," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 3342–3349 (cit. on p. 74).

[110] D. Bauer, T. Patten, and M. Vincze, "SporeAgent: Reinforced scene-level plausibility for object pose refinement," in *Proc. of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022, pp. 654–662 (cit. on p. 74).

[111] S. V. Alexandrov, J. Prankl, M. Zillich, and M. Vincze, "Calibration and correction of vignetting effects with an application to 3D mapping," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 4217–4223 (cit. on p. 80).

[112] S. Sajjan, M. Moore, M. Pan, G. Nagaraja, J. Lee, A. Zeng, and S. Song, "Clear grasp: 3D shape estimation of transparent objects for manipulation," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 3634–3642 (cit. on p. 80).

Erklärung

Hiermit erkläre ich, dass die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt wurde. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.

Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder in ähnlicher Form in anderen Prüfungsverfahren vorgelegt.

Vienna, March 2023

Dipl.-Ing. Edith Langer