TU WIEN Informatics

Unüberwachtes Inkrementelles Lernen für Objekt-zentriertes Mapping in Open-World-Umgebungen

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Visual Computing

eingereicht von

Stefan Fiedler, BSc
Matrikelnummer 00155490

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Markus Vincze
Mitwirkung: Michael Schwärzler, PhD (Industrial Supervisor)

Wien, 9. März 2023

Stefan Fiedler	Markus Vincze

TU WIEN Informatics

Unsupervised Incremental Learning in Open-World Object-Centric Mapping for Mobile Autonomous Robots

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Visual Computing

by

Stefan Fiedler, BSc
Registration Number 00155490

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Markus Vincze
Assistance: Michael Schwärzler, PhD (Industrial Supervisor)

Vienna, 9th March, 2023

Stefan Fiedler	Markus Vincze

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Stefan Fiedler, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 9. März 2023

Stefan Fiedler

Danksagung

Diese Arbeit wurde im Rahmen eines bezahlten Praktikums bei Baxalta Innovations GmbH, einem Unternehmen von Takeda Pharmaceuticals, Inc., geschrieben. Ich möchte mich bei allen bedanken die mir die Gelegenheit zur Arbeit an diesem Thema geboten haben, insbesondere bei Christoph Pistek, Larissa Kahr und Patricia Wildberger.

Mein besonderer Dank gilt meinem akademischen Betreuer Markus Vincze und Michael Schwärzler als meinem Industrial Supervisor. Ich bedanke mich ebenfalls bei Robert Sablatnig and allen Teilnehmerinnen und Teilnehmern des Diplomandenseminars für ihre Verbesserungsvorschläge zu dem Thema und zu den Zwischenberichten.

Ich möchte mich bei Janek Janßen für das Teilen seiner Erfahrung zur Programmierung mit ROS bedanken, bei Ádám Wolf für sein Feedback zu Anwendungen mobiler autonomer Roboter in Laborumgebungen und bei Jason Young für zusätzliche Diskussionen über Anwendungen mobiler Roboter.

Darüber hinaus gilt mein Dank Alexandra Umprecht für ihre technische Unterstützung, und Quang-Hieu Pham für die Zusendung einer Kopie des Supplementary zu “Real-time Progressive 3D Semantic Segmentation for Indoor Scenes” für meine Literaturrecherche.

Acknowledgements

This thesis was written as part of a paid internship at Baxalta Innovations GmbH, a company of Takeda Pharmaceuticals, Inc. I would like to thank everyone who provided me the opportunity to work on this topic, in particular Christoph Pistek, Larissa Kahr, and Patricia Wildberger.

Special thanks go to my academic supervisor Markus Vincze and Michael Schwärzler as my industrial supervisor. My thanks also go to Robert Sablatnig and everyone at the graduand seminar for their feedback on the topic and progress reports.

I would like to thank Janek Janßen for sharing his insights into ROS programming, Ádám Wolf for his feedback on applications of mobile autonomous robots in lab environments, and Jason Young for further discussions on mobile robotic applications.

In addition, my thanks go to Alexandra Umprecht for her technical support, and Quang-Hieu Pham for providing me with a copy of the Supplementary on "Real-time Progressive 3D Semantic Segmentation for Indoor Scenes" for my literature research.

Kurzfassung

Objekt-zentriertes Mapping ist eine naturgemäße und effiziente Darstellung von 3D-Umgebungen im Einsatz von Robotern, die eine große Zahl abstrakter Tätigkeiten ermöglicht. Aktuelle Ansätze zum dreidimensionalen Bildverstehen sind oft auf die Annahme geschlossener Trainingsdatensätze beschränkt.

In praxisnahen Szenarien begegnen autonome Agenten häufig neuartigen Objekten und Umgebungen. Modelle für das Bildverstehen so anzupassen, dass neue Objektklassen erkannt und segmentiert werden, ist eine herausfordernde Aufgabe in Bereich des maschinellen Sehens und Lernens. Zusätzlich ist es wünschenswert dass mobile autonome Roboter unbekannte Objekte erkennen und ohne menschliche Eingriffe aus Beobachtungen sinnvolle neue Kategorien bilden können.

In dieser Arbeit kombinieren wir unüberwachtes inkrementelles Lernen, die Erkennung neuartiger Objekte und die Bildung neuer Kategorien mit Objekt-zentriertem Mapping von unstrukturierten Innenräumen. Unser Framework ermöglicht es mobilen autonomen Robotern in vollständig automatisierter Weise Objektkategorien in RGB-D Sensordaten zu entdecken und zu lernen, und dieses neue Wissen auf Instanz-basierte metrisch-semantische 3D-Rekonstruktionen anzuwenden.

Eine Evaluierung mit vier Sätzen von Aufnahmen von Alltagsszenerien demonstriert die generelle Anwendbarkeit unseres Ansatzes, misst die Auswirkungen von inkrementellem Lernen auf Objekterkennung und Segmentierung und illustriert die Funktionalität des Frameworks mit zahlreichen Beispielen. Wir schließen der Evaluierung eine Erörterung der Ergebnisse und potentieller zukünftiger Verbesserungen unserer Methode an.

Abstract

Object-centric mapping is a natural and efficient representation of 3D environments in robot applications that enables a large range of high-level tasks. Current approaches to 3D scene understanding are often limited by the assumption of closed sets of training data.

In real-world scenarios, autonomous agents frequently encounter novel objects and unseen environments. Adapting models for scene understanding to detect and segment new object classes is a challenging problem in computer vision and machine learning. In addition, it is desirable for mobile autonomous robots to discover unknown objects and form meaningful new categories from observations without human supervision.

In this work, we combine unsupervised incremental learning, novelty detection, and category discovery with dense object-centric mapping of real-world indoor environments. Our framework enables mobile autonomous robots to discover and learn object categories from RGB-D sensor data, and apply this new knowledge to instance-aware metric-semantic 3D reconstructions in a completely automated fashion.

Evaluations on four sets of real-world scene recordings demonstrate the general practicality of our approach, measure the effects of incremental learning on object detection and segmentation, and illustrate the functionality of the framework with numerous examples. We follow this with a discussion of evaluation results and of potential future improvements to our method.

For those who would stand
against the weight of a
universe to rise but
one step further
over its horizon.

~

D. W. Bradley

Contents

CHAPTER 1

Introduction

Mobile robots are a promising new asset in the field of lab automation, with current research and development mainly focusing on use cases in data acquisition and pick-and-place tasks [WWT+21a]. The functionality and autonomy of mobile robots are in part restricted by their limited artificial intelligence capabilities. This is in turn partly due to a lack of scene understanding [KMvB+20].

Current dense SLAM methods can create 3D scene reconstructions in high detail, incrementally and in real-time [HCC22]. Annotating the geometry with object class, and instance, information results in metric-semantic and instance-aware mapping [WNT22]. Object-centric SLAM methods model scene geometry at object granularity through reconstruction [MCB+18], scene differencing [LPV22], or geometric-semantic segmentation [GFN+19].

From 3D mappings higher-level representations, such as free space skeletonization or scene graphs, can be derived to enable efficient computations, e.g. for navigation [RVA+21], collision avoidance [OTSN18], as well as visual questioning and answering and task planning [KPSK19].

Object detection and semantic segmentation typically operate under a closed-set assumption. Current methods achieve high levels of accuracy, necessary for meaningful object-level representations and scene graph generation. The closed-set assumption however limits the applicability to domains with a fixed set of a priori known object classes. For many real-world applications, systems that are capable of incrementally learning classes and adapting to new domains would be highly advantageous [KMvB+20].

The aim of this thesis lies in the research and development of a method for mobile autonomous robots that enhances the autonomy in and understanding of open-world environments through unsupervised adaptation to new domains. In particular, we intend to design and implement a general, modular, and extensible framework including object-centric SLAM, open-world object detection, new class discovery, and incremental learning

from real-world RGB-D data sets based on state-of-the-art software and development methodologies.

1.1 Open-World Scene Understanding in Robotic Applications

Mobile autonomous robots rely on detailed, accurate, and up-to-date models of their environment for navigation, task planning, and object manipulation. Consequently, SLAM (Simultaneous Localization and Mapping) is a common task for mobile robots intended to create a consistent global 3D scene representation from sensor data [KPSK19].

Advances in algorithms and processing power allow to execute SLAM online and in an incremental fashion, suitable for real-time applications [HCC22]. Improved sensor technology and memory capacity increase the accuracy and detail of 3D mapping and reconstruction, but also lead to more data that needs to be processed [KMvB+20].

Processing this data in higher-level tasks, such as navigation, semantic querying, or scene manipulation, puts high computational requirements on algorithms and can hinder real-time interactions. It is therefore beneficial to have methods for scene understanding that go beyond data in the form of point clouds, volumes, or meshes [Dav18].

Scene graphs are a compact, abstract, efficient, and versatile representation of environments. In this context, scene graphs model semantic relationships between scene elements, e.g. geometric and comparative relations, and store attributes, such as color and shape [AHG+19].

Scene graph construction typically depends on correct object detection or scene segmentation. Realizing object detection and segmentation with deep neural networks achieves state-of-the-art results but requires training with large annotated, domain-specific datasets [WNT22]. This approach can achieve excellent results under closed-set conditions, but may assign unknown objects to wrong classes with high confidence, limiting their applicability to the domain of the dataset [LPV22].

Collecting datasets for new domains, such as lab environments, is labor-intensive and time-consuming. Re-training networks has to be carried out each time a new class of objects is introduced [AKC+22]. Advances in persistent mapping, in particular scalable and generic solutions that are weakly, self- or unsupervised, are a promising approach to many challenges in long-term and dynamic 3D scene understanding [WNT22]

1.2 Robotic Perception and Task Execution

As can be seen from the above considerations, perception plays a central role in robotic systems and task execution. According to Batra et al. [BCC+20] “perception converts sensory input to a representation of the world, from which planning produces actions.” Mobile autonomous robots can be regarded as a form of embodied AI. Batra et al. define

embodied AI as "the study and development of intelligent systems with a physical or virtual embodiment." They note the importance of object detection, pose estimation, navigation, and consistent representations of the environment for robotic applications, as they can all be required for higher-level tasks.

Several examples illustrate this point. In their work, Batra et al. consider rearrangement as a high-level task with a wide range of applications. Informally, rearrangement in the robotic context can be defined as transforming a scene from the current state to a specific goal state. The goal state can be described for example as a geometric configuration, using natural language or a formal predicate-based specification, or by a robot examining the goal state beforehand.

Puente et al. [dlPBE+14] consider several user-relevant tasks for home environments, including ambient assisted living. These tasks typically involve some form of human-robot interaction (HRI), i.e., the two-way communication between a robot and a human user in way that is natural to humans. From these tasks they define multiple technical requirements for robotic perception.

Tasks considered include calling the robot, finding the user, bringing an object, and multimodal HRI. Perception requirements following from these tasks are 2D localization and mapping, obstacle avoidance for safe navigation, object detection and pick up, as well as person detection and gesture recognition.

Rosinol et al. [RVA+21] demonstrate the importance and usefulness of higher-level scene representations in their spatial perception engine. They use a dynamic scene graph automatically created from 3D scene reconstructions to perform hierarchical semantic path planning in an efficient and scalable manner.

As another example of robots working in human environments, Langer et al. [LPV22] approach the challenging task of a household robot cleaning and picking up objects. They propose scene and object change detection as a method for robotic perception in unstructured and unseen environments, which can be useful for a wide variety of tasks.

Hughes et al. [HCC22] reiterate the need for high-level, consistent representations of environments in future robots to understand instructions, plan task execution, and accomplish tasks. They also note how persistence is important for long-term autonomy, in scaling to large environments, corrections in face of new evidence, and model sizes that are proportional to the size of the environment.

1.3 Challenges in Open-World Scene Understanding

These ambitious goals are faced with a number of challenging problems. Hughes et al. [HCC22] see a rise of interest in metric-semantic mapping, with recent works including object-based maps, and dense maps from volumetric models, point clouds, or meshes. However, dense maps do not directly lend themselves to navigation.

Hierarchical mapping in early robotics is limited to 2D, to bridge the gap between topology and geometry. More recently, scene graphs are investigated as hierarchical models of 3D scenes.

Kasaei et al. [KMvB+20] note time complexity as a large issue in real-world robotic applications. Other issues that service robots currently cannot deal with appropriately are unknown environments, open-ended learning of object categories, 3D representations for global navigation, object recognition for local navigation, and collision detection in object manipulation.

On the other hand, Kasaei et al. conclude that other major issues are close to completion in specific types of environments. These include path planning and grasp planning in known, reliable environments. Object recognition, as a prerequisite for these planning tasks, can achieve almost perfect scores in predetermined scenarios.

Under open-set conditions, novel objects are explicitly assigned the unknown label. Developing for these conditions can handle objects not seen in training more gracefully. Methods for open-world environments go one step further and incrementally learn unknown classes to be able to recognize them in further tasks [JKKB21]. For unsupervised class-incremental learning it is necessary to also perform category discovery on novel objects [URG22].

Open-world class-incremental learning bypasses the problem of training datasets lacking samples of domain-specific classes. This allows to apply methods to new domains without additional preparations and to extend domain knowledge on the task [QRX+21].

Recently, researchers have begun to tackle the problem of incremental and continuous learning for object detection [JKKB21, JRK+21, Dav21, GNJ+22, ZLH+22, ZLS+22, YPRL22, DWGL22], semantic segmentation [MZ19, CYC+21, URG22], and instance segmentation [CGFC22].

Major challenges in this area include catastrophic forgetting, i.e., the inability of systems to retain knowledge of old tasks as new tasks are learned, as well as computational and storage limitations with respect to system performance [PKP+19]. While incremental learning methods are typically less accurate than fine-tuning to a specific domain, they perform substantially better than the baseline of fine-tuning with new data only, and thus show the general feasibility of the approach.

With regards to sensor hardware, Zollhöfer et al. [ZSG+18] report as additional challenges resilience against background light, in particular outdoors, the quality of depth data, objects made of semi-transparent media, and multi-path effects, i.e., indirect paths of active light in certain types of depth sensors.

Wald et al. [WNT22] report several challenges related to creating scene graphs, including noise, distortion, and occlusions in real-world data, and ambiguities in the description of graph nodes and edges. Objects may be assigned different classes depending on their location, for example ‘towel’ and ‘blanket’, and partially reconstructed objects can appear to look alike.

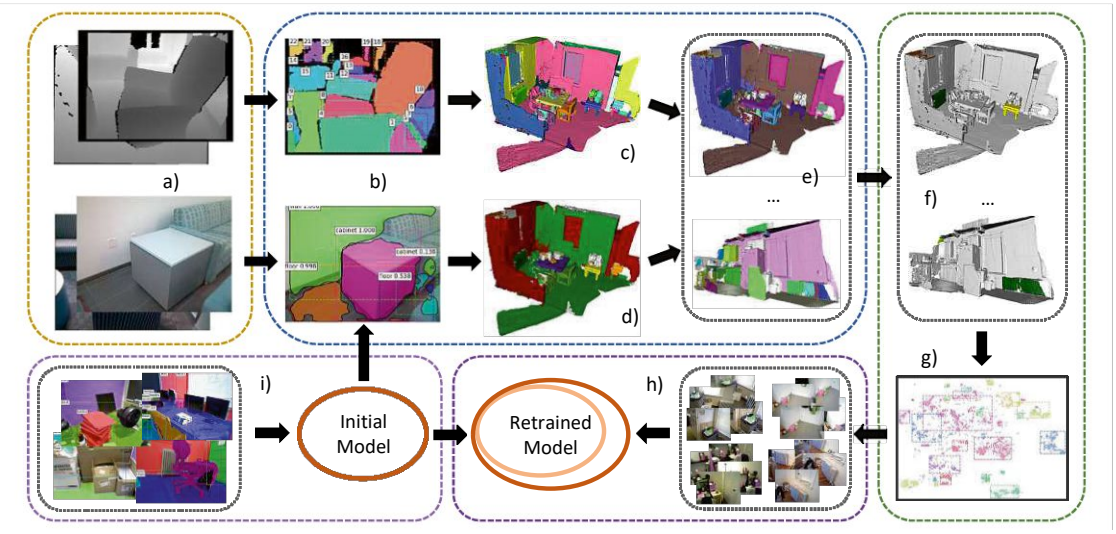


Figure 1.1: In dense SLAM, RGB-D sensor data a) is segmented b) and fused into global map annotations c) and d). Depth segmentation typically leads to over-segmented maps c). Semantic annotation alone lacks segmentation at the object level d). Combining both creates scene reconstructions e) with object instance and class information. However, object detection and semantic information is limited to the closed-set training classes i) of the instance segmentation model. Our framework extends the initial model with new categories in a completely automated fashion. Leveraging information from a set e) of global maps, it detects novel objects f), clusters object views based on visual similarity g), and creates a new training set h) to retrain the model using incremental learning.

In conclusion, Kasaei et al. [KMvB+20] note that improvements in research areas are not always linear. Instead, different methods are faced with trade-offs, for example between accuracy and speed. The authors see this mainly in planning modules.

In other areas, different approaches are developed in parallel, without a clear indication which is best suited for a task. Kasaei et al. state this type of situation exists in many areas concerning service robots. This includes object representations, using hand-crafted or learned descriptors for object perception, and different approaches to scene perception based on bounding boxes or image segmentation.

1.4 Contribution

This work addresses the limitations of closed-set object detection in creating object-centric 3D scene representations, and intends to enable robots to learn new object categories during operation with no human intervention. To this end, our approach combines object-level SLAM with unsupervised open-world class-incremental learning as a basis for higher-level scene representations. In this way it is possible to learn new object classes and adapt to new domains in a completely automated fashion, while avoiding re-training

the system [PKP+19]. The problem formulation and a concept of our contribution are illustrated in Figure 1.1.

We construct a software framework that consists of several recently published methods for metric-semantic mapping, category discovery, and incremental learning for open-world object detection and semantic segmentation. The underlying 3D model is specifically chosen so as to serve as a foundation for higher-level scene representations, such as scene graphs, and collaborative tasks, including interaction with and teaching by humans. It is our expectation that advancements in open-world scene understanding will provide an important step towards the development of truly autonomous robots [Dav18].

Our main contribution lies in the combination of unsupervised incremental learning with open-world object detection and semantic segmentation for metric-semantic mapping. To this end, we leverage the geometric segmentation of novel objects in the SLAM algorithm to provide expressive samples for class discovery and incremental learning.

The newly learned classes for object detection and semantic segmentation in turn are used to improve the geometric segmentation of objects in the 3D model. We perform several experiments on real-world scene data, measure the effects of incremental learning on object detection and segmentation, and illustrate the performance of our framework with scene visualizations and examples from class discovery.

This work is structured as follows: related work is presented in chapter 2, roughly grouped into sections related to scene reconstruction (2.1), SLAM (2.2), scene graphs (2.3), scene understanding (2.4), object discovery (2.5), incremental learning (2.6), and open-world semantic parsing (2.7). Our framework is presented in detail in chapter 3. The design choices are listed in section 3.1, and section 3.2 provides an overview of the system design. The remaining sections of this chapter focus on each of the three main components of the framework: object-centric mapping (3.3), instance-aware semantic segmentation (3.4), and novelty detection and category discovery (3.5). We present the evaluation of our framework in chapter 4, including the choice of dataset (4.1) and dataset generation (4.2), evaluation protocol (4.3), as well as a summary and discussion of experimental results (4.4). We provide a conclusion and an outlook on future work in chapter 5.

CHAPTER 2

Related Work

Given the combination of several methods from computer vision and machine learning in our framework, related work can be grouped roughly into four different topics:

- scene reconstruction, SLAM, and scene graphs,
- object-related scene understanding,
- novelty detection, object and category discovery, as well as
- incremental learning and real-world semantic parsing.

With the multi-faceted nature of these topics, however, there is considerable overlap between different sections, as many of the works listed herein are related to more than one aspect of our framework.

2.1 Scene Reconstruction

Object-centric mapping is part of the broader topic of scene reconstruction, where the goal is to create a faithful geometric 3D representation of an environment, or scene, through visual means. Scene reconstruction methods can be categorized into offline and online methods. While the latter allow robotic systems to incrementally reconstruct scenes during operation, offline methods typically trade speed and immediate updates for higher accuracy and reconstruction quality [ZSG+18].

As Zheng et al. [ZZZ+19] mention, larger environments require robust, global pose estimation typically found in offline methods. However, improved algorithms and GPU-based optimizations allow for global pose estimation in online scene reconstruction as well.

Another categorization of reconstruction methods relates to static and dynamic scenes. Dynamic scenes may contain moving objects and persons, and methods designed for such scenes can handle and represent movement and dynamic changes appropriately.

Methods for static scenes only consider non-moving parts of the environment and filter any dynamic scene elements as outliers [ZSG+18]. Since our framework only deals with static scenes, the focus of the related work is also on such methods.

2.1.1 3D Scene Representations

Reconstruction methods are based on 3D representations for individual objects or the whole scene, and the choice of representation can affect the performance and expressiveness of the reconstruction. Zollhöfer et al. [ZSG+18] provide an extensive overview of methods and representations. They classify representations as either voxel-based, point-based, or hybrid.

In general, the 3D representation must model the geometry of the scene, and support the fusion of new data into the model. Since there is a potentially large amount of data to integrate, efficient implementations are necessary, in particular for online methods.

Voxel-based approaches store values of a signed distance function (SDF) in a regular voxel grid, with negative values typically representing points inside a volume, and positive values outside. All surfaces are implicitly defined as the zero-crossing of this distance field. Each voxel can store additional attributes, such as color values.

In a basic voxel grid, the size of grid elements and the grid as a whole is predefined. Hierarchical models and voxel hashing allow for efficient storage, making it possible to adapt recordings to arbitrary scene dimensions not limited by GPU memory. Adapted voxel hashing also supports irregular grids and can record objects and scenes at different levels of detail.

In point-based approaches, sensor data is stored directly as point or surfel elements. These points can store additional attributes, such as color or point size. Surfels contain additional data that allows for fast direct rendering, and as such provides a trade-off between memory consumption and rendering speed.

Hybrid approaches combine different representations. One example includes clustering planar and non-planar regions. Planar regions are then defined by a shared normal vector, and non-planar regions are represented as point clouds.

In all representations, the integration or fusion of new data has to deal with the uncertainty of camera localization and sensor noise. For voxels, the weighted averaging of incoming and existing values in the voxel grid is a common merging step. This eliminates sensor noise through temporal integration of samples.

In point-based representations, points have associated states: incoming points are unstable at first, and merged several times with other points before becoming stable to eliminate

outliers. In order to merge points, it is necessary to find correspondences between points, e.g. with index maps and dense correspondence finding.

As Grinvald et al. [GFN+19] point out, an important advantage of volumetric representations is the explicit modeling of empty space, and that object models retain information about surface connectivity. This means they are directly suitable for navigation and motion planning in robotic applications. On the other hand, surfels can handle loop closures efficiently, but the lack of surface connectivity makes visibility calculations and collision detection much harder.

Schöps et al. [SSP19] additionally mention the ability of surfels as a dynamic data structure to change the resolution in scenes during recordings. Surfels can also represent thin objects that do not enclose a volume, which would not be possible with voxels.

2.2 Simultaneous Localization and Mapping

As Zollhöfer et al. [ZSG+18] point out, "online reconstruction is directly related to Simultaneous Localization and Mapping (SLAM)." In SLAM, the position and orientation of the sensor, typically mounted on a mobile platform, is tracked, and the recorded data is integrated into a global map. Here, there is a strong focus on correct localization, while the reconstruction is typically limited in detail.

2.2.1 Scene Reconstruction Pipeline

Zollhöfer et al. [ZSG+18] provide an overview of typical steps in static scene reconstruction pipelines. These include depth map processing, camera pose estimation, and depth map fusion.

Depth map preprocessing primarily aims at noise reduction and outlier removal. In this step it is also possible to derive additional information from the range map if needed, such as surface normal vectors.

Camera pose estimation computes the best alignment transformation for the current image frame. This transformation can be estimated frame-to-frame, frame-to-model, or with a global approach.

Depth map fusion transforms points from current the frame using the estimated transformation, and merges them with the common model. Since the scene reconstruction pipeline is only tangentially related to the scope of our work, we refer to [ZSG+18] for a more detailed introduction to the topic.

2.2.2 Dense SLAM

With dense SLAM, it is possible to model continuous surfaces in greater detail, and provide accurate localization. Dense SLAM also offers "the potential for detailed semantic scene understanding," as Whelan et al. [WSMG+16] put it.

In a similar vein, Davison [Dav18] predicts the development of SLAM towards Spatial AI systems. He defines Spatial AI as

> "the online problem where vision is to be used, usually alongside other sensors, as part of the Artificial Intelligence (AI) which permits an embodied device to interact usefully with its environment."

As such, it is not bound to a specific, abstract representation, but must continuously and in real-time provide useful and relevant information to other parts of the system. In this context, an embodied device could be an autonomous robot or a mobile platform.

While there is still a huge gap between the requirements of potential future robots and what is technically possible today, Davison provides two hypotheses regarding important properties of such systems. First of all, he notes:

> "When a device must operate for an extended period of time, carry out a wide variety of tasks (not all of which are necessarily known at design time), and communicate with other entities including humans, its Spatial AI system should build a general and persistent scene representation which is close to metric 3D geometry, at least locally, and is human understandable."

Secondly, regarding the performance of Spatial AI systems, Davison notes their usefulness "for a wide range of tasks is well represented by a relatively small number of performance measures." Metrics mentioned in [Dav18] include, among others, local pose accuracy in new areas, long term pose repeatability, tracking and relocalization robustness, but also object detection and classification accuracy, and scene change detection accuracy.

2.2.3 Semantic Maps

Metric-semantic maps combine geometric and semantic information. Approaches in this area of research annotate map elements with semantic class labels and potentially other semantic information. Such a representation can be an efficient model of semantic knowledge, for example for navigation or robotic interaction.

As Rosinol et al. [RVA+21] put it, "metric-semantic understanding is the capability of grounding semantic concepts [...] into a spatial representation." This development is driven by the availability of affordable cameras with depth sensors and the advancement of real-time dense SLAM [WWT+21b]. Wu et al. [WWT+21b] note that "the research focus has shifted from reconstructing the 3D scene geometry to enhancing the 3D maps with semantic information about scene components."

2.2.4 Object-Level SLAM

While metric-semantic maps add semantic class attributes to map elements, this information alone does not allow a distinction between individual instances of objects from the

same class [GFN+19]. Object-oriented maps are a way to add instance-level information to metric maps, by making objects a main building block of the scene representation. McCormac et al. [MCB+18] argue that

> "this is a natural and efficient way to represent the things that are most important for robotic scene understanding, planning and interaction; and it is also highly suitable as the basis for human-robot communication."

Tateno et al. [TTN15] present a method for incremental object segmentation in SLAM frameworks. Their general approach is suitable for any frame-wise segmentation and SLAM algorithm. Geometric segmentation is performed in each depth image, and a Global Segmentation Map (GSM) is built on top of the 3D reconstruction.

A merging procedure projects the GSM into each camera frame and merges new and existing segments. This procedure runs in constant time, irrespective of the GSM size or the number of depth maps, and makes processing of depth maps for segmentation possible in real-time.

McCormac et al. [MCB+18] apply a voxel-base approach to reconstruct objects in a scene at different resolutions, by using a separate TSDF (Truncated Signed Distance Field) for each object. SLAM incrementally refines object representations, which are used for tracking, relocalization and loop closure detection.

With their method, each scene consists of an optimizable pose graph of objects. This approach aims to capture objects in detail, distributing drift and registration errors in pose graph edges. In particular, this means no intra-object TSDF warping is necessary.

The work of McCormac et al. uses 2D instance mask predictions with Mask R-CNN and 3D voxel masks for instance foreground fusing. More specifically, they treat foreground and background detections as (α, β) parameters of a Beta distribution conjugate prior, and update foreground mask accordingly to provide a clear separation between objects and background.

The Incremental Object Database by Furrer et al. [FNF+18] scans scenes with a robot to extract 3D object models. It stores partial observations in a database, and merges matching observations to update models. Unobserved parts of a scene are reconstructed with information from the database. This method uses the segmentation of depth image to generate partial shapes and voxel-based TSDF representations for models.

To find similar objects, the method applies keypoint detection and descriptors on point clouds of objects, then performs matching and registration. It provides geometric-only segmentation and builds a database of exact shapes, which lacks semantic information for high-level planning.

Grinvald et al. [GFN+19] employ a voxel-based representation to incrementally build volumetric, object-level semantic maps from RGB-D sensor input. They combine unsupervised geometric segmentation with instance-aware semantic predictions from a

Mask R-CNN network, and fuse geometric and instance-aware semantic information into a global segmentation map. The main steps in this approach consist of geometric segmentation, semantic instance-aware segmentation refinement, data association, and map integration.

The main contributions comprise of combined geometric-semantic mapping that extends object-detection to novel objects of unseen categories, tracking and matching instance predictions across multiple frames with a global data association strategy, and evaluation on real-world datasets and an online robotic setup. It is based on previous work for incremental geometry-based scene segmentation in the Incremental Object Database [FNF+18] and the voxel-based scene representation VoxBlox [OTF+17].

Schmid et al. [SDS+21] build a semantic volumetric map at the object level that can deal consistently with scene changes, e.g. moved furniture, over long periods of time. This method takes as input RGB-D images and a panoptic segmentation of the images to produce one sub-map per semantic entity.

In this way, it uses multiple resolutions to store the room layout and objects efficiently, and can integrate multi-resolution TSDFs and manage maps consistently. In particular, it can handle moved and updated objects at the object-level, without reconstruction artifacts, and separately model nearby objects with different resolutions to ensure geometric details are captured at an appropriate level.

Grinvald et al. [GTSN21] reconstruct scenes and track dynamic objects in a single volume. Their method is designed to store multiple surfaces in the same data structure, and explicitly handle occlusions of objects. Most other methods use separate volumes for each moving object, but this creates a problem when scaling to a larger number of objects and with defining an explicit occlusion handling strategy. The per-frame segmentation method is the same as in [GFN+19]. For this method, exact segmentation is especially important for accurate reconstruction results.

2.3 Scene Graphs

Scene graphs are an abstract representation of objects and object relations that is especially useful for higher-level scene understanding [WWT+21b]. Originating in computer graphics, scene graphs allow for compact and complex representations of 3D environments.

3D data results in large scene graphs, which have the advantage of being human-readable representations that present scene information in a compact form [WNT22]. Hughes et al. [HCC22] define a scene graph as

> "a layered graph where nodes represent spatial concepts at multiple levels of abstraction (from low-level geometry to high-level semantics including objects, places, rooms, buildings, etc.) and edges represent relations between concepts."

Armeni et al. [AHG+19] propose hierarchical scene graphs with attributes and relations to model 3D scene reconstructions. Their models consist of four layers with increasing levels of abstraction, ranging from camera views to objects, rooms, and buildings.

Attributes on graph nodes include camera parameters, object class, material, shape, and other descriptive information such as room type and illumination. Edges in the graph model different kinds of relations between nodes, including spatial order, relative magnitude, occlusion relationships, and parent entities, i.e., rooms or buildings.

In order to reduce the effort to create scene graphs manually, Armeni et al. make two contributions to automate scene graph annotations. To improve 2D detection, they frame query images in panorama pictures and run object detection on these extended images. Secondly, to deal with misclassifications, they enforce detection consistency across multiple views from different camera locations.

Wu et al. [WWT+21b] present SceneGraphFusion as a method for real-time incremental panoptic segmentation and incremental scene graph generation from RGB-D image sequences. They use the geometric segmentation of depth images by [TTN15] for node segmentation, and PointNet for feature extraction from segments.

Geometric segmentation separates the point cloud into nodes, and node and edge features are computed from the properties and neighborhood relations of updated segments. The node and edge features are then used to predict a scene graph which is fused into a globally consistent model.

Since geometric segmentation can lead to over-segmentation, 'same-as' relations are used to connect parts of the same object where needed. In addition to incremental scene graph generation, the method of Wu et al. produces accurate panoptic segmentation of large-scale scenes, and runs at interactive speed with a frame rate of 35 Hz.

An extension of scene graphs towards a spatial perception engine that captures actionable information for robotic agents is presented by Rosinol et al. [RVA+21]. Motivating examples for the development of their Kimera engine include obstacle avoidance and planning, human interaction, long-term autonomy, and prediction.

Dynamic Scene Graphs (DSG) in the Kimera engine contain a bounding volume hierarchy (BVH) suitable for fast collision checking. It allows for the visualization of agents' trajectories and to answer more complex queries in visual question answering. The model 'forgets' parts of scenes by pruning graph nodes, and stores instances of the same object efficiently as duplicates. Lastly, it can use the metric-semantic mesh as input to a physics simulation and perform robotic agent actions in this simulation to predict their outcomes.

The spatial perception engine is split into core and DSG modules. The core module contains visual-inertial odometry, fast single-frame and multi-frame mesh generation, semantic segmentation based on video frames, and loop detection as well as pose graph and mesh optimization. The core module is described in more detail in [RACC20].

The DSG module performs tasks for higher levels of the scene graph, including unknown object detection, pose detection for objects with known 3D models, human detection,

and the Kimera BuildingParser for places, structures, rooms and buildings. The BuildingParser uses Voxblox Skeleton from [OTSN18] for place detection.

As Rosinol et al. point out, an important advantage over [AHG+19] is the provision of traversability and other actionable information, allowing robots to plan navigation and interactions with the environment. In addition, their spatial perception engine takes dynamic entities, such as humans, into account, which is necessary for navigation in crowded areas.

Hughes et al. [HCC22] optimize the spatial perception engine of [RVA+21] for the online generation of hierarchical scene graphs in real-time. Their work addresses several performance issues of the Kimera engine, in particular the fact that the ESDF used in [RVA+21] does not scale well with scene size.

The ESDF is based on an estimated robot trajectory, and as the trajectory changes with loop closures, each loop closure would require a re-computation of the scene graph. In addition, places and rooms are extracted with batch processing, which is incompatible with real-time requirements.

As the first real-time spatial inception engine, the main contributions of this paper include the reconstruction of a local ESDF, which builds meshes and generalized Voronoi diagrams incrementally from the ESDF. This allows to create a graph of places, and to cluster places into rooms in real-time, with room segmentation based on community-detection.

Loop closures and graph optimization are based on hierarchical descriptors in the scene graph. They use top-down closure detection with statistics collected in graph nodes, and bottom-up geometric verification to register putative matches.

In their evaluation on real and simulated data, the engine shows comparable performance to batch processing with respect to the accuracy of 3D reconstructions, and outperforms existing approaches in the quality and number of loop closures.

Wald et al. [WNT22] extend on previous work [WDNT20] and provide a method that works on point cloud data, uses novel sparse convolutions, and learns semantic segmentation and instance embeddings from 3D data. A graph prediction module predicts class labels of object nodes and edges directly from scene features. As in previous work, scene graphs in this paper are "semantically rich and particularly dense." Nodes contain attributes and subject-predicate-object relationships as in [WDNT20].

In contrast to earlier publications, this work does not rely on prior knowledge in the form of segmentation masks, and uses a 3D backbone with sparse convolutions, as opposed to PointNet in [WDNT20]. It includes color and normals for semantic feature learning and uses an embedding space to segment nodes. Graph nodes are initialized with segmented clusters, as opposed to the ground-truth segmentation of previous work. As such, this method can be applied to real-world setups.

The authors note that scene graphs can be especially helpful e.g. for changing indoor scenes and matching image against different 3D scenes with lighting and object changes.

For scene similarity, scene graphs are an intermediate representation, and "fundamentally resilient to dynamic environments." The usefulness of this representation is demonstrated in their application to scene retrieval tasks.

2.4 Semantic Parsing

As can be seen from related works on scene graphs and object-oriented SLAM, scene understanding often relies on semantic parsing, i.e., object detection and segmentation, to create higher-level models. Zheng et al. [ZZZ+19] identify semantic parsing and scene classification as the two main problems of scene understanding.

Here we only provide a short introduction to the topic as it relates to scene understanding. Since the focus of our work is less on the methods themselves and more on their application in the context of object-centric mapping, we refer the reader to additional literature for a more in-depth discussion and comparison of related work.

In the image domain, common terminology refers to object detection as detection with bounding boxes [HGDG17]. Semantic segmentation applies per-pixel class labels to images, but does not detect individual objects. Instance segmentation combines both approaches in that it detects and classifies object instances, and generates a segmentation mask for each. As Wan et al. [WLY+21] note, semantic segmentation methods are "sensitive to viewpoint changes and also suffer from inconsistent predictions across different views."

2.4.1 Object Detection

Mask R-CNN by He et al. [HGDG17] is an object detection and instance segmentation network that belongs to the class of two-stage detectors. Two-stage detectors first predict a set of region proposals based on the output of a feature extraction network. These proposals are refined in the box regression head, and each detection is classified. For instance segmentation, an additional instance mask is generated.

One-stage, or single-shot detectors, on the other hand, directly predict classes and bounding boxes without region proposals. According to [CGFC22], two-stage detectors typically have better performance at the expense of efficiency.

Both kinds of detectors can be seen as meta-architectures, as they can be used with different kinds of feature extraction networks. For example, [HGDG17] report significant gains in accuracy and speed for a Feature Pyramid Network (FPN) as the backbone to Mask R-CNN, compared to a Residual Network (ResNet).

2.4.2 Semantic Segmentation in 2D

Pham et al. [PDC+18] combine bounding boxes from object detection with the Detectron network and edge detection in images, and apply simulated annealing for the final segmentation result. Their image partition sampler is one of the main contributions

of this paper and generates a region boundary hierarchy, where object instances are represented by regions of the hierarchy.

Instead of predicting a segmentation mask for each detection, this approach generates a global pixel-wise image segmentation. It can detect known and unknown objects in an open-set environment, and does not require instance masks for training.

Panoptic Segmentation

By definition, instance segmentation tasks are focused on detecting and segmenting only countable objects or persons, also referred to as 'things' in the literature. In addition, segmentation masks can overlap, as each object is segmented separately. This is in contrast to semantic segmentation, where each pixel in an image is classified exactly once. Classes in semantic segmentation can represent 'things' as well as uncountable parts, or 'stuff', such as grass, sky, walls, or roads.

Kirillov et al. [KHG+18] introduce panoptic segmentation as a new type of task that combines semantic and instance segmentation in a coherent way. They also propose Panoptic Quality (PQ) as a new evaluation metric, designed to capture all aspects of panoptic segmentation and being easy to compute and interpret.

The task definition includes both stuff and thing classes and, like semantic segmentation, must assign a class label to each pixel. Additionally, each pixel is assigned an instance ID, and pixels with the same class and instance belong to the same object. For classes of uncountable parts, the instance ID is ignored. A special void label is used for unknown or ambiguous objects.

Kirillov et al. note that the definition of which classes are countable, i.e., the split between things and stuff, is a design choice for the creation of datatsets, following the convention of previous works. A recent survey of panoptic segmentation methods can be found in the paper by Li et al. [LC22].

2.4.3 Semantic Segmentation in 3D

Segmentation tasks can also be performed in 3D, where they segmentation is typically applied to each 3D element, i.e., voxel, point, or surfel, or to regions in a scene. Wald et al. [WTS+18] present a method for online scene reconstruction and real-time semantic segmentation that does not require a GPU and is fast and efficient enough to run on mobile devices.

They use the geometric segmentation of [TTN15] to extract 3D segments from RGB-D sensor data, and compute fully incremental feature descriptors for segments based on their point clouds. A random forest is used for the classification of segments, and updated segments are fused into a globally consistent map.

The computational complexity of segmentation and data fusion depends only on sensor resolution, not on the size of the scene, as only updated segments have to be processed.

Evaluation shows the performance is comparable to other methods, but faster and more efficient.

Nakajima [Nak20] proposes two semantic mapping methods that assign class probabilities to regions of a map, or semantically annotate only known objects; that is, both approaches reduce the computational complexity and memory requirements compared to annotations of each map element, and use the geometric segmentation method of [TTN16].

Their first work [NTTS18] presents a surfel-based method for real-time semantic scene reconstruction that combines geometric and semantic segmentation. It is in part motivated by the demand for efficient semantic mapping suitable for mobile or embedded devices and applicable to unknown environments.

The framework consists of four components: SLAM, semantic segmentation with a CNN, incremental geometric 3D mapping, and class probability updates for 3D map segments. Camera pose tracking, SLAM, as proposed by [TTN15], and 2D semantic segmentation run in parallel, with a low-res CNN being used for semantic segmentation as inaccuracies are resolved by the label-fusion approach.

For each frame, geometric edge detection from the depth image is refined with semantic segmentation, as geometric segmentation alone cannot detect edges, e.g. between two flat objects. Segments are extracted from the refined edge map as connected components and fused into the geometric 3D reconstruction. The segment map of the scene is then rendered to the current image frame and combined with semantic segmentation for probability fusion to create a semantic 3D map.

This method assigns class labels not to surfels but to segments, in order to reduce computational and memory requirements. Grinvald et al. [GFN+19] note that geometric segmentation approaches tends to over-segment objects, and grouping of articulated scene elements into separate objects is generally not possible without instance-level information.

The second framework by Nakajima et al. [NS18] provides object-oriented mapping as a natural representation for object manipulation and task planning. It stores scenes at the object level, and can be applied to SLAM extensions, e.g. pose graph optimization and dynamic scenes.

According to Nakajima et al., most systems employ 2D object detection and instance segmentation to fuse class and instance labels with the 3D map. Their approach uses YOLO v2 as a fast object detector, and annotates each geometrically segmented region with bounding boxes. In this way, parts of objects that are over-segmented by the geometric approach can be detected and merged.

As an advantage over instance segmentation, geometric segmentation provides clear boundaries between objects in the 3D reconstruction. Similar to [NTTS18], this framework renders the geometric reconstruction to the current image frame and combines it with the object bounding boxes. The refined geometric map is then fused with class labels from the object detector, and class probabilities are updated in the final map.

Instance-Aware Semantic Segmentation

Pham et al. [PHNY19] generate 3D scene reconstructions in real-time on top of the voxel hashing pipeline, and combine 2D semantic segmentation with 3D mapping. The geometric reconstruction is integrated with the results of a semantic segmentation CNN to provide an initial 3D segmentation. This intermediate model is continually refined with a higher-order Conditional Random Field (CRF) to incrementally correct segmentation errors.

As this process can be computationally intensive, Pham et al. employ a progressive super-voxel clustering scheme to provide a new domain to the CRF and meet real-time constraints. To save memory for the integration of new data, each voxel only stores the current class label and its confidence, following the update strategy of SemanticFusion.

This work also includes an extension to instance segmentation, where the CRF outputs instance IDs instead of class labels. Instead of directly using instance information from 2D segmentation, which would require tracking segments over multiple frames, new instances are spawned from the largest connected component in each CRF inference step that is not assigned to an instance ID yet.

2D/3D Joint Semantic Segmentation

Hou et al. [HDN19] are the first to present an approach that jointly uses 2D and 3D features to predict object bounding boxes and semantic instance segmentation in 3D scans and is trained in an end-to-end fashion. They employ a fully-convolutional 3D architecture that is trained on scene parts and can perform single-shot inference. The system takes 3D scene geometry and RGB-D images as input, and back-projects 2D image features extracted with a segmentation network onto the voxel geometry. Features from multiple images are combined with max-pooling.

The network consists of two main components for object detection and masking. Each part has its own backbone for feature extraction, as the two-backbone structure is found to converge faster during training and provide better segmentation performance.

Object detection has two sets of 3D convolutions for small and large receptive fields, and each set defines anchors for small and large objects (larger than 1 m^3). A 3D RPN predicts object bounding boxes, and a 3D RoI pooling layer predicts classes. 2D color and 3D geometry features are forwarded to object detection and per-voxel instance mask segmentation.

Mask segmentation maintains the spatial resolution of input for best performance, and reduces features from object detection to class probabilities. On evaluation with real-world data, this method is shown to perform significantly better than other state-of-the-art methods.

Hu et al. [HZJ+21] note that 2D and 3D segmentation contain complementary information, such as fine details and geometry, that can be used to supplement each other at different levels, yet joint information processing is usually very limited and mostly unidirectional.

They propose a bidirectional projection module (BPM) that can transfer information between 2D and 3D processing and demonstrate its application on two U-Net type networks for 2D and 3D segmentation. At each layer of the U-Net networks, the BPM computes a link matrix and bidirectionally projects features to the 2D and 3D space.

Evaluations on real-world datasets show benefits for both 2D and 3D segmentation tasks and top performance on the ScanNetv2 benchmark. The authors believe the BPM to be useful for other tasks as well, such as classification, detection, and instance segmentation.

The work of Wan et al. [WLY+21] is motivated by the observation that many segmentation methods on 3D point clouds such as PointNet++, MCCNN, and MinkowskiNet focus on point clouds, but ignore geometry and semantic information details available in RGB-D sensor data. On the other hand, 2D semantic segmentation is sensitive to viewpoint changes and can vary considerably between views, making it difficult to achieve consistent results. To overcome these issues, 2D-3D joint end-to-end segmentation methods use information from 3D and 2D for improved accuracy.

Wan et al. propose a new architecture for the incremental generation of 3D reconstructions with a joint 2D-3D segmentation method. Their architecture includes object tracking, mapping, as well as 2D and 3D segmentation modules, to extend traditional SLAM with scene understanding. 2D object detection is based on geometric and semantic segmentation, and instances are detected as the union of segments from both segmentations.

In order to correct the segmentation results of partial object views, a sparse map stores geometric landmarks and semantic segmentations. The map is projected into the current image and projected regions are checked against instance segmentation results. The system keeps track of and updates class probabilities, and adds or removes objects based on predefined threshold values.

A dense map is generated in parallel for the final reconstruction and joint 2D-3D segmentation. The architecture extracts features from 2D segmentation and projects them onto the 3D voxel model. 2D features from multiple views are transformed by a Semantic Projection Block, and fused with the 3D features from a 3D-Net encoder. The combined features are then transformed back by the 3D-Net decoder for semantic predictions. Evaluation shows the ability to simultaneously create accurate reconstructions and semantic predictions.

2.5 Object Discovery

In addition to their frameworks for semantic mapping, Nakajima et al. [NKSK19] propose a novel method to discover new categories of objects for 3D scene understanding. Their work incrementally builds a 3D map, assigns deep and geometric features to regions, and clusters regions based on these features. A surfel-based 3D segmentation map is the key data structure for this method, and incrementally built and updated in 4 steps, for each input frame.

First, the geometric information from depth images is integrated in SLAM. For object-level segmentation, Nakajima et al. use an adapted RGB-D SLIC superpixel segmentation based on color and depth information. The superpixels are further merged with agglomerative clustering, based on the similarity of color, position, and normals, to penalize concave shapes. These 2D labels are then assigned to or updated in the 3D segmentation map.

Class discovery relies on this 3D segmentation map, and clusters object-level segments based on deep and geometric features. Geometric features are extracted with the Global Orthographic Object Descriptor (GOOD), computed from the 3D segmentation map projected into the local video frame. Deep features come from the last layer of a CNN applied to the 2D images and projected onto the 3D map.

Clustering of 3D segments is performed with the Markov clustering algorithm, which allows for a flexible number of nodes. For clustering, the distance between segments is defined as the weighted distances of geometric and deep features, with weights based on the entropy of deep features to account for unknown objects. The output of clustering is then stored as a cluster label per object.

2.5.1 Scene Change Detection

Langer et al. [LPV20] discover objects in 3D environments with a form of change detection: scene differencing uses multiple observations of environments and compares 3D models to detect changes in the form of new or moved objects.This approach does not depend on the knowledge of objects or prior training. However, global differencing suffers from sensor noise and mapping errors, and cannot distinguish between moved and new objects. When applied to single frames, scene differencing may miss objects if viewpoints do not cover the whole scene.

Langer et al. address these issues with post-processing on 3D data. Semantic segmentation of the whole scene is used to find object candidates. Post-processing extracts planes from the scene, and considers objects on planes as likely candidates. They then perform local matching of candidates using a previously recorded reference map of the scene. With this method it is possible to detect novel objects, even if they are only partially visible, in a manner that is robust to sensor noise.

Another work by Langer et al. [LPV22] proposes an object mapping method from partial 3D reconstructions in open-world scenarios that does not depend on trained classifiers or known 3D object models. Their method identifies objects as clusters on top of planes, and uses Point-Pair Features (PPF) for matching corresponding observations. It categorizes objects into static, moved, removed, and novel instances, can deal with partial reconstructions and clutter, and detects individual objects in heaps using only change detection.

Plane detection uses down-sampled and filtered point clouds extracted by Voxblox [OTF+17]. Plane parameters and way points are used to navigate a robot around each plane, focused at its center, to collect more detailed 3D data of objects.

In order to group objects into four classes for robotic tasks – static, moved, removed, and novel – the framework uses three different object matching strategies. For static objects, local matching finds potential candidates nearby. It registers point clouds with ICP (Iterative Closest Point), and if ICP converges, assigns object to the static class. If a match is limited to a subset of points, it performs region growing using matching points as seed points, in order to prevent over- and under-segmentation of splits.

Semi-local matching is based on PPF descriptors. It finds the ten best hypotheses, and filters planar and small objects to avoid false matches. Matching is performed with a bipartite graph. Found matches belong to the static or moved class. A second matching attempt with relaxed conditions can match objects in heaps, and also works for incomplete reconstructions. It uses the same region growing as for local matching, and the process is repeated until no more matches are found.

Global matching employs the same procedure as for semi-local matching, but considers objects from all surfaces. The PPF descriptor is used to create hypotheses and confidence scores as edge weights in a bipartite graph. Objects for which no match is found are assigned to the new or removed class.

2.6 Incremental Learning

While deep-learning based methods can provide excellent results on the tasks they are trained on, one of their main limitations is the fact that the knowledge is fixed once training is finished. Extending the knowledge to new tasks, classes, or domains typically requires extensive re-training [KMvB+20].

Pretraining and fine-tuning networks is a method to adapt or extend models faster and more easily. However, it leads to the loss of performance on old knowledge if fine-tuning is not jointly performed on old and new training data [SSA17]. Training models with a subset of classes after pretraining violates the statistical assumption about samples being independent and identically distributed, which leads to catastrophic forgetting [YPRL22]. The concept of incremental learning provides methods for extending networks that take existing knowledge into account to reduce the re-training effort.

A distinction can be made between incremental and open-ended learning in the way learning is scheduled. While incremental learning uses a predefined set of new classes to learn, open-ended learning presents the model with a continuously growing set of classes [KLT20].

Continual learning can also be described as task-, domain-, or class-incremental. Class-incremental learning presents new classes at each learning step, for example in object detection. Task-incremental methods use a classifier with multiple heads, one for each task, e.g. object detection and segmentation. Domain-incremental methods are targeted at performing the same task on different, but related data distributions, i.e., the focus is on a label shift in data instead of new classes [HZ21].

Continual learning can be further restricted to online or offline scenarios. In an online setting, training data is presented to the system only once, while offline training can process the same data multiple times. The online scenario is typically closer to real-world applications, but also more challenging [HZ21].

Incremental learning is generally categorized into rehearsal-, regularization- and parameter isolation-based methods [CGFC22]. Rehearsal uses a subset of old training data to prevent the loss of old knowledge when training on new data. This old data can be stored from previous training sessions, or it can be generated from a probabilistic model, for example if old data is no longer available [PKP+19].

Regularization can be prior-focused, in which case it prevents changes to important old parameters. Data-focused regularization takes the difference in activations between the old and new model as a regularization term to retain old knowledge. Parameter-isolation based methods use a subset of parameters for each task that are prevented from changing during re-trainig.

While incremental learning is a broad topic with a large area of applications, here we consider related work only in the context of scene understanding and its applicability to object-centric mapping. For a more extensive coverage of the topic, we refer the reader to several recent publications.

Parisi et al. [PKP+19] provide a review of lifelong learning methods with neural networks. Kasaei et al. [KMvB+20] review lifelong learning methods with a focus on object recognition and manipulation tasks for service robots. Qu et al. [QRX+21] present an overview of recent advances of continual learning methods in computer vision.

Joseph et al. [JRK+21] present an approach to class-incremental object detection based on meta-learning. While most approaches to incremental learning use a form of knowledge distillation, their method learns to modify the model gradient during training in order to adapt more quickly to new classes while avoiding catastrophic forgetting. This approach is task-agnostic and highly scalable in terms of model size.

Adaptation of the model gradient is realised by a set of gradient preconditioning matrices, which are inserted between layers of the object detector. A meta-training objective is used to learn these matrices such that updates for old and new tasks are optimized.

Joseph et al. also note that knowledge distillation poses a form of intransigence in the way that the distillation loss enforces predictions of the old model. They propose a new loss function that learns generalizable gradients to improve adaptability and counter the loss of old knowledge.

He et al. [HZ21] introduce the use of pseudo-labels derived from clustering to continual learning. This general approach can be applied to supervised incremental learning methods and allows to use training data for which no human-annotated labels are available. They propose to apply the old model as a feature extractor to new, unlabeled training data in each training step. They then cluster the extracted features and create a

new pseudo label for each cluster. Finally the old model is re-trained using the training data with added pseudo-labels.

Integration into regularization methods with a distillation loss is performed by replacing labels with pseudo-labels. For rehearsal-based methods it is possible to use exemplars based on cluster means. For bias-based methods that require a validation set, both the training and validation set can be created with pseudo-labels. He et al. also apply different clustering methods to their approach, but evaluation results indicate that the choice of clustering method is not critical for the learning performance in their experimental setup.

2.7 Open-World Scene Understanding

Joseph et al. [JKKB21] introduce the task of open-world object detection that extends object detection to an open-world setting. Unlike closed-set scenarios, where all object classes are known from training, images in this task may contain objects from unknown classes, which should be detected as such. In addition, the model should learn new classes in an incremental fashion as new information, in the form of class labels, becomes available.

In their work, they propose ORE as a method for open-world object detection that uses contrastive clustering, an unknown-aware proposal network, and energy based unknown identification. The object detection model is adapted from Faster R-CNN, but applies contrastive clustering to features from the residual block in the RoI head.

Contrastive clustering stores one prototype vector for each known class, and minimizes contrastive loss to separate classes in latent space. Clustering requires a prototype for the unknown class too, but labels for unknown objects are not available. Instead, background region proposals are used as potential unknown objects as a simple heuristic.

The RPN and classification head are modified to identify unknowns and label them accordingly. The modified network describes object features in latent space with energy based models. Due to contrastive clustering, there is a clear separation between known classes and the unknown class, and the energy distributions of their features can be modeled with a Weibull distribution.

To classify a prediction as unknown, the modified classification head compares the energy values of latent features for the known and unknown class distributions and assigns the unknown label to an object if the corresponding energy value is higher. During training, to mitigate catastrophic forgetting, ORE stores a balanced set of exemplars for all classes and use these for fine-tuning in each incremental learning step.

Gupta et al. [GNJ+22] present OW-DETR as a transformer-based approach to open-world object detection, noting several shortcomings of ORE. These include the use of a held-out validation set for estimating the energy-based classifier probability distributions, and the selection of a single latent prototype for the unknown category for performing contrastive clustering, which does not capture intra-class variations and may affect separation between known and unknown classes.

ORE uses background region proposals as pseudo-unknowns, which Gupta et al. conclude causes a bias towards known classes. They argue their pseudo-labeling scheme provides better generalization and is suitable for single-stage object detectors. The main features of OW-DETR include an attention-driven pseudo-labeling mechanism that proposes unknown candidates, a novelty classification branch that detects unknowns, and an objectness branch that distinguishes objects from background.

The pseudo-labeling mechanism selects region proposals with high attention scores that do not match any known class as unknowns. The novelty classification branch is then trained on known classes and the pseudo-unknowns to distinguish known and unknown objects.

For the objectness branch, knowledge transfer is used to learn the characteristics of known foreground objects for the unknown class. Evaluations show improved performance for the open-world and incremental object detection tasks over ORE with the MS-COCO dataset.

Zheng et al. [ZLH+22] introduce the task of open set object detection and discovery. A distinct feature of this new type of task is the automated labelling of novel classes for incremental learning, which has to be performed by a human in other settings. They propose a two-stage approach to this type of problem, consisting of Object Detection and Retrieval (ODR) and Object Category Discovery (OCD) modules.

The ODR module implements an open-set detector to detect known objects and localize unknown objects, and stores samples of known and unknown objects. The OCD module discovers new categories of objects from unknown classes with unsupervised contrastive learning and semi-supervised clustering.

For class discovery, unsupervised domain-agnostic mix-up augmentation and contrastive learning are applied to the set of samples from known and unknown objects to create more discriminating features in a latent space. These embedded features are clustered with a semi-supervised constrained k-means algorithm, which enforces the assignment of labelled instances to their ground truth classes.

Since k-means requires to specify the number of clusters beforehand, the number of novel categories is estimated with deep transfer clustering. For the initialization of clustering step, the centroids of known classes are manually computed. Experiments show that this method outperforms other baseline methods.

Zhao et al. [ZLS+22] revisit open-world object detection in their RE-OWOD framework. Its prominent features are a Proposal Advisor and a Class-specific Expelling Classifier (CEC). The advisor helps the RPN to detect unknown objects and distinguish them from background. The CEC removes unsure predictions, and prevents model from assigning known class labels to unknown objects. It adjusts the probabilities of test samples belonging to certain classes based on training set proposals and training ground truth. The expelling indicator adjusts activation boundaries with a class-specific expelling indicator. If the indicator is positive for a class, a proposal is not assigned to this class.

Instead, the proposal is assigned to the class with the highest score, selected from the remaining classes, or to the unknown class.

Zhao et al. additionally define five principles for benchmark construction and two fair evaluation metrics for open-world object detection from the perspective of the unknown class. The new metrics are Unknown Detection Recall (UDR) and Unknown Detection Precision (UDP). Both emphasize the distinction of unknown objects from background which is especially challenging in object detection without supervision.

Zhao et al. identify several challenges regarding metrics for open-world object detection evaluation. Unknown objectness must distinguish unknowns from background. Unknown discrimination must distinguish unknowns from known classes. Incremental conflict balances learning of previous and current known classes, with a focus on incremental learning.

The benchmarking principles are summarized as follows:

- Class openness: known and unknown class sets do not overlap and can exist in images at the same time during inference; for training, only known classes are labeled.
- Task increment: the set of known classes is increased incrementally. At each step a subset of unknown classes are annotated and added to the known set.
- Annotation specificity: for training and validation, only annotate known classes; for testing, in addition to known classes, annotate unknown instances as 'unknown'.
- Label integrity: label all objects for testing; unlabeled objects can lead to correctly identified unknown objects being considered as false positives.
- Data specificity: no overlap between training, validation, test datasets; no duplication within datasets.

Sylph is a hyper-network-based, continuous, fine-tune free incremental few-shot object detector proposed by Yin et al. [YPRL22]. It provides several key improvements over previous approaches, including decoupling object localization and classification, and a class-agnostic detector pretrained on base classes, i.e., classes with a high number of instances.

This work is motivated by the observation that traditional object detection relies on large training sets for building a model, with labor-intensive and time-consuming training. Real-world datasets with a long tail of classes contain few samples and take much longer to train.

Few-shot object detection tries to alleviate this problem. It can use base classes mainly in two ways. A model is either pretrained on base classes, and fine-tuned on a selection of seen and new classes. However, this is often not practical in real-world scenarios. On the

other hand, meta-learning teaches a model how to learn: the model is trained episodically in order to adapt to new classes faster.

The framework consists of two main components: an object detector with a class-agnostic bounding box regressor and a classification subnetwork, and a hyper-network that generates class-specific codes for the classifier. Both components share the same weights for their feature extraction backbones.

The hyper-network extracts features for a small number of support samples per class, and inputs these to a code generator module. The code generator is split into a code predictor head and a code process module. The former predicts weights and biases for each sample, while the latter aggregates the predictions into a single weight and bias, and normalizes the results to avoid gradient explosion.

Using class-agnostic localization is preferred for few-shot detection as it is difficult to learn localization from a few samples. Per-class binary classifiers allow adding new classes without changing the detection of seen classes.

The method achieves three important goals: incremental, sequential learning without additional training, detecting new and seen classes in one pass, and avoiding forgetting of learned classes. Evaluation shows that it provides effective training and improves accuracy over previous methods. It is the first incremental few-shot object detection method without test-time training with a performance close to fine-tuning-based methods on large scale datasets.

2.7.1 Open World Semantic Segmentation

A formal introduction of incremental learning for semantic segmentation is provided by Michieli et al. [MZ19]. Their focus lies on the most general setting, in which images from previous learning steps are not used, and new images can contain instances of known classes in addition to new classes. The general setting also requires an approach for scaling to a large number of classes.

Michieli et al. present an approach that distills knowledge from a previous model to a new one in each incremental learning step, and does not store images of previously learned classes, or older models. The loss function consists of a cross-entropy and distillation loss term, and several distillation loss functions are evaluated. The knowledge distillation scheme also includes freezing the encoder part of the new model as a way to preserve existing knowledge during re-training. Evaluation on the Pascal VOC2012 dataset show that this method outperforms the baseline of fine-tuning in most cases.

Cen et al. [CYC+21] are the first to introduce an open-world semantic segmentation system, consisting of a deep metric learning network (DMLNet) with contrastive clustering and two proposed methods for few-shot incremental learning. The authors motivate their work by the shortcomings of two common approaches to anomaly semantic segmentation: uncertainty estimation and generative models. These methods tend to either produce many false-positive outlier detections, or cannot model complex environments.

Cen et al. combine open-set semantic segmentation module and incremental few-shot learning to accomplish open-world semantic segmentation. Open-set segmentation consists of two parts, a closed-set and an anomaly segmentation module. The latter creates an anomalous probability map, where pixels with high probability are considered an anomaly. Class labels of other pixels are assigned by the closed-set segmentation module.

The closed-set segmentation module is realized with DMLNet, which embeds image features in a metric space using contrastive clustering. Classification is based on a set of prototypes, and pixels are assigned to classes by measuring the similarity of their embedded feature vectors to each of the prototypes.

Anomaly segmentation includes two identification criteria for out-of-distribution pixels: metric-based maximum softmax probability, based on the maximum class probability of closed-set segmentation, and Euclidean distance sum, based on the distance to class prototypes. During the lifetime of the model, class prototypes are fixed. Cen et al. note that learnable prototypes "cause instability during training and make no contribution to better performance."

Incremental few-shot learning updates the closed-set module when new labels are provided by a human oracle, and uses up to five images for re-training. For incremental learning, two methods are proposed. Both assume that a single class is added in each learning step.

The Pseudo-Labeling Method (PLM) is based on several branch heads, one for predicting base classes, the others for predicting one new class each. Given a labelled image, this method combines all predicted masks and the label image to train the new branch head. For inference, the same procedure is used, without a label image. The Novel Prototype Method (NPM) calculates a prototype for the new class as a mean feature of all instances, and assigns the new class label if a prediction is closest to the new prototype and its distance is below a threshold.

The authors report "state-of-the-art performance on three challenging open-set semantic segmentation datasets without using additional data or generative models."

The method of Uhlemeyer et al. [URG22] provides unsupervised incremental semantic segmentation and class discovery in open-world settings. Their main contribution is a modular procedure for learning novel objects without manual annotation. They use segmentation quality estimates to identify regions of unknown classes in segmented images, then cluster regions and apply pseudo-labels to segments in order to retrain the segmentation model with new classes.

This method is related to novelty detection with cluster similarity measures for class discovery, class-incremental learning with knowledge distillation and rehearsal methods, and open-world recognition. In this work, Uhlemeyer et al. define novelties as unseen objects that constitute a new concept. Their approach to semantic segmentation with unknown classes first assigns a softmax probability to each pixel for every known class using a conventional network.

They then estimate the prediction quality on segments, i.e., connected regions with the same label, using the average dispersion measure over segments. Additionally, geometric properties are recorded for each segment, and both metrics are fed into a meta-regressor.

This regressor is trained on the metrics and true IoU of all segments to estimate the prediction quality. It then identifies segments below a threshold as unknown. Connected segments that are predicted as unknown are merged into image patches as potential unknown objects.

These image patches are clustered into new categories in two steps. First, they are processed by a CNN trained on ImageNet to extract high-level and high-dimensional feature vectors. The feature vectors are compressed with PCA and t-SNE to create a low-dimensional embedding.

These embeddings are clustered with DBSCAN, where only core points of clusters are considered as cluster members, and clusters below a minimum size are discarded. Each remaining cluster represents a novel category that is used to re-train the segmentation model. Under ideal conditions, the segmentation performs perfectly on in-distribution data, the meta-model detects all (but only) unknowns, and novel objects of different classes are separable.

The semantic incremental learning uses a rehearsal- and knowledge distillation-based approach. It extends the initial model by replacing the final layer in the model and re-initializing affected weights. The class weights of the cross-entropy loss are set to the inverse class frequency for each batch. The rehearsal method adds images to the re-training set that contain underrepresented known classes, or ones that are similar to novel classes. To find similar classes for novel objects, this method counts the pixels from each novelty assigned to known classes by the old model.

2.7.2 Open World Instance Segmentation

Cermelli et al. [CGFC22] revisit the standard data-focused regularization-based knowledge distillation framework to model missing annotations for incremental learning in object detection. Following conventional incremental learning protocols, each learning step contains only annotations for new classes, and any annotations for old or future classes are missing. Their work is founded in the observation that in this type of setting, new training steps containing only annotations for new classes leads to models learning to classify old classes as background. In addition, new classes are considered as background in previous training steps, making it more difficult to extend the network.

Cermelli et al. provide adapted loss functions to handle missing annotations in training data for object detection, and include an extension of their approach to the task of incremental learning for instance segmentation. They reformulate the loss functions in the knowledge distillation framework such that they take into account the probability of a model predicting old classes on regions without annotations, or predicting current classes as background. Evaluation shows that their approach outperforms other state-of-the-art

methods for the object detection task, and outperforms other baselines for instance segmentation.

Kim et al. [KLA+22] propose a classifier-free Object Localization Network (OLN) based on localization-based objectness learning. Their approach is capable of learning generalizable objectness purely on how well proposals match annotated regions, and uses localization quality estimators for region proposals at the image level and for each RoI. Their work is motivated by the insight that classification in existing methods can overfit to labeled objects, associating novel objects with the background. The OLN avoids such foreground-background classification in favor of increased generalization capabilities.

OLN is build on Faster R-CNN, which is also used as a baseline for evaluation. The model replaces classification in the RPN and the RoI head with localization quality estimates, such as centerness and IoU. Kim et al. demonstrate that this approach shows better generalization to unknown classes and across different datasets, and present an extension to open-world instance segmentation by adding a class-agnostic mask head akin to Mask R-CNN.

The authors see potential applications for learning-based proposals in various areas including open-world object detection, segmentation, robot grasping, video understanding, and large vocabulary detection. Their evaluation shows that this method outperforms other approaches on rare and common classes for the large vocabulary dataset LVIS.

CHAPTER 3

Unsupervised Incremental Learning in Open-World Object-Centric Mapping

Our system is intended to provide a perceptional basis on which to perform various lab monitoring tasks, as mentioned previously. In particular, this involves tasks performed by mobile autonomous robots, as stationary cameras cannot cover all use cases and the continuing development and advancements in robotics promise to free lab workers from highly repetitive, tedious, and dangerous tasks.

Object-centric scene models are a natural and versatile abstraction of computational representations of physical environments. These models also form the basis for higher level abstractions, in particular scene graphs. In practical applications, mobile robotic systems are often confronted with unstructured, open-world environments.

There we can make little assumptions about the geometric nature of the environment, and most object instances that a computer vision system encounters are unknown at the time of system design and implementation. It is therefore necessary to be able to extend the system's knowledge during operation in an efficient and effective manner.

Our problem statement can be summarized as follows: we want to create a system that generates a metric-semantic map of unstructured environments from visual input (P1). We specifically target visual input because of its feature-rich, contextualized imagery which unstructured point clouds obtained from 3D scanners often lack.

The system shall segment the map at the object level in an online fashion (P2), and be able to detect unknown objects (P3) using category discovery, and group instances of unknown objects by similarity (P4). Finally, it shall learn new categories of objects

in an unsupervised incremental learning step to extend the object detection model and semantic segmentation (P5).

Various methods from related works exist that have already covered each of these steps individually or in combination with other methods. However, to the best of our knowledge this is the first time that all five are combined in one system. This problem statement poses the question of how the interaction between different parts of the system should occur.

To approach this issue, we also need a system design that facilitates a working solution (P6). This framework should be modular and flexible so that it is possible to exchange and compare different methods for each part with minimal changes to the overall design.

To evaluate the system performance, we need to define a primary metric and evaluation protocol. Since we do not know of any previous work which approaches the same computer vision task, it is necessary to come up with our own evaluation protocol (P7).

One particular challenge is the fact that methods for novelty detection and category discovery are inherently stochastic and it is not possible to predict in advance which categories will be detected and learned. Hence we adapt the evaluation from related computer vision tasks, such as panoptic segmentation and class-incremental learning in 2D.

The performance evaluation is specifically intended to gain insights into the following research questions:

1. How does unsupervised incremental learning in object-centric SLAM affect object detection performance?
2. How does novelty detection perform?
3. How does category discovery perform?

3.1 Design Choices

Following these premises, there are several practical considerations and constraints that limit the selection of methods, software, and hardware for the implementation. The system should be able to work on mobile robots equipped with small, low-cost visual sensors. In particular, this excludes larger and more expensive LiDAR scanners.

In order to generate a 3D reconstruction of a scene, depth information is needed as well. According to Labbé and Michaud [LM19], monocular SLAM approaches suffer from scale drift over time. Alternatives include stereo or RGB-D cameras and visual-inertial odometry.

With the advent of affordable, high-quality RGB-D sensors, these are a good match for this task. The proposed framework does not make any assumptions about the source of

Figure 3.1: Scene 286 from the SceneNN dataset reconstructed as a point cloud using our framework, and visualized with flat rendering in Open3D. Point colors correspond to global segment labels.

sensor data, but accurate alignment between RGB and depth frames is desirable. With more recent consumer hardware it is also possible to purchase RGB-D cameras where RGB and depth sensor are mounted on the same stiffener, allowing for a high-quality and affordable integrated solution with better alignment for 3D reconstruction tasks.

3.1.1 Scene and Object Representations

The robotic tasks this system is targeted at also affect the choice of scene and object representation. Common choices are unstructured representations, namely point clouds and surfels, regular representations using voxels in combination with Signed Distance Fields (SDFs), and meshes as irregular representations.

Point clouds and surfels (short for surface elements) are unstructured representations that are the result of point sampling the environment with a depth sensor. Besides 3D coordinates, points can have additional attributes, including color and point normals [ZSG+18]. An example of a colored point cloud created with our framework is shown in Figure 3.1.

Point clouds are a lightweight data structure in terms of storage requirements and required post-processing from raw sensor data. They support arbitrary resolutions for objects and scenes, and resolutions can be highly varying within and between individual objects, depending on the required amount of detail.

Surfels are point primitives with additional attributes computed in a preprocessing step optimized for fast, high-quality rendering. Algorithms specialized for hardware-accelerated visibility calculations and rendering can achieve real-time frame rates even for large datasets. In this way, surfels essentially provide a fine-grained trade-off between memory overhead, quality and performance [PZVBG00].

Point clouds and surfels lend themselves to global map optimizations through local transformations, in particular for loop closures in online scene reconstructions, as demonstrated by Schöps et al. [SSP19]. Since this type of representation does not enclose a volume, it can also be used to model small and thin objects without loss of detail. However, Schöps

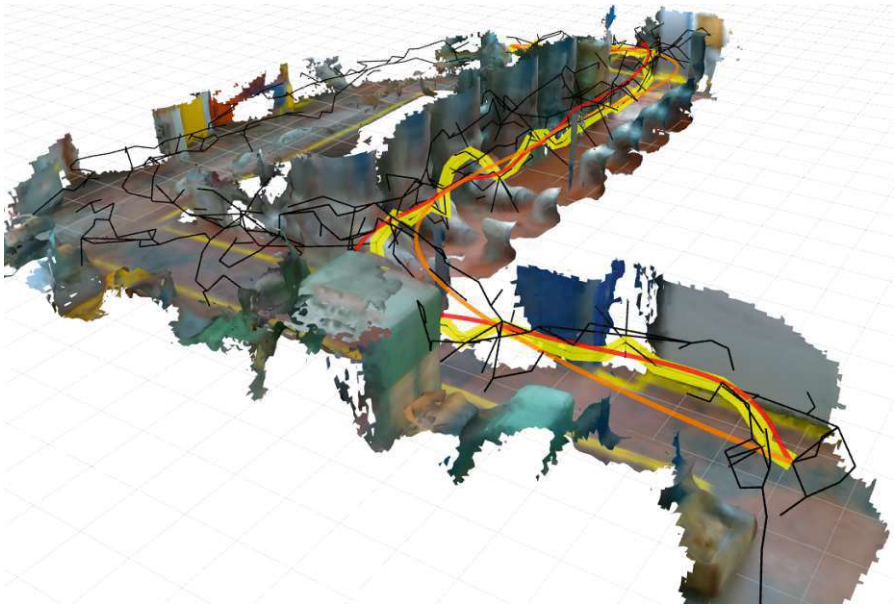


Figure 3.2: A scene reconstructed for navigation and path planning with a voxel-based SDF representation, from Oleynikova et al. [OTSN18].

et al. also cite the discrete representation of objects with point-based methods as a main drawback, and meshing as a possible solution.

On the downside, unstructured 3D representations do not provide any information on surface connectivity. As such, visibility calculations and collision detection are more difficult to compute. A disadvantage, in particular for mobile robotics applications, is the lack of distinction between free and unobserved space, i.e., there is no explicit modelling of free space. This means point clouds need additional information for collision-free path planning [OTF+17].

Voxels are a regular volumetric 3D representation with a fixed resolution for each scene or object. The memory overhead is higher than for point clouds as the voxel data structures explicitly store free and occupied voxels, as well as the spatial layout of each volumetric element. On the other hand this allows to directly model observed free space, for example to facilitate robotic path planning and navigation tasks [OTSN18].

Basic implementations of voxel data structures also have a fixed size, limiting the amount of space that a recording can cover. However, more advanced approaches dynamically allocate voxels to overcome this limitation. Simply storing free and occupied space as voxels provides poor surface approximations. Additional per-voxel distance information leads to a higher-order, implicit surface representation based on different forms of signed distance fields. Figure 3.2 contains an example of a scene reconstructed with TSDF and ESDF representations.

Voxel-based 3D representations typically integrate data over multiple frames to account

for imprecise and noisy measurements, and can store additional probabilities for class and instance labels [GFN+19]. More recent voxel-based approaches also allow for tracking and reconstructing multiple dynamic objects simultaneously [GTSN21]. As Grinvald et al. [GFN+19] note, reconstructions from voxel grids contain explicit surface connectivity information which can be used e.g. for robotic object manipulation tasks.

One particular downside of voxel representations are more difficult intra-object transformations compared to point clouds. This makes retargeting parts of a global map during online reconstruction computationally more expensive [HCC22]. As point clouds are more suitable for keypoint detection and registration (e.g. with ICP), voxel-based SDFs can be transformed into point clouds with the marching cubes algorithm as is done for example by [FNF+18].

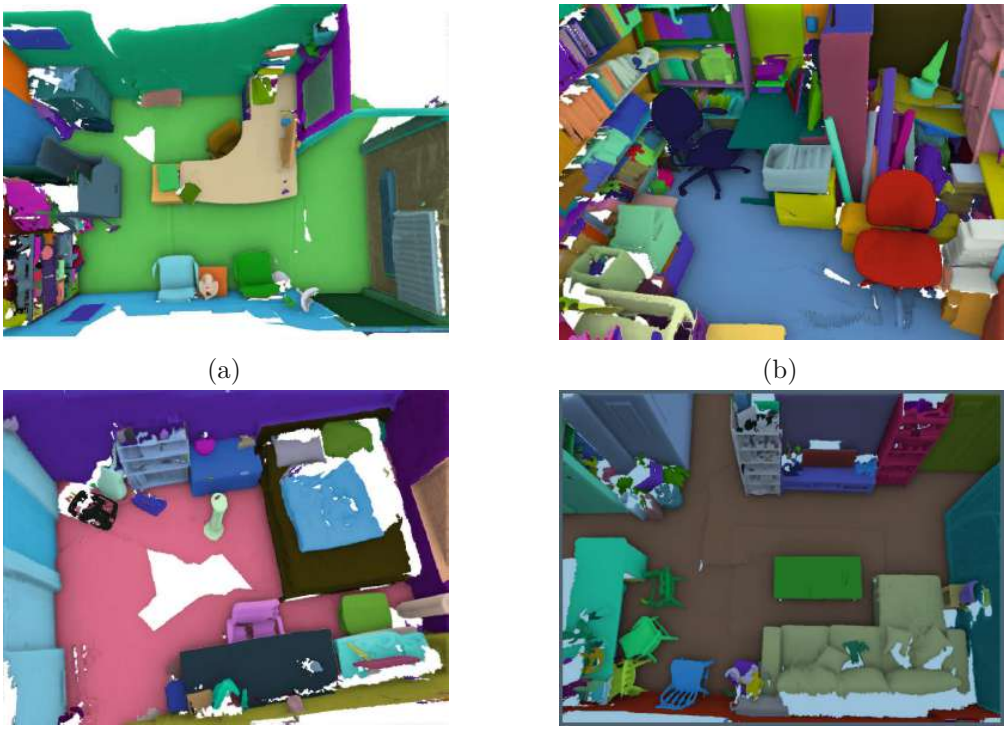
Meshes are an irregular 3D representation and another common option for modeling scenes. Meshes can be stored in compact data structures and support surface connectivity. In particular, for scenes with many planar regions it is possible to approximate these surfaces with a low-resolution model without a significant loss of precision. However, the piece-wise linear nature of meshes leads to poor approximations for curved surfaces unless the amount of detail is appropriately increased. Figure 3.3 gives several examples of scene reconstructions from the SceneNN2016 dataset.

The inherent surface connectivity in meshes facilitates visibility calculations and collision detection. Like surfels, meshes allow for textured rendering, which is used in some methods of scene understanding for projections of global segmentation map into video frames. Geometric segmentation approaches can also make use of mesh connectivity when determining object boundaries, as is done for segmentation refinement in [HPN+16], for example. As a disadvantage, algorithms for mesh processing tend to be more complex when taking topology into account.

Because of their intricate nature, meshes are typically provided as output of post-processing point clouds or voxels in SLAM applications, and not as intermediate representations for scene reconstruction [RACC20]. Algorithms that convert point clouds and surfels into meshes include Marching Cubes and SurfelMeshing [SSP19]. Similarly, voxels can be transformed into point clouds with Marching Cubes [FNF+18].

The approach of using an unstructured internal representation for scene reconstruction and triangulating meshes from them combines the advantages of both. In particular, this provides SLAM with a dense scene representation that can quickly adapt to loop closures. During scene reconstruction the generated mesh can be used as a compact model e.g. for a conventional rendering pipeline or collision detection [RVA+21].

For the purpose of this system, a voxel representation is the primary choice as it allows to integrate noisy data and uncertain semantic and instance segmentations into a consistent global map in a unified manner. This is crucial to achieve reliable results for metric-semantic mapping in the face of segmentation predictions varying from frame to frame. As mentioned, a volumetric representation also stores observed free space explicitly, allowing to use this information directly in potential path planning and navigation



(a) (b) (c) (d)

Figure 3.3: Several scenes from the SceneNN dataset, reconstructed as meshes and colored with instance labels, from Hua et al. [HPN+16].

tasks. Voxblox++ [GFN+19], the method used in our system design, features a meshing algorithm for transforming the voxel model. In our implementation, the mesh output serves as the model for all visualizations of results and for the comparison of scene segmentations with ground truth in the performance evaluation.

3.1.2 Target Domain

The application of our method is primarily targeted at lab automation, and thus includes mostly industrial settings with various devices and pieces of equipment as well as workstations and offices. Even if those indoor environments show a high degree of structure, this method is designed to make little assumptions about scene geometry.

More specifically, the environment does not have to conform to a Manhattan or Atlanta geometry, i.e., the assumption that walls are aligned to a rectangular grid, or there is one vertical direction and several directions orthogonal to it. We also do not employ the plane hypothesis for object segmentation, that is, there is no assumption that objects are most likely to be found on flat surfaces that are mostly horizontal, as is done, for example, in [LPV22].

The geometry is only taken into account during depth segmentation, where the depth

image is segmented based on the convexity of contiguous areas. This type of segmentation however is not limited to a particular scene category but derived from a more general idea of 'objectness' [KMFF13].

There is no hard limit on the size of scenes or the level of detail with which they are recorded, although in the current implementation there are no provisions for out-of-core data processing during recording. Hence, scene size is mainly limited by the amount of main memory.

For the novelty detection node, which consumes most of the main memory in our implementation, out-of-core processing can be achieved by simply storing collected data on disk until it is processed in a sequential fashion for category discovery. More sophisticated data collection schemes could also be devised, providing a trade-off between memory, storage and processing requirements, as briefly described later on.

3.1.3 Scene Perception

Scene perception is the process of transforming sensory input into a model of objects and the environment. In this regard, the object representation module is an important aspect as its output is used for learning as well as object recognition [AKC+22].

On one hand, the type of object representation depends on the kind of sensory input to the system. RGB cameras typically have a higher resolution than depth sensors, and provide photometric information, i.e., color and texture, as additional low-level features [BGT+20]. Some 3D laser-based depth sensors, as used in many recent hardware, may provide less accurate measurements with reflective surfaces and around object edges [TDCH21]. For this reason it can be beneficial to use both modalities for 3D reconstruction.

Another design choice involves the dimensionality of representations for the object recognition task. In this case, we use Mask R-CNN as a state-of-the-art object detection approach that also provides high-level features for the class discovery module. This approach allows us to easily combine the geometric-semantic segmentation of Voxblox++ with the incremental learning method of MMA and class discovery akin to [URG22]. The system therefore detects objects and segments the scene in the color and depth input images before integrating each 3D frame into the global map, including semantic and instance labels.

Arguably, it would be conceptually simpler to integrate each input image into the global map first and then perform segmentation on the 3D representation. Object detection in 2D has the advantage that its result are available to the robotic system as early as possible, preferably at each video frame. With segmentation in 3D, 3D data collection and geometric segmentation precedes object detection and semantic segmentation, as for example in [WWT+21b].

One of the well-known disadvantages of deep learning-based methods is the need for large training datasets [GFN+19]. For classes with small numbers of samples, deep learning

tends to perform poorly, resulting in low detection rates and frequent misclassifications. This is a problem in particular for long-tailed class distributions, with a small number of frequent classes and a large number of rare classes, that often occur in real-world settings. The same need for 'big data' poses a major challenge in incremental learning. Training on only new classes violates the i.i.d. (independent and identically distributed) assumption about the training data and thus has a negative effect on previously learned classes [YPRL22].

Exemplar-based representations of object classes such as Sylph in [YPRL22] provide a retraining-free alternative to class-incremental deep learning. However, Sylph only features object recognition without segmentation and is thus not directly applicable to our framework.

Topic modelling as with Local-LDA [KLT20] and Local-HDP [AKC+22] is a generative model that uses topics as a latent structure to represent data. Its object representations are compact and can be processed efficiently. In contrast to instance segmentation in 2D, Local-LDA assumes objects are already segmented in 3D before categorization.

This segmentation would have to be performed beforehand, either based on some general geometric assumptions such as the plane hypothesis, or with a separate segmentation model. In the latter case, this model would have to be (re-)trained separately as well, making this approach less attractive for incremental learning.

3.1.4 Novelty Detection, Category Discovery, and Incremental Learning

The object detection and segmentation model for our system is intended to be trained initially on a set of common classes that occur in many different categories of environments, such as walls, floors, desks, and chairs. The idea is that these classes serve as a form of basic knowledge of the environment and guide the category discovery task as the system tries to 'fill in the gaps' between known objects and structures in the scene.

For novelty detection, all unknown segments of the scene are considered novel, following the panoptic segmentation approach in [URG22]. For simplicity, we assume that all newly discovered categories belong to the object ('thing') type, as opposed to structures ('stuff'), i.e., each manifestation of a category is a separate instance in the segmentation model.

During operation, the robot system occasionally encounters novel objects in real-time over longer periods of time. During each recording, the system only gets to view objects from a few different angles. Since active exploration is not included in the framework, it is not possible to optimize viewpoint selection, resulting in limited data for the incremental learning step.

It is however possible to retroactively select for each novel object a variety of views from different angles, once the recording is finished or a certain amount of data has been collected. This intends to make the best use of the limited data at hand. We combine this

idea with the recording of multiple scenes before each category discovery and incremental learning step.

In the best case scenario, this allows the system to collect observations of similar novel objects from different scenes, and propose more generalized categories for incremental learning. Following these design decisions, the system design opts for a simpler batch approach to incremental learning instead of a continuous one.

3.1.5 Software Modules

In order to limit the implementation of necessary functionality to a reasonable amount of work, we rely on existing open-source software for the framework where possible. Due to the complexity and tight schedule, re-implementing methods from the literature is considered beyond the scope of this project.

As a consequence, the choice of methods for each part is limited to those whose authors have made their implementations publicly available. This in turn affects the system design as a whole, as not all methods can be combined and integrated with equal ease.

Thankfully, there are a number of recent papers with code available from which to choose from. We adapt these methods where necessary to implement the functionality of our framework. The framework also relies on a large number of other open-source software to facilitate rapid prototyping of the implementation, most notably ROS[1], PyTorch[2], Open3D[3], and related projects.

3.1.6 Exclusions from Scope

Since the framework already covers a wide variety of methods to solve the problem at hand, it is especially important to define any exclusions from scope. This helps focus on the most relevant aspects of the problem, and measure the performance of the system with respect to a small number of components that form the proposed solution.

Regarding robotic platforms that use the framework, the system does not concern itself with path planning, navigation or low-level motion control. It should however be noted that such tasks can be implemented on top of a voxel-based scene representation as provided by the system.

The system does not include any method for localizing the robot or camera, and thus relies on an external source of reliable odometry. Likewise, there is no mechanism for camera calibration or alignment between RGB and depth images in the framework, instead sensor data is taken as-is.

The evaluation does not consider the effects of odometry, calibration and alignment on mapping. Since evaluation mostly deals with the effects of incremental learning on two

[1] https://www.ros.org/
[2] https://pytorch.org/
[3] http://www.open3d.org/

different mappings produced from the same sensor data, these are of less concern to the given problem and the interpretation of results.

The framework does not provide any means to detect or model quasi-static, moving, or flexible objects over time. It also does not perform object tracking or human sensing. While there are various recent publications dealing with these topics, these are excluded from the scope of this work to focus on unsupervised incremental learning in the context of object-centric mapping.

For the same reason, active exploration and object manipulation for the purpose of interactive learning are not included. However, these topics are of great interest as well in extending the capabilities of robots in real-world scenarios and may be considered complementary approaches to unsupervised incremental learning.

With regards to unsupervised incremental learning, the focus is on the application and integration of existing learning techniques in the context of open-world object-centric SLAM, rather than developing new methods in this area.

Lastly, this framework relies on publicly available, annotated datasets for the purpose of training, validating and testing our implementation. Creating high-quality, fully annotated datasets of 3D environments is a time-consuming and labor-intensive task, as indicated by Dai et al. [DCS+17], that goes beyond the scope of this work. Using existing datasets also has the advantage that it allows for easier comparison with previous and future methods in this field of research.

3.1.7 Software Framework

With the software targeted at deployment in mobile autonomous robots, it is important to take into account the distributed nature of such applications. The operational range and runtime of mobile robots is usually limited by their battery capacity. For this reason, the on-board computational resources are typically much more limited than that of a contemporary consumer PC or server hardware.

Integrating less computing hardware reduces the load on and power consumption of the robot, but also reduces their capabilities as a computing platform. In particular, robots often lack a high-performance GPU (or any at all), which means they are not well suited for many deep learning based approaches that rely on fast parallelization of memory-intensive calculations.

Retrofitting high-end PC components to a robot not only faces the issue of increased power consumption. It is also hindered by a lack of, or high cost, for ruggedized hardware that is designed to withstand the sometimes extreme conditions robots have to operate in.

It is therefore desirable to have a system that can distribute computations over several machines, or nodes, connected by varying network links. In such systems, mobile platforms are at the edge of the computational cloud, executing tasks like data acquisition and

inspection. They transfer collected sensor data over the network to other nodes, typically servers, for processing or long-term storage.

Data transfer can occur continuously, during operation, or in batch mode, whenever the robot has finished a mission. The latter case is of particular interest when robots operate in environments without a permanent network connection or the bandwidth is too low to support large data transfers.

Developing an application in a distributed framework is a fundamental design decision that affects the overall system design as well as individual components that comprise the communication nodes [WWWK96]. It has long been known that programming paradigms which work well in tightly coupled, local applications do not scale at the network level, as Treichel et al. note [TH01]:

> "The infrastructure, it was thought, could be built to remove all of the differences between references to a remote service and references to a local service. This paper argues that all such attempts are doomed to failure and can, at best, enable the development of systems that are fragile, prone to unavoidable errors, and restricted in scale."

Instead, it is necessary for distributed software development to take potential network and system failures into account and explicitly handle concurrency issues:

> "In particular, we argued that distributed infrastructures must present a model of partial failure to the programmer, since only at the application level can such failure be dealt with; must deal with concurrency issues, rather than leaving them to the infrastructure; and must at the application level realize what parts of the program are local and what parts are at least potentially remote." (ibid)

Data-Oriented Architecture

Data-Oriented Architecture (DOA) [Jos07] is a data-centric approach to the design of distributed systems that offers loose coupling between components and fault tolerance at the network level with explicit Quality of Service (QoS).

DOA replaces remote procedure calls (RPCs) between networked components with a publish-subscribe model with deterministic resource management. Each computational node in the network can publish and subscribe to an unlimited number of topics. Topics are channels for the transfer of messages, or application-specific data types. Each domain contains a set of topics that are shared between nodes.

Multiple nodes can receive the same message, and publish messages on the same topic. Message filters limit the range of data nodes receive on a topic, and can synchronize and combine data from different topics, e.g. RGB and depth images as well as odometry.

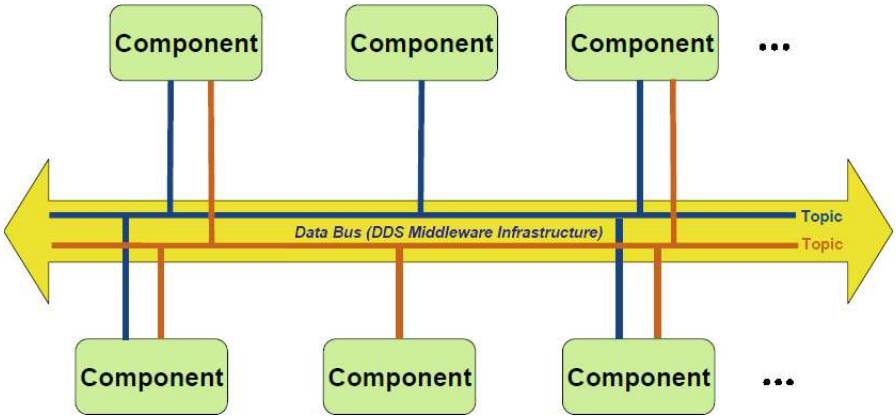


Figure 3.4: Illustration of Data-Oriented Architecture concepts: data instead of function calls are the main aspect of communication in a distributed system. Loosely-coupled components publish and subscribe to data topics over a central message bus. The middleware dynamically manages topics and QoS, and supports establishing direct data paths between components. Image from Joshi [Jos07].

Nodes are publishers, subscribers, or both, on one or multiple topics. Figure 3.4 illustrates the concept of DOA.

In this model, neither publishers nor subscribers need to know about other nodes sending or receiving messages, allowing for loose coupling between nodes. The only information that nodes need to share is the metadata about topic data types in the form of message definitions. It is noteworthy that RPCs and Service-Oriented Architecture (SOA) can be implemented on top of a DOA middleware [Jos07]. In this regard, they are considered a special case of DOA.

DOA can be useful for different types of machines or nodes in a distributed system: edge systems have real-time constraints with 'real-world' functions, for example robots, instrumentation, radars, or communications equipment.

Enterprise systems provide higher-level user interaction, decision support, and storage and retrieval of historical data. They work in 'soft' real-time, with a time scale at human level. Examples of enterprise systems include application servers, web servers, and applications.

Systems-of-Systems (SoS) are comprised of many edge or enterprise systems, including other SoS. They are loosely coupled with many independent entry points, an independent control domain, and realize multiple objectives. Figure 3.5 exemplifies the SoS approach on a fictitious airport control organisation.

ROS, the Robot Operating System[4], is a set of libraries and framework for the development and robotic applications. In addition to a wide variety of algorithms and methods related

[4] https://www.ros.org

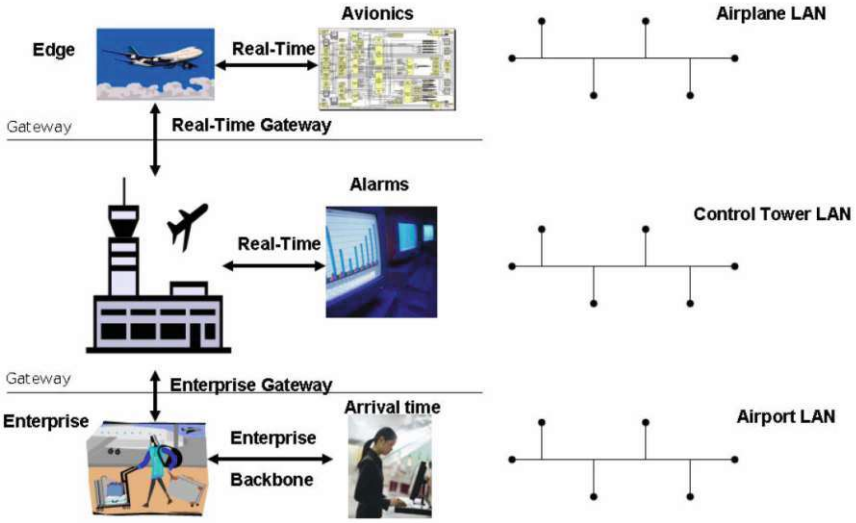


Figure 3.5: Illustration of a data-oriented system-of-systems on the example of an airport control operation: DOA ties together multiple heterogeneous systems with different time scales. Each system uses its own communication network and connects to other systems over strictly defined network gateways. Image from Joshi [Jos07].

to computer vision and robotic tasks, it features a DOA middleware for data transfer between ROS nodes, with support for QoS parameters to declare the quality of data transmission.

ROS version 1 provides a proprietary implementation of DOA, while ROS 2 uses the Open Management Group's (OMG) Data Distribution Service (DDS) [Spe07] and Real-Time Publish Subscribe Protocol[5] (RTPS) standards. The discovery of components requires a central 'core' node in ROS 1, but is decentralized in ROS 2. While ROS 2 marks an important technological update to the ROS framework, our implementation is based on ROS 1 for easier integration with some of the existing software used in the project.

3.2 System Design

Following the general problem description and design decisions, the system design can be broken down into five main components. Each component implements a specific functionality within the framework, and is implemented as its own computational node. Nodes communicate with each other over ROS topics, and can be run on different machines if necessary. In particular, the RGB-D camera input node is intended to be run on the robot itself, while the more computational-heavy components can be offloaded to server hardware.

[5] https://www.omg.org/spec/DDSI-RTPS

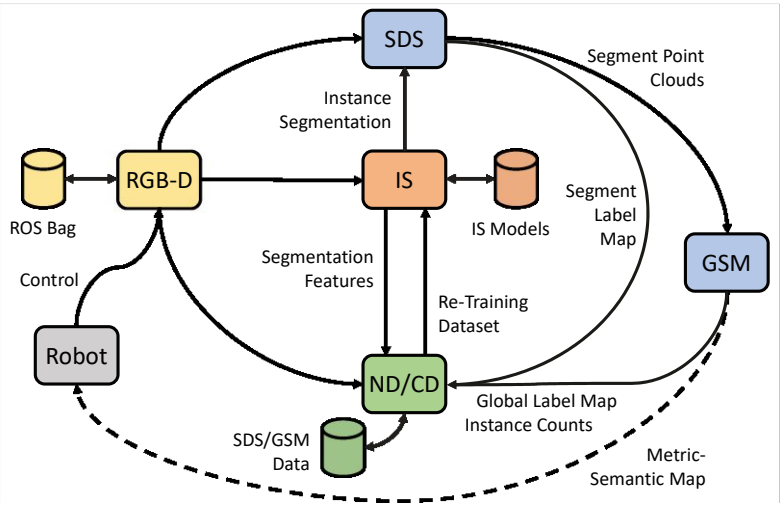


Figure 3.6: Overview of the system design, including main components, data flow, and data storage. The system consists of five main components, implemented as distributed computing nodes. Video input from the RGB-D sensor node is processed by instance (IS) and semantic depth segmentation (SDS), and integrated into the global segmentation map (GSM). The novelty detection node (ND) collects data during scene reconstructions and performs category discovery (CD) on selected segments to prepare a retraining dataset for the instance segmentation model. Three data stores facilitate testing and debugging of the framework, and store intermediate and persistent data.

Figure 3.6 shows the high-level concept of the system design with its five main components, including all data transfers between nodes and essential data storage required for operation. In this diagram, the outer loop, containing the RGB-D sensor input, depth segmentation, and global segment map, represents a basic SLAM approach without instance segmentation or category discovery. Two inner nodes provide instance segmentation, novelty detection and category discovery to the system. Not shown in this diagram are the service calls that individual nodes offer and which can be used, for example, to orchestrate the test and evaluation setup.

3.2.1 RGB-D Sensor Node

The RGB-D node publishes RGB and depth images, including camera calibration information, as individual messages, which are received by the depth segmentation, instance segmentation, and category discovery nodes. Besides providing color and depth images, the RGB-D sensor node also transmits odometry data as a separate topic.

While odometry is essential for the operation of the SLAM algorithm, its computation is considered outside the scope of this work. Instead we assume an external source that provides this data with sufficient accuracy. In practice, this would come from the robot's internal sensors, an additional module implementing, for example, visual-inertial

odometry, or in the case of a test setup, be included with the dataset.

3.2.2 Metric-Semantic Segmentation

The instance segmentation node (IS) performs instance-aware semantic segmentation on each RGB video frame and publishes the results for geometric-semantic segmentation. Segmentation results include a bounding box, segmentation mask, class label, and confidence score for each object detection. Geometric-semantic segmentation (SDS) partitions the depth image into convex regions, or segments, and combines them with the semantic segmentation results.

Segments in the depth image that overlap the same instance mask are merged and assigned the class label of the instance. Segments that do not match an instance are labeled as unknown. The depth segmentation node then transforms each 2D segment into a 3D point cloud and publishes these point clouds for the global segment map.

The 2D segmentation is also published in the form of a segment label map. This label map assigns a unique label to all pixels of each segment in the depth image, i.e., it acts as a segmentation mask for the geometric segments.

3.2.3 Global Segmentation Map

The global segmentation map (GSM) node takes on the task of integrating the per-frame data into a consistent, global metric-semantic map. Primarily, this task consists of merging points into voxels of the map, and keeping track of class and instance labels for each voxel. This includes mapping the segment labels, which are unique only within a frame, to global segment labels. Segments that overlap by a certain amount are also merged in this stage.

The map is continuously updated during the system's operation and can be published as a labeled mesh at regular intervals (for example, every two seconds) or following an RPC request. Vertex attributes of the mesh include segment, instance, and class labels. In this way, the global map can be easily used as input to the generation of higher-level scene representations such as scene graphs.

3.2.4 Novelty Detection and Category Discovery

The final parts of the framework are novelty detection (ND) and category discovery (CD), implemented in a single node. This node collects scene data over longer periods of time (in the range of several minutes to half an hour) and, when triggered, performs novelty detection and category discovery on collected data to generate a data set for retraining the segmentation model in an incremental learning step.

For this task, the node collects data from all other nodes, including RGB images from the sensor node, segmentation feature maps from instance segmentation, segment label maps from geometric-semantic segmentation, and global segment maps. Global segment

maps are published by the global map node and assign a global segment label to each segment in the segment label maps.

This way the novelty detection node can keep track of which segments in each video frame represent the same global segments. The global segment map also contains information about merged segment, to keep the internal representation of segments the novelty detection node in sync with the global map.

While there is no use of the global map on the side of the robot in our system, Figure 3.6 indicates potential uses in future systems by a dashed line between the global segment map node and the robot. The global map could for example be used to steer exploration of the environment by actively optimizing viewpoints and planning paths to improve the 3D reconstruction. With such an extension, the framework would form a closed control loop allowing robots to continuously explore and improve their understanding of the environment.

3.2.5 Data Stores

The system design contains three data stores, of which two are essential for operation. The two essential ones store the instance segmentation models and data sets pertaining to category discovery and incremental learning. The instance segmentation models include the model that is currently used by the segmentation network at each point in time, and loaded from the store during startup of the segmentation node. Models adapted from the current one by means of retraining in an incremental learning step are also stored here for use at the next iteration of scene recordings.

The data store for category discovery is mainly needed because of the size of the recorded data sets, which range from 5 to 15 GB per scene (uncompressed), depending on the amount of RGB-D video data. Since we record multiple scenes (five in our evaluations) before each incremental learning step, the system may run out of memory if intermediate data is not stored on disk after each scene recording. For novelty detection, the collected data is loaded from the data store and preprocessed scene by scene before collectively processing it in the category discovery step.

The final training data set for incremental learning is then written to the data store as well, together with metadata for the data loader in the deep learning framework. Storing this intermediate data also has the advantage that it allows to test novelty detection, category discovery, and incremental learning with different algorithms and parameters without having to re-run the whole object detection and mapping pipeline.

The third data store is connected to the sensor node, and can record or replay sensor data. Data is stored in the form of ROS messages, with full message headers including time stamps and divided into topics, in a ROS-specific file format called ‘bags’. Replaying such files mimics the output of ROS nodes and as such can be used to test the system repeatedly with new recordings, or existing recordings from publicly available data sets. It also allows to run the system in offline mode, where a robot only records raw sensor

data during its mission and puts it in the data store afterwards, from which it is then processed by the framework.

3.2.6 Synchronization

Nodes can be restarted individually and in any order without disrupting the system operation. Message queues in publishers and subscribers store data until the respective node is ready to transmit or receive messages. However, queue sizes are limited to prevent out of memory conditions and nodes can miss messages if they are offline or too slow, which can affect the results of 3D reconstruction.

For evaluation, we receive multiple messages from different topics with strict time synchronization where possible. For example, to ensure that RGB video frames are processed with their corresponding segmentation results, the instance segmentation node publishes its results with the same time stamp as the RGB image the results are derived from. On the receiving end, the depth segmentation node uses strict synchronization on the RGB image and segmentation result topics so that corresponding messages are always received together.

However, strict synchronisation is not possible for sensor input and odometry due to sampling at different points in time (and possibly at different rates). Depending on the type of RGB-D sensor, color and depth images may be captured independently, resulting in slight time shifts and thus time stamps for each message.

Likewise, sampling points of odometry data may not match visual sensor data if derived from a different source such as visual-inertial tracking built into a robot. In this case, nodes rely on approximate time synchronization of topics, and for odometry, on interpolation provided by the ROS tf2[6] library.

It should be noted that, due to slight variations in timing and the concurrent nature of processing, the number of processed messages, and consequently the results of 3D reconstruction and object detection, may vary between different runs even when using the exact same sensor inputs, loaded from a data file.

As an additional optimization, we route RGB image messages through the instance segmentation node, instead of passing it directly from the sensor node to depth segmentation and category discovery. The instance segmentation node receives all RGB image messages, and re-publishes those on a new topic which it has processed and published results for.

Depth segmentation and category discovery nodes only subscribe to this new topic and thus only receive messages processed by instance segmentation. In this way, network load is reduced and RGB image topic subscriber queues do not fill up with messages that are never processed.

[6] http://wiki.ros.org/tf2

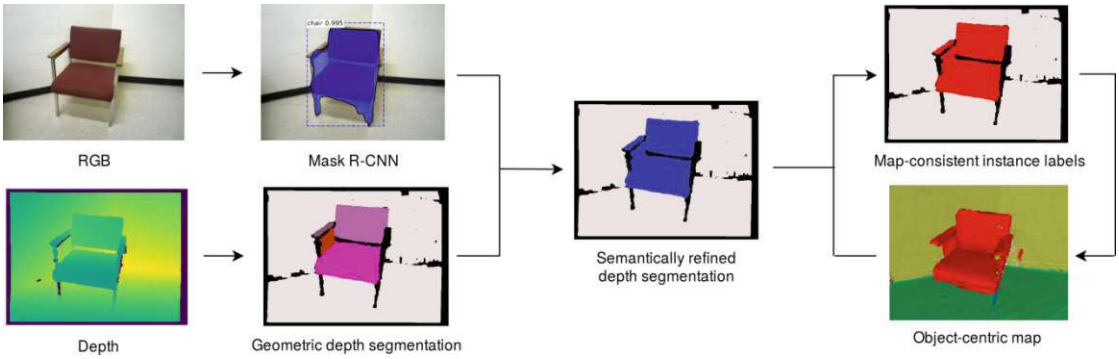


Figure 3.7: Overview of the data processing pipeline in Voxblox++: Each pair of RGB and depth images is segmented with instance and geometric segmentation, respectively. Both results are combined in a semantic refinement step, matching segments to object instances. The object-centric map integrates each video frame, and keeps track of segments and object instances over a complete recording session. From Grinvald et al. [GFN+19]

3.2.7 Additional Modules and Tools

The core functionality is extended by additional modules and tools to cover the full scope of work from initial training to evaluation of results. For training and evaluating the system, we combine the scene geometry, annotations, and 2D instance segmentations of the SceneNN2016 and SceneNN2018 datasets, and convert the recordings of sensor data into the bag file format with our own tools. A separate tool extracts a subset of this data, including metadata, as an initial training set for instance segmentation.

For the initial training and incremental learning steps of the instance segmentation model, we re-use the code supplied with [CGFC22] and adapted to our framework. Another tool of ours performs the evaluation of the metric-semantic mapping results before and after each retraining step. Additional code was written to provide the visualizations of results for this thesis.

Test runs for generating data for evaluation and retraining are largely automated with an additional ROS node. This node orchestrates the playback of sensor data for multiple scenes in sequence, data collection and category discovery in the novelty learning node, and storing meshes from the global segmentation map after each scene reconstruction session. The incremental learning steps and segmentation model updates themselves are not fully automated, as the additional complexity of implementing and testing this functionality was not considered worthwhile in the context of this project.

3.3 Object-Centric Mapping

The 3D scene reconstruction and semantic mapping part of the framework is based on Voxblox++ by Grinvald et al. [GFN+19]. It is an approach to image-based semantic

mapping and object discovery and incrementally builds volumetric, object-centric maps from RGB-D sensor data.

Voxblox++ combines works from object detection in RGB images and dense 3D reconstruction, and extends object detection to novel objects of unseen categories. It builds accurate metric-semantic maps with pose and shape information for objects and unknown object-like instances.

The volumetric map representation is targeted at navigation and interaction planning tasks, allowing these tasks to derive free space and connectivity information from the reconstructed maps. The 3D representation is based on Voxblox by Oleynikova et al. [OTF+17] and uses a voxel data structure to store a truncated signed distance field (TSDF).

The main features of Voxblox++ in the context of this framework are the geometric-semantic segmentation scheme, the data association strategy, and map integration. Geometric-semantic segmentation can be further divided into geometric segmentation and instance-aware semantic refinement. These features comprise the four basic steps that are used to process each RGB-D video frame for scene mapping. Figure 3.7 illustrates these steps of the Voxblox++ pipeline.

3.3.1 Geometric Depth Segmentation

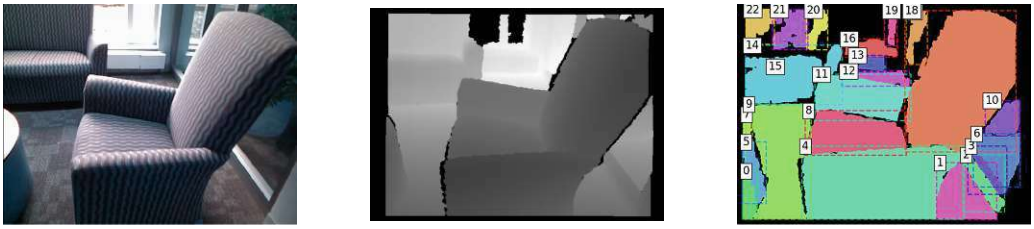
Geometric segmentation follows the method of Furrer et al. [FNF+18]. It is a convexity-based approach that detects real-world physical boundaries in depth images associated with each video frame, and outputs a 2D image of segment contours. Figure 3.8 shows an example of RGB and depth input and the resulting geometric segmentation.

The segmentation process detects contours based on depth information. It assumes objects are mostly convex shapes or comprised of convex shapes. The segmentation process first estimates normal vectors from the depth image. It then finds concave regions by comparing angles between adjacent normal vectors.

Lastly, it measures the 3D distance of adjacent pixels to locate large depth discontinuities. These depth discontinuities indicate physical boundaries between different objects. The geometric segmentation then creates a set of regions from both the concavity and depth discontinuity measurements.

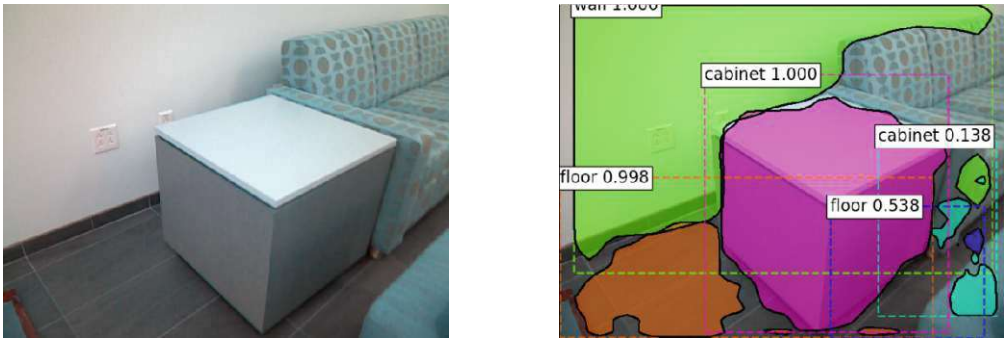
This partition of the image into a set of closed 2D regions forms the basis of map elements for the 3D reconstruction. Using information about the current camera location, the 2D regions can be transformed into corresponding 3D segments in the global map coordinate system.

This step alone is usually not sufficient as an object detection mechanism to provide meaningful segmentation, as a convexity-based approach tends to over-segment objects and cannot make a distinction between different object instances. An example of over-segmentation can be seen in the geometric depth segmentation step of Figure 3.7.



(a) (b) (c)

Figure 3.8: Example of geometric depth segmentation on a single video frame, as performed by Voxblox++. Subplots show, in order: a pair of RGB and depth input images, and the segmentation result of similar depth frame. In the result, each segment is marked with a numbered bounding box for visualization purposes. RGB input is included for reference only and not used in geometric segmentation.



(a) (b)

Figure 3.9: Example of instance-aware semantic segmentation on a single video frame, as performed by Mask R-CNN Benchmark. Subplots show RGB input on the left and the segmentation result on the right. Results are visualized with colored segmentation masks, bounding boxes, and class labels.

3.3.2 Instance-Aware Semantic Refinement

Instance-aware semantic refinement has the purpose of inferring categories for 3D segments and grouping them into distinct object instances. The refinement step first applies instance segmentation to the RGB image, which detects object instances in the image with bounding boxes, and predicts segmentation masks and class labels for each object. It then assigns segments from the geometric segmentation step to object instances. Figure 3.9 gives an example of instance segmentation results visualized with object bounding boxes, segmentation masks, and class labels.

Semantic refinement associates segments with objects by the overlap between their corresponding 2D regions and segmentation masks. Each segment is assigned to the object with the highest overlap, if the overlap percentage is above a predefined threshold. Such segments are then said to belong to this object instance, and to the category of the

(a) Object-Centric Map

(b) Ground Truth Instance Map

(c) Semantic Instance Segmentation

(d) Geometric Depth Segmentation

Figure 3.10: Comparison between instance segmentation, geometric segmentation, as in [FNF+18], and object-centric mapping, as provided by Voxblox++. Compared to geometric segmentation, the object-centric mapping approach can prevent over-segmentation of non-convex objects (blue circle). In comparison with instance segmentation, Voxblox++ can also detect objects without clear geometric boundaries (red circle). All images from Grinvald et al. [GFN+19].

object. Segments not assigned to any object are left without semantic information.

If multiple segments are associated with the same object they are presumably over-segmented and belong to the same non-convex object instance. A comparison between examples of object-centric mapping depth segmentation, and semantic instance segmentation is shown in Figure 3.10.

3.3.3 Persistent Data Association

Initially, there is no correspondence between segments from different frames. As Pham et al. [PHNY19] point out, the straightforward approach of propagating instance-based segmentation from 2D to 3D is complicated by the fact that the segmentation network only predicts one frame at a time.

It is therefore necessary to track instances over time, which Pham et al. regard as a challenging problem in this approach. In this regard, instance tracking is the most

important feature of Voxblox++ when it comes to instance-aware semantic mapping.

Voxblox++ solves this problem and matches per-frame predictions to segments in the global map with its data association strategy. This strategy defines persistent geometric segment and object instance labels that are unique and valid during the whole scene mapping session. It maps local segments and object instances in the current frame to persistent, or global, labels based on the overlap of local and global segments in the global map.

For each frame, the data association strategy matches predicted segments to segments in the global map, and object instances from the current frame to persistent instance labels. Persistent segments that match predicted segments can only be those which are visible in the current frame, as there can be no overlap otherwise. Finding the best match for each segment involves projecting the segment points into the voxel map and counting the segment labels to which the voxels belong.

For each visible segment, the corresponding predicted segment label is the one with the highest overlap. Matches only occur if the overlap is greater than a threshold of 20 %, to prevent matches between poorly overlapping segments.

The data association formulation disallows matching more than one segment in the current frame to each segment label in the map. This makes it possible to fix under-segmentations over time, i.e., two or more distinct segments stored as one. If matching more than one segment to each label were allowed, distinct segments would always be matched to the same under-segmented label of the map.

Instance tracking matches persistent segment labels to object labels, based on pairwise counts in the global map between segment and instance labels. Specifically, for each segment with an object instance in the current frame, if it is not assigned to a persistent instance label yet, it is assigned to the instance label with the highest pairwise count. If there is no match, a new object label for this instance is created. These counts are initialized and updated in the final data integration step.

3.3.4 Global Data Integration

The last major step in the processing pipeline involves the incremental fusion of geometry, class, and instance information into a global TSDF volume. For this purpose, Voxblox++ extends Voxblox with the incremental fusion of class and instance labels.

As in Voxblox, this step fuses 3D segments in current frame into the global map, and in a similar fashion does so for class and instance labels. We refer the reader to [OTF+17] and [GFN+19] for a detailed description of the map integration method and implementation. Through the additional information stored in the map, each 3D segment is defined by the sets of voxels containing its segment label.

Finally, for each per-frame segment with an associated per-frame object instance, this step updates the pairwise counts between its corresponding global segment and object

label. Pairwise counts between global segment and class labels are updated in the same way. A formal description of all steps can be found in [GFN+19].

As Grinvald et al. note, the quality of the 3D reconstruction is in part affected by camera pose estimation errors that, in their evaluation, accumulate to up to 0.5 m. However, as their work focuses on mapping, they leave the issue of inaccurate localization and its effect on map quality to future work.

In a similar vein, the focus of this work lies in the effect of incremental learning on metric-semantic mapping, and the use of object-centric scene representations for novelty detection and category discovery. Inaccuracies in 3D reconstruction are therefore less of a concern for evaluation, especially when the same or very similar scene reconstructions can be recreated from recorded sensor data repeatedly, but with different instance segmentation models.

3.3.5 Adaptation for Novelty Detection

The Voxblox++ implementation provides several additional features that are particularly useful for the visualization and evaluation of mapping results. These include publishing the global volumetric map as a point cloud, and storing the map and individual segments as meshes. The map mesh provides per-vertex annotations for persistent segment, instance, and class labels.

The global segmentation map node offers RPC services for requesting a snapshot of the map and resetting the map after each reconstruction session. We use these services in the control node for orchestrating the evaluation of different scene recordings from the SceneNN datasets.

There are only a few changes to Voxblox++ needed to integrate it into our framework. First of all, while the code base of Voxblox++ comes with an implementation of Mask R-CNN for instance segmentation, we use a different implementation, namely Mask R-CNN Benchmark[7] from Cermelli et al. [CGFC22], that is modified for incremental learning. This change is as simple as replacing the instance segmentation node in Voxblox++ with our own version, as the format of inputs and outputs of the nodes are basically the same.

We extend the geometric segmentation node to publish a segment label map for each video frame, i.e., a 2D image that assigns a per-frame segment label to each pixel of the depth image. Likewise, we collect and publish the mapping of per-frame segments to persistent segment labels in so-called global segment maps.

These global segment maps also include information about segment merges that occur in the semantic refinement stage. in In this way, the category discovery node has all the information required to track segments over multiple video frames.

[7] https://github.com/facebookresearch/maskrcnn-benchmark

3.4 Instance-Aware Semantic Segmentation

Instance-aware semantic segmentation is performed by a Mask R-CNN network, adapted for class-incremental learning by Cermelli et al. [CGFC22]. They in turn base their adaptation on the so-called Mask R-CNN Benchmark[8] implementation.

Mask R-CNN is itself a further development of Faster R-CNN, a two-stage object detection network with improved region alignment, and adapted for instance segmentation. As such, it first applies a class-agnostic region proposal network (RPN) to the image.

This network produces a configurable number of region proposals from a limited number of rectangular areas with fixed aspect ratios, called anchors, and with binary objectness scores. Non-maximum suppression (NMS) eliminates overlapping proposals, and the remaining proposals are fed to the classification head.

The classifier assigns for each region and every class a confidence score and a bounding box regression. Regions that are assigned a class label other than 'unknown' are then processed by the class-specific segmentation head to produce the instance segmentation masks, in the form of one binary mask per region and class. Proposals labeled as unknown, or background, do not have an associated segmentation mask.

3.4.1 Inference Output

The inference stage of the network directly outputs the segmentation results as bounding boxes with associated class labels and segmentation masks. This stage of the network is used as-is, without any changes to its functionality. The segmentation results are akin to panoptic segmentation [KHG+18] in that the networks predicts both objects and structures, also referred to as 'things' and 'stuff' in the literature.

One difference to panoptic segmentation is the fact that structure segments, such as walls and floors, receive unique instance labels during map integration. However, segments with a geometric continuation are typically merged so that there are only a small number of structure instances in each scene.

It should also be noted that scene graph generation can make use of structures as instances when deriving support relationships and other relations between scene elements, for example in the work of Wald et al. [WNT22].

Unlike panoptic segmentation, the masks in instance-aware semantic segmentation can overlap, due to the process of calculating each mask individually for every detection. However, this does not pose a problem for subsequent stages in our framework, as segments are only assigned to the object instance of a mask with which they have the largest intersection, and instance assignments are integrated over multiple frames to eliminate noise and spurious detection results.

[8] https://github.com/facebookresearch/maskrcnn-benchmark

3.4.2 Backbone Network

The RPN, classifier, and mask head process each image based on features extracted by a 'backbone' network. Mask R-CNN, in the implementation at hand, comes with two types of backbones: ResNet and FPN. According to [HGDG17], using an FPN backbone leads to gains in both accuracy and speed.

We use a ResNet-50 backbone with C4 architecture as the default, the same that is used for incremental learning in instance segmentation by [CGFC22]. The backbone network has a depth of 50 layers and the final convolution layer 4 is used for feature extraction by the downstream tasks.

Inputs to the inference stage of the network use the native resolution of RGB sensor data at 640×480 pixels. The final layer of the backbone network outputs a feature map with a spatial resolution of 40×30 and 1024 channels. This feature map is used by the box and mask heads, and also published for the category discovery stage.

3.4.3 Initial Training

Initial training of the segmentation network is done with ImageNet pretraining initialization and fine-tuning on the domain dataset, which can be considered a de-facto standard approach for object detection [LZZ21]. ImageNet is a large-scale visual classification challenge and dataset. The pretraining approach uses classification data and low-resolution images to train a classification network from scratch.

The feature extraction stage of this network is then transferred to a network designed for the downstream task, for example object detection, instance segmentation, or pose estimation. Fine-tuning of the second network is applied with a dataset for the downstream task, and with higher-resolution images.

This 'backbone initialization' intends to provide the second network with a more generalizable feature extraction stage from which the target vision tasks can benefit. A big drawback is the GPU and RAM heavy nature of this pretraining process, which can take multiple days to complete even on large compute clusters, depending on the batch size and training schedule [GDG+17].

For initial training of the instance segmentation network, we use a freely available Detectron model[9] pretrained on ImageNet as the feature extraction backbone. Fine-tuning is performed with a set of images and metadata extracted from the SceneNN2016 and SceneNN2018 datasets, and containing a selected subset of classes forming the basic knowledge of the model.

Pretraining vs. Training from Scratch

Alternatives to ImageNet pretraining include Direct pretraining, Montage pretraining, and training from scratch. He et al. [HGD18] show that it is possible to train object detectors

[9] https://dl.fbaipublicfiles.com/detectron/ImageNetPretrained/MSRA/R-50.pkl

from scratch with longer training times. They also show that ImageNet pretraining accelerates convergence but does not improve the detection performance of the final network.

Training from scratch uses larger image sizes in the training stage, which leads to increased memory consumption and in turn reduced batch sizes. To counteract this issue, training from scratch introduces Group Normalization (GN) and Synchronized Batch Normalization (SyncBN) as new modules for normalization. However, GN increases memory consumption and model complexity, and SyncBN increases the runtime of the training procedure as it requires cross-GPU communication.

The Montage pretraining approach trains the feature extraction network on a classification task just like ImageNet pretraining, but needs a classification head that is specific to this method. For this training step it uses montages of images from the downstream task data set as input, which consist of the assembly of multiple cropped images. According to the comparison by [LZZ21], Montage pretraining achieves comparable performance to ImageNet pretraining in a fraction of the time. The downside of this approach is the additional complexity in training.

Direct pretraining [LZZ21] offers a simpler approach to pretraining with higher accuracy and faster training steps. Compared to training from scratch with GN, inference is also faster. This method provides a trade-off between image resolution and batch size, with a small image resolution during initial training and a large resolution in fine-tuning, equivalent to ImageNet pretraining.

This allows for larger batch sizes in the first training step, and increases the performance when combined with regular batch normalization. Like training from scratch, this approach only uses the target task dataset as input. The study of [LZZ21] shows that performance, as measured by mAP, increases with batch size and image resolution, and saturates at higher parameter values. In their experiments, a batch size of 8 and image resolution of 640 in width achieve the best performance.

3.4.4 Training Parameters

Parameter	Initial Training	Incremental Learning
Batch Size	8	4
Base Learning Rate	0.01	0.0004
Steps	10000, 12500	3250
Iterations	18000	10000
Weight Decay	0.0001	0.0001
Learning Rate Decay	0.1	0.1
Momentum	0.9	0.9

Table 3.1: Instance segmentation learning parameters

We choose learning parameters based on the same network architecture in [CGFC22], which are summarized in Table 3.1. In contrast to [CGFC22], the batch size is increased to 8 in initial training, and the base learning rate and number of iterations are increased to 0.01 and reduced to 18000, respectively, according to the Linear Scaling Rule [GDG+17].

In the retraining step, all images are cropped first to remove the black borders in depth images, which do not cover the full field of view of the RGB sensor. This border and its effect on depth segmentation masks can be seen in Figure 3.8.

Since the RGB image extends beyond the borders of the visible area in the depth image, not cropping the dataset images would skew the retrained model towards ignoring objects near image borders. For initial training, this step is not required as the instance segmentation masks and bounding boxes are derived from renderings of the ground truth mesh and not from depth images.

For data augmentation in initial training, we use random horizontal flip as the only transformation, which is enabled in Mask R-CNN Benchmark by default. In retraining, transformations additionally include random crops to a size of 480×360 and resizing to full image dimensions.

Image normalization uses the same values as used by the pretrained model, which are $(102.9801, 115.9465, 122.7717)$ for pixel means, and $(1.0, 1.0, 1.0)$ for standard deviation, with RGB pixel values ranging from 0 to 255 in each color channel.

3.4.5 Incremental Learning

When supplied with a data set of new classes, the instance segmentation model can be retrained in one or more incremental learning steps. For this step, we use the method of Modelling Missing Annotations (MMA) from Cermelli et al. [CGFC22].

This work provides a new approach to class-incremental learning for object detection and an extension of the method to instance segmentation. It is applicable to two-stage object detectors, in particular Faster R-CNN and related architectures such as Mask R-CNN. The implementation is based on Mask R-CNN Benchmark, however its application is not limited to this particular network architecture.

Knowledge Distillation

MMA addresses 'catastrophic forgetting', that is the loss of performance on known classes when retraining the network, by revisiting the knowledge distillation framework from Shmelkov et al. [SSA17] for object detection and instance segmentation tasks. Knowledge distillation is a form of regularization that involves two models, a student and a teacher model.

During learning, the student model is trained to mimic the output of the teacher model, and learn the new classes from the retraining dataset at the same time. As per the standard training protocol for class-incremental learning, the retraining dataset contains

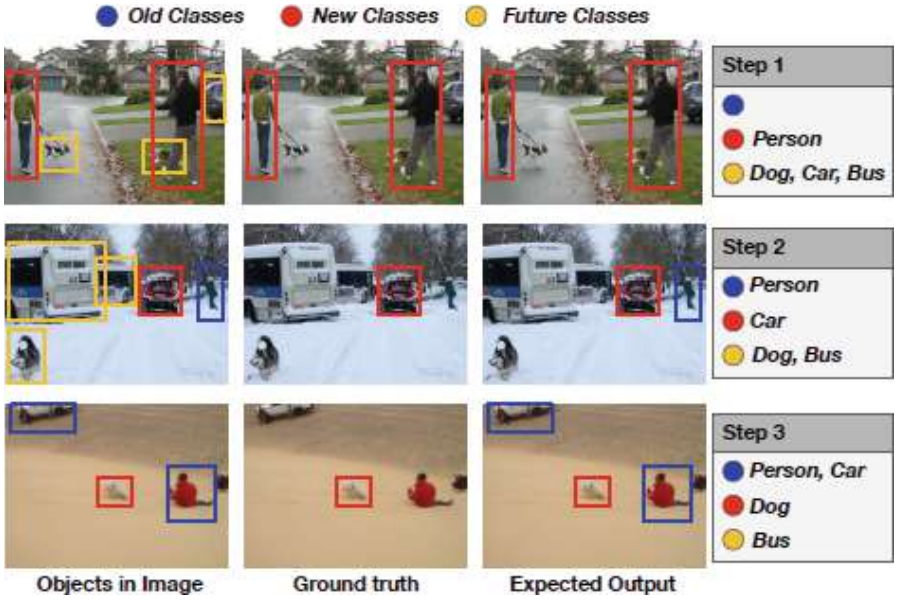


Figure 3.11: Illustrative example of an incremental object detection learning task. Each row represents a single step in incremental learning. According to the training protocol, a model is only presented with new classes in ground truth data, but expected to predict old and new classes alike. In every image, future classes can be included that have not been learned yet. From Cermelli et al. [CGFC22].

annotations only for instances of classes that are new in this learning step. However, training images typically contain instances of old and future classes as well.

In the standard formulation of knowledge distillation, this leads to the student model learning to associate these classes with the background, exacerbating the problem of catastrophic forgetting and making learning new classes more difficult. Figure 3.11 illustrates this issue for three training steps and a set of old, new, and future classes.

Full retraining, with a complete dataset containing all classes, does not have the problem of catastrophic forgetting but can quickly become prohibitively expense in terms of computational overhead, especially when frequently retraining the network. It might also not be possible to store the original dataset indefinitely, for example because of copyright issues. While it is be possible to avoid full retraining with a rehearsal method, this would still require retaining on a subset of all previous training sets.

Modelling Missing Annotations

Cermelli et al. [CGFC22] identify a particular problem with incremental learning in object detection that has not been addressed so far. In a standard object detection training pipeline, regions of interest (RoIs) that match a ground truth annotation (positive RoIs) are assigned the annotation's class label, and RoIs that do not match any annotation (negative RoIs) are assigned to the background.

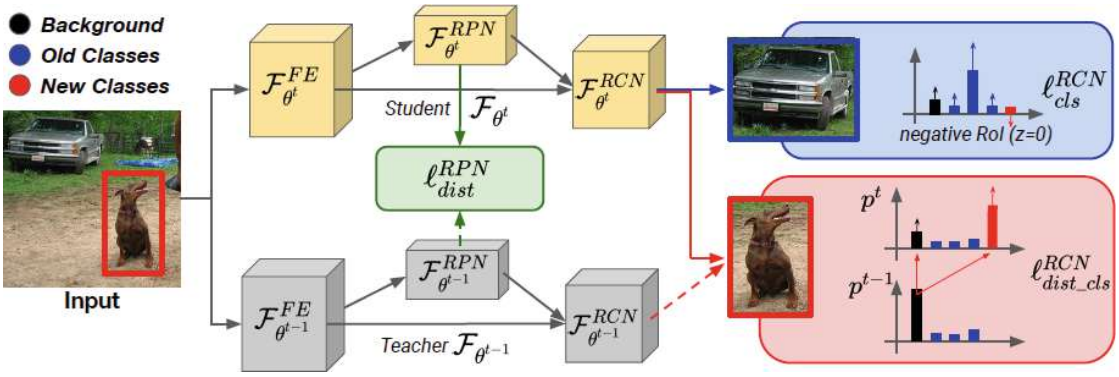


Figure 3.12: Overview of MMA in a knowledge distillation framework for object detection: the student and teacher models consist of feature extraction (FE), region proposal network (RPN), and fully convolutional network (FCN). Both models are input an image where only new classes are annotated. The student model associates unannotated objects (old class, in blue) with old classes or the background. For objects with annotations (new class, in red), the student maximizes the probability of the teacher's background belonging to the background or a new class. Image from Cermelli et al. [CGFC22]

When a negative RoI is associated with the background in incremental learning tasks, this causes two types of problems: first, if the RoI contains an old class, the model learns to predict this class as background, amplifying catastrophic forgetting. If, on the other hand, the RoI contains a future class, and it is predicted as background, this will make learning this class in following training steps more difficult.

MMA reformulates the classification and distillation loss to associate negative RoIs with the background or an old class, and positive RoIs with the background or a new class. In other words, it allows the student model to predict either an old class or the background in any regions not associated with an annotation of a new class, and adjusting the background probability of the teacher model to match the probability of having a new class or the background.

The effect of MMA on knowledge distillation loss is illustrated in Figure 3.12. With this formulation, experiments in [CGFC22] show that this method outperforms others on single-step and multi-step object detection tasks on the Pascal VOC dataset without reusing images from earlier training steps, and outperforms other baselines for the instance segmentation task on the Pascal SBD 2012 dataset.

Adapted Knowledge Distillation Loss

Describing the changes of MMA in more detail, we start with the training loss of Faster R-CNN, which consists of classification and regression terms for the RPN and classification head:

$$\ell_{faster} = \ell_{cls}^{RPN} + \ell_{reg}^{RPN} + \ell_{cls}^{RCN} + \ell_{reg}^{RCN}$$

MMA adapts the term ℓ_{cls}^{RCN} for unbiased classification loss by learning only new classes on positive RoIs, and avoids assigning negative RoIs to background by maximizing over the sum of background and old class probabilities. This handles missing annotations within the regular loss function. Knowledge distillation introduces two loss terms, in addition to the common term for Faster R-CNN ℓ_{faster}:

$$\ell = \ell_{faster} + \gamma_1 \ell_{dist}^{RCN} + \gamma_2 \ell_{dist}^{RPN}$$

These terms force the class scores and box coordinates of the student to closely follow the teacher's output and thus avoid forgetting on old classes. However, as mentioned earlier, new classes observed by the teacher in past training steps were assigned to the background, and thus make learning harder for the student network.

MMA changes ℓ_{dist}^{RCN} to allow the student to predict new classes or background on regions classified as background by the teacher, and follows the teacher prediction on old classes. For the second term, ℓ_{dist}^{RPN}, the student model follows the teacher only if its objectness score is lower than the teacher's. The idea being that a higher objectness score in the student likely means it has detected an instance of a new class.

For the task of instance segmentation, the loss function includes additional terms for distillation and classification loss:

$$\ell = \ell_{mask} + \gamma_1 \ell_{dist}^{RCN} + \gamma_2 \ell_{dist}^{RPN} + \gamma_3 \ell_{dist}^{MASK},\ \ell_{mask} = \ell_{faster} + \ell_{cls}^{MASK}$$

The distillation loss term ℓ_{dist}^{MASK} with hyper-parameter γ_3 computes the per-pixel binary cross-entropy loss between teacher and student model masks, but only for old classes. We refer the reader to [CGFC22] for a complete description of the computation of the loss function.

The hyper-parameters γ_1, γ_2, and γ_3, are set to 1, 0.1, and 1, respectively. In [CGFC22], γ_2 is decreased, and the learning rate increased when learning more classes at once. In this framework, both parameters are fixed for simplicity, even though the number of new categories may vary depending on cluster selection.

3.5 Novelty Detection and Class Discovery

Our framework draws its main inspiration for novelty detection and class discovery from two previously published works. From Nakajima [Nak20] we incorporate the idea of using 3D segments as the basis of novelty detection and feature extraction, and from Uhlemeyer et al. [URG22] we apply their method of category discovery with feature embedding and clustering. However, our approach is different from both in several key aspects.

With regards to [URG22], both our and their approach cluster unknown objects based on visual similarity, as defined by features extracted from a Convolutional Neural Network (CNN). In each case, clusters define new classes and the training data for incremental learning steps. According to [URG22], this is also the prevailing method for class discovery in other works. Figure 3.13 illustrates the general idea of class discovery based on feature extraction.

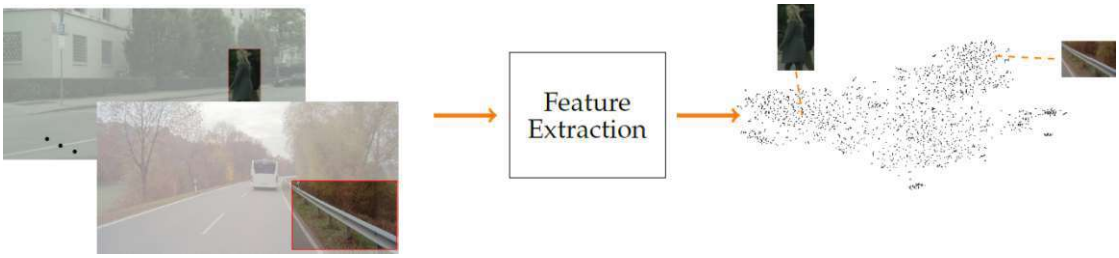


Figure 3.13: Feature extraction for category discovery, as proposed by Uhlemeyer et al. [URG22]. Segments from novelty detection are input to a CNN to generate feature vectors, which are embedded in a lower-dimensional space. Clustering groups similar segments to create novel categories. Image from Uhlemeyer et al. [URG22].

For clustering, we use constrained clustering, i.e., we add constraints to the cluster membership of individual samples, while the clustering in [URG22] is unconstrained. The intention being that different views of the same segment are supposed to remain in the same cluster, as they are already known to belong to the same object due to segment tracking in the global map.

Uhlemeyer et al. assess segmentation quality estimates on connected components of semantic segmentation, and use components with low estimates as candidates for clustering. In contrast, we use segments from depth segmentation with no associated semantic instances as candidates for clustering.

For feature extraction, [URG22] crop rectangular candidate regions from RGB images and feed them into an image classification CNN trained on ImageNet. We extract features directly by cropping candidate regions from a feature map using geometric-semantic segmentation masks. Our feature map is reused from instance segmentation, which is pretrained on ImageNet. In both cases, the features are taken from the final layer of the CNN before classification or object detection.

3.5.1 Overview

In our framework, novelty detection and category discovery work on labelled segments extracted from RGB-D images by the geometric-semantic segmentation of Voxblox++. That is, the boundaries of these segments are defined by geometrical features of the environment and not by the segmentation masks from object detection.

This has the positive side effect of novelty detection being less reliant on segmentation masks with highly accurate predictions. In addition, all of the processing in this stage is image-based and done on a per-frame basis, and does not involve any 3D data.

3.5.2 Scene Data Recording

Novelty detection and category discovery are performed in a semi-offline fashion. During each scene reconstruction session, all relevant data about global segments, including

RGB video frames and segmentation feature maps, is recorded continuously as they are published by the other nodes in the framework.

When a certain amount of data has been collected, the novelty detection node is signalled to stop data collection, find novel segments and perform category discovery, and output a new training set for re-training the instance segmentation model. In our evaluation, we signal the novelty detection node after sets of five scene reconstructions each.

However, other signal conditions could be used as well, for example the duration of recordings and the amount of data collected, or thresholding based on a metric that periodically measures data quality or category discovery performance with the data at hand.

3.5.3 Data Processing Pipeline

The main steps of data processing for novelty detection and category discovery in the framework are summarized as follows: data collection gathers all segment data and metadata that is relevant for category discovery. Novelty detection groups the data into known and unknown segments, and only uses the latter for category discovery.

Category discovery computes one or more low-dimensional embeddings for each segment, and groups these embeddings with constrained clustering. We determine an internal measure of the quality of each cluster and select the top N clusters as new categories. The final segment data from category discovery is then stored, to be consumed as a training dataset for incremental learning of the instance segmentation model.

It should be noted that, while most of the data processing is performed offline, other nodes in the system do not have to be suspended during category discovery. Instead, category discovery and incremental learning can be relegated to a background process until the updated segmentation model is ready to be incorporated into the system.

There is also some room to perform parts of segment data preprocessing online, at the expense of additional computational complexity during data collection. However, in this framework we have opted to include preprocessing in the offline stage to simplify data handling and to allow for easy testing and fine-tuning of novelty detection and category discovery.

3.5.4 Segment & Instance Tracking

As the first step of novelty detection, the system continuously collects a subset of the data that is published. This includes data that is processed by instance and geometric-semantic segmentation and the global segmentation map for each video frame.

Each global segment can appear in multiple video frames, and we want to process multiple observations of every segment for category discovery. Hence we refer to each observation as a segment view, to distinguish it from the global segment itself which is unique across views. In order to keep track of multiple observations for each segment, the system

processes the association between per-frame and global segment labels as well as segment merging, as published in the global segment map.

For the extraction of features for each segment view, the segmentation features published by instance segmentation, and the segment label map from depth segmentation are stored as well. Lastly, the original RGB images are collected for training data generation.

Segment data and metadata is stored in a hierarchical data structure that keeps track of segments over multiple frames and associates them with the per-frame data, i.e., RGB images, segment label maps, and segmentation feature maps. This data structure stores all information for a single scene. When recording multiple scenes, each has its own data set to keep persistent labels separate, as they are unique only within each scene.

3.5.5 Novelty Detection

In the next step, we group all segments into known and unknown objects for novelty detection, based on their association with an object instance. To this end, the novelty learning node retrieves the pairwise counts between segment and object instance labels, referred to as instance counts in Figure 3.6, from the global segmentation map. Known segments are those which are assigned to an object instance, i.e., their pairwise counts are strictly positive. All other segments are unknown and candidates for category discovery.

In this step we also filter segment views to increase the quality of the resulting training data set and the balance with respect to potential novel categories. We only include a maximum ratio, and a fixed total number n of segment views per segment, 15 in this case, to prevent objects that are frequently visible from dominating the training data set. That is, with the number of segment views v_i for segment i. we use only $max(1, min(v_i/r, n))$ of the views.

This subset is randomly sampled from all available views for each segment. More sophisticated methods for sample selection would also be possible, e.g. by taking the position of the camera into account, to capture each object from a variety of angles that is as large as possible.

3.5.6 Feature Extraction

After the selection of segment views has been determined, each segment view is assigned a feature vector based on the feature map from instance segmentation. The feature map is masked with the binary segmentation mask of the segment, for the video frame the segment view appears in. The segmentation mask is again derived from the segment label map.

Since the feature map has a low spatial resolution of 40×30, it is first up-scaled to the resolution of the segmentation mask, which equals that of the depth image, using bi-linear interpolation. The channels in the feature map are then averaged over the segment area, resulting in a single, high-dimensional feature vector per segment view.

If the segment view consists of two or more merged segments, we use the union of their areas to compute the feature vector. We hypothesize that the up-scaled low-resolution feature map in Mask R-CNN includes spatial context when extracting segments, similar to rectangular crops in [URG22].

This feature vector is stored alongside the other segment data so it is possible to keep track of which features belong to which global segment label and frame. For performance reasons, up-scaling and averaging segmentation features is implemented on the GPU, while all other computations in this stage are done on the CPU.

Computing average features over segments is done offline for simplicity, instead of in the novelty learning node when frames are received. Segment masks may be merged during recordings, e.g. for segments that are disjoint in the first frame and connected later on by a segment not visible initially. For this reason it is not sufficient to compute the feature vector of each segment as it is published.

To handle these cases in an online fashion, an incremental update scheme could be devised that updates average feature vectors using the segmentation maps and feature maps when segments are merged. However, this update scheme is beyond the scope of the current implementation.

Segment views are processed frame-by-frame, as caching up-scaled feature maps would require too much memory, and recomputing the interpolated feature maps for each view would be too computationally heavy. Likewise, features are computed per-scene since loading more than one scene data set would exceed available memory on the test system.

3.5.7 Feature Embeddings

At this point each feature is assigned an ID that is unique across multiple scenes, as global labels alone are not sufficient to distinguish features from different scenes. In this framework, feature IDs simply consist of the scene name and global segment label combined.

The feature vectors are not well suited for conventional clustering methods, due to the fact that distances between arbitrary pairs of vectors are likely to be very similar in high-dimensional spaces. This well-known observation is part of a series of problems when working with high-dimensional data or algorithms, and often subsumed under the notion of the ‘curse of dimensionality’ [Dom12]. Therefore, in order to compute suitable embeddings for clustering, we follow the same approach as [URG22].

We apply Principal Component Analysis (PCA) to reduce the dimensionality of feature vectors, and use t-distributed Stochastic Neighbor Embedding (t-SNE) with an Euclidean distance metric to compute 2D embeddings. This procedure, as used by Uhlemeyer et al., is taken from an earlier evaluation of feature extractors, distance metrics, and feature dimensions, being the best performing setup in comparison [ORF20]. The parameters used for both methods are listed in Table 3.2.

Parameter	Value
PCA Components	50
t-SNE Components	2
t-SNE Perplexity	30
t-SNE Learning Rate	200
t-SNE Early Exaggeration	12
t-SNE Max. Iterations	5000

Table 3.2: Parameters for segment feature embedding with PCA and t-SNE.

PCA transforms the feature vectors into a lower-dimensional space while minimizing the sum of squared errors of the reconstruction, and minimizes information loss from dimensionality reduction. Embedding with t-SNE is originally designed for the visualization of high-dimensional data in two or three dimensions. It tries to minimize the Kullback-Leibler divergence between joint probabilities of the high- and low-dimensional space.

Since the Kullback-Leibler divergence is not convex, the optimization process can end up in local minima, and the output of t-SNE depend on its initialization[10]. It can therefore be useful to start t-SNE with different initial values and pick the best result, however this is not done in our framework.

In the embedding step, we use features from all scenes combined to compute the principal components and the embeddings with t-SNE. This intends to allow the following clustering step to group objects from different scenes in one cluster, and thus provide an opportunity for category discovery to form more general novel categories across scenes.

3.5.8 Category Discovery

Following the approach by [URG22], a traditional clustering method is used to group the feature embeddings into novel categories. The clustering method of choice is Density-Based Spatial Clustering of Applications with Noise, or DBSCAN for short [EKX+96].

This method assumes clusters to form areas of high sample density, separated by low-density areas. As such, it can unambiguously detect noise and clusters that are non-convex, unlike for example k-Means.

DBSCAN classifies samples in the data set as core and non-core points or noise. Core points are those with a minimum number of neighbors n_{min} within a distance ϵ. Non-core points are within the distance threshold of core points but do not have a minimum number of neighbors, and intuitively represent the fringe of clusters.

The clustering method recursively groups core points within distance ϵ to from clusters, and assigns each a cluster label. Non-core points that are neighbors of core points are

[10] https://scikit-learn.org/stable/modules/manifold.html#t-sne

Parameter	Value
Distance Metric	Euclidean
Constraint Factor	0.125
DBSCAN ϵ	2.5
DBSCAN Neighbors	10
Cluster Min. Size	30
Cluster Min. Score	0
Max. Pseudo-Labels	20

Table 3.3: Parameters for DBSCAN clustering and must-link constraints approximation.

assigned to the respective cluster as well. Isolated points are labeled as noise and do not belong to any cluster.

DBSCAN has two essential parameters, the minimum number of neighbors n_{min} and the maximum neighbor distance ϵ. The maximum distance is the most critical parameter and needs to be set according to the data set. A value too small or too large will result in no clustering occurring or most samples ending up in a single cluster.

The number of neighbors controls the tolerance of clustering towards noise. For larger data sets and noisy data it can be beneficial to increase this value. Table 3.3 lists the parameters used in constrained clustering.

When clustering samples with multiple segment views for the same segment, we want these features to end up in the same cluster because they must belong to the same object. Constrained clustering can model this information with must-link constraints, i.e., additional constraints that two or more samples must always be assigned to the same cluster.

For DBSCAN, we can use a precomputed distance matrix and reduce the distance between features from the same segment to approximate must-link constraints. Instead of setting the distances for these feature pairs to zero, we relax the must-link constraints as a way to balance similarity based on geometric segmentation and visual features.

This may lead to assigning segment views from the same segment to different clusters, although geometric segmentation may not be correct to begin with, e.g. when two adjacent objects are detected as one segment.

The results of clustering are one cluster label per embedding and thus per segment view, and the labelling of samples as core or non-core points. The results of DBSCAN are deterministic, but the assignment of non-core points to clusters can vary depending on the ordering of data[11]. In the following steps, only core points are considered as belonging to clusters. Non-core points and their associated segment views are treated as noise and discarded.

[11] https://scikit-learn.org/stable/modules/clustering.html#dbscan

3.5.9 Cluster Selection

As the clustering step may return a large number of clusters, it is useful to limit the total number of novel categories in each incremental learning step. For this purpose, the clustering quality is measured, and only the top k clusters are used as training data. We estimate cluster quality based on Silhouette Coefficients [Rou87] as an internal measure of cluster cohesion and separation, as well as on cluster sizes and numbers of segments.

The Silhouette Coefficient for a point i is defined using the mean intra-cluster distance a_i and the mean nearest-cluster distance b_i, as $c_i = (b_i - a_i)/max(a_i, b_i)$. The Silhouette Score for the clustering is taken to be the average of all coefficients. Similarly, we can define scores for single clusters as the average of coefficients of its samples.

Silhouette Coefficients are bounded by -1 and 1, with large negative values indicating incorrect clustering and 1 indicating highly dense clusters. Values around zero are typical for overlapping clusters. Well-separated and dense clusters produce higher scores[12].

In addition to cluster cohesion and separation, we also want to take cluster sizes and the number of different segments within each cluster into account. Intuitively, each cluster should not only be dense and well-separated, but generalize to a larger set of segments to increase its usefulness as a new category.

In order to capture this idea, the number of all segment views for each unique segment in each cluster is counted to compute a weight for its Silhouette Score. Given a set of n unique segments l_i in a cluster c, and their respective number of segment views k_i, the cluster weight w_c is computed as

$$w_c = log(\prod_{i=1}^{n} k_i) = \sum_{i=1}^{n} log(k_i)$$

With the Silhouette Score s_c of cluster c, its final score f_c is defined as $f_c = w_c * s_c^2 * sign(s_c)$. Here, the Silhouette Score is squared to favor smaller clusters with higher positive scores over larger clusters with lower positive scores. The *sign* function preserves the sign of s_c such that the final score is negative if s_c is negative. Sorting clusters by their score and taking the top k then gives the final result of category discovery.

3.5.10 Training Data Preparation

In preparation for the incremental learning step, a new training data set is written based on the results from category discovery. First, we group all segment views associated with sample points by scene, based on their unique IDs. Since the scene data sets are too large to fit into memory at once, the segment views are processed scene-by-scene.

For each scene, the information about associated segment views and the video frames they appear in are collected from the scene data set. The video frames are then written as image data for the training set. This includes the segmentation mask derived from the

[12] https://scikit-learn.org/stable/modules/clustering.html#silhouette-coefficient

segment label map, bounding box coordinates computed from the segmentation mask, and a pseudo-label based on the cluster index as a class label.

For each video frame, we enumerate the segments that appear in this frame, and write out all segments of each new category as an instance segmentation annotation. In this step, it is important to include all segment views from selected segments, not just the segment views used for clustering. Otherwise segmentation masks would be incomplete, violating annotation specificity as one of the open-world object detection benchmarking principles defined by Zhao et al. [ZLS+22], affecting re-training. Training data must be fully annotated with respect to new categories, to allow new categories to be learned without the ambiguity of missing annotations, as outlined by Cermelli et al. [CGFC22].

CHAPTER 4

Evaluation and Results

The testing and evaluation of this framework requires a public dataset of RGB-D recordings with accompanying ground truth 3D reconstructions and instance-aware semantic segmentations. This dataset has to fulfill several criteria in order to be usable for evaluation and to compare our method to other publications.

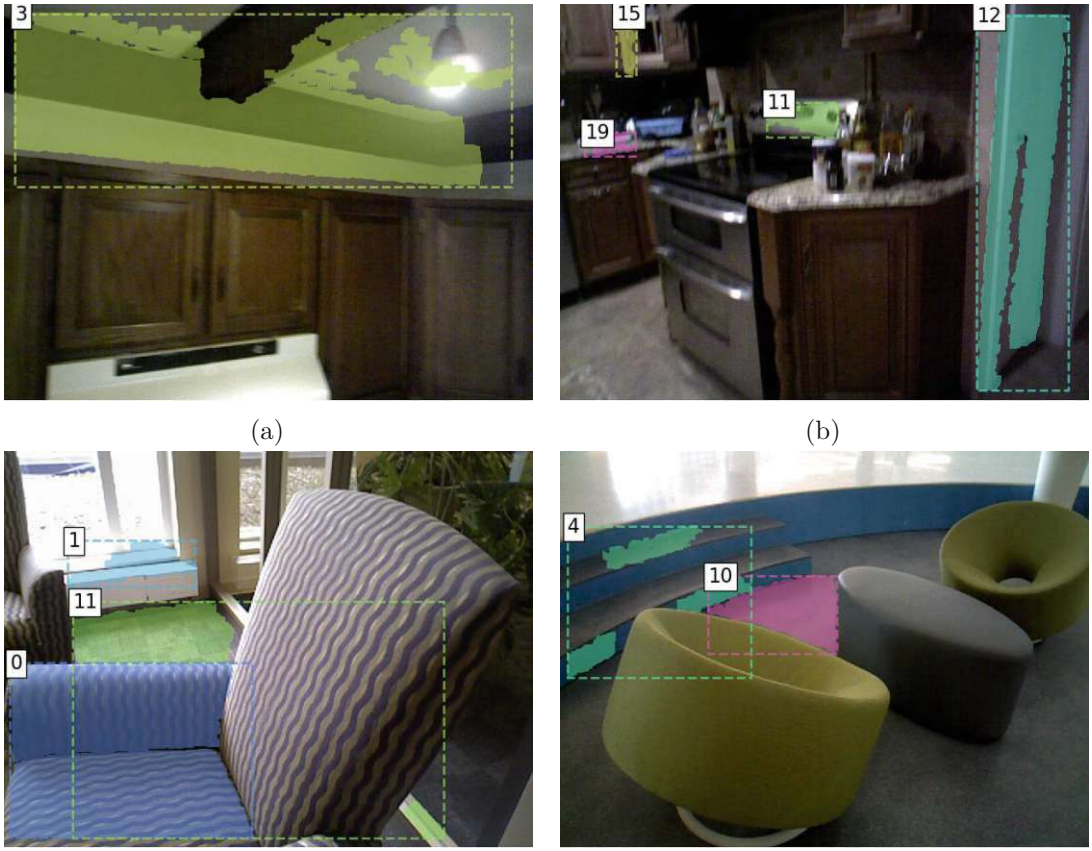
4.1 Choice of Dataset

First of all, the dataset must contain full RGB-D video sequences for the mapping to reconstruct each scene with sufficient detail. The resolution of the RGB-D frames must be high enough to support object detection and instance segmentation, and the frame rate should be stable enough to guarantee an approximately constant frame rate within the framework.

We do not consider synthetic datasets for training or evaluation but rather use actual recordings of real-world scenes. As Dai et al. [DCS+17] report in their experiments, training with a synthetic dataset provides limited knowledge transfer for future tasks. However, synthetic datasets may be a good option to complement other data for training or evaluation [PJY+21].

Since camera localization is beyond the scope of the implementation of this framework, the dataset has to provide this information too. This can also be useful for comparison with other works, since metric-semantic mapping can be evaluated independent of a specific localization method.

In order to measure the effect of incremental learning on object-centric mapping, a ground truth mesh or point cloud with instance-aware semantic annotations is required. These annotations have to be dense, i.e., cover the whole reconstruction, and include not only objects but also structures such as walls, floors, and ceilings. Since novelty detection



(a) (b)

(c) (d)

Figure 4.1: Examples of wall, floor, and ceiling segments erroneously picked up by novelty detection, and included in novel categories. These segments were not correctly labeled by instance segmentation, and thus considered novel objects in the framework.

picks up objects and structures, it is necessary to have annotations for both to accurately measure the results of category discovery.

These panoptic annotations are also necessary when using the dataset for training of the initial instance segmentation model. This framework is intended to perform novelty detection based on a model that is pretrained on a set of common classes including structure classes. Without these structure classes in the training set, novelty detection picks up a large number of wall and floor segments for category detection.

Even with structure classes included in initial training, segments that are missed by instance segmentation and remain unlabeled are picked up by novelty detection and can be included in novel categories. Figure 4.1 gives several examples of wall, floor, and ceiling segments taken from category discovery results.

For evaluation, we favor a dataset with a small set of more general classes for semantic

annotations, instead of an open-dictionary approach, where the set of classes is fundamentally unlimited. Finally, the dataset must include a sufficient number of scene recordings to provide enough sequences for initial training as well as validation and test setups.

4.1.1 Instance Segmentation Dataset - SceneNN2016

As the dataset for training, testing, and evaluation, we choose SceneNN[1], initially created by Hua et al. [HPN+16], and updated in [HTY18]. The original SceneNN, also referred to as SceneNN2016, comprises a real-world annotated dataset of 100 scenes from different categories, including office scenes, living rooms, kitchens, bedrooms, and others. The scenes are named with three-digit numbers, which we also use to refer to individual scenes in the evaluation.

Each scene features per-vertex annotations in a ground truth mesh generated from RGB-D videos. The RGB-D videos from which scenes were reconstructed are included as well, with a resolution of 640×480 for the RGB and depth images, and a frame rate of approximately 30 fps.

Each image is timestamped so that it is possible to accurately synchronize RGB and depth data. The estimated camera positions in each recording are provided as 4×4 rotation matrices, with respect to a global coordinate system, for each video frame.

All objects in each scene are segmented using per-vertex instance IDs, and each instance comes with additional information stored in one XML file per scene. This information comes in the form of instance colors and oriented bounding boxes. Most objects are also labeled with an open-dictionary text, however this is not the case for all instances, in particular for structures such as walls and floors.

The instance segmentation in the work of Hua et al. [HPN+16] is performed with their own annotation tool called 'sese'. This tool includes a feature to render the segmented ground truth meshes using the estimated camera positions, resulting in per-frame 2D instance segmentations with a unique color for each object instance. These colors can then be used to map segments in the renderings to the additional instance information from the XML file.

4.1.2 Semantic Segmentation Dataset - SceneNN2018

Hua et al. [HTY18] re-annotate 76 scenes of the original SceneNN dataset for use in their semantic segmentation method. The semantic labels span 40 classes matching NYU-D dataset annotations, including objects, structures, and super-categories such as 'furniture', 'structure', and 'prop'. The complete list of classes is available online[2]. We refer to this re-annotated dataset as SceneNN2018.

While SceneNN2016 creates ground truth meshes from scene reconstructions, this dataset only contains per-vertex data in the form of annotated point clouds. That is, there is no

[1] https://hkust-vgd.github.io/scenenn/

[2] https://hkust-vgd.ust.hk/scenenn/home/cvpr18/data/nyu_color.xml

connectivity information in ground truth geometry. Additionally, scenes are cropped at a height of approximately 2 m and spurious segments outside the main area are removed, leading to missing class labels for parts of the original scenes, in particular ceilings.

Since the text labels in SceneNN2016 are incomplete and sometimes inconsistent, we transfer class labels from SceneNN2018 to create a combined dataset with instance-aware semantic annotations in 2D and 3D. This combined dataset, consisting of 76 scenes from SceneNN2018 with class and instance labels, is then used to create training data for fine-tuning the initial instance segmentation model, and to test and evaluate the framework.

4.1.3 Dataset Issues

Despite its usefulness, there are several issues with SceneNN, some of which are specific to this dataset, and others are more general in nature. Regarding the text labels in SceneNN2016, these are not always applied consistently or correctly and, in some cases, contain spelling and labelling errors.

Common examples for inconsistent labels include ‘window’, ‘blinds’, and ‘curtains.’ Examples of incorrect labels include ‘pot’ vs. ‘stove’, or a large plastic bottle in scene 276 labelled as ‘kettle.’ The latter case suggests that visual confirmation from real video footage could prove helpful when labelling 3D meshes, as the distinction between these object classes is arguably more apparent in color images than in 3D reconstructions. Some classes in SceneNN2018 are not easily distinguishable, even for humans, or the classification of objects may depend on their usage. Examples include ‘desk’ vs. ‘table’ vs ‘night stand’ and ‘shelf’ vs. ‘bookshelf.’

A similar problem arises when trying to match open-dictionary text labels from SceneNN2016 with the 40 classes used in SceneNN2018. Here it is not always obvious, or ambiguous, which class an instance with a certain label belongs to without looking at the object in question. For example, a ‘TV stand’ might best fit into the ‘cabinet’ or ‘furniture’ class, depending on its specifics.

It turns out that instance colors in the annotation files do not always match the colors in the ground truth mesh. These are required to associate 2D instance segmentations in video frames with other instance metadata.

More generally, real-world datasets typically feature long-tail distributions of classes, that is, most instances belong to a small number of classes, while all other classes appear in only a few instances. This can create problems when learning and evaluating object detection and segmentation tasks for rare classes, as the available training data is severely limited [YPRL22]. In the dataset generation step, we try to address each of these issues.

wall	curtain	table
floor	shelves	desk
ceiling	bookshelf	books
door	cabinet	paper
window	chair	lamp

Table 4.1: List of classes from SceneNN used for the initial training set, structure classes in italics.

4.2 Dataset Generation

For the initial training set, we select 15 common classes that the instance segmentation should be fine-tuned to. The selection is highlighted in Table 4.1, and includes objects and structures that can be found in most indoor scenes.

To create a training set for the initial object detection and segmentation model, it is advantageous to include a sufficient number of images containing approximately equally distributed class occurrences. Each image must be annotated with bounding boxes, class labels, and segmentation masks for each instance of the 15 included base classes. This follows the principle of annotation specificity for open-world object detection, as outlined by Zhao et al. [ZLS+22].

The main steps in combining both datasets for our dataset generation are therefore as follows: first, the ground truth vertex data of SceneNN2016 and SceneNN2018 are registered for each scene, and class labels transferred per-vertex to determine the most likely class for each instance.

By rendering the ground truth meshes of SceneNN2016 with instance colors for all camera positions, a 2D instance label map is generated for each frame. From these label maps, in combination with the class annotations, per-scene statistics are collected regarding the occurrences of classes and instances in each scene and video frame.

Finally a number of RGB images are selected from the scene recordings, the segmentation masks, bounding boxes, and class labels of all instances in these images belonging to the 15 base classes are computed, and all information and meta-data is stored on disk.

The subsequent evaluation of the framework requires the raw RGB-D sensor data, including camera localization, and the ground truth meshes with combined instance and semantic labels. For this task, raw sensor data from scene recordings are converted to the ROS-specific bag file format, and the same label transfer method as in the initial dataset generation is used. The complete set of scenes is also split into training, validation, and test data, based on the scene categories provided by [HPN+16], which are also available online[3].

[3] https://hkust-vgd.ust.hk/scenenn/main/category.csv

4.2.1 Label Transfer

In preparation for label transfer and dataset generation, timestamped sensor images are extracted from source files and stored in the ROS bag file format as RGB and depth image topics.

ROS bag files store streams of ROS messages for one or more topics, thus allowing to replay data is if it were published directly by a ROS node or set of nodes. For camera localization, the transformation matrices of each frame are converted to the ROS-internal representation in the form of positions and quaternion rotations, and included in the bag files.

The label transfer from SceneNN2018 point clouds to SceneNN2016 meshes consists of four main steps: point cloud registration to align the ground truth data sets, matching individual vertices to points with a radius-limited nearest neighbor search, determining the most likely class for each instance based on the class labels of matched points, and assigning final class labels by incorporating information from original text labels.

Since label transfer does not require mesh connectivity information, only the vertex coordinates and annotations are loaded from the SceneNN2016 meshes, resulting in corresponding point clouds for further processing. Registration between the original SceneNN2016 point clouds and the re-annotated data from SceneNN2018 is necessary because the latter reorients points relative to the coordinate origin and the ground plane.

Alignment is performed for each scene, and divided into two steps. First, both point clouds are downsampled, FPFH features are computed for each, and a coarse registration is estimated. This registration is then refined on the full point clouds with Iterative Closest Point (ICP). The result is a rigid transformation that aligns both point clouds.

Once aligned, each vertex in the SceneNN2016 data is matched with points from SceneNN2018 using nearest neighbor search. Since the latter is cropped, not every vertex has a match. In order to avoid matching vertices to far-away, unrelated points, nearest neighbor search is limited by a radius approximately equal to the sampling resolution of the data. In this formulation, multiple vertices may match to the same point, but this is unlikely to happen and generally not an issue for the label transfer method.

Class labels are transferred for each instance, that is, all vertices with the same label are grouped into disjoint sets. Then, the class labels of matching points are counted for each set, and the class with the highest count is taken as the preliminary class label for this instance. Points without a nearest neighbor are assigned the unknown label in this step.

For registration and nearest neighbor search we use the implementation provided by Open3D [ZPK18]. In the label transfer step we also check if the point colors of each instance match those stored in the scene annotations file, and correct entries if necessary.

Some instances in the ground truth meshes are not listed in the annotations. These instances appear to represent smaller spurious segments caused by noisy reconstructions, and are ignored for dataset generation and evaluation.

Set	Category	Scenes
Training	Work Place & Bedroom	082, 074, 073, 080, 045, 032, 025, 078, 084, 052, 076, 041, 047, 231, 209, 086, 038, 057, 036, 201, 054, 069, 062, 252, 016, 021, 227
Test 1	Kitchen & Other	527, 294, 286, 621, 522
Test 2	Kitchen & Other	260, 255, 276, 270, 609
Test 3	Living Room & Bed Room	272, 087, 223, 093, 005
Test 4	Bed Room	249, 240, 234, 265, 237

Table 4.2: Sets of scenes from SceneNN used for initial training, and for evaluation (Test 1 to 4). Also listed are the scene categories each set falls into.

As the point clouds in SceneNN2018 are cropped, the annotations are incomplete with respect to the ground truth meshes, and it is necessary to check preliminary class labels against the original text labels. Text labels are normalized by first removing all spelling variants, e.g. underscores vs. spaces, digits, and extra white-space. Then, all text labels are aggregated, sorted, and duplicates removed to create a mapping to a canonical set of labels. This mapping removes all spelling errors, chooses one of each set of synonyms, and uses only one of plural and singular forms.

If the preliminary class label is unknown, and the text label matches one of the 40 class names from the SceneNN2018 annotations, we use the corresponding class label as the final label. Otherwise, the preliminary class label is assigned to an instance.

This is primarily done to automatically label segments of ceilings, which are cropped from SceneNN2018, but also serves as a confirmation that class labels are transferred correctly. If both the preliminary class label and the canonical text label map to a known class, but are different, a warning is issued. However, this only affects instances were class labels are ambiguous, as mentioned above.

4.2.2 Initial Training Set

For the initial training set, all instances in the training set of scenes belonging to the base classes are selected. Instances for which the area values in the scene annotation is below a class-specific threshold are excluded.

This threshold is set to 5000 for object classes with larger instances (i.e., excluding ‘books’, ‘paper’, and ‘lamp’), based on typical instance values, and zero for all others. In this way, random small segments from reconstruction errors are excluded to prevent erroneous annotations that could degrade model performance during fine-tuning.

For each remaining instance, a set of video frames is selected, and all selected frames are

annotated and added to the training set. In order to create a more balanced set, we only select a subset of frames per instance.

Specifically, if the set of frames containing instance i is of size n_i, only n_i/r or n_{max}, whichever is lower, are drawn randomly from this set. For each instance, at least 1 frame is used. In this case, n_{max} is set to 50 and r to 6.

For each selected video frame, annotations for all instances of base classes are included in the training metadata. Base classes are split into objects and structures. For each object instance, a separate annotation, containing bounding box, class label, and segmentation mask, is added.

Structure instances of the same class and within the same frame are merged, and one annotation is added for the combined bounding box and segmentation mask. This is done to approximate a panoptic segmentation approach within the instance segmentation model.

In both cases, annotations with (combined) segmentation masks smaller than a predefined size of 500 pixels are discarded. The size limit is derived from the COCO dataset, which groups objects into small ($< 32^2$), medium ($< 64^2$), and large ($> 96^2$) segments for evaluation [LMB+14].

For the training set, we extract images and annotations from 27 scenes, as listed in Table 4.2. 23 of these scenes are from the 'work place' category, and 4 from the 'bedroom' category. This results in a dataset of 1350 images with 6772 annotations in total. Of these, 20 % are used for testing, and the rest for training, resulting in a 1080/270 split.

4.2.3 Training & Evaluation Datasets

For validation and testing, the scenes not used for initial training are split into four groups related to the available scene categories. Since the training set mostly contains scenes from the 'work place' category, the idea is to measure how the framework performs in different domains, namely scenes from different categories, including kitchens, living rooms, lounges, and study spaces.

Each of the four groups contains two sets of scenes, one for development and validation, i.e., hyper-parameter tuning, and the other for evaluation. Each set contains five scenes from the respective categories.

Some scenes remain unused as they are not assigned to a specific category by the SceneNN dataset. Following standard practice, evaluation on the test sets is only performed once the framework development and hyper-parameter tuning is complete.

The evaluation process compares the results of scene reconstruction with ground truth meshes from the dataset generation step, containing combined semantic and instance annotations. This means the evaluation, at least to some degree, relies on accurate reconstructions, proper alignment between ground truth and reconstruction, and on the correct labeling of segments and instances. This is not self-evident, as the reconstruction

methods and the data input, i.e., the subset of processed video frames, differ between ground truth and predicted data.

As mentioned earlier, the dataset may contain mislabeled ground truth segments which can affect final results. However, since our main focus is on the effect of incremental learning on object detection performance, and not on the performance of object detection itself, this is less of an issue in our evaluation. We leave investigations on the influence of reconstruction errors and ground truth data quality on metric-semantic mapping to future research.

The framework does not make a distinction between background and unknown objects. This is intentional as the system is supposed to detect objects and structures, similar to panoptic segmentation. Hence there is no background class in the dataset or instance segmentation. Instead, all segments in the ground truth meshes are labeled as unknown or one of the base classes, and all results are assigned a base class label or pseudo-label.

4.3 Evaluation Protocol

In order to achieve reproducible results, the evaluation of this framework does not only define the dataset that is used, but also the metrics and sequence of steps required to perform each measurement. As [ZLS+22] phrase it, evaluation metrics are intended to "quantify the performance of a model targeted for a specific task." For the evaluation of our work, the performance regarding the following research questions are of primary interest:

- How does unsupervised incremental learning affect object-centric mapping performance?
- How does novelty detection perform?
- How does category discovery perform?

Since there is, to the best of our knowledge, no previous work on unsupervised incremental learning in object-centric mapping, the evaluation protocol takes inspiration from the evaluation of open-world object detection tasks, in particular [URG22] and [ZLS+22].

Class-incremental learning is evaluated in two main steps for every scene category set. First, each scene in the set is reconstructed with the initial segmentation model, and the segmentation performance is measured. Then, the initial model is retrained with novel categories extracted from the recordings, and used to reconstruct the same scenes. Segmentation performance is measured again, and changes to the first measurement are reported.

4.3.1 2D Instance Segmentation Metric

For the evaluation of instance segmentation models on base classes before and after retraining, we use mean Average Precision (mAP) as the primary metric. The mAP metric is averaged over all classes and instances, as commonly used in object detection tasks, and outlined for example in the COCO dataset [LMB+14] evaluation guidelines[4].

In this context, precision is based on the ratio of properly detected object instances, or true positives TP, and all detections, i.e., true positives and false positives FP. This precision value is calculated per class and averaged over all classes, excluding the unknown class label.

In order to determine if a detection is correct, each detection is assigned a confidence score, and a threshold is set for this score above which the detection is considered a true positive. For object detection and instance segmentation tasks such as for the COCO dataset, the Intersection over Union (IoU) is commonly used. In the general case of two sets, it can be defined as the ratio of the cardinality of their intersection and the cardinality of their union.

The average precision can be calculated for a single IoU threshold, or for a range of values, depending on the evaluation task. COCO guidelines use the latter approach, which is also used in the evaluation of Mask R-CNN Benchmark.

In this evaluation, mAP is measured for multiple IoU scores, ranging from 0.5 to 0.95 at 0.05 steps, per instance. Calculating the mAP for different IoU values favors models with better localization [LMB+14]. For 2D object detection, the IoU sets are typically rectangular bounding boxes, and for instance segmentation, pixel-wise binary segmentation masks.

4.3.2 3D Instance Segmentation Metric

In the evaluation of 3D instance segmentation in this framework, the IoU is based on sets of points from point clouds, representing volume elements of objects. An example of a 3D instance segmentation method that performs its evaluation in 2D is the work of McCormac et al. [MHDL16]. Here, the segmented mesh is projected into the current frame, and evaluated against single-frame segmentation with a CNN to measure the performance increase.

As Pham et al. [PHNY19] note, evaluation in 2D, by projecting of 3D labels to video frames, faces the problem of 2D images not covering the whole scene and ambiguities in 2D-3D projections. For these reasons, and because our focus lies on 3D instance segmentation, we choose subsets of point clouds as the basis of IoUcalculations.

Unsupervised incremental learning poses two main challenges regarding novel categories: the object classes these categories represent are not known in advance, and they are likely to not fall into the same set of classes with which the ground truth is labeled.

[4] https://cocodataset.org/#detection-eval

We approach this problem by calculating a confusion matrix between pseudo-labels and ground truth classes, and assigning pseudo-labels to the most prominently detected classes.

To compute the confusion matrix, the first important step requires matching instances between reconstructions and ground truth. For this step, we use the same approach as for label transfer in dataset generation to assign a ground truth class and instance label to each point in the reconstruction.

This means the ground truth and reconstruction point cloud are registered to find an accurate alignment, and radius-limited nearest neighbor search is used to match each point in the reconstruction with at most one point in ground truth. Then, the ground truth instance labels are counted for each instance in the reconstruction, and each pair of instances with the greatest overlap is matched.

In this instance matching, points in the reconstruction without matching points in ground truth are ignored, following the approach by Wu et al. [WWT+21b]. Using an IoU score of over 50 % to detect a match ensures that this matching is unique [KHG+18].

Based on these instance pairs, it is possible to compute the confusion matrix for each scene and over all scenes of every category group. We refrain from using mAP as an evaluation metric for 3D instance segmentation as there are not enough instances of all classes in the test scenes to compute a valid score.

4.3.3 Clustering Evaluation

For the category discovery module, visualizations of the feature embeddings and clusters are provided to assess its performance. These are useful for a qualitative evaluation in particular of cluster separation, cohesion, and overlap.

Visualizations are included for selected clusters, i.e., those used as new categories in incremental learning, and other, unused clusters. Additionally, individual instances of detected pseudo-labels are highlighted to provide a more detailed picture of the results.

4.4 Results and Discussion

All tests and measurements are performed on an Ubuntu 20.04 instance running on Windows 10 WSL (Windows Subsystem for Linux), on a dedicated workstation with an Intel Core i9-10900X CPU clocked at 3.70 GHz and 64 GB of RAM. Instance segmentation training and inference, and a small part of category discovery data preprocessing, use an Nvidia Geforce RTX 3080 GPU (Ampere architecture) with 24 GB of RAM.

4.4.1 Hyper-Parameter Tuning

The validation scene sets are used to tune hyper-parameters and assess the correctness of individual modules. This includes different approaches to category detection, in particular

different clustering methods and cluster selection mechanisms. Only after completing these steps is the whole framework evaluated on the test sets to produce the final results.

Different values for several hyper-parameters are tried, with the aim of improving category detection results and the size of the retraining sets. With a specific set of segments of feature maps gathered during a scene reconstruction session, the results of category detection are influenced only be the feature embedding method, clustering, and cluster selection.

Parameters for feature embedding are mostly the same as in [ORF20], as they are already the best-performing in their evaluation. Changes in these parameters also do not show noticeable differences in results, but increasing the number of steps for t-SNE does improve the KL-divergence slightly.

The size of the retraining set is primarily influenced by the number of segment views, assuming that clusters are approximately equal in size, as might be expected when segment views come from a multitude of objects and the clustering is valid. When most samples are lumped into a few clusters or most clusters are below the size threshold, this will skew the size of the retraining set.

As for clustering, DBSCAN is used in the validation step with and without approximated must-link constraints in the form of a modified precomputed distance matrix. For the multiplicative factor of distances between same-segment samples, several values are tried, ranging from 1/8 to 1/2.

For the number of samples typically collected in our scene recordings, the more general clustering method OPTICS proves to be too slow when using a precomputed distance matrix and larger minimum sample distances, at least in the provided open-source implementation[5].

In this fine-tuning stage, all assessments of cluster results are done with visual inspections only, mostly due to the lack of an objective measurement of clustering quality, and because testing the whole framework with different sets of hyper-parameters is beyond the scope of this thesis. Therefore, larger and more rigorous experiments to study the effects of hyper-parameters on our framework are left for future research.

4.4.2 Initial Training

The first part of the framework to evaluate is the initial training of the instance segmentation model. It uses a backbone pretrained on ImageNet as a feature extractor, and is fine-tuned on an initial dataset generated from a subset of SceneNN data containing 15 base classes.

This means the initial model is fine-tuned on domain-specific data, and the evaluation measures how well the fine-tuned model detects the base classes. We use mAP as the

[5] https://scikit-learn.org/stable/modules/clustering.html#optics

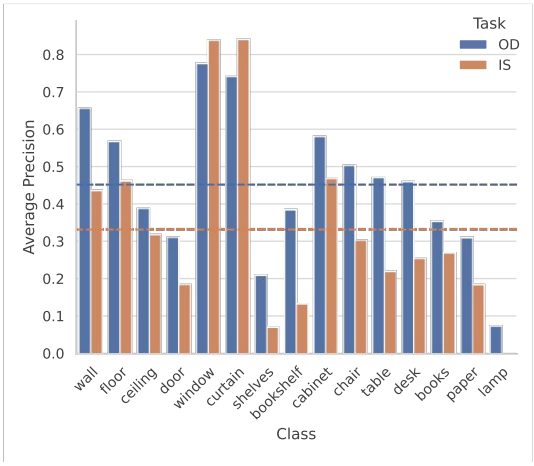


Figure 4.2: Object detection (OD) and instance segmentation (IS) performance of the Mask R-CNN model, fine-tuned and tested on the initial training dataset. Dashed bars represent mAP values over all classes for each task.

primary evaluation metric, and report per-class and overall Average Precision for the object detection and instance segmentation tasks.

Both tasks are evaluated on the test set of the initial training data. These results also serve as the baseline for measuring the effects of incremental learning on the detection and segmentation performance with regards to old classes. Figure 4.2 summarizes the evaluation of the initial model after fine-tuning.

As can be seen, precision is highest for common classes, such as ‘window’, ‘curtain’, ‘wall’, or ‘cabinet.’ Precision is generally higher for the object detection task, with the exception of the ‘window’ and ‘curtain’ classes. The instance segmentation task shows particularly low scores for the classes of ‘shelves’ and ‘bookshelf’, and also for ‘chair’, ‘table’, and ‘desk’.

The lower instance segmentation scores for certain classes might be related to additional scene clutter around these objects that makes it difficult to segment them properly. This effect could be exacerbated by the low spatial resolution of the segmentation network feature map, however a closer investigation would be necessary to confirm this hypothesis.

The ‘lamp’ class has the lowest score overall, presumably related to the low number of occurrences in the training set. With lamps, there is also the problem that some of its instances are turned on during scene recordings, leading to overexposure by the emitted light and making proper segmentation impossible. This could explain why the instance segmentation score is zero for this class, while the object detection score is slightly better.

Training Data

Some observations can also be made on the generated training data, of which Figure 4.3 shows several examples. As mentioned earlier, training data annotations, i.e., segmentation masks and bounding boxes, are derived from rendering ground truth meshes using estimated camera positions.

These camera positions are not always accurate, and can lead to misalignment between segmentation masks and the actual image content. This is noticeable in several frames, and can arguably have a negative effect on model performance when trained with inaccurate data.

Likewise, but for different reasons, boundaries between instances in renderings of ground truth meshes can appear jagged when close to each other in the scene. This might be related to the rendering method used, and in particular to 'z-fighting', where objects that occupy the same screen space and are approximately at the same distance to the camera have an arbitrary depth order, based on the precision of the z-buffer. However, we did not investigate this issue further, as its effects on fine-tuning are presumably small.

The number of samples for certain classes in the training set is quite small. This is due to the nature of most real-world datasets following a long-tail class distribution with a small number of frequent classes and a large number of infrequent ones.

Because some scene recordings are particularly short, randomly selected images from these scenes can be quite similar, reducing the quality of training data. While it is possible to counterbalance the effects of image and class distributions in the dataset, we use it as-is since the most important classes are represented with sufficient frequency.

4.4.3 Object-Centric Mapping

With the instance segmentation model in place, it is possible to test the framework on all scene sets. Each scene recording generates a 3D reconstruction with per-point annotations for class, instance, and segment labels. Figure 4.4 shows annotated visualizations for four different scenes as reconstructed by our framework.

As can be seen, the reconstructions are generally detailed and accurate enough to capture many typical household items, ranging from cups to bowls, bottles, boxes, chairs, tables, and other furniture. Surfaces are considerably more noisy than the ground truth meshes from SceneNN.

This is in part due to low frame rates of approximately 1-2 fps during online reconstruction. Low frame rates lead to less data being integrated into the global map and thus less corrections for noisy depth images.

However, since the focus of the framework is on robotic applications and not on visually pleasing 3D reconstructions, this is less of an issue in itself as long as the global map is accurate enough as a foundation for higher-level tasks. It should also be noted that the ground truth dataset uses a 3D reconstruction method with global optimization that

Figure 4.3: Examples of initial training set images with annotations, as shown in the center column. Each annotation consists of a segmentation mask, bounding box, and class label. For visualization purposes, masks and bounding boxes are colored with random colors. The original RGB camera frames (left column) and 2D instance segmentations (right) are provided for reference. Instance segmentations are rendered from ground truth meshes with a unique, random color per instance.

does not run at interactive frame rates [CZK15], and as such can not be used directly to replace the online reconstruction method in our framework.

Judging from the visualizations, segmentation performance is generally at the level expected from the segmentation model mAP results. In some cases, objects are over-segmented, for example the chair to the right in scene 294. In other cases, segments from two distinct objects are merged, as in scene 234 with the cabinet and floor to the right. Presumably, more accurate, per-frame instance segmentation and higher frame rates should be able to improve these segmentation errors.

4.4.4 3D Instance Segmentation

To quantify the performance of instance-aware semantic segmentation in 3D reconstructions, we compute the confusion matrices for each scene, based on the IoU between reconstructed and ground truth instances. The confusion matrices are summarized for each set of scenes in Figure 4.6.

Confusion matrices for each set show generally good results for structure classes, i.e., walls and floors, as indicated by a large number of matching prediction and ground truth instances. The number of correctly detected ceiling instances is lower, presumably because of the low number of correctly annotated ceiling segments in the dataset.

As far as object classes are concerned, the best results are achieved for the 'chair' and 'cabinet' classes. Results for other classes is considerably lower, e.g. for 'table' and 'desk' instances.

In general, the 3D instance segmentation results are in-line with the performance of the segmentation model. With limited detection capabilities of the instance segmentation model, object instances tend to get confused with objects of classes that are close either physically or semantically.

For the first test set, it can be seen that instances of the 'sofa' class, which is not part of the initial training set, are detected as chairs instead. Further examples of similar objects include 'cabinet' and 'counter', 'desk' and 'cabinet', and 'bookshelf' and 'cabinet.'

Example of object instances misclassified based on physical proximity feature a whiteboard mounted on a wall, assigned to the 'wall' class, and a set of books on a desk, assigned to the 'desk' class.

As is known behavior from closed-set object detection networks, instances of unknown classes tend to be assigned to one of the known classes with high confidence scores [LPV22]. In this regard it could be helpful to not only increase the performance of the instance segmentation network on known classes, but also extend the framework to include a prediction quality estimate, as for example proposed by Rottmann et al. [RCH+20].

In this context, it is worth mentioning again that the reconstruction quality differs between scenes processed online in our framework, with Voxblox++ at low frame rates,

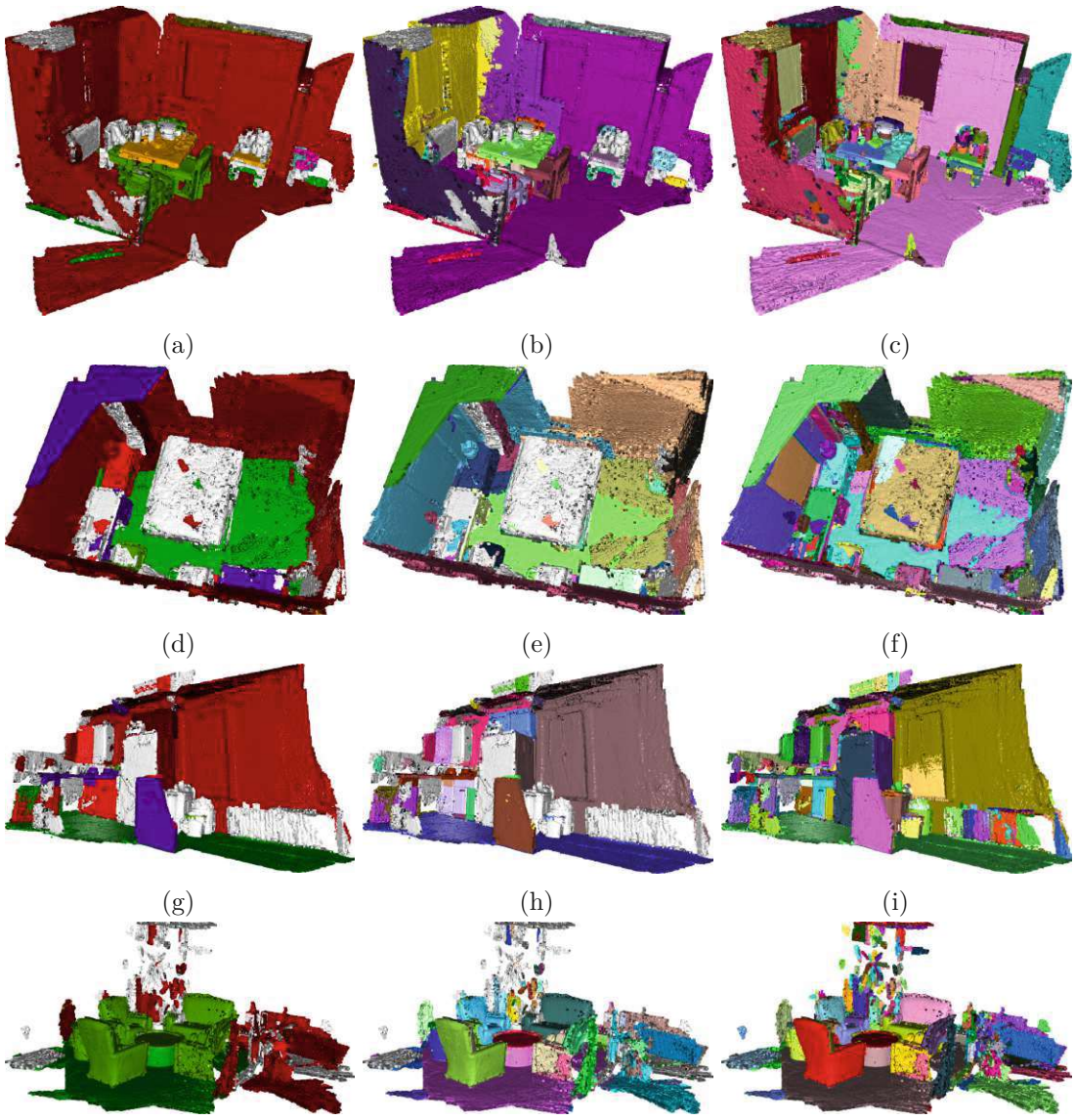
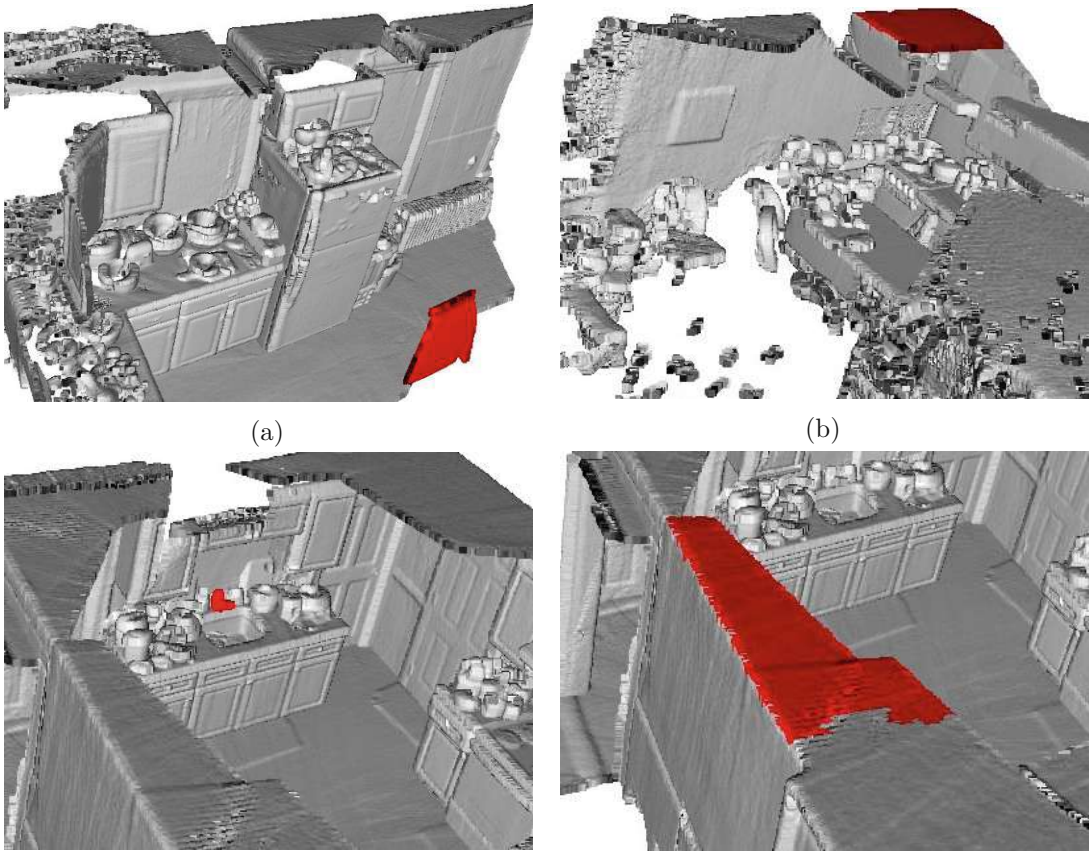


Figure 4.4: Examples of scenes reconstructed with our framework and the initial instance segmentation model. Scenes shown are, from top to bottom row: 223, 234, 255, and 294. The reconstructions are colored based on different per-vertex attributes, from left to right: semantic, instance, and segment labels. Areas without attributes are colored in light grey. Best viewed with magnification.



(a) (b) (c) (d)

Figure 4.5: Examples of unlabeled segments in ground truth mesh models, mostly representing smaller segments of walls, ceilings, and various background items.

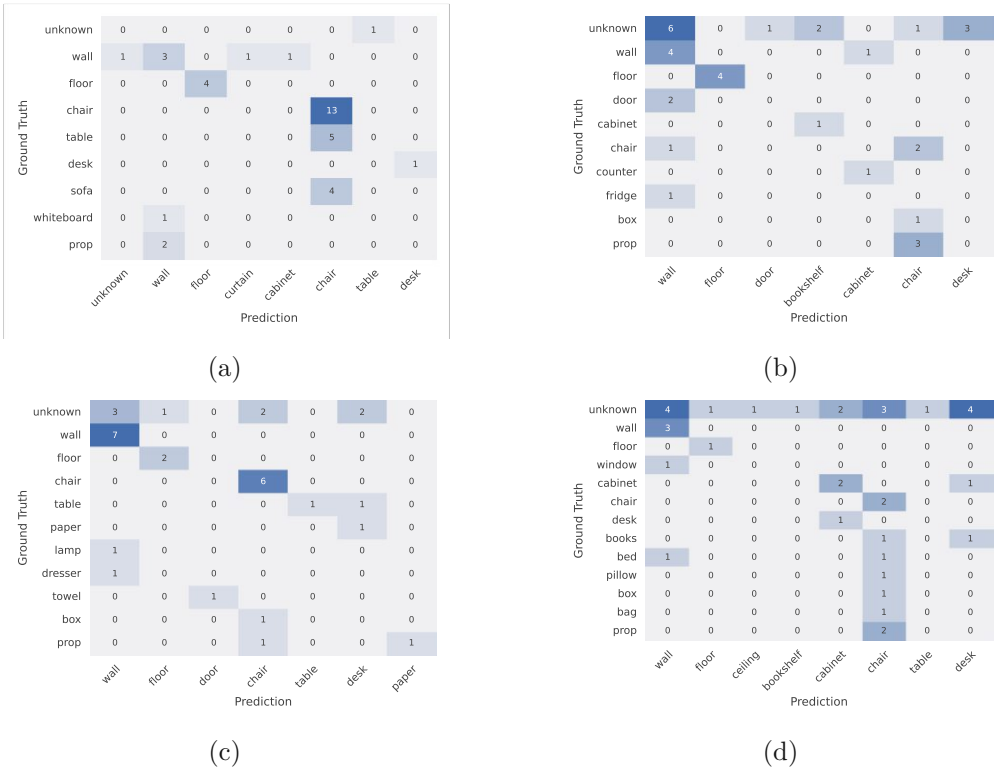
and the ground truth mesh models, reconstructed with a global optimization method running offline at non-interactive frame rates [CZK15].

These differences in reconstructed geometry may directly affect evaluation results, as any mismatch between ground truth and predicted geometry can change the overlap measures with IoU scores. This in turn influences the measurement of object detection precision and confusion matrices.

There are several instances of segments without class labels in the ground truth dataset. These are typically small segments at the boundaries of scenes that were left unlabeled. Figure 4.5 illustrates several of these segments in ground truth data.

4.4.5 Novelty Detection and Category Discovery

With each completed recording of a scene set, the novelty detection node prepares a subset of novel segment views for category discovery. These samples are embedded into a



(a) (b) (c) (d)

Figure 4.6: Combined confusion matrices for the initial model and scene test sets 1 to 4, in order. Each matrix plots the classes of predicted instances vs. classes of matching ground truth instances. Instance matching is performed based on an Intersection over Union (IoU) score of > 0.5. Rows and columns with all zeros are removed for brevity.

low-dimensional feature space, grouped into clusters, and each cluster is scored. The top N clusters are then used as new categories in the following incremental learning step.

Here we visualize the cluster results using the low-dimensional embeddings of all samples, as produced by PCA and t-SNE, and the DBSCAN clustering method. To provide more readable results, the visualizations are split into selected clusters, other (not selected) clusters, and all clusters, for each scene test set. All visualizations are shown in Figure 4.7.

With the exception of the first clustering, all show a high number of clusters with large numbers of overlaps. However, selected clusters are well-separated, dense, and mostly non-overlapping. This indicates that the cluster selection method at hand is suitable to detect new categories within the clustering results.

4.4.6 Retraining Datasets

To complement the category discovery visualizations, we present several samples from selected clusters. All of these samples are included in the retraining datasets and used in the incremental learning steps. The samples are shown in Figure 4.8, Figure 4.9,

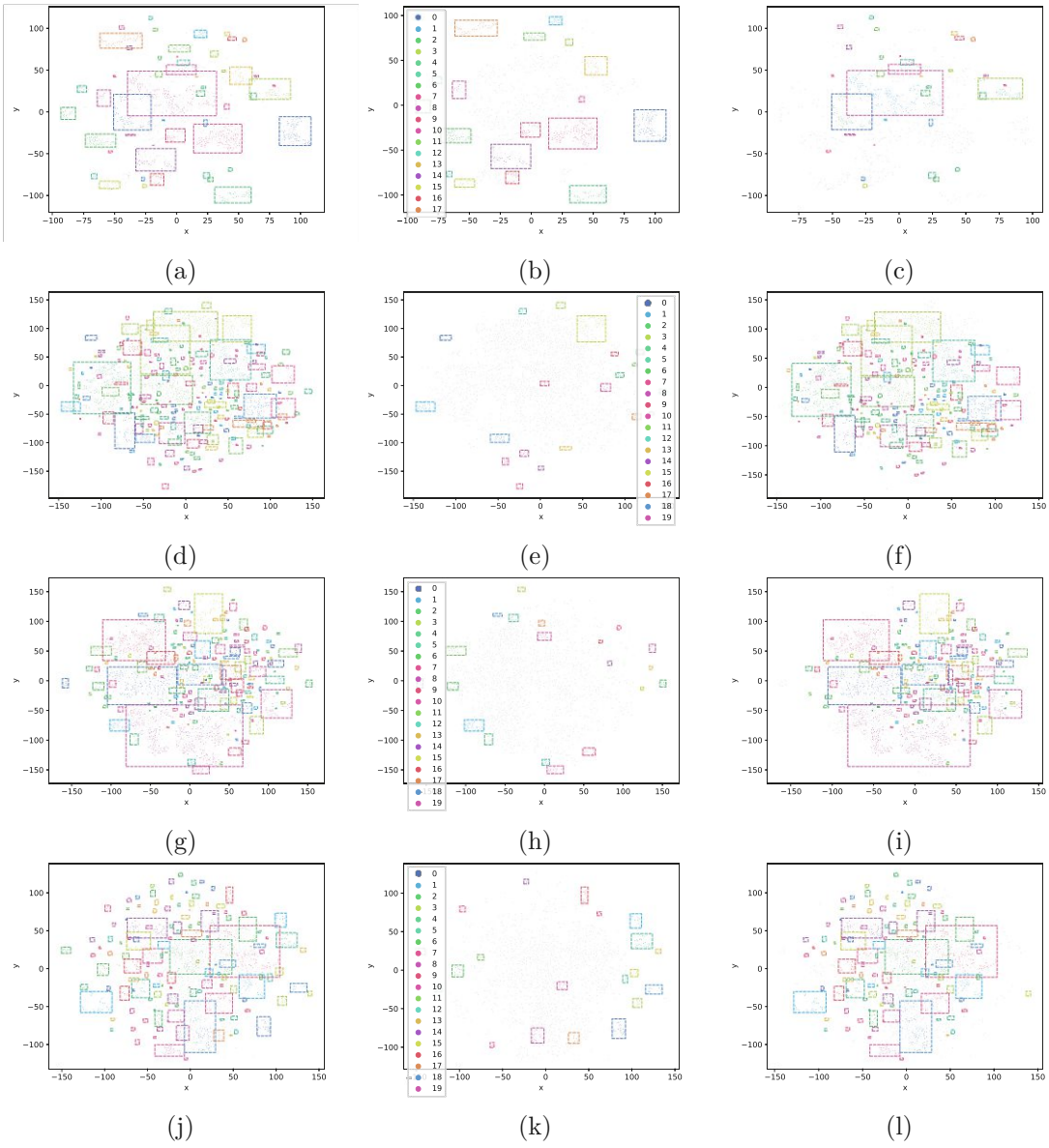


Figure 4.7: Scatterplots of clustering results in category discovery, for the test scene sets. Clusters selected for incremental learning are shown in the center column, other clusters on the right. The left column shows all clusters combined. Rows 1 to 4 show plots for test sets 1 to 4, respectively. Samples displayed in grey are non-core points or noise. For clarity, cluster memberships are indicated with dashed bounding boxes. Best viewed with magnification.

Figure 4.10, and Figure 4.11. These samples only include a small part of the retraining datasets, as there are up to 20 new categories in each.

In some images, a misalignment between RGB and depth images can be observed. This is presumably caused by a lack of hardware synchronization between the RGB and depth sensors employed for scene recordings, and as such depends on the type of sensors used. The misalignment is especially noticeable with fast camera movements, and may affect the quality of segmentation masks in the retraining dataset.

Novel categories include a variety of objects, from kitchen utensils to household items and furniture. In several instances, category discovery is able to group similar objects from the same scene and across different scenes into the same category. This indicates that the approach in our framework is capable of generalizing novelty detection and category discovery across multiple scenes, or multiple recordings of the same scene.

A significant portion of the categories come from spurious novelty detections. These are the result of instance segmentation failing on segments from known classes, including walls and floors. Since these segments are not semantically labeled, novelty detection considers them as new objects. Such segments are therefore also processed in category discovery and can end up as samples in the retraining set.

The most straightforward solution to this problem would be to improve instance segmentation for known classes. If all segments of walls, floors, or other known objects are properly detected and semantically labeled, they will by definition not end up in the novelty detection and category discovery pool of segment views.

4.4.7 Incremental Learning Steps

With the retraining datasets that are generated by category discovery from each scene test set, it is possible to update the initial instance segmentation model. Each incremental learning step takes the initial model and extends it with one of the four retraining datasets. Every dataset contains up to 20 new categories. This number may vary depending on the number of clusters detected in category discovery and the results of cluster selection.

When a model is retrained, one important aspect to evaluate is the model performance on the detection and segmentation of old classes. Since ‘catastrophic forgetting’ is a major concern in incremental learning methods, it is necessary to ensure knowledge of old classes is retained.

To this end, we evaluate each model on the initial test set for model fine-tuning. Figure 4.12 summarizes the results. In general, the performance on both the object detection and the instance segmentation tasks are comparable to that of the initial model. This indicates that the incremental learning method of Cermelli et al. [CGFC22] is able to prevent catastrophic forgetting, and retain model performance on old classes.

However, for test sets 2, 3, and 4, there is a distinct drop in performance for the ‘wall’ and ‘floor’ classes. This might be explained by the retraining sets containing one or more novel categories that consist of samples belonging to one of these classes.

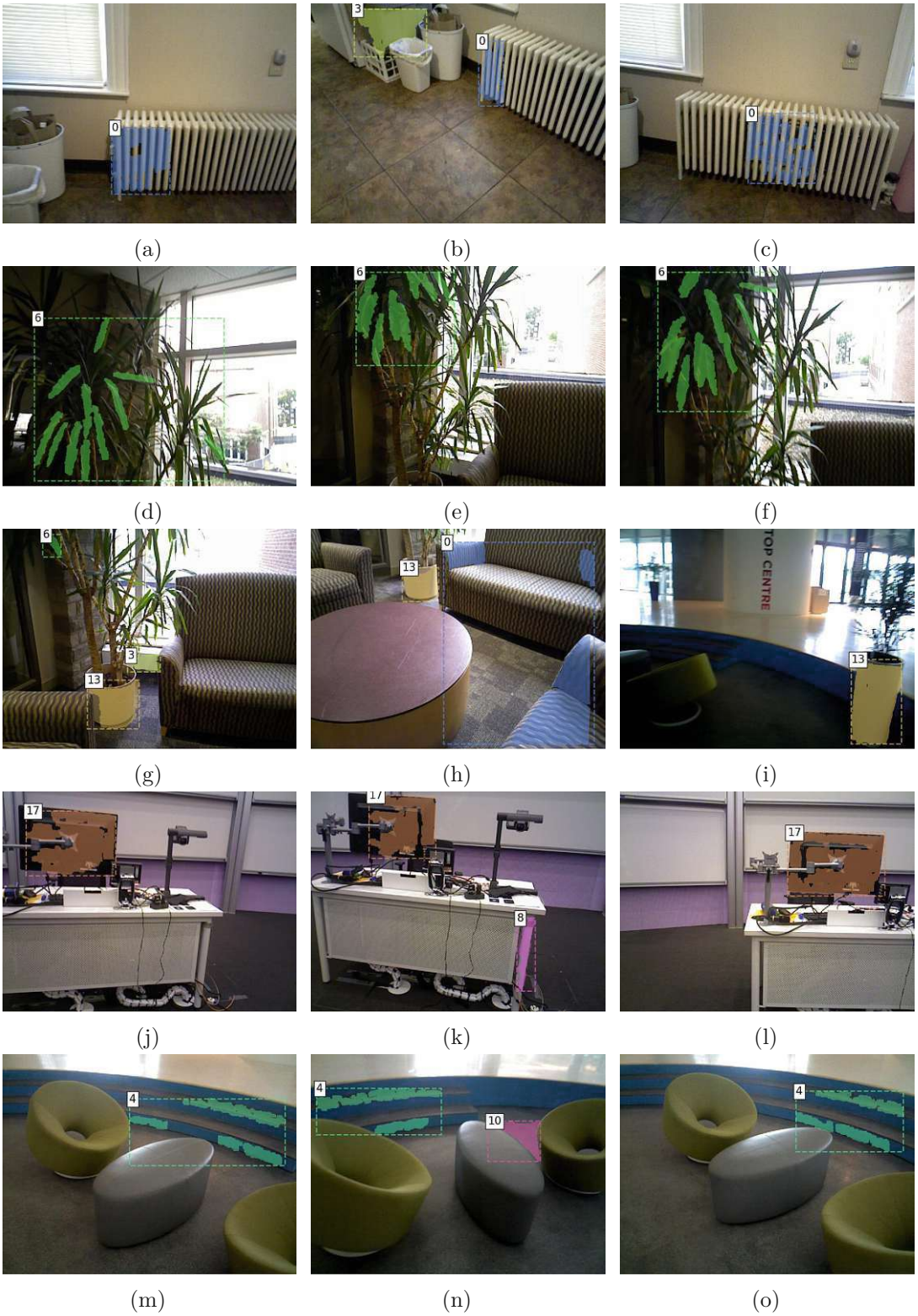


Figure 4.8: Examples of category discovery samples from scene test set 1. Objects include, from top to bottom row: a radiator, a potted plant, a flower pot, a computer monitor, and a set of stairs. Note how (g) to (i) groups pots from two different scenes into one category.

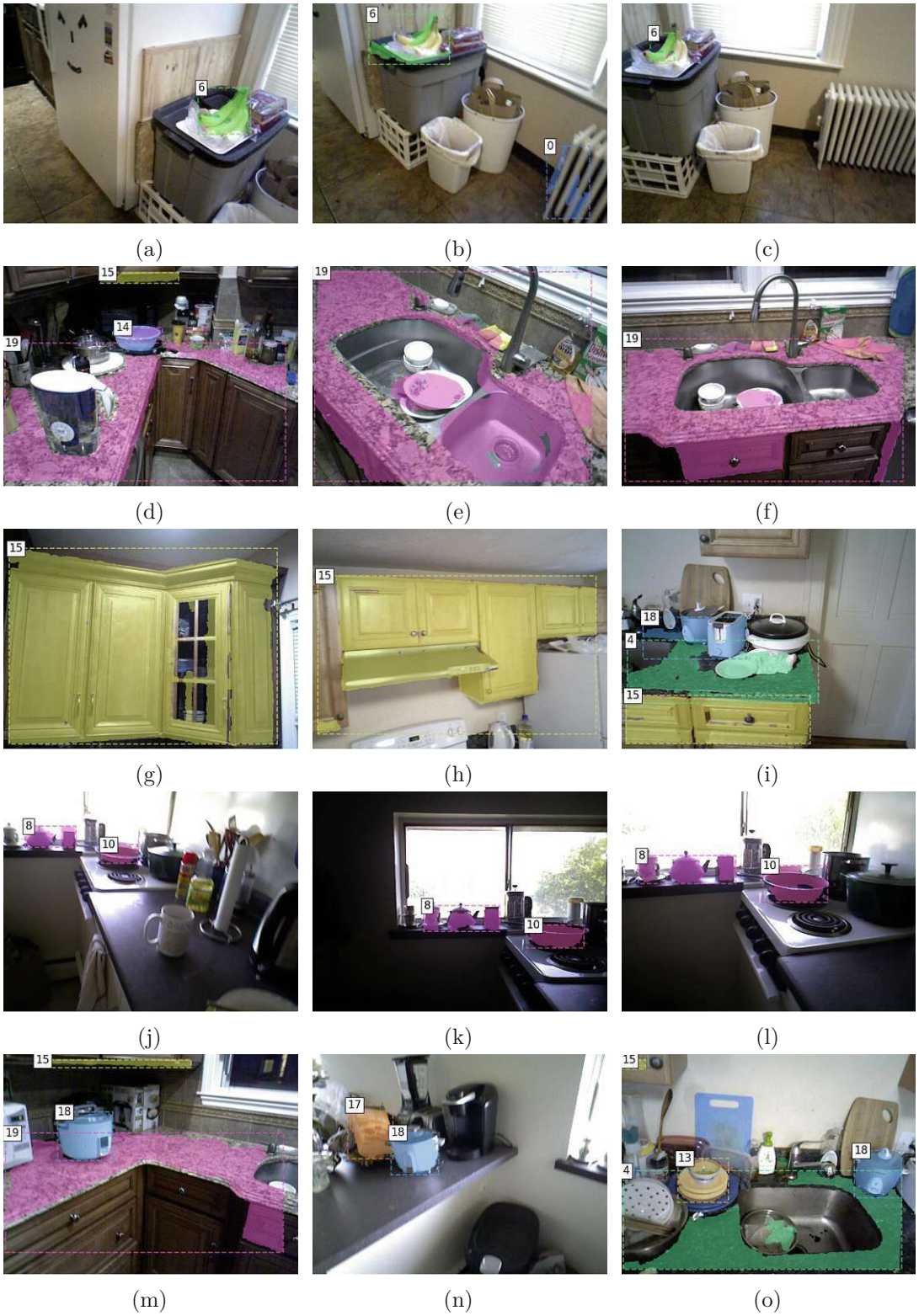


Figure 4.9: Examples of category discovery samples from scene test set 2. Objects include, from top to bottom row: a bunch of bananas, a counter, cupboards, a pan and teapots, and rice cookers. Note how (g) to (i) and (m) to (o) group objects from different scenes into one category, respectively.

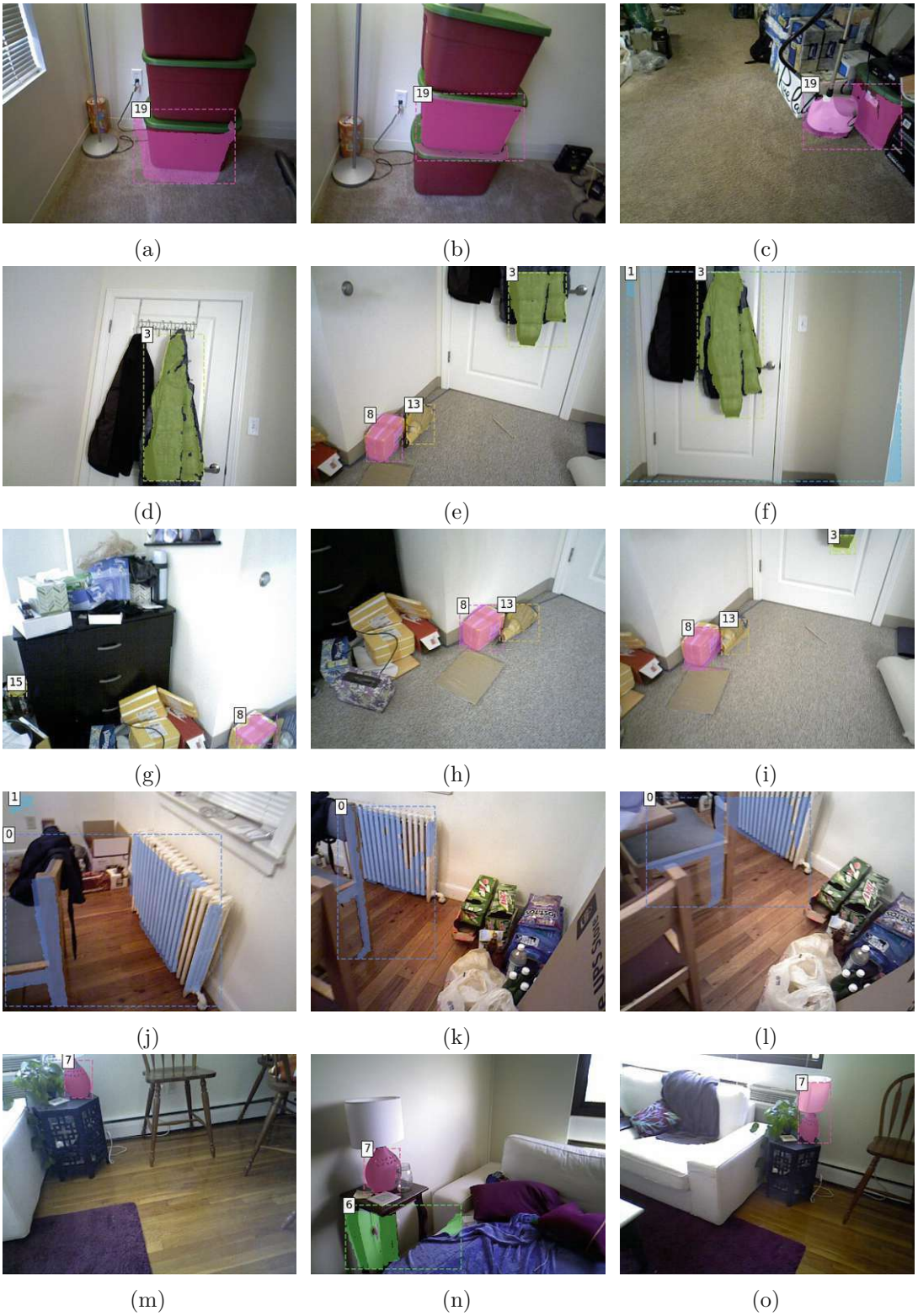


Figure 4.10: Examples of category discovery samples from scene test set 3. Objects include, from top to bottom row: storage boxes, a jacket, a cardboard box, a radiator, and a lamp. In (m) to (o), two similar lamps are grouped into one category.

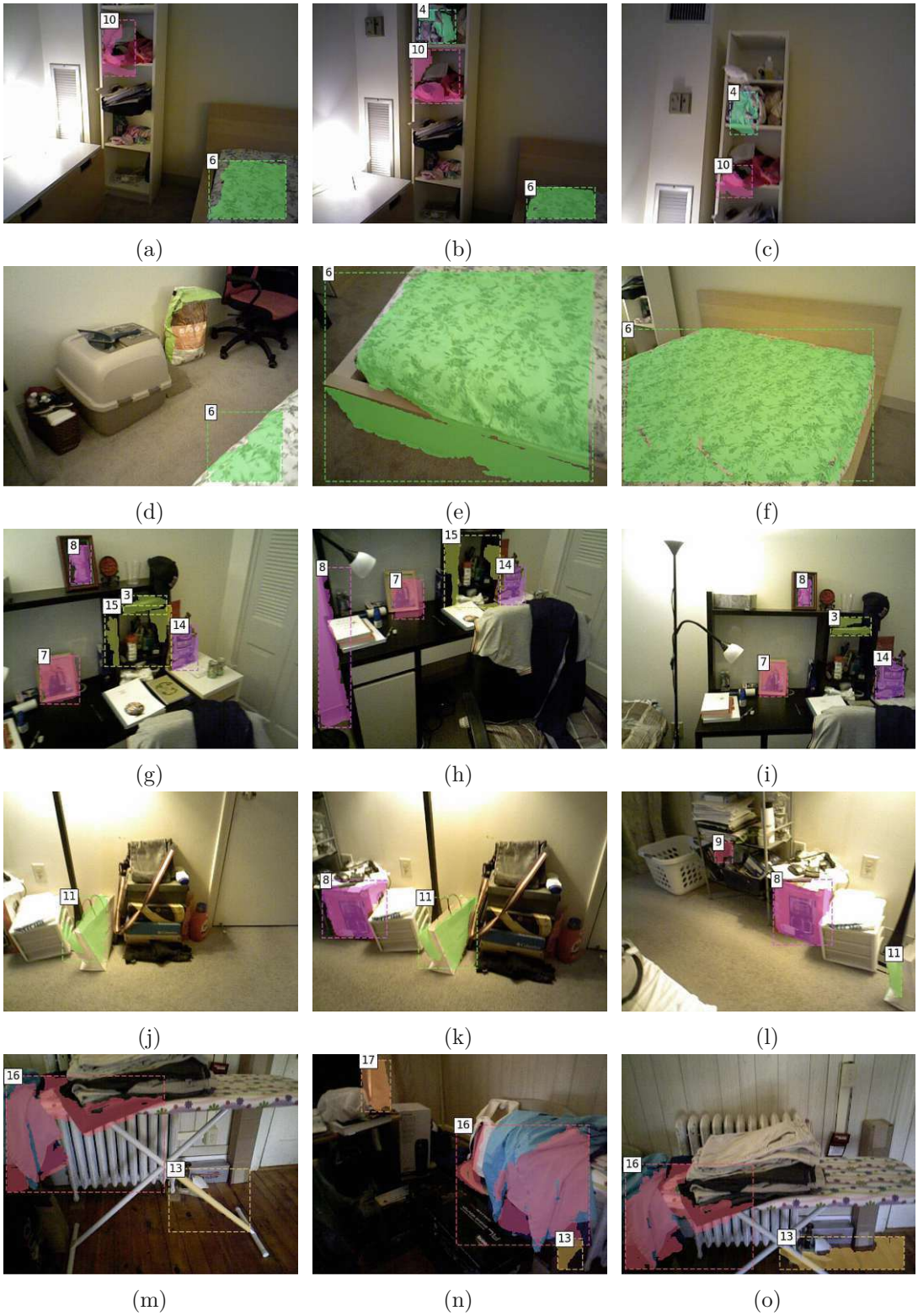


Figure 4.11: Examples of category discovery samples from scene test set 4. Objects include, from top to bottom row: clothes on a shelf, a bed, picture frames, a paper bag, and clothes on an ironing board.

If the original model is retrained to label 'wall' and 'floor' classes with a new pseudo-label, it will consequently impact the performance on the original classes. Likewise, the lower performance on the 'curtain' class in scene test set 3, and on the 'window' class in set 4, might be related to overlaps between the novel categories and old classes.

Interestingly, for the model retrained on the novel categories from scene set 2, the performance on the 'ceiling', 'window', and 'curtain' classes is above that of the original model. It can be hypothesized that the reduced performance on 'floor' and 'wall' classes, leads to increased confidence scores and, in turn, higher detection rates for semantically related classes such as 'ceiling', 'window', and 'curtain.' However, this remains to be tested with additional evaluations before a clear conclusion can be drawn.

4.4.8 Retrained Object-Centric Mapping

The instance segmentation models retrained in the incremental learning steps are used again to reconstruct the same sets of scenes as with the initial network. With the models refined to detect novel categories, here we show the effects of the extended instance segmentation on the reconstructions of several scenes.

Test 1

For scene 294 from the first test set, as shown in Figure 4.13, the network is able to detect part of the potted plant in the background as a new object. While not easily visible, there is another part of a second potted plant to the right that is detected as well. While the complete plant is not detected, presumably related to the poor quality of training data segmentation annotations caused by misaligned sensor hardware, this result indicates that the framework is able to detect even small and intricate structures, and transfer results to distinct, but visually similar objects.

Test 2

In scene 255, visualized in Figure 4.14, the radiator to the right is detected in large parts, even though the retraining dataset only contains smaller segments as annotations. From the scene visualizations one can also see how individual segments are merged into larger instances by the semantic refinement process of Voxblox++. Another novel segment is part of the ceiling in the upper left corner.

Test 3

Scene 223 from test set 3, shown in Figure 4.15, features another radiator, distinct from the one in scene set 2, and the front of a small table. The radiator is almost completely detected, but split into two instances. This indicates that the segment merging was not entirely successful for this instance.

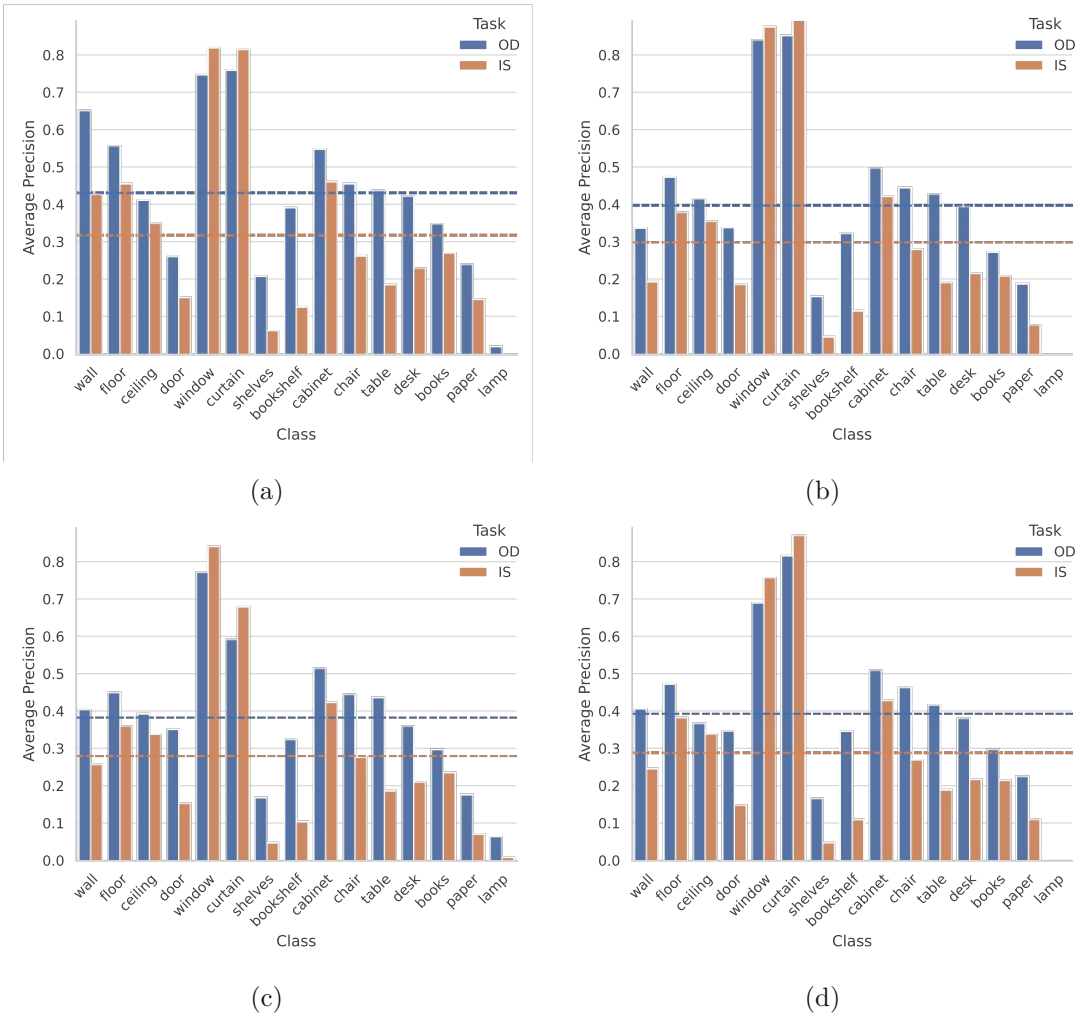
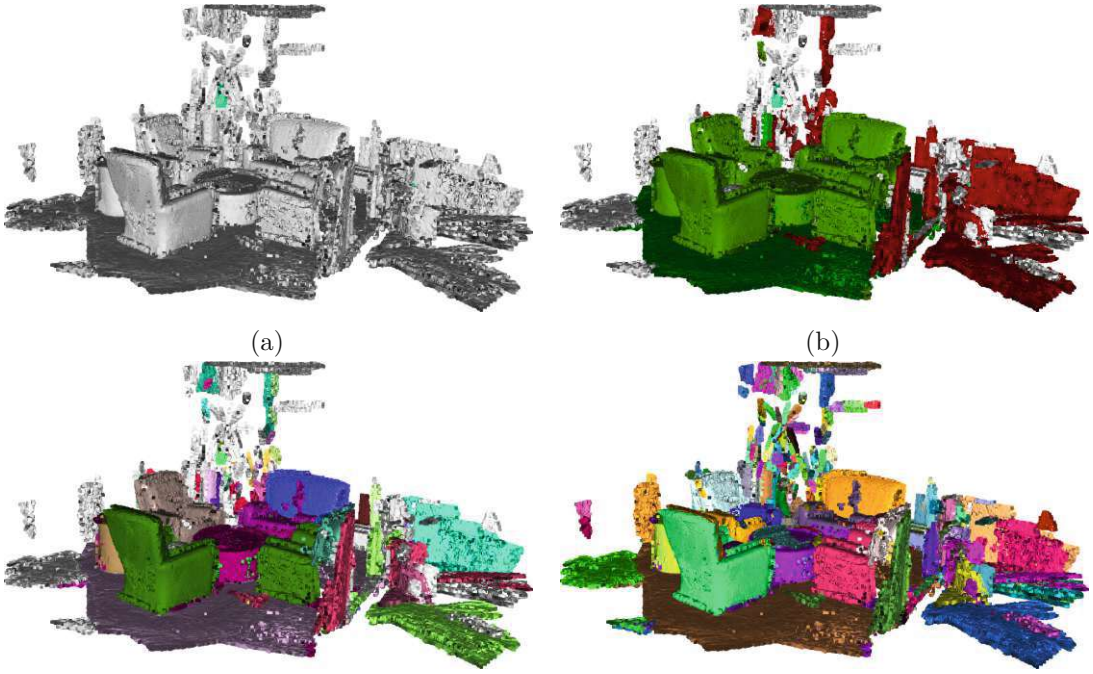


Figure 4.12: Object detection (OD) and instance segmentation (IS) performance of the Mask R-CNN model after retraining on novel categories from test scene sets, evaluated on the initial training dataset. Subplots show results for scene test sets 1 to 4, in order. Dashed bars represent mAP values over all classes for each task.



(a) (b) (c) (d)

Figure 4.13: Four renderings of scene 294, reconstructed with a retrained instance segmentation model and colored with different sets of labels. Subplots show, in order: newly detected categories, all class labels, instance labels, and segment labels. Best viewed with magnification.

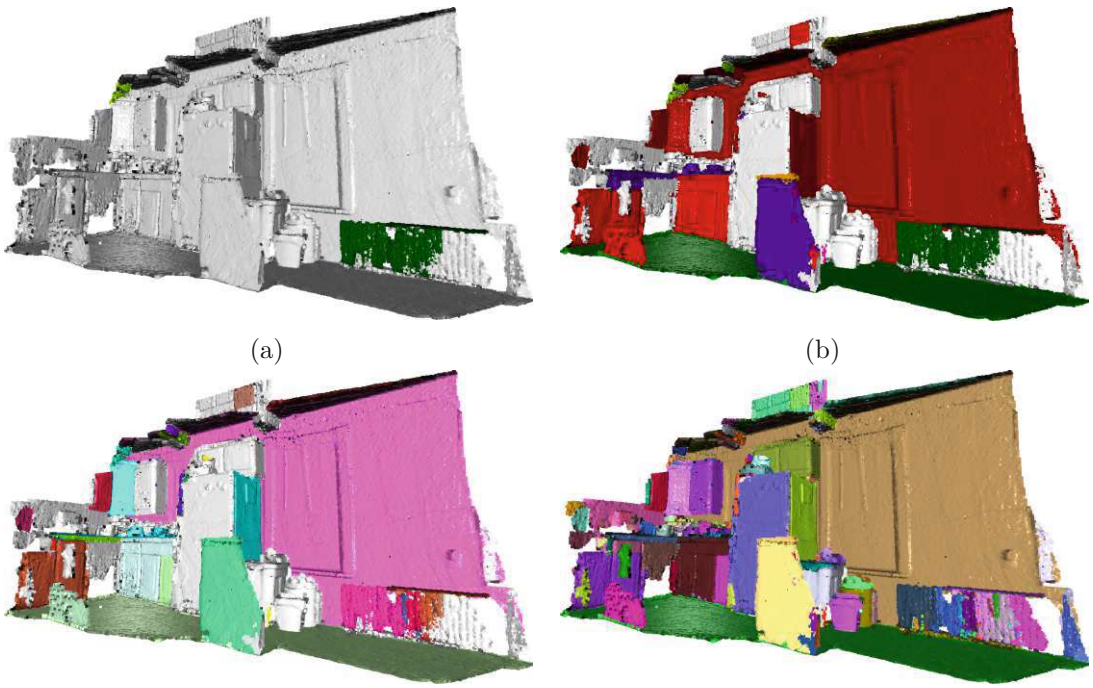
Test 4

In scene 234, a bed and part of a cat tree are detected from the unknown objects. This scene is shown in Figure 4.16. The bed is almost completely detected, except for part of the blanket. Part of the floor is merged into the object, presumably as the instance segmentation mask is overlapping this segment during the scene recording. The cat tree is also detected in large parts, however its category is not specific to the object but rather related to a cluster of wall segments.

Other Scenes

Additional images show more examples of objects detected with the retrained models. In Figure 4.17, these objects are: a jacket and parts of the ceiling, a cardboard box and part of a blanket, a lamp and books in a bookshelf, and the backrests of folding chairs. Figure 4.18 shows the following objects: the lid of a cardboard box and parts of a kitchen counter, a sink and kitchen utensils, segments of a wall below a whiteboard, and parts of stairs.

As these objects are only partially detected, this suggests the amount or quality of



(a) (b) (c) (d)

Figure 4.14: Four renderings of scene 255, reconstructed with a retrained instance segmentation model and colored with different sets of labels. Subplots show, in order: newly detected categories, all class labels, instance labels, and segment labels. Best viewed with magnification.

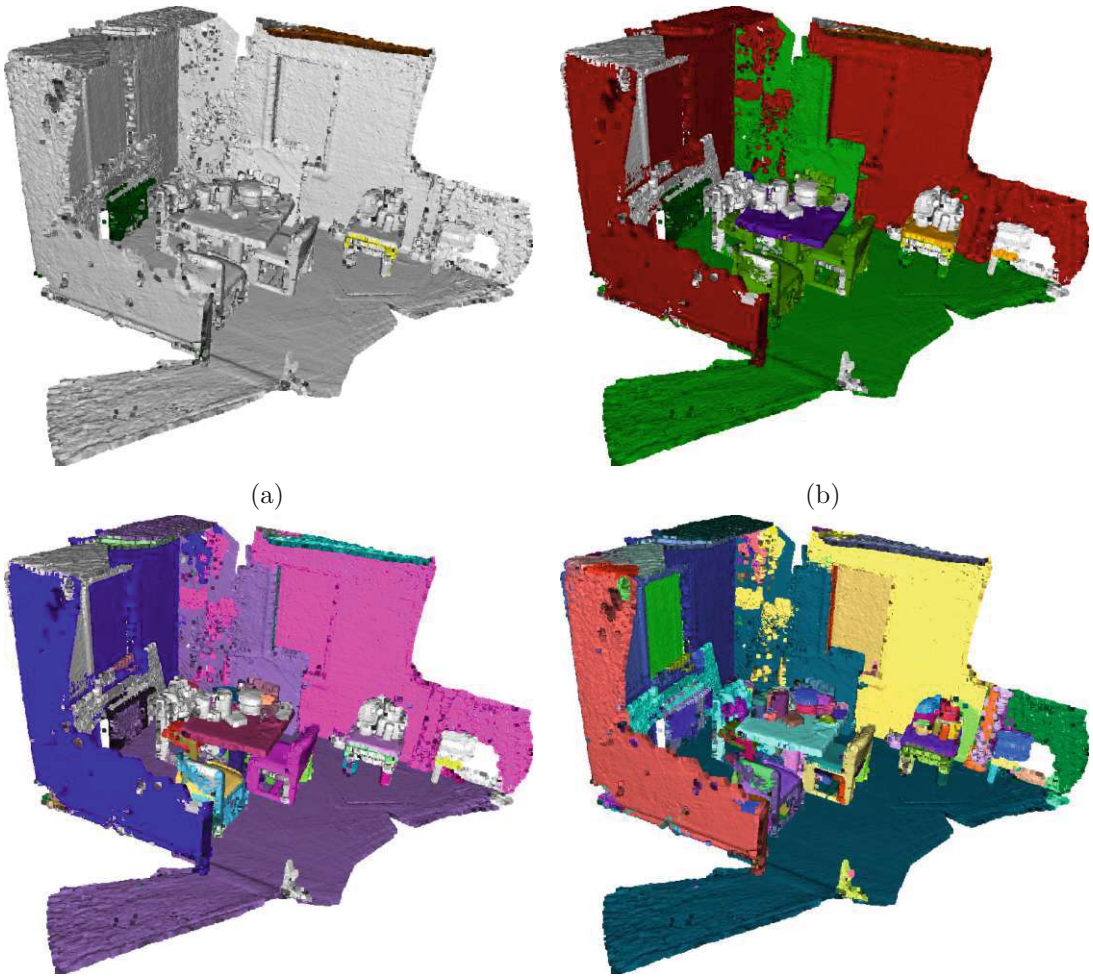
retraining data was not sufficient, or that retraining was stopped too early. On the other hand, these results confirm that, in principle, the framework is able to extract meaningful new categories from observations without human intervention, and recognize these objects in future recordings.

4.4.9 Retrained 3D Instance Segmentation

The confusion matrices are computed for scenes reconstructed with retrained models as well. Results are comparable with the initial scene reconstructions, with some scores lower and a few cases of improved detections.

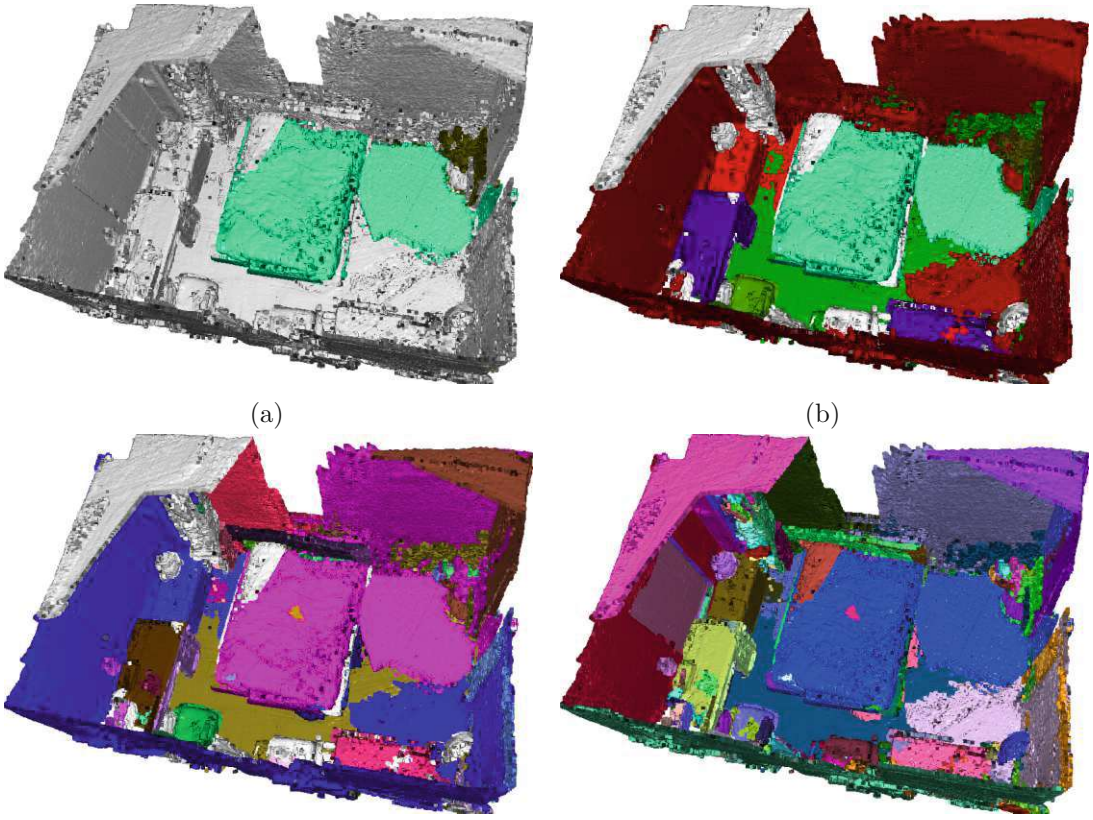
For example, in the retrained scene test set 3, a piece of paper is correctly detected that was not detected in the original reconstruction. In the retrained scene test set 4 there is an additional correct detection of a desk instance.

Most object instances from new categories do not show up in the confusion matrices. These instances do not overlap the respective ground truth instances by more than 50 %, as measured by IoU, and thus do not count as positive predictions. As with the initial



(a) (b) (c) (d)

Figure 4.15: Four renderings of scene 223, reconstructed with a retrained instance segmentation model and colored with different sets of labels. Subplots show, in order: newly detected categories, all class labels, instance labels, and segment labels. Best viewed with magnification.



(a) (b) (c) (d)

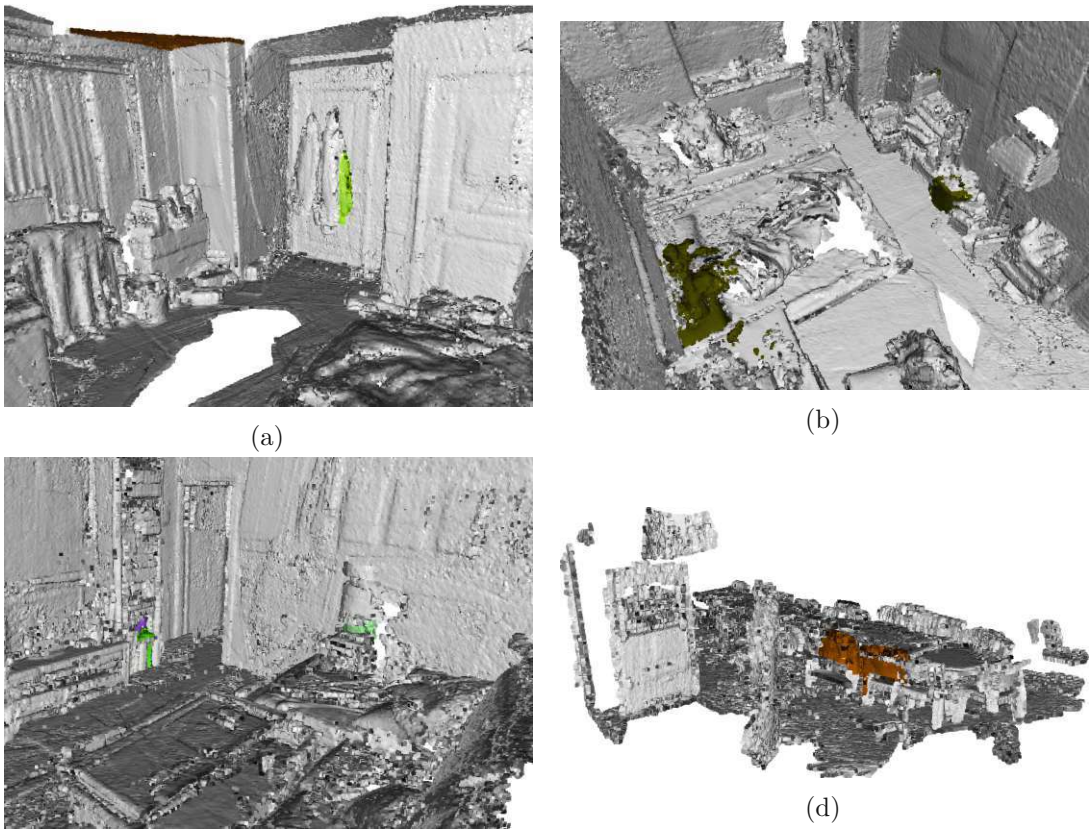
Figure 4.16: Four renderings of scene 234, reconstructed with a retrained instance segmentation model and colored with different sets of labels. Subplots show, in order: newly detected categories, all class labels, instance labels, and segment labels. Best viewed with magnification.

scene sets, the effects of reconstruction quality on evaluation results have to be taken into account.

In order for new category instances to be properly detected, instance segmentation would have to be improved. These improvements would typically come from larger retraining sets with higher-quality images and annotations, including accurately aligned depth sensor data, and from longer incremental learning schedules.

4.4.10 Memory & Storage Requirements

Finally, we list the approximate memory and storage requirements for the framework at runtime, as far as they are different from the original works used herein. The memory consumption of the global segmentation map, Voxblox++, and the instance segmentation network, Mask R-CNN Benchmark, are unchanged to their original releases.

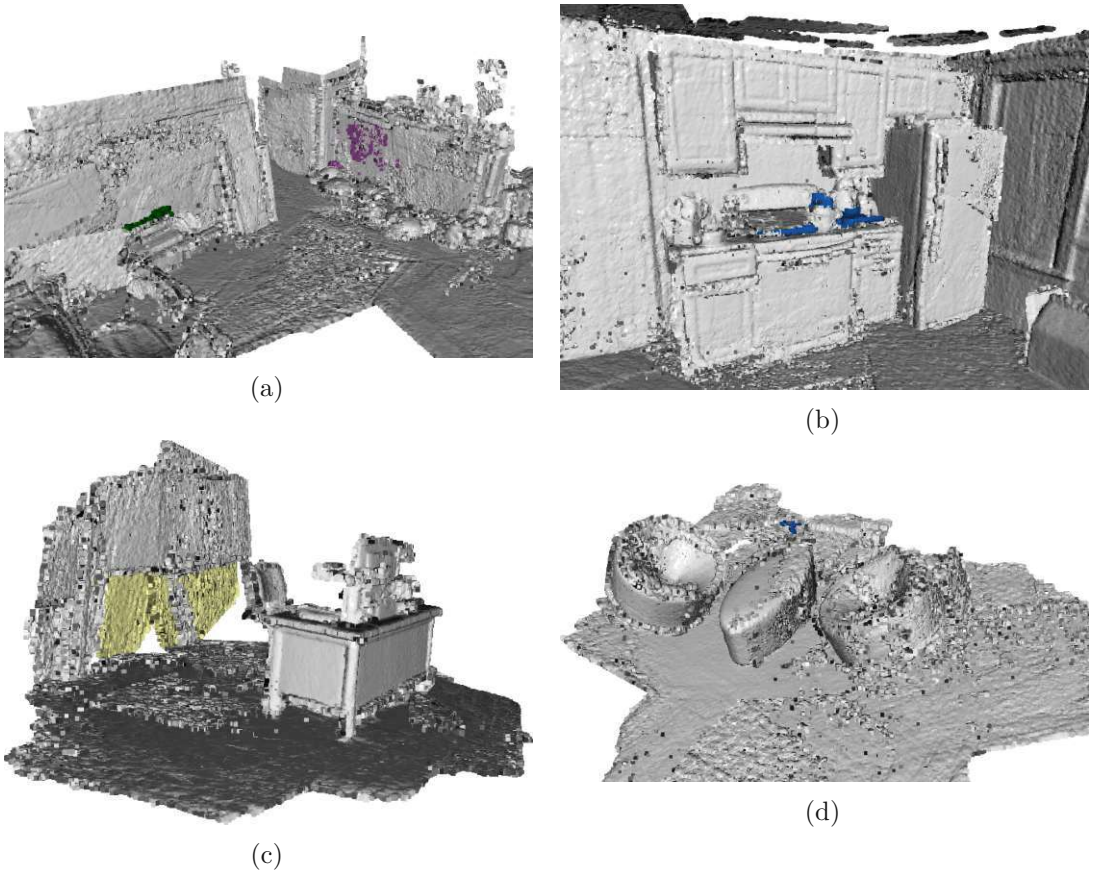


(a) (b) (c) (d)

Figure 4.17: Further examples of newly discovered categories in scenes reconstructed by our framework, with a retrained instance segmentation model. Base classes are unlabeled in this visualization. Scenes shown, in order: 093, 240, 272, and 609. Best viewed with magnification.

The largest memory and storage requirement comes from novelty detection, which collects all RGB frames, segmentation maps, feature maps, and related segment metadata during scene recordings. In our implementation, this data is stored uncompressed and thus takes several gigabytes of space before it is processed for category discovery.

As outlined earlier, most of the data could be processed online with a more sophisticated update scheme. Image compression could also help to reduce the memory footprint, at the expense of additional computations from compression and decompression. The segment metadata itself is very compact and typically consists of less than 20 segments per-frame, plus time stamps, pointers to per-frame data and merged segments.



(a) (b) (c) (d)

Figure 4.18: Further examples of newly discovered categories in scenes reconstructed by our framework, with a retrained instance segmentation model. Base classes are unlabeled in this visualization. Scenes shown, in order: 087, 276, 286, and 527. Best viewed with magnification.

4.4.11 Timings

For the initial segmentation model fine-tuning, the training time is over 13 h. The retraining times for incremental learning vary between 2 h 57 min and 4 h 36 min.

The system runs at an overall frame rate of about 1 to 2 frames per second during scene recordings. The instance segmentation network can perform multiple inferences per second on input frames of size 640×480. The Mask R-CNN benchmark reports an approximate inference time of 0.1678 s per image, or about 5.96 images per second.

This indicates that the overall frame rate is limited by processing in the depth segmentation and global map nodes. This could be accelerated with multi-threading, however not all parts of the implementation are thread-safe as of now.

The runtime for offline processing of category discovery data is dominated by preprocessing,

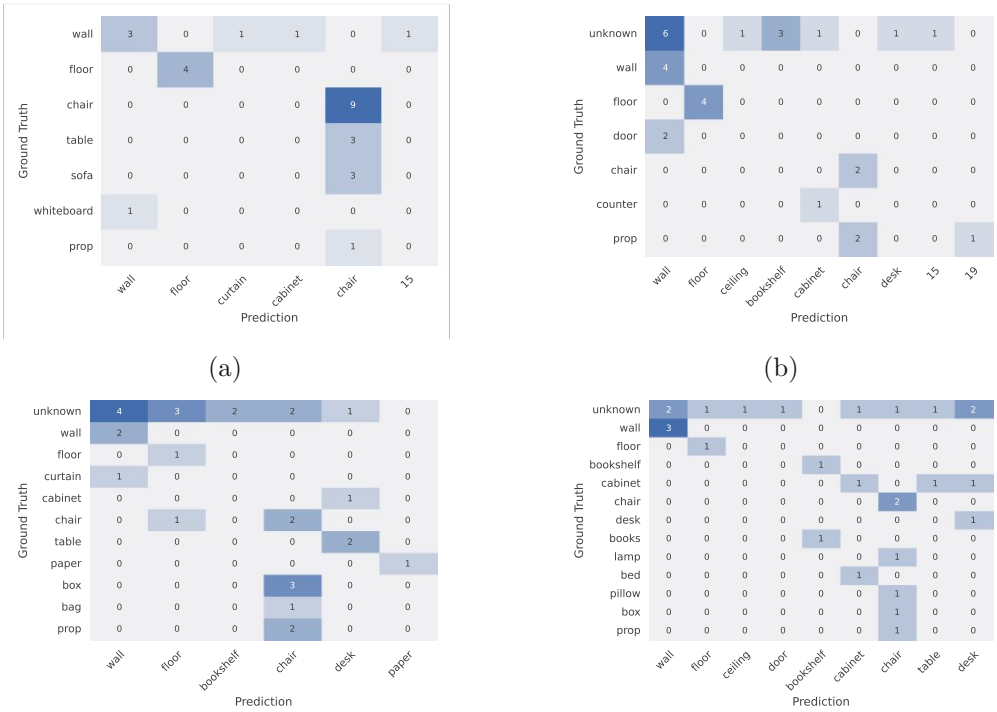


Figure 4.19: Combined confusion matrices after retraining for scene test sets 1 to 4, in order. Each matrix plots the classes of predicted instances vs. classes of matching ground truth instances. Instance matching is performed based on an IoU score of > 0.5. Rows and columns with all zeros are removed for brevity.

i.e., extracting feature vectors for each segment. Feature embedding with t-SNE takes the second longest time to complete, if we discount the actual loading and storing of data in the category discovery node.

For scene test set 1, the computation of feature vectors takes approximately 47,0 s for all five scenes combined. Computing the feature embeddings takes about 39,7 s. All other steps in category discovery, for example computing the distance matrix, clustering, and cluster selection, take less than a second to complete. The numbers for the other test sets are in a similar range.

4.5 Ablation Study

To better understand the effect of different system parameters on object detection and segmentation performance, we perform two additional sets of experiments. The first set involves instance segmentation models that are retrained with extended learning schedules. The second set of experiments uses the same models as the first, but additionally simulates

an increased processing frame rate of the system.

Both are performed on and evaluated for each of the four scene test sets. While these two additional evaluations do not provide enough data to draw definitive conclusions, they illustrate the importance of instance segmentation performance and processing frame rate on object detection and reconstruction quality in object-centric mapping.

4.5.1 Extended Incremental Learning Schedule

For the first set of tests, the incremental learning schedule for retraining is extended by a factor of 1.8, equal to 18000 iterations. Figure 4.20 shows the performance of the retrained models on the initial test set.

The performance for the 'wall' and 'floor' classes is improved for test sets 2, 3, and 4, and also for 'ceiling' and 'window' classes in set 4, compared to the models with a shorter retraining schedule. Results are significantly worse for the 'curtain' class in set 4. Other classes show little to no noticeable variation in performance.

Figure 4.21 summarizes the confusion matrices for each scene test set. In comparison to the regular training schedule, the number of correct detections has increased overall, but also the number of misclassified objects. With an increased learning schedule, there are also more objects classified as belonging to one of the new categories.

Examples of scene reconstructions with semantic, instance, and segment labels are shown in Figures 4.24, 4.25, 4.26, and 4.27. Figure 4.28 shows some of the new objects detected in the test scenes.

4.5.2 Increased Processing Frame Rate

The second set of experiments also uses the retrained models with an extended incremental learning schedule. In addition, the processing frame rate is increased approximately by a factor of five. To this end, we decrease the playback speed of test data during evaluation by the same amount, resulting in an effective increase in the number of processed frames.

Most notably, this improves the reconstruction quality of scene recordings, owing to the fusion process in the global map. Following a better surface reconstruction, clutter in geometric segmentation is reduced as well. This leads to a significantly lower number of segments, as can be seen in Figure 4.22. Whether the effect is due to the fusion process or segment merging in Voxblox++, or both, remains to be investigated.

The confusion matrices for each scene test set are shown in Figure 4.23. The number of correct detections is higher than in the previous experiment, with the number of misclassifications staying approximately the same. In particular, there are more objects detected from new categories. Figures 4.29, 4.30, 4.31, and 4.32 show reconstruction examples for four scenes, and Figure 4.33 gives additional examples of detected objects in this setting.

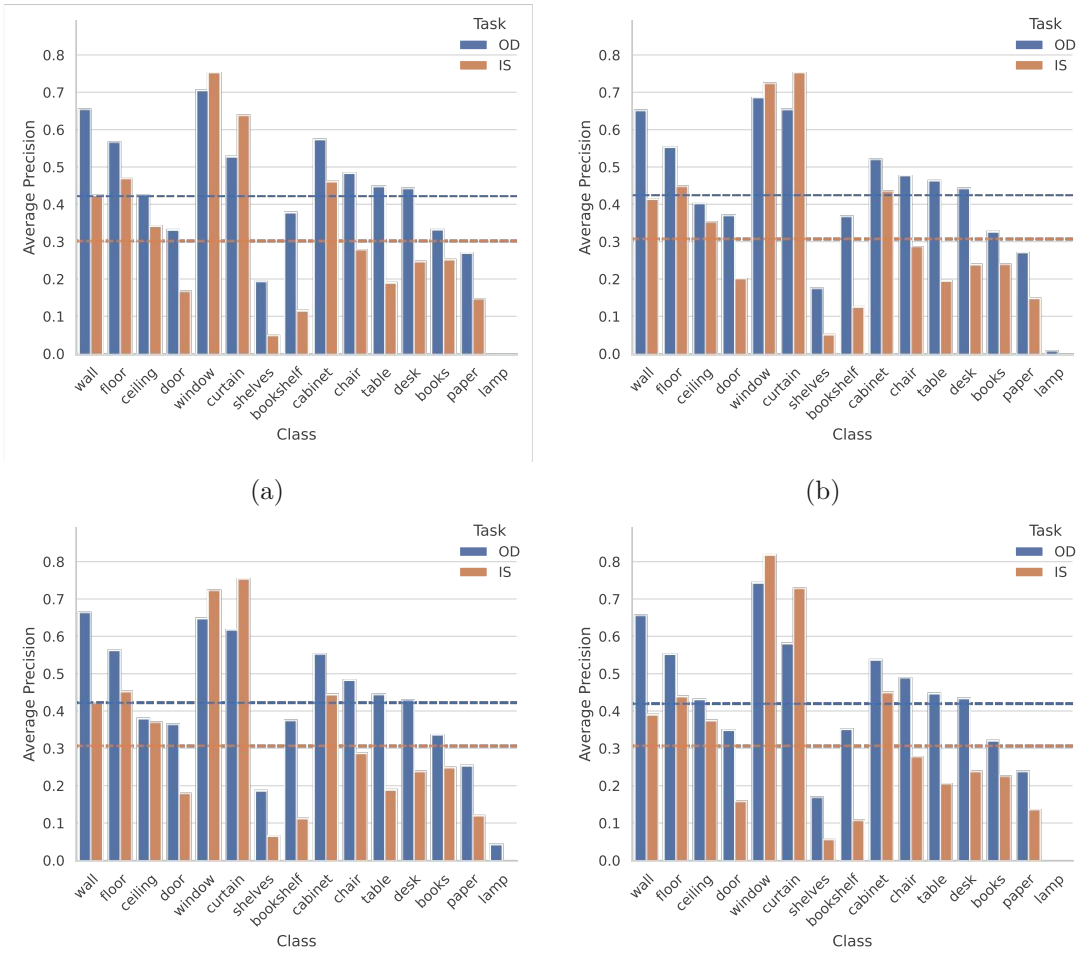
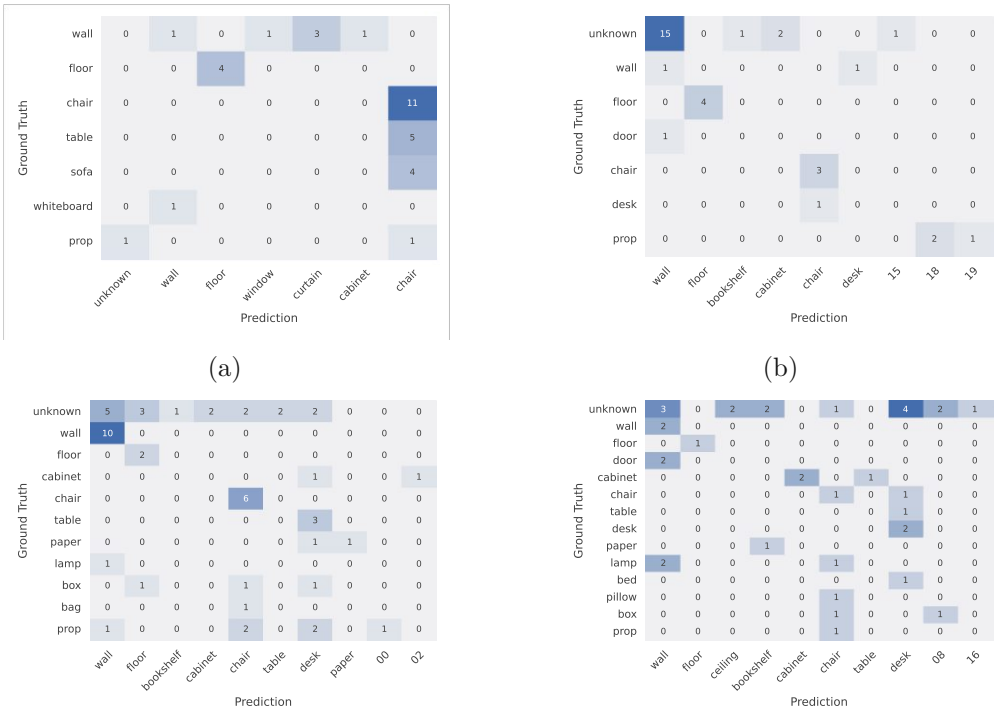
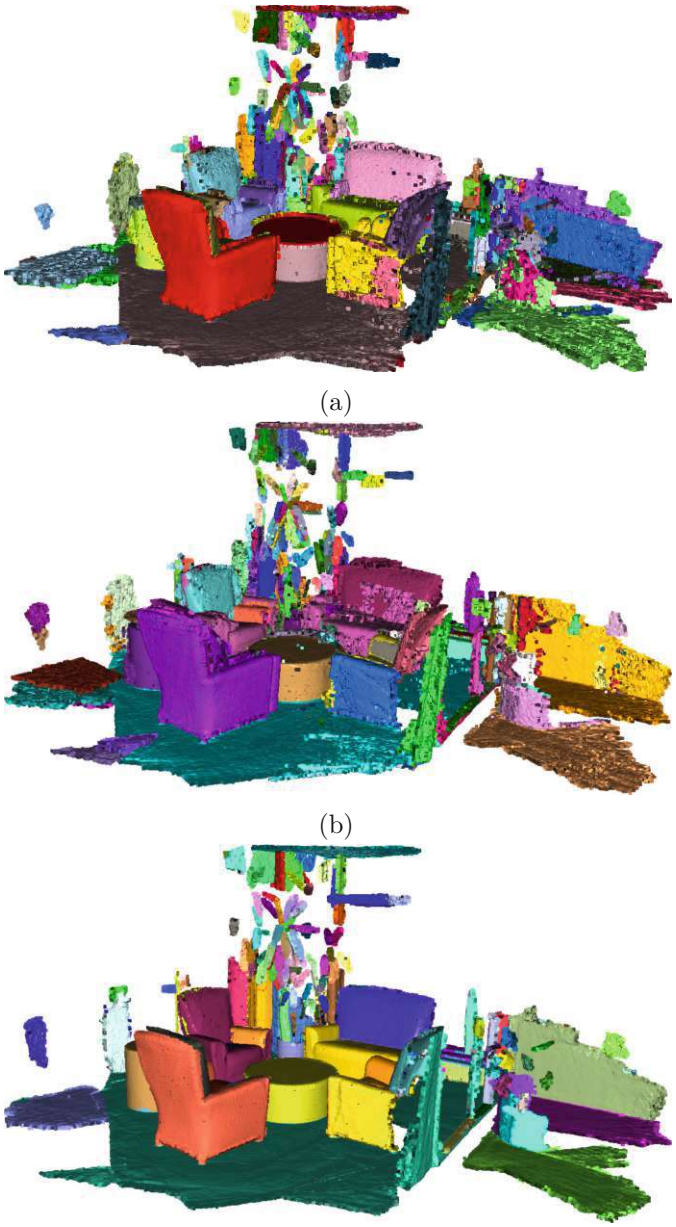


Figure 4.20: Object detection (OD) and instance segmentation (IS) performance of the Mask R-CNN model after extended retraining on novel categories from test scene sets, evaluated on the initial training dataset. Subplots show results for scene test sets 1 to 4, in order. Dashed bars represent mAP values over all classes for each task.



(a) (b) (c) (d)

Figure 4.21: Combined confusion matrices after retraining with an extended learning schedule for scene test sets 1 to 4, in order. Rows and columns with all zeros are removed for brevity.



(a)

(b)

(c)

Figure 4.22: Renderings of scene 294 with color coded segment labels and for three different evaluation settings. Subplots show reconstructions with the initial model, the retrained model with extended learning schedule, and the latter model with an increased frame rate, in order. With a higher frame rate, surface reconstruction improves and the number of segments decreases, particularly noticeable on the right. Best viewed with magnification.

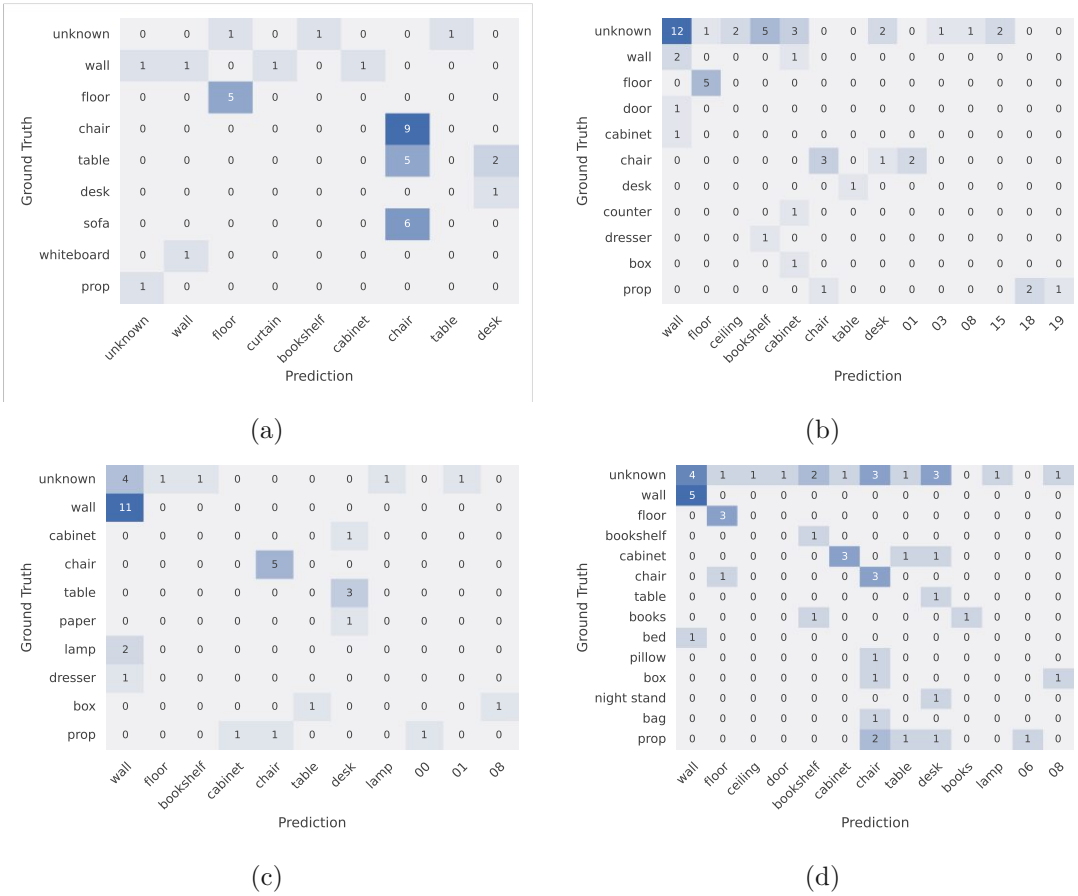
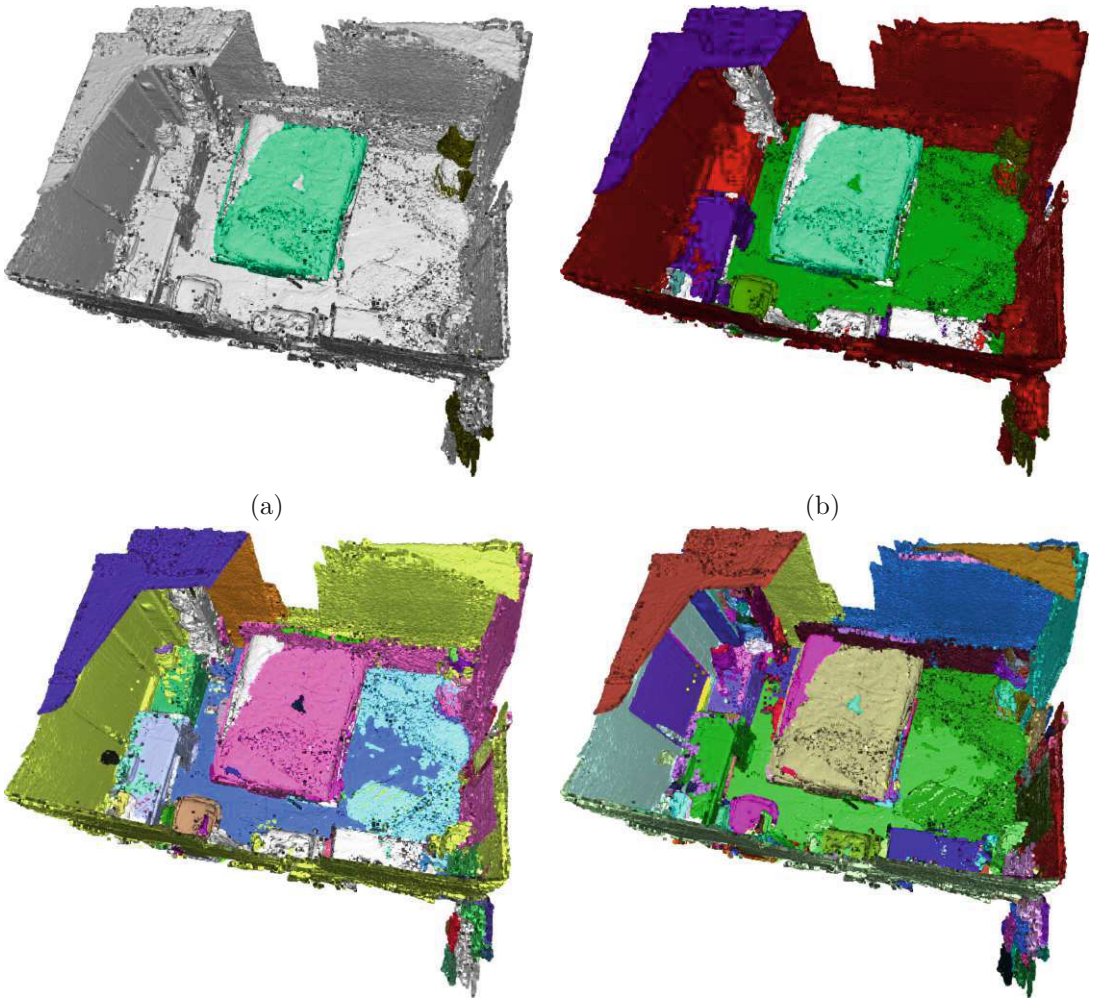


Figure 4.23: Combined confusion matrices after retraining with an extended learning schedule and increased frame rate for scene test sets 1 to 4, in order. Rows and columns with all zeros are removed for brevity.

(a) (b) (c) (d)

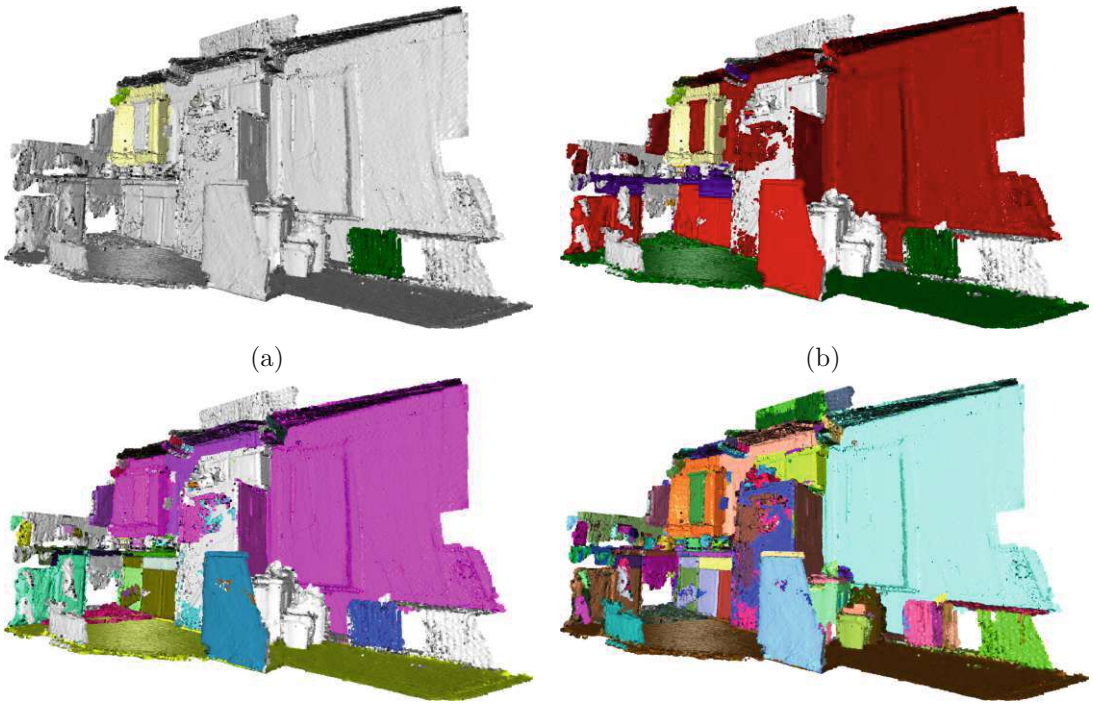
Figure 4.24: Scene 223, reconstructed with extended retraining and colored with different sets of labels. Subplots show, in order: newly detected categories, all class labels, instance labels, and segment labels. Best viewed with magnification.



(a) (b)

(c) (d)

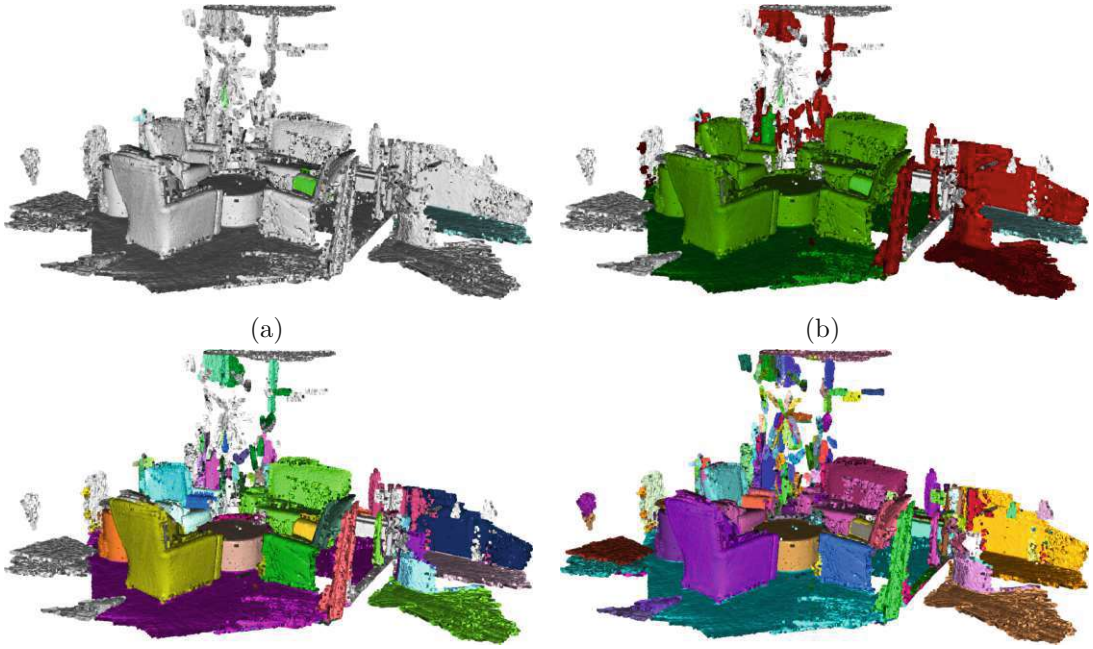
Figure 4.25: Scene 234, reconstructed with extended retraining and colored with different sets of labels. Subplots show, in order: newly detected categories, all class labels, instance labels, and segment labels. Best viewed with magnification.



(a) (b)

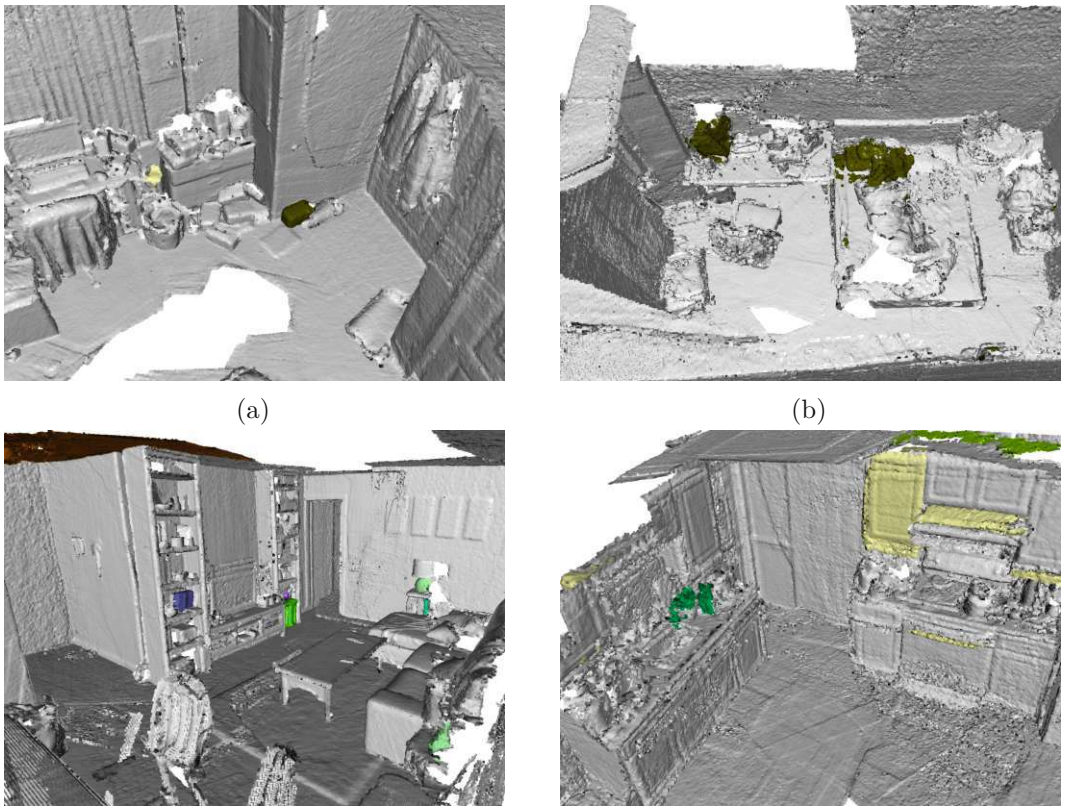
(c) (d)

Figure 4.26: Scene 255, reconstructed with extended retraining and colored with different sets of labels. Subplots show, in order: newly detected categories, all class labels, instance labels, and segment labels. Best viewed with magnification.



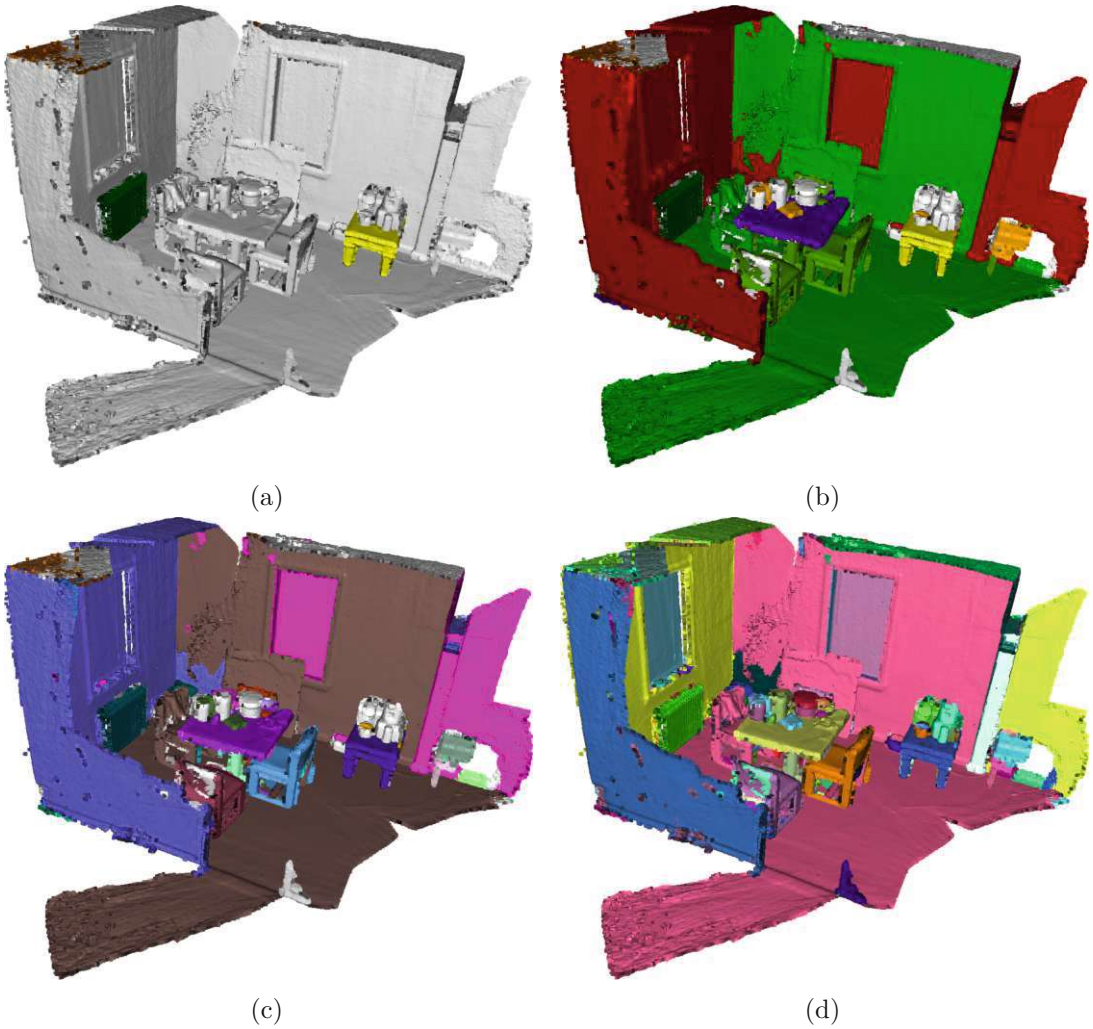
(a) (b) (c) (d)

Figure 4.27: Scene 294, reconstructed with extended retraining and colored with different sets of labels. Subplots show, in order: newly detected categories, all class labels, instance labels, and segment labels. Best viewed with magnification.



(a) (b) (c) (d)

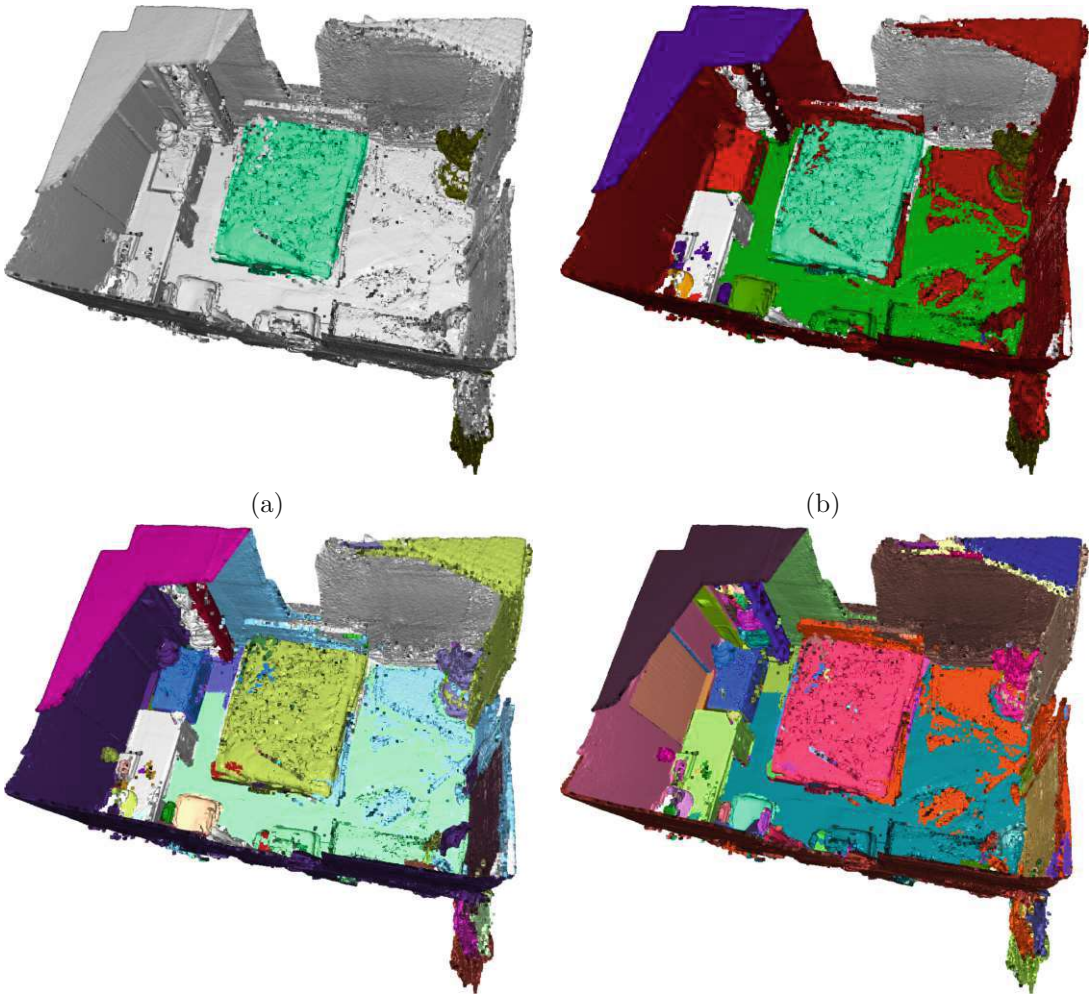
Figure 4.28: Examples of additional categories discovered in scenes reconstructed with extended retraining. Base classes are unlabeled in this visualization. Scenes shown are 093, 260, 272, and 276, in order. Best viewed with magnification.



(a) (b)

(c) (d)

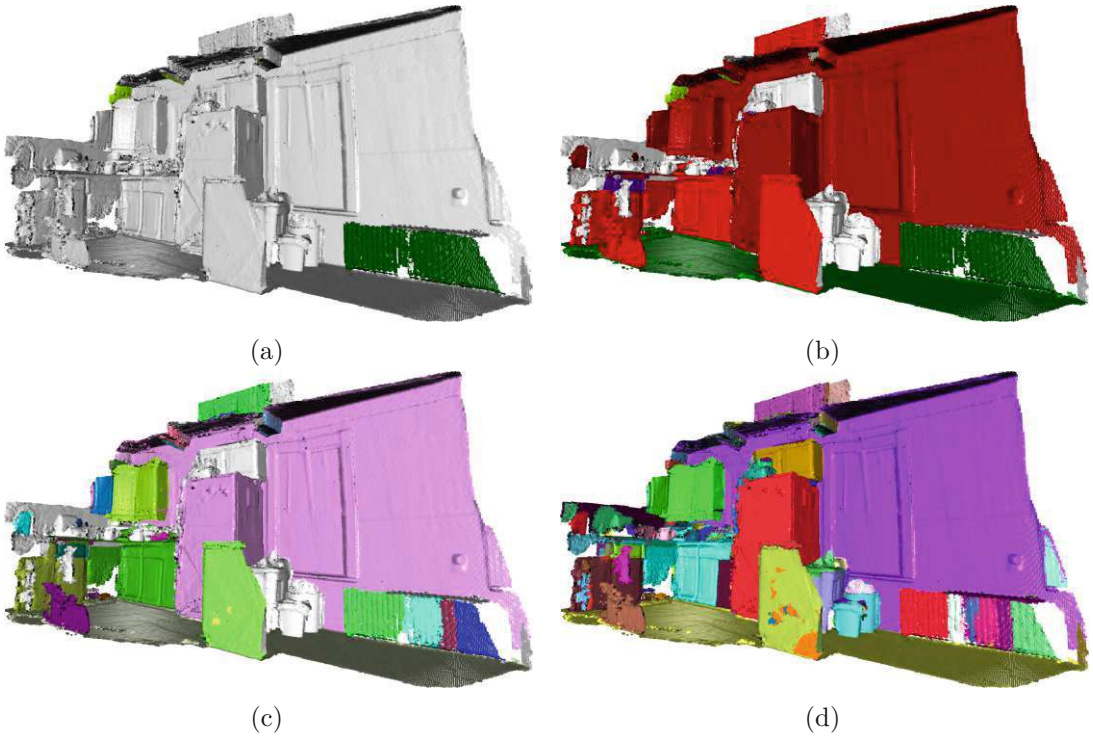
Figure 4.29: Scene 223, reconstructed with extended retraining and higher frame rates, and colored with different sets of labels. Subplots show, in order: newly detected categories, all class labels, instance labels, and segment labels. Best viewed with magnification.



(a) (b)

(c) (d)

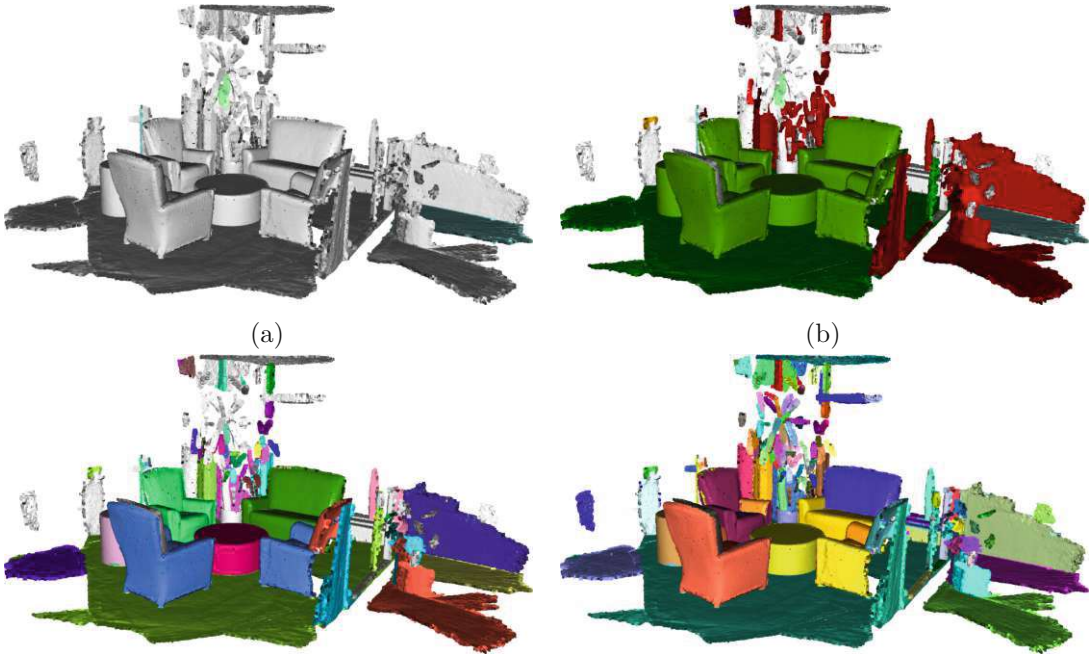
Figure 4.30: Scene 234, reconstructed with extended retraining and higher frame rates, and colored with different sets of labels. Subplots show, in order: newly detected categories, all class labels, instance labels, and segment labels. Best viewed with magnification.



(a) (b)

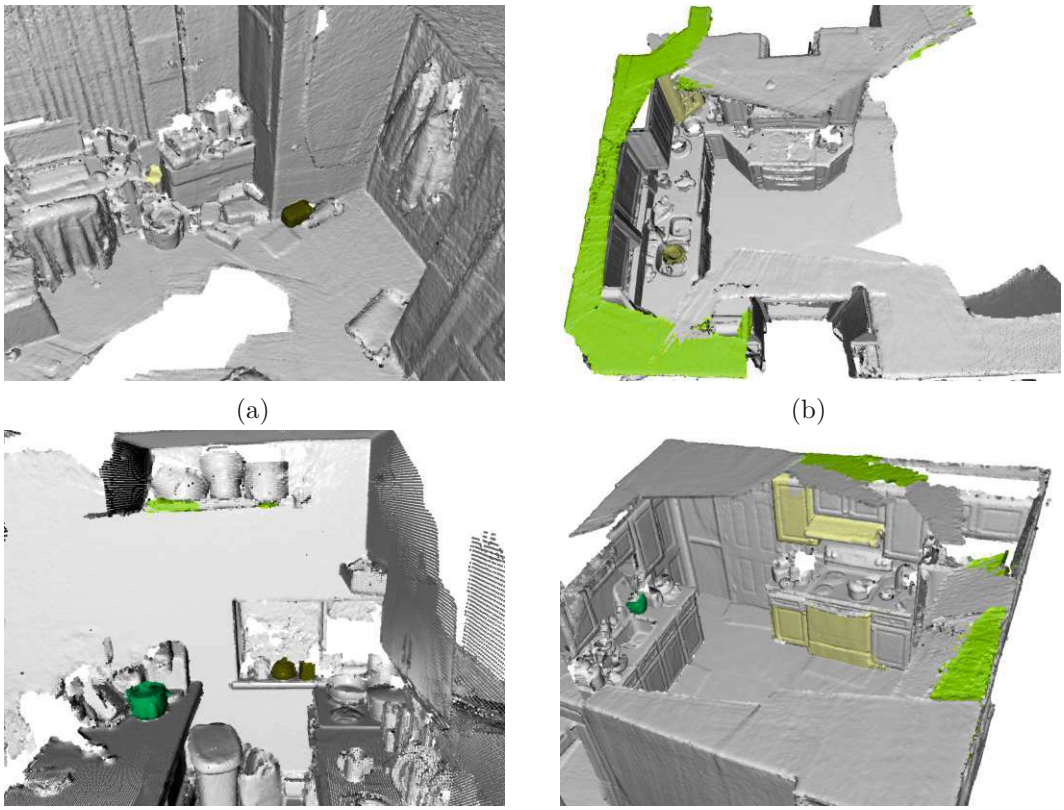
(c) (d)

Figure 4.31: Scene 255, reconstructed with extended retraining and higher frame rates, and colored with different sets of labels. Subplots show, in order: newly detected categories, all class labels, instance labels, and segment labels. Best viewed with magnification.



(a) (b) (c) (d)

Figure 4.32: Scene 294, reconstructed with extended retraining and higher frame rates, and colored with different sets of labels. Subplots show, in order: newly detected categories, all class labels, instance labels, and segment labels. Best viewed with magnification.



(a) (b) (c) (d)

Figure 4.33: Examples of additional categories discovered in scenes reconstructed with extended retraining and higher frame rates. Base classes are unlabeled in this visualization. Scenes shown are 093, 260, 270, and 276, in order. Best viewed with magnification.

CHAPTER 5

Conclusion

In this thesis, we presented a framework and method for unsupervised class-incremental learning in object-centric mapping. This work is intended to extend the capabilities of mobile autonomous robots in application areas related to object detection and instance-aware semantic mapping.

To the best of our knowledge, we are the first to tackle the problem of unsupervised class-incremental learning in the context of object-centric mapping in an integrated fashion. Our framework is designed to extract novel segments from scene recordings, form new categories from this data, and retrain the segmentation model to detect the novel categories in future observations, all without human intervention.

Our contributions are as follows:

- we propose, implement, test, and evaluate a framework for unsupervised incremental learning in open-world object-centric mapping, and
- we adapt the method of category discovery from 2D segmented images to persistent segmentations in 3D global maps, using per-view feature vectors from segmentation feature maps and constrained clustering.

We have shown, through theoretical considerations and practical experiments, that our framework and method is able to detect novel objects, create meaningful new categories, and learn to recognize objects from these categories in new recordings. Experiments have demonstrated that, given enough data, the framework is capable of grouping multiple similar objects into categories within and across different scenes.

These categories can be learned incrementally and used to detect included objects in later scene reconstructions. Furthermore, results indicate that the system is able to recognize

novel objects it has not seen before, based on categories introduced through incremental learning.

All components in this framework are automated and do not require any supervision for mapping, novelty detection, category discovery, or incremental learning. Training and evaluation is performed on a real-world dataset consisting of 47 scene recordings with raw RGB-D footage and semantic instance annotations on ground truth meshes.

This thesis initially fine-tunes the instance segmentation model on training data from a set of scenes in the work place category, and executes the framework on four different scene sets from other domains, including kitchens, bed rooms, and living rooms. Evaluations on these scenes indicate that, in principle, knowledge from instance-aware semantic annotations transfers to novel environments.

As the development of our framework represents early research, there are several shortcomings that we were not able to fully address in the course of this work. First of all, and probably most importantly, the training data is limited, both in variety and in quantity, and causes the initial model to fall short of state-of-the-art performance in instance segmentation.

This not only limits the semantic mapping aspect of the framework, but also causes issues further down the line, in particular with novelty detection where segments of known classes are detected as novelties. Retraining the segmentation models on novel categories that overlap with old classes might accelerate forgetting of these old classes and negatively affect segmentation performance.

A natural solution to this problem would be to use more extensive, higher quality data for fine-tuning the initial model. Using datasets with no misalignment between RGB and depth images could improve segmentation performance for initial fine-tuning as well as incremental learning. Inaccurate segmentation masks from geometric segmentation also affect feature extraction and thus have a potentially negative effect on clustering in the category discovery component.

Conceptually, the tracking of segments as implemented requires recording all mappings from local to persistent segment and instance labels, and all changes including segment merges. This approach is potentially error-prone, e.g. if a mapping or merge is missed by the novelty detection node due to synchronization issues.

Since we need to match per-frame segment labels to global labels for each input image, it is not possible to simply read global labels from the reconstruction in the global map at the end of each recording session. However, integrating segment tracking and novelty detection into the global segmentation map could provide a more robust solution.

In its current formulation, the framework cannot detect boundaries of unknown objects, and groups all segments of the same pseudo-label into one instance. This is due to the nature of the Mask R-CNN network used for instance segmentation, which does not provide bounding box detections or segmentation masks for unknown object instances.

5.1 Future Work

For category discovery, an adapted cluster selection that eliminates clusters resembling existing classes could be useful, in addition to more accurate fine-tuning on the initial class set. This could prevent new categories in incremental learning from interfering with the knowledge of existing classes, and increase overall segmentation performance.

As it is, the novelty detection and category discovery component assumes all unlabeled segments belong to novel objects, and labeled segments to known classes. However, this assumption does not always hold, and it might be beneficial to employ prediction quality estimates as part of novelty detection, as for example proposed by Rottmann et al. [RCH+20].

In order to more accurately detect unknown objects, and prevent over-segmentation of unknown object instances, combining the framework with another object detection model would be an interesting topic for future work. In particular, a network for object detection without classification such as OLN-Mask by Kim et al. [KLA+22] could prove useful for semantic refinement and novelty detection in the context of object-centric mapping. Such an approach would however require the application of an incremental learning method to this network in order to use it within the framework.

Replacing the backbone of Mask R-CNN with another network, for example FPN, might also be beneficial. Such a network could provide higher-resolution feature maps, and thus more accurately localized feature vectors, which might achieve a better distinction between adjacent segments in a scene.

Alleviating the strictly unsupervised nature of the approach, and incorporating human guidance and feedback, would allow the framework to be extended towards a system capable of human-robot interaction. This could have several useful implications, for example guiding the learning process towards objects of particular interest to the user, and correcting misclassifications that a robot has made in the past [KX22].

As the implementation has not been optimized for speed and high throughput, the frame rate is on the lower end of that for robotic applications. As the ablation study has shown, it would be worthwhile to optimize the implementation, either to improve the segmentation and reconstruction quality, or to run the framework on mobile platforms with limited computational resources.

Lastly, in its current form, the application of our method does not take into account an active role of mobile autonomous robots in their exploration of the environment, although it is briefly hinted at in the system design overview. A robotic system could steer the scene recording towards unknown objects, based on the knowledge contained in the global object-centric map, and thus create higher-quality scene recordings for novelty detection and category discovery [ZSG+18]. On the other hand, active or interactive exploration could be helpful in correctly classifying objects that might be obstructed or otherwise difficult to recognize [ZZZ+19].

List of Figures

List of Tables

Bibliography

[AHG+19] Iro Armeni, Zhi-Yang He, JunYoung Gwak, Amir R Zamir, Martin Fischer, Jitendra Malik, and Silvio Savarese. 3D scene graph: A structure for unified semantics, 3D space, and camera. In *IEEE/CVF International Conference on Computer Vision*, pages 5664–5673, 2019.

[AKC+22] Hamed Ayoobi, Hamidreza Mohades Kasaei, Ming Cao, Rineke Verbrugge, and Bart Verheij. Local-HDP: Interactive open-ended 3D object category recognition in real-time robotic scenarios. *Robotics and Autonomous Systems*, 147:103911, 2022.

[BCC+20] Dhruv Batra, Angel X Chang, Sonia Chernova, Andrew J Davison, Jia Deng, Vladlen Koltun, Sergey Levine, Jitendra Malik, Igor Mordatch, Roozbeh Mottaghi, et al. Rearrangement: A challenge for embodied AI. *arXiv e-prints*, pages arXiv–2011, 2020.

[BGT+20] Mohammed Bennamoun, Yulan Guo, Federico Tombari, Kamal Youcef-Toumi, and Ko Nishino. Guest editors' introduction to the special issue on RGB-D vision: Methods and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42:2329–2332, 10 2020.

[CGFC22] Fabio Cermelli, Antonino Geraci, Dario Fontanel, and Barbara Caputo. Modeling missing annotations for incremental learning in object detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3700–3710. IEEE, 2022.

[CYC+21] Jun Cen, Peng Yun, Junhao Cai, Michael Yu Wang, and Ming Liu. Deep metric learning for open world semantic segmentation. In *IEEE/CVF International Conference on Computer Vision*, pages 15333–15342, October 2021.

[CZK15] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust reconstruction of indoor scenes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5556–5565, 2015.

[Dav18] Andrew J. Davison. FutureMapping: The computational structure of spatial AI systems. *CoRR*, abs/1803.11288, 2018.

[Dav21] Achal Dave. *Open-world Object Detection and Tracking*. PhD thesis, Carnegie Mellon University, 2021.

[DCS+17] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017.

[dlPBE+14] Paloma de la Puente, Markus Bajones, Peter Einramhof, Daniel Wolf, David Fischinger, and Markus Vincze. RGB-D sensor setup for multiple tasks of home robots and experimental results. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2587–2594. IEEE, 2014.

[Dom12] Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.

[DWGL22] Xuefeng Du, Xin Wang, Gabriel Gozum, and Yixuan Li. Unknown-aware object detection: Learning what you don't know from videos in the wild. *arXiv e-prints*, pages arXiv–2203, 2022.

[EKX+96] Martin Ester, Hans-Peter Kriegel, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Second International Conference on Knowledge Discovery and Data Mining*. AAAI Press, 1996.

[FNF+18] Fadri Furrer, Tonci Novkovic, Marius Fehr, Abel Gawel, Margarita Grinvald, Torsten Sattler, Roland Siegwart, and Juan Nieto. Incremental object database: Building 3D models from multiple partial observations. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 6835–6842. IEEE, 2018.

[GDG+17] Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: training ImageNet in 1 hour. *CoRR*, abs/1706.02677, 2017.

[GFN+19] Margarita Grinvald, Fadri Furrer, Tonci Novkovic, Jen Jen Chung, Cesar Cadena, Roland Siegwart, and Juan Nieto. Volumetric instance-aware semantic mapping and 3D object discovery. *IEEE Robotics and Automation Letters*, 4(3):3037–3044, 2019.

[GNJ+22] Akshita Gupta, Sanath Narayan, KJ Joseph, Salman Khan, Fahad Shahbaz Khan, and Mubarak Shah. OW-DETR: Open-world detection transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9235–9244. IEEE, 2022.

[GTSN21] Margarita Grinvald, Federico Tombari, Roland Siegwart, and Juan Nieto. TSDF++: A multi-object formulation for dynamic object tracking and reconstruction. In *IEEE International Conference on Robotics and Automation*, pages 14192–14198. IEEE, 2021.

[HCC22] Nathan Hughes, Yun Chang, and Luca Carlone. Hydra: A real-time spatial perception engine for 3D scene graph construction and optimization. *arXiv e-prints*, pages arXiv–2201, 2022.

[HDN19] Ji Hou, Angela Dai, and Matthias Nießner. 3D-SIS: 3D semantic instance segmentation of RGB-D scans. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4421–4430. IEEE, 2019.

[HGD18] Kaiming He, Ross B. Girshick, and Piotr Dollár. Rethinking ImageNet pre-training. *CoRR*, abs/1811.08883, 2018.

[HGDG17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision*, pages 2961–2969, 2017.

[HPN+16] Binh-Son Hua, Quang-Hieu Pham, Duc Thanh Nguyen, Minh-Khoi Tran, Lap-Fai Yu, and Sai-Kit Yeung. SceneNN: A scene meshes dataset with annotations. In *Fourth International Conference on 3D Vision*, pages 92–101. Ieee, 2016.

[HTY18] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Pointwise convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[HZ21] Jiangpeng He and Fengqing Zhu. Unsupervised continual learning via pseudo labels. *arXiv e-prints*, pages arXiv–2104, 2021.

[HZJ+21] Wenbo Hu, Hengshuang Zhao, Li Jiang, Jiaya Jia, and Tien-Tsin Wong. Bidirectional projection network for cross dimension scene understanding. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14373–14382. IEEE, June 2021.

[JKKB21] K J Joseph, Salman Khan, Fahad Shahbaz Khan, and Vineeth N Balasubramanian. Towards open world object detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5830–5840. IEEE, 2021.

[Jos07] Rajive Joshi. Data-oriented architecture: A loosely-coupled real-time SOA. Technical report, Realtime Innovations, Inc., 2007.

[JRK+21] K J Joseph, Jathushan Rajasegaran, Salman Khan, Fahad Shahbaz Khan, and Vineeth N Balasubramanian. Incremental object detection via meta-learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[KHG+18] Alexander Kirillov, Kaiming He, Ross B. Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. *CoRR*, abs/1801.00868, 2018.

[KLA+22] Dahun Kim, Tsung-Yi Lin, Anelia Angelova, In So Kweon, and Weicheng Kuo. Learning open-world object proposals without learning to classify. *IEEE Robotics and Automation Letters*, 7(2):5453–5460, 2022.

[KLT20] Seyed Hamidreza Mohades Kasaei, Luís Seabra Lopes, and Ana Maria Tomé. Local-LDA: Open-ended learning of latent topics for 3D object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42:2567–2580, 2020.

[KMFF13] Andrej Karpathy, Stephen Miller, and Li Fei-Fei. Object discovery in 3D scenes via shape analysis. In *IEEE International Conference on Robotics and Automation*, pages 2088–2095. IEEE, 2013.

[KMvB+20] S. Hamidreza Kasaei, Jorik Melsen, Floris van Beers, Christiaan Steenkist, and Klemen Voncina. The state of service robots: Current bottlenecks in object perception and manipulation. *CoRR*, abs/2003.08151, 2020.

[KPSK19] Ue-Hwan Kim, Jin-Man Park, Taek-Jin Song, and Jong-Hwan Kim. 3-D scene graph: A sparse and semantic representation of physical environments for intelligent agents. *IEEE Transactions on Cybernetics*, 50(12):4921–4933, 2019.

[KX22] Hamidreza Kasaei and Songsong Xiong. Lifelong ensemble learning based on multiple representations for few-shot object recognition. *arXiv e-prints*, pages arXiv–2205, 2022.

[LC22] Xinye Li and Ding Chen. A survey on deep learning-based panoptic segmentation. *Digital Signal Processing*, 120:103283, 2022.

[LM19] Mathieu Labbé and François Michaud. RTAB-Map as an open-source LiDAR and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics*, 36(2):416–446, 2019.

[LMB+14] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[LPV20] Edith Langer, Timothy Patten, and Markus Vincze. Robust and efficient object change detection by combining global semantic information and local geometric verification. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 8453–8460. IEEE, 2020.

[LPV22] Edith Langer, Timothy Patten, and Markus Vincze. Where does it belong? Autonomous object mapping in open-world settings. *Frontiers in Robotics and AI*, 9, 2022.

[LZZ21] Yang Li, Hong Zhang, and Yu Zhang. Rethinking training from scratch for object detection. *arXiv e-prints*, pages arXiv–2106, 2021.

[MCB+18] John McCormac, Ronald Clark, Michael Bloesch, Andrew Davison, and Stefan Leutenegger. Fusion++: Volumetric object-level SLAM. In *International Conference on 3D Vision*, pages 32–41. IEEE, 2018.

[MHDL16] John McCormac, Ankur Handa, Andrew J. Davison, and Stefan Leutenegger. SemanticFusion: Dense 3D semantic mapping with convolutional neural networks. *CoRR*, abs/1609.05130, 2016.

[MZ19] Umberto Michieli and Pietro Zanuttigh. Incremental learning techniques for semantic segmentation. In *IEEE/CVF International Conference on Computer Vision Workshop*, pages 3205–3212. IEEE, 2019.

[Nak20] Yoshikatsu Nakajima. *New Class Discovery Based on Efficient Fusion of Semantics and Geometry for Incremental 3D Scene Understanding*. PhD thesis, Graduate School of Science and Technology, Keio University, 2020.

[NKSK19] Yoshikatsu Nakajima, Byeongkeun Kang, Hideo Saito, and Kris Kitani. Incremental class discovery for semantic segmentation with RGBD sensing. In *IEEE/CVF International Conference on Computer Vision*, pages 972–981, 2019.

[NS18] Yoshikatsu Nakajima and Hideo Saito. Efficient object-oriented semantic mapping with object detector. *IEEE Access*, 7:3206–3213, 2018.

[NTTS18] Yoshikatsu Nakajima, Keisuke Tateno, Federico Tombari, and Hideo Saito. Fast and accurate semantic mapping through geometric-based incremental segmentation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 385–392. IEEE, 10 2018.

[ORF20] Philipp Oberdiek, Matthias Rottmann, and Gernot A Fink. Detection and retrieval of out-of-distribution objects in semantic segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 328–329, 2020.

[OTF+17] Helen Oleynikova, Zachary Taylor, Marius Fehr, Roland Siegwart, and Juan Nieto. Voxblox: Incremental 3D euclidean signed distance fields for on-board MAV planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1366–1373. IEEE, 2017.

[OTSN18] Helen Oleynikova, Zachary Taylor, Roland Siegwart, and Juan Nieto. Sparse 3D topological graphs for micro-aerial vehicle planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1–9. IEEE, 2018.

[PDC+18] Trung Pham, Thanh-Toan Do, Gustavo Carneiro, Ian Reid, et al. Bayesian semantic instance segmentation in open set world. In *European Conference on Computer Vision*, pages 3–18, 2018.

[PHNY19] Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Real-time progressive 3D semantic segmentation for indoor scenes. In *IEEE Winter Conference on Applications of Computer Vision*, pages 1089–1098. IEEE, 2019.

[PJY+21] Jin-Man Park, Jae-Hyuk Jang, Sahng-Min Yoo, Sun-Kyung Lee, Ue-Hwan Kim, and Jong-Hwan Kim. ChangeSim: towards end-to-end online scene change detection in industrial indoor environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 8578–8585. IEEE, 2021.

[PKP+19] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

[PZVBG00] Hanspeter Pfister, Matthias Zwicker, Jeroen Van Baar, and Markus Gross. Surfels: Surface elements as rendering primitives. In *Annual Conference on Computer Graphics and Interactive Techniques*, pages 335–342, 2000.

[QRX+21] Haoxuan Qu, Hossein Rahmani, Li Xu, Bryan Williams, and Jun Liu. Recent advances of continual learning in computer vision: An overview. *arXiv e-prints*, pages arXiv–2109, 2021.

[RACC20] Antoni Rosinol, Marcus Abate, Yun Chang, and Luca Carlone. Kimera: an open-source library for real-time metric-semantic localization and mapping. In *IEEE International Conference on Robotics and Automation*, pages 1689–1696. IEEE, 2020.

[RCH+20] Matthias Rottmann, Pascal Colling, Thomas Paul Hack, Robin Chan, Fabian Hüger, Peter Schlicht, and Hanno Gottschalk. Prediction error meta classification in semantic segmentation: Detection via aggregated dispersion measures of softmax probabilities. In *International Joint Conference on Neural Networks*, pages 1–9. IEEE, 2020.

[Rou87] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

[RVA+21] Antoni Rosinol, Andrew Violette, Marcus Abate, Nathan Hughes, Yun Chang, Jingnan Shi, Arjun Gupta, and Luca Carlone. Kimera: From SLAM to spatial perception with 3D dynamic scene graphs. *The International Journal of Robotics Research*, 40(12-14):1510–1546, 2021.

[SDS+21] Lukas Schmid, Jeffrey Delmerico, Johannes Schönberger, Juan Nieto, Marc Pollefeys, Roland Siegwart, and Cesar Cadena. Panoptic Multi-TSDFs: a flexible representation for online multi-resolution volumetric mapping and long-term dynamic scene consistency. *arXiv e-prints*, pages arXiv–2109, 2021.

[Spe07] OMG Available Specification. Data Distribution Service for real-time systems version 1.2. *Manual of Object Management Group*, 1, 2007.

[SSA17] Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. Incremental learning of object detectors without catastrophic forgetting. *CoRR*, abs/1708.06977, 2017.

[SSP19] Thomas Schöps, Torsten Sattler, and Marc Pollefeys. SurfelMeshing: Online surfel-based mesh reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP:1–1, 10 2019.

[TDCH21] Michal Tölgyessy, Martin Dekan, L'uboš Chovanec, and Peter Hubinskỳ. Evaluation of the Azure Kinect and its comparison to Kinect V1 and Kinect V2. *Sensors*, 21(2):413, 2021.

[TH01] Jeanie Treichel and Mary Holzer. Sun Microsystems Laboratories: The first ten years. *Perspectives*, page 5, 2001.

[TTN15] Keisuke Tateno, Federico Tombari, and Nassir Navab. Real-time and scalable incremental segmentation on dense SLAM. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4465–4472. IEEE, 2015.

[TTN16] Keisuke Tateno, Federico Tombari, and Nassir Navab. Large scale and long standing simultaneous reconstruction and segmentation. *Computer Vision and Image Understanding*, 157, 05 2016.

[URG22] Svenja Uhlemeyer, Matthias Rottmann, and Hanno Gottschalk. Towards unsupervised open world semantic segmentation. *CoRR*, abs/2201.01073, 2022.

[WDNT20] Johanna Wald, Helisa Dhamo, Nassir Navab, and Federico Tombari. Learning 3D semantic scene graphs from 3D indoor reconstructions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3960–3969. IEEE, 06 2020.

[WLY+21] Yingcai Wan, Yanyan Li, Yingxuan You, Cheng Guo, Lijin Fang, and Federico Tombari. Semantic dense reconstruction with consistent scene segments. *arXiv e-prints*, pages arXiv–2109, 2021.

[WNT22] Johanna Wald, Nassir Navab, and Federico Tombari. Learning 3D semantic scene graphs with instance embeddings. *International Journal of Computer Vision*, 130:1–22, 03 2022.

[WSMG+16] Thomas Whelan, Renato F Salas-Moreno, Ben Glocker, Andrew J Davison, and Stefan Leutenegger. ElasticFusion: Real-time dense SLAM and light source estimation. *The International Journal of Robotics Research*, 35(14):1697–1716, 2016.

[WTS+18] Johanna Wald, Keisuke Tateno, Jürgen Sturm, Nassir Navab, and Federico Tombari. Real-time fully incremental scene understanding on mobile platforms. *IEEE Robotics and Automation Letters*, 3(4):3402–3409, 2018.

[WWT+21a] Ádám Wolf, David Wolton, Josef Trapl, Julien Janda, Stefan Romeder-Finger, Thomas Gatternig, Jean-Baptiste Farcet, Péter Galambos, and Károly Széll. Towards robotic laboratory automation plug & play: The "LAPP" framework. *SLAS Technology*, 2021.

[WWT+21b] Shun-Cheng Wu, Johanna Wald, Keisuke Tateno, Nassir Navab, and Federico Tombari. SceneGraphFusion: Incremental 3D scene graph prediction from RGB-D sequences. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7515–7525. IEEE, 2021.

[WWWK96] Jim Waldo, Geoff Wyant, Ann Wollrath, and Sam Kendall. A note on distributed computing. In *International Workshop on Mobile Object Systems*, pages 49–64. Springer, 1996.

[YPRL22] Li Yin, Juan M Perez-Rua, and Kevin J Liang. Sylph: A hypernetwork framework for incremental few-shot object detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9035–9045. IEEE, 2022.

[ZLH+22] Jiyang Zheng, Weihao Li, Jie Hong, Lars Petersson, and Nick Barnes. Towards open-set object detection and discovery. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3961–3970. IEEE, 2022.

[ZLS+22] Xiaowei Zhao, Xianglong Liu, Yifan Shen, Yuqing Ma, Yixuan Qiao, and Duorui Wang. Revisiting open world object detection. *arXiv e-prints*, pages arXiv–2201, 2022.

[ZPK18] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.

[ZSG+18] Michael Zollhöfer, Patrick Stotko, Andreas Görlitz, Christian Theobalt, Matthias Nießner, Reinhard Klein, and Andreas Kolb. State of the art on 3D reconstruction with RGB-D cameras. In *Computer Graphics Forum*, volume 37, pages 625–652. Wiley Online Library, 2018.

[ZZZ+19] Lintao Zheng, Chenyang Zhu, Jiazhao Zhang, Hang Zhao, Hui Huang, Matthias Niessner, and Kai Xu. Active scene understanding via online semantic reconstruction. In *Computer Graphics Forum*, volume 38, pages 103–114. Wiley Online Library, 2019.