Dissertation

3D Scene Understanding of Static and Dynamic Indoor Environments

Johanna Wald

Technische Universität München
Fakultät für Informatik

3D Scene Understanding of Static and Dynamic Indoor Environments

Johanna Wald

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung einer

Doktorin der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzende(r):	Prof. Dr.-Ing. Darius Burschka
Prüfer der Dissertation:	1. Priv.-Doz. Dr. Federico Tombari
	2. Prof. Dr. Angela Dai
	3. Prof. Dr. Lourdes Agapito

Die Dissertation wurde am 23.06.2021 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 17.11.2021 angenommen.

Johanna Wald
3D Scene Understanding of Static and Dynamic Indoor Environments
Dissertation, Version 1.0

Technische Universität München
Fakultät für Informatik
Lehrstuhl für Informatikanwendungen in der Medizin
Boltzmannstraße 3
85748 Garching bei München

Abstract

In modern society, the majority of people spend most of their lives indoors in artificial environments. Visual perception of these indoor scenes is one of our most important abilities and inherently natural for humans, but it is challenging for computers. 3D scene understanding, the process of scanning and parsing a scene into a higher-level representation, enables next-level technological advances and applications that autonomously function in our complex world for AR/VR, indoor robotics, self-driving cars, or drones. Truly understanding a scene means perceiving and analyzing its entities' geometric and semantic relationships and their intrinsic and extrinsic nature. It requires sophisticated spatial and semantic reasoning: segmenting objects and structures, parsing patterns, colors, and complex geometry, and, importantly, putting them into temporal context.

This thesis claims methodological and data contributions targeting 3D scene understanding research beyond static setups considering scene dynamics and changes. To do so, we propose the first large-scale real-world RGB-D indoor dataset *RIO* of naturally changing indoor scenes. We then present *3DSSG* that augments the scenes in *RIO* with 3D scene graphs, a semantically rich and compact data representation. Scene graphs encode object entities and their semantics and are a promising representation to handle scene persistence. Understanding dynamics requires the extraction of rich semantics. Therefore, we propose 3D semantic segmentation and instance segmentation methods and transfer well-known 2D tasks such as scene graph prediction to 3D. To this end, we propose the first learned method that generates scene graphs from 3D point clouds; we then suggest an improved scene graph prediction method that jointly learns instance embeddings and semantics.

Finally, this dissertation investigates new, relatively unexplored research areas handling changes in 3D scenes. We propose new benchmarks and tasks for scene, camera as well as object re-localization. We show how scene graphs are applicable in a domain agnostic scene retrieval task and then describe *RIO10*, an indoor camera re-localization benchmark. We propose methods to quantify scene changes, based on the geometric and semantic information provided in *RIO*, to ultimately understand their impact on camera re-localization performance. Finally, a learned method for object instance re-localization is proposed where 6DoF transformations for objects of an RGB-D scan are estimated aligning them with a reference scan, taken at a different point in time.

While there is still a lot to be explored in long-term 3D scene understanding, the contributions in this dissertations show that the sophisticated semantic parsing of static spaces paired with re-localization of scenes, cameras, and objects, is a promising direction. While we have explored various tasks in several different works, we believe that they could be processed jointly as they might benefit from each other.

Zusammenfassung

In der modernen Gesellschaft verbringt ein Großteil der Menschen die meiste Zeit des Lebens in von Menschenhand geschaffenen Innenräumen. Während die visuelle Szenenwahrnehmung eine unserer wichtigsten Fähigkeiten ist, ist sie für Computer eine Herausforderung. Das Verständnis von 3D-Szenen beinhaltet das Scannen und Verarbeiten eines Raumes in eine übergeordnete Darstellung und ermöglicht technologische Fortschritte und neuartige Anwendungen, die autonom in unserer komplexen Welt agieren in Bereichen wie AR/VR, Indoor-Robotik, selbstfahrenden Autos oder Drohnen. Eine Szene wirklich zu verstehen bedeutet den geometrischen und semantischen Kontext ihres Inhalts und ihre intrinsischen Beziehungen wahrzunehmen und zu analysieren. Dies erfordert ein ausgeklügeltes räumliches und semantisches Verständnis: Segmentieren von Objekten und Strukturen, Analysieren von Mustern, Farben und komplexer Geometrie und vor allem das Einordnen dieser in einen zeitlichen Kontext.

In dieser Arbeit stellen wir Methoden und Daten vor, die eine 3D-Szenenverständnisforschung ermöglichen, die über Statik hinausgeht und stattdessen Szenen Dynamik und -änderungen berücksichtigt. Zu diesem Zweck beschreiben wir zunächst *RIO*, ein RGB-D Datensatz der natürlich ändernde, reale Innenraum Szenen enthält. Anschließend präsentieren wir *3DSSG*, eine Erweiterung von *RIO* um 3D-Szenengraphen, eine semantische, kompakte Datendarstellung die Objekte beschreibt und möglicherweise helfen könnte Persistenz zu modellieren. Da Verständnis von Dynamik auf Statik aufbaut; präsentieren wir zunächst Methoden zur semantischen 3D-Segmentierung und Instanz Segmentierung statischer Szenen, übertragen aber auch 2D-Aufgaben wie die Bestimmung von Szenengraphen in die 3D Domäne. Wir beschreiben die erste Methode, die Szenen-Graphen aus 3D-Punktwolken generiert. Anschließend stellen wir einen verbesserten Algorithmus zur Vorhersage von Szenen Diagrammen vor, der Instanz Vektoren und Semantik gemeinsam erlernt.

Schließlich erkundet diese Dissertationen neue, relativ unerforschte Bereiche, die sich mit 3D Szenen-Änderungen befassen. Dazu stellen wir neue Benchmarks und Aufgaben für die Szenen-, Kamera- und Objektlokalisierung vor. Wir zeigen, wie Szenengraphs in einer domänenunabhängigen Aufgabe genutzt werden können. Außerdem veröffentlichen wir *RIO10*, ein Lokalisierung Datensatz, bei dem wir Methoden zur Quantifizierung von Szenen Änderungen bereitstellen, die auf geometrischen und semantischen Daten basieren. Das ermöglicht die Analyse der Auswirkungen von Veränderungen auf die Genauigkeit verschiedener Methoden. Schließlich präsentieren wir ein Verfahren zur Lokalisierung von Objekten, bei dem 6DoF-Transformationen bestimmt werden, um ein Objekt mit einem Referenz Scan auszurichten, der zu einem anderen Zeitpunkt erfasst wurde.

Während es im Bereich des langfristigen 3D-Szenenverständnisses noch viel zu erforschen gibt, zeigen die Beiträge in diesen Dissertationen, dass eine ausführliche semantische Analyse statischer Szenen in Kombination mit Re-Lokalisierung von Räumen, Kameras bis hin zu Objekten eine gute Basis darstellen könnte. Während wir diese verschiedenen Richtungen in diversen Arbeiten erforscht haben, glauben wir dass sie gemeinsam in einer Methode kombinierbar sind und voneinander profitieren könnten.

Acknowledgments

When I first walked into TUM, I would have never expected it to be such a life-changing experience. The last few years were surely an emotional roller-coaster and, to be honest, writing these lines feel quite unreal. I'm so incredibly grateful for all the experiences and memories filled with excitement, a steep learning curve, a lot of personal growth, and most importantly, many incredibly intelligent, funny, simply remarkable people!

I will never forget the feeling of having to speak in front of thousands of people, the summits, summer school, and CV meetups. I feel so privileged to have visited so many exciting places and conferences during my Ph.D., feeling like an intruder at tech company-sponsored lunches, dinners, or parties, and having had the chance to meet and collaborate with some of my research role models. I will most definitely miss the BBQs and all the fun I had at CAMPing in Faak, the music sessions where we sang our souls out, and even thinking about deadline pizza at late night work sessions puts a huge smile on my face.

I'm so incredibly grateful for everyone who made it such an unforgettable experience. First and foremost, thanks to my immensely talented supervisor, Federico, for always supporting me, for believing in me, and giving me the freedom to follow my passion while supporting my ideas; you are truly an inspiration for me! Also, thanks to all other mentors and supervisors I met along the way and my committee, especially Prof. Angela Dai and Prof. Lourdes Agapito, for reviewing this thesis. A special "*Thank you*" also goes to Prof. Nassir Navab for all his wisdom and making CAMP such a heartwarming place, and of course "*Danke*" to Martina for holding it all up together. Thanks to *ZD.B./bidt* for building such a great network and giving me flexibility and support at every stage of my Ph.D. journey. Thanks to all the CAMP members for creating such a supportive environment, especially to the CV team for all the exciting discussions and valuable feedback. To all my co-authors Nassir, Jürgen, Keisuke, Armen, Dario, Evin, Helisa, Matthias, Shristi, Shun-Cheng, Stuart, Tommaso, Torsten and Yida, I really enjoyed working with you: "*You're awesome!*". Also, "Kudos" to Fabian, Hendrik, Christoph, Vera, Jack, Mira, Magda, Mahdi, David, Huseyin, Tolga, Niko, YanYan and numerous others who made my time at TUM so joyful.

Besonderer Dank geht zu guter Letzt an meine Familie, besonders an meine Eltern, Emilia und Nikolaus, für Eure Unterstützung und Euer Vertrauen in mich und Danke an Dominik für Deinen kontinuierlichen Support, Dein Verständnis und all Deine Liebe, Ich bin unglaublich dankbar, dass ich mich immer auf Dich verlassen kann.

Contents

Part I

Introduction and Background

Introduction

1

1.1 Outline

This thesis is divided into three main parts; the first discusses datasets, while the third and fourth cover methods and applications. The dataset section introduces the benchmark and data contributions, while the method section focuses on extracting semantics and handling scene changes. This thesis is based on previous works that have been published or are under submission at major conferences and journals. While the content is significantly restructured and rephrased in the following sections, the main findings are taken from individual papers and are mentioned before the respective sections.

Part I explains the problem statement (see Chapter 1) and introduces scene understanding as a research topic (Sec. 1.2.1). Then, neural visual perception is defined from a cognitive perspective (Sec. 1.2.2). Finally, the tasks and challenges are introduced necessary to digitalize real-world scenes (Sec. 1.2.3) which leads to the summary of contributions of this work (Chapter 2).

Part II mainly focuses on the data contributions of this thesis. It gives an overview of some fundamental computer vision techniques and introduces most of the mathematical notations. We discuss benchmarks of changing indoor scenes (Chapter 3) and give some basic data documentation as a reference for anyone interested in using *RIO* (Sec. 3.2) and *RIO10* (Sec. 3.3). Later, in chapter 4, the related work and terminology of 3D semantic scene graphs are introduced. It again, also serves as a data documentation and reference for the *3DSSG* dataset (Sec. 4.2).

Part III gives insides about some of the proposed methods and applications mainly around scene semantics. It covers related work and our proposed approaches for 3D semantic segmentation (Sec. 5) and scene graph generation (Sec. 6). Specifically a method for real-time incremental semantic segmentation (Sec. 5.3), a scalable learned method for point cloud processing (Sec. 5.4) and several different scene graph prediction methods on point clouds (Sec. 6.3), jointly learning of graphs and instance embeddings (Sec. 6.4) and an incremental scene graph fusion method (Sec. 6.6) are discussed.

Part IV focuses on methods and applications targeting scene changes. It introduces a retrieval method to re-identify a scene using graphs (Chapter 7). Camera re-localization (Chapter 8) is discussed where multiple algorithms are tested and evaluated on a newly introduced long-term camera re-localization benchmark. Finally, object instance re-localization is introduced and solved with a learning-based method (Chapter 9).

Part V summarizes the proposed contributions in Sec. 10.1 and then in Sec. 10.2 discusses the limitations of the proposed works and suggests new directions for future research that builds upon the outcome and datasets.

1.2 Background

Humans can parse complicated visual scenes with fantastic speed and precision, grasping the substance of an environment with a single glance. While we experience scenes through all of our senses – compared to most animals – visual perception is one of the most important abilities of homo sapiens as we primarily rely on vision. Our incredibly powerful visual processing system develops at an early age by interacting, exploring, and discovering the unknown. By doing so, we obtain knowledge about our surroundings which over time ultimately translates to environmental awareness. Remarkably, this study of our surroundings by sensation and perception is a naturally learned and subconscious process.

No wonder researchers from different disciplines ranging from neuroscience to computer vision are fascinated by the human visual perception system's speed, integrity, and performance. It is intriguing how infants naturally learn and collect experiences to better understand the world around them. Re-identifying objects and people seem effortless and requires very little or even no supervision at all.

In the following chapter, we first give a high-level definition of scene understanding before diving into a psychological perspective, see Sec. 1.2.1. Sec. 1.2.2 explains how our surroundings are perceived and processed in the human brain. We then conclude by explaining the efforts to mimic this perception, listing the main principles of computer vision aiming to digitalize the real world, see Sec. 1.2.3.

1.2.1 Understanding Scene Understanding

In the following, we explain how our surroundings are identified and review some modern and historical views of scene understanding from the psychological literature. Biederman claims that acoustic and linguistics share characteristics with visual perception. Scenes, often also referred to as schemas in the literature [25], are the visual counterpart of sentences as they consist of a configuration or construct of meaningful visual information [25]. Epstein categorized real-world environments – and the artificial equivalent – into the foreground and background objects and compared them with contextless objects [70]. In contrast to a collection of unrelated objects, a scene is defined by meaningful relationships that give the composition higher-level significance. A scene, therefore, consists of two main elements: background (layout or structure) and foreground (objects or humans). In general, this segmentation is a subconscious process. It has been shown that infants learn to segment objects from the background by utilizing motion. Extracting objects by combining patterns on the image to form fore- and background has been studied in many different theories *e.g.* the Gestalt school of psychology.

While these definitions might be reasonable and intuitive, one might question the significance of scenes over objects. Scenes matter because they help us navigate the world; we re-identify places we have visited before [70] and almost immediately re-localize ourselves within them. Whenever we turn around, we subconsciously keep track of what kind of objects are behind us and simultaneously store and retrieve new information with a blink of an eye without even noticing. Past experiences help us navigate in environments that we have never visited before. We know that doors, stairs, and elevators connect us to new spaces and, therefore, associate them with transportation. Objects give us cues about what to find where, *e.g.* in a kitchen, which we recognize due to their unique appearances. Scene understanding is crucial for survival and is being helpful to humans even in modern days to *e.g.* find food, water, or restrooms. Humans can identify the functionality of furniture and items they had never seen before because they collected general knowledge through life-long experiences. *This is so fascinating, let's appreciate this for a bit!*

While outdoor scenes are incredibly diverse, they are – from a high-level perspective – strangely similar. This thesis focuses on indoor scenes instead, as nowadays, most people spend a significant amount of their life indoors in artificial environments. This sad truth makes indoor environments our natural habitat - or *how much time do you spend outdoors?*.

Indoor scenes are unique because of their variance. Structural components such as doors and windows are almost always present in indoor environments. We know that walls and floors are stable, and a considerable collection of indoor items and furniture is designed for human interaction. Indoor scenes are made by and for humans and often follow certain regularities, such as the so-called Manhattan world assumption [51]. This assumes that structures and furniture are oriented in a grid and that lines most likely meet at an orthogonal angle. Another unique constraint in indoor scenes is that they are, compared to outdoor scenes, relatively small and densely packed with items which regularly causes occlusion where objects largely hide parts of another object. Chairs are standing next to tables and are often hidden by them. Pillows are regularly found on couches or beds and cover some of their surfaces. Whenever we put an item on another piece of furniture, this creates occlusion.

As if this was not challenging enough, indoor objects frequently move around. While our world consists of 3D dimensional data, the temporal aspect is often considered the fourth dimension.

To understand a scene means to perceive and analyze the "*geometric and semantic context of its contents and the intrinsic relationships between them*". [198]. It involves segmenting objects and structures, parsing patterns and colors, analyzing complex geometry, shapes, and sizes, and putting this into temporal context. Understanding means to intrinsically and extrinsically be able to answer questions about the scene and its objects [177]. When discussing scene understanding, the literature often refers to semantics. Semantics helps to identify items and to categorize them into classes by *matching them with a vocabulary of known objects* [208]. We argue that identifying one object might help identify another; this is where context joins the game. According to Biderman *et al.* [25] parsing and identifying objects involves analyzing semantic relationships, and consequently, understanding a scene means understanding not only the items within but the bigger context they are in.

1.2.2 Neural Visual Perception

Neural visual perception is the capacity to perceive our environment through the light that enters our eyes. The following section gives an overview of some of the basic ideas and concepts from a cognitive science perspective. It has been studied for decades and draws from developments in neuroscience and psychology to better understand the processing of signals in our body [292].

It all starts with light being scattered on a surface. This light travels through our cornea and lens and ultimately hits the retina. The retina is cluttered with rods and cones which convert the light waves into electrical signals. These are then transferred to the brain through the optic canal via the optic nerve. From there, it passes the lateral geniculate nucleus in the thalamus to get to the visual cortex, which is a prominent part of the brain known for processing visual sensory information and located in the occipital lobe.

Since our brain receives a massive amount of data, it needs to draw attention to what matters to make sense of this stimulating mess of shapes and colorful patterns. To help us focus, the visual cortex extracts saliency maps to guide our attention. Such filtering is vital when looking at real-world scenes consisting of multiple parts and stands in contrast to isolated objects processed differently in our brain. The signals are then passed to the visual association cortex used for depth perception and foreground-background segmentation by parsing the images captured with our eyes.

The well-accepted *two-streams hypothesis* from Goodale and Milner [92] differentiates between two main transmission streams, namely the dorsal and the ventral stream. The dorsal, often referred to as the *where*-stream, is located between the primary visual cortex and the superior parietal lobule and is associated with spatial object recognition as well as estimating distances, motion, and orientation. The ventral stream, also known as the *what* pathway, transmits signals from the primary visual cortex along the ventral side of the brain to the temporal cortex and is related to the detection and shape of objects. This is where the visual signal meets the semantic and human language – *quite fascinating if you ask me*.

The visual cortex is located in the cerebral cortex, the largest region in the human brain. It is responsible for higher-level functionalities and controls our feelings and memory, and importantly, a large section is dedicated to visual processing. The exact biological processes in the brain are complex, and it is hard to measure and reproduce patterns of brain activity in an organ as complex as the human brain. And while perception involves sensing, selecting, and processing as just described, it also involves remembering and associating previous stimuli. Visual scene understanding is connected to abstract brain functions such as memorization and short and long-term memory to retrieve past experiences and knowledge when processing visual signals.

” *Our understanding is correlative to our perception*

— **Robert Delaunay**
(1885 - 1941)

Every one of us is involved with perception and understanding, and even if what we sense is identical, everyone perceives the world in their own way. Intriguingly, perception is a very individual and personal process that employs past experiences obtained from life-long learning. People unconsciously complete and derive content from things they either do *not* see or do not understand. Undoubtedly, visual perception develops as we collect new experiences and will surely remain an exciting research topic in the future.

1.2.3 Digitization of Real-World Scenes

While visual perception is inherently natural for humans, it is a challenge for computers. Computer vision, and more specifically, 3D scene understanding, enables sophisticated spatial and semantic reasoning and is the basis of many different applications from AR/VR [20, 40, 90, 173, 210, 224, 285] to indoor robotics and self-driving cars [35] or even drones [164]. It has been used for educational and entertainment purposes and finds applications in e-commerce, marketing, or product development. Higher-level scene knowledge is essential to enable next-level augmented reality experiences that realistically add virtual content to the real world. Furthermore, computer vision could potentially be part of personal assistants or even support visually impaired people. An autonomous robotic system requires full spatial and semantic awareness [58] ideally obtained in real-time for navigation and collision avoidance but also to perform sophisticated higher-level, potentially interactive actions such as grasping and manipulation.

It comes as no surprise that the foundation of computer vision lies in the simulation of the neural visual perception system, see Sec. 1.2.2. Some computational models try to mimic biological vision, but since large parts of this system are still unknown and inherently complex, it is often impossible to emulate [102]. A well-known but also very successful example is deep learning and convolutional neural networks, which are inspired by the neural connection in the brain and are usually trained with a large amount of labeled data.

Marr [178] described visual perception as the process of turning 2D input data into a 3D description of the world. He divides this process into several stages, starting with a) the extraction of low-level features such as edges and corners which is then converted into a b) 2.5D sketch of the scene with depth and textures. Finally, in the last stage c) a global 3D model is generated in a continuous 3-dimensional space. Interestingly, his top-down approach describes several classical image-based 3D computer vision approaches.

From a more modern perspective, digitizing real-world scenes can be split into two major components, perception and interpretation. The perception of the environment with sensors – 2D or 3D – is equivalent to our senses, while the interpretation and processing help us understanding the input given a collection of data samples and potential supervision. In the following, we explain the main principles of these two processes in more depth.

First and foremost, for a computer to read complex structures of our 3D world, it has to be turned into a numerical representation. To do so, data has to be captured with a sensor *e.g.* cameras that acquire images. They are easy to process with a machine since it is simply a grid of color values. Range images are an alternative and provide, additionally to the color channel, a distance measure to the surfaces at a certain 2D point. Computer vision algorithms process either single images, sequences, or videos, offline or on the fly. Offline methods optionally construct more advanced 3D representations such as unordered 3D point sets or ordered 3D grids, which strictly speaking also starts with images. While each input representation has different advantages and disadvantages, it is unclear what representation is best suited for different tasks.

Besides data representation, algorithms are needed to extract meaningful semantic knowledge from a given numerical representation. These usually recognize known objects but sometimes also humans or faces, scenes, and even complex actives. Historically, computer vision is task-oriented, providing a plethora of scene understanding algorithms. Even though it quickly gets overwhelming, many approaches try to solve a similar problem: extracting semantic information from a scene and the estimation of object properties therein.

To start with, turning sensory data into a meaningful representation contains capturing the 3D geometry of surfaces. This involves being able to handle occlusion and combining or fusing measurements into one comprehensive 3D model. Capturing these 3D models requires accurately estimating the location of the camera. In addition to capture 3D maps via 3D reconstruction systems [56, 202, 205, 303] researchers have used these representations to perform 3D scene understanding. This followed several new fields of research that quickly gained traction. To do so, advanced computer vision and deep learning techniques are proposed to extract underlying geometric and semantic clues of the 3D world from optical measurements. For instance, 3D object detection is the task of simultaneously detecting objects of interest and estimating a 6DoF pose given a point cloud or volumetric map. The output of 3D object detection approaches are 3D bounding boxes. 3D semantic segmentation, on the other hand, assigns a semantic label to each 3D point in the scene. Scene layout parsing, generating floor plans, or simply extracting scene structure – the list is endless. More recently, inter-object relationships and support connections, as well as temporal changes, are being investigated. In summary, given a 3D indoor scene, a vision-based algorithm needs to understand the spatial support layout, extract objects and their semantic context, and understand contextual and temporal relationships.

Finally, visual re-localization is *e.g.* defined as estimating the orientation and location of the current camera in a scene based on an image. While computing the camera pose given some observations with respect to the scene classically does not involve understanding. However, temporal changes require higher-level knowledge, and therefore, algorithms are required to detect and associate movable objects and scene structure. Indoor scene changes are immensely challenging, ranging from varying illumination, object poses, and deformations in geometric appearance as well as (dis-)appearing objects.

Solving all these tasks is a formidable challenge, and computer vision is still far from robustly and efficiently digitizing real-world scenes. *Computer vision researchers will surely not get bored any time soon!*

Contributions 2

This dissertation explores 3D scene representations that go beyond controlled environments. Existing scene representations typically only provide geometric and, more recently, semantic information, while long-term methods that try to enforce persistence are often missed. While scene segmentation tells us where and what is in the environment, it does not provide any higher-level understanding of objects, making it harder for vision-based applications to perceive and understand scenes and interact with objects within. We believe true higher-level scene understanding requires not only to capture geometric but sophisticated semantics, relational, temporal, or even affordance information at the level of the individual objects. In this thesis, we claim significant methodological and data contributions around scene changes and semantics.

We propose the first large-scale real-world RGB-D indoor dataset *RIO* of naturally changing indoor scenes. Our data is densely annotated and features 3D semantics, reconstructions, RGB-D sequences, and several object-level annotations. *3DSSG* then augments each scene in *RIO* with 3D scene graphs given additional semantic data from object instance attributes and class hierarchies to semantic relationships such as support or proximity, providing a 3D alternative to well known and used 2D semantic scene graph datasets.

We believe scene graphs could help to better understand object entities and their semantics, poses and potentially handle persistence. Therefore, we propose methods on existing challenges such as 3D semantic segmentation and instance segmentation [222, 297, 299, 308] but also transfer well-known 2D tasks such as scene graph prediction to 3D. To this end, we propose the first learned method that generates scene graphs from 3D point clouds [296]. [296] then suggests an improved scene graph prediction method that relies on sparse convolutions and does not require scene segmentations but instead jointly learns instance embeddings and semantics.

Finally, this dissertations explores new, novel directions when handling changes in 3D scenes. To do so, we propose new benchmarks and tasks for scene, camera re-localization, and object re-localization. We show how scene graphs are helpful in a domain agnostic scene retrieval task [296]. We utilize *RIO10*, an indoor camera re-localization benchmark where we propose methods to quantify scene changes, based on the geometric and semantic information provided in *RIO*, to ultimately understand their impact on re-localization performance. Finally, in [299] we propose a learned method for object instance re-localization where 6DoF transformations for objects of an RGB-D scan are estimated to align them with a reference scan, taken at a different point in time.

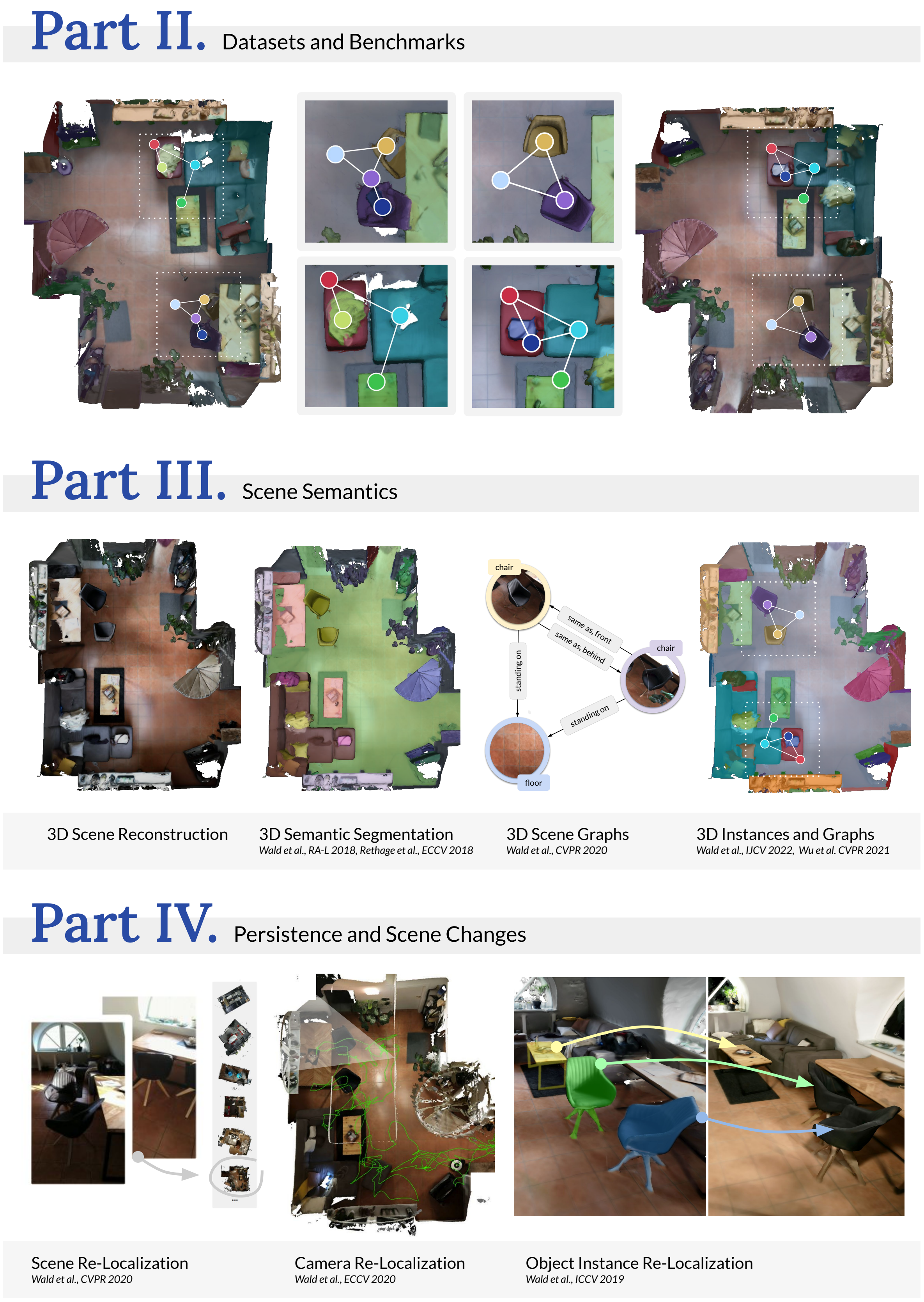


Fig. 2.1. Overview of the contributions of this dissertations. Datasets and benchmarks are discussed in Part II, scene semantics in Part III and persistence and re-localization in changing environments in Part IV.

Part II

Datasets and Benchmarks

data is a precious thing and will last longer than the systems themselves.

— **Tim Berners-Lee**
Computer scientist

Changing 3D Indoor Scenes

3

The content of this chapter is based on the following publications:

[295] **Johanna Wald**, Armen Avetisyan, Nassir Navab, Tombari Federico* and Matthias Nießner*. *RIO: 3D Object Instance Re-Localization in Changing Indoor Environments*. Oral, *International Conference on Computer Vision*. © 2019 IEEE.

[298] **Johanna Wald**, Torsten Sattler, Stuart Golodetz, Tommaso Cavallari and Federico Tombari. *Beyond Controlled Environments: 3D Camera Re-Localization in Changing Indoor Scenes*. *European Conference on Computer Vision*, 2020, Springer, Cham.

[207] Evin Pinar Örnek, Shristi Mudgal, **Johanna Wald**, Yida Wang and Federico Tombari. *From 2D to 3D: Re-thinking Benchmarking of Monocular Depth Prediction*. *under submission*.

3.1 Introduction

While RGB cameras capture visual appearance with primary colors (red, green, and blue) of a scene, depth sensors perceive geometry instead. In range imaging, for each pixel of a depth image, the distances to the observed 3D surface are obtained. In indoor setups, this is often done with a time-of-flight technique based on a built-in infrared sensor. Historical range imaging devices were a big investment and quite complex to use, making them challenging for quick prototyping and everyday users. This changed when the Microsoft Kinect was introduced in November 2010. While initially designed for games with the Xbox, it conquered the academic market by surprise. It was affordable and pretty straightforward to use, featuring all of the basic functionality in an API, such as calibration and optimization techniques. The Kinect as a 3D data source allowed quick prototyping, and soon various algorithms were proposed [114, 125, 202, 205, 303]. Online 3D reconstruction, also called dense SLAM, became one of the most important topics explored with these mobile depth sensors. Other sensors followed quickly, similar in affordability and offering excellent mobility, including Intel Real Sense or similar sensors that are today directly build into smartphones such as Google Tango devices or modern iPhones.

Dense SLAM algorithms such as KinectFusion [114], Voxel Hashing [205], Point Based Fusion [125], ElasticFusion [303] or later BundleFusion [56] enable 3D capturing of entire rooms to full buildings resulting in accurate 3D geometry. Importantly, the resulting RGB-D data has shown to be a useful representation for indoor scene understanding. While it became easier to record, process, create and store 2.5D/3D data, a plethora of RGB-D benchmarks for object tracking and detection [141] SLAM [100, 261] as well as head [71] human, hand and gesture analysis started to emerge [38]. One of the best-known RGB-D datasets for indoor scene understanding is arguably *NYUv2* [199]. Compared to other datasets at that time that often focused on objects and lacked scene variety, NYUv1 and v2 challenged the community with a more diverse set of indoor setups such as bedrooms, kitchens, or bathrooms. They further provided dense 2D semantics for 2347 images, and since its release, it has been used for many different indoor scene understanding tasks ranging from monocular depth estimation [67, 79, 143] to scene completion [259, 307]. Due to its small size of only 67 scenes, it is often too small for data-hungry methods, and proposed benchmarks are getting saturated.

In recent years, the research shifted from 3D scene perception to 3D scene understanding. Specifically, Given the 3D representations obtained from reconstruction algorithms, a rapidly emerging research field focuses on extracting higher-level semantic knowledge. Novel methods have been proposed to solve new challenging 3D tasks such as 3D object detection, 3D pose estimation, 3D semantic [54, 55, 216, 218, 222, 259] or instance segmentation [68, 99, 108, 117, 140, 320]. There are many large-scale collections of RGB-D sequences proposed for scene understanding research [75]. Based on these, various benchmarks were proposed to solve the tasks mentioned above, all with the common assumption that 3D scenes are static. Even though our surroundings are highly dynamic and changes happen regularly, they are infrequently discussed in the computer vision community. These include but are not limited to variations in illumination, appearance, and geometry. They are either caused by natural sources *e.g.* changes in light from day and night time cycles, or by humans *e.g.* changes in room layout through re-decoration or re-configuration, switching on/off external light sources, and simply using objects for their designed purpose. The lack of data arguably limits the possibilities of research and industry. To understand long-term changes in indoor environments, a dataset for evaluation and training is required. Tbl. 3.1 gives an overview of the most relevant 3D benchmarks listing their main characteristics, such as their size and acquisition modality. In this thesis, we propose *RIO*, the only large-scale dataset of changing scenes with semantic annotations. Most related datasets are static and do not feature changes such as *ScanNet* [54], which is commonly used in scene understanding applications and algorithms. It comprises of 1500 scans of approx. 750 unique scenes.

Only recently, the community started to explore long-term scene changes in indoor environments. InteriorNet is extremely large but synthetic. A physics engine was used to mimic realistic furniture movements and changes in lighting. Therefore, InteriorNet features variations in geometry and appearance [153]. Additionally, Li *et al*. measures the performance of two state-of-the-art SLAM methods, namely ORB-SLAM2 [193] and ElasticFusion [303] and unsurprisingly, show that scene changes cause challenges in mapping as well as localization. Even though InteriorNet is impressive in size, it still suffers from the domain gap problem due to its synthetic nature and photographs of real-world scenes. While the usage of synthetic data is promising, real-world recordings are inevitable, especially when evaluating long-term scene understanding algorithms. As of today, little real-world 3D indoor data is available. The data proposed by Fehr *et al*. [72] consists of only 23 sequences of 3 different indoor rooms is an example. It is used to detect moving furniture based on a TSDF representation. Notably, annotations are not included. Another real-world but small dataset is Rescan [98]. In contrast to [72], it does include annotations of changes, but its size is not sufficient for training and testing of data-hungry deep learning methods. Another dataset that features scene changes is Replica [260]. The data is of high quality but was mainly designed for virtual reality (VR) applications and is relatively small and lacks some diversity.

Another popular vision task with a lot of data availability is camera re-localization, though most benchmarks [45, 113, 129, 157, 163, 244, 253, 286] are either static [129, 253, 286] or also – as previously mentioned with 3D scene understanding benchmarks – do not provide the annotations and data needed to quantify and measure change impact [45, 113, 157, 163, 244, 266].

Tbl. 3.1. Overview of 3D indoor scene understanding datasets in comparison to *RIO*. © 2019 IEEE.

Dataset	Size	Real	Data Source	Benchmark Task	Changes
NYUv2 [199]	464 scenes	✓	Kinect	Depth and Semantics	✗
SUN RGB-D [257]	$10k$ frames	✓	4 different sensors	Object Detection	✗
SUN-CG [259]	$45K$ rooms, $500K$ images	✗	rendered, designed	Scene-Completion	✗
ScanNet [54]	1513 scans, $2.5M$ images	✓	Structure Sensor	Semantic Segmentation	✗
Fehr *et al*. [72]	23 scans of 3 scenes	✓	Tango	Change Detection	✓
Matterport3D [44]	$\sim 200k$ images	✓	Matterport	multiple	✗
SceneNet [183]	$5M$ images	✗	rendered, random	SLAM	✗
InteriorNet [153]	*millions*/unknown	✗	rendered, designed	SLAM	✓
Rescan [98]	45 scans, 13 scenes	✓	Structure Sensor	Instance Segmentation	✓
Replica [260]	18 scenes, 36 rooms	✓	Custom-built	VR Appplications	✓
***RIO* (Ours)**	1482 scans of 478 scenes	✓	Tango	Object Re-Localization	✓

3D scene understanding and camera re-localization are often viewed as two fully independent tasks. Therefore, most camera re-localization datasets do not include any semantic annotations. Some recently published benchmarks specifically designed for long-term camera re-localization provide test and train images under different conditions *e.g*. the Aachen Day-Night [243, 244], (extended) CMU Seasons [19, 243], RobotCar Seasons [176, 243], as well as SILDa [21]. They focus on scene changes and show that existing re-localization algorithms designed and evaluated on static scenes do not generalize well to changing setups. They show that robust methods are required to achieve better results on such realistic yet under-tested scenarios. Many methods [9, 24, 65, 85, 144, 214, 239, 276, 277] showed impressive results in long-term camera re-localization and their findings motivate more research in this field. While they mainly target outdoor environments, changes in indoor scenes are quite different, and generalization to indoor scenarios remains an unsolved task. Even though changes outdoors are quite prominent, they are often cyclic and state-dependent; this involves seasons, weather, and day/night cycles. These types of changes are easier to capture with a learned method [9, 214] and un-cyclic changes such as road work are arguably quite rare. While Tbl. 3.1 mainly focused on scene understanding, Tbl. 3.2 instead lists datasets that have been proposed for camera re-localization. It excludes the previously mentioned works [13, 44, 54, 109, 199, 257] and also omits other datasets that cannot be used to measure and quantify long-term changes [45, 113, 129, 157, 163]. Golodetz *et al.* [89] proposes a dataset for sub-map merging [89]. It neither features changes nor has previously been used to evaluate camera re-localization. Synthetic data has also been excluded due to the previously mentioned unavoidable domain gap [153, 235].

Tbl. 3.2. Overview of indoor and outdoor camera re-localization datasets in comparison to *RIO10* [298].

Dataset	Setup	Train Size	Test/Val Size	Sequences	Semantics	Time Span
7-Scenes [87]	indoor	26000	17000	✓	✗	no
12-Scenes [286]	indoor	16926	5702	✓	✗	no
NCLT [39]	both	N/A	N/A	✓	✗	15 months
InLoc [266]	indoor	9972	329	✗	✗	few days
Aachen Day-Night [243]	outdoor	4328	922	✗	✗	few years
Extended CMU-Seasons [243]	outdoor	60937	56613	✓	✗	24 months
RobotCar Seasons [243]	outdoor	26121	11934	✓	✗	12 months
SILDa [21]	outdoor	8334	6064	✓	✗	12 months
***RIO10* (Ours)**	indoor	52562	200159	✓	✓	12 months

The large amount of re-localization benchmarks in outdoor environments is prominent in Tbl. 3.2. They feature day and night time, seasonal changes that happen over the course of up to 1 year and – in some time zones more than others – weather changes such as rain, fog, or snow. Furthermore, long-term scene modifications such as growing and dying vegetation or construction work are sometimes covered. An example is the *Aachen Day-Night* [243] dataset that extends the Aachen dataset [244]. *Aachen Day-Night* [243] includes re-recordings approximately a year after their initial captures. It mainly focuses on day and night-time changes aiming to evaluate the effect of night- and day-time differences. The training data consists of a sparse 3D model [1] [247, 248] of the reference day-time scene and corresponding, registered images. The test images are recorded with different RGB sensors; notably, no sequences are available. Another alternative to [243] is the *RobotCar Seasons* [243] which is based on Oxford RobotCar [176]. It offers a challenge that features low-quality images taken at night time and includes weather and season changes. The RobotCar Seasons has an urban setup, similarly to *SILDa* [21] where a few building blocks in London were captured under changing weather and illumination. More variability in scenery is provided by the (extended) *CMU-Seasons* [19, 243] that covers more vegetation, compared to the previously mentioned datasets. The most commonly used indoor benchmarks are still *7-Scenes* [253] and *12-Scenes* [286]. Both of these have Kinect recorded sequences in static environments. The train and test images of neither *7-Scenes* [253] nor *12-Scenes* [286], include changes; contrary to *InLoc* [266], which recently has been proposed as a long-term indoor benchmark for camera re-localization. The query data in *InLoc* are stand-alone RGB photographs rather than images of a video sequence. Each photograph was taken with a hand-held device and later registered

[1] `https://colmap.github.io`

to the floor plan of the building. The variability of the dataset is rather limited since the captures are mostly within university buildings where object interaction is restricted [305]. Therefore, *InLoc* rarely features moved objects, similarly to *NCLT* [39]. Notably, none of the mentioned benchmarks can measure the impact of scene changes on camera re-localization since they lack methods to quantify the amount of change between query and source data. In [298] we published *RIO10*, which is described in Sec. 3.3, that enables detailed evaluation and analysis of long term camera re-localization in dynamic indoor setups.

3.2 RIO Dataset

In [295] a novel dataset called *3RScan* was published. In the following, we refer to this dataset as *RIO*. It is an extensive collection of real-world sequences, with re-scans, where each indoor environment was scanned multiple times $(2 - 12)$. It captured natural scene changes and was designed for long-term SLAM and camera and object instance re-localization. Detailed statistics of the size of the data are given in Tbl. 3.3.

Tbl. 3.3. Scene statistics of the splits of the *RIO* dataset [295].

	Train	Test	Val	Total	Description
scenes	385	46	47	478	unique environments, initial scans
re-scans	793	101	110	1004	re-scans of the same scene, excluding initial scan
scans	1178	147	157	1482	including both, initial scan and all of the re-scans

Fig. 3.1 gives an overview of the data by showing a reference and re-scan pair, and the corresponding provided annotations. In summary, *RIO* provides a large number of annotations and 3D data such as semantic instances or textured 3D models as well as 2D sequences including a) sequences of RGB and depth images as well as corresponding camera poses and calibration parameters, b) texture-mapped 3D meshes and semantic segmentation with temporal consistency, c) scan-to-scene transformations as well as d) ambiguity and symmetry aware object transformations. The next section gives an overview of the data acquisition (Sec. 3.2.1) algorithms and tools (Sec. 3.2.2) as well as the annotation process (Sec. 3.2.3).

3.2.1 Acquisition

Google Tango devices were utilized to record the data. It offers excellent flexibility and versatility due to its ease of use and its mobile and hand-held nature. Two different device types, Asus Zenfone AR and Lenovo Phab 2 Pro, both equipped with depth sensors, were used to record the data in *RIO*. To this end, a mobile application based on the Tango app *constructor* was developed, see Fig. 3.2.

The 3D scanning application enables the user to capture sequences and reconstructions of indoor scenes by simply pointing the camera at the objects and structure of the scene. A 3D viewer shows the online 3D reconstruction on the fly. Its instant feedback guides the user and helps understand which regions have already been scanned and which ones are still missing

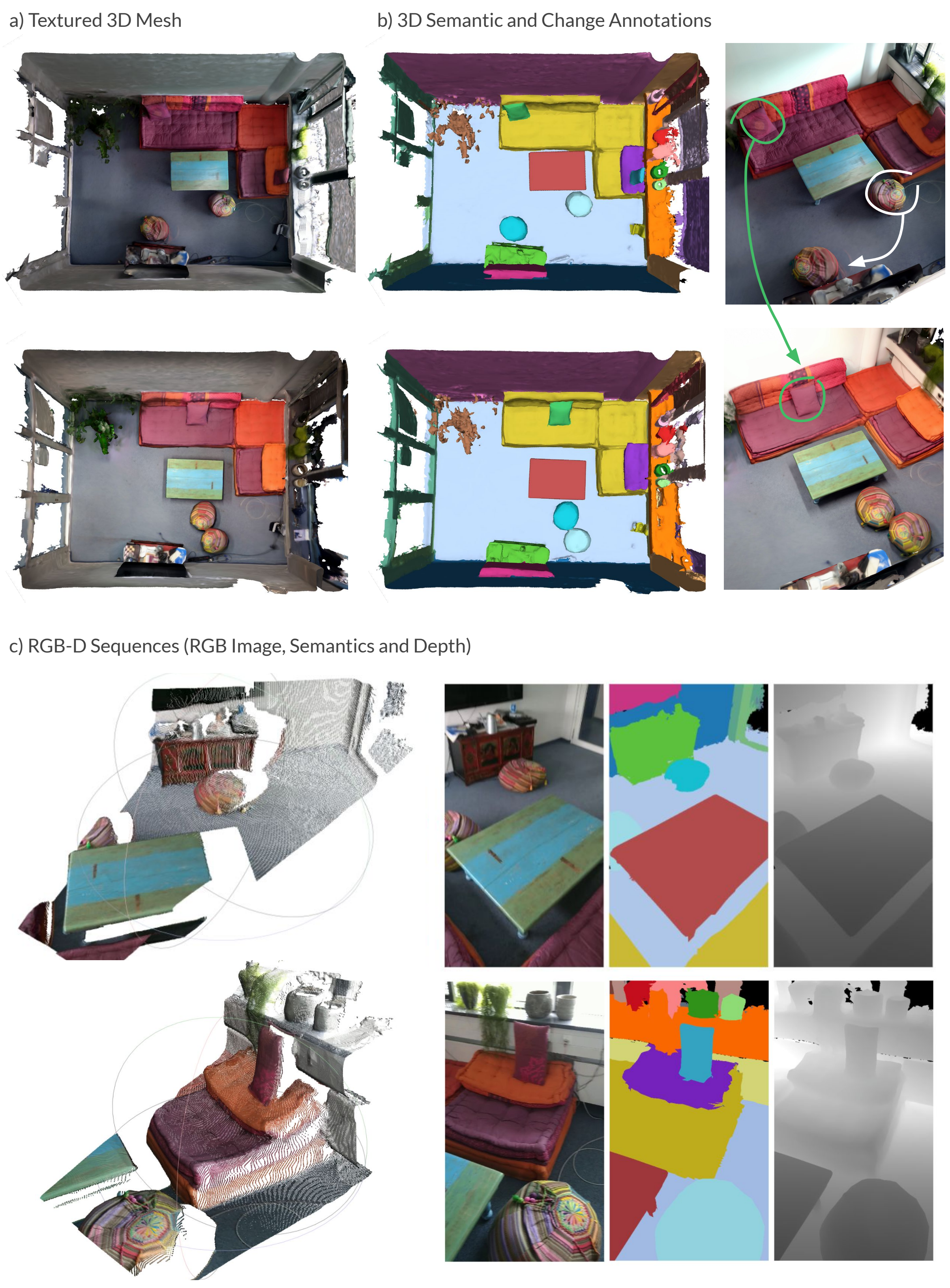
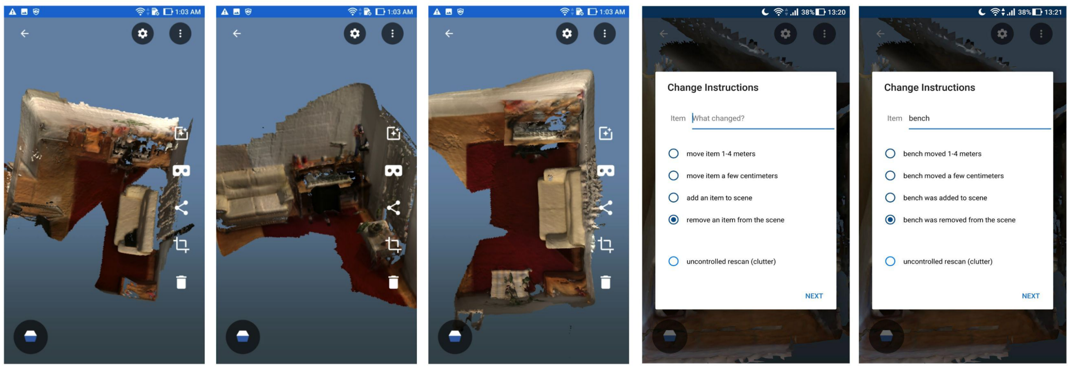

a) Textured 3D Mesh
b) 3D Semantic and Change Annotations
c) RGB-D Sequences (RGB Image, Semantics and Depth)

Fig. 3.1. Example of a 3D scene pair of the *RIO* dataset [295]. It provides a) registered, texture-mapped 3D reconstructions, c) dense instance-level semantic segmentation and symmetry-aware transformations of rigid changes as well as c) calibrated RGB-D sequences.



Change Instructions

Item What changed?

move item 1-4 meters

move item a few centimeters

add an item to scene

[✓ remove an item from the scene]

uncontrolled rescan (clutter)

NEXT

Change Instructions

Item bench

bench moved 1-4 meters

bench moved a few centimeters

bench was added to scene

[✓ bench was removed from the scene]

uncontrolled rescan (clutter)

NEXT

Fig. 3.2. Mobile application used for data acquisition with a live view of the online 3D reconstruction (left) and a GUI to optionally enter change instructions (right).

(see three images in the left of Fig. 3.2). The application was designed to allow easy usage for untrained users.

In the two screens on the right, a pre-annotation was possible. This interface was optional and allowed instructions and notes to be included before recording a new re-scan. If entered, these were used in the annotation process. Due to the ease of the recording, scenes could quickly be scanned. In the end, more than 45 different people gathered a set of diverse scenes in more than 13 different countries.

During the capture, we attempted to obtain a wide range of real-world scenes. The data acquired was either *controlled* and recorded within a few minutes or *uncontrolled*. In the letter, there was usually no knowledge about what has happened in the environment while *controlled* scans mimicked natural changes *e.g.* sitting on a chair, moving objects around.

The most extended period between uncontrolled recordings in *RIO* is approximately one year. While the goal was to get as many re-scans as possible, the number of scans per scene varies from $2-12$. Some, especially those with a significant amount of time between captures, feature quite some drastic changes, such as utterly different light setups and re-decorations. The continuous scanning of rooms where people manipulate the scene and objects over a considerable period of time makes our data and the captured changes highly variable. These range from objects being removed from the scene *e.g.* an ironing board or a washing basket which hides behind a cabinet and are therefore barely visible in the re-scans, see Fig. 3.3. Classical rigid movements happen regularly where objects move a few centimeters around their original place. Nonrigid deformations of objects like curtains or blankets introduce another type of change. Notably, drastic appearance variances are potentially caused by the slightest modifications *e.g.* open/closed door or sun/shade from outside. While *RIO* features all these changes as snapshots in long-term setups, dynamics and motion are purposefully not included. Also, humans were avoided and are therefore not captured within our dataset.

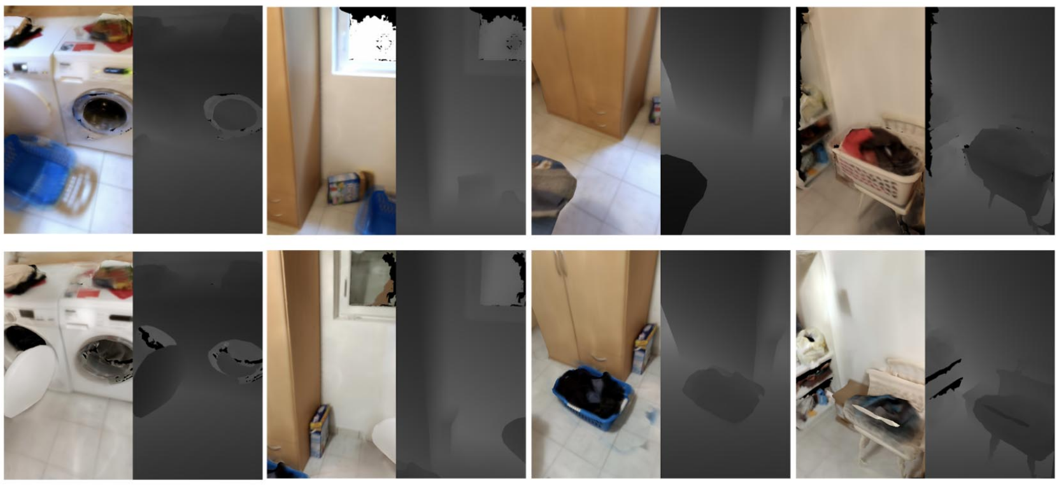


Fig. 3.3. Renderings (color and depth) of a reference and re-scan using the same camera pose. © 2019 IEEE.

3.2.2 Processing

The data recorded with the Tango device contains several sensor streams. Each scan consists of fish-eye and RGB images, depth maps, and IMU measurements, each with different frame rates and corresponding timestamps.

The data is first uploaded to our server to be processed with the Google Tango API. An offline process triggers camera pose optimization, which uses a bundle adjustment algorithm [282]. It utilizes 2D features extracted from the fish-eye images and simultaneously optimizes these with the 3D geometry of the map to get the best possible alignments. On the contrarily, the live viewer shown in Fig. 3.2 uses Tangos online visual-inertial odometry (VIO) [173, 200] based on IMU and visual image information.

The images are temporally calibrated from the stream of depth images by selecting corresponding frames with a maximum time difference below a certain, low threshold.

A TSDF-based 3D reconstruction is computed by representing the scene with a fixed grid of voxels of $2\,\mathrm{cm}$ resolution. The voxels store the distance to the closest surface using the Signed Distance Function (SDF) [52, 202]. The SDF determines the distance of a point $\mathbf{x}$ to its closest surface. $f_{SDF}(\mathbf{x}) = 0$ is the zero-crossing and the location of the surface, a positive value $f_{SDF}(\mathbf{x}) > 0$ describes the region in front of the surface and negative values $f_{SDF}(\mathbf{x}) < 0$ are behind or inside objects.

These measurements can easily be computed from a given camera image and are then merged in a global volume by taking the weighted average. Based on the TSDF representation, a 3D mesh with textures is extracted using the marching cubes algorithm [170]. To optimize the 3D model, planes are fitted onto the 3D mesh to flatten planar structures and to reduce noise [66]. Formally, the resulting 3D mesh consists of a set of vertices and faces

$$\mathcal{M} = (\mathcal{P}, \mathcal{F}). \tag{3.1}$$

The vertices $\mathcal{P}$ consist of points $\mathbf{v}_i = (\mathbf{p}_i, \mathbf{n}_i, \mathbf{c}_i, \mathrm{l}_i, \mathrm{i}_i)$ with point coordinates $\mathbf{p}_i$, surface normals $\mathbf{n}_i$, color values $\mathbf{c}_i$ and semantic l_i and instance labels i_i defined as

$$\mathcal{P} = \{(\mathbf{p}_i, \mathbf{n}_i, \mathbf{c}_i, \mathrm{l}_i, \mathrm{i}_i) \mid \mathrm{l}_i, \mathrm{i}_i \in \mathbb{N}_0, \mathbf{c}_i \in \mathbb{C}^3 \text{ and } \mathbf{n}_i, \mathbf{p}_i \in \mathbb{R}^3\}_{i=1}^{\mathrm{N_P}}, \tag{3.2}$$

$$\mathbb{C} = \{\forall k \in \mathbb{R}_0 \mid k \leq 1\}. \tag{3.3}$$

In the following chapters, we refer to a point set that holds only point coordinates $\mathbf{p}_i$ as

$$\mathcal{P}^{(\mathrm{p})} = \{\mathbf{p}_i\}_{i=1}^{\mathrm{N_P}}. \tag{3.4}$$

Finally, the images require alignment, and therefore – in a final processing step – depth and RGB images are calibrated by utilizing the camera intrinsics.

In summary, each scan of *RIO* provides texture-mapped 3D reconstructions and calibrated RGB-D sequences. Even though the fish-eye images were used to compute camera poses, they were not released since they often unknowingly captured the person scanning the room.

Scene Matching and Alignment

In the first annotation step, sequences of the same environment are clustered together. To do so, potential matches are obtained by using the texture map of the 3D model. A feature encoding per image is computed to then estimate the similarity between all UV-maps. The most similar matches are the initial scan-to-scene assignment and are manually corrected in a subsequent verification procedure.

Once the reference scan was assigned to all of the 478 scans, they are globally aligned. Hence, a transformation was computed that optimally registers a set of static keypoints $\mathcal{K} \subset \mathcal{P}_s^{(\mathrm{p})}$ on the re-scan $\mathcal{M}_s$ to the surface of the reference $\mathcal{P}_0^{(\mathrm{p})}$ using a transformation matrix $\mathbf{H}_s \in \mathbb{R}^{4\times4}$.

Since the 3D models are roughly gravity oriented, only the z-offset and a rotation around the z-axis are significant. A coarse to fine ICP was used, which first utilized a top-down 2D ICP followed by a 6DoF 3D ICP for further refinement. Due to heavy changes, noise, and drastic differences in the scans, this automatic computation resulted in several failure cases.

Therefore, each transformation was carefully verified and optimally either refined or corrected using a key-point based alignment similar to the later-described object pose annotation, described in Sec. 3.2.3. There, 3D keypoints are first chosen on the reference scan and the correspondences are subsequently selected on the re-scan surface. These distinct key points are then used to compute an optimal 6DoF pose that aligns the re-scan to the reference, such that $\mathbf{H}_s$

$$\underset{\mathbf{H}_s}{\arg\min} \sum_{\mathbf{p}_s \in \mathcal{K}} \min_{\mathbf{p}_i \in \mathcal{P}_0^{(\mathbf{p})}} ||\mathbf{p}_i - \mathbf{H}_s \mathbf{p}_s||^2. \tag{3.5}$$

RGB-D Sequences

Additionally to the 3D data, we also provide RGB-D sequences for each scan s that consist of RGB data $(\mathbf{I}_{s,i}^{(r)})_{i=1}^{\mathrm{N_F}}$, depth maps $(\mathbf{I}_{s,i}^{(d)})_{i=1}^{\mathrm{N_F}}$ and camera poses $(\mathbf{P}'_{s,i})_{i=1}^{\mathrm{N_F}}$, see eq. 3.7). Each sequence has timestamps $(t_{s,i})_{i=1}^{\mathrm{N_F}}$. The sequence timestamp is defined as the first and therefore smallest measurement

$$t_s = t_{s,1} = \min_{i=1}^{\mathrm{N_F}}(t_{s,i}). \tag{3.6}$$

Each camera pose consists of a rotation $\mathbf{R}_{s,i} \in \mathbb{R}^{3\times3}$ and translation $\mathbf{t}_{s,i} \in \mathbb{R}^3$. $\mathbf{P}'_{s,i}$ is the camera pose in re-scan coordinate frame while $\mathbf{P}_{s,i}$ is the respective pose in the reference coordinate frame such that

$$\mathbf{P}'_{s,i} \in \mathbb{R}^{4\times4} = \begin{bmatrix} \mathbf{R}_{s,i} & \mathbf{t}_{s,i} \\ \mathbf{0}^\top & 1 \end{bmatrix} \tag{3.7}$$

$$\mathbf{P}_{s,i} \in \mathbb{R}^{4\times4} = \mathbf{H}_s\mathbf{P}'_{s,i}. \tag{3.8}$$

Please note, in the following chapters we often omit the s and/or i-index and refer to a camera pose simply as $\mathbf{P}$. The RGB images $\mathbf{I}^{(r)}$ are recorded in portrait mode and have a resolution of 960×540 pixels. The corresponding raw sensor depth $\mathbf{I}^{(d)}$ is of relatively low quality with a resolution of only 224×117 pixels. In order to compensate for this, a rendering function that generates synthetic depth images from the 3D reconstruction given a camera pose $\mathbf{P}$ is provided.

Compared to the raw data initially captured, the released sequences in *RIO* have a reduced number of frames. Contrarily to the sequences in *RIO10*, see Sec. 3.3, this is due to the low frame rate of only $5\,\mathrm{FPS}$ of the Tango depth sensor and the temporal calibration procedure which selects RGB, depth and IMU measurements only iff their timestamps are reasonably similar *i.e.* within a small threshold. Since RGB and depth images are also *spatially* calibrated, the depth measurement at pixel $\mathbf{u}$ directly corresponds to the color value at $\mathbf{u}$ in the scaled RGB image.

Fig. 3.4 visualizes camera trajectories on examples scenes of *RIO*. Given the camera sequence of a re-scan and the global transformation to its reference, the re-scan camera poses in reference coordinate frame can easily be computed as stated in eq. 3.8. Examples are shown in the two bottom rows of Fig. 3.4. Such an alignment is quite convenient since it enables mapping the poses of a re-scan into its reference as done in *RIO10*, see Sec. 3.3. It further allows rendering the same environment at two different times with the same camera pose, see Fig. 3.3.

The rendering of synthetic images uses camera parameters. These camera intrinsics $\mathbf{K}_s$ are provided for each sequence and are defined as

$$\mathbf{K}_s \in \mathbb{R}^{3\times3} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_x & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.9}$$

where f_x, f_y refer to the focal lengths and c_x, c_y are the principle points or camera center of the image plane. To keep the notation as simple as possible, $\mathbf{K}_s$ is in the following, often referred to as $\mathbf{K}$. Furthermore, radial distortions on the RGB images of Google Tango devices are insignificant for our use-cases and are ignored in our computational models. Given this simplified camera model, a 3D point in camera coordinates space is projected into the camera space as follows

$$\begin{pmatrix} q_x \\ q_y \\ q_z \end{pmatrix} = \mathbf{K}^{-1}\mathbf{P}_i \begin{pmatrix} p_x \\ p_y \\ p_z \\ 1 \end{pmatrix}. \tag{3.10}$$

3.2.3 Annotation

Great efforts have been made to provide labeled data such as the earlier mentioned global scan-to-scene alignment (see Sec. 3.2.2), dense semantic instance segmentation, and change annotations with object transformations, described in the following.

Semantics

The annotation interface for 3D semantic segmentation is based on a prior geometric segmentation of the 3D scene. Similar to [54] we utilized a modified version of the *Graph Based Image Segmentation* algorithm proposed by Felzenswalb and Huttenlocher [74] which segments a mesh based on its surface normals. Since the resulting geometric segmentation is the input of our 3D annotation interface, every 3D model of our dataset is segmented before semantics comes into play. Our semantic segmentation aims to ultimately assign unique instance IDs – which link to a semantic class label – to every geometric segment of the scene. For data annotation, we deployed the publicly available *sstk* `https://github.com/smartscenes/sstk` on a crowd-sourced server interface. Expert annotators were instructed first to create object instances and name them with a class label. In this step, a unique ID is assigned to all newly created instances. For each of these instances, the corresponding segments on the 3D surfaces are selected by simply hovering over the respective 3D geometry while holding the left mouse button down.

Each reference scan in *RIO* is annotated from scratch with this procedure and, notably, neither uses prior knowledge nor any constraints. The annotation of the re-scans on the other side is pre-annotated, to begin with, given the annotations of its reference model. To map these annotations to the corresponding re-scan segments, the 3D alignments $\mathbf{H}_s$, see eq. 3.5, are combined with a segment based kNN search. Where the label of a segment in a re-scan is obtained from the nearest, most overlapping segment in its reference scan. An example of such an automatically propagated semantic segmentation of a re-scan with changes is given in Fig. 3.5.

This propagation is based on the assumption that most of the scene structure is still static, which is mostly true for *e.g.* walls, the floor, or big furniture. Even though some noise and artifacts are present, relatively good segmentation is achieved, especially in static regions and mainly the changed surfaces were wrongly segmented. In a subsequent procedure, these re-scan annotations are cleaned by correcting segments wrongly propagated using the same

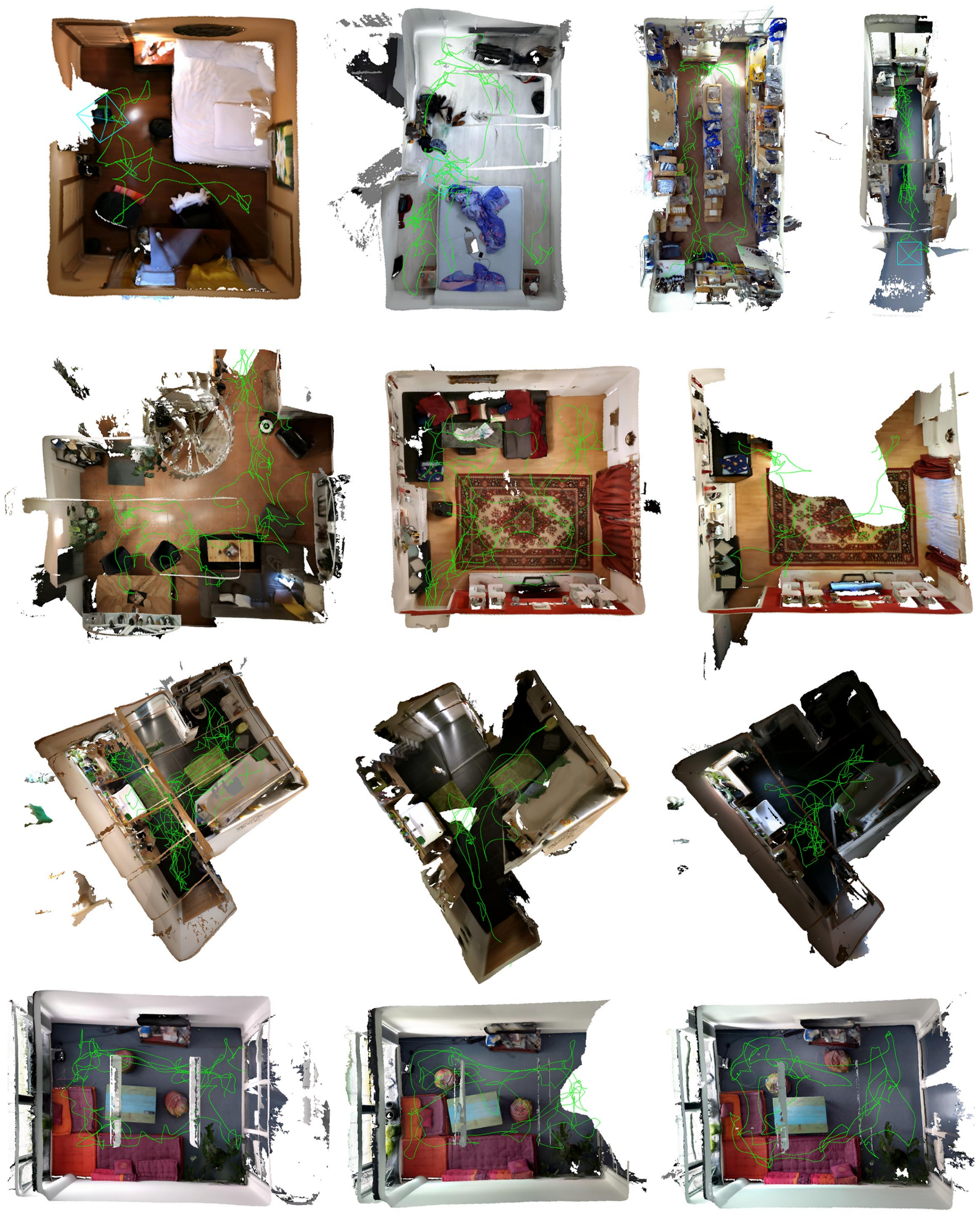


Fig. 3.4. Camera trajectories of different scans and scenes of the *RIO* dataset. © 2019 IEEE.

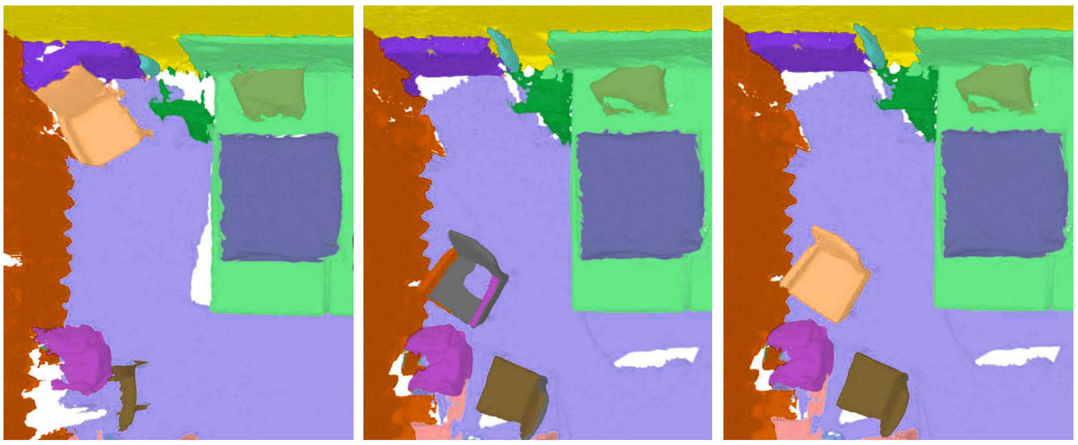


Fig. 3.5. Semantic segmentation of a re-scan in *RIO*: Annotations are propagated from the initial scan (left) to a re-scan (center). Incorrect propagation of changed regions *e.g.* armchair, are fixed manually (right). © 2019 IEEE.

annotation interface described before. Notably, when propagating and annotating re-scans, instance IDs are kept consistent. Furthermore, it is ensured that they are retained over time *e.g.* a chair that moves around has the same IDs in each scan of the same scene. In our illustrations, unique IDs correspond to unique colors, see Fig. 3.5.

The annotators were instructed to label as much surface area as possible, and eventually a surface coverage of above 98% was achieved, see Fig. 3.7. We additionally introduced a *remove*-label to select areas in the reconstruction that were artifacts or incorrectly captured, *e.g.* mirrored surfaces. Consequently, they have been deleted from the 3D data.

Example of final 3D semantic segmentations with several re-scans and reference are shown in Fig. 3.6. Formally defined, these semantic and instance segmentation of a point set $\mathcal{P}^{(\mathrm{p})}$, see eq. 3.4, are

$$\mathcal{P}^{(\mathrm{l})} = \{\mathrm{l}_i\}_{i=1}^{\mathrm{N_p}} \text{ and } \mathcal{P}^{(\mathrm{i})} = \{\mathrm{i}_i\}_{i=1}^{\mathrm{N_p}}, \tag{3.11}$$

where l_i describes the semantic label and i_i refers to the instance ID of a 3D point $\mathbf{p}$, see eq. 3.2. Annotation time per scan took approx. $45\,\mathrm{min}$ in average. Notably, each scan was verified multiple times by the authors. In the end, *RIO* consists of 48k instances and features 534 free-text class labels.

Changes

Once the semantics and instances are correctly segmented, scene changes are to be annotated. For this purpose, a web-based annotation interface was developed based on the CAD alignment tool of Avetisyan *et al.* [17]. In our interface, see Fig. 3.9, a re-scan and the corresponding reference are visualized next to each other. An expert annotator first selects a changed object in the re-scan by clicking on the visible surface area. For additional guidance and to ensure a correct selection, a class label and an instance ID are shown while the mouse is hovering over objects in the scene.

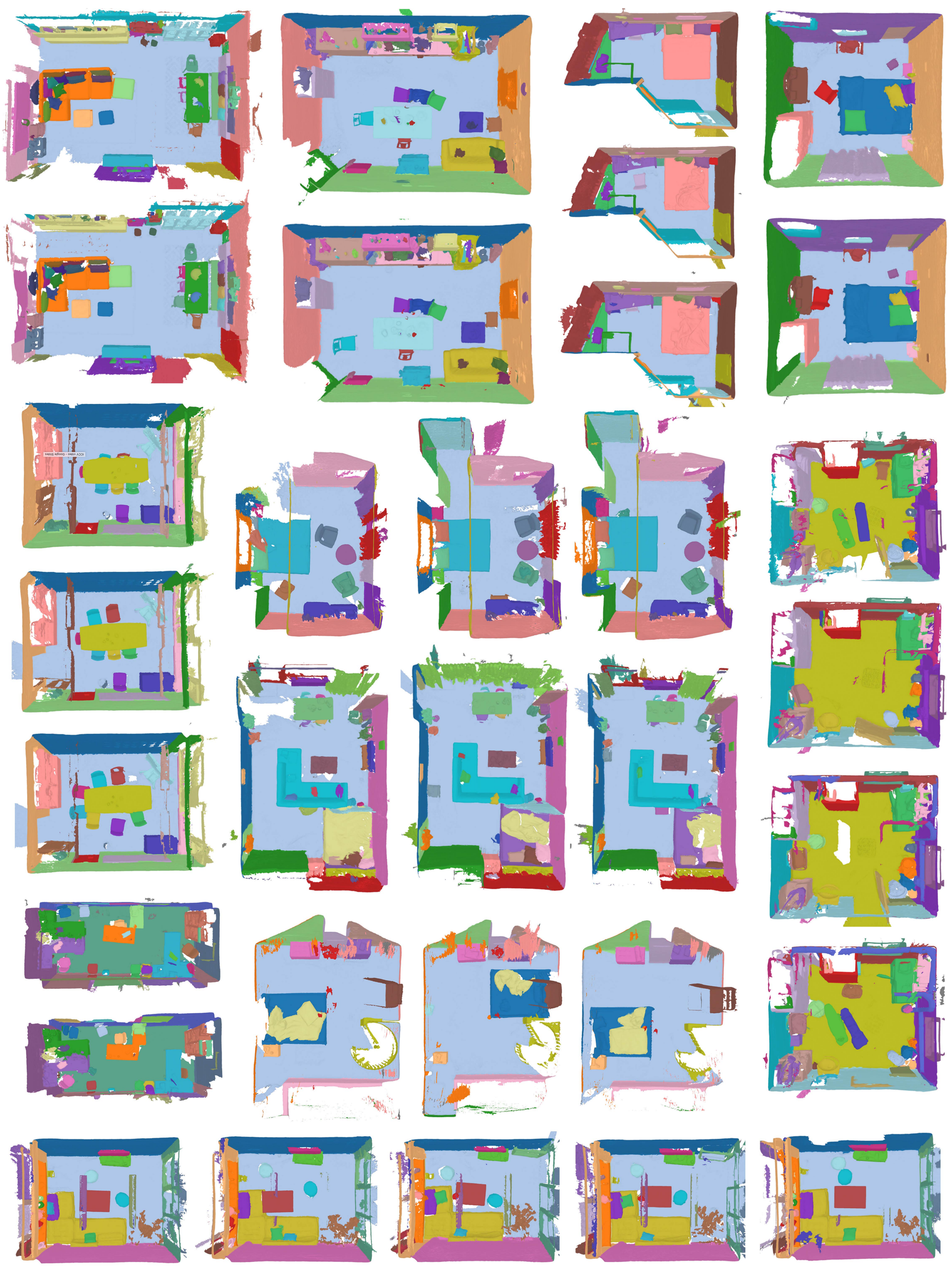


Fig. 3.6. 3D instance segmentation examples in *RIO*. © 2019 IEEE.

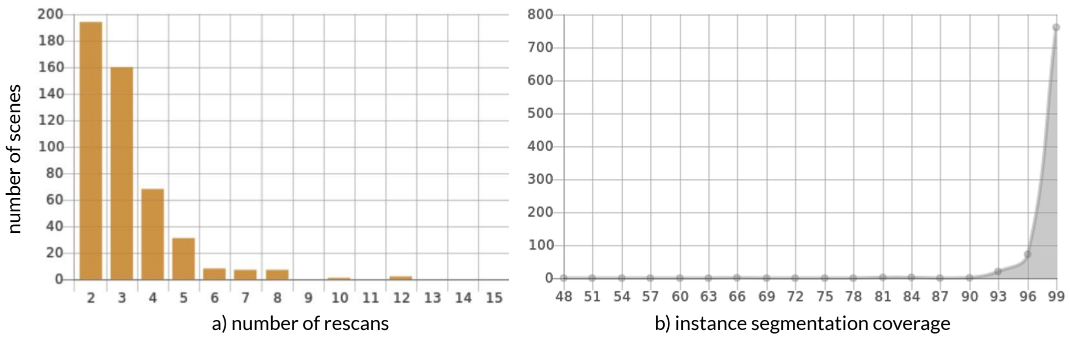


Fig. 3.7. a) number of re-scans per scene (left) and b) instance segmentation coverage of the scenes (right). © 2019 IEEE.

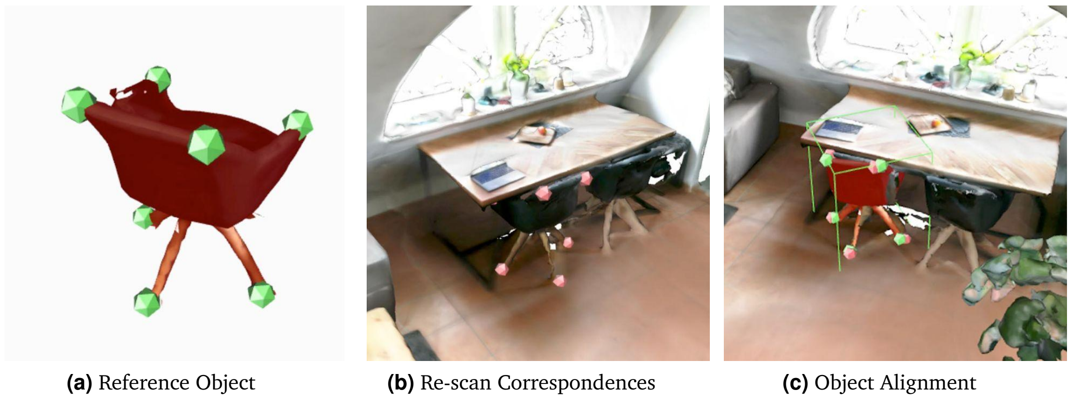


Fig. 3.8. Keypoint based annotation interface to align instances of a re-scan to its reference scan. © 2019 IEEE.

Since the instance IDs i_i are consistent across multiple scans of the same scene, vertices have to exist for the selected instance in the re-scan as well as in the reference. This serves as a verification step and another round of labeling and correction might be necessary if potential annotation inconsistencies in the semantic segmentation are identified.

We used the previously mentioned instance segmentation $\mathcal{P}^{(\mathrm{i})}$ to extract a partial 3D model of the object of interest, see Fig. 3.8a. For this purpose, the set of indices of the selected instance $k \in \mathbb{N}_1$ are first extracted from the instance segmentation $\mathcal{P}^{(\mathrm{i})} = \{\mathrm{i}_i\}_{i=1}^{\mathrm{N_P}}$, where i_i is the instance ID of $\mathbf{p}_i$, see eq. 3.12. The 3D object data of instance k is then given by combining the vertices of the 3D scene reconstruction and the extracted indices $\mathbf{S}_k$, such that

$$\mathbf{S}_k = \{i \mid \mathrm{i}_i = k\}_{i=1}^{\mathrm{N_P}}, \tag{3.12}$$

$$\mathcal{P}_k = \{\mathbf{v}_i\}_{\mathbf{S}_k} \text{ where } \mathbf{v}_i = (\mathbf{p}_i, \mathbf{n}_i, \mathbf{c}_i, \mathrm{l}_i, \mathrm{i}_i). \tag{3.13}$$

Once successfully selected, the resulting 3D object $\mathcal{P}_k$ is rendered in a new window next to its reference scan. In this view, the task at hand is the selection of key points. First, on the changed object in the re-scan (see green points in Fig. 3.8a) and then on the corresponding reference counterpart (see red points in Fig. 3.8b). This selection continues until enough points are selected. The chosen key points are used to compute a 3D alignment that registers the object to the scene. We employ Procrustes [249] and apply the Kabsch algorithm to

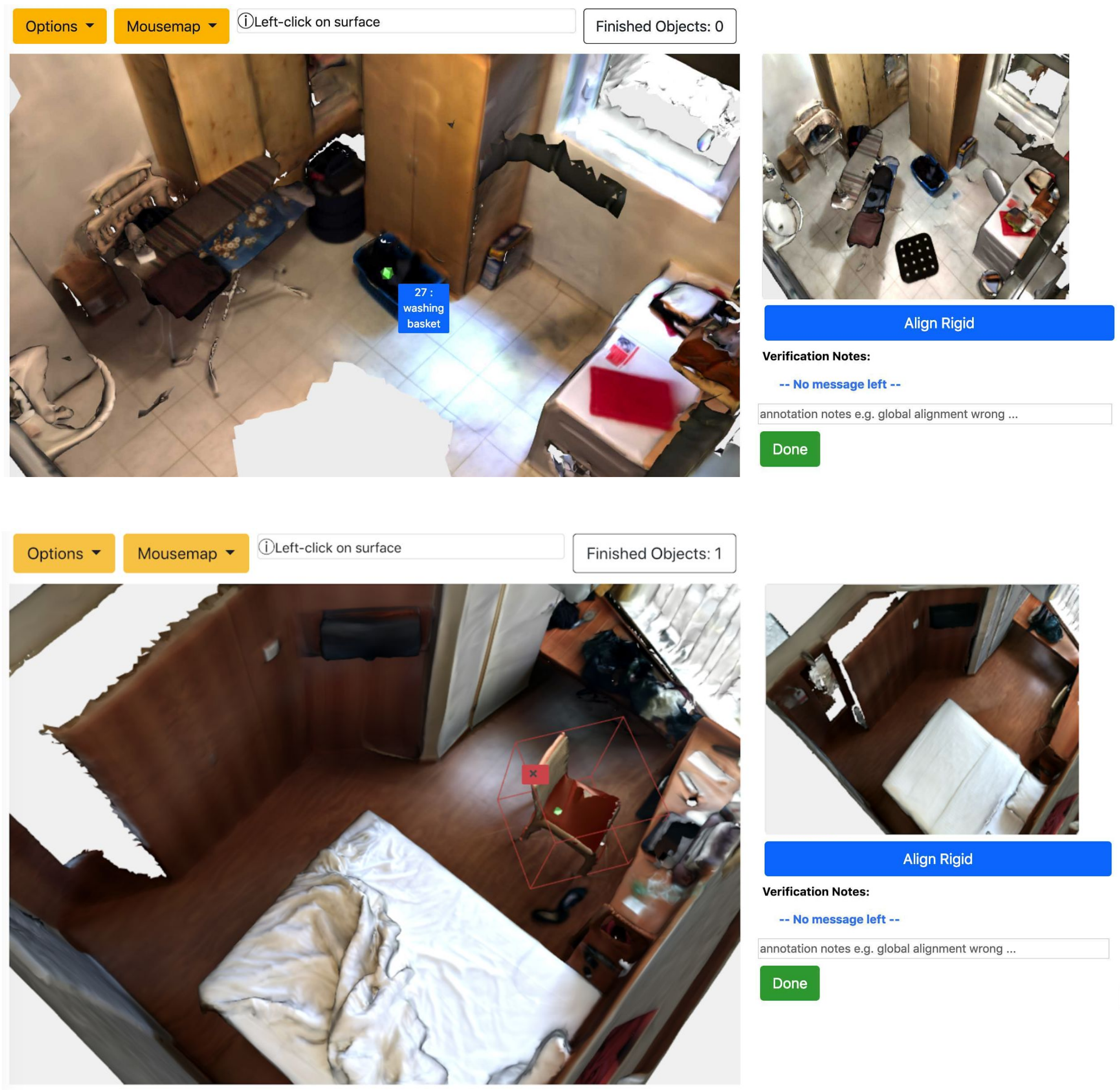


Fig. 3.9. Web-interface to annotate scene changes. © 2019 IEEE.

compute the correct 6DoF pose, which is subsequently visualized for verification and potential re-annotation purposes, see Fig. 3.8c.

A major problem when annotating these changes are symmetric objects, see Fig. 3.10. In case of symmetries, transformations are not unique *e.g.* a round carpet or rectangular cushions have multiple correct transformations and therefore keypoint correspondences are ambiguous. The symmetry is important not only when annotating transformations but also when evaluating object instance re-localization which will be discussed in chapter 9. We follow previous work [17] and assign a symmetry label $\omega \in \{$ 🟦 0 (none), 🟩 1, 🟪 2, 🟧 3 $\}$ to each rigidly transformed object. Notably, the symmetry is rotational around the z axis with either one ($\omega = 1$), two axis ($\omega = 2$) or a full rotational symmetry ($\omega = 3$). Interestingly, more than 20% of objects have some form of symmetry where $\omega \geq 1$.

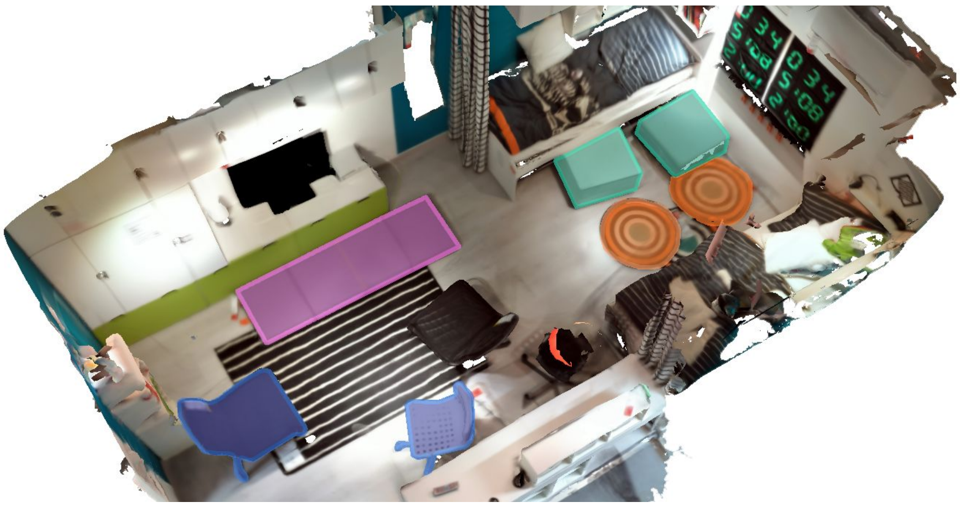


Fig. 3.10. Visualization of different symmetries in a scene of *RIO*: no transformation symmetry (chairs), symmetry on one axis (floor mat), symmetry on two axis (ottomans) or symmetry on multiple axes (round curtains).

Another important ambiguity to consider is objects that look alike, see Fig. 3.11. To recover all valid transformations for evaluation purposes, alignments are computed from one instance in the references scan to all other objects that look alike. These transformations are then used when evaluating object instance re-localization, see Sec. 9.4.

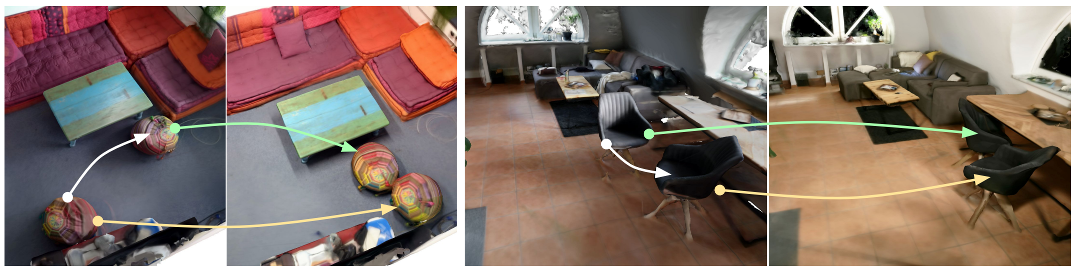


Fig. 3.11. Instance ambiguities in presence of scene changes and unknown instance mappings across scans.

A side note on the data annotations in *RIO*: instance IDs of disappearing objects are tracked as well as newly added objects since they require manually creating a new instance ID when labeling re-scans. In total, 3289 transformations of 1947 instances with 187 different classes were annotated. Most moving objects are portable and often interacted with *e.g.* bins, pillows, chairs, but *RIO* also features unusual scene changes where big furniture was re-located *e.g.* beds or big tables.

3.2.4 Data Organization

In the following, we give an overview of the files and formats of the dataset. This section serves as high-level documentation for anyone interested in using the data. First and foremost, *RIO* is organized by RGB-D sequence. A hash ID uniquely identifies each sequence. Once downloaded, the 2D sequences and 3D reconstructions of each scan are grouped in a folder named after this ID *e.g.* `5630cfcf-12bf-2860-8784-83d28a611a83`. The directory structure of a scan in *RIO* is listed in Tbl. 3.4.

Tbl. 3.4. Directory structure of a scan of the *RIO* dataset.

Filename	Description
`mesh.refined.obj`	Reconstructed surface mesh file (`*.obj`): OBJ format mesh with +Z axis in upright orientation, includes surface normals.
`mesh.refined.mtl`	Corresponding material file (the same for all scans)
`mesh.refined_0.png`	Corresponding mesh texture
`sequence.zip`	Calibrated RGB-D sequence and intrinsics. Depth maps are 16 bit images and store depth as unsigned short in millimeter. The camera poses `*.pose.txt` give the pose as a transformation from the RGB camera to the world coordinate system. The translation is in meters.
`labels.instances.*.ply`	Visualization of semantic segmentation
`mesh.refined.*.segs.json`	Over-segmentation of annotation mesh
`semseg.json`	Instance segmentation of the mesh (contains the class labels).

Furthermore, a rendering pipeline[2] is available, which additionally provides functionality to compute occlusion and truncation measures of the objects in a given frame.

A segmentation file `mesh.refined.*.segs.json` could look like this:

```
{"sceneId": "8eabc405-5af7",
 "segGroups": [{
     "id": 15,
     "label": "window",
     "segments": [ 21, 175, ... ],
     "obb": {...}, ... },
    {"id": 29, "label": "plant",  ... },
    {"id": 14, "label": "windowsill",  ... } ...
```

And the meta data file `3RScan.json` is structured as follows:

```
[{"reference": "531cff08-0021",          // scene id
  "type": "train"
  "ambiguity": [[{
        "instance_source": 34,
        "instance_target": 35,
        "transform": [...]}, {...}]],    // aligns instance 34 with 35
  "scans": [{                            // re-scans
        "reference": "531cff10-0021",    // scan id
        "transform": [...]               // aligns re-scan with reference
        "nonrigid": [...],               // instances with nonrigid changes
        "removed": [],                   // removed instances
        "rigid": [{                      // rigid changes
            "instance_reference": 35,    // instance ID reference
            "instance_rescan": 35,       // instance ID re-scan
            "symmetry": 0,               // symmetry (0 = none)
            "transform": [...], ...      // aligns instance to re-scan
```

[2] https://github.com/WaldJohannaU/3RScan/c++/rio_renderer

3.3 RIO10 Dataset

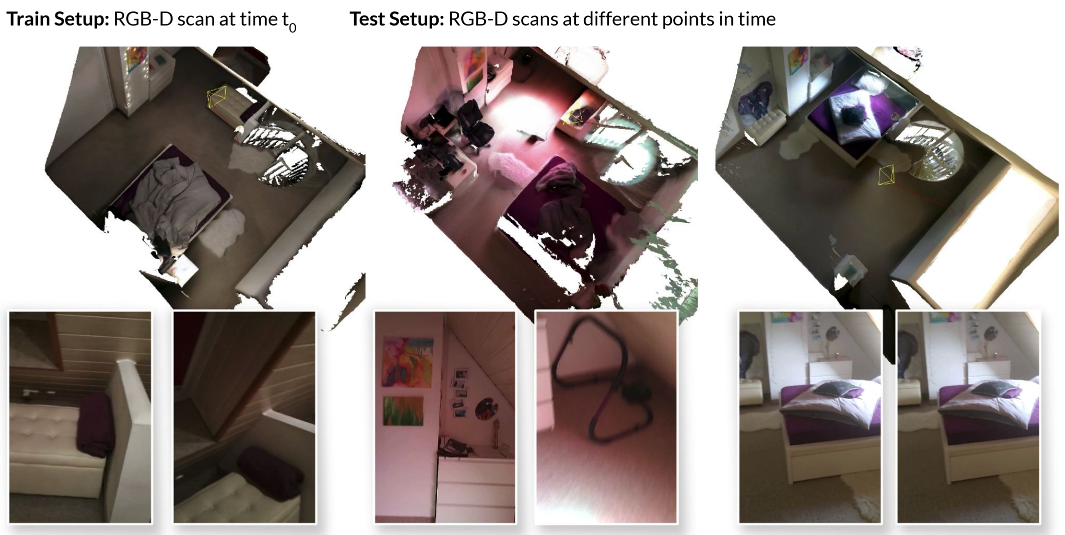

Train Setup: RGB-D scan at time t_0
Test Setup: RGB-D scans at different points in time

Fig. 3.12. Camera re-localization in changing setups performs re-localization given a sequence recorded at start time t_0 (left), on query images captured at a different point in time (center and right) [298].

The *RIO10* dataset is an indoor re-localization benchmark based on *RIO* [295], see Fig. 3.12. Though originally developed for object re-localization, *RIO* can also be used to benchmark other related long-term tasks when marginally modified. Since the data in *RIO* captured scene changes over a period of 12 months, it is precious for the camera re-localization community that currently mostly focuses on static scenes *e.g.* 7-Scenes [87].

Since the camera poses $(\mathbf{P}'_{s,i})_{i=1}^{\mathrm{N_F}}$, see eq. 3.7, of each sequence s of length $\mathrm{N_F}$ are computed based on features of a wide-angle image as well as the full 3D model in an offline process, they provide appropriate ground truth quality for image-based camera re-localization. The necessary modification, re-calibration as well as the selected scenes are described in the following sections.

3.3.1 Scene Selection

Tbl. 3.5. Overview of the selected scenes in *RIO10* [298]

Scene ID	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**
Re-scans	6	8	7	10	5	12	8	5	5	8
Objects	39	33	20	28	44	49	39	61	67	63
Changed Objects	$5-9$	$5-6$	$2-3$	$1-5$	$1-2$	$1-6$	$6-10$	$1-5$	$5-6$	$7-9$
Re-scan Day Span	176	165	369	176	163	173	104	229	1	168

Due to the overwhelming size of the initial dataset, we decided to base our experiments and validation protocol – described in chapter 8 – on a subset of 10 scenes, which gave *RIO10* its name. Fig. 3.13 shows a subset of the selected sequences while Tbl. 3.5 gives a comprehensive overview by listing the number of re-scans, objects, changed objects, as well as the maximum day span between captures of each scene. The subset of scans from the very large *RIO* dataset was selected for different reasons. The chosen scenes have a relatively high scanning frequency ($5-12$), resulting in many re-scan sequences and a lot of data per scene. Furthermore, indoor setups were chosen to have many, potentially drastic, changes. As mentioned, the scene setups are pretty diverse and range from living rooms (S02) to laundry basements (S01), bathrooms (S05), bedrooms (S3), to complete apartments (S10). The dynamics in *RIO10* are also quite versatile and cover varying illumination as well as different types of appearance change. From drastic lighting change caused by recording at different times of the day to the rigid movement of small as well as big objects, *e.g.* bed/sofa in S03 and S04 or nonrigid deformations *e.g.* blankets in S03 – *RIO10* has it all! Additionally, most scenes feature some ambiguous instances such as a set of identical chairs.

The remaining unused scans in *RIO* are officially not part of our camera re-localization benchmark. However, since they share the same domain, they might potentially be helpful for training a network that learns *e.g.* change-invariant feature embeddings.

In contrast to *RIO* [295] that categories all sequences of a scene into either train, test, or validation, camera re-localization requires a different splitting. At least one sequence per scene is needed for training, validation as well as testing. While the training sequence is chosen as the reference recording at t_0, see eq. 3.6, we formally refer to the validation sequence as the one captured at t_1 and any scan at t_s where $s \geq 2$ is dedicated for testing. This results in 10 train, 10 validation and 54 test sequences with $52\,562$, $34\,415$ and $165\,744$ images for training, validation and testing respectively. Our test data is significantly larger than the train and validation set. This ensures to cover a wide range of conditions as our goal is to quantify the influence of changing conditions on localization performance accurately. A practically adequate localization method has to operate robustly in different conditions without having to observe all of them.

In summary, each scene consists of several textured 3D meshes $\{\mathcal{M}_s : 0 \leq s < m\}$ which store color, surface normals as well as semantics, see Fig. 3.1. Similarly to the timestamp definition of t_0, t_1 and t_s, $\mathcal{M}_0$ is defined as the 3D training data, while the remaining 3D models $\mathcal{M}_1$ and $\mathcal{M}_s$ are used for validation and test accordingly.

3.3.2 Processing

In *RIO10* we provide aligned poses by applying the global scan-to-scene transformation to every pose of each re-scan sequence, see eq. 3.8. Thus, the poses of each scene are all in a common reference frame, see Fig. 3.13. The camera's extrinsic matrix $\mathbf{P}$ is provided as transformations to the 3D model coordinate system, as commonly done in other camera re-localization benchmarks. Notably, the temporal calibration in *RIO* requires a depth image frame within a small maximum temporal offset. Since the frame rate of the Tango depth camera is relatively low, this filters out a significant amount of RGB and IMU measurements. In *RIO10*, we decided to generate semi-synthetic depth maps by rendering the 3D models to compensate for the low resolution and frame rate of the Google Tango depth sensor, which

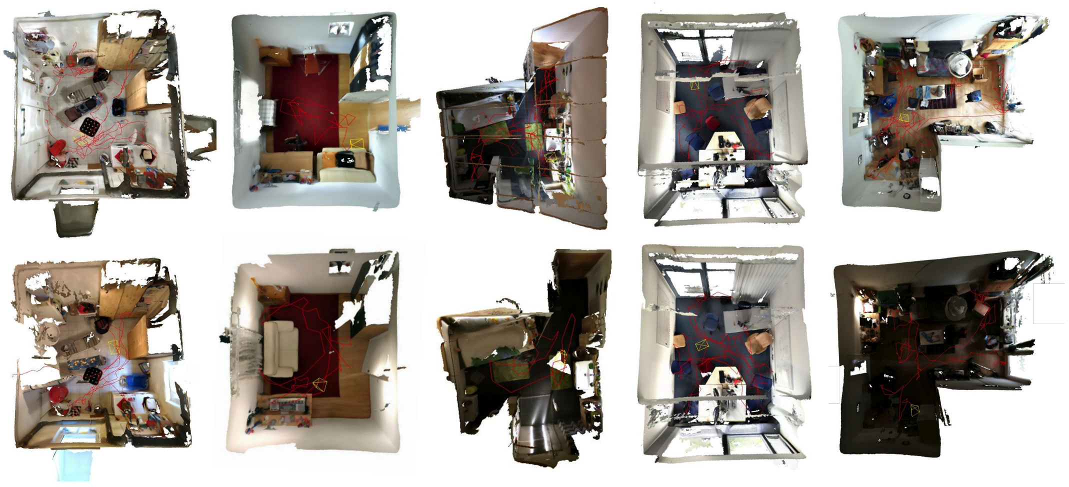


Fig. 3.13. Reference (top) and re-scan 3D models in *RIO10*. From left to right: S01, S04, S05, S07, S09.

is arguably not sufficient for many real-world applications. Based on these renderings, the sequences were re-calibrated. Due to a newly carried out temporal calibration, the trajectories of *RIO10*, compared to *RIO*, are significantly more comprehensive and therefore smoother due to the increase of frames per RGB-D sequences.

The ground truth test data is kept hidden by anonymizing the scan IDs and applying a different hidden transformation to each scene. The sequences, optimally including ground truth poses, as well as the 3D models, transformed with the scan-to-scene matrix are released on our project website[3]. Additionally, rendered 2D semantic images are provided. The semantics and instance knowledge are used to compute frame statistics in our evaluation protocol described in chapter 8.

RIO10 is one of the benchmarks used to evaluate long-term camera re-localization in the workshop *Workshop on Long-Term Visual Localization under Changing Conditions* at the International Conference on Computer Vision 2021.

3.3.3 Data Organization

The sequence data of *RIO10* follows the naming convention `seq<scene_id>_<scan_id>`. They are stored per scene together with the 3D models (`models<scene_id>.zip`) and semantics (`semantics<scene_id>.zip`) and consist of multiple sequences *e.g.* `seq02.zip` includes a folder for `seq02_01`, `seq02_02`, `seq02_03` etc. Each `seq<scene_id>_<scan_id>` stores files `*.color.jpg`, `*.pose.txt` and `*.rendered.depth.png`. `semantics<scene_id>.zip` provides 2D instance maps (`*.instances.png`) where each pixel corresponds to a unique instance in the scene. The mapping to the semantic class label is found in the file `instances.txt`. The full 3D models of each scan are available in the `models<scene_id>.zip` archive and stores textured 3D meshes (`mesh.obj`, `mesh.refined_0.png` and `mesh.refined.mtl`) as well as a semantic instance segmentation as `labels.ply`, that provides instance IDs and semantics in the custom vertex properties of the file.

[3] waldjohannau.github.io/RIO10

Tbl. 3.6. The frame statistics of the *RIO10* dataset are stored in `stats.txt`.

frame	NSSD	NCCOEFF	SemanticChange	GeometricChange	VoL	Context	PoseNovelity
`seq09_02/frame-000000`	0.571	0.531	0.312	296.868	72.369	1.580	316.253
`seq09_02/frame-000001`	0.571	0.531	0.312	296.882	72.847	1.581	316.264
`seq09_02/frame-000002`	0.571	0.531	0.312	296.995	71.599	1.583	316.413
`seq09_02/frame-000003`	0.571	0.531	0.312	297.063	73.363	1.582	316.497
...							

Sequence `seq<scene_id>_01` is always the train sequence (with `*.color.jpg`, `*.pose.txt` and `*.rendered.depth.png`) and `seqXX_02` is the corresponding validation sequence (with `*.color.jpg`, `*.pose .txt` and `*.rendered.depth.png`). Please note that ground truth (including the semantics) is not provided for our test set and all `seqXX_02+` are hidden sequences with only `*.color.jpg` and `*.rendered.depth.png` files. Please use our evaluation service[4] to compute the final performance score on the hidden test set. The poses `*.pose.txt` of the test (re-scan) sequences are aligned with the train (reference) and validation scans. These transform the camera coordinate system to the world coordinate system. The corresponding camera calibration parameters f_x, f_y, c_x, c_y of $\mathbf{K}$, see eq. 3.9, are stored in `intrinsics.txt`. Finally, statistics are provided in a file `stats.txt`, see Tbl. 3.6.

[4]`http://vmnavab26.in.tum.de/RIO10`

3.4 Monocular Depth Prediction Dataset

RIO was also adapted to benchmark 3D monocular depth estimation [207], see Fig. 3.14 and abstract from the following paper:

[207] Evin Pinar Örnek, Shristi Mudgal, **Johanna Wald**, Yida Wang and Federico Tombari. *From 2D to 3D: Re-thinking Benchmarking of Monocular Depth Prediction. under submission.*

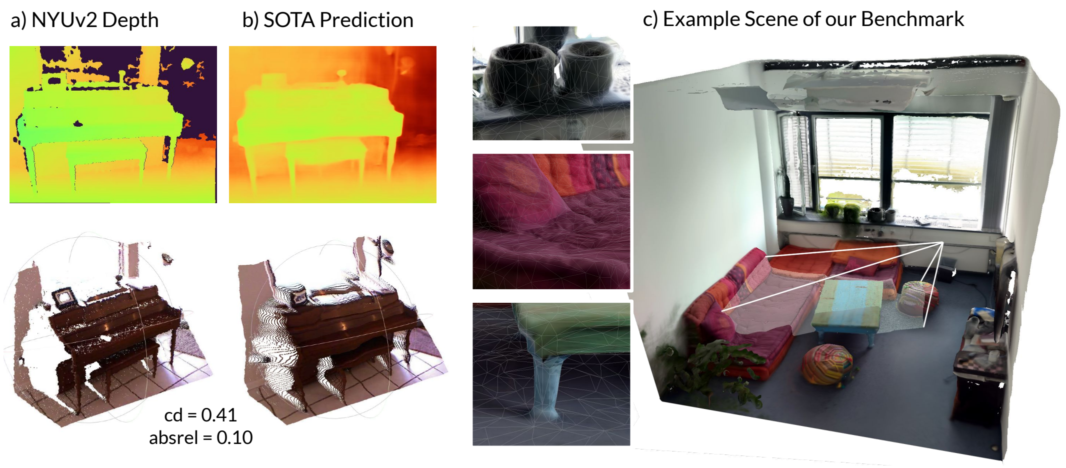


Fig. 3.14. b) State-of-the-art depth prediction with low absolute relative error often do not exhibit correct 3D structures when back-projected in 3D space. We suggest to use 3D metrics to measure the quality of the prediction and propose d) a new depth prediction benchmark with full 3D geometry and high quality depth compared to raw sensor measurements (a) [207].

Given the numerous methods recently proposed for the task of monocular depth prediction (MDP) not counterparted by an equally rapid evolution of benchmarking tools, we argue that MDP is currently witnessing benchmark over-fitting and relying on metrics that provide insufficient cues to reveal main weaknesses in predictions. In particular, this limits the design and development of novel methods that are truly aware of – and improving towards estimating – the 3D structure of the scene. In this work, we focus on indoor scenes and try to bring 3D awareness to MDP, an inherently 3D task, by analyzing how common evaluation metrics are not able to assess the quality of the 3D geometry. To this aim, we propose a novel indoor benchmark that provides full 3D geometry, semantics, normals, and occlusion boundaries, as well as a set of metrics better suited to evaluate the 3D geometry of MDP approaches. Our benchmark is based on a real-world dataset and features a high amount of high-quality rendered depth maps obtained from RGB-D reconstructions, which we demonstrate to be useful also to benchmark other relevant tasks such as 3D scene completion. In our experimental section, we evaluate different aspects such as 3D information, a variety of objects, and context. Our benchmark dataset and evaluation codes will be publicly available.

3D Semantic Scene Graphs

4

The content of this chapter is based on the following publication:

[296] **Johanna Wald***, Helisa Dhamo*, Nassir Navab, and Federico Tombari. *Learning 3D Semantic Scene Graphs from 3D Indoor Reconstructions*. *Conference on Computer Vision and Pattern Recognition (CVPR)*. © 2020 IEEE.

4.1 Introduction

While a large selection of 3D datasets exists for 3D understanding (see Sec. 3.1), scene graph research mostly focuses on the image domain. An overview about interesting graph datasets is given in Tbl. 4.1.

2D semantic scene graphs were originally introduced by Johnson *et al.* [119] where a collection of $5,000$ images with corresponding graphs were published. It included 2D bounding boxes of the objects *grounded* to different regions of the image. Attributes are further assigned to instances and (subject-predicate-object)-tuples are used to describe node connections in the graph using an open predicate vocabulary. The release of Visual Genome [136] – two years later – contributed significantly to the popularity and success of scene graph methods and applications. Visual Genome is quite large and features many different semantic relationships, including human actions such as throwing, eating, or sitting. However, the semantic connections are sometimes ambiguous, and predicates such as *on* and *standing on* are used interchangeably. While [136] also introduces attributes, research rarely includes them. Furthermore, 2D bounding boxes are provided to identify objects, and more descriptive instance masks are an often missed asset.

Tbl. 4.1. Overview of the main scene graph datasets in comparison to *3DSSG*.

Dataset	2D	3D	Data	Relationships
Johnson *et al.* [119]	✓	✗	YFCC100m [275] and COCO [166]	68 Different
Visual Genome [136]	✓	✗	108k images	diverse, semantically meaningful
3D Scene Graph [12]	✓	✓	Gibson [310]	occlusion, spatial order, relative volume
Scannet-SGG [83]	✓	✓	ScanNet v1	same set, same plane, part of, support
NYUv2 [199]	✓	✗	1449 Kinect images	vertical and horizontal support
SceneCAD [18]	✓	✓	ScanNet v2	vertical and horizontal support
***3DSSG* (Ours)**	✓	✓	*RIO* [295]	diverse, semantically meaningful

In the last decade, scene graphs have been demonstrated to be a great representation: compact yet informative and therefore suitable for many different higher-level understanding tasks. While primarily used in the image domain, we believe semantic scene graphs are an excellent encoding, especially for 3D data. To date, only a few works explored 3D scene graphs, including [83] and [12]. Gay *et al.* propose a relatively simple graph dataset based on *ScanNet* [54]. Similarly, [12] build their graphs upon the Gibson dataset [310] and propose a hierarchical 3D scene graphs representation. They define a scene as a construction of multiple levels of abstraction, starting with buildings that contain rooms, that consist of objects which link back to the camera views. Their work was later extended by Rosinol *et al.* [227] adding a dynamic layer that encodes humans too. Notably, the graphs proposed by [12] omit support relationships due to the lack of modeling structural components, such as walls or the floor. These types of relationships are prominent in the relatively small *NYUv2* [199] which is strictly speaking no 3D dataset but offers 2D semantics and RGB-D sequences, which partially enables to recover 3D geometry. Also, more recently *SceneCAD* [18] was proposed which provides support surfaces additionally to layout annotations for ScanNet [54] but does not focus on other semantic connections.

4.2 3DSSG Dataset

Given these shortcomings, we published *3DSSG*, a large-scale 3D scene graph dataset build upon the scenes in *RIO* [295]. It provides 48k object nodes with attributes and 544k relationships for the 1482 scenes. The graphs are based on existing annotations such as the 3D semantic instances segmentation, see Sec. 3.2.3, the 3D geometry, and several new annotations. Even though it was constructed from *RIO*, its creation was a great effort and involved the development of new annotation interfaces, manual labeling by expert annotators, as well as several verification procedures that required multiple passes over the data. In the following sections, we first give a formal definition of 3D scene graphs and then explain how *3DSSG* was created, covering details about the annotation process and the respective work packages.

4.2.1 Notation

The following paragraph introduces the mathematical notation as well as some terminology. Nodes, attributes, edges, predicates as well as relationships are described and their respective differences are mentioned. Please note that several related works use some of this terminology interchangeably *e.g.* relationship vs. predicate vs. edge, and while it is often more or less a matter of taste or convenience, it could potentially cause misunderstandings.

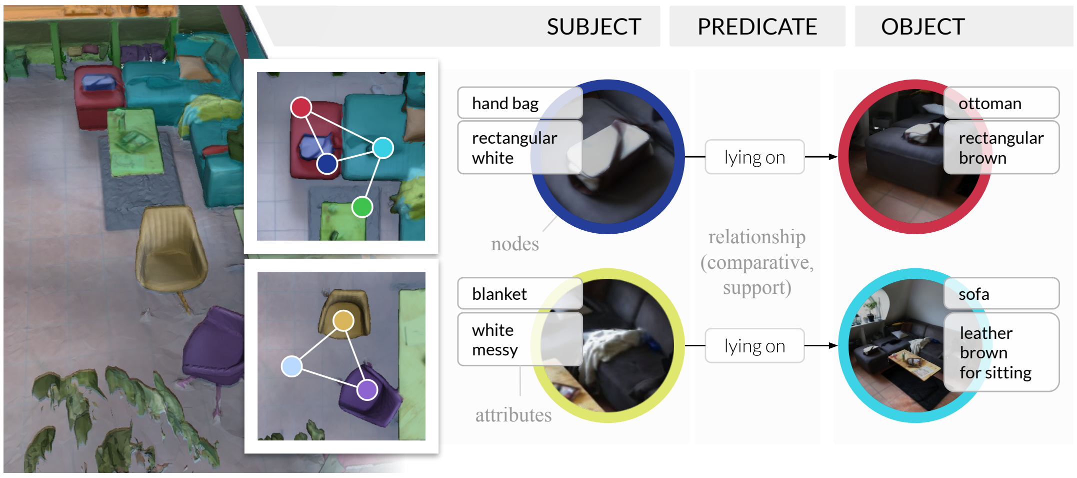


Fig. 4.1. Visualization of 3D semantic scene graphs (*3DSSG*): a graph consists of object nodes that are connected to subject nodes via directed, semantic relationships. © 2020 IEEE.

Formally defined, a scene graph is a directed graph that consists nodes and edges $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, see Fig. 4.1. A node in the graph refers to an object instance while an edge is a directed connection between two nodes, typically defined by a predicate $\mathcal{E} \subseteq \mathcal{N} \times \mathbb{P} \times \mathcal{N}$. Example predicates are *e.g.* *lying on, leaning against, higher than* or *more comfortable than* and are usually encoded as an integer that links back to a predicate dictionary. A directed semantic edge between two nodes is a relation or synonymous, a relationship tuple *subject-predicate-object* $(n_k, p, n_j) \in \mathcal{E}$ that connects $n_k \in \mathcal{N}$ with $n_j \in \mathcal{N}$. Accordingly, the edges $\mathcal{E}$ of the graph are then,

$$\mathcal{E} \subseteq \{(n_k, p, n_j) \mid n_k, n_j \in \mathcal{N}, j \neq k \wedge p \in \mathbb{P}\}. \tag{4.1}$$

As mentioned earlier, the semantic l_i and instance labels i_i of the objects n_i in the graph $\mathcal{N} = \{n_i\}_{i=1}^{\mathrm{N_N}}$ are consistent between multiple scans of the same environment.

4.2.2 Nodes

Graph nodes represent a 3D object instances in a particular 3D scene *e.g.* the *floor* or one of the *chairs* in Fig. 4.1. Each object node is unique and refers back to a 3D scene. It is usually defined by an object category as well as a bounding box or instance mask. In related works [12, 54, 136, 295], a node is defined by a single class C. In *3DSSG* however, multiple categories are assign to a single node to counteract annotation ambiguities. In order to do so, a hierarchy of classes is used such that $\mathbf{C} = (c_i)_{i=1}^{\mathrm{N_c}}$ where $c \in \mathbb{N}_1$. Notably, c_1 refers to the semantic label $\mathrm{l} \in \mathbb{N}_1$ provided in *RIO* and every c_s where $s \geq 2$ is derived from parsing the lexical definition of the first label c_1 using the WordNet dataset [73]. The label "*baby bed*" is defined as "*a small bed for babies; enclosed by sides to prevent the baby from falling*" and a bed is further defined as "*a piece of furniture*". This results in c_{n+2} = *furniture*, c_{n+1} = *bed* and c_n = *baby bed* with $n = 1$ for objects with "*baby bed*" as the initial label.

Noteworthy, some words are logically ambiguous *e.g.* chair is either a) "*a seat for one person, with a support for the back*" or b) "*the position of professor*". In an initial annotation step, valid textural descriptions were selected for all 528 class labels in *3DSSG* based on given data constraints, which is, in our case, the known indoor setup, see example above. Since the data does not feature humans, the description a) is chosen. The number of 534 class labels in the original dataset was slightly reduced to 528 by fixing typos and spelling inconsistencies. Once unique descriptions were selected, we recursively parsed these to get the final class hierarchy. A full list of descriptions with corresponding class structures is provided for each label in *3DSSG*. Fig. 4.2 visualizes a lexical tree of class connections *e.g.* $\{$*baby bed* $\rightarrow$ *bed* $\rightarrow$ *furniture* $\rightarrow$ ... $\rightarrow$ *entity*$\}$. There, the original labels of *RIO*, c_1, are colored while labels in white or without a background are more abstract categories derived from the WordNet [73] descriptions.

Finally, scene graph attributes $\mathcal{A}$ are assigned aiming to describe the physical and appearance properties as well as interaction possibilities, called affordances [311]. We categories attributes $\mathcal{A}$ into static $\mathcal{A}_S$ and dynamic properties $\mathcal{A}_D$ as well as affordances $\mathcal{A}_A$ whereas each attribute $a \in \mathcal{A}$ is assigned to only one group and, therefore

$$\mathcal{A} = \mathcal{A}_A \cup \mathcal{A}_S \cup \mathcal{A}_D, \text{ and} \tag{4.2}$$

$$|\mathcal{A}| = |\mathcal{A}_A| + |\mathcal{A}_S| + |\mathcal{A}_D| \tag{4.3}$$

holds. Since there is a large number of instances in our data, we designed a semi-automatic and efficient data annotation pipeline based on the manual selection of predefined attributes $a \in \mathcal{A}$ to avoid wrong and redundant labels, see Fig. 4.4. This annotation process is described in more detail in the following. Overall, *3DSSG* features 93 different attributes on approx. 21k object instances.

Static Properties

We introduce static properties defined as the attributes that do not change on an everyday basis, including but not limited to appearance, shape, form, size, color, texture, as well as deformation properties such as (non-)rigidity. In an initial pre-processing step, the 3D geometry of all instances is extracted and clustered by the class label c_1. Based on the data in the clusters, the relative size of an object is computed for all other objects in the same category. To avoid mistakenly including partial objects, only the largest observation of an instance – across all scans of a scene – is used in the computations. Another method to assign labels efficiently is to annotate 528 categories rather than 48k instances. This implies inherently descriptive attributes specific to all instances of a given label *e.g.* the *roundness* of a *ball*.

Contrary to this, material and shape, and textures are assigned on the instance level of each reference scan. This includes labels such as (1) *wooden, concrete* (2) *flat* or *L-shaped* and (3) *flower-patterned* or *red*. A few properties can directly be extracted from *RIO* such as the rigidity and symmetry, see Sec. 3.2.3 while others were annotated from scratch. For this purpose, an annotation interface was implemented. The interface, shown in Fig. 4.4, highlights an instance within a scene and based on the visualization an expert annotator verified and checked several listed properties, see Fig. 4.4, bottom left. Since all of the just mentioned attributes are static and therefore do not change over time, they can automatically be transferred to the new observations of the scene.

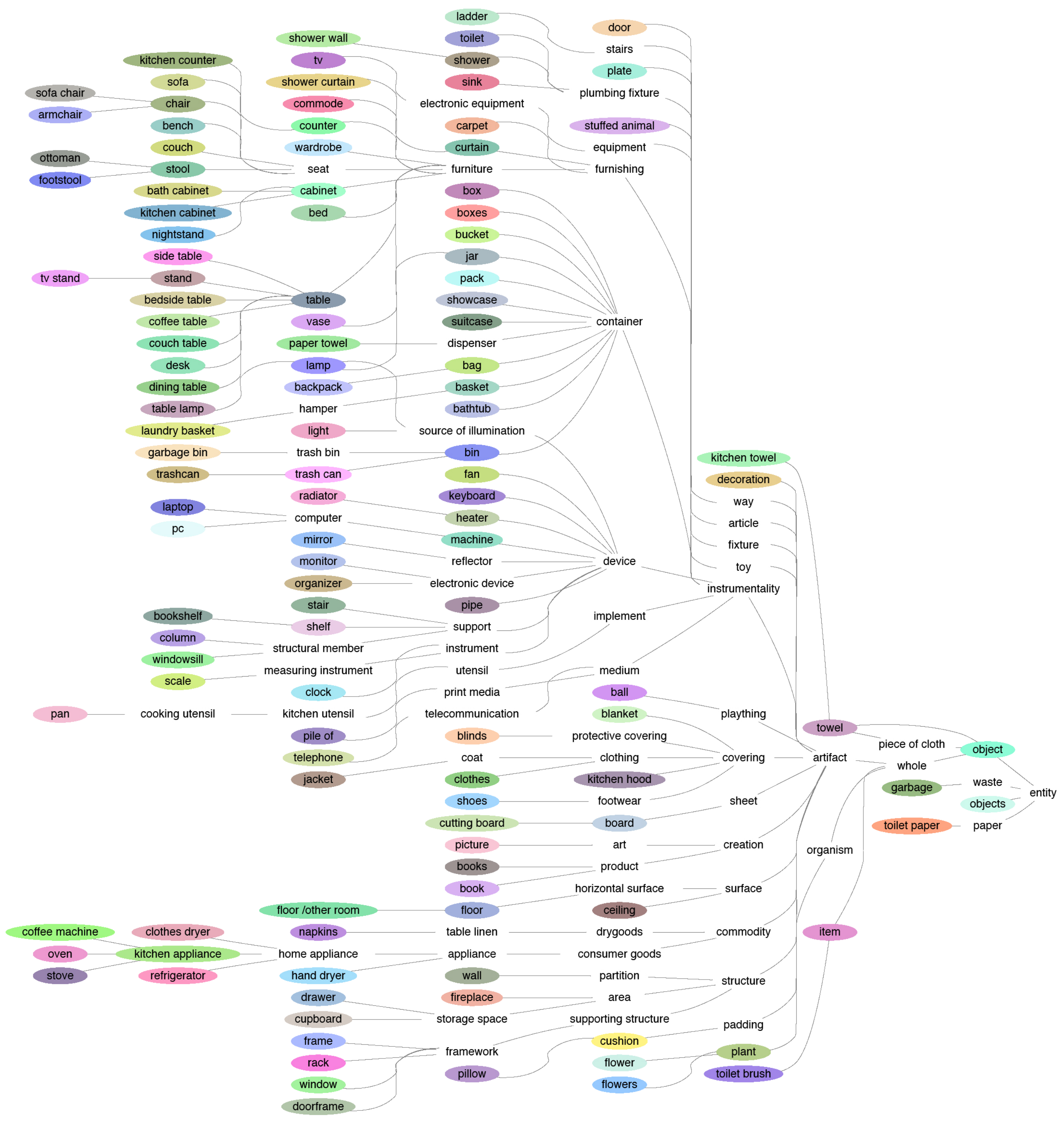


Fig. 4.2. Visualization of semantic connections between the hypernyms on some class labels of *RIO*. © 2020 IEEE.

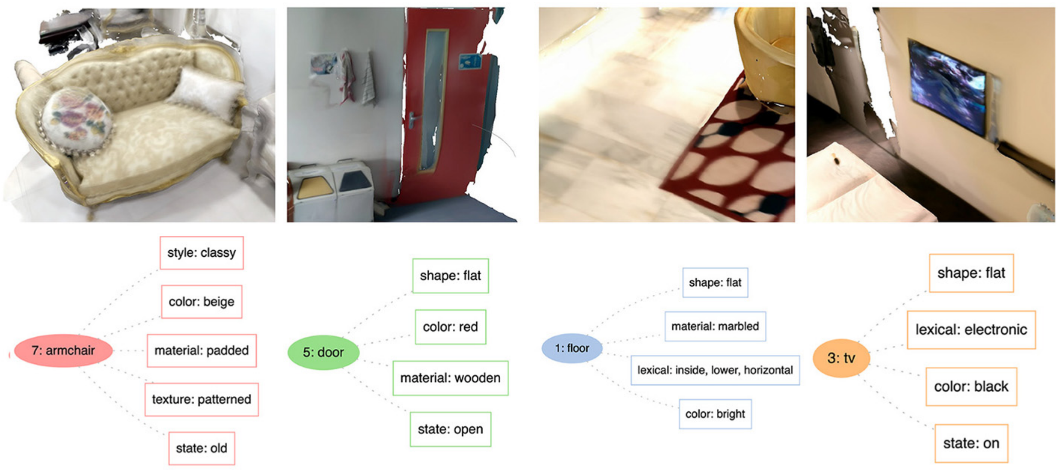


Fig. 4.3. Semantic object properties (bellow) of some example object nodes shown in the top row. © 2020 IEEE.

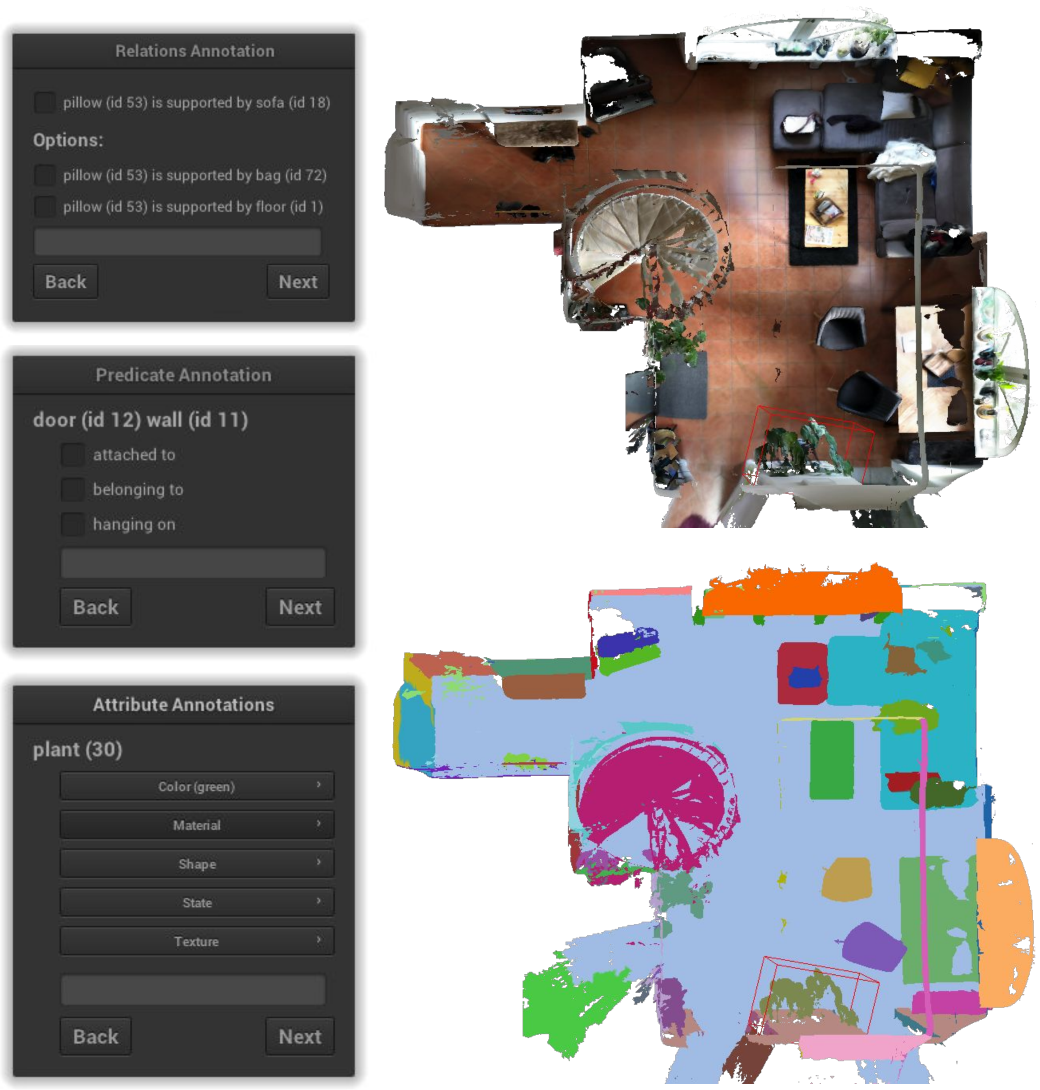


Fig. 4.4. Interface for annotating support (relations and predicate) as well as attributes with the corresponding scene as texture mapped model and instance segmentation.

Dynamic Properties

While the aforementioned static properties do not change, dynamic properties do. They mostly describe object states, such as *on/off, full/empty* or *tidy/messy* and are interestingly an indication of human activity, see Fig. 4.5. They likely change when people interact with the environment and simply live in a space. Notably, not every object can have every state *e.g.* an electrical divide can be *turned on* and *off*, a window is instead either *closed* or *open* while an oven can be both.

In a first step, all the possible object states – together with its default – are assigned to the c_1 class labels *e.g.* windows are usually *closed* but can instead also be *open* and a tv is most of the time *off* but rarely also *on*. Expert annotators then assigned applicable states to all the objects in *RIO*. In contrast to static properties that were only annotated in reference scans, this annotation was conducted on all re-scans and the references. In conclusion, Fig. 4.3 shows several examples of static and dynamic attribute.

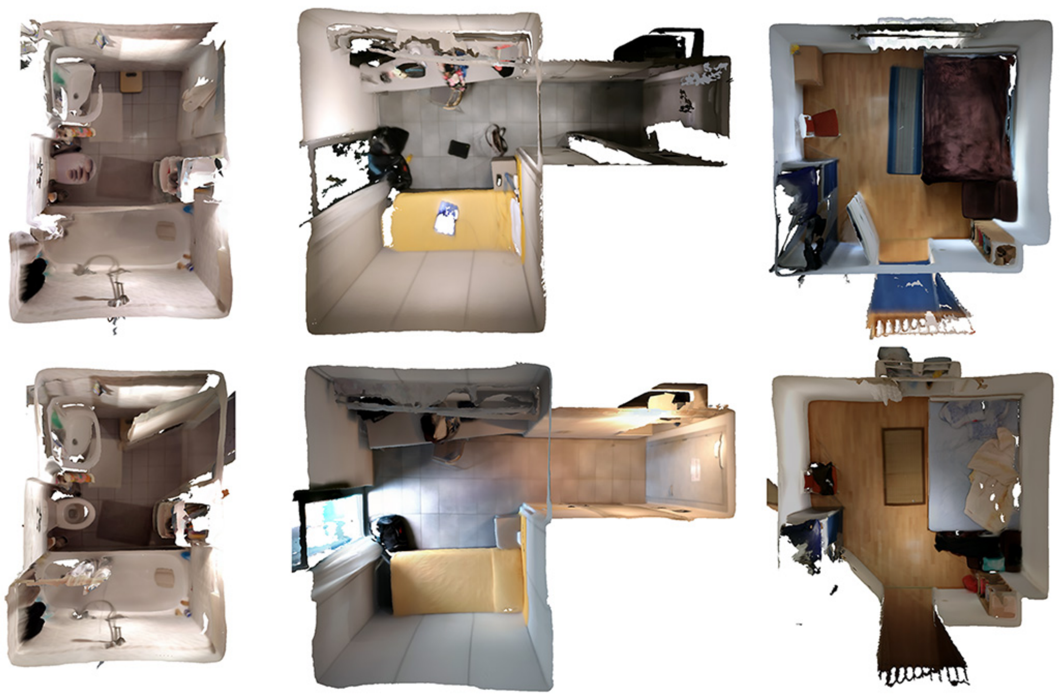


Fig. 4.5. Dynamic object properties in scenes of *RIO*. *Left:* the toilet was likely used (seat is *down/up*), *Center:* room was cleaned (*desk* and *floor* are *messy/tidy*), *right:* room was slept in (*bed* is *tidy/messy*). © 2020 IEEE.

Affordances

When referring to humans interacting with objects affordances are often mentioned. An affordance describes what "*the environment offers the individual*" [86] and has already been successfully used in several scene understanding datasets and applications [12, 311]. A simple example of an affordance is *sitting* often assigned to *chairs* or *sofas* but it also describes a valid interaction possibility with a *bed*. An interesting view on affordances is conditioning. While a door can be closed and opened in a general setup, it needs to be *open* before it can be *closed* and vice-versa.

This links the affordances to the previously described dynamic properties and potentially gives further hints about the human activity that took place and ultimately caused the natural changes in *RIO*. In a final annotation step, 70 different affordances were assigned to all 528 semantic classes with an optional state conditioning.

4.2.3 Edges

Relationships are edges that connect graph nodes via predicates and are – besides the object nodes – the second major pillar of the scene graph representation. Fig. 4.6 shows an aggregated scene graph with the most common (subject-predicate-object) relationships in the *3DSSG* dataset. A predicate can be structured into three different classes: support, proximity, and comparative. Details for each of these classes are given in the following.

Support Relationships

Support is an essential concept in artificial physical environments and has already been successfully explored in other publications, one of the most famous being *NYUv2* [199]. Its definitions are inspired by physics. The support graph of a space gives information about the steadiness of a scene; which is relevant knowledge, especially when interacting with the environment.

Each object responsible for the stability of another object is defined as a support parent, and moving it affects all of the support children. Notably, multiple support parents are legitimate *e.g.* when a guitar is standing on the floor and leans against the wall. In our theoretical support model, the floor of a room is the root node and the only instance that does not require a supporting structure. Please note that a support relation can either be from the side (*e.g.* picture hanging on the wall), from the bottom (*e.g.* sofa standing on the floor), or from the top (light hanging on the ceiling).

While it might seem straightforward to extract scene support from perfect 3D data, many challenges arise mainly due to the quality of the real-world 3D reconstructions in *RIO* caused by some noise, partial reconstructions (re-scans) as well as non-ideal annotations.

A possible heuristic to extract the support candidates is to consider the neighboring instances of all 3D points of an object within a small search radius *e.g.* $5\,\mathrm{cm}$. To do so, we define a function $f_{in}(\cdot)$, see eq. 4.5 that checks if a point is within a close neighbourhood using the euclidean distance $||\cdot||_2$, see eq. 4.4. It is used to obtain a set of instance IDs, see eq. 4.7 that are nearby the points of the object of interest as following

$$||\mathbf{p},\mathbf{q}||_2 = ||\mathbf{q},\mathbf{p}||_2 = \sqrt{((\mathrm{p}_x-\mathrm{q}_x)^2+(\mathrm{p}_y-\mathrm{q}_y)^2+(\mathrm{p}_z-\mathrm{q}_z)^2)}, \tag{4.4}$$

$$f_{in}(\mathbf{x},i) = \begin{cases} i & \text{if } ||\mathbf{p}_i,\mathbf{x}||_2 \leq r, \\ \emptyset & \text{otherwise.} \end{cases} \tag{4.5}$$

$$\mathbf{S}_r = \Big\{ f_{in}(\mathbf{x},i) \Big\}_i^{\mathrm{N_P}} \tag{4.6}$$

$$\mathcal{S} = \{\mathrm{i}_i\}_{\mathbf{S}_r} \tag{4.7}$$

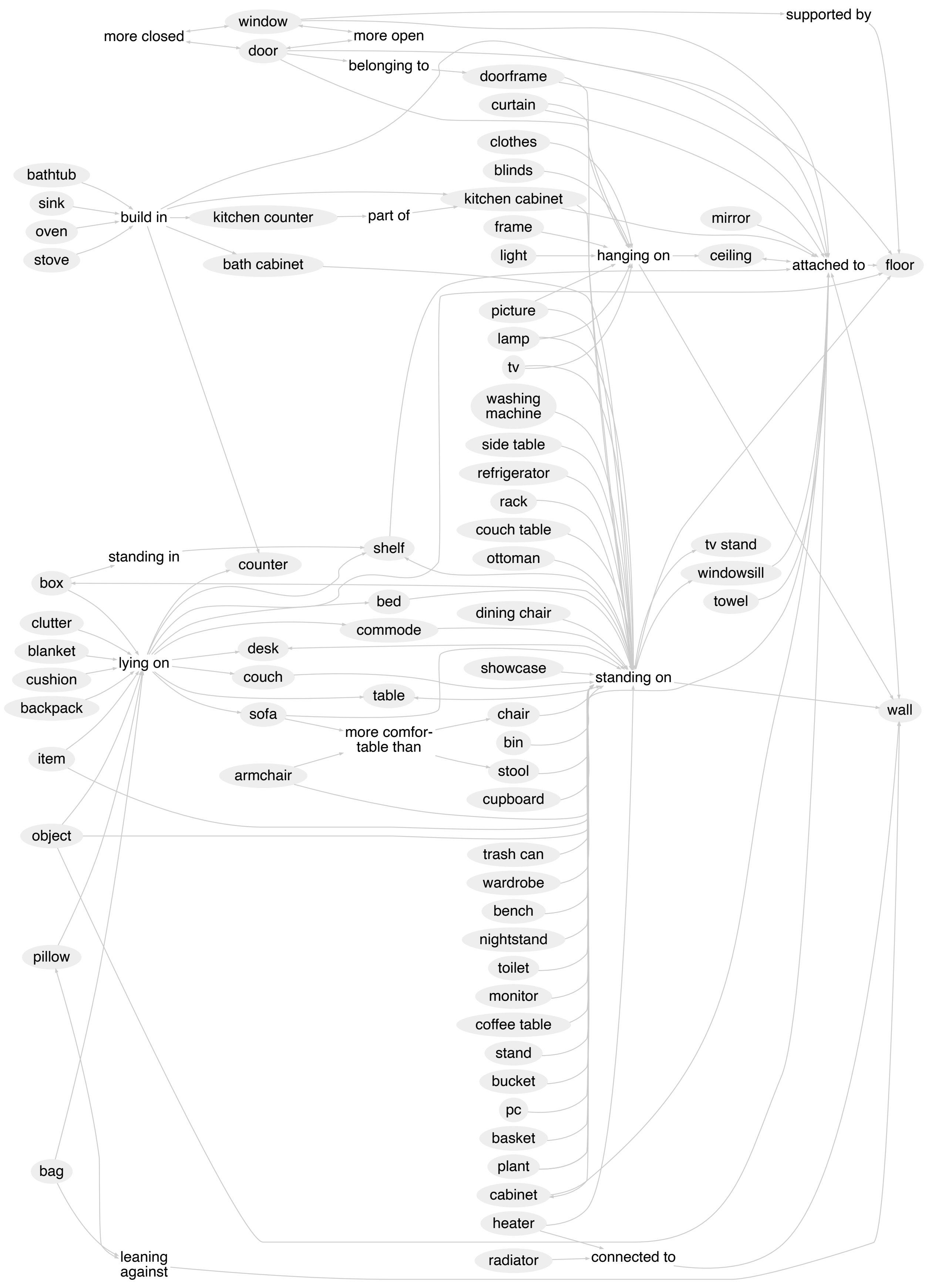


Fig. 4.6. Aggregated scene graph inspired by [119] with the most common *(subject-predicate-object)*-tuples in *3DSSG*. We only show connections that occur more than 50 times and exclude proximity as well as most comparative relationships for visualization purposes. © 2020 IEEE.

Since 3D indoor scenes are pretty complex, the resulting set of instance IDs with all possible support candidates is potentially incorrect *e.g.* a pillow next to another pillow might be incorrectly considered as its support parent. Hence, instead of using the algorithm's output directly, the automatically extracted candidates are used as an annotation basis. In a specifically designed support interface, an object is displayed with a list of support candidates to chose from. Next, the human annotator selects the instance ID of the correct support given the list of options. In some rare cases, the aforementioned algorithms miss the correct support parent, and manual addition is required *e.g.* in the case of a floating table with missing legs where the supporter, the floor, is relatively far from the supporting surface and is therefore missed. By doing so, binary support relations are obtained such as *e.g. bottle supported by table*.

Once these support connections are all verified, we move to *semantic support* relations where predicates *e.g. lying, hanging, leaning* are added to all subject-object pairs. This semantic annotation is based on the previously obtained support-edges and therefore selecting correct connections is a prerequisite before any further annotation can be conducted. This process is carried out for every support connection in the data and is required since *e.g.* a bottle can *stand* but also *lie* on a surface. Fig. 4.4 shows the support annotation interface for binary support as well as predicate selection.

Proximity Relationships

Proximity relationships give information about the spatial configuration of a room and describe how objects are positioned with respect to one another *e.g. behind, left, right*. To be able to compute proximity relationships, a reference view of the scene is required. We define *right* as the positive $+x$ direction and *behind* as the positive $+y$, consequential $-x$ and $-y$ refer to *left* and *front* accordingly.

Redundant connections are avoided by incorporating the previously described support relations. An object *on* another object inherits the proximity relationships of its parent, *e.g.* a *keyboard* on a *table* which is to the *right* of a *door* inherits the proximity relationship *right* from its parent node. Proximity relations are automatically computed based on the 3D instance mask, the object location, and the support annotations.

Comparative Relationships

Comparative relationships reflect the object's properties and are therefore the semantically richest. They are derived from attributes, see Sec. 4.2.2, including but are not limited to *more comfortable than, brighter than, the same material as*. An object is brighter or darker if it has the same color but additionally the label *dark* or *bright*, or if one of the connecting objects is either *white* and *black*. Other comparative relationships are computed similarly. Moreover, instance ambiguities from *RIO* are used to derive *same as* relationships between objects that lookalike in the same scene.

4.2.4 2D Graphs

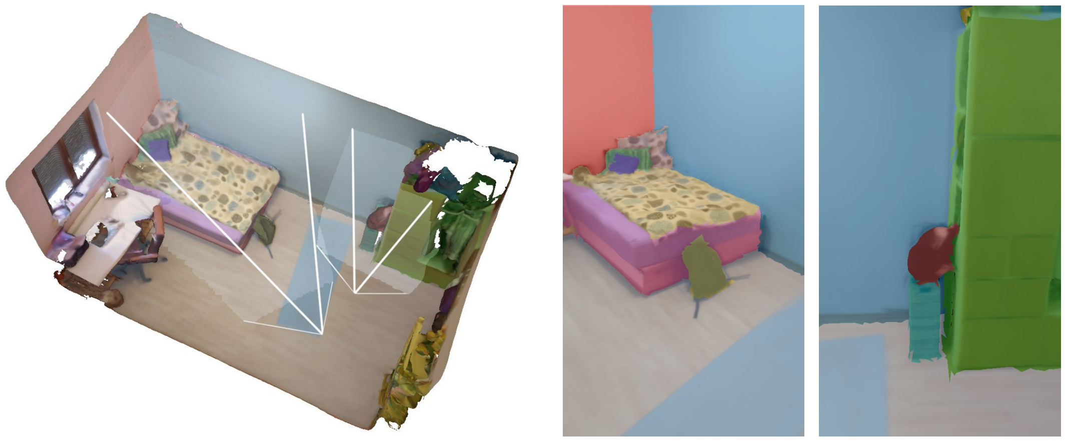
Our graphs are 3D, but can easily be mapped to any given 2D view in a process we refer to as *scene graph rendering*, see Fig. 4.7 and Fig. 4.8. To do so, a sub-graph $\mathcal{G}' = (\mathcal{N}', \mathcal{E}')$ is extracted from the 3D graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$. The new nodes $\mathcal{N}'$ and edges $\mathcal{E}'$ are generated by utilizing the 3D data $\mathcal{P}^{(\mathrm{l})}_{\mathbf{I}} = \{\forall \mathrm{l}_i \in \mathcal{P}^{(\mathrm{l})} \mid \mathrm{l}_i \in \mathbf{I}_{s,i}\}$ such that

$$\mathcal{N}' = \{\forall n_i \mid c_1^{(n_i)} \in \mathcal{P}^{(\mathrm{l})}_{\mathbf{I}}\} \tag{4.8}$$

$$\mathcal{E}' = \{\forall (n_k, p, n_j) \in \mathcal{E} \mid n_k \in \mathcal{N}' \wedge n_j \in \mathcal{N}' \wedge p \notin \mathbb{P}'\}. \tag{4.9}$$

Intuitively, this filters nodes and edges from the entire 3D scene graph given the unique instances present in an image $\mathbf{I}_{s,i}$. Since directional 3D proximity relationships, here defined as $\mathbb{P}'$, such as *left* or *right* are incorrectly transferring to 2D, they require to be updated by using the newly obtained 2D bounding box.

This way, scene graphs can be generated for any frame of the sequences in *RIO* [295]. Other 2D scene graph datasets such as Visual genome [136] provide RGB images as well as 2D bounding boxes. Additionally, 2D renderings of *3DSSG* could offer 2D instances masks computed from the 3D semantic segmentation, as well as depth maps. Another related RGB-D dataset worth mentioning is *NYUv2* [199]. It provides RGB and depth maps together with support relations but is relatively small in size.



(a) 3D Semantic Reconstruction $\mathcal{M}$ **(b)** Camera View $\mathbf{I}_i$ **(c)** Camera View $\mathbf{I}_j$

Fig. 4.7. Two *3DSSG* projections (b) $\mathbf{I}_i = \mathcal{R}(\mathcal{M}, \mathbf{P}_i)$ and (c) $\mathbf{I}_j = \mathcal{R}(\mathcal{M}, \mathbf{P}_j)$ of the 3D scene $\mathcal{M}$ a) given the camera poses $\mathbf{P}_i$ and $\mathbf{P}_j$.

4.2.5 Data Organization

In the following, an overview is given about the files and file formats used in *3DSSG*. First and foremost, the `3DSSG.zip` archive contains the main files of the dataset, listed in Tbl. 4.2.

The data about the object nodes is stored in `objects.json` as follows:

```
{ "scans": [{ ... },
    { "scan": "531cff10-0021",      // scan id in RIO
      "objects": [{
          "global_id": "6",         // global instance id from classes.txt
          "id": "1",                // id from semseg.json
          "label": "floor",
          "ply_color": "#aec7e8",   // ply color from labels.*.ply
```

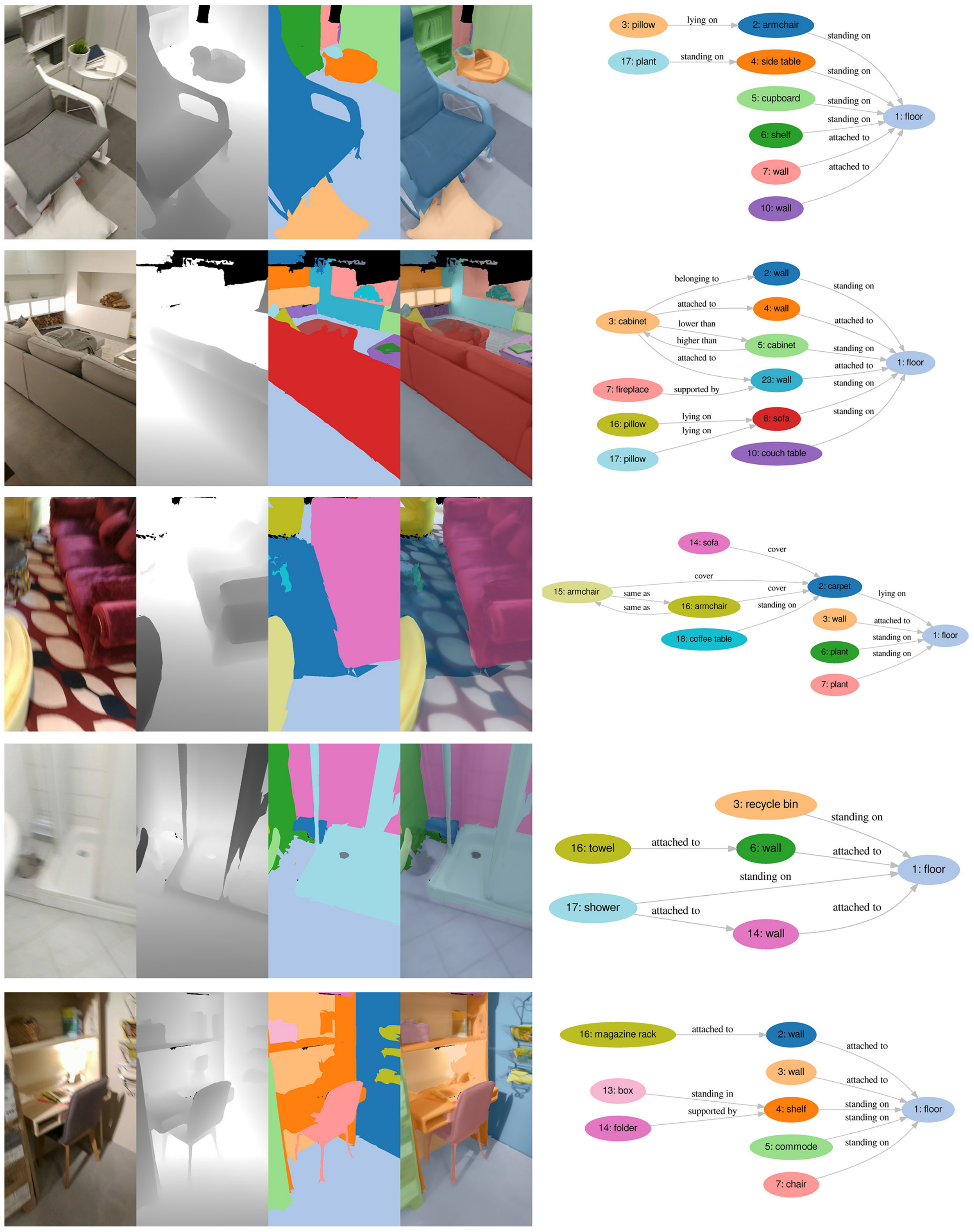


Fig. 4.8. Rendered and simplified 2D scene graphs of *3DSSG*. a) RGB image, b) rendered depth map, c) instance segmentation, d) instance segmentation on RGB rendering. © 2020 IEEE.

Tbl. 4.2. File documentation of the *3DSSG* dataset.

Filename	Description
objects.json	List of all object instances (nodes) of our semantic graphs
relationships.json	List of all semantic relationships (edges) of our semantic graphs
affordances.txt	Provides affordances for each of the 528 classes
attributes.txt	List of all 93 attributes together with their category
classes.txt	List of all 528 classes in *RIO* with corresponding lexcial description
relationships.txt	List of all 41 relationships
wordnet_attributes.txt	Extracted lexcial attributes of each of the 528 classes

```
            "affordances": ["placing items on", "walking on"],
            "attributes": {
              "texture": ["tiled"],
              "lexical": ["inside", "lower", "horizontal"],
              "color": ["brown"], "material": ["ceramic"],
              "shape": ["flat"] } },
          { "global_id": "8",
            "id": "30", "label": "plant", ... }, { ... } ]
      }]}
```

Object relationships and edges are stored in relationships.json:

```
{ "scans": [ {
        "relationships": [
          [ 8, 1,                    // subject, object id of semseg.json
            15, "standing on" ],     // id, predicate of relationships.txt
          [ 9, 1, 14, "attached to" ],
          [ 11, 14, 10, "higher than" ],
          [ 35, 34, 13, "same as" ], ... ],
        "scan": "531cff10-0021"
      }, { ... } ]
}
```

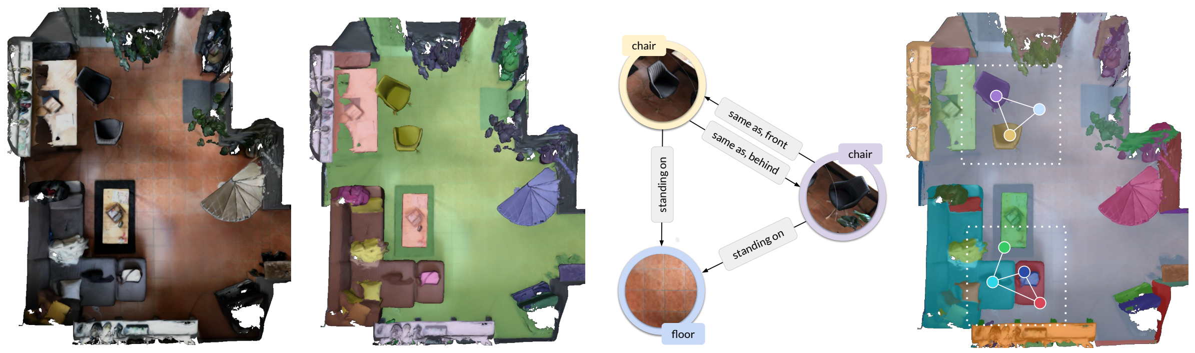
Additionally, 3DSSG_subset.zip stores files for validating and training, see Tbl. 4.3.

Tbl. 4.3. File documentation of the subset *3DSSG* dataset.

Filename	Description
relationships.json	Subset of semantic relationships used for training
relationships_validation.json	Splitted graphs with subset of edges used for validation
relationships_train.json	Splitted graphs with subset of edges used for training
classes.txt	Subset of 160 classes used for training
relationships.txt	Subset of 27 relationships used for training

Part III

Scene Semantics



3D Scene Reconstruction

3D Semantic Segmentation
Wald et al., RA-L2018, Rethage et al., ECCV 2018

3D Scene Graphs
Wald et al., CVPR 2020

3D Instances and Graphs
Wald et al., IJCV 2022, Wu et al. CVPR 2021

the purpose of abstracting is not to be vague, but to create a new semantic level in which one can be absolutely precise.

— **Edsger W. Dijkstra**
The Humble Programmer

5 Learning 3D Semantic Segmentation

The content of this chapter is based on the following publications:

[222] Dario Rethage, **Johanna Wald**, Jürgen Sturm, Nassir Navab and Federico Tombari. *Fully-Convolutional Point Networksfor Large-Scale Point Clouds*. *European Conference on Computer Vision*, 2018, Springer, Cham.

[299] **Johanna Wald**, Keisuke Tateno, Jürgen Sturm, Nassir Navab and Federico Tombari. *Real-Time Fully Incremental Scene Understanding on Mobile Platforms*. *Robotics and Automation Letters (RAL)*, presented at *International Conference on Intelligent Robots and Systems (IROS)*. © 2018 IEEE.

5.1 Introduction

An essential tool of visual scene understanding is 3D semantic (instance) segmentation. It is one of the main building blocks of many higher-level tasks as well as a wide range of applications including but not limited to autonomous driving, medical imaging, and even agriculture. The goal of semantic segmentation is to associate every input data sample with a class category given a list of predefined labels. Additionally, in semantic instance segmentation, often simply termed *instance segmentation* a unique, random identifier is assigned to each foreground object. In the 2D version, a regular grid of pixels is taken as input while in the 3D counterpart 3D points or voxels are being processed directly. Semantic (instance) segmentation aims to encode the complex geometry of a scene to ultimately extract semantic information.

The predefined list of categories depends on the underlying training dataset and its available annotations. In indoor setups, prominent classes such as walls, floor, or windows are usually included while others, less frequently found objects, such as lanterns, dish soap, or exercise balls are rarely considered. Depending on this fixed class set and the knowledge and preference of the human annotator, some labels are commonly mixed up *e.g.* table vs. desk or sofa vs. couch. And while it might be challenging for computers and humans to differentiate, misclassifications are usually acceptable due to the similarities of the involved classes and their inherent nature. The required level of abstraction mainly depends on the application area; in simple words: knowing that something is a piece of furniture or a place to sit might sometimes be enough knowledge for a given task.

Current methods often require a 3D reconstruction before segmentation and rely on computationally heavy algorithms such as deep learning. For many applications such as robotics, the processing of scene data in real-time is a hard constraint. To use the algorithms on *e.g.* mobile autonomous robots, the compatibility with typical resources available on low-powered and memory-limited hardware platforms such as smartphones or embedded computers is of high importance. Therefore, there is a growing interest in the research community and industry to develop scalable and efficient scene understanding algorithms and methods, some of which can meet real-time constraints on mobile platforms. In the following, we discuss two of our methodological contributions, namely the scalable 3D semantic segmentation methods FISS [299] and FCPN [222].

5.2 Related Work

In the following section, a general overview of 3D scene understanding is given, focusing specifically on semantic segmentation, semantic instance segmentation, and object detection. Impressive advances in parsing image data have been made in the recent years [126, 251, 291] and while 2D approaches are mentioned whenever of importance, the focus of this thesis and, in particular, this chapter is mainly the understanding of 3D data from either processed 3D reconstructions or RGB-D sequences obtained from a depth sensor.

We first mention the application of 3D descriptors (see Sec. 5.2.1), and then classify scene understanding into incremental online methods such as semantic or object-level SLAM (see Sec. 5.2.2) and offline scene understanding mainly discussing methods for volumetric data and point clouds (see Sec. 5.2.3).

5.2.1 3D Descriptors in Scene Understanding

Hand-crafted feature descriptors have a pretty long history going back to the early days of computer vision. They are designed to encode the geometry, and sometimes also the color and/or normals of the neighborhood around a selected keypoint [118, 232, 279, 283]. They have been deployed in a variety of different tasks but are most often found in 3D object recognition and pose estimation as well as point cloud registration [231] where they are usually applied in a classical correspondence matching framework using RANSAC [76].

Additionally, their worth has successfully been proved in several different SLAM frameworks [237, 271]. SLAM++ [237] propose Point Pair Features [283] to match CAD objects in a predefined database with the data of a current scene reconstruction based on ElasticFusion [303]. Similarly, Tateno *et al.* [271] evaluates a selection of different descriptors [6, 7, 233] when matching the geometric segments of the scene with a collection of scanned 3D models and can detect several objects of interest and additionally determining their 6DoF pose.

The computation of these descriptors usually requires performing the nearest neighbor search for all key points of interest, which has a complexity of $\mathcal{O}(n^2)$. Computing a feature based on a large point cloud neighborhood is, therefore, a quite costly procedure even when space partitioning methods such as kD-Trees or Octrees [184] are used. This might not be a problem when working with sparse point clouds *e.g.* from classical SLAM methods such as [59, 134] but is potentially problematic for high-resolution point clouds obtained from *e.g.* dense 3D reconstruction algorithms [114]. There, the map quickly grows by thousands of points within seconds, and running these algorithms online, even without the computation of descriptors, requires an efficient implementation *e.g.* on the GPU, and special optimization strategies [205]. In Sec. 5.3 we propose *FISS*, a method that computes a descriptor *in the loop* based on an incremental geometric segmentation of the scene.

With the success of data-driven research – especially deep learning – several learned descriptors have been proposed [216, 324] based on fully-convolutional networks [251] or, more recently, employing graph neural networks [238]. Traditionally, only very limited real-world data was available; some methods, therefore, utilize synthetic training datasets [259] and are consequently prone to be affected by a domain gap problem. Nowadays, large scale, real-world 3D dataset such as *ScanNet* [54] or the proposed *RIO* dataset, mentioned in Sec. 3.2, offer a large collection of sequences and 3D models to train data-demanding methods and many of the recently proposed learned descriptors are efficient and applicable to large scale scenes. Specifically, in [295] we learn a change invariant feature for object re-localization trained with *RIO*, see Sec. 9.3, while in [222], *FCPN*, a scalable hybrid network, is proposed that generates a spatially-aware feature descriptor given an unordered point set of a scene.

5.2.2 Incremental Scene Understanding

Online scene understanding approaches often build upon classical spatial SLAM, such as KinectFusion [202] or Point-Based Fusion [125]. A so-called semantic SLAM system localizes the camera while simultaneously constructing and updating a semantic map of an initially unknown scene. Renato *et al.* extends a dense SLAM framework by detecting and fusing planes from depth images [236]. While efficiently modeling the planar structures of a scene, their approach is unable to capture curved surfaces.

An interesting work that relies on interaction is SemanticPaint [285]. The scene is semantically annotated within an online 3D reconstruction system based on user input such as pointing and hovering over surfaces with the fingers. To run efficiently, GPU-supported voxel hashing [205] and random forests for classification are employed. SemanticFusion [182] augments the geometric data of a globally consistent 3D model by the projection of image features from a convolutional neural network (CNN) to the 3D surfel reconstruction of ElasticFusion [303]. Their method runs at *25*Hz on selected keyframes using a powerful workstation equipped with a GPU. CNN-SLAM [269] is, contrarily to all other previously mentioned approaches, a monocular method and therefore only uses RGB as input. It achieves remarkable results in reconstructing 3D geometry by combining the advantages of monocular depth prediction and 3D keypoints obtained from sparse SLAM. Similar to other fusion-based methods such as [182], a regularization technique is needed balancing potentially contradictory estimates and ensuring temporal consistency of geometry or semantics [183, 197, 269]. Progressive-Fusion [212] and FusionAware [327] followed recently, setting new standards for real-time semantic 3D SLAM.

Notably, most mentioned methods focus mainly on semantics and lack instance knowledge, unlike object-level SLAM. Some apply descriptors, see Sec. 5.2.1, that encode the geometric and visual properties of different scene regions. An example is the aforementioned SLAM++ [237] where the map consists of several known objects which are used for global map optimization and localization. Some of the previously introduced descriptors have a predefined reference frame useful for determining an objects' pose. This requires objects to be known beforehand and therefore limits the scalability [80, 96, 181, 237, 271]. This stands in contrast to the approach by Yang *et al*. Their real-time method uses a surface of revolution (SoR), which generalizes to all symmetric objects [315]. More recently, *FroDO* [150] creates 3D object-oriented maps from a RGB-D sequences by regressing a learned *DeepSDF* [209] encoding combined with a shape and pose estimation.

A common approach to meet real-time constraints is the utilization of segmentation techniques [270] which reduces the computational overhead from points or surfels to segments. In the last few years, a multitude of efficient segmentation-based methods have been proposed [96, 148, 195, 271, 299]. Many of them are based on InSeg [270], which builds a geometric segmentation of the scene by segmenting and fusing 2D depth images in a globally consistent 3D map. Li *et al*. [148] uses [270] and incrementally fuses object predictions from multiple 2D images into a global model. Tateno *et al*. extends their approach with a descriptor matching to detect and track objects [271]. Notably, two of our proposed contributions build upon [271]. Specifically, in [299] we suggest combining the geometric segmentation with an incremental descriptor and an extremely efficient semantic classification, while in [309] a graph neural network is proposed that computes and fuses scene graph predictions obtained from the geometric segmentation of the scene.

5.2.3 Offline Scene Understanding

Offline scene understanding methods operate on the full 3D volume to solve semantic segmentation [54, 69, 110, 216, 218, 222], instance segmentation [68, 108, 140], object detection [215] as well as panoptic segmentation [133, 197].

The majority of the methods can be divided into either a) volumetric models that process organized data in the form of TSDF or occupancy values or b) point-based models that process unordered point sets directly. Even though many works suggest the application of multi-view features [55, 103, 139, 217, 262], 3D capsule networks [331], point transformers [329], as well as operating on the 3D mesh directly [110]. Most relevant for this thesis are voxel- and point networks, both discussed in the following.

Voxel-based Networks

There is extensive literature on learning with volumetric representations showing impressive results [48, 54, 180, 189, 259]. A volumetric approach requires the input to be stored in a dense, ordered grid of 3D voxels of uniform size. Mapping a point cloud to an occupancy or TSDF grid [114, 259] introduces discretization, and the input is not the raw data anymore but an encoding of it.

The arguably first voxel network was VoxNet [180]. It applied 3D CNNs to the task of object recognition. Similarly, [258] proposes a network for 3D object detection using Deep Sliding Shapes which estimates 3D bounding boxes on a given 3D input. 3D ShapeNets [309] utilize a volumetric network for recognition as well as completion [309]. 3D U-Net, on the other hand, segments 3D data in a medical context [48] and SSCNet [259] tackles semantic scene completion transforming a depth image into a flipped Truncated Signed Distance Function (fTSDF) to further process it with 3D CNN. Later, semantic voxel segmentation was introduced by Dai *et al*. [54] together with the well-known *ScanNet* dataset. They provide a compact 3D baseline network that processes volumes of the scene in a sliding window fashion.

All these pioneering 3D deep learning networks used very low resolutions *e.g. ScanNet* [54] uses $5\,\mathrm{cm}$ voxels and [259] is able to process a maximum scene size of $2.26\,\mathrm{m}^3$ for semantic scene completion. Notably, most volumetric methods inefficiently deal with empty space and therefore struggle to process larger scenes at once, especially using high resolutions. To overcome memory constraints spatial partitioning strategies are proposed. Klokov *et al*. suggest Deep Kd-Networks [135] while Riegler *et al*. similarly release OctNet [223] that internally uses Octrees [184] to partition the space. Convolutions are then applied to the Octree data structure instead of the raw volume. While this method still requires discretization, they are more efficient and flexible and significantly reduces the memory footprint. They build upon the assumption that real-world 3D data is often very sparse which makes most operations in a dense 3D convolutional network unnecessary [94]. 3D sparse convolutions [47, 94] seem to be an effective solution to bypass this and have been used in our advanced scene graph prediction method described in Sec. 6.4.

Point-based Networks

A more lightweight representation compared to volumetric data are point sets. Only a few years ago, point sets were typically encoded with hand-crafted features. Classical machine learning techniques such as support vector machines (SVM) [105] or random forests [34] were used to assign certain regions of the point cloud to known semantic classes [142].

This changed in 2017 when the pioneering work PointNet [216] suggests a permutation invariant network architecture that is able to process a collection of unordered points directly.

They evaluated their method on segmentation and object detection and showed robustness to outliers and variable point density while operating on the raw point cloud data without the need to discretize. Qi *et al*. later proposed PointNet++ [218], a hierarchical point-based network that employs PointNet in a multi-scale fashion in order to capture local properties of the point set rather than just global characteristics. Although PointNet++ achieved remarkable results in various tasks, its point-based design prevents it from taking the advantages 3D convolutions provide. To be specific, PointNet++ redundantly computes and stores the context of 3D points even when there is significant spatial overlap. In their implementation, 8192 input points are sampled in the network's initial layer, which circumvents parsing large-scale point clouds. As a solution the authors suggest a 3NN-interpolation of the latent features to map semantics from the sub-sampled point cloud to the original input as well as a sliding shape approach which processes chunks of size $1.5 \times 1.5 \times 3$ meters.

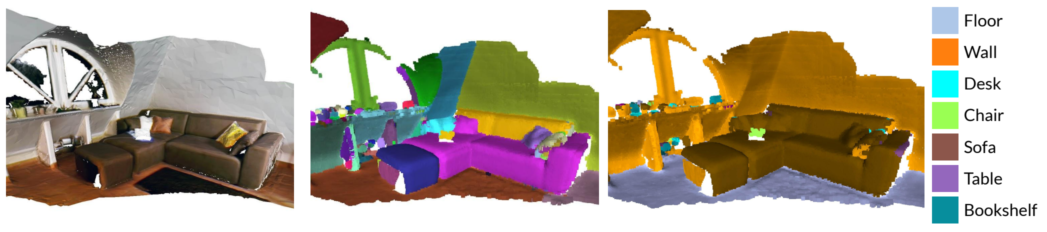
As of today, several other point-based networks were proposed such as Dynamic Graph CNNs (DGCNN) [302], ShellNet [333] or PointCNN [159]. Furthermore, PointNet is used as a low-level descriptor in several point-based networks including two of our proposed graph prediction methods [296, 308], see Sec. 6.3 and 6.6, as well as *FCPN* [222], see Sec. 5.4 which is a hybrid network positioned between volumetric and point-based.

Operating on the complete 3D data with full context and without occlusion undoubtedly has advantages proven by the state-of-the-art status of many of the methods described in this section. However, sometimes the target application requires instant feedback to be able to operate in an online matter on incoming RGB-D data [183, 226, 285]. In such setups, the online scene understanding approaches described in Sec. 5.2.2 as well as the Fully Incremental Semantic Segmentation [299], which will be discussed in detail next, are clearly the best choice.

5.3 Fully Incremental Semantic Segmentation

Fully Incremental Semantic Segmentation or short *FISS*, is – who would have guessed? – an incremental approach that reconstructs as well as geometrically and semantically segments incoming RGB-D sequences in real-time. Its main advantage is the impressive scale-ability attributed to the fully incremental nature of all involved processing steps, including a novel point cloud descriptor that can be updated efficiently as new data arrives, see Sec. 5.3.2. The descriptor is computed for all 3D segments of an online geometric segmentation, see Sec. 5.3.1, which is turned into a semantic segmentation by classifying the features into known classes *e.g*. walls, floor, chairs or sofa, see Fig. 5.1. This is done in a lightweight and efficient way by using a random forest which is described in more detail in Sec. 5.3.3. The reconstruction, geometric segmentation, segment description, and classification are all fully integrated into a surfel-based reconstruction framework [271].

A real-time runtime is ensured since the method re-uses and caches data to guarantee a constant update time at each frame, no matter how big the global map is. Since the global segmentation and encoding only process points from the newly arriving depth maps, the complexity of the method only depends on the size of the input frame rather than the size of the global map. To prove the viability of our approach, we test it using the benchmark



(a) 3D reconstruction **(b)** 3D Geometric Segmentation **(c)** 3D Semantic Segmentation

Fig. 5.1. Based on an a) incremental 3D reconstruction constructed from an RGB-D sequence of a Google Tango device, our method incrementally creates a b) geometric segmentation which we ultimately turn into a c) semantic classification. © 2019 IEEE.

dataset, *ScanNet* [54] and while the segmentation quality is comparable to other baseline methods *FISS* has the advantages of being significantly more efficient. We conduct runtime experiments on Google Tango featured devices, see Sec. 5.3.4, which proves its scalability as the map grows. A more detailed overview of our method is given in Fig. 5.2. While training

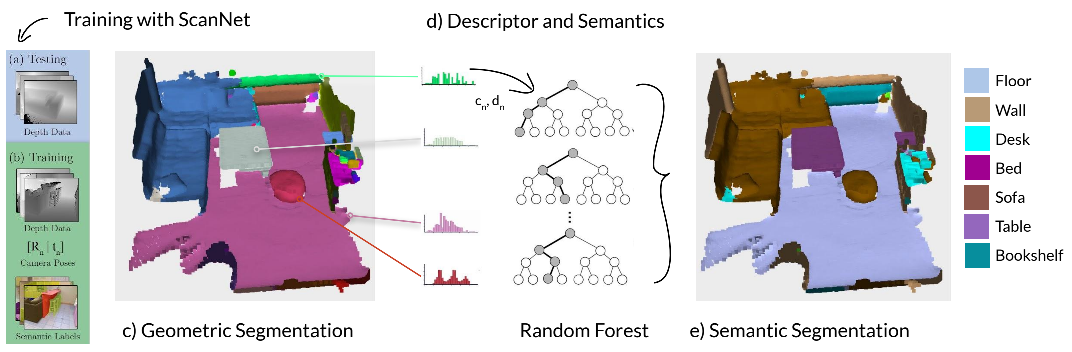


Fig. 5.2. Our fully incremental semantic segmentation (*FISS*), creates a 3D reconstruction and jointly c) segments the scene and efficiently computes descriptors for the segments. The descriptors are used with a random forest d) to predict semantic classes. The output of our method is e) an incremental semantic segmentation. © 2019 IEEE.

requires RGB-D sequences of 2D color, depth, and semantics, at test time, only depth is used specifically to construct the geometric SLAM segmentation, Fig. 5.2c. To do so, we ported and optimized Tateno *et al*. [270] for execution on mobile devices, see Fig. 5.3. The resulting 3D segments capture different regions of the scene, also visualized in Fig. 5.2c with unique colors. A descriptor is computed for each of the colored segments, which is done within the 3D reconstruction loop as mentioned before. Each updated segment descriptor is then fed into a classifier. Please note that classification is only needed for segments that have been actively affected by the data from the current depth frame. Even when the map grows – only a small set of segments change and therefore require re-classification. The classification with the random forest brings semantics into the play where, see Fig. 5.2d, descriptors and their underlying segments are ultimately turned into a semantic representation of the scene. Notably, the final e) semantic segmentation is an accumulation of the predictions, see Sec. 5.3.3.

In summary, its fully incremental pipeline enables to perceive and semantically segment large scenes in real-time without the need for special hardware. *FISS* is, therefore, an excellent choice for embedded systems such as robots or mobile phones with limited memory and computational power.

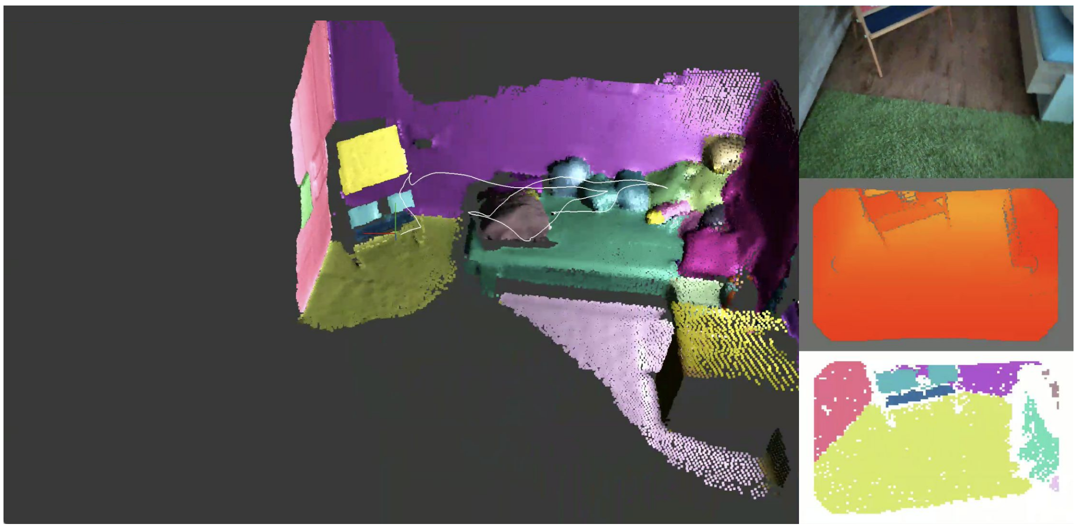


Fig. 5.3. Geometric segmentation running in a Google Tango mobile application.

5.3.1 Incremental Reconstruction and Segmentation

The first component of our framework is the incremental reconstruction which also geometrically segments the scene [270]. We utilize camera poses obtained from Google Tango's visual inertial odometry (VIO). Their estimates are pretty accurate, combining features on the fish-eye camera with sensory data from the IMU within an Extended Kalman filter framework [95]. Tango's camera re-localization is particularly reliable since it detects loop closures [250] and re-localizes when the tracking is lost. To achieve better segmentation quality, the depth maps are smoothed using a biliteral filter [278]. Together with the camera pose, a point cloud of 3D points $\mathbf{p}'$ is generated by back-projection the depth $\mathrm{d} = \mathbf{I}^{(d)}(\mathbf{u})$ for each valid pixel $\mathbf{u} = (\mathrm{u}_x, \mathrm{u}_y)$ where $\mathrm{d} > 0$ as follows

$$\mathbf{p}' = \begin{pmatrix} \mathrm{d} \cdot (\mathrm{u}_x - c_x)/f_x \\ \mathrm{d} \cdot (\mathrm{u}_y - c_y)/f_y \\ \mathrm{d} \end{pmatrix}. \tag{5.1}$$

This is repeated for several sub-sampled versions of the depth map. The normal $\mathbf{n}$ at pixel $\mathbf{u} = (\mathrm{u}_x, \mathrm{u}_y)$ is then estimated by taking the central difference of the left, $\mathbf{n}^{(-\mathrm{x})}$, right $\mathbf{n}^{(+\mathrm{x})}$, top $\mathbf{n}^{(+\mathrm{y})}$ and bottom $\mathbf{n}^{(-\mathrm{y})}$ neighbour at $(\mathrm{u}_{\mathrm{x}-1}, \mathrm{u}_\mathrm{y})$, $(\mathrm{u}_{\mathrm{x}+1}, \mathrm{u}_\mathrm{y})$, $(\mathrm{u}_\mathrm{x}, \mathrm{u}_{\mathrm{y}+1})$ and $(\mathrm{u}_\mathrm{x}, \mathrm{u}_{\mathrm{y}-1})$ as follows

$$\mathbf{n}'' = (\mathbf{n}^{(-\mathrm{x})} - \mathbf{n}^{(+\mathrm{x})}) \times (\mathbf{n}^{(+\mathrm{y})} - \mathbf{n}^{(-\mathrm{y})}) \tag{5.2}$$

$$\mathbf{n}' = \frac{\mathbf{n}''}{||\mathbf{n}''||_2} \tag{5.3}$$

$$\mathbf{n} = \begin{cases} -\mathbf{n}' & \text{if } \mathrm{n}'_\mathrm{z} > 0 \\ \mathbf{n}' & \text{otherwise.} \end{cases} \tag{5.4}$$

Based on the 2.5D point clouds and their normals, connected components are extracted and ultimately combined in a global 3D surfel map [270]. A surfel in the map is defined by a 3D location, a confidence score as well as a radius as initially introduced by Keller *et al.* [125]

$$\mathcal{S} = \{(\mathbf{p}_i, \mathbf{n}_i, \gamma_i, \mathrm{l}_i) \mid \mathrm{l}_i \in \mathbb{N}_0, \gamma_i \in \mathbb{R} \wedge \mathbf{n}_i, \mathbf{p}_i \in \mathbb{R}^3\}_{i=1}^{\mathrm{N_p}}, \tag{5.5}$$

$$\mathcal{S}_k = \{\forall(\mathbf{p}_i, \mathbf{n}_i, \gamma_i, \mathrm{l}_i) \in \mathcal{S} \mid \mathrm{l}_i = k\}. \tag{5.6}$$

Additionally, our 3D surfel map $\mathcal{S}$ stores segment IDs l_i which cluster surfels into different geometric regions $\{\mathcal{S}_1, \cdots \mathcal{S}_{\mathrm{N_s}}\}$. Notably, this segmentation is calculated using only 2D operations and is therefore entirely performed on the pixel level. Concave edges are extracted by computing the dot product between each normal vector and its neighborhood. To obtain a globally consistent map, the clusters on the depth map need to be mapped to the already perceived and stored segmentation $\mathcal{S}_k$. This is done – once again – in the image domain by simply rendering the data of the global map. Please note, only the small number of segments visible in the current frame are considered and further processed. To solve this efficiently, an index map needs to be generated. A 2D overlap factor is computed, and whenever a significant intersection is discovered in consecutive frames, the new measurements are stored in the associated segments of the global map.

Ultimately, each segment $\mathcal{S}_k$ is encoded with a feature vector $\phi_k \in \mathbb{R}^{\mathrm{N_d}}$. Merging two segments of the global map is extremely efficient since adding their descriptors is a constant $\mathcal{O}(1)$ operation, see Sec. 5.3.2, regardless of the size of the segments they encode. Finally, the geometric segments store a confidence value which is updated in the global update loop or whenever segments are combined. It helps to regulate noisy sensor measurements and is particularly beneficial when cheap mobile depth sensors are used.

5.3.2 3D Incremental Descriptor

The main contribution of our work is its fully incremental nature supported by a class of feature descriptors $\phi_{\mathcal{S}_k}$ that simply, yet efficiently encode the geometry of the scene segments $\mathcal{S}_k$. They are fully integrated into the reconstruction loop and require very little overhead, see Sec. 5.3.1.

In particular, the computation of the descriptor is linear with respect to the point cloud cardinality of the segment, a segment of size $n = |\mathcal{S}_k|$ requires n computational steps which can nicely be incorporated in a 3D reconstruction framework. This stands in contrast to other descriptors that incorporate neighborhood information where adding a single point to an existing point cloud of n elements requires up to $n + 1$ processing steps. In *FISS*, adding a point is always a constant operation and interestingly, descriptors that encodes two different point sets $\phi_{\mathcal{S}_j}$ and $\phi_{\mathcal{S}_k}$ can be combined in $\mathcal{O}(1)$. Since a re-computation and processing of the full data is never required, our descriptors are perfectly suited for runtime critical online methods.

In the following, the different incremental descriptor properties, namely a) the arithmetic mean, b) the normal vector histogram, c) bounding box and d) height encoding, are explained in more detail and while our work is only using geometry, other properties such as the distribution or the statistical variance could similarly be integrated, depending on the availability of *e.g.* color or texture.

Arithmetic Mean

The arithmetic mean of a point cloud encodes significant information of the segment $\mathcal{S}_k$. While the mean is a quite simple measure with a high value of abstraction, we show that its useful for semantic classification. The average 3D coordinates $\bar{\mathbf{p}} \in \mathbb{R}^3$ of the ceiling, floor or even a hanging lamp are quite descriptive. Given the coordinates $\mathcal{S}_k^{(\mathbf{p})} = \{\mathbf{p}_i\}_{\mathcal{S}_k}$, $\bar{\mathbf{p}}$ the arithmetic mean is formally defined as

$$\bar{\mathbf{p}} = \frac{1}{|\mathcal{S}_k|} \sum_{i=1}^{|\mathcal{S}_k|} \mathbf{p}_i. \tag{5.7}$$

Notably, each point is assigned to one unique geometric segment and updating the mean with an additional measurement $\mathbf{p}_j$ is simply done

$$\bar{\mathbf{p}} \leftarrow \frac{1}{|\mathcal{S}_k| + 1} \big(|\mathcal{S}_k| \cdot \bar{\mathbf{p}} + \mathbf{p}_j \big), \tag{5.8}$$

$$\mathcal{S}_k \leftarrow \mathcal{S}_k + \mathbf{p}_j. \tag{5.9}$$

Normal Vector Histogram

Another effective property is the distribution of the normal vectors of the point set. Normals $\mathbf{n} \in \mathbb{R}^3$ describe the surface properties of the geometric segments. A classical and well-known representation of surface normals are Cartesian coordinates

$$\mathbf{n} = (\mathrm{n}_x, \mathrm{n}_y, \mathrm{n}_z) \in \mathbb{R}^3, \text{ with } \|\mathbf{n}\| = 1. \tag{5.10}$$

Another way to represent normals is via spherical coordinates (r, θ, φ)

$$r = \|\mathbf{n}\|, \quad \theta = \text{acos}\left(\frac{\mathrm{n}_z}{r}\right), \quad \varphi = \text{atan}\left(\frac{\mathrm{n}_y}{\mathrm{n}_x}\right), \tag{5.11}$$

In our incremental descriptor we tested both representations and propose to store the normal vectors either as described in eq. 5.10 or eq. 5.11 in a histogram. The advantage of the spherical representation is the reduced dimensionality, since the length of a normal vector is always $\|\mathbf{n}\| = 1$ and can therefore be omitted. In certain setups the computation of *acos*$(\cdot)$ and *atan*$(\cdot)$ are potentially costly which can be avoided using a precomputed lookup table. Three different histogram examples are visualized in Fig. 5.4 using 20 bins respectively. When two segments are merged, their normal vector histograms are efficiently added, which is – similar to the arithmetic mean – a constant operation of $\mathcal{O}(b) = \mathcal{O}(1)$.

Bounding Box

An easy way to include the physical size of the objects is via its bounding box. Our descriptor is computed on the back-projected point clouds, described in eq. 5.1 of Sec. 5.3.1, and since the camera parameters are known, the objects in our point clouds are in actual metric space.

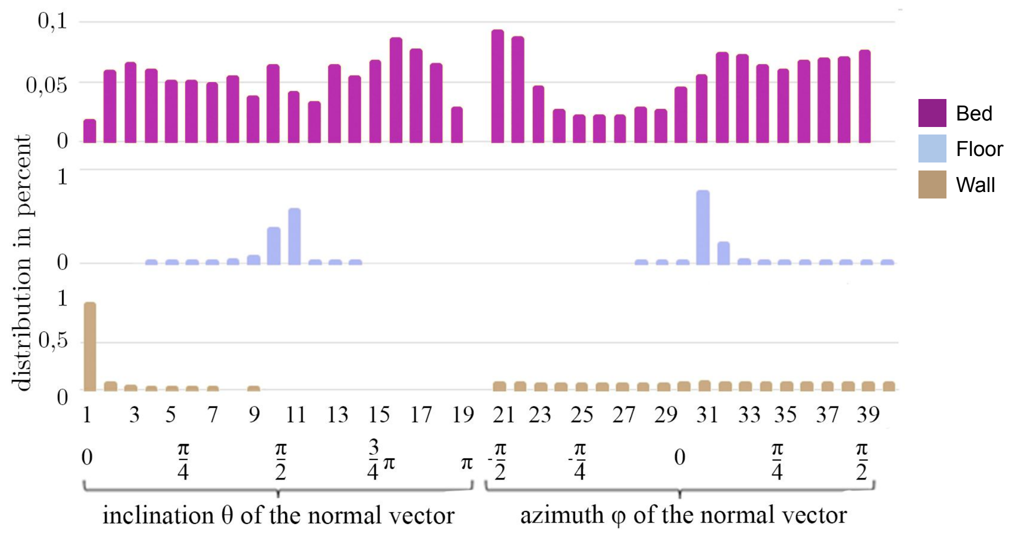


Fig. 5.4. Example histogram of 20 bins of the surface normals using spherical coordinates of an object, wall, floor. © 2019 IEEE.

The real-world physical size of an object undoubtedly provides information that – we believe – is especially useful when identifying the object categories of our interest *e.g.* floor, wall, table, chair, or sofa.

We propose to progressively keep track of the axis-aligned bounding box of each 3D segment. To do so, the algorithm needs to store the lowest $\min \mathrm{p}_i$ and highest $\max \mathrm{p}_i$ coordinate of each 3D position. A new measurement is updated as follows

$$\mathbf{b} = || \max_{\mathrm{p} \in \mathbf{S}_k^{(p)}} (\mathrm{p}_x, \mathrm{p}_y, \mathrm{p}_z) - \min_{\mathrm{p} \in \mathbf{S}_k^{(p)}} (\mathrm{p}_x, \mathrm{p}_y, \mathrm{p}_z) ||^2, \tag{5.12}$$

$$\mathbf{b} = (\mathrm{b}_x, \mathrm{b}_y, \mathrm{b}_z) \in \mathbb{R}^3 \tag{5.13}$$

which is another – surprise, surprise – constant operation.

Height Encoding

The $\mathrm{p_z}$ coordinates reveal a lot about an object and therefore are a quite abstract yet informative representation. Ceilings are always high up; the floor is the lowest flat surface, while the seatings of a chair are usually at the same height across instances.

To do so, for all points of a segment the $\mathrm{p_z}$ component is extracted and saved in a 15-bin histogram. To avoid noise, we clip all values greater than a specific value. We benefit from the fact that *ScanNet* provides normalized 3D reconstructions where the floor is being aligned roughly at $\mathrm{p}_z \approx 0$. Additionally, all the scene coordinates are moved to the positive Octant, see [54]. Such normalization is not only beneficial when learning semantics with a deep neural network but also helps simpler classification approaches. Our classifier is trained with *ScanNet* and as long as real-world scenes fulfill the same constraints, the classification will work nicely. We ensure this by normalizing the real-world Google Tango data similarly. To do so, the IMU is used for ground plane alignment. Second, a ground plane is estimated and used to normalize the 3D reconstruction. Updating the ground plane requires updating the descriptors, which is a tedious process, yet efficient.

The final descriptor is obtained by concatenation of the arithmetic mean, a 15-bin surface normal description in spherical coordinates, the bounding box data as well as the height encoding using 35 bins and a clipping value of $2\,\mathrm{m}$. The contribution of the different properties on the overall performance is analyzed in Sec. 5.3.4.

5.3.3 Semantic Segmentation

Finally, this geometric segmentation is turned into a semantic map by assigning a class label to the segments by feeding their descriptors into a regression forest classifier. Regression forests were chosen since they are highly efficient, especially when runtime constraints need to be met *e.g.* within semantic SLAM setups [285] or camera re-localization [42, 287] as they are appropriate for low-performance embedded architectures.

During training, also shown in Fig. 5.2, the depth map sequences of *ScanNet* [54] are processed with the incremental segmentation of Tateno *et al.* [270]. For convenience purposes, the provided camera poses computed via BundleFusion [56] are utilized. When constructing the training dataset, 2D semantics are back-projected and mapped to the geometric components. We use the well-known NYU20 class set proposed by [199]. The class categories are accumulated for every 3D segment. Since neither depth nor ground truth camera poses and segmentations are perfect, the projection of 2D labels to 3D might introduce model inconsistencies that require additional filtering. The filtering excludes all segments that are too small, containing too little information to determine a semantic class. Furthermore, contradictory segments need to be filtered *e.g.* when the geometric map is over-segmenting and less than 80% of the points of a segment belong to the same, most occurring category. While this rarely happens, it is important to be considered when generating the training dataset.

By following this procedure, approximately $50k$ geometric segments are generated from all training sequences of *ScanNet* [54]. To obtain a more balanced training set, lower represented classes such as refrigerators or bathtubs and toilets were additionally augmented. This way, we end up with around $10k$ training samples per class. To augment the training data further, we randomly transform the point clouds introducing more naturally occurring variations. Finally, the descriptor is computed for all geometric samples in the training set. The most occurring semantic class is the assigned ground-truth label. Please note, that our pipeline is not only fast during inference but also has a short training time; building the regression forest is a task completed in a matter of minutes.

During inference, the classifier is called for each segment that has changed since the last iteration. The descriptors belong to segments that map back to 3D surfaces, which finally turns the classification of descriptors into a 3D semantic segmentation of the underlying point cloud. The continuous predictions of the classifier are fused and refined over time. To deal with multiple, potentially contradictory estimates, a probability distribution is stored per segment. The final semantic segmentation is obtained from the most occurring class stored in the distribution. Related works have proposed similar fusion techniques but operate on the point or surfel-level, which is quite memory-intensive [182]. Since classification and fusion in *FISS* operate on the segmentation level, our method has a small memory footprint due to the relatively low amount of segments per scene.

5.3.4 Evaluation

An experimental analysis of the suggested approach is discussed in the following section. We evaluate the 3D semantic segmentation of *FISS* on *ScanNet* [54] and also report its performance on *NYUv2* [199] where our method is compared against [106, 182] and [54] using the dense semantic pixel segmentation accuracy. We analyze the effectiveness of our descriptor in our segmentation pipeline by replacing it with other global feature vectors. The experiment specifically evaluates the simple yet effective descriptor as well as the full pipeline.

Finally, we compare the runtime of our feature with other common descriptors and later analyze the performance of all the different components of our fully incremental semantic segmentation pipeline.

Fully Incremental Pipeline

Tbl. 5.1 evaluates the 3D semantic segmentation on *ScanNet* [54]. *ScanNet*'s 3D voxel semantic classification benchmark uses a subset of 20 out of 40 categories initially proposed by *NYUv2* [199]. As listed in Tbl. 5.1, these include different classes of furniture *e.g.* bookshelf, cabinet or bed, as well as major structural components such as floors, walls, doors and windows. Notably, besides *FISS* [299] we also report the performance of *FCPN* [222] which is briefly discussed in Sec. 5.4.

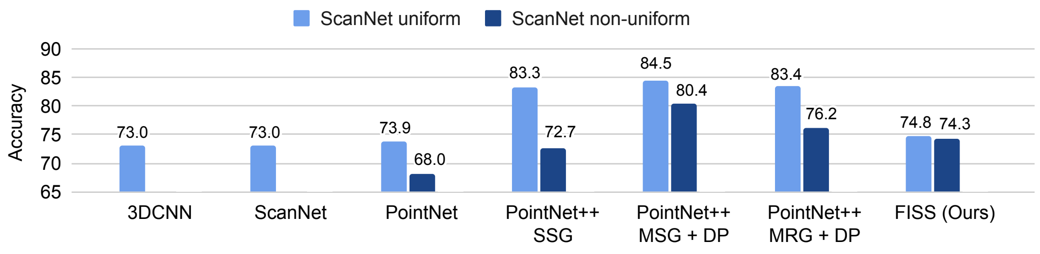


Fig. 5.5. Overall semantic segmentation performance of 3DCNN [36], PointNet [216] and PointNet++ [218].

To avoid distorted performance measures towards highly detailed, and possibly noisy, regions *ScanNet* originally evaluated on a voxel level of approximately $5\,\mathrm{cm}$ size. For a fair comparison, we sampled our 3D meshes in the same way. In contrast to the initial evaluation [54] where the average of all voxels is computed, which favors large classes such as floor and wall, we additionally report the overall class average. Notably, this is nowadays also the default measure on their public benchmark[1]. Our method achieves a weighted mean of 74.8% which is based on the great performance on large structural classes. The unweighted mean is a stricter measure and illustrates the potentially limited descriptiveness of our encoding. Arguably, it is always a balance between optimization for performance and speed. Notably, *FISS* can run on

[1]`kaldir.vc.in.tum.de/scannet_benchmark`

mobile hardware and does not require a GPU, see runtime experiments in Tbl. 5.6, at the end of this section.

Additionally, the overall semantic segmentation scores of 3DCNN [36], PointNet [216] and PointNet++ [218], weighted and unweighted are plotted in Fig. 5.5.

Tbl. 5.1. 3D semantic segmentation accuracy on the *ScanNet* validation set. Numbers for *FISS* do not directly compare since a different reconstruction algorithm was used. For, ScanComplete [57] only 6/20 classes are reported.

	Class	PointNet++ [218]	ScanNet [54]	ScanComplete [57]	*FISS* (Ours) [299]	*FCPN* [222]
🟦	Floor	97.8%	90.3%	90.2%	95.6%	96.3%
🟫	Wall	89.5%	70.1%	88.8%	93.9%	87.7%
🟩	Chair	86.0%	69.3%	60.3%	53.0%	81.6%
🟫	Sofa	68.3%	75.7%	72.5%	71.6%	76.0%
🟪	Table	59.6%	68.4%	N/A	65.5%	67.6%
🟩	Door	27.5%	48.9%	N/A	7.1%	16.6%
🟨	Cabinet	39.8%	49.8%	N/A	36.8%	52.1%
🟪	Bed	80.7%	62.4%	52.8%	50.4%	65.9%
🟦	Desk	66.7%	36.8%	N/A	30.6%	58.5%
🟩	Toilet	84.8%	69.9%	N/A	49.0%	86.7%
🟥	Sink	62.8%	39.4%	N/A	30.0%	53.5%
🟦	Window	23.7%	20.1%	36.1%	1.3%	12.5%
⬛	Picture	0%	3.4%	N/A	0.9%	1.8%
🟦	Bookshelf	84.3%	64.6%	N/A	29.2%	81.0%
🟪	Curtain	48.7%	7.0%	N/A	6.1%	6.1%
🟩	Shower Curtain	85.0%	46.8%	N/A	6.9%	48.0%
🟨	Counter	37.6%	32.1%	N/A	19.3%	31.6%
🟪	Refrigerator	54.7%	66.4%	N/A	32.0%	50.5%
🟨	Bathtub	86.1%	74.3%	N/A	34.0%	79.1%
🟦	Other Furniture	30.7%	19.5%	N/A	8.9%	30.2%
	Unweighted Mean	**60.2%**	50.8%	N/A	36.1%	54.2%
	Weighted Mean	**84.5%**	73.0%	N/A	74.8%	82.6%

The teaser, see Fig. 5.1, shows the semantic segmentation of *FISS* on a Google Tango reconstruction. Notably, our method was trained with *ScanNet* which was acquired with a different range sensor. This demonstrates the generalization capabilities of *FISS* to another indoor dataset that captures similar classes. More qualitative results on *ScanNet* sequences are shown in Fig. 5.6 and Fig. 5.7. Misclassifications are due to noise, segmentation failures, and the geometric similarities of certain classes. Small segments lack descriptiveness and are therefore extremely hard to classify. Additionally, flat objects are challenging to segment with a depth-based segmentation method which *e.g.* likely causes the low performance on the *picture* and *door* category.

Tbl. 5.2 shows the 2D semantic segmentation accuracy of *FISS* on the *NYUv2* dataset [199]. Since our method does not perform 2D data directly, the image estimates are obtained by rendering the final 3D model. While our method struggles to capture *e.g.* televisions and windows (2.9, 3.6), it performs very well on flat, structural classes such as walls, floors, and

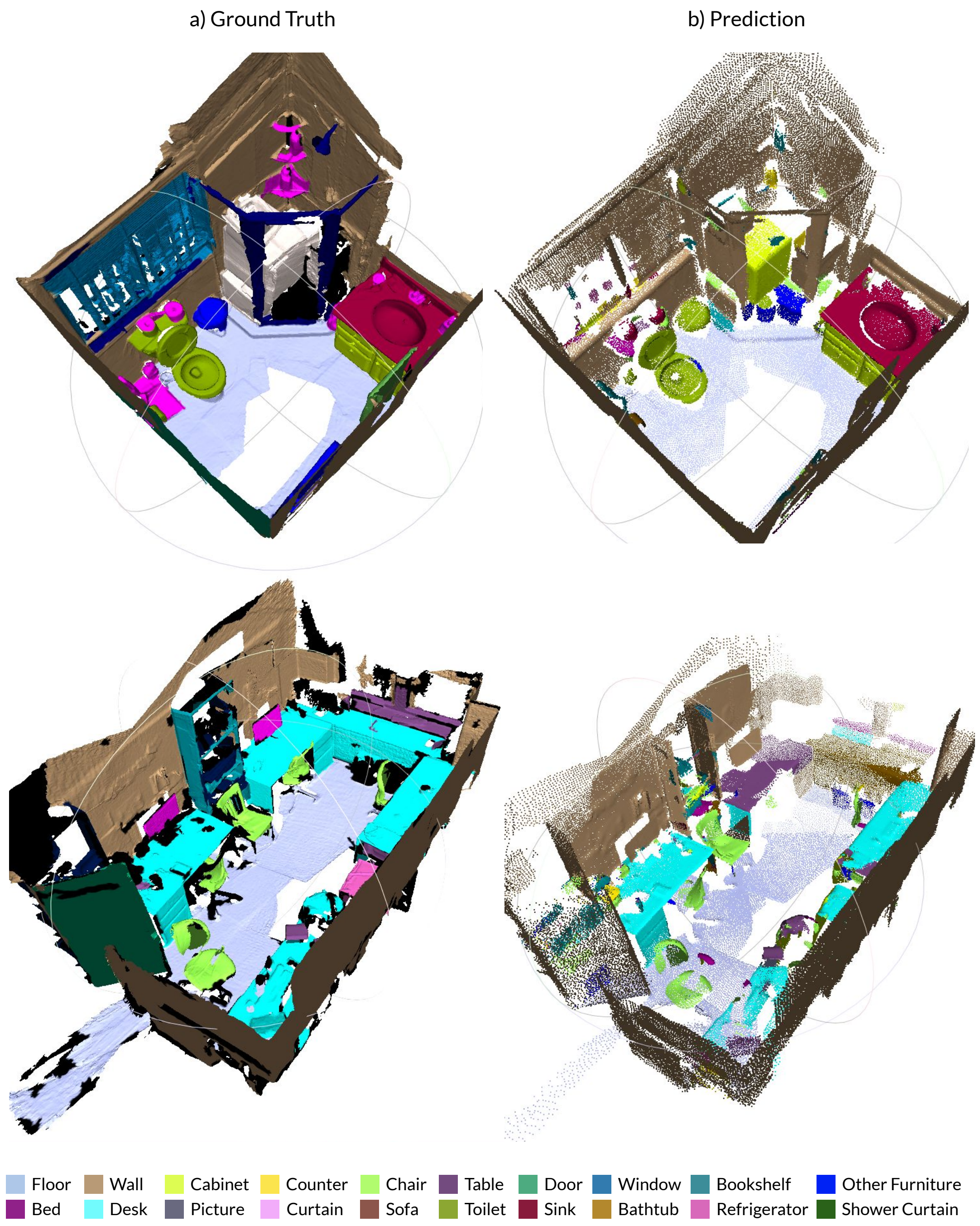


Fig. 5.6. Qualitative results of *FISS* on *ScanNet* example sequences. a) shows the ground truth semantic segmentation 3D model next to b) our predictions. Differences in 3D geometry arise due to different reconstruction algorithms used, BundleFusion [56] for the ground truth on the left and PBF/InSeg [125, 270] for our predictions.

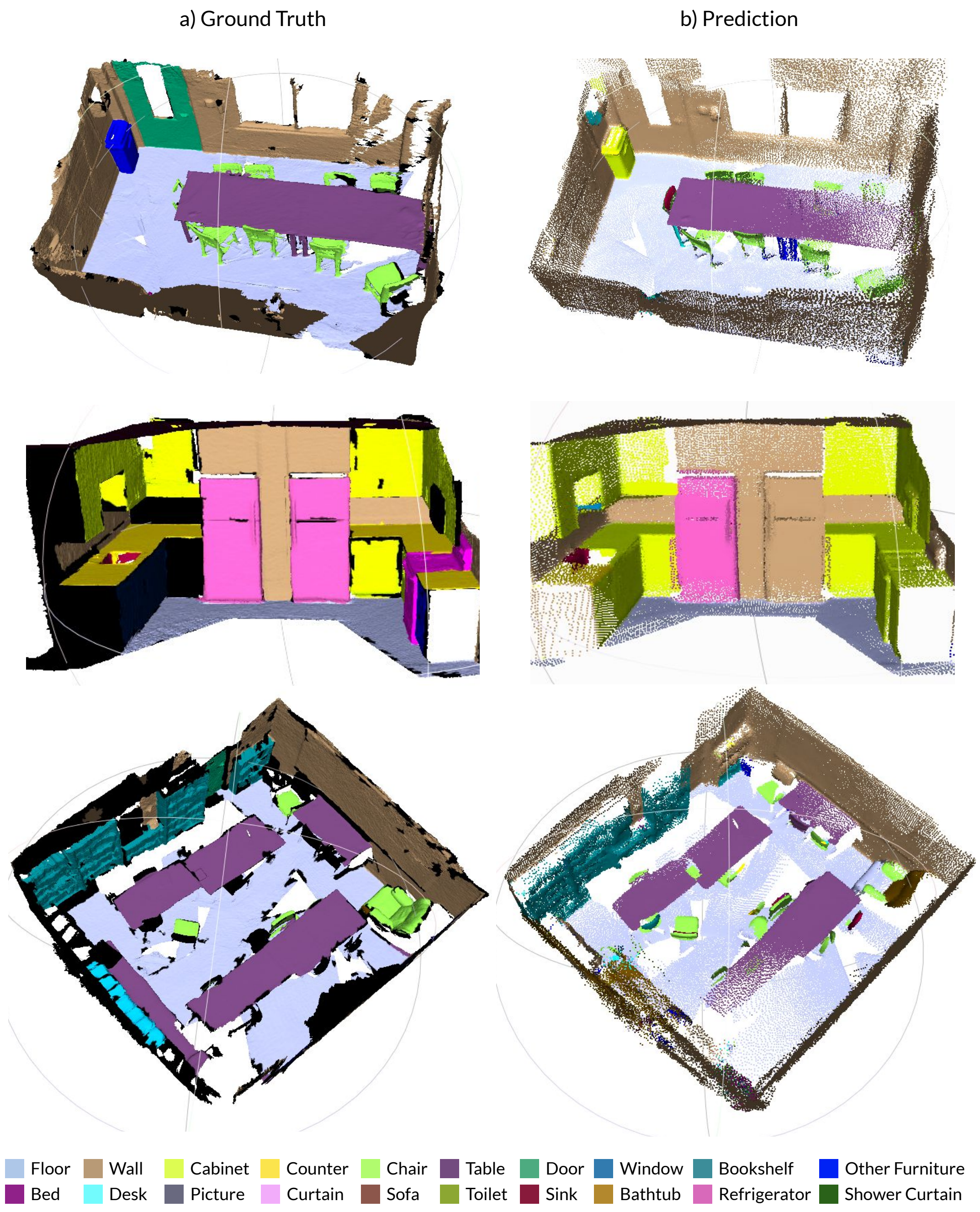


Fig. 5.7. Qualitative results of *FISS* on *ScanNet* example sequences. a) shows the ground truth semantic segmentation 3D model next to b) our prediction. Differences in 3D geometry arise due to different reconstruction algorithms used, BundleFusion [56] for the ground truth on the left and PBF/InSeg [125, 270] for our predictions.

Tbl. 5.2. Dense semantic pixel segmentation accuracy of color-methods (Hermans *et al.* [106] and SemanticFusion [182]) and geometry only (Ours and *ScanNet* [54]) on *NYUv2* [199]. © 2019 IEEE.

	Avg.	Floor	Wall	Chair	Tbl.	Window	Bed	Sofa	TV	Obj.	Furniture	Ceiling
Hermans *et al.*	49.4	91.5	71.8	41.9	27.2	46.1	68.4	28.5	**38.4**	8.6	37.1	83.4
SemanticFusion	58.2	92.6	86.0	58.4	34.0	60.5	61.7	47.3	33.9	**59.1**	63.7	43.4
ScanNet	**60.7**	**99.0**	55.8	**67.6**	**50.9**	**63.1**	**81.4**	**67.2**	35.8	34.6	**65.6**	46.2
Ours	40.6	96.3	**94.3**	48.3	33.3	3.6	23.2	21.4	2.9	12.1	23.7	**87.0**

ceilings (94.3, 96.3, 87.0). Notably, our method uses geometric knowledge only while RGB data is not being considered. Furthermore, *FISS* was not trained with any data from *NYUv2* [199] but instead only utilizes sequences of the *ScanNet* dataset. This – once again – shows how our method generalizes to other similar indoor setups.

Before we move on to ablate the proposed descriptor, a quick note on segmentation performance concerning the regression forest hyper-parameters. The depth and number of trees are analyzed in Fig. 5.8 and – as expected – the voxel prediction accuracy first rapidly increases before saturating at around 40 and 20 for the number of trees and their depth respectively.

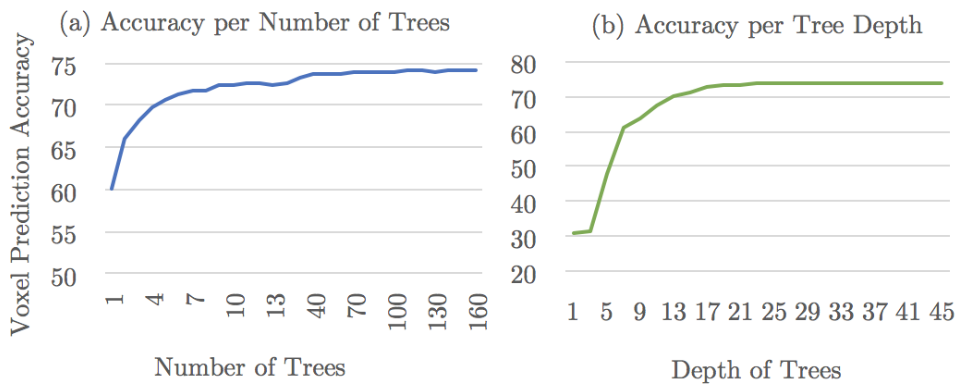


Fig. 5.8. Semantic segmentation performance for different numbers of trees t and tree depth d (with $t = 150$). © 2019 IEEE.

Incremental Descriptor

Having discussed the overall performance of our method on the semantic segmentation task, the following section analyses the effectiveness of the proposed descriptor.

Tbl. 5.3. 3D semantic segmentation using different descriptors within our incremental framework on *ScanNet* [54] validation sequences. © 2019 IEEE.

	FPFH [232]	VFH [233]	CVFH [7]	OUR-CVFH [6]	Ours [299]
Point Accuracy	7.4%	59.5%	61.4%	60.9%	**63.8%**
Unweighted Voxel Accuracy	23.2%	25.9%	28.2%	28.0%	**36.1%**
Weighted Voxel Accuracy	20.4%	53.8%	55.3%	55.3%	**74.8%**

Tbl. 5.3 lists the average 3D semantic segmentation accuracy on *ScanNet* [54] compared to *FPFH* [232], *VFH* [233], *CVFH* [7], *OUR-CVFH* [6]. An implementation of these descriptors is

available in the point cloud library (PCL)[2]. Following other comparative evaluation studies [8, 192] our experiments use their default parameter configurations[3]. The proposed incremental descriptor reaches a classification accuracy of 74.8%, which is a remarkable increase of $+19.5\%$ over the second best encodings, *OUR-CVFH* [6] and *CVFH* [7] (55.3%) while being significantly faster, see Fig. 5.9.

Tbl. 5.4. Evaluation of the effect of different feature components. © 2019 IEEE.

Without Descriptor Property	Weighted Accuracy	Difference	Mean Accuracy	Difference
Arithmetic Mean	73.8%	-1.0	33.8%	-2.3
Normal Vectors Inclination	72.9%	-1.9	29.8%	-6.3
Normal Vectors Azimuth	73.9%	-0.9	33.6%	-2.5
Segment Size	68.4%	-6.4	19.4%	-16.7
Mean Euclidean Normal	73.8%	-1.0	33.0%	-3.1
Mean Inclination	73.9%	-0.9	33.2%	-2.9
Mean Azimuth	74.2%	-0.6	33.4%	-2.7
Height Distribution	71.9%	-2.9	29.9%	-6.2

In the next experiment in Tbl. 5.4 we ablate the different components of the descriptor by deducting them one by one. Compared to the arithmetic mean and the normal representations, the size of the segment seems to have the strongest influence on classification accuracy along with the height distribution.

Scalability

In the final set of experiments, the runtime and scalability of our method are analyzed. We chose a relatively weak hardware platform for run time comparison: a 2017 MacBook Pro with 16GB RAM and 2.3 GHz Intel Core i5. We then added 1000 points one at a time to a 3D point cloud, mimicking an online 3D reconstruction pipeline. Fig. 5.9 shows how the computation time behaves for different descriptors as the input point cloud increases in size – do not miss the logarithmic scale!

Unsurprisingly – compared to the non-incremental classical descriptors – our descriptor does not scale *at all* as the map grows in size. Its computation is significantly faster with an average runtime of $100\mu s$, which is 120 times faster than *VFH* [233], the second-fastest option. When analyzing the time needed to add a single point to an existing point cloud, the difference is most significant. Also, merging two feature vectors does not require iterating over all of the points in the point sets.

Our incremental descriptor does not only convince by an excellent run time but also with great performance within the semantic segmentation framework, see Tbl. 5.3. This is likely since other existing feature vectors were designed for object recognition rather than semantic segmentation.

An overview of the run time of the entire framework is given in Tbl. 5.5. As mentioned earlier, the run time of reconstruction and semantic classification mostly depends on the input

[2]`https://pointclouds.org/`
[3]`https://pointclouds.org/documentation/tutorials/`

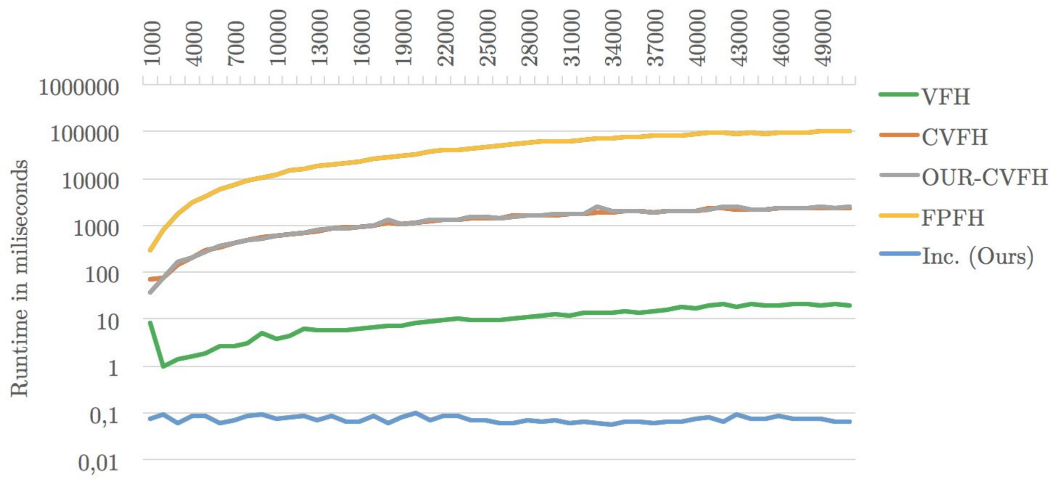


Fig. 5.9. Run time of the global feature descriptors *VFH* [233], *CVFH* [7], *OUR-CVFH* [6] and *FPFH* [232] on point clouds of increasing size. Notability, the y-coordinate scale is logarithmic. © 2019 IEEE.

Tbl. 5.5. Run time analysis of our proposed semantic classification framework running on a Lenovo Phab 2 Pro, a mobile Google Tango device, using different input resolutions. © 2019 IEEE.

Depth Map Resolution	224 × 117 (Original)	112 × 59 (Subsampled)	56 × 30 (Subsampled)
(1) Pre-Processing	104.50 ms	28.13 ms	10.03 ms
(2) Global Model Rendering	72.34 ms	21.91 ms	4.65 ms
(3) Global Model Update	116.64 ms	28.73 ms	5.16 ms
(4) Depth Map Segmentation	45.50 ms	10.86 ms	2.75 ms
(5) Label Propagation	11.47 ms	3.51 ms	0.66 ms
(6) Semantic Classification	34.11 ms	10.49 ms	2.61 ms
Total	384.57 ms	103.64 ms	25.87 ms

resolution of the depth map. We, therefore, analyze different image resolutions. Notably, specific operations within the geometric reconstruction algorithm, such as merging segments and highly efficient feature computation, are part of the global model update (3) and therefore not explicitly listed. In summary, our method runs in either 10 or 39 FPS depending on the depth map resolution. Since the Google Tango range sensors provide depth data with a frequency of 5 Hz our algorithm can process the data as it arrives and down-sampling is not strictly required. It is however recommended when using a structural sensor with a higher resolution *e.g.* as present in the sequences of the *ScanNet* dataset.

Notably, our method is way more efficient than other fast semantic segmentation methods such as SemanticFusion [182], PointNet [216] or PointNet++ [218], see Tbl. 5.6 since it is – and that should not be news to you – highly efficient and scalable, even when the map increases in size.

Tbl. 5.6. Run time of our proposed incremental semantic segmentation compared to PointNet, PointNet++ and SemanticFusion. © 2019 IEEE.

Method	Run time	Hardware
SemanticFusion [182]	39.5 ms	i7-5820K 3.30GHz, Nvidia Titan Black
PointNet [216]	25.3 ms	GeForce GTX 1080
PointNet++ (SSG) [218]	82.4 ms	GeForce GTX 1080
PointNet++ (MSG) [218]	163.2 ms	GeForce GTX 1080
PointNet++ (MRG) [218]	87.0 ms	GeForce GTX 1080
Ours (*FISS*)	25.9 ms	Mobile Phone (Lenovo Phab 2 Pro)

5.4 Fully-Convolutional Point Networks

The following section briefly describes another semantic segmentation approach that operates on full 3D indoor reconstructions. The method is called Fully-Convolutional Point Network (*FCPN*), see Fig. 5.10. Even though it is an offline method, it is quite efficient and scale-able.

The main contribution of the method is its hybrid architecture. It is positioned between volumetric [54, 259] and point-based [78, 216, 218] methods since the input of *FCPN* is an unordered set of 3D points while the internal network follows an ordered structure using classical 3D convolutions. This makes it the first fully convolutional network that can process large-scale 3D point sets. Its scalability allows it to process large regions of $80\,\mathrm{m}^2$ with up to $200k$ points in a single pass while related methods such as *e.g.* PointNet++, propose to apply sliding window techniques using point clouds with less than $10k$ points. Notably, training is still done on scene chunks of *e.g.* $2.4 \times 2.4 \times 2.4$ meters.

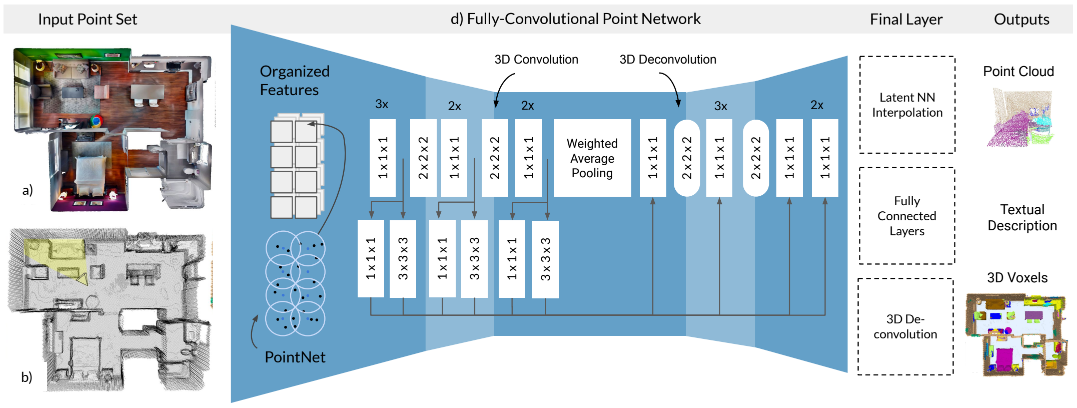


Fig. 5.10. The a) textured Tango 3D model (not used) and its 3D point set b) which is fed into of the Fully Convolutional Point Networks (*FCPN*) [222] to output a 3D semantic segmentation either as a voxel grid or a point cloud or optionally even a textural description of the scene.

During inference, the method produces either volumetric data or point sets as an output. The learned feature is spatially ordered; it can therefore also perform tasks such as 3D captioning [222] where textural descriptions are produced for a given 3D scene. The produced feature is not global but instead spatially aware and therefore transformation variant. The different outputs of the network are obtained from either latent feature interpolation, which maps volumetric features back to 3D points or – when scene captions are learned – by applying a set of fully connected layers to ultimately obtain the textural description.

Unordered point sets are processed by locally applying a simplified PointNet [216]. This is a similar strategy as in PointNet++ [218], but instead of furthest point sampling and grouping, a uniform sampling strategy is applied. This way, the unordered point set is transformed into a regular structure resulting in an ordered regional descriptor that can further be processed with higher-level abstraction layers, specifically 3D convolutions, to learn higher-level semantic features on larger scales.

Each abstraction layer of our proposed network architecture has a kernel size and stride of 2. Furthermore skip connections are used inspired by 3D U-Net [48], see Fig. 5.10. We additionally propose a pooling layer with a novel weighted-averaging scheme to incorporate additional spatial context. Given a $(2.4\,\mathrm{m})^3$ volume, several 3D convolutions turn the original point set into 8 abstract features – so-called Octants – which are then accumulated using a cost-effective pooling function which weights the features of cells of around $1\,\mathrm{m}$ distance the highest.

Even though the main target application of the network is scalable scene understanding, the network generalizes to smaller scales *e.g.* depth maps or even small-scale tasks such as semantic part segmentation, see Tbl. 5.7. There, *FCPN* outperforms other methods in $12/16$ classes on the popular part segmentation benchmark proposed by [318].

Tbl. 5.7. Part segmentation results on ShapeNet compared to the state of the art. *FCPN* (Ours) performs best in 12 / 16 categories.

	Yi *et al.* [160]	SSCNN [319]	PointNet [216]	PointNet++ [218]	*FCPN* [222]
Airplane	81.0	81.6	83.4	82.4	**84.0**
Bag	78.4	81.7	78.7	79.0	**82.8**
Cap	77.7	81.9	82.5	**87.7**	86.4
Car	75.7	75.2	74.9	77.3	**88.3**
Chair	87.6	90.2	89.6	**90.8**	83.3
Earphone	61.9	**74.9**	73.0	71.8	73.6
Guitar	92.0	93.0	91.5	91.0	**93.4**
Knife	85.4	86.1	85.9	85.9	**87.4**
Lamp	82.5	**84.7**	80.8	83.7	77.4
Laptop	95.7	95.6	95.3	95.3	**97.7**
Motorbike	70.6	66.7	65.2	71.6	**81.4**
Mug	91.9	92.7	93.0	94.1	**95.8**
Pistol	85.9	81.6	81.2	81.3	**87.7**
Rocket	53.1	60.6	57.9	58.7	**68.4**
Skateboard	69.8	82.9	72.8	76.4	**83.6**
Table	75.3	82.1	80.6	**82.6**	73.4
Average	81.4	84.7	83.7	**85.1**	84.0

Results for large scale semantic segmentation on *ScanNet* [54] have been reported earlier in Tbl. 5.1. Additional results on the hidden test set of *ScanNet* according to their benchmark are

[4] http://kaldir.vc.in.tum.de/scannet_benchmark

Tbl. 5.8. IoU on the hidden test set of the *ScanNet* Benchmark[4].

Average	Bathtub	Bed	Bookshelf	Cabinet	Chair	Counter	Curtain	Desk	Door	Floor	Other	Picture	Fridge	Shower	Sink	Sofa	Table	Toilet	Wall	Window
44.7	67.9	60.4	57.8	38.0	68.2	29.1	10.6	48.3	25.8	92.0	25.8	2.5	23.1	32.5	48.0	56.0	46.3	72.5	66.6	23.1

listed in Tbl. 5.8. Its performance is similar to the state of the art but *FCPN* is significantly more scalable.

It can parse large-scale scenes at once instead of splitting and merging predictions of different chunks, *e.g.* as done in PointNet++ [218]. Qualitative results of *FCPN* can be found in Fig. 5.11. It nicely segments common categories such as floor, wall, chairs, and cabinets and correctly captures the refrigerator, sink, and bathtub.

To prove the method's efficiency, an analysis of memory and run time has been carried out. To do so, the inference speed of the network running on a Titan Xp GPU was measured on different input point cloud resolutions. $150k$ points on $80\,\mathrm{m}^2$ surface area takes $9.1\,\mathrm{s}$ and $9033\,\mathrm{MB}$ to process, $36k$ in $36\,\mathrm{m}^2$ requires $2.90\,\mathrm{s}$ and $8515\,\mathrm{MB}$ while $15k$ on $16\,\mathrm{m}^2$ run in $0.57\,\mathrm{s}$ using $6481\,\mathrm{MB}$. Our method can still process this sheer amount of data in a reasonable time, and scaling the point cloud $5-10$ times resulted in an upper memory usage limit of only around 40%.

5.5 Conclusion

In the previous Sections 5.3 and 5.4 we have presented two scalable methods for 3D semantic segmentation. While *FCPN* [222] is an offline method, *FISS* [299] runs online and operates directly on RGB-D sequences.

[299] is an incremental approach that accumulates data in a global map running in real-time on embedded architectures. The efficiency is based on the fully incremental nature of all processing steps, including the feature vector computation. We have shown that our proposed feature encoding outperforms other descriptors within our semantic segmentation framework, analyzing the run time and semantic segmentation accuracy. Even though the descriptor has a high level of abstraction, it might be transferable to other applications *e.g.* object recognition. The overall framework is way faster than other semantic segmentation methods making it a potential baseline for other higher-level tasks and applications in AR/VR or robotics.

In [308] we proposed an incremental scene graph prediction method called *SceneGraphFusion*, which is based on two similar ideas a) it uses the same underlying geometric segmentation to pre-segment the scene and b) it computes a simple, yet effective hand-crafted feature from the segmentation. It applies these concepts in a more modern setup using a graph neural network [308], and while it can achieve state-of-the-art results in semantic and panoptic segmentation, it is – similarly to *FISS* – extremely efficient. *SceneGraphFusion* will briefly be discussed in Sec. 6.6.

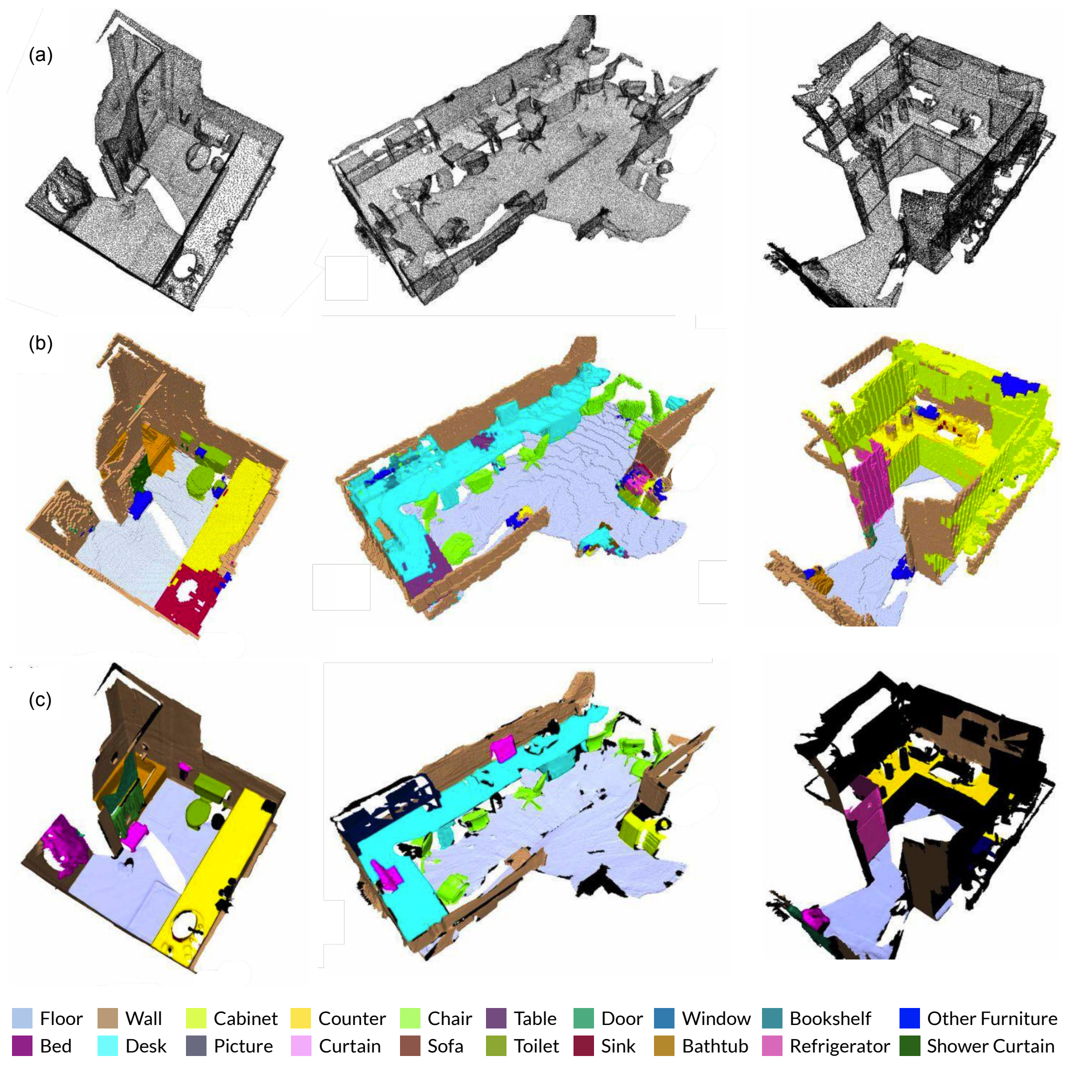


Fig. 5.11. Qualitative semantic segmentation result of *FCPN* [222] on the *ScanNet* v1 validation set. a) Input point set, b) predicted semantic segmentation (voxelized) c) ground truth segmentation.

With *FCPN* [222] we described an offline method for 3D semantic segmentation of point clouds. It is a hybrid network architecture that efficiently parses unordered point sets using an ordered network structure and 3D convolutions. This architecture enables the processing of large spaces, at once, unlike related point cloud networks. Generalization capabilities are analyzed when evaluating its performance in large and small scale semantic segmentation, segmenting indoor scenes and objects into their semantic parts. Finally, the spatially ordered feature can be used in higher-level applications such as 3D captioning; see [222] for more details. With its competitive performance on benchmark tasks and the ability to process significantly larger spaces in a single shot, we believe the proposed method is an appropriate starting point for further research on 3D point clouds.

Learning 3D Scene Graphs 6

The content of this chapter is based on the following publications:

[296] **Johanna Wald***, Helisa Dhamo*, Nassir Navab, and Federico Tombari. *Learning 3D Semantic Scene Graphs from 3D Indoor Reconstructions*. *Conference on Computer Vision and Pattern Recognition (CVPR)*. © 2020 IEEE.

[297] **Johanna Wald**, Nassir Navab and Federico Tombari. *Learning 3D Semantic Scene Graphs with Instance Embeddings*. *International Journal of Computer Vision (IJCV)*, 2022, Springer.

[308] Shun-Cheng Wu, **Johanna Wald**, Keisuke Tateno, Nassir Navab and Federico Tombari. *SceneGraphFusion: Incremental 3D Scene Graph Prediction from RGB-D Sequences*. *Conference on Computer Vision and Pattern Recognition (CVPR)*. © 2021 IEEE.

6.1 Introduction

3D scene understanding involves the perception and interpretation of 3D data to capture the semantic and geometric structure. This involves identifying and localizing the 3D objects present in a 3D scene and their context and relationships. Such an in-depth understanding might be beneficial for a variety of applications including but not limited to interior design, marketing, drones/robotic navigation but also general-purpose augmented and virtual reality.

Object-focused 3D scene understanding has already been extensively discussed in the previous chapters. Specifically, in Sec. 5.2.2 and Sec. 5.2.3, 3D semantic segmentation [55, 216, 218, 222] as well as methods for 3D instance segmentation and object detection and classification [108, 140, 216, 217, 257, 274, 320, 335] are introduced. All of these tasks are trained and evaluated with one of the many available 3D scene understanding datasets, see Sec. 3.1. While the majority of methods focus on entity semantics, its context is mainly utilized for refinement and improvement of object detection or classification.

Classical inter-object relationships are, however, important to fully understand the complexity of a real-world scene. A comprehensive representation that includes objects and their connections are scene graphs, shown in Fig. 6.1. Such graphs have been used for many different tasks, including image generation and modification, image retrieval, and captioning, to name a few. Scene graphs consist of a collection of object nodes and edges that describe semantic relationships *e.g. next to* or *smaller than*. Attributes such as *red, shiny* or *thin* as well as affordances (*e.g. holding* or *sitting*) are used to further describe the instances and their underlying properties. A detailed description about scene graphs is given in Chapter 4 where our scene graph dataset *3DSSG*, in Sec. 4.2, is introduced.

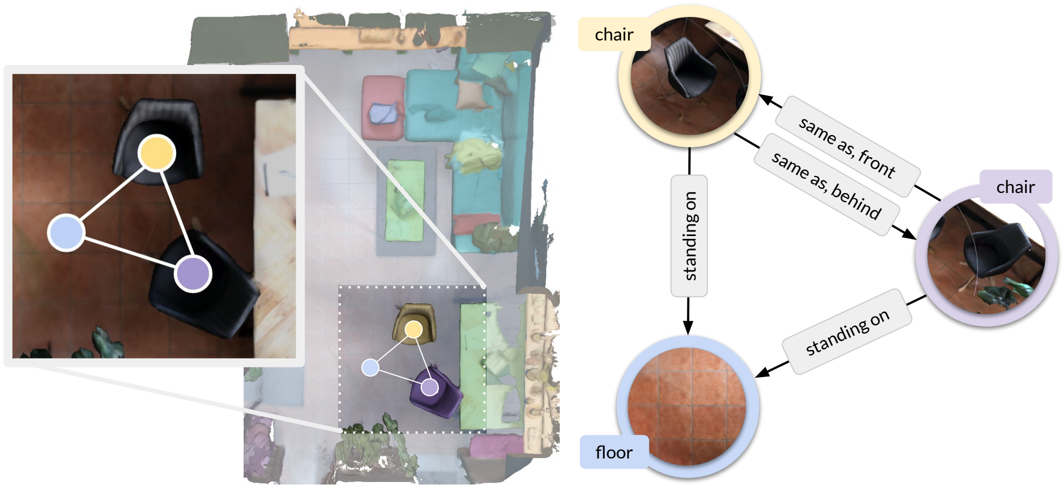


Fig. 6.1. Semantic scene graph $\mathcal{G}$ on a 3D scene [296]. In 3D scene graph prediction a graph (right) is computed given the 3D reconstruction of a scene (left). © 2020 IEEE.

We argue that semantic scene graphs are particularly well suited for the 3D domain. They are an a) compact representation that efficiently encodes a possibly large three-dimensional structure, which b) abstracts away minor scene changes and noise and c) has the potential to bridge the gap between different data representations *e.g.* images, 3D data, or natural language. This makes them an excellent choice for cross-domain tasks like retrieval search and Visual Question Answering.

In the following sections, an overview of related scene graph literature is given. Semantic scene graphs on images are addressed in Sec. 6.2.1, we then in Sec. 6.2.2 briefly review 3D instance segmentation methods related to our advanced graph prediction method of Sec. 6.4. Finally, we describe efforts moving towards structural understanding which also briefly touches on relationship prediction, see Sec. 6.2.3. Please note that data-related literature is not included in our review since it was already mentioned in Sec. 4.1, particularly in Tbl. 4.1.

We then finally present our methods for 3D scene graph prediction [296, 297, 308], which are an important part of the contributions of this thesis. We believe, regressing 3D scene graphs – as a way of learning and representing object relationships and contextual details – could be a critical component when understanding 3D environments.

In [296], see Sec. 6.3, we combined Graph Convolutional Networks (GCNs) [132] with PointNet [216] to construct a point-based architecture for scene graph prediction. Notably, it is the first learned method for 3D scene graph prediction. It infers a 3D scene graph by jointly classifying graph nodes and edges based on a class-agnostic segmentation of the scene. In Sec. 6.4 a more advanced scene graph prediction method is introduced, which utilizes sparse 3D convolutions [94] to learn semantics and instance embeddings jointly. The network can regress a scene graph given a real-world 3D reconstruction without any strict constraints or prior knowledge. Notably, our 3D scene graph dataset, *3DSSG* – which is publicly available on our project website[1] – serves as training data for all of the proposed methods.

One a side note: a cross-domain application, specifically 2D-3D scene retrieval using 3D semantic scene graphs, has been explored in one of our publications [296] and will be described later in chapter 7.

6.2 Related Work

In the following we review the literature on 2D scene graph (see Sec. 6.2.1), 3D instance segmentation (see Sec. 6.2.2) and give an overview about the research dealing with 3D semantics and context (see Sec. 6.2.3).

6.2.1 Scene Graphs with Images

Scene graphs are an extremely useful data abstraction to store connections and context between objects and describe the objects themselves. Therefore, they are a well-accepted representation in 2D scene understanding research, and many large-scale datasets for training and evaluation purposes exit, see review in Sec. 4.1.

As briefly mentioned earlier, 2D semantic scene graphs have been applied to a variety of different areas such as scene captioning [317], visual question answering [273], image retrieval and matching [119, 168, 301], image generation from graphs [120], interactive image generation [15] or image editing [63, 190]. Scene graph prediction is the foundation of many of these techniques and has therefore been studied extensively [107, 161, 162, 171, 203, 211, 219, 313, 314, 322, 323].

[1] `https://3DSSG.github.io`

Traditional scene graph approaches can be divided into two stages. First, object entities are detected using a classical object detector, often simply Faster R-CNN [221]. Based on these, relationships are extracted by grouping estimated object detections into pairs aiming to assign a predicate label *e.g*. *standing on* or *throwing* to all valid connections. This step is often simply a classification task that takes the feature vectors of all proposals as input. For this purpose, multiple encodings for objects and relationships have been proposed ranging from low-level hand-crafted features such as bounding boxes or relative transformations given object centroids – similar to our proposed descriptor in Sec. 5.3.2 – to higher-level features extracted from a neural network [313, 314]. Some works propose a combination of these features [211] while others suggest to additionally utilize data from other domains *e.g*. linguistic priors [171, 219].

To consider all possible relationships between all n^2 object pairs is a pretty costly operation. Therefore, Yang *et al*. propose to filter the relationship tuples before classification [314]. Li *et al*., on the other hand, suggest factorizing their graphs into sub-graphs that follow spatial constraints [162]. A simple form of relationship prediction, that does not need to handle scalability is known as visual relationship detection [171, 211]. It usually operates locally and classifies each *subject-object*-pair separately.

A popular technique to process graph data structures are GNNs, naturally designed for this task [50, 78, 138, 219, 238, 288, 289]. Most interestingly, [219, 314] use message passing within an RNN [313] or attention graph neural networks [313] which distribute attention and refine relationships and objects iteratively. Similarly, [161] also applies message passing to propagate learned features from the objects to the relationships. Other techniques propose a new, novel contrastive loss [326] to learn embeddings from pixels [203] or explore permutation invariance architectures [107]. Quite explorative is the approach by Zareian *et al*. [322] – they suggest an iterative graph-based neural network that learns to transform a given knowledge graph to the scene graph of an image.

Scene graphs for a large number of images are available in datasets like Visual Genome [136] or VRD (Visual Relationship Detection) [171], allowing the implementation and experimental analysis of the approaches mentioned above. Even though significant improvements have been made in 2D scene graph prediction, it remains a difficult challenge attributed to the interdependence and complexity of object and relationship detection and predicate and node classification.

Even though we contributed a scene graph dataset of 3D scenes, which provides a large-scale training set for 3D methods, see Sec. 4.2, memory constraints associated with the complexity of 3D representations do not allow to transfer concepts and principles applied on images directly to 3D.

6.2.2 3D Instance Segmentation

3D semantic segmentation is undoubtedly a significant component of 3D scene understanding. It classifies voxels or points of the 3D scene by assigning a single semantic class given a predefined list of labels, see Sec. 5.2.3. In 3D semantic instance segmentation, a unique identifier is additionally assigned, allowing to segment the geometry belonging to the same object instance. Learned 3D instance segmentation is usually implemented using a bottom-up or top-down approach, with the latter generating proposals from the input.

A classical top-down approach is Hou *et al*. [108]. They generate bounding boxes similar to 2D Mask R-CNN [103] but operate on the 3D data augmented with multi-view 2D features extracted from underlying image sequences [108], similar to [55]. VoteNet, on the other hand, proposes to use a novel object center voting scheme to predict bounding boxes on point clouds [215].

Bottom-up instance segmentation is often associated with metric learning [99, 117, 140]. Classical bottom-up approaches are usually proposal-free and learn feature embeddings instead, which the goal to produce an embedding space where points on the same instance are grouped while features on other instances are distant and distinguishable. To do so, Lahoud *et al*. [140] suggested a multi-task learning technique. Han *et al*. base their method on many of the concepts proposed in [140] but additionally add a novel occupancy loss [99]. Once feature embeddings are learned, they are clustered into instances. PointGroup proposes an improved clustering technique that utilizes the void space between objects [117].

Filtering techniques are applied to reduce noise in the proposals, such as non-maximum suppression or short NMS. 3D-MPA, which is inspired by the VoteNet architecture, suggests a novel graph neural network to filter proposals and show its benefits over classical NMS [68]. Interestingly, 3D-MPA is a hybrid method combining concepts of bottom-up and top-down approaches.

Notably, all 3D instance segmentation methods evaluate their performance by only considering foreground objects while ignoring the background. Panoptic segmentation additionally semantically classifies the background classes but still does not treat *e.g*. walls or the floor as separate instances. Arguably, it is unclear whether structures should be identified as an instance or not given the available segmentations in datasets such as *ScanNet* or *RIO*. We believe that the identification of all components is required to be able to parse important groups of relationships *e.g*. support. We, therefore, treat all classes, including structural components, as nodes in our scene graph prediction approach presented in Sec. 6.4.

6.2.3 3D Object Context and Scene Layout

Scene graphs have been used in computer graphics (CG) for decades to spatially organize scenes where nodes represent entities and edges model relative transformations between them [112, 130]. They have been proven to be a versatile scene representation modeling multiple non-trivial spatial relations for *e.g*. animations and gaming.

In the 3D computer vision (CV) community, some related graph concepts such as support have briefly been explored in datasets and methods [199]. Compared to the classical CG graph definition, the focus of CV instead is to model semantically rich connections *e.g*. in Visual Genome [136]. 3D semantic and instance segmentation, see Sec. 5.2 and Sec. 6.2.2, are object-focused tasks. They stand in contrast to holistic scene understanding, where scene understanding is defined as a combination of several tasks ranging from layout prediction [18, 111, 257] to shape reconstruction and pose estimation [194, 204] or a combination of these [338]. [46, 325] both use a holistic approach to generate a 3D scene representation given a single 2D image. To do so, Choi *et al*. combines object detection and layout estimation [46]. More recently, [325] propose a new loss and combine an image-based local structured implicit network with an implicit scene graph neural network and remarkably achieve state-of-the-art results in shape and layout estimation as well as in object detection.

Other compact scene presentations related to scene graphs are stochastic grammars [167, 330]. Zhao *et al*. use the rules AND, OR, and SET to describe a scene with layout, objects, and primitive shapes, including lines and planes [330]. Other proposed data structures are hierarchical trees where the leaf nodes represent independent entities such as the parts of an object. In such a representation, parent nodes group their children into functional parts [167]. These – or similar representations – are used in scene synthesis to produce a realistic 3D scene model given an abstract encoding of the scene. The scene layout either explicitly or implicitly integrates objects and scene structure [116, 252, 300]. Kernel functions are proposed in [77] to retrieve scene components featuring similar configurations. Jiang *et al*. introduce a programmable 3D scene synthesis pipeline that uses stochastic grammars [116] which they refer to as spatial and-or graphs. A recursive VAE combined with an iterative retrieval is proposed in GRAINS [151] to build a scene layout by iteratively adding objects in realistic locations. Another iterative approach is [252]. Their method is similar as it also uses a variational recursive auto-encoder. [138], on the other hand, synthesize a 3D scene from a 2D image by exploring relative poses between objects within a graph convolutional neural network. In [174], natural language is used. They convert a scene into a graph which is utilized within a retrieval pipeline similar to [77]. Noteworthy, graphs are commonly used to describe objects or generate scene descriptions either in static or dynamic setups [154].

Interestingly, graph representations are not limited to scenes but have also been used to encode objects and their components. In such a definition, nodes represent object parts *e.g*. the *leg* or *backrest* of a chair. With this in mind, Te *et al*. segment point clouds of objects into parts via a novel regularized graph CNN [272]. Another fascinating work is StructureNet [191]. Different segments of an object are encoded in a graph of latent features. This enables them to interpolate between different object instances or even sample novel shapes. The relationships in an object graph are known physical connections – similar to the definition in computer graphics. A major disadvantage of StructureNet is that a separate network has to be trained for every object class.

Computing graphs for 3D scenes is still a relatively unexplored research topic [12, 83]. Creating 3D semantic scene graphs from real-world scans is quite tricky due to the complexity of 3D data and potentially ambiguous real-world relationships between objects. Gay *et al*. use an off-the-shelf 2D object detector on image sequences and merge objects in a global 3D map of 3D quadrics. They then process the data with a recurrent neural network. Armeni *et al*. operates on panoramic images combining a sampling and regularization technique to optimize the mask predictions, ultimately improving 3D detection of the object nodes [12]. Notably, none of the aforementioned methods regresses a scene graph given the 3D data directly but instead operates on multiple images or sequences.

In the following, we introduce three different scene graph prediction methods. While the *Scene Graph Point Network (SGPN)* proposed in [296] operates on point sets directly, it requires a ground truth segmentation of the input data, see Sec. 6.3. The second method is called *Scene Graph Embedding Network (SGEN)* [297] and avoids this assumption operating on the full raw 3D scene, see Sec. 6.4. Nodes are identified by learning instance embeddings with a bottom-up approach, see Sec. 6.2.2. [297] jointly estimates the scene graph and 3D instances and does not require any segmentation making it directly applicable in real-world scenarios. Finally, in Sec. 6.6, *SceneGraphFusion* [308] is briefly mentioned which incrementally constructs a scene graph given an RGB-D sequence.

6.3 Scene Graph Point Network

Having introduced a novel 3D scene graph dataset in chapter 4, the following sections focus on scene graph methodology as proposed in [296] and [297]. Scene graph prediction is a particularly challenging task as it requires identifying objects and complex and potentially ambiguous relationships based on noisy and partial point clouds. As an example, two chairs of the same type potentially have drastically different poses and exposure, while a jacket *lying on* one of them easily occludes most of its visible surface.

Our first method [296] aims to compute a dense graphs given a 3D scene and its class-agnostic segmentation, see Fig. 6.2a. Object and edge features are computed using point-based architectures to be re-arranged in a fully-connected graph of features (see Fig. 6.2b). The encodings of the tuples of this graph are then processed with a graph neural network where each node and edge is classified respectively to ultimately obtain a 3D semantic scene graph as visualized in Fig. 6.2c. The architecture operates end-to-end and is designed to estimate multiple relationships per edge. The source code[2] and used dataset[3] are publicly available.

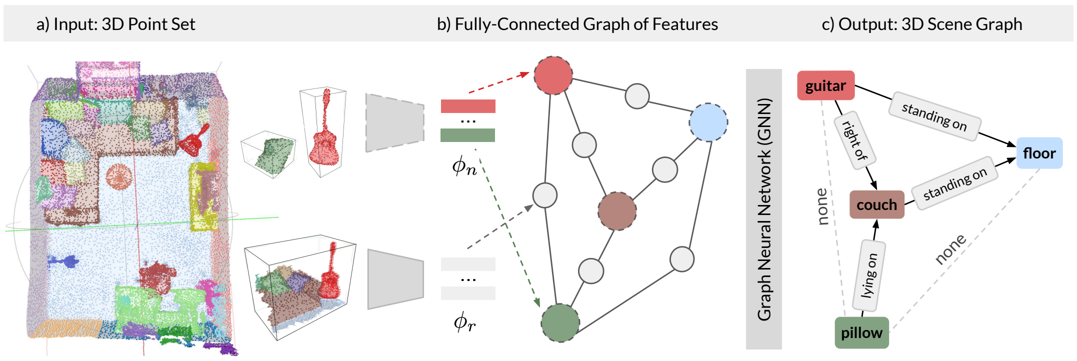


Fig. 6.2. Our scene graph prediction method takes an a) segmented point set of a scene as input. Object and edge features are re-arranged in a graph structure b). The output of the method is a 3D semantic scene graph that encodes node and edge semantics. © 2020 IEEE.

Formally defined, the unordered point set $\mathcal{P}^{(\mathrm{p})} = \{\mathbf{p}_i\}_{i=1}^{\mathrm{N_P}}$ and its ground truth segmentation $\mathcal{P}^{(\mathrm{i})}\{\mathrm{i}_i\}_{i=1}^{\mathrm{N_P}}$, without the semantic component $\mathcal{P}^{(\mathrm{l})}\{\mathrm{l}_i\}_{i=1}^{\mathrm{N_P}}$, is the input of our method. The output of the *Scene Graph Point Network (SGPN)* is – as you might have guessed – a scene

[2] https://github.com/ShunChengWu/3DSSG
[3] https://3dssg.github.io/

graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$. Objects, such as the pillow or the guitar in Fig. 6.2, are the nodes of the graph $\mathcal{N}$. Example relationships $\mathcal{E}$ of Fig. 6.2c are the *standing on* and *lying on* relations between guitar and floor and pillow and couch respectively.

Similar to previous works [171, 313, 314], visual features are extracted separately per object as well as per edge. For this purpose, we employ a low-level feature encoder based on PointNet [216]. Two separate networks, *3D-ObjPointNet* and *3D-RelPointNet* are utilized, to extract features for the object nodes ϕ_o, as well as for the relationships ϕ_r respectively. Both networks are trained jointly but do not share any weights since they process different types of data. For *3D-ObjPointNet*, the point set $\mathcal{P}_k^{(\mathrm{p})}$ of an instance k *e.g.* the guitar is extracted using the the Kronecker delta δ as follows

$$\mathcal{P}_k^{(\mathrm{p})} = \{\delta_{ki} \odot \mathbf{p}_i\}_{i=1}^{\mathrm{N_P}}, \tag{6.1}$$

$$\delta_{ki} = 1 \iff k = i. \tag{6.2}$$

Please note that the points of each object as well as the later-described predicate data, in eq. 6.5, are normalized

$$\acute{\mathcal{P}} = \{\mathbf{p}_i - \bar{\mathbf{p}}\}_{i=1}^{|\mathcal{P}|}, \tag{6.3}$$

before fed into the respective networks. To keep the notation simple, we do not specifically state this in later equations.

The set of normalized 3D point coordinates $\mathcal{P}^{(\mathrm{p})} \in \mathbb{R}^{3\times \mathrm{N_P}}$ is the input of *3D-ObjPointNet*, formally defined as $f_\phi^{(o)}(\cdot)$. It produces an object feature

$$f_\phi^{(o)}(\mathcal{P}^{(\mathrm{p})}) \to \phi_o, \tag{6.4}$$

which is further utilized within our graph neural network.

The features for predicate prediction describe a relationship between two instances k and j. Its computation requires the extraction of the point set within the joint bounding box $\mathbf{b}_j \cup \mathbf{b}_k$,

$$\mathcal{P}_{kj} = \{\forall \mathbf{p}_i \in \mathcal{P} \mid \mathbf{p}_i \in (\mathbf{b}_k \cup \mathbf{b}_j)\}. \tag{6.5}$$

The final feature $\dot{\mathcal{P}}_{kj} \in \mathbb{R}^{4\times \mathrm{N_P}}$ is the input of the *3D-RelPointNet* function $f_\phi^{(r)}(\cdot)$ obtained by concatenating $[\cdot]$, see eq. 6.7, a binary instance mask and the point set $\mathcal{P}_{kj}$ within the joint bounding box of instance k and j. The masking function and $\dot{\mathcal{P}}_{kj}$ are defined as follows

$$f_m(x) = \begin{cases} 1, & \text{if } x = k \\ 2, & \text{if } x = j \\ 0, & \text{otherwise,} \end{cases} \tag{6.6}$$

$$\dot{\mathcal{P}}_{kj} = \left[\mathcal{P}_{kj}, (f_m(\mathrm{i}))_{i=1}^{\mathrm{N_P}}\right]. \tag{6.7}$$

The concatenation is required as the point set $P_{kj}^{(\mathrm{p})}$ is identical for the edge $k \rightarrow j$ and $j \rightarrow k$, specifically $P_{kj}^{(\mathrm{p})} = P_{jk}^{(\mathrm{p})}$. Therefore, $f_m(\cdot)$ introduces necessary directional information. Notably, we also keep the orientation of the node to not invalidate proximity relationships and therefore avoid rotationally augmentation. However, some random jitter is applied to introduce some variation to the point clouds which simulates natural noise.

Finally, the object and relationship features obtained from *3D-ObjPointNet* and *3D-RelPointNet* are re-arranged into a graph structure, see Fig. 6.2b. Nodes are connected to form relationship triples featuring a *subject, predicate* and *object*. A triple relationship feature ϕ_r is constructed by simply combining the two object features ϕ_o and ϕ_s with the predicate feature ϕ_p

$$\phi_r = [\phi_s, \phi_p, \phi_o]. \tag{6.8}$$

The triple features are passed onto a graph convolutional network (GCN) [132], explained in the following. Notably, the GCN of *SGPN* is inspired by the image generation network of Johnson *et al*. [120]. As scenes can be of varying complexity, we seek a flexible graph network in terms of input nodes. Therefore, in each layer l of the GCN, a list of triples is processed with two MLPs $g_1(\cdot)$ and $g_2(\cdot)$ such as

$$(\psi_{s,ij}^{(l)}, \phi_{p,ij}^{(l+1)}, \psi_{o,ij}^{(l)}) = g_1(\phi_{s,ij}^{(l)}, \phi_{p,ij}^{(l)}, \phi_{s,ij}^{(l)}). \tag{6.9}$$

where s stands for subject, o for object and p for predicate. To propagate information from neighborhood nodes, the inputs of the connected edges are averaged using the set of connections of a node $f_{\mathcal{R}}(x)$

$$\chi_i^{(l)} = \frac{1}{|f_{\mathcal{R}}(i,s)| + |f_{\mathcal{R}}(i,o)|}\Big(\sum_{j \in f_{\mathcal{R}}(s)} \psi_{s,ij}^{(l)} + \sum_{j \in f_{\mathcal{R}}(o)} \psi_{o,ji}^{(l)}\Big). \tag{6.10}$$

Finally, this output is passed to another MLP, which we dub $g_2(\cdot)$. Since this potentially causes Laplacian smoothing, a residual connection is applied, and the final feature is obtained as follows

$$\phi_i^{(l+1)} = \phi_i^{(l)} + g_2(\chi_i^{(l)}). \tag{6.11}$$

Notably, $g_1(\cdot)$ and $g_2(\cdot)$ are linear layers followed by a Rectified Linear Unit (ReLU) activation

$$\mathrm{ReLU}(\mathbf{x}) = \max(0, \mathbf{x}). \tag{6.12}$$

This process is repeated for every layer of the graph network by simply taking the features of $\phi_{s,ij}^{(l+1)}, \phi_{p,ij}^{(l+1)}, \phi_{o,ij}^{(l+1)}$ and processing them as stated in equation 6.9. After each layer, there is *one-hop* more context stored in each node encoding, therefore the complexity of relationships that can be captured increases. However, more layers also introduce potential over-smoothing. While our implementation uses $l = 5$ layers a network with only two layer $l = 2$, performed similarly.

To obtain the final labels of the predicates and object nodes, three MLPs finalize the network architecture. The output of the final node and edge encoding is a vector of size 256.

SGPN is trained end-to-end with a loss that is jointly optimized for object ℓ_o and predicate classification ℓ_p using a weighting factor λ_o which was set to $\lambda_o = 0.1$ in all of our experiments

$$\ell = \lambda_o \ell_o + \ell_p \tag{6.13}$$

Notably, multiple predicates potentially exist between a pair of objects, *e.g.* a chair which is standing *next to* a table could also have the *same color, texture or material*. To capture this, the predicate prediction is modeled with a per-class binary cross-entropy. An example is earlier given in Fig. 6.1, where a chair is in front of another and also has the same appearance (*same as*).

Class imbalance is handled with a weighted focal loss [165]

$$\ell = -\lambda_\alpha (1-\beta)^{\alpha'} \log \beta. \tag{6.14}$$

where β are the logits, and λ_α is the weighting factor computed from the class frequency. The network was trained using Adam [131] with an initial learning rate of 10^{-4}.

6.4 Scene Graph Embedding Network

In the following section, the *Scene Graph Embedding Network* (*SGEN*) proposed in [297] is discussed. An overview of the method is given in Fig. 6.3. Similarly to *SGPN* [296] it goes beyond object awareness and estimates semantic scene graphs (right) by combining features of a 3D semantic instance segmentation network with scene graph prediction, Fig. 6.3c. Given the full 3D scene reconstruction visualized in the left of Fig. 6.3, a 3D sparse convolutional neural networks processes its 3D point coordinates, normals and colors to learn instance embedding as well as semantic, see Fig. 6.3b.

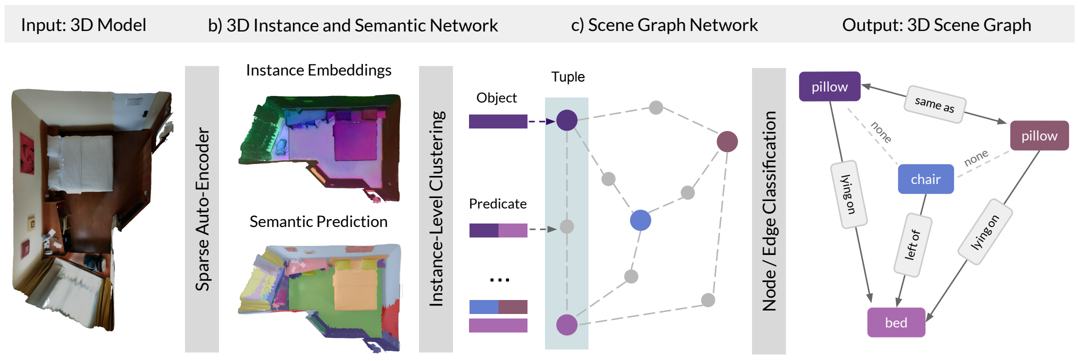


Fig. 6.3. Scene Graph Embedding Network (SGEN) predicts a 3D scene graph and instances jointly by combing a 3D network with graph prediction.

The 3D instances and graphs are learned jointly in an *end-to-end* network by directly passing the latent features of the scene into a graph prediction model. Within this model, class labels for both object nodes and edges are predicted.

Arguably, instance segmentation and scene graph prediction are two closely related tasks as instances are one of the main components of semantic scene graphs. While the first part of our network learns features that are passed to the scene graph stage, its purpose is also to recognize the instances as graph nodes. Nodes are identified by segmenting the embedding space, similar to other bottom-up instance segmentation methods reviewed in Sec. 6.2.2.

This has the advantage over the previous approach [296] that no ground truth segmentation is required when computing graphs, which makes this approach directly applicable to real-world situations in which no prior scene knowledge is available. Additionally, *SGEN* [297] is more scalable than *SGPN* [296], especially when the number of objects grows. Instead of passing each segment and each object pair into a PointNet architecture [216], as done in [296], the object and edge features are extracted from a volumetric 3D feature map of the scene which, notably, only requires a single network pass.

Moreover, the proposed methods utilize 3D sparse convolutions [94] and therefore only process non-empty regions of the 3D scene volume. Also, resulting features are more descriptive than [296] as object and edge features encode the 3D data of the foreground as well as background, incorporating surface normals and color within the networks' receptive field. The training and evaluation of the proposed method are conducted on the scene graph dataset described in Sec. 4.2.

A more detailed explanation and implementation details of the two main parts of the proposed method are explained in the following with more detail, including the 3D network that learns 3D semantic instance features, see Sec. 6.4.1, as well as the scene graph prediction network, see Sec. 6.4.2.

6.4.1 3D Instance and Semantic Network

The point set $\mathcal{P} = \{\mathbf{p}_i, \mathbf{n}_i, \mathbf{c}_i\}_{i=1}^{\mathrm{N_P}}$ containing 3D coordinates $\mathbf{p}_i$, normals $\mathbf{n}_i$ and colors $\mathbf{c}_i$, is the input of our method. The color values $\mathbf{c}_i$ are obtained from utilizing the UV-coordinate of the vertex which maps to the corresponding RGB pixel value on the 2D texture image. The 3D network $f_\phi^{(3D)}(\cdot)$ processes the whole scene at once,

$$f_\phi^{(3D)}(\mathcal{P}) \rightarrow \mathcal{P}^{(\phi)} = \{\phi\}_{i=1}^{\mathrm{N_P}}, \tag{6.15}$$

and obtains object-level features by clustering $f_\rho(\cdot)$ the points and their encodings afterwards to obtain a set of object features $\{\phi_i\}_{i=1}^{\mathrm{N_N}}$

$$f_\rho(\mathcal{P}^{(\phi)}) \rightarrow \{\phi_i\}_{i=1}^{\mathrm{N_N}}. \tag{6.16}$$

This increases descriptiveness and scalability compared to computing features for each object node and relationship separately [296]. In contrast to other instance segmentation approaches, in *SGEN*, all points are considered to be the foreground, as regular instances, including walls and floors. Since structural elements are essential entities in the scene graphs of *3DSSG* [296], they are purposefully included.

Like the network proposed by [194], an encoder-decoder architecture is used with filter sizes 64, 96, 128, 160, 192, and 224 for the convolutional layers and a backbone size of 256. In the encoder, maximum pooling layers are used. Furthermore, inspired by the U-Net architecture [48, 225], skip connections are incorporated in our network. The last two layers of the network are two convolutional layers with ReLu, eq. 6.12, and batch norm outputting logits for semantic segmentation $\mathbf{f}_i$ and embeddings $\psi_i^{(e)}$ as follows

$$\phi = \{(\mathbf{f}_i, \psi_i^{(e)})\}_{i=1}^{\mathrm{N_P}}. \tag{6.17}$$

The features of the embedding space are then clustered as stated in eq. 6.16 to obtain separated object instances as proposed by other bottom-up instance segmentation methods [99, 117, 140]. They are learned with an N-pair metric learning loss [256], that uniformly samples indices $\mathbf{S}_i$ on the objects to counteract data imbalance and uses a similarity metric $f_s(i,j)$ to compute the similarity between the embedding vectors $\psi_j^{(e)}$ and $\psi_i^{(e)}$ of i and j such that

$$\ell_n = -\frac{1}{|\mathbf{S}_i|} \sum_{i,j \in \mathbf{S}_i} \ell_\psi(i,j), \tag{6.18}$$

$$\ell_\psi(i,j) = \begin{cases} \log(\ell_s(i,j)), & \text{if } \mathrm{l}_i = \mathrm{l}_j, \\ \log(1 - \ell_s(i,j)) & \text{otherwise;} \end{cases}. \tag{6.19}$$

$$\ell_s(i,j) = \frac{2}{1 + \exp(||\psi_i^{(e)} - \psi_j^{(e)}||_2)}. \tag{6.20}$$

In addition to the embedding space, a cross-entropy loss ℓ_e is used to learn semantic classes per point. We learn the semantic and instance embeddings together and equally weigh the losses such that

$$\ell = \ell_\mathrm{s} + \ell_\psi. \tag{6.21}$$

When extracting the semantics of a scene, the instance embeddings are incorporated. To do so, the scene is first over-segmented into several small clusters. For each cluster, the semantics are averaged, aiming to reduce noise and refine the semantic segmentation.

The training uses $2\,\mathrm{cm}$ voxels which are mapped back to the point set at test time. The applied augmentation involves translation, scale, and rotation variations. The sparse convolutions implemented in *tf3d* [93] are used within this work.

6.4.2 Scene Graph Network

Given the features of the 3D network, $\{\phi\}_{i=1}^{\mathrm{N_P}}$, a scene graph, formally defined as $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is estimated. To do so, a fully connected graph, see Fig. 6.3c, is constructed based on the detected nodes such that

$$\mathcal{E} = \mathcal{N} \times \mathbb{P} \times \mathcal{N}. \tag{6.22}$$

During training the features of the 3D network are clustered using the instance segmentation $\mathcal{P}^{(\mathrm{i})} = \{\mathrm{i}_i\}_{i=1}^{\mathrm{N_P}}$. At inference time, the embeddings are clustered with a function $f_\rho(\cdot)$, which is chosen to be kmeans++ [14] in our implementation, such that

$$\hat{\mathcal{P}}^{(\mathrm{l})} = \{\hat{\mathrm{l}}_i\}_{i=1}^{\mathrm{N_P}} = f_\rho(\phi). \tag{6.23}$$

The final feature vectors for nodes ϕ_k and edges ϕ_r are obtained by simply averaging the embedding and semantic features of all indices of the respective instance such that

$$\bar{\psi}_k = \frac{1}{|\mathbf{S}_k|}\sum_{i\in\mathbf{S}_k}\psi_i, \text{ and } \bar{\mathbf{f}}_k = \frac{1}{|\mathbf{S}_k|}\sum_{i\in\mathbf{S}_k}\mathbf{f}_i \quad \text{and } \mathbf{S}_k = \{i \mid \mathrm{l}_i = k\}_{i=1}^{\mathrm{N_P}}. \tag{6.24}$$

Based on these node and edge features, a scene graph is built by constructing (*subject, predicate, object*)-triples. The predicate encodings are obtained from concatenation of subject ϕ_s and object node features ϕ_o. Additionally hand-crafted features such as the relative position of the two connected nodes, $\bar{\mathbf{p}}_s$ and $\bar{\mathbf{p}}_o$ or their bounding boxes are incorporated

$$\phi_r = [\phi_s, \dot{\mathbf{p}}_s - \dot{\mathbf{p}}_o, \mathbf{b}_s - \mathbf{b}_o, \phi_o]. \tag{6.25}$$

The graph classification uses two output branches that optimize a cross-entropy loss for two different semantic class sets with varying levels of detail. Similar to [296] the predicate prediction is modeled with a per-class binary cross-entropy to handle multiple class occurrences between nodes. The object and predicate prediction is combined in a single loss function with a weighting factor of 0.5 such that

$$\ell = 0.5 \cdot \ell_o + \ell_p \tag{6.26}$$

Compared to *SGPN* [296], this method does not require splitting the scene into sub-graphs when training but instead processes the original scene graphs in a single pass. Within the *SGPN*, convolutional layers use batch norm and ReLu, see eq. 6.12; the object and relationship predictors have four and six fully connected layers, respectively. For optimization purposes, an SGD optimizer with an initial learning rate of 10^{-2} is used. While theoretically, both networks can be trained together, they were trained independently to stabilize the convergence. The 3D network is trained first, and its weights are frozen when the scene graph prediction is trained.

6.5 Evaluation

In this section, both of the presented methods [296, 297] are evaluated. Scene graph prediction experiments are carried out for *SGPN* and *SGEN* and the 3D semantic segmentation performance of *SGEN* is evaluated. Its learned embedding space is visualized, and the effect of the input feature and embedding dimension is ablated.

Notably, all of the following experiments are conducted on the *3DSSG* dataset. *3DSSG* [296] is a collection of semantic scene graphs for *RIO*, see Sec. 4.2 for a detailed description of the graph dataset and Sec. 3.2 for an overview of *RIO* [295]. For testing and training purposes, the proposed scene split of *RIO* is used [295]. Since the graphs in *3DSSG* are extremely large, the following experiments utilize the smaller sub-graphs of *3DSSG* where all nodes and connections are processable on a single, off-the-shelf GPU. The sub-graphs consist of a maximum of 9 nodes and only a subset of the relationships classes. Finally, only the most occurring object classes are included, resulting in 160 object categories and 26 predicates, see Sec 4.2.5.

6.5.1 Scene Graph Prediction

In scene graph prediction, *SGPN* and *SGEN* are compared against a relationship prediction baseline inspired by *VRD* [171]. *3D-VRD*, is a re-implemented 3D version of [171], where node and edge point sets are extracted in a similarly manner as explained in the method section of *SGPN*, see Sec. 6.3. Feature vectors are then obtained from utilizing PointNet [217]. For this, an object and relationship predictor was trained that uses three fully connected layers, a batch norm and ReLu, eq. 6.12. In Tbl. 6.1, the multi-predicate classifier and the graph features of *SGPN* are ablated and compared against the baseline method, *3D-VRD*.

Tbl. 6.1. Scene graph prediction accuracy on *3DSSG*. © 2020 IEEE.

	Relationship Prediction		Object Prediction		Predicate Prediction	
Method	R@50	R@100	R@5	R@10	R@3	R@5
3D-VRD	0.39	0.45	0.66	0.77	0.62	0.88
SGPN Single Predicate, Object PointNet	**0.46**	0.52	**0.69**	**0.79**	0.70	0.85
SGPN Multi Predicate, Object PointNet	0.41	**0.67**	0.68	0.78	**0.92**	**0.96**
SGPN Multi Predicate, Object GCN	0.30	0.60	0.60	0.73	0.79	0.91

Following previous methods such as [313], predicates and objects are evaluated independently from each other, see columns *Relationship Prediction, Object Prediction* and *Predicate Prediction* in Tbl. 6.1. And while in other works, relationship tuples are predicated jointly, they are independently regressed in this work. The triple classification confidence is computed by combining the scores of the object, subject, and predicate, similar to [314]. Their confidence then ranks the (subject, predicate, object)-tuples, and a top-n recall metric is reported together with predicates and object scores, see Tbl. 6.1.

Even though [313] computes confidences similarly and also ranks their results to obtain a top-n recall, they combine all the relations of a scene while we retain edge mappings. This means a prediction *e.g.* chair-standing on-floor is only considered correct if it also maps to the correct set of objects. While their evaluation scheme might be beneficial for smaller graphs such as those present in images or lower-quality annotations, it does not work well for large scenes with dense graphs such as *3DSSG* [296]. Relatively low confidence but correct triplet prediction could end up further down a top-n ranking if other edges in the same scene rank higher, especially given a dense graph with several other connections.

Given the results presented in Tbl. 6.1, *SGPN* outperforms the baseline slightly when estimating objects and significantly when predicting relationships and predicates. Notably, the multiple predicate prediction model (*SGPN* Multi Pred.) contributes to a higher accuracy, which we attribute to the inherent uncertainty in real-world scenes where relations are potentially ambiguous, and multiple outputs are plausible. Also, the model that uses the PointNet features outperforms the ones that use the output of the GCN, probably due to the feature aggregation within the fully connected graph and the lack of sophisticated attention mechanisms.

Tbl. 6.2. Scene Graph prediction accuracy on *3DSSG*.

Method	Object Prediction	Predicate Prediction	Relationship Prediction
3D-VRD	35.1	15.0	5.4
SGPN Multi Predicate, Object GCN	39.9	58.5	20.3
SGPN Multi Predicate, Object PointNet	33.4	64.2	31.5
SGEN	**52.0**	**71.2**	**42.5**

Predicting as well as evaluating scene graphs is quite challenging since multiple interdependent tasks are involved. Even though *SGEN* is able to detect instances and therefore does not require a scene segmentation, it is utilized in this experiment to fairly compare against [296], see Tbl. 6.2. Notably, a classification is only considered correct if predictions are perfect or, specifically: if the confidence of the ground truth label is highest and therefore ranked top-1. This is a stricter measure as the scores reported in Tbl. 6.1 and quite challenging given 160 class labels. It can be seen that *SGEN* performs best in all categories with a large margin likely due to the descriptiveness of the produced features. Notably, this also significantly improves scene retrieval, as discussed in the following chapter, see chapter 7.

Qualitative results of our scene graphs are given in Fig. 6.4 and 6.5. Fig. 6.4 shows graphs of relatively small scenes while in Fig. 6.5 the results are simplified by rendering respective sub-graphs. Notably, mis-classifications are reasonable and likely due to similarity of specific class labels *e.g. desk* vs. *computer desk*, *object* vs. *toilet brush* as well as mixing up objects that are visually identical in a geometric representation such as *picture* and *tv*. The graph edges are highlighted in different colors depending on the prediction. 🟩 A *bright green* edge means the prediction is fully accurate, 🟩 *dark green* is an indicator for partial correctness when a subset of multiple predicates is correct. Potentially false positives where ground truth is missing are 🟦 *bright blue* which visually seem to be mostly correct. Finally, 🟥 *red* edges are incorrect and ⬜ *gray* is used when the connection was missed. For reference purposes, the ground truth label is writing in brackets after the predicted label, in both Fig. 6.4 and Fig. 6.5.

6.5.2 3D Semantics and Instance Embeddings

Next, the 3D semantic and instance segmentation performance of *SGEN* is evaluated. First, Tbl. 6.3 reports a per class evaluation of 3D semantic segmentation on the *RIO* dataset. Furthermore, the semantic segmentation of *SGEN* is compared against [169] using the per class average of the mean IoU of the semantic masks. This experiment is also conducted on the *RIO* [295] dataset and 27 classes. Our method achieves a mIoU of 44.8 compared to

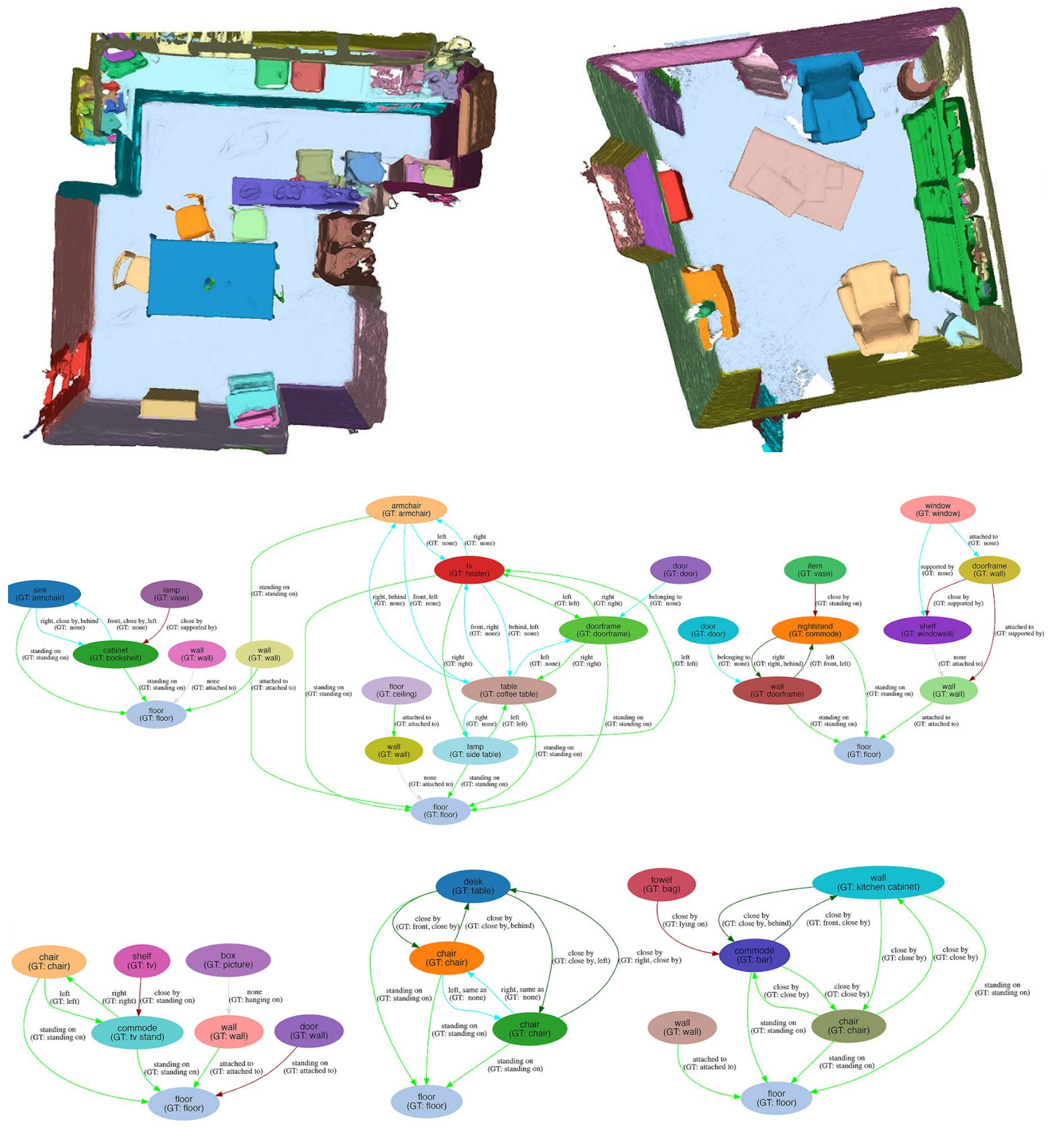


Fig. 6.4. Scene graph predictions of *SGPN* [296], 🟩 correct predicates, 🟦 unknown or missing in ground truth, 🟥 wrongly classified, ⬜ missed (*none* predicted). © 2020 IEEE.

40.8 and 44.2 using 3D and 2D-3D context, respectively and can achieve a reasonably good segmentation performance on challenging classes such as windows or curtains.

Instance clustering is evaluated in the next experiment, where different embedding vector dimensions and different input features are tested. As commonly done when evaluating instance segmentation, the mean average precision is reported, see Tbl. 6.4. While the dimension of the feature seems to have only minor impacts on segmentation quality, a feature size of 256 gives the best results with a minimal margin. Tbl. 6.4 also analyses the impact of the input of the clustering, specifically using instance embeddings with point coordinates (spatial) and semantics. Interestingly, using spatial knowledge and semantics seems to reduce its performance. Therefore, the best-performing setup was only using embeddings.

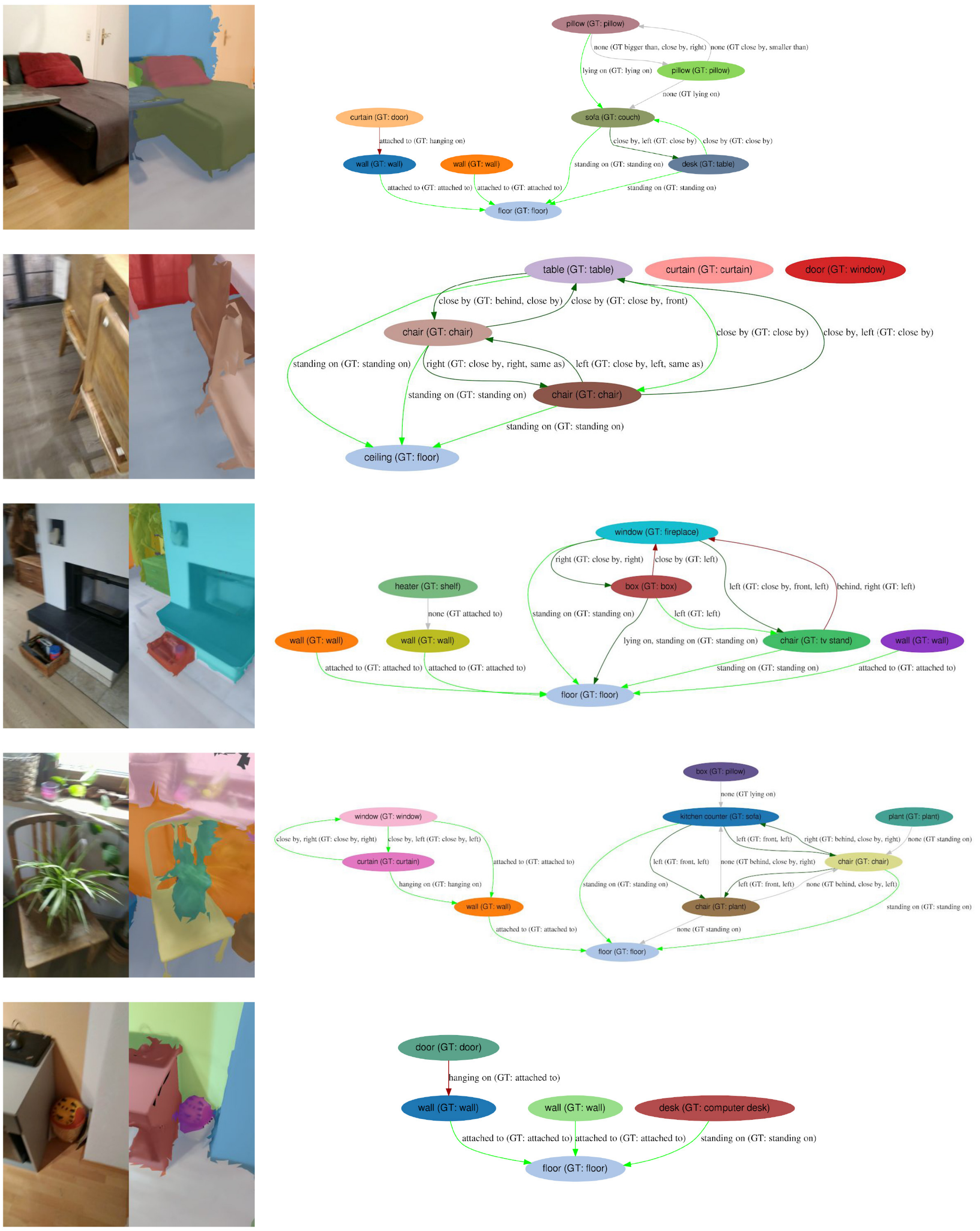

pillow (GT: pillow)
none (GT bigger than, close by, right)
none (GT close by, smaller than)
lying on (GT: lying on)
pillow (GT: pillow)
none (GT lying on)
curtain (GT: door)
sofa (GT: couch)
attached to (GT: hanging on)
close by, left (GT: close by)
close by (GT: close by)
wall (GT: wall)
wall (GT: wall)
standing on (GT: standing on)
desk (GT: table)
attached to (GT: attached to)
attached to (GT: attached to)
standing on (GT: standing on)
floor (GT: floor)
table (GT: table)
curtain (GT: curtain)
door (GT: window)
close by (GT: behind, close by)
close by (GT: close by, front)
chair (GT: chair)
close by (GT: close by)
close by, left (GT: close by)
standing on (GT: standing on)
right (GT: close by, right, same as)
left (GT: close by, left, same as)
standing on (GT: standing on)
chair (GT: chair)
standing on (GT: standing on)
ceiling (GT: floor)
window (GT: fireplace)
right (GT: close by, right)
close by (GT: left)
heater (GT: shelf)
box (GT: box)
left (GT: close by, front, left)
behind, right (GT: left)
none (GT attached to)
standing on (GT: standing on)
left (GT: left)
wall (GT: wall)
wall (GT: wall)
lying on, standing on (GT: standing on)
chair (GT: tv stand)
wall (GT: wall)
attached to (GT: attached to)
attached to (GT: attached to)
standing on (GT: standing on)
attached to (GT: attached to)
floor (GT: floor)
box (GT: pillow)
none (GT lying on)
window (GT: window)
kitchen counter (GT: sofa)
plant (GT: plant)
close by, right (GT: close by, right)
close by, left (GT: close by, left)
left (GT: front, left)
right (GT: behind, close by, right)
none (GT standing on)
curtain (GT: curtain)
attached to (GT: attached to)
left (GT: front, left)
none (GT behind, close by, right)
chair (GT: chair)
hanging on (GT: hanging on)
standing on (GT: standing on)
left (GT: front, left)
none (GT behind, close by, left)
wall (GT: wall)
chair (GT: plant)
standing on (GT: standing on)
attached to (GT: attached to)
none (GT standing on)
floor (GT: floor)
door (GT: door)
hanging on (GT: attached to)
wall (GT: wall)
wall (GT: wall)
desk (GT: computer desk)
attached to (GT: attached to)
attached to (GT: attached to)
standing on (GT: standing on)
floor (GT: floor)

Fig. 6.5. Qualitative scene graph prediction result mapped to the 2D image domain. © 2020 IEEE.

Tbl. 6.3. 3D semantic segmentation of our network on the validation split of *RIO* using the NYU40 class set.

mIoU	recall	prec.	tub	bed	shelf	cab.	chair	counter	curtain	desk	door	floor	other	pic	fridge	shower	sink	sofa	table	toilet	wall	wind.
53.0	65.8	72.4	76.6	41.8	22.5	52.4	74.3	21.1	70.0	16.7	36.2	81.9	29.5	23.1	33.4	78.5	61.9	67.4	54.2	84.7	71.2	61.7

Tbl. 6.4. 3D node segmentation on the validation split of *RIO* [295] with different embedding dimensions and input features for clustering.

Input Feature	**Embedding Vector Size**	AP	mAP25	mAP50
(1) embedding	64-dimensional	8.0	18.6	41.6
(1) embedding	128-dimensional	9.3	19.8	41.6
(1) embedding	256-dimensional	**9.4**	**21.7**	**44.0**
(2) embedding, semantic	256-dimensional	3.8	9.8	27.7
(3) embedding, spatial	256-dimensional	6.9	16.4	41.3
(4) embedding, semantic, spatial	256-dimensional	4.9	11.7	28.0

The embedding space of the feature vectors is illustrated in Fig. 6.6. The dimensionality of the high-dimensional feature vectors of *SGEN* is reduced with t-SNE [175]. The figure is generated by combining features of the elements of the validation set of *RIO* in one graph and using the corresponding ground truth classes as colors *e.g.* a) shows toilets in green, b) curtains in a light yellow, c) doors in red, d) chairs in a darker yellow and e) tables and desk in pink. Notably, different semantic classes are separated while similar semantic categories *e.g.* desks and tables are clustered together.

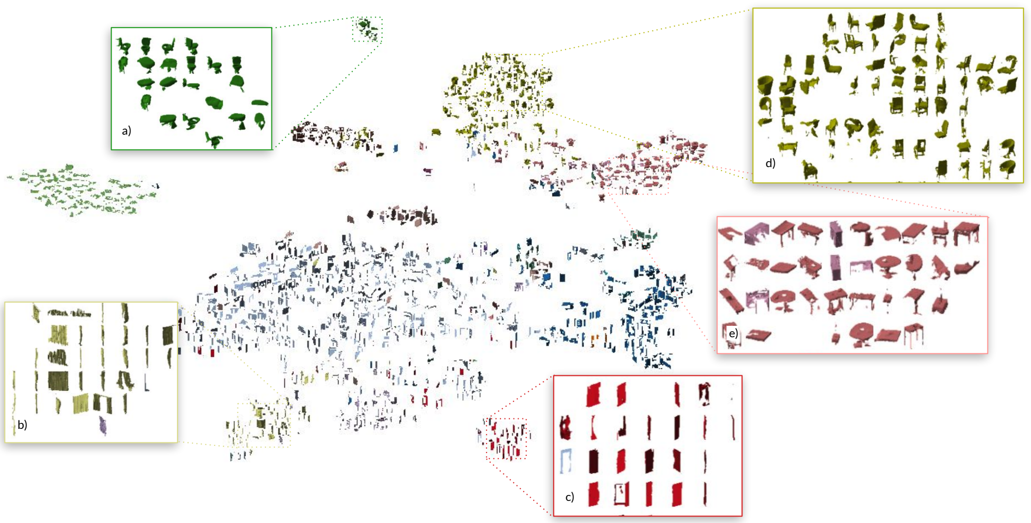


Fig. 6.6. Visualization of the embedding space learned in the 3D network of *SGEN* on 3D instances of *3DSSG*.

Finally, qualitative results of 3D semantic segmentation and instance embeddings are shown in Fig. 6.7. As it can also be seen in Tbl. 6.3, the segmentation quality is very good and *SGEN* is able to robustly segment challenging categories and even small objects.

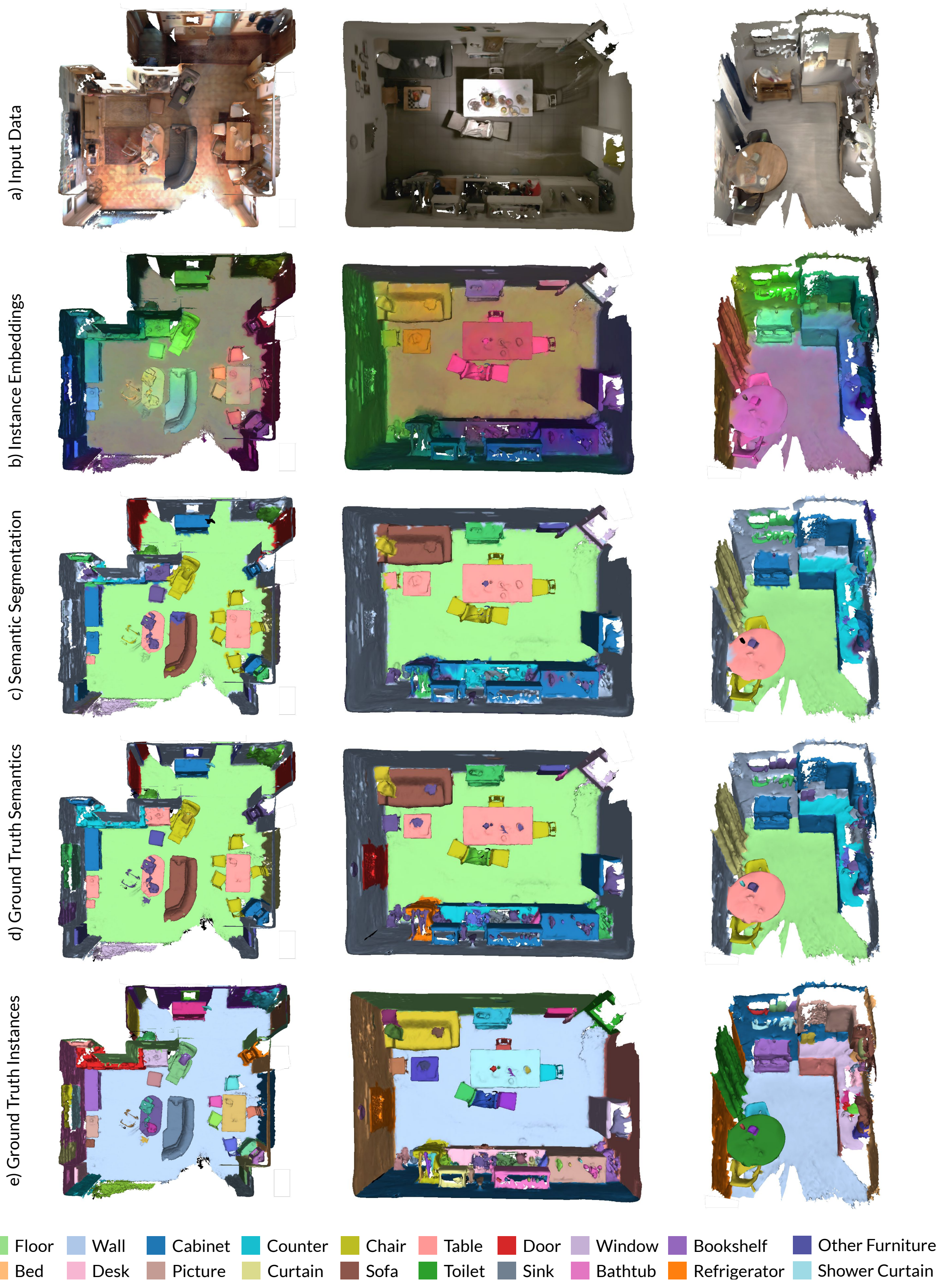


Fig. 6.7. Qualitative results of 3D semantic and instance embeddings of *SGEN* [297] on example scenes of the *RIO* dataset.

In Fig. 6.7, the a) ground truth instance and b) semantic segmentation are shown, together with c) predicted semantics and d) instance embeddings visualized with PCA [121]. Finally, the input data is shown in the top-row, Fig. 6.7a.

6.6 Incremental 3D Scene Graph Prediction

Notably, we also proposed an incremental scene graph prediction method, called *SceneGraph-Fusion* [308]. It constructs a semantic scene graph of a 3D scene in an online manner given a sequence of RGB-D frames, see Fig. 6.8. The method is more efficient than the other graph prediction models, *SGPN* and *SGEN*, as its optimized to operate on sequences and has similarities to the semantic segmentation approach presented in Sec. 5.3. Similarly to *FISS*, *SceneGraphFusion* is based on the geometric segmentation method, *InSeg* [270] which incrementally reconstructs and segments the scene. However, instead of a regression forest, a graph neural network is used for node and edge classification, and PointNet is employed as a low-level feature encoding. For more information, we refer the reader to the teaser, abstract of the following work for more details:

[308] Shun-Cheng Wu, **Johanna Wald**, Keisuke Tateno, Nassir Navab and Federico Tombari. *SceneGraphFusion: Incremental 3D Scene Graph Prediction from RGB-D Sequences*. *Conference on Computer Vision and Pattern Recognition (CVPR)*. © 2021 IEEE.

Scene graphs are a compact and explicit representation successfully used in a variety of 2D scene understanding tasks. This work proposes a method to incrementally build up semantic scene graphs from a 3D environment given a sequence of RGB-D frames. To this end, we aggregate PointNet features from primitive scene components using a graph neural network. We also propose a novel attention mechanism well suited for partial and missing graph data present in such an incremental reconstruction scenario. Although our proposed method is designed to run on sub-maps of the scene, we show it also transfers to entire 3D scenes. Experiments show that our approach outperforms 3D scene graph prediction methods by a large margin and its accuracy is on par with other 3D semantic and panoptic segmentation methods while running at 35 Hz.

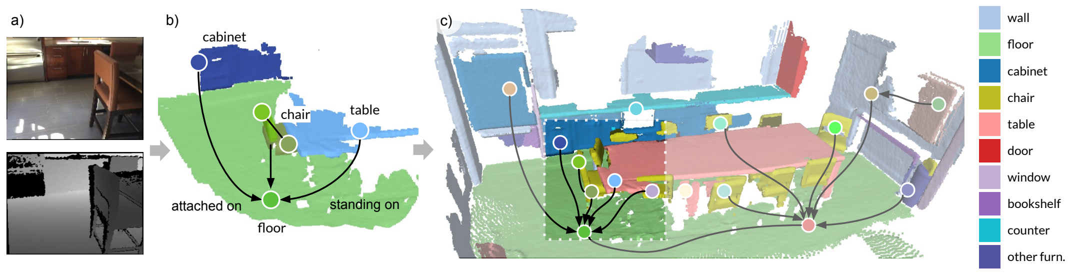


Fig. 6.8. A globally consistent 3D scene graph is constructed b) by merging predictions of a GNN given a geometric segmentation a). Predictions of the nodes of the same instance improve over time c) and a panoptic segmentation of the full 3D scene is produced as a by-product. © 2021 IEEE.

6.7 Conclusion

3D scene understanding does not only involve the extraction of information from 3D, including scene geometry and surface reconstruction, but also semantic categories of objects and structure as well as spatial and semantic interactions between entities. Even though semantic scene graphs are an important representation within 3D scene understanding, regressing them with a neural network from 3D data has been – to the best of our knowledge – for the first time explored by the methods proposed within this thesis.

We have presented three different methods for scene graph prediction, namely *Scene Graph Point Network (SGPN)* [296], *Scene Graph Embedding Network (SGEN)* [297] as well as *SceneGraphFusion* [308].

These methods go beyond object-level scene understanding and predict nodes and edges representing objects' semantics and relationships while directly operating on 3D scans [296] or sequences [308] of indoor scenes. All of the methods have been trained with *3DSSG* and *RIO*, two datasets that have been covered in Sec. 3.2 and 4.2 respectively. While *SGPN* is the first method to learn semantic scene graphs from point clouds, *SGPN* is directly applicable to real-world scans since it leverages a 3D semantic and instance network that regresses instance embeddings for node detection. Scene graphs are an abstract yet rich representation that could potentially be the basis of many other scene understanding methods and real-world applications, such as robotics and virtual or augmented reality.

In the following chapter, we show the applications of scene graphs in a novel task, called *image-based 3D scene retrieval in changing indoor environments* and teaser their usage in *semantic change detection*. We believe graphs are a great representation perfectly suitable to bridge domain gaps. This opens up new possibilities for other new novel use cases such as *text-based retrieval in (changing) 3D scenes* or *Visual Question Answering*.

Part IV

Persistence and Scene Changes

Scene Re-Localization
Wald et al., CVPR 2020

Camera Re-Localization
Wald et al., ECCV 2020

Object Instance Re-Localization
Wald et al., ICCV 2019

If you want to truly understand something try to change it.

— **Kurt Lewin**
(1890 – 1947)

Scene Re-Localization 7

The content of this chapter is based on the following publication:

[296] **Johanna Wald***, Helisa Dhamo*, Nassir Navab, and Federico Tombari. *Learning 3D Semantic Scene Graphs from 3D Indoor Reconstructions. Conference on Computer Vision and Pattern Recognition (CVPR).* © 2020 IEEE.

7.1 Introduction

Visual retrieval has a long tradition in computer vision. It is often used in combination with other tasks such as object detection, registration, or camera pose estimation [9, 10, 61, 81, 88, 172, 280]. If the source and query data do not share the same domain, or scene changes are present, retrieval becomes particularly difficult [1, 2, 17, 18, 53, 295, 298]. A vast amount of literature exists on retrieving CAD models from RGB images [115, 264] or 3D data directly [17, 18]. Proven by its success, we believe retrieval is potentially applicable and helpful for various tasks in changing scenes.

Autonomous agents that operate in real-world indoor environments often visit multiple known areas within their physical reach. One way to keep track of the known spaces is to store a list of 3D maps *e.g.* for every room or floor of a building. Once an autonomous agent is manually moved or switched on after some offline time, the first task at hand is scene identification, given the list of the previously visited spaces. While it is possible and even very likely that a scene has been modified since the last visit, scene changes need to be considered.

In the following, the task of *scene retrieval in changing indoor scenes* also referred to as *scene re-localization* is introduced. The scene re-localization application has been proposed in [296] and discusses the usage of scene graphs to bridge the domain gap of 3D and 2D data and help with changes in scene appearance. Once the scene was identified with a *scene re-localization* method, the camera is usually re-localized with respect to the retrieved 3D reference model, see chapter 8. Finally, updating the map *e.g.* in the form of understanding rigid motion when re-localizing objects is discussed in Sec. 9.

7.2 Scene Retrieval with Graphs

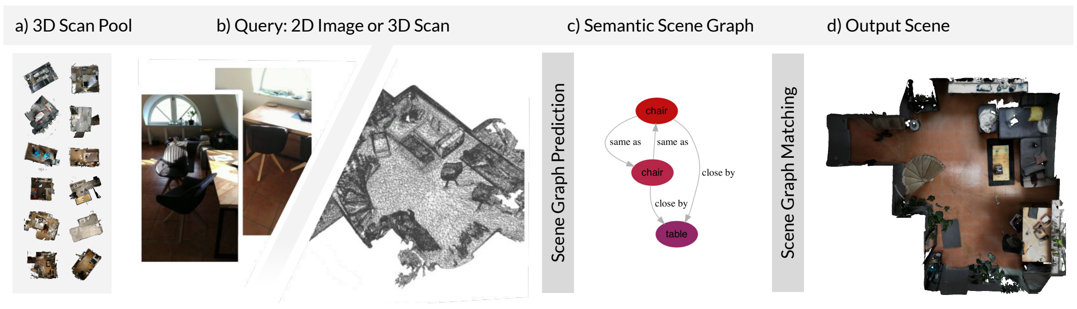


Fig. 7.1. In scene re-localization in changing indoor environments. Given an a) pool of source 3D indoor reconstructions, as well as a b) query image or 3D scan a c) scene graph based retrieval method is proposed to obtain d) the correct output scene in the source.

[296] introduced the task of *image-based 3D scene retrieval in changing indoor environments*. While it focuses on cross-domain matching, specifically 2D to 3D, we also experiment with 3D to 3D retrieval. While the source and query data is potentially from a different domain *e.g*. 2D vs. 3D, the scenes are taken at a different time and therefore feature changes, see Sec. 3.2. Hence, the challenge is not only to identify a 3D scene given a single 2D image or 3D point cloud but also be able to robustly handle natural scene changes such as varying illumination, the disappearance of an object, or even the reconfiguration of furniture. To evaluate the retrieval on *RIO* [295], a new test split is proposed featuring a set of rich 2D re-scan images and the corresponding ground truth reference 3D models.

In [296] we propose to solve this novel re-localization task using scene graphs, which act as a shared domain between 2D and 3D; furthermore, scene graphs are quite robust towards scene changes compared to other data representations such as 2D image data or 3D geometry. The proposed scene graph-based retrieval method is visualized in Fig. 7.1. It turns the a) source and b) query data into scene graphs and conducts matching and retrieval in the graph space by computing a pair-wise similarity between the query graph and all the data samples in the source pool, see Fig. 7.1. For this purpose, two different similarity functions, the Jaccard and Szymkiewicz-Simpson coefficient, are compared. The scene with the maximum similarity is the output of the graph-based retrieval and visualized in Fig. 7.1d. While the experiments mainly focus on 2D and 3D data, the method could – in theory – be applied to any domain that is transformable to scene graphs such as natural language or video. For example, such a high-level retrieval method could be applied in a search engine to find spaces that satisfy certain requirements such as the availability of specific objects *e.g*. two beds or ten chairs.

Since computing the similarity between graphs is an NP-complete problem, graph sets are used instead, which is an efficient, yet accurate approximation. To do so, the semantic scene graphs are first transformed into multi-sets containing node categories and semantic connections in the form of tuples. This process ensures to keep repetitive entities. Rather than matching graphs directly through their graph edit distance, a similarity score τ is computed based on these resulting multi-sets, see eq. 7.3. The Jaccard $\tau_J(\mathcal{H}, \mathcal{H}')$, eq. 7.1 and Szymkiewicz-

a) 3D-3D			Graph	Top-1	Top-3	Top-5	b) 2D-3D			Graph	Top-1	Top-3	Top-5
τ_S	$\mathcal{N}_{3D}$	$\mathcal{N}'_{3D}$	GT	0.86	0.99	1.00	τ_J	$\mathcal{N}_{2D}$	$\mathcal{N}'_{3D}$	GT	0.49	0.75	0.84
τ_S	$\mathcal{G}_{3D}$	$\mathcal{G}'_{3D}$	GT	0.96	1.00	1.00	τ_S	$\mathcal{N}_{2D}$	$\mathcal{N}'_{3D}$	GT	0.98	0.99	1.00
τ_J	$\mathcal{N}_{3D}$	$\mathcal{N}'_{3D}$	GT	0.89	0.95	0.95	τ_J	$\mathcal{G}_{2D}$	$\mathcal{G}'_{3D}$	GT	0.55	0.85	0.86
τ_J	$\mathcal{G}_{3D}$	$\mathcal{G}'_{3D}$	GT	0.95	0.96	0.98	τ_S	$\mathcal{G}_{2D}$	$\mathcal{G}'_{3D}$	GT	1.00	1.00	1.00
τ_J	$\mathcal{N}_{3D}$	$\mathcal{N}'_{3D}$	*3D-VRD*	0.15	0.40	0.45	τ_S	$\mathcal{N}_{2D}$	$\mathcal{N}'_{3D}$	*3D-VRD*	0.17	0.36	0.42
τ_J	$\mathcal{G}_{3D}$	$\mathcal{G}'_{3D}$	*3D-VRD*	0.29	0.50	0.59	τ_S	$\mathcal{G}_{2D}$	$\mathcal{G}'_{3D}$	*3D-VRD*	0.10	0.25	0.32
τ_J	$\mathcal{N}_{3D}$	$\mathcal{N}'_{3D}$	*SGPN*	0.32	0.46	0.50	τ_S	$\mathcal{N}_{2D}$	$\mathcal{N}'_{3D}$	*SGPN*	0.17	0.36	0.41
τ_J	$\mathcal{G}_{3D}$	$\mathcal{G}'_{3D}$	*SGPN*	0.34	0.51	0.56	τ_S	$\mathcal{G}_{2D}$	$\mathcal{G}'_{3D}$	*SGPN*	0.13	0.38	0.42
τ_J	$\mathcal{G}_{3D}$	$\mathcal{G}'_{3D}$	*SGEN*	**0.64**	**0.79**	**0.80**	τ_S	$\hat{\mathcal{G}}_{2D}$	$\hat{\mathcal{G}}_{3D}$	*SGEN*	**0.43**	**0.71**	**0.74**

Tbl. 7.1. Scene retrieval of changing indoor scenes in a) 3D to 3D (left) and b) 2D to 3D (right). We report the used similarity function followed by the query and target sets either in 2D/3D and using only nodes $\mathcal{N}$ or the full graph $\mathcal{G}$. © 2020 IEEE.

Simpson $\tau_S(\mathcal{H}, \mathcal{H}')$, eq. 7.2 are the two similarity functions which are explored in the following. They are generally defined as

$$\tau_J(\mathcal{H}, \mathcal{H}') = \frac{|\mathcal{H} \cap \mathcal{H}'|}{|\mathcal{H} \cup \mathcal{H}'|} \tag{7.1}$$

$$\tau_S(\mathcal{H}, \mathcal{H}') = \frac{|\mathcal{H} \cap \mathcal{H}'|}{\min(|\mathcal{H}|, |\mathcal{H}'|)}. \tag{7.2}$$

The multi-sets used in this operation refer to the objects $f_s(\mathcal{N})$, generic node edges $f_e(\mathcal{E})$ and predicates $\mathcal{R}$ defined as follows

$$f_s(\mathcal{N}) = \{(c_1^{(i)}, \sum_{j=1}^{|\mathcal{N}|} (c_1^{(j)} = c_1^{(i)}))\}_{j=1}^{|\mathcal{N}|}, \tag{7.3}$$

$$f_e(\mathcal{E}) = \{(n_o, n_s)\}_{i=1}^{|\mathcal{E}|}, \tag{7.4}$$

$$S = (f_s(\mathcal{N}), f_e(\mathcal{E}), \mathcal{R})). \tag{7.5}$$

When comparing two graphs, $\mathcal{G}$ and $\mathcal{G}'$, their similarity scores are combined for each multi-set $\mathcal{S}^{(i)}$ and $\mathcal{S}'^{(i)}$ respectively such that

$$f_\tau(\mathcal{S}, \mathcal{S}') = \frac{1}{|\mathcal{S}|} \sum_{i=1}^{|\mathcal{S}|} \tau(\mathcal{S}^{(i)}, \mathcal{S}'^{(i)}). \tag{7.6}$$

where τ is defined as either the Jaccard τ_J or the Szymkiewicz-Simpson metric τ_S respectively. Although the Jaccard coefficient is commonly used, the Szymkiewicz-Simpson is better suited when the two sets $\mathcal{H}$ and $\mathcal{H}'$ are different in size *e.g.* in the 2D-3D scenario.

The performance of the similarity scores and scene graph prediction methods is evaluated in Tbl. 7.1a) and 7.1b). While Tbl. 7.1b) tests matching of a 2D image from a re-scan against a pool of 3D references scenes, Tbl. 7.1a) evaluates the 3D counterpart where graphs are more similar in size, which is still a quite ambitious task. In particular, the changes between re-scan and reference make 3D to 3D retrieval challenging. For each re-scan (2D or 3D), the similarity is computed with the target graph, the tables then report the percentage of matches that are within the top-n most similar samples when ranked by similarity. In this experiment, 2D image graphs are obtained by rendering the predictions or ground truth 3D graphs as explained in Sec. 4.2.4.

To obtain an upper bound for the graph matching, the retrieval method is tested in isolation from the graph prediction quality by reporting the performance using ground truth graphs, see *GT*-columns in Tbl. 7.1. These *GT* experiments also allow to judge the effectiveness of the different similarity functions, namely $\tau_J(\mathcal{H}, \mathcal{H}')$ and $\tau_S(\mathcal{H}, \mathcal{H}')$. As expected, since image and 3D scene graphs are of different sizes, the Szymkiewicz-Simpson coefficient performs better in 2D-3D matching, while the Jaccard coefficient performs on-par in 3D-3D retrieval. The experimental results in Tbl. 7.1 further illustrate how scene context in the form of semantically rich relationships – in addition to object semantics – is beneficial for identifying the target scene; the matching accuracy is, therefore, higher when relations are included.

A final observation: compared to *3D-VRD* the predicted graphs from *SGPN* [296] of Sec. 6.3, and in particular *SGEN* of Sec. 6.4, have a higher matching accuracy.

7.3 Semantic Change Detection

Using ground truth scene graphs for 3D to 3D retrieval, see Tbl. 7.1, does not only give an upper bound for the graph matching method but interestingly, additionally detects high-level semantic scene changes. Notably, the change in a scene is the inverse of its ground truth similarity.

A visual example of this by-product is shown in Fig. 7.2. Only portions of the graph, namely nodes and edges, that have changed and could not be matched in the ground truth 3D-3D retrieval are shown.

In the first example, a chair was moved and is therefore not *close by* the bed anymore, thus the *chair-close by-bed*-relationship disappeared. Also, the green pillow with id 20 changed its position and is consequently not *close by* the other two pillows, yellow id 22 and purple id 21, anymore. In the second example in the bottom row, the bag appeared on the couch and the cushions moved, which is nicely reflected in the change graph.

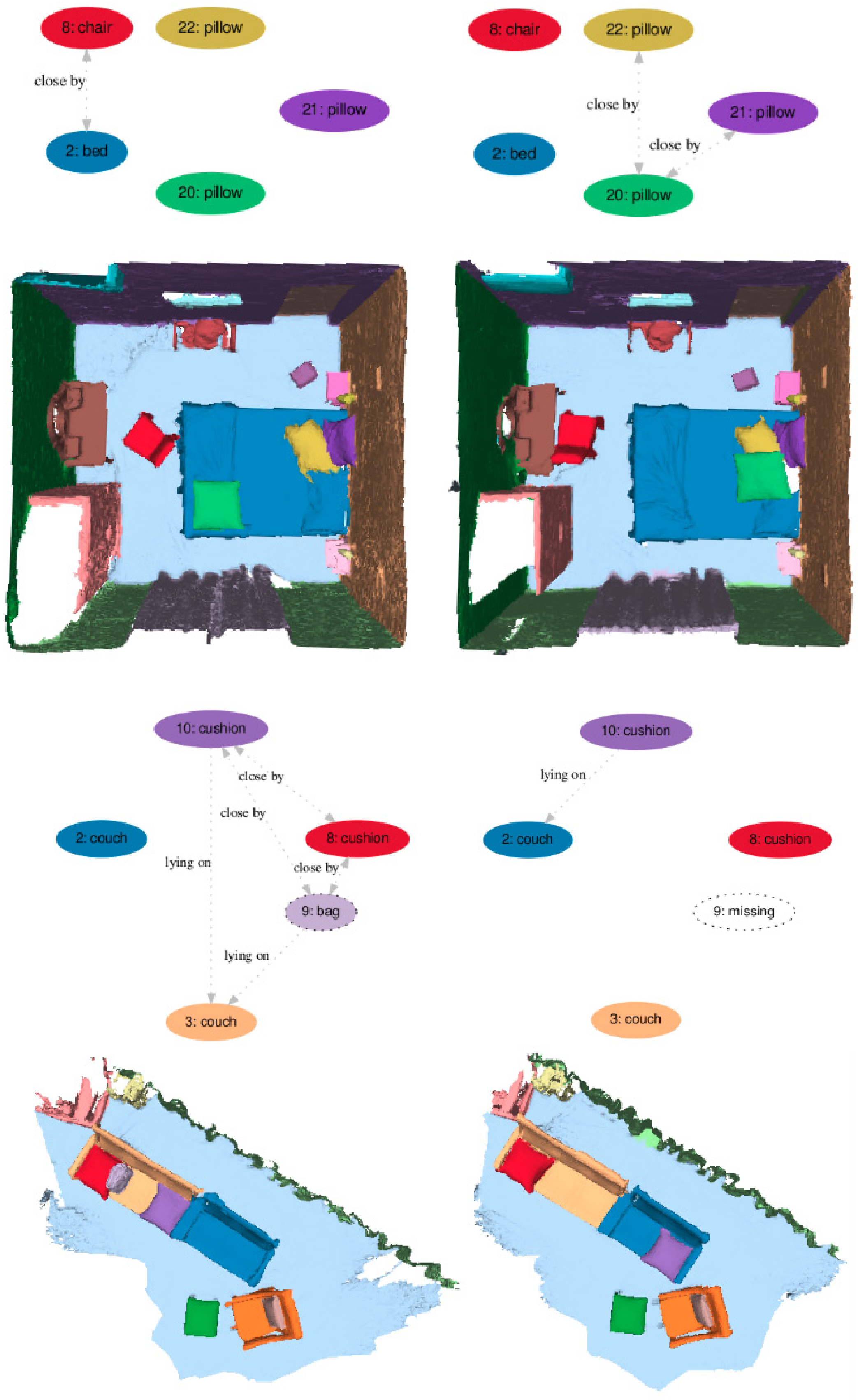


Fig. 7.2. Semantic change detection as a by-product of ground truth 3D-3D scene retrieval. Only the portion of the graph that has changed is shown. © 2020 IEEE.

8 Camera Re-Localization

The content of this chapter is based on the following publication:

[298] **Johanna Wald**, Torsten Sattler, Stuart Golodetz, Tommaso Cavallari and Federico Tombari. *Beyond Controlled Environments: 3D Camera Re-Localization in Changing Indoor Scenes*. *European Conference on Computer Vision*, 2020, Springer, Cham.

8.1 Introduction

In the following, methods are discussed that measure the amount of scene change of semantic, geometric, and visual nature given the frames of the re-scan sequence of an indoor scene. This allows quantifying the effect of scene change on the precision of state-of-the-art visual camera re-localization and enables – for the first time – to rate algorithms based on their ability to handle natural changes.

Notably, the experiments in Sec. 8.3 are conducted on the *RIO10* [298] dataset which has been introduced in Sec. 3.3. To do so, in various experiments several state-of-the-art methods are evaluated and even though other camera re-localization benchmarks such as *7-Scenes* [253] seem to be saturated our experiments show that indoor re-localization, especially in dynamic setups is far from being solved. For this purpose, a benchmark is created with a set of open challenges.

To evaluate the accuracy of a predicted camera pose, a novel evaluation measure, called *Dense Correspondence Re-projection Error (DCRE)* is used. *DCRE* correlates with the visual perception which stands in contrast to other typical evaluation metrics such as the absolute pose error, see Sec. 8.3.3. Notably, the evaluation framework and metrics are publicly available [1] to encourage the community and future research to focus no non-static indoor scene setups following the footsteps of other high-impact benchmarks that analyze scene changes in outdoor environments [19, 176, 243, 244].

8.2 Related Work

While a review of long-term camera re-localization benchmarks and datasets can be found in chapter 8, the respective methodological literature is discussed in the following section and is categorized into image retrieval (Sec. 8.2.1), direct pose regression (Sec 8.2.2), structure-based method (Sec. 8.2.3) and scene coordinate regression (Sec. 8.2.4).

8.2.1 Image Retrieval

Image retrieval approaches tackle camera re-localization by matching a given image against a pool of training samples [81, 88] using descriptors such as bags of binary words [81] or binary codes obtained from random ferns [88]. One of the significant disadvantages of image retrieval approaches is their limited generalization capability regarding poses that are far from the training trajectory.

Two common strategies to mitigate this drawback are to a) obtain the poses of the nearest neighbors of the query image, and average them to get the final pose estimate [22, 145, 281, 321]. Such a method potentially helps to predict poses that are *between* the training samples but still struggle with more distant, novel poses. Another technique to overcome this issue is b) matching the query image against a set of synthesized views of the scene [84]. While [84, 280] suggest to interpolate between several matches, [328, 334] use triangulation in combination with relative poses. A major challenge, in this case, is scalability, both in terms of memory and computational complexity, which involves calculating and store the descriptors for all the synthetic views and the computation required to match against them. Efficient memory and run time are achieved in place recognition algorithms such as DenseVLAD [280] and NetVLAD [10] by using efficient, compact feature descriptors. For example, DenseVLAD [280] uses compact VLAD descriptors to reduce memory usage and limits the number of synthetic views needed by only generating them at $5\,\mathrm{m}$ intervals, within $20\,\mathrm{m}$ of the database images at a fixed pitch. This limits the accuracy of the resulting poses but allows the methods to work at a larger scale. While trading pose accuracy for scalability, such methods perform mediocre when viewpoint and appearance changes are limited [243].

[1] github.com/WaldJohannaU/RIO10

8.2.2 Direct Pose Regression

Regressing the camera pose is one of the most recent camera re-localization research areas and the proposed methods are mostly based on pose regression neural networks [3, 127, 128, 129, 185, 306] but some use decision forests [123], GANs [37] or LSTMs [49, 293].

Direct pose regression generally performs worse than the later discussed structure-based or scene coordinate regression methods, see Sec. 8.2.3 and 8.2.4 respectively. This is likely due to their experimental setup, but they have also been proven to be useful for large-scale, coarse, RGB-only re-localization, mostly in outdoor setups, that require coarse poses only [129]).

An interesting set of experiments was conducted by Sattler *et al*. [245]. They show that direct pose regression and retrieval methods achieve very similar results. The generalization capabilities of pose regression to new, novel poses are very limited, and achieving highly accurate pose predictions is nearly impossible.

Several works [33, 152, 220, 284] improve camera pose regression accuracy by exploring relative poses. Impressively, some of the works [220, 284] achieve performance on par with state of the art. However, these methods rely on estimations from previous frames, making them ineffective camera tracking approaches compared to techniques that can re-localize in a single shot.

8.2.3 Structure-based Methods

Classical structure-based methods extract 2D features and match them with 3D keypoints of the scene. Found correspondences are filtered with RANSAC [76] to get the final camera pose estimate. An example that is still performing surprisingly competitive to more recent, learning-based methods is Active Search [240]. It matches SIFT-based vocabularies from 2D-to-3D and 3D-to-2D. Their main contribution is considering neighborhood 3D points after a 2D-to-3D match is established to validate if they can be used to establish additional 3D-to-2D correspondences.

Hierarchical localization approaches [113, 239, 242, 266, 267] use image retrieval as a first step of their pipeline and therefore significantly reduce the search space. Based on a set of proposals the feature matching is restricted and therefore quite efficient.

The literature that considered scene changes and long-term localization has proven to be effective when using learned features [65, 85, 239, 304]. These methods perform state-of-the-art on competitive benchmarks [243] outperforming other, more classical approaches based on feature matching [85, 266, 304]. Another possible solution to solve long-term camera re-localization is the inclusion of semantic segmentation [144, 246, 276, 277]. While this seems to work outdoors, it does not directly transfer to indoor scenes [267].

InLoc [266] matches the query image with reference images from a database using NetVLAD descriptors and then performs dense feature matching between the query and reference images using VGG-16 features to establish correspondences. Recently, this approach was extended by InLoc++ [267], which introduces an additional pose verification stage based on appearance, geometry, and semantics. In contrast, S2DHM [85] uses sparse-to-dense matching based on *hypercolumn* features [101] to improve robustness to changing conditions and efficiency, while HF-Net [239] first performs image retrieval using NetVLAD descriptors, ultimately SuperPoint [62] and DOAP [104] are used rather than VGG-16 or hypercolumn features.

Finally, object-centric methods [11, 16, 149, 237] focus on the creation of semantic object-level maps but are not effective in re-localization the camera when many objects change their location.

8.2.4 Scene Coordinate Regression

Scene coordinate regression often referred to as SCoRe, densely regresses the coordinates of a query scene image using a neural network [27, 29, 30, 31, 32, 64, 155, 156, 316], a random forest [28, 41, 42, 43, 89, 97, 186, 187, 188, 253, 287] or a combination of these [179].

The regressed scene correspondences are combined with PnP/Kabsch [122] to get hypotheses which are refined with RANSAC [76]. [27, 28, 29, 30, 31, 64, 155, 156, 186, 187, 316] use RGB input only, while [41, 42, 43, 97, 188, 253, 287] process RGB-D at test time which usually leads to better performance [43], even though RGB methods are quite close in performance. While most methods require to be trained offline, some are designed to run online [41, 42, 43]. One of the best scene regression methods is Grove v2 [43]. The regression forest variant of it performs best indoors while the neural network version shows great performance in outdoor setups [41].

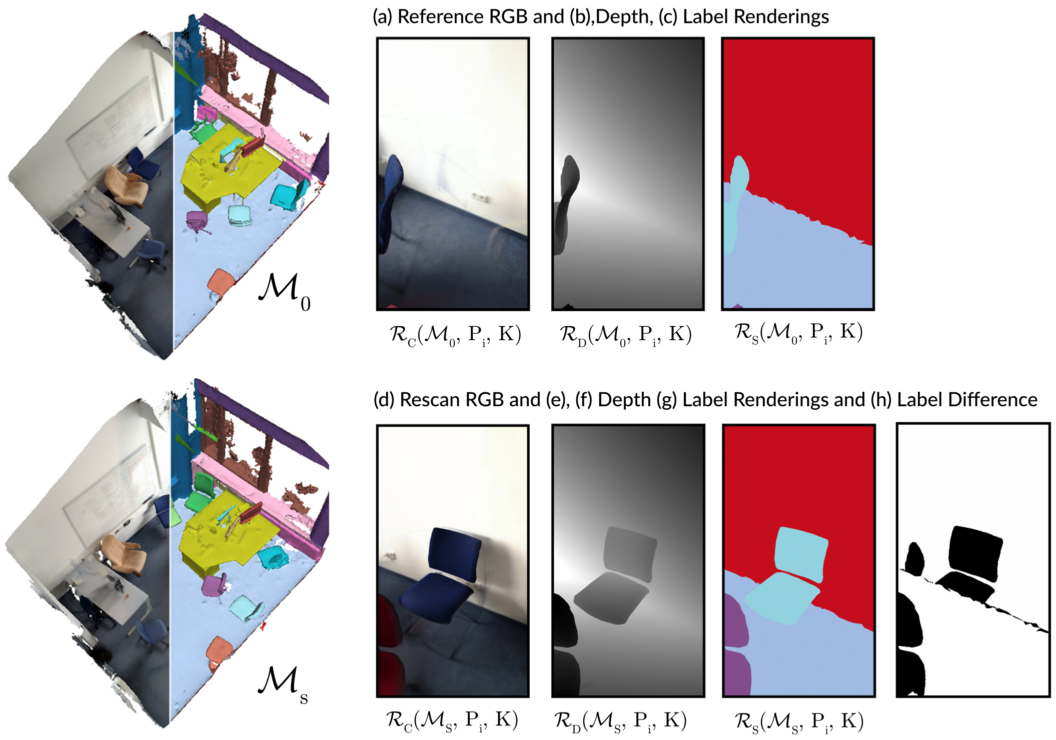
A few approaches do not fit neatly into any of the aforementioned categories and are neither a retrieval, direct pose regression, structure, or scene coordinate regression method. [286] use a continuous optimization technique to improve the poses of a retrieval forest by proposing a multi-scale navigation graph-based initial matching method. Nakashima *et al*. [196] is related to SCoRe but replaces the matching with a dense regression approach. Alternative approaches perform retrieval based on 3D data [61, 172]. The 3D data is usually built from several images or a constructed database [246]. Finally, some approaches use RANSAC [196, 266] or continuous pose optimisation [286] to refine the results of initial pose estimation.

8.3 Evaluation in Changing Indoor Scenes

Having described our long-term camera re-localization benchmark dataset in Sec. 3.3 the following section proposes an evaluation methodology. Novel ways to measure the amount of scene change are presented (see Sec. 8.3.1) and a new metric, namely the normalized absolute correspondence re-projection error is discussed to evaluate a predicted camera pose against its ground truth (see Sec. 8.3.3).

8.3.1 Quantifying Change in Indoor Scenes

In the following, we describe different ways to measure scene change in indoor scans and differentiate between semantic, visual, and geometric change.



(a) Reference RGB and (b),Depth, (c) Label Renderings
$\mathcal{M}_0$
$\mathcal{R}_C(\mathcal{M}_0, P_i, K)$
$\mathcal{R}_D(\mathcal{M}_0, P_i, K)$
$\mathcal{R}_S(\mathcal{M}_0, P_i, K)$
(d) Rescan RGB and (e), (f) Depth (g) Label Renderings and (h) Label Difference
$\mathcal{M}_s$
$\mathcal{R}_C(\mathcal{M}_S, P_i, K)$
$\mathcal{R}_D(\mathcal{M}_S, P_i, K)$
$\mathcal{R}_S(\mathcal{M}_S, P_i, K)$

Fig. 8.1. Scene change is computed from synthetic color, depth and semantics of the re-scan and reference 3D model $\mathcal{M}_0$ and $\mathcal{M}_s$ respectively.

The change values are computed for every RGB-D re-scan frame with respect to the references observation. To do so, rendered re-scan and reference images are first generated with color, depth, and semantics using the same camera view. This process utilizes the aligned instance segmentation and colored 3D reconstruction of *RIO*, see Sec. 3.2. Formally defined, rendering functions are introduced that produce RGB $\mathcal{R}_C$, depth $\mathcal{R}_D$ and semantic images $\mathcal{R}_S$ respectively. The rendering functions use the camera view $\mathbf{P} \in \mathbb{R}^{4\times4}$, camera parameters $\mathbf{K} \in \mathbb{R}^{3\times3}$ as well as 3D models $\mathcal{M}$ to generate an w $\times$ h image, see Fig. 8.1.

Visual Change

Based on these previously defined rendering functions a measure for the visual change can be computed. The visual difference of two 3D models $\mathcal{M}$ and $\mathcal{M}'$ viewed from $\mathbf{P}$ is computed given the rendered color images $\mathbf{I}_C = \mathcal{R}_C(\mathcal{M}, \mathbf{P}, \mathbf{K})$ and $\mathbf{I}'_C = \mathcal{R}_C(\mathcal{M}', \mathbf{P}, \mathbf{K}) \in \mathbb{R}^{\mathrm{w}\times\mathrm{h}}$, see Fig. 8.1. To measure visual change two change measures are introduced, the normalized correlation coefficient ρ_v

$$\rho_v = \frac{\sum_{\mathbf{u}}(\mathbf{I}_C(\mathbf{u}) - \mathbf{I}'_C(\mathbf{u}))^2}{\sqrt{(\sum_{\mathbf{u}} \mathbf{I}_C(\mathbf{u})^2) \cdot (\sum_{\mathbf{u}} \mathbf{I}'_C(\mathbf{u})^2)}}, \tag{8.1}$$

as well as the sum of squared differences ζ_v

$$\zeta_v = \frac{\sum_{\mathbf{u}}(\bar{\mathbf{I}}_C(\mathbf{u}) \cdot \bar{\mathbf{I}}'_C(\mathbf{u}))^2}{\sqrt{\sum_{\mathbf{u}}(\bar{\mathbf{I}}_C(\mathbf{u}) \cdot \bar{\mathbf{I}}'_C(\mathbf{u}))^2}}, \tag{8.2}$$

where $\bar{\mathbf{I}}_C(\mathbf{u})$ is defined as follows

$$\bar{\mathbf{I}}_C(\mathbf{u}) = \mathbf{I}_C(\mathbf{u}) - \frac{1}{\mathrm{w} \cdot \mathrm{h}} \sum_{\mathbf{u}'} \mathbf{I}(\mathbf{u}'). \tag{8.3}$$

On our dataset $\mathcal{M}$ is the 3D model of the re-scan taken from the test or validation set while $\mathcal{M}'$ is the reference in the train set and the poses $\mathbf{P}$ used in this procedure are always poses from the re-scan sequences.

Semantic Change

Next, we introduce a measure to quantify semantic change ζ_s. Similarly, ζ_s reports the percentage of semantically changed pixels given 2D instance masks. This computation is based on the instance images viewed from $\mathbf{P}$ of re-scan and reference $\mathbf{I}_S = \mathcal{R}_S(\mathcal{M}, \mathbf{P}, \mathbf{K})$ and $\mathbf{I}'_S = \mathcal{R}_S(\mathcal{M}', \mathbf{P}, \mathbf{K})$ respectively. $\mathbf{S}^{(s)}_{\{\mathbf{I}_S, \mathbf{I}'_S\}}$ is defined as the set of pixels with valid instance in $\mathbf{I}_S$ and $\mathbf{I}'_S$. ζ_s and computed as follows

$$\zeta_s = \frac{1}{\left|\mathbf{S}^{(s)}_{\{\mathbf{I}_S, \mathbf{I}'_S\}}\right|} \sum_{\mathbf{u} \in \mathbf{S}^{(s)}_{\{\mathbf{I}_S, \mathbf{I}'_S\}}} \mathbb{1}\left[\mathbf{I}_S(\mathbf{u}) \neq \mathbf{I}'_S(\mathbf{u})\right]. \tag{8.4}$$

Geometric Change

Finally, the geometric change, called ζ_g reports the average geometric difference which is namely the average pixel offset in millimeters. Its computation is based on synthetic depth images from $\mathbf{P}$ of the 3D models $\mathcal{M}$ and $\mathcal{M}'$ defined as $\mathbf{I}_D = \mathcal{R}_D(\mathcal{M}, \mathbf{P}, \mathbf{K})$ and $\mathbf{I}'_D = \mathcal{R}_D(\mathcal{M}', \mathbf{P}, \mathbf{K})$. The geometric change is then computed as

$$\zeta_g = \frac{1}{\left|\mathbf{S}^{(d)}_{\{\mathbf{I}_D, \mathbf{I}'_D\}}\right|} \sum_{\mathbf{u} \in \mathbf{S}^{(d)}_{\{\mathbf{I}_D, \mathbf{I}'_D\}}} \|\mathbf{I}_D(\mathbf{u}) - \mathbf{I}'_D(\mathbf{u})\|_2 \tag{8.5}$$

where $\mathbf{S}^{(d)}$ is similarly, the set of 2D point coordinates with a depth value in a valid range for both, $\mathbf{I}_D$ and $\mathbf{I}'_D$. A very high geometric change is *e.g.* given when a door is in the view or if objects are re-arranged.

Given these measures, statistics can be computed on all sequences of *RIO10*, see Sec. 3.3, and it can be determined that scene 4 has a high value overall, including semantic and geometric change while scene 8 and 9 are exceptionally high in visual change.

8.3.2 Measuring Re-Localization Performance

Having described ways to measure visual, semantic, and geometric change, evaluation of camera re-localization performance is discussed next.

Traditionally, evaluating predicted camera poses uses absolute pose errors. Given estimated 3D rotations $\{\hat{\mathbf{R}}_i\}_i^{\mathrm{N_F}}$ and 3D positions $\{\hat{\mathbf{t}}_i\}_i^{\mathrm{N_F}}$ as well as corresponding ground truth data $\{\mathbf{R}_i\}_i^{\mathrm{N_F}}$ and $\{\mathbf{t}_i\}_i^{\mathrm{N_F}}$, where $q(\mathbf{R})$ is quaternion of $\mathbf{R}$, the absolute pose error consist of an absolute translation error which simply is the Euclidean distance measured in meters (see eq. 8.6), and an orientation error based on the rotation matrix (see eq. 8.7), defined as follows

$$\Delta t_i = ||\hat{\mathbf{t}}_i - \mathbf{t}_i||^2, \tag{8.6}$$

$$\Delta\vartheta_i = ||\frac{180}{\pi} \cdot 2 \cdot \arccos[q(\mathbf{R}_i)^{-1} \cdot q(\hat{\mathbf{R}}_i)]||. \tag{8.7}$$

This absolute pose error is then used to compute a recall e_a, the percentage of data samples that are within a given error threshold $(\epsilon_t, \epsilon_\vartheta)$ as follows

$$e_a(\epsilon_t, \epsilon_\vartheta) = \frac{1}{\mathrm{N_F}} \sum_{i=1}^{\mathrm{N_F}} \mathbb{1}\,[\Delta t_i < \epsilon_t \wedge \Delta\vartheta_i < \epsilon_\vartheta]. \tag{8.8}$$

The outlier measure reports the opposite, the frames outside of some error threshold, computed as follows

$$\bar{e}_a(\epsilon_t, \epsilon_\vartheta) = \frac{1}{\mathrm{N_F}} \sum_{i=1}^{\mathrm{N_F}} \mathbb{1}\,[\Delta t_i \geq \epsilon_t \vee \Delta\theta_i \geq \epsilon_\vartheta]. \tag{8.9}$$

Often, these thresholds are manually selected *e.g.* $(0.05\,\mathrm{m},\ 5°)$ and $(0.1\,\mathrm{m},\ 10°)$ in indoor scenes. Furthermore, these absolute values do not correlate with visual perception at a given frame and pose. Specifically, when objects are close to the camera sensor, a one-pixel change could result in a pose error of just a few millimeters, but a few meters if objects are far away. An alternative to the absolute pose error was introduced by PoseNet [129]. They report the median translation and rotation error, namely $\widetilde{\Delta t}$ and $\widetilde{\Delta\vartheta}$. Although if both of the medians are low, it is not ensured that they are both from the same frame. In [298] we proposed an alternative, a metric that correlates with the visual perception, which measures the absolute pixel difference between the ground truth image and the predicted view. This new measure is discussed in the following Sec. 8.3.3.

8.3.3 Dense Correspondence Re-Projection Error

The following sections introduce the *Dense Correspondence Re-Projection Error (DCRE)* which is the ground truth 2D re-projection error of the points of a 3D model projected on the image plane and visualized in Fig. 8.2. This 2D displacement is computed based on the ground truth and projected camera poses. To do so, first a depth map

$$\mathbf{I}_D = \mathcal{R}_D(\mathcal{M}_s, \mathbf{P}_{s,i}, \mathbf{K}_s), \tag{8.10}$$

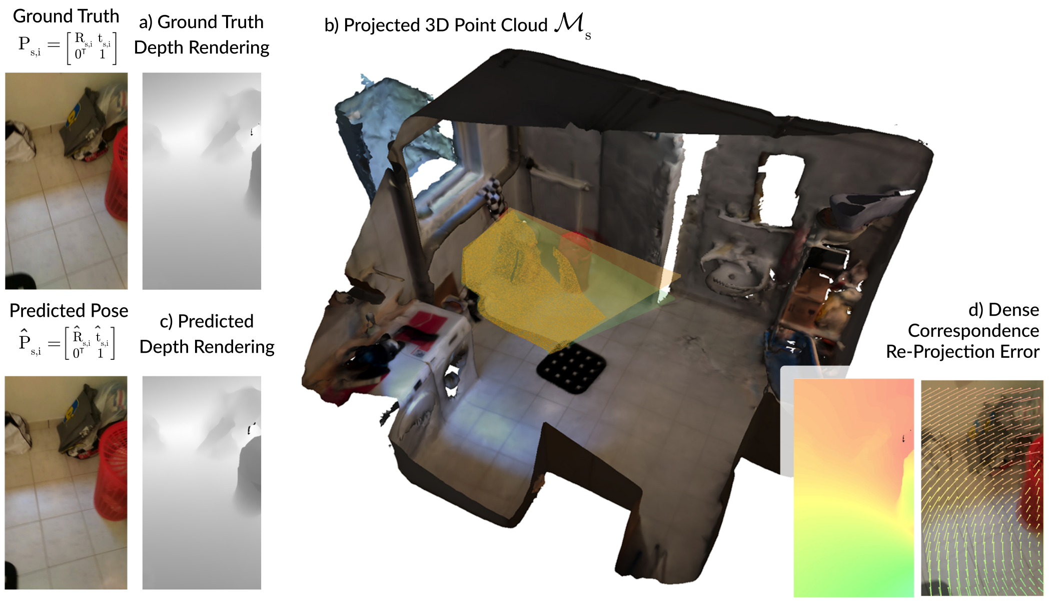

Ground Truth
$P_{s,i} = \begin{bmatrix} R_{s,i} & t_{s,i} \\ 0^T & 1 \end{bmatrix}$
a) Ground Truth Depth Rendering
b) Projected 3D Point Cloud $\mathcal{M}_s$
Predicted Pose
$\hat{P}_{s,i} = \begin{bmatrix} \hat{R}_{s,i} & \hat{t}_{s,i} \\ 0^T & 1 \end{bmatrix}$
c) Predicted Depth Rendering
d) Dense Correspondence Re-Projection Error

Fig. 8.2. The DCRE $\delta_f^{(i)}(u)$ is the 2D displacement error computed by back-projecting the synthetic depth map $\mathbf{I}_D = \mathcal{R}_D(\mathcal{M}_s, \mathbf{P}_{s,i}, \mathbf{K}_s)$ with the ground truth camera pose (a) $\Pi_{\mathbf{K}_s}^{-1}$ and is (b) then transformed with the prediction $\hat{\mathbf{P}}_{s,i}^{-1}\mathbf{P}_{s,i}$. Please note, the ground truth is tinted in an orange hue while the predicted view is shown in a greenish color [298].

is rendered by projecting the 3D reconstruction $\mathcal{M}_s$ of scene s using the ground truth camera view $\mathbf{P}_{s,i}$ of the re-scan. $\mathbf{I}_D$ is then turned into a point set by back-projection via $\Pi_{\mathbf{K}_s}^{-1}$. The 3D point cloud is subsequently transformed $\hat{\mathbf{P}}_{s,i}^{-1}\mathbf{P}_{s,i}$ and rendered back with the predicted view by using $\Pi_{\mathbf{K}_s}$. The displacement error $f_\delta^{(i)}(\cdot)$ of a frame i is computed as the offset between the original pixel position $\mathbf{u}$ and the newly computed one. This summarizes as follows

$$f_\delta^{(i)}(\mathbf{u}) = \Pi_{K_s}(\hat{\mathbf{P}}_{s,i}^{-1}\mathbf{P}_{s,i}\Pi_{\mathbf{K}_s}^{-1}(\mathbf{u}, \mathbf{I}_{D_i})) - \mathbf{u}. \tag{8.11}$$

Based on the 2D displacement error $f_\delta^{(i)}(\cdot)$, the normalized, mean length of all displacements $e_{DCRE}^{(i)}$ is computed as

$$e_{DCRE}^{(i)} = \frac{1}{\left|\mathbf{S}_{\mathbf{I}_{D_i}}^{(d)}\right|} \sum_{\mathbf{u} \in \mathbf{S}_{\mathbf{I}_{D_i}}^{(d)}} \min\left(\frac{||f_\delta^{(i)}(\mathbf{u})||}{\sqrt{\mathrm{w}^2 + \mathrm{h}^2}}, 1\right). \tag{8.12}$$

A sequence recall value $e_f(\cdot)$, similar to eq. 8.8, can be computed using the DCRE as follows

$$e_f(\epsilon_f) = \frac{1}{\mathrm{N_F}} \sum_{i=1}^{\mathrm{N_F}} \mathbb{1}\left[e_{DCRE}^{(i)} < \epsilon_f\right]. \tag{8.13}$$

Normalizing the error is essential since $5\,\mathrm{px}$ might not be much in $5k$ images, but it could be significant at lower image resolutions. We believe that normalization is not strictly necessary for this dataset as all images have the same resolution but could help future research compare errors across datasets.

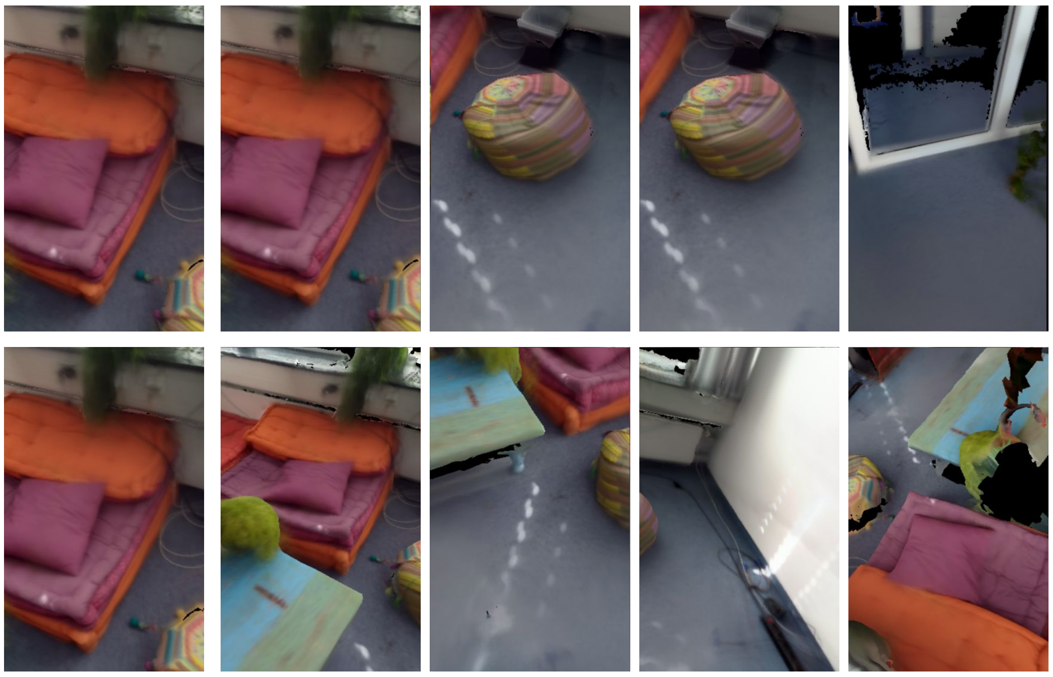

(a) (0.07, 1.8) / 2.5 **(b)** (0.71, 10) / 11.3 **(c)** (0.19, 46) / 60.3 **(d)** (0.7, 46) / 80.1 **(e)** (3.2, 57) / >100

Fig. 8.3. Comparison of ground truth (top) and predicted poses (bottom) on rendered images of scene 01 of *RIO10* [298]. For each of these rendered images we report the absolute pose error and the newly introduced DCRE measure $(\Delta t, \Delta\vartheta)$ / e_{DCRE} reported in [meters], [degree] and [%] respectively.

Furthermore, as previously mentioned, the *DCRE* correlates with the appearance of the scene, see Fig. 8.3. Notably, it is essentially a dense re-projection error of ground-truth correspondences and to the best of our knowledge, it has not yet been proposed to evaluate camera re-localization. In contrast to more classical applications of the re-projection error *e.g.* calibration, dense correspondences are obtained from rendered depth maps generated from a 3D model. These result in dense ground truth mappings of pixel features. Notably, the *DCRE* is 0 if the pose is perfectly estimated, which is not the case when minimizing the re-projection error of a point cloud obtained from Structure from Motion (SfM) due to natural noise [4]. Also, our correspondences are dense and therefore available for most of the image, where the depth is valid, instead of just textured regions and corners.

While absolute pose errors are two distinct values, *DCRE* combines the pose error in a single number which is beneficial when *e.g.* plotting histograms or choosing thresholds as done in cumulative plots, see Fig. 8.4.

In the following, we evaluate the performance of different camera re-localization approaches that perform best on public benchmarks in static indoor scenes or long-term outdoor environments. We analyze their efficiency on *RIO10*, see Sec. 3.3, using both standard metrics as well as our newly proposed *DCRE*. Furthermore, the measures to quantify scene change are used to understand better how different methods are affected.

State-of-the-art methods of different categories are used, ranging from hand-crafted structure-based methods such as Active Search [241], to learned RGB [65, 239] or RGB-D methods [42, 43]. HF-Net [239] was trained for $50k$ iterations on the $52k$ images available in the training set of *RIO10*. Finally, for completeness retrieval approaches such as NetVALD, DenseVLAD [10, 280] are also included. Each method is evaluated on all $165\,744$ test images of the *RIO10* dataset.

The reported measures in Tbl. 8.1 are the classical metrics namely $e_f(0.05)$ and $e_f(0.15)$ as well as $e^{(i)}_{DCRE}$. For completeness we additionally list the absolute pose error $e_a(\cdot)$ with the thresholds $(\epsilon_t, \epsilon_\vartheta) = (0.05\,\text{m}, 5°)$ and define

$$\bar{e}_f(\epsilon_f) = \frac{1}{p}\sum_{i=1}^{p} \mathbb{1}\left[e^{(i)}_{DCRE} \geq \epsilon_f\right]. \tag{8.14}$$

$$e_f(\epsilon_f) = \frac{1}{\mathrm{N_F}}\sum_{i=1}^{\mathrm{N_F}} \mathbb{1}\left[e^{(i)}_{DCRE} < \epsilon_f\right]. \tag{8.15}$$

Tbl. 8.1. Camera Re-localization performance on *RIO10* [298]. *N/A* lists the fraction of missed predictions.

		Inlier				**Outlier**		
	Method	$e_a(0.05m, 5°)$	$(\widetilde{\Delta t}, \widetilde{\Delta \theta})$	$e_f(0.05)$	$e_f(0.15)$	N/A	$\bar{e}_a(0.5m, 25°)$	$\bar{e}_f(0.5)$
🟦	Active Search [240]	0.0696	$(0.16, 4.68)$	0.171	0.2430	0.684	0.0891	0.028
🟩	Grove [42]	0.2300	$(0.06, 1.74)$	0.334	0.3910	0.452	0.144	0.106
🟥	Grove v2 [43]	0.2742	$(0.11, 2.60)$	0.406	0.4850	0.162	0.332	0.262
🟦	HFNet [239]	0.0182	$(1.56, 72.33)$	0.057	0.0980	0	0.900	0.714
🟪	HF-Net Trained [239]	0.0725	$(0.84, 24.17)$	0.180	0.2880	0	0.685	0.427
🟧	D2Net [65]	0.1553	$(0.55, 14.90)$	0.365	0.5060	0.014	0.513	0.194
🟪	NetVLAD [10]	0.0002	$(0.93, 31.44)$	0.006	0.1250	0	0.798	0.452
⬛	NetVLAD 20-NN [10]	0.0003	$(0.88, 38.36)$	0.007	0.0999	0	0.840	0.531
🟨	DenseVLAD [280]	0.0003	$(0.98, 32.26)$	0.008	0.1240	0.006	0.772	0.520
🟫	DenseVLAD 20-NN [280]	0.0002	$(1.00, 50.26)$	0.008	0.0967	0.006	0.827	0.612

To capture the amount of mispredictions produced by the different methods, an outlier measure is reported using thresholds $(\epsilon_t, \epsilon_\vartheta) = (0.5\,\text{m}, 25°)$ and $e_f(0.5)$. The fraction of missing poses is also measured in the NaN-column. Notably, Active Search [240] has a relatively high NaN value but rarely produces outliers, while the retrieval methods, NetVLAD [10] and DenseVLAD [280], perform quite poorly overall and interpolating the 20 nearest retrieved neighbours, obtains very similar performance, see Tbl. 8.1. D2-Net [43] achieves the best performance according to the *DCRE* and traditional error metrics, but indicate that approximately half of the predicted poses of the test frames are incorrectly estimated.

Additionally to the numbers in Tbl. 8.1, cumulative plots are shown in Fig. 8.4 using both, *DCRE* as well as absolute pose errors. Cumulative plots help to understand better the underlying characteristics of a camera pose re-localization method. It *e.g.* can be seen that the best-performing methods Grove v2 and D2-Net tend to produce poses that have quite high pose errors indicated by a steady increase of its graphs in the plot. Active Search however behaves

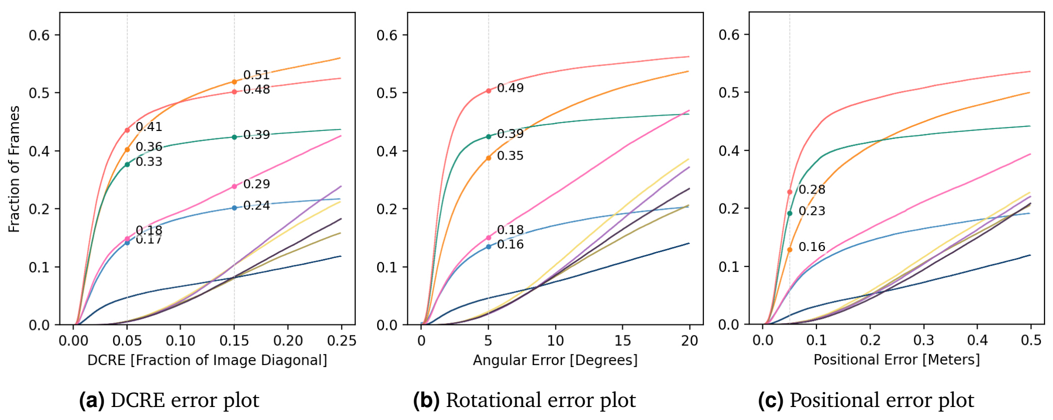


Fig. 8.4. Cumulative recall plot of the absolute (b), (c) and DCRE error (a) of *RIO10* [298]. See Tbl. 8.1 for the respective color legend.

differently and instead reliably provides quite accurate poses or skips the prediction completely, seen by the quick saturation and plateauing of the respective line graph. Notably, an ideal algorithm would produce a line graph approximating a step function.

Furthermore, a clear correlation of scene change and a methods' performance is observable in Fig. 8.5. It plots the re-localization error of all frames with growing visual ζ_v, ρ_v, geometric ζ_g and semantic ζ_s change.

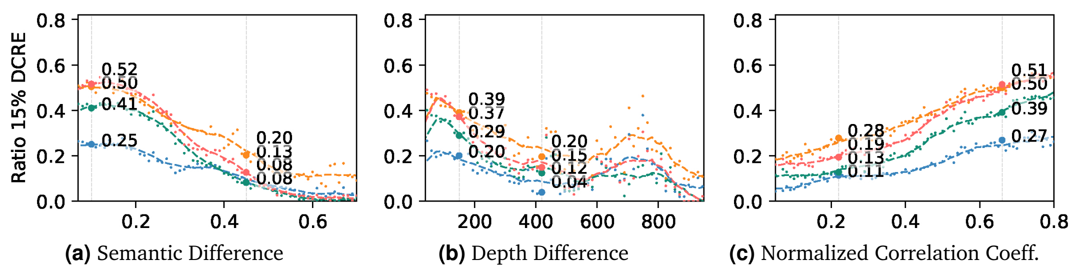


Fig. 8.5. Effect of scene change with respect to methods' performance $e_f(0.15)$ on *RIO10* [298]. Dashed lines highlight running averages. See Tbl. 8.1 for the respective color legend.

Object Re-Localization vs. Camera Re-Localization

When scenes change, camera poses are potentially not correctly computed *e.g.* when a new, never observed, or moved object dominates the view. An interesting example of this problem is visualized in Fig. 8.6. Given the query image (a) it is pretty challenging to estimate the correct camera pose given only the train scan (c) since the bed, a major object in the new camera image has moved, see (b). Since the correct reference view is very different from the test image, an algorithm likely produces a pose with respect to the object as shown in (e), even though the correct view is (d).

Tbl. 8.2, reports the fraction of cases when the camera was aligned with respect to an object rather than the global scene. While this might be a substantial problem, only solvable with more context, it happens quite rarely.

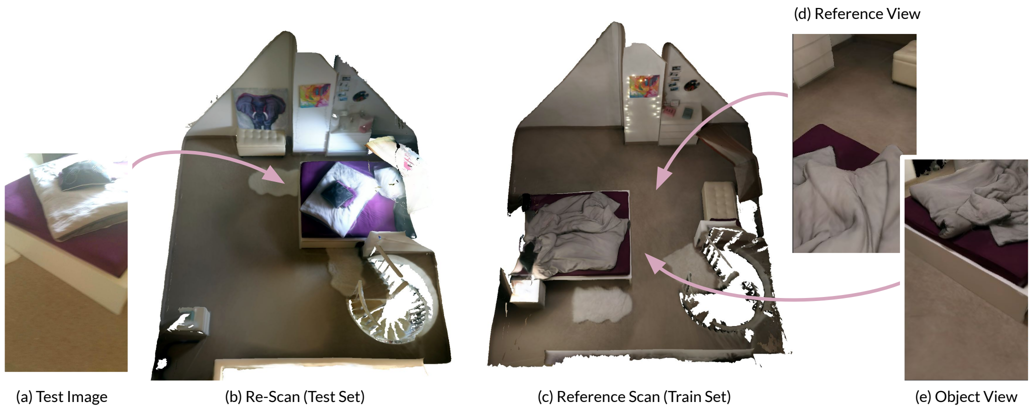


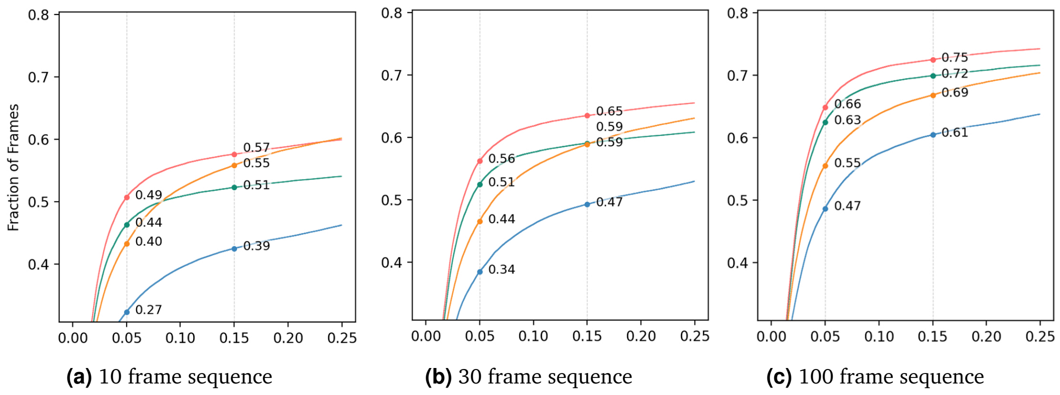
Fig. 8.6. Camera localization ambiguities in presence of objects movements. Given a test image (a) in the re-scan (b) the correct prediction is (d) in the global, reference scene but a method would more likely align (e) with respect to the object.

Tbl. 8.2. Camera Re-localization failures where a pose is computed with respect to an object rather than the global scene.

AS [240]	Grove [42]	v2 [43]	HFNet [239]	Trained [239]	D2Net [65]	NetVLAD [10]	DenseVLAD [280]
0.149	0.065	0.051	0.005	0.065	0.033	0.016	0.014

Sequences

To evaluate if more context improves re-localization performance, experiments are conducted using sequences. Instead of a single image, the algorithm localizes multiple successive frames at the same time.



(a) 10 frame sequence **(b)** 30 frame sequence **(c)** 100 frame sequence

Fig. 8.7. Cumulative DCRE recall plot with sequences of (a) short, (b) medium, and (c) long length on *RIO10* [298]. See Tbl. 8.1 for the respective color legend.

In Fig. 8.7 sequences of different lengths with s_Δ short being 10, medium 30 and long 100 are evaluated. 10 and 30 are sequences available in interactive applications where the camera pose is not needed right away and a bit of waiting is acceptable. Collecting a few frames could overcome potential motion blur or help with object-change ambiguities. Sequences of length 100 are, on the other side obtainable in applications where real-time is not a constraint and which are less interactive.

The experiments show that sequences indeed help significantly to localize but the performance is still not good enough to beat the benchmark. In terms of prediction with sequences, two different implementations for RGB and RGB-D methods are proposed.

Active Search and HF-Net use sequences with known intrinsic and relative poses within a generalized camera model [213] which has multiple centers of projection and all of the estimated matches help with the prediction of this general camera. A minimal solver and RANSAC [76] are used to solves the generalized perspective-n-point pose (gPnP) problem [137, 147, 265, 290].

Grove v1 and v2, on the other hand, localize each frame independently from each other. The poses of each frame of the test sequence are transformed relative to the previous pose. The output is then iteratively clustered, which mostly results in a single large cluster with a few outliers. Dual-quaternion blending [124] is used on the center of the cluster to obtain the final camera poses.

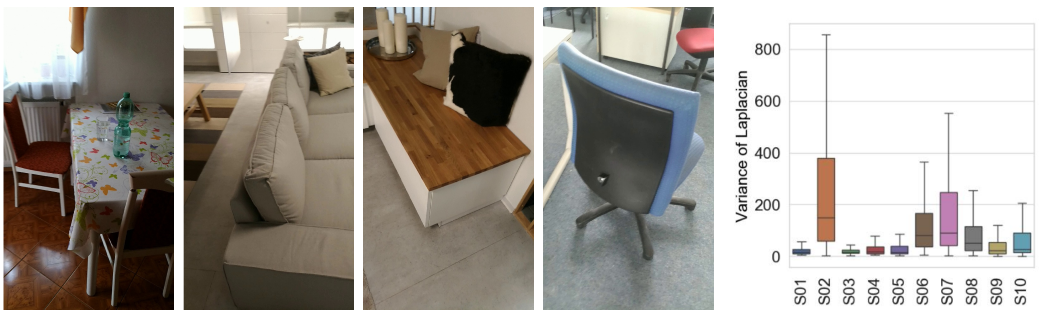
The code of both of these sequence merging techniques has been released in open-source repositories allowing other research to be built upon.

8.3.4 Classifying Frame Difficulty

We have discussed several reasons why image-based localization is challenging, mostly centered around scene changes but – don't worry – there is more. For the sake of completeness, we want to also mention a few other ways to measure if an image is *hard* or *easy* to re-localize. These include, but are not limited to a) the availability of scene context, b) sharpness and textures as well as c) novelty of the test poses with respect to the training trajectory.

Texture and Blurriness

Many image-based re-localization methods rely on texture features extracted from the image. Therefore, blur or plain surfaces are known to potentially cause issues. We propose to use the Variance of Laplacian σ to measure both motion blur and lack of textures. A selection of images with the respective VoL values are visualized in Fig. 8.8.



(a) 405.924 **(b)** 111.441 **(c)** 62.002 **(d)** 53.17 **(e)** Variance of Laplacian Statistics

Fig. 8.8. Example frames with corresponding VoL values (a-d) and (e) averaged Variance of Laplacian (VoL) for the sequences in *RIO10* [298].

On the right side, statistics across all frames of the different sequences in *RIO10* are reported. Sharp images with many features, such as the most left image (a) have a pretty high texture value. Notably, RGB-feature-based localization algorithms perform better on images with a high VoL value as they are more informative [241].

View Coverage

The second factor to consider when assessing the re-localization difficulty of a frame is the lack of context. We suggest measuring the context by back-projection the depth image. The context is then estimated by computing the convex hull of all the 3D points obtained from the depth maps, including its camera center. Examples of a low, medium, and high field of view are shown in Fig. 8.9.

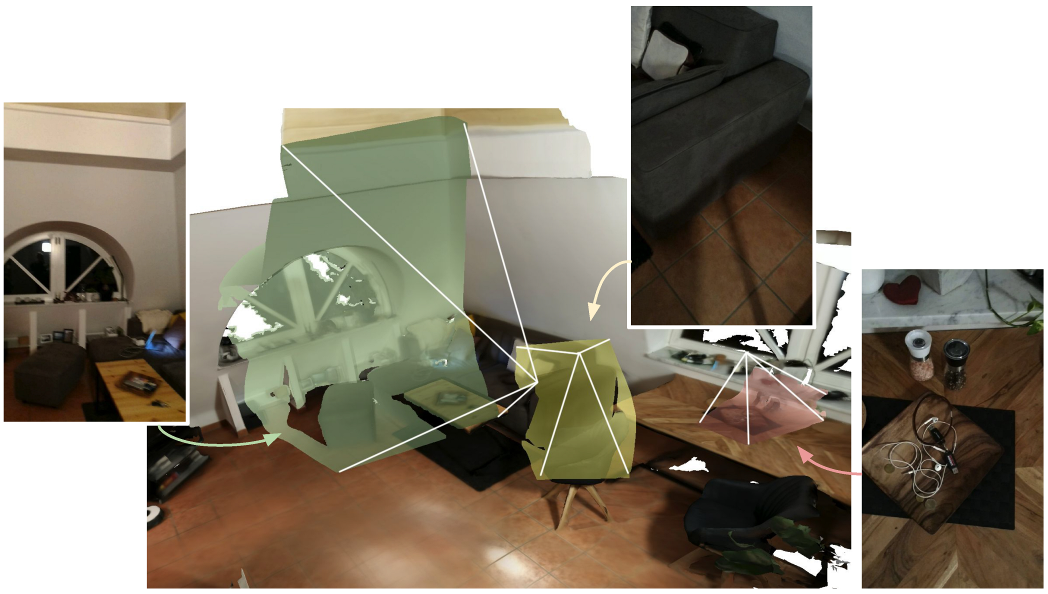


Fig. 8.9. low (red), medium (yellow) and high (green) field of view/frame coverage on scene 10 of *RIO10* [298].

Notably, methods struggle with low-context frames but interestingly, high-context images seem to be challenging too, potentially caused by noise, motion blur or the number of objects present in the scene.

Pose Novelty

The final measure quantifies the novelty of a test pose concerning the training pose trajectory. It therefore is an indicator if the scene has been observed in a similar pose before or if the particular view is novel in that sense *e.g.* from another unusual angle. It is computed given all the camera poses of the train set $\{\mathbf{P}'_i\}_{i=1}^{\mathrm{N_F}}$ and a query pose $\mathbf{P}$ from a re-scan in the test or validation set.

The pose novelty η is calculated as the smallest difference between the query pose $\mathbf{P}$ and all the train poses using a similarity function $\epsilon_\eta(\cdot)$ such that

$$\eta = \min_{\forall \mathbf{P}'_i \in \{\mathbf{P}'_i\}_{i=1}^{\mathrm{N_F}}} \epsilon_\eta(\mathbf{P}, \mathbf{P}'_i). \tag{8.16}$$

A selection of similar views of the test and train set is visualized in Fig. 8.10. The similarity measure $\epsilon_\eta(\cdot)$ used is eq. 8.16 is the newly introduced *DCRE* metric. In the following experi-

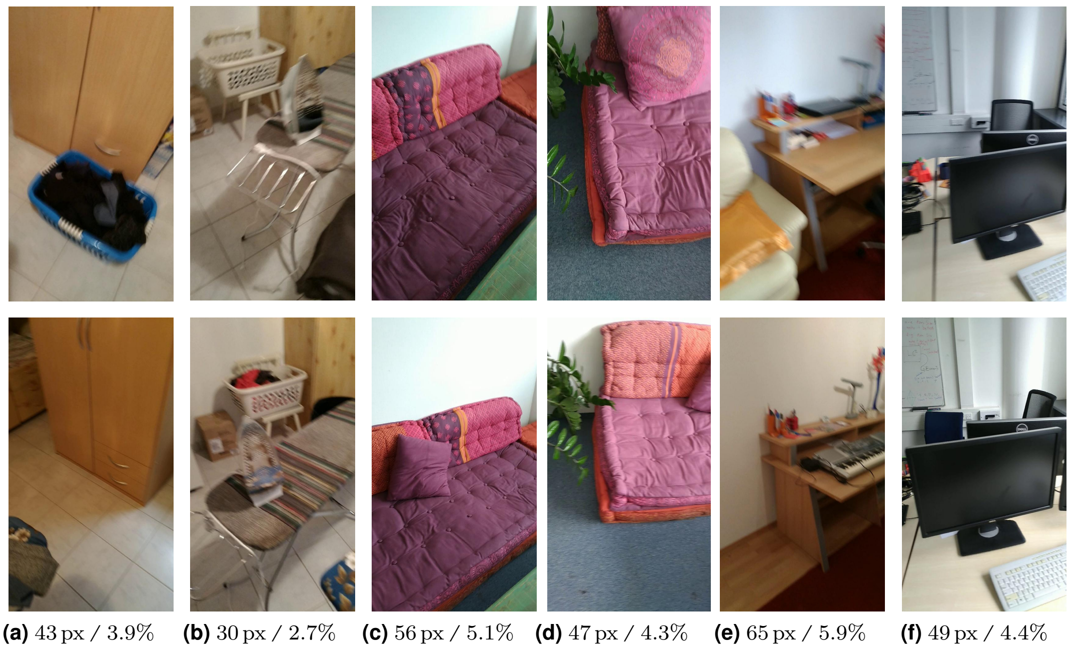

(a) 43 px / 3.9% **(b)** 30 px / 2.7% **(c)** 56 px / 5.1% **(d)** 47 px / 4.3% **(e)** 65 px / 5.9% **(f)** 49 px / 4.4%

Fig. 8.10. Test images (top row) and the corresponding nearest neighbours in the training sequence of *RIO10* [298] the with the DCRE score (bottom) of scene 01 (a,b), scene 2 (c), scene 4 (e) and scene 7 (f).

ment, the texture, view coverage, and pose novelty are used together with the change measure to set up a set of challenges, see Tbl. 8.4.

Tbl. 8.3. Challenges based on the proposed measures (see Tbl. 8.4) on *RIO10* [298].

Filter	# Images	Visual	Semantic	Geometric	Sharpness	FoV	Pose Novelty
well-textured	84 946				$\sigma > 33$	$\nu \in [0.2, 8]$	$\eta \leq 650$
texture-less	84 704				$\sigma \leq 33$	$\nu \in [0.2, 8]$	$\eta \leq 650$
high context	63 041				$\sigma > 7.2$	$\nu > 2.4$	$\eta \leq 650$
medium context	62 264				$\sigma > 7.2$	$\nu \in [0.9, 2.4]$	$\eta \leq 650$
low context	55 344				$\sigma > 7.2$	$\nu \leq 0.9$	$\eta \leq 650$
novel poses	20 281				$\sigma > 7.2$	$\nu \in [0.2, 8]$	$\eta > 500$
not novel poses	36 495				$\sigma > 7.2$	$\nu \in [0.2, 8]$	$\eta \leq 150$
easy changes	5783	$\rho_v > 0.8$	$\zeta_s \leq 0.1$	$\zeta_g \leq 30$	$\sigma > 7.2$	$\nu \in [0.2, 8]$	$\eta \leq 650$
hard changes	13 363	$\rho_v \leq 0.7$	$\zeta_s > 0.4$	$\zeta_g > 30$	$\sigma > 7.2$	$\nu \in [0.2, 8]$	$\eta \leq 650$

Each method is tested for a different test set *filters e.g.* only on images with high textures, low context or hard changes, see Tbl. 8.3 and 8.4.

It can be observed that Active Search benefits more from high textured images than *e.g.* Grove and Grove v2. Furthermore, it is no surprise that the image retrieval methods, NetVLAD or DenseVLAD perform best on non-novel poses but struggle severely with novel views. D2-Net seems to perform best overall in these experiments. However, its performance also significantly drops from 0.735 to 0.244 when faced with images that contain *hard changes* similar to Grove and Grove v2 where the performance decreases by 0.538 and 0.602 respectively.

Tbl. 8.4. Camera Re-localization performance on *RIO10* [298] $e_f(0.15)$ with different subsets (see Tbl. 8.3).

Filter	AS	Grove	Grove v2	HF-Net	Trained	D2-Net	NetVLAD	DenseVLAD
well-textured	0.388	0.471	0.570	0.162	0.404	**0.608**	0.162	0.160
texture-less	0.156	0.345	0.430	0.054	0.214	**0.448**	0.107	0.102
high context	0.296	0.447	0.559	0.129	0.354	**0.630**	0.124	0.135
medium context	0.303	0.423	0.514	0.132	0.343	**0.559**	0.156	0.153
low context	0.218	0.349	**0.425**	0.063	0.223	0.406	0.118	0.105
novel poses	0.236	0.327	**0.413**	0.074	0.229	0.407	0.097	0.110
not novel poses	0.442	0.631	0.715	0.239	0.577	**0.775**	0.299	0.307
easy changes	0.405	0.616	0.714	0.226	0.468	**0.735**	0.242	0.237
hard changes	0.113	0.078	0.112	0.022	0.115	**0.244**	0.056	0.049

8.4 Conclusion

In the previous section, we have described a series of metrics for quantifying changes in indoor scenes, see Sec. 8.3.1, that build upon the long-term indoor camera re-localization dataset, *RIO10*, see Sec. 3.3. For the first time, the performance of camera re-localizers is evaluated in terms of robustness towards changes, filling a major gap in the literature. The work demonstrated that when exposed to scenes with changes of visual, geometric, and semantic nature induced by rigid/non-rigid object motions and varying lighting, many state-of-the-art methods perform poorly. Additionally, in Sec. 8.3.2, a new metric *DCRE* was introduced to measure camera re-localization performance which correlates with visual perception in contrast to previously used metrics.

Using the change measures and the *DCRE* metric, the behavior of camera re-localizers was analyzed on frames that captured rigidly moving subjects, and it was found in particular that significant changes such as moving objects are generally a problem. In such a scenario, a method potentially re-localizes with the object that dominates the view rather than with the global scene which potentially is not even wrong depending on the remaining background context that is given within the frame of interest. The results of our recent benchmark for state-of-the-art re-localizers reveal that none of the state-of-the-art methods are capable of managing changes well enough and there is significant space for improvement. When sequences are processed instead of single frames, the accuracy increases but not significantly, leaving room for further exploration and research.

We believe long-term camera re-localization in indoor scenes requires acquiring higher-level scene understanding using semantics, object understanding, and parsing of dynamic changes, similarly to approaches that have been presented in part III of this thesis. In the future, we expect long-term camera pose prediction to become closely coupled with general scene understanding tasks such as semantic parsing of the scene or persistent object-level SLAM.

Object Re-Localization 9

The content of this chapter is based on the following publication:

[295] **Johanna Wald**, Armen Avetisyan, Nassir Navab, Tombari Federico* and Matthias Nießner*. *RIO: 3D Object Instance Re-Localization in Changing Indoor Environments*. Oral, *International Conference on Computer Vision*. © 2019 IEEE.

9.1 Introduction

Having discussed scene and camera re-localization in chapter 7 and 8 respectively, this section focuses on objects. For this purpose, [295] introduced a new, novel task which addresses re-localization of objects in changing indoor scenes[1]. Given the objects of an RGB-D scan, the goal of object instance re-localization is to compute the transformations that align the reference objects with the re-scan. Understanding object association across time helps to parse scene changes and could potentially be beneficial for camera and scene re-localization. It might even develop into an important task for higher-level persistence SLAM and other applications in AR/VR or robotics.

It is challenging since it requires finding adequate correspondences on the objects and the scene, which is particularly hard when the environment changes additionally to noise and varying scanning patterns. Classical approaches use handcrafted descriptors such as FPFH [232]

[1]https://waldjohannau.github.io/RIO/

or SHOT [279]. These are, however not very effective, similarly to learned features *e.g.* 3DMatch [60, 324] since they are primarily trained in a self-supervised fashion and work best on static setups. Even when trained with dynamic data, the descriptors are not rich enough and struggle to capture correspondences within these changing environments which is understandable, as they were originally designed to match static key points.

The scarcity of supervised training data of dynamic 3D indoor scenes has been discussed earlier in chapter 3. It also introduces the *RIO* dataset in Sec. 3.2 which is utilized in the following for training and evaluation purposes. The scenes in *RIO* feature many different temporal changes, including annotations of rigidly moving objects between references and re-scans, see Fig. 9.1. The transformations $\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3$, which are the desired output of the tackled object instance re-localization task, align the 3D data of the couch table and the two chairs in the reference (left) to the re-scan observations (right).

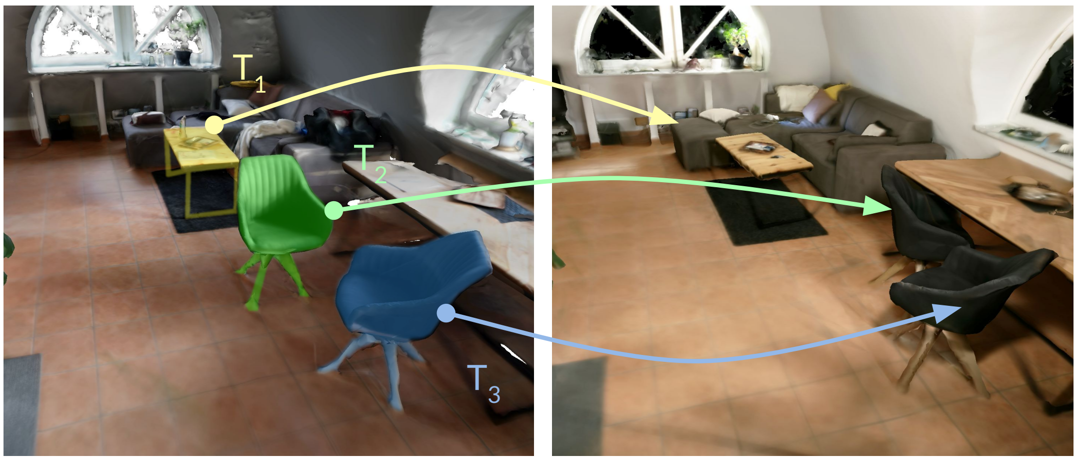


Fig. 9.1. 3D object instance re-localization computes transformations of rigidly changed instances from a source to a target reconstruction in *RIO* [295].

To solve this, a fully convolutional multi-scale network is proposed capable of learning geometric features in these complex environments. It is trained in a self-supervised manner on *RIO* using TSDF (truncated signed distance function) patches of different resolutions extracted on rigidly moving objects. This way, change-invariant local features are learned that outperform baselines in both correspondence matching and object instance re-localization, see respective experiments in Sec. 9.4.

9.2 Related Work

In the following section a review of of the related methodological literature is given. We specifically refer to methods for RGB-D scene understanding (see Sec. 9.2.1) and describe 3D object localization and keypoint matching methods (see Sec. 9.2.2) before we discuss approaches for long-term change detection (see Sec. 9.2.3).

9.2.1 RGB-D Scene Understanding

Scene understanding approaches use RGB-D data as input and represent the scene's 3D geometry either in the form of a volumetric or point-based representation *e.g.* within a surfel-based SLAM framework. Semantic knowledge is then extracted using *e.g.* Random Forests [285, 299] or CNNs [182, 195] while other methods incorporate semantics by segmenting the 2D images directly [182, 228, 229].

Object-level SLAM in the contrarily builds a semantic scene for SLAM [181, 237]. While these methods are incremental, there also exist a large amount of non-incremental approaches such as [55, 108, 216, 218, 222] that process the scene directly and learn semantics or instances while operating on the full 3D scene. A comprehensive overview about static scene understanding and most prominently semantic segmentation in online and offline setups has been given in Sec. 5.2 and 6.2.2. Notably, all of the discussed works – regardless of the type of approach – assume static setups and non-changing configurations, which stand in contrast to our long-term setup.

Finally, a large field of research tackles dynamics or object-level dynamics in particular. Rünz *et al.* proposes the generation of a globally consistent semantic map paired with tracking of moving of objects in a video stream [229]. The map representation is a dense surfel map, and scene regions are segmented and temporally tracked over time. This approach was later improved in *MaskFusion* [228] which uses Mask-RCNN [103] to detect, extract, and associate entities of a scene and impressively operates in real-time. Both methods [228, 229] employ a memory-efficient surfel map representation where it is non-trivial to extract empty space. This stands in contrast to *MID-Fusion* [312] which constructs a dynamic object-level octree-based volumetric representation that similarly claims to be able to track objects and the camera in real-time. More recently, *SplitFusion* [158] proposed an approach that additionally incorporates non-rigid surfaces. There exists a vast literature around non-rigid, and dynamic scene reconstructions [5, 82, 201, 230, 255, 337], which is different from our low-dynamic setup where changes happen off-camera and are therefore quite different *e.g.* capturing long-range changes. More details on state-of-the-art 3D reconstruction with RGB-D cameras are given in a survey by Zollhöfer *et al.* [336]. Notably, all the mentioned approaches use an explicit representation of surfaces while, more recently, implicit encodings have been suggested to represent scenes [26, 263, 332].

9.2.2 3D Object Localization and Keypoint Matching

3D object localization and pose estimation have a long history in computer vision research. Many methods follow the *classical pipeline* which combines keypoint detection and description followed by correspondence matching. In such a framework, key points are extracted on a source object and a target scene, descriptors are then computed based on the point cloud data, and the resulting feature vectors are finally matched. Object poses are then often obtained by first filtering outliers and applying an optimization algorithm to obtain the best object transformation given the remaining correspondences.

Traditionally, proposed descriptors were handcrafted encodings such as SHOT [279] or FPFH [232] which today still show impressive performance and robustness in a variety of tasks. As machine learning became more common, attention shifted to learned 3D encodings. These are capable of describing different 3D representations – such as meshes or point clouds – in a rich and discriminate latent space [60, 216, 324]. A comprehensive overview of feature descriptors has been given in Sec. 5.2.1. Despite their promising performance on several tasks like correspondence matching and registration, they are designed for static environments and do not generalize to dynamic scenes.

In this work, we go beyond this assumption and locate given 3D objects within a source scan in a cluttered target mesh reconstructed at a later point in time. This is challenging since it requires a robust feature encoding, ideally even invariant to 3D noise and changing local object context.

9.2.3 Long-Term Scene Understanding

In the robotics literature, changing environments where objects change position over time are sometimes referred to as *low dynamic environments*. In an early work, Walcott-Bryant *et al*. proposes a dynamic pose graph approach that detects changes and updates a sparse SLAM map accordingly [294]. Similarly, [146] proposes an RGB-D SLAM method based on pose-graphs where nodes are clustered into objects based on their covariance values. Even though they operate in 3D indoor space, their method models the localization task as a 2D problem.

[268] proposed a change detection method that operates on panoramic street-view images of urban environments. They ignore the dynamics of cars and pedestrians and instead consider only long-term, structural changes for map updating. In the 3D domain, Nunez *et al*. proposes a change detection algorithm that uses a gaussian mixture model to detect changes in the current 3D observations given a previous reference map [206]. Similarly, [72] describe an algorithm based on a TSDF-volume that can detect moving objects. It automatically obtains a database of discovered objects and can extract the static geometry in the form of a dense scene reconstruction. They also release a small dataset of changing rooms but lack semantic annotations; see Sec. 3.1 for more details. While all these works are relevant as they operate on changing scenes, they do not focus on obtaining poses of changing object instances as the method proposed in the following section.

9.3 3D Object Instance Re-Localization

A new data-driven approach is proposed to solve the re-localization problem by leveraging a 3D correspondence network to find matching features in 3D scans. It utilizes multi-scale patches to learns a change-invariant encoding of the neighborhood around a key point. Correspondences are then found and processed within a RANSAC [76] framework. The remaining keypoint pairs are then used to compute the 6DoF transformation. Sec. 9.3.1 gives more information about the network architecture while Sec. 9.3.2 explains in-depth how the final object poses are obtained.

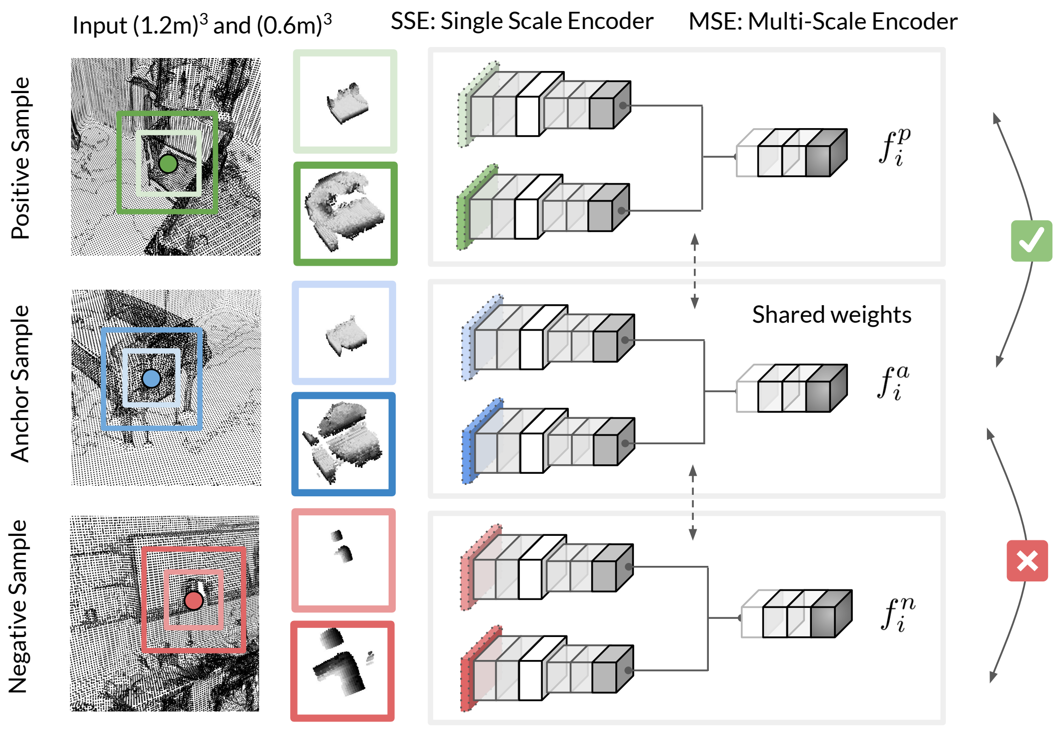


Fig. 9.2. RIO triplet network architecture [295] trained with anchor samples (blue), positive samples (green) and negative samples (red). © 2019 IEEE.

9.3.1 Network Architecture

The proposed network architecture is visualized in Fig. 9.2. It is designed as a triplet network and therefore maximizes the distance between the anchor and negative and minimizes the distance of the anchor and positive feature. The features are produced by processing different scales, namely a single scale encoder, *SSE* and a multi-scale encoder, called *MSE*. Needless to say, weights are shared in the *SSE* and *MSE* branches respectively, see respective dotted arrows in Fig. 9.2.

The network's output dimension of both single scale encoder branches are identical; they are therefore concatenated before being fed into the *MSE*. After applying non-padded convolutions and pooling layers, the feature vectors ends up being of size 512.

Such a multi-scale architecture helps to encode small details as well as high-level semantic features. Notably, in the experimental section, we show how such an architecture significantly outperforms other networks that use a single-scale architecture.

The input of the networks is a volumetric encoding. For each 3D point, the *TSDF* data is extracted for the anchor and positive sample, on the reference and re-scan respectively. The negative sample is obtained from another, randomly selected scene. The two different scaled inputs are both encoded in a $32 \times 32 \times 32$ grid with a voxel size of $1.875\,\mathrm{cm}/1.2\,\mathrm{m}^3$ and

$3.75\,\mathrm{cm}/0.6\,\mathrm{m}^3$ respectively. Notably, the raw *TSDF* is inverted in the first layer of the network to obtain a high gradient near the object surface [259]

$$\hat{f}_{TSDF}(\mathbf{x}) = 1 - |f_{TSDF}(\mathbf{x})|. \tag{9.1}$$

A triplet loss based on the ℓ_2 distance, see eq. 9.2, between the features of the positive, anchor and negative patches is optimized using Adam [131] with an initial learning rate of 10^{-3} and a margin of $\ell_\alpha = 1$ such that

$$\ell_2(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{|\mathrm{p}|}(\mathrm{p}_i - \mathrm{q}_i)^2}. \tag{9.2}$$

$$\sum_{i=1}^{N} N\Big[||f_a^{(i)} - f_p^{(i)}||_2^2 - ||f_a^{(i)} - f_n^{(i)}||_2^2 + \ell_\alpha\Big] \tag{9.3}$$

Our method extracts static *TSDF* patches in a self-supervised way similar to 3DMatch [324] for pre-training purposes. To do so, sub-sequences of varying lengths are processed to generate partial 3D reconstructions of the same scene with different scanning patterns. Key points in the overlapping region are then obtained by detected 3D corners on one scene using Harris 3D [254]. The response of the keypoint detector is then optimized for the key points on the other scene via non-maxima suppression using a small radius around the 3D location. All key points that have a response value above a threshold are included in the training set of our network.

Our approach aims to create a feature encoding that describes the local neighborhood of a 3D point in a change invariant way. To ensure the network is also able to deal with dynamically changing environments additionally, training data on moving objects is required.

To do so, hand-annotated rigid transformations are utilized to extract dynamic patches on scans of the *RIO*-dataset. They are generated in a very similar way as the static ones but additionally use object transformations. First, key points are extracted on objects known to have changed their location; then these key points are transformed to the reference by using the provided 6DoF ground truth transformation. The corresponding 3D patch on the reference is then found by optimizing the response around a small radius similar to how it is done in the static case.

Based on the pre-trained network of static scenes, a fine-tuning stage with dynamic data forces the network to learn higher-level concepts rather than overfit to low-level features. To ensure this, the first layers are frozen and consequently, only the multi-scale encoder branch is trained. This is required as there is way less dynamic data available and helps to avoid over-fitting to the dynamic training patches.

Also, identical to the static pre-training scenario, negative samples are extracted from random scenes while adding a small percentage of hard negatives. These negative patches are around key points on objects that disappeared from a scene, therefore potentially showing similarities in their background.

9.3.2 6DoF Pose Alignment

During test time, the algorithm first extracts key points on the source objects, as well as on the full scene. Next, correspondences are found by doing a k-nearest neighbor search in the feature space. The best matches are included and fed into RANSAC [76] to filter outliers. The remaining correspondences are then processed within a 6DoF pose optimization *e.g.* Singular Value Decomposition (SVD) [91] which tries to find an optimal rotation $\mathbf{R}$ and translation $\mathbf{t}$ given the set of correspondences on the object $\mathcal{K} = \{\mathbf{k}_i\}_{i=1}^{\mathrm{N_K}}$ and the scene $\mathcal{K}' = \{\mathbf{k}'_i\}_{i=1}^{\mathrm{N_K}}$ such that

$$(\mathbf{R}, \mathbf{t}) = \underset{\mathbf{R} \in SO(d), t \in \mathbb{R}^3}{\arg\min} \sum_{i=1}^{\mathrm{N_K}} ||(\mathbf{R} \cdot \mathbf{k}_i + \mathbf{t}) - \mathbf{k}'_i||_2^2. \tag{9.4}$$

The output of eq. 9.4 obtains a rigid transformation optimally aligning the object with the scene.

9.4 Evaluation

The task introduced in this work aims to estimate corresponding 6DoF transformations that transform a set of known objects in a segmented scene to a target scan of the same environment, see Fig. 9.1 at the beginning of this section. This requires computing transformations $\{\mathbf{T}_i\}_{i=1}^{m}$, where

$$\mathbf{T} \in \mathbb{R}^{4\times4} = \begin{bmatrix} \mathbf{R}_i & \mathbf{t}_i \\ \mathbf{0}^\top & 1 \end{bmatrix}, \tag{9.5}$$

consist of a translation $\{\mathbf{t}_i\}_{i=1}^{m}$ and rotation component $\{\mathbf{R}_i\}_{i=1}^{m}$. The following section first evaluates the correspondence matching accuracy of our approach (see Sec. 9.4.1) and then evaluates the object instance re-localization (see Sec. 9.4.2).

All experiments are conducted on the *RIO* dataset following the suggested train and test/val splits, see Sec. 3.2. A transformation $\mathbf{T}$ is considered correct in Sec. 9.4.2, if its absolute pose errors of the translation $\mathbf{t}$ (see eq. 9.6) and rotational component $\mathbf{R}$ (see eq. 9.7) defined as

$$\delta_t(\hat{\mathbf{t}}, \mathbf{t}) = ||\hat{\mathbf{t}} - \mathbf{t}||_2, \tag{9.6}$$

$$\delta_\vartheta(\hat{\mathbf{R}}, \hat{\mathbf{R}}) = ||\frac{180}{\pi} \cdot 2 \cdot \arccos[q(\mathbf{R})^{-1} \cdot q(\hat{\mathbf{R}})]||. \tag{9.7}$$

are within a small threshold given ground truth pose annotations. The thresholds used in the following are $(\epsilon_t, \epsilon_\vartheta) = (10\,\mathrm{cm}, 10°)$ and $(20\,\mathrm{cm}, 20°)$ respectively. Since multiple correct alignments exist for symmetric objects such as round or rectangular tables, a symmetry aware pose evaluation is proposed where the error is computed for all correct transformations using ambiguity transformations.

Tbl. 9.1. Matching dynamic TSDF patches in *RIO* at 95% recall. © 2019 IEEE.

	Method (trained)	F1	Accuracy	Precision	FPR	ER	Top-1	Top-3	Top-5
🟪	60cm (static)	71.54	62.21	57.37	70.60	75.59	2.17	4.12	5.96
🟨	120cm (static)	74.17	66.92	60.83	61.18	66.16	3.94	4.58	8.21
🟧	120cm (dynamic)	78.71	74.29	67.17	46.43	51.41	6.26	7.26	9.58
🟩	MS (static)	85.58	83.98	77.82	27.09	32.04	30.73	53.48	69.61
🟩	MS (dynamic)	**94.37**	**94.33**	**93.61**	**6.50**	**11.35**	**64.10**	**86.20**	**93.40**

9.4.1 Correspondence Matching

Keypoint-based pose estimation undoubtedly requires robust correspondence matching to obtain accurate pose estimates. Therefore, we first analyze how well different architectures compare in matching accuracy, see Tbl. 9.1. It reports the accuracy, F1 score, precision, false-positive rate (FPR), and the error rate (ER) at 95% recall. The corresponding PRC graphs can be found in Fig. 9.3. The aforementioned pre-trained network is named *MS (static)* while the fine-tuned counter-part lists under the name *MS (dynamic)*. Our proposed multi-scale method performs best, even when trained with static data and its performance increases even more when fine-tuned on the dynamic patches, see *MS (dynamic)*.

Instead of only listing the top-1 matching accuracy, we also report the cases where the positive sample was ranked within the top-5 or top-3 given 50 random samples. We believe this is a great way to test how this would generalize in real-world setups when the positive sample has to be matched against a pool of negatives.

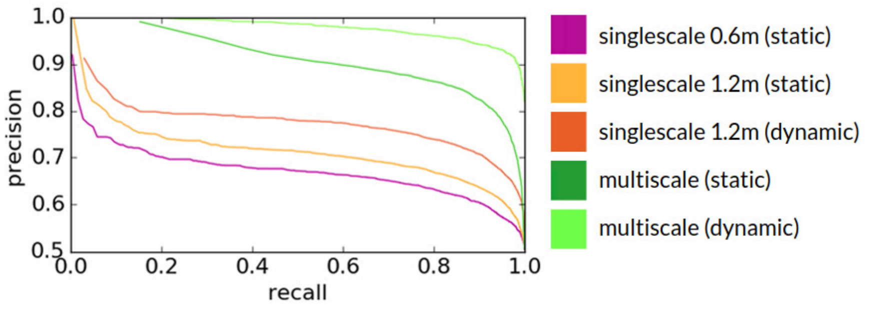


Fig. 9.3. Precision-Recall Curves (PRC) for matching dynamic patches, see Tbl. 9.1. © 2019 IEEE.

9.4.2 Object Instance Re-Localization

In the following, the object instance re-localization performance of the proposed networks compared to baseline approaches, namely FPFH [232], SHOT [279] as well as 3DMatch [324] is analyzed. Tbl. 9.2 shows our performance on the full dataset while Tbl. 9.3 gives insides about the accuracy on different classes. The reported performance measures are the recall, the fraction of samples within the given error thresholds.

Tbl. 9.2. Object instance re-localization evaluation on *RIO*-dataset [295]. © 2019 IEEE.

Method (trained)	<0.1m, $10°$	MRE [deg]	MTE [m]	<0.2m, $20°$	MRE [deg]	MTE [m]
FPFH [232]	2.61	7.25	0.0645	8.36	10.57	0.0776
SHOT [279]	6.79	5.35	0.0268	12.27	8.18	0.0393
3DMatch (dynamic) [324]	5.48	5.81	0.0542	13.05	7.30	0.0708
RIO (static)	9.92	4.33	0.0425	17.75	6.39	0.0545
RIO (dynamic)	**15.14**	4.75	0.0437	**23.76**	6.08	0.0547

Tbl. 9.3. Object instance re-localization evaluation on different semantic classes of *RIO* [295]. © 2019 IEEE.

Cateogry	FPFH [232]	SHOT [279]	3DMatch [324]	RIO-S [295]	RIO-D [295]
(1) Seating	5.08	12.71	6.78	14.41	21.19
(2) Tables	9.33	5.33	21.33	25.33	29.33
(3) Items	5.06	13.92	7.59	11.39	16.46
(4) Beds or Sofas	56.52	21.74	34.78	34.78	47.83
(5) Cushions	0.00	15.52	8.62	8.62	10.34
(6) Appliances	11.11	16.67	33.33	44.44	55.56
(7) Structures	0.00	0.00	8.33	16.67	33.33
Average	12.44	12.27	17.25	22.23	**30.58**

For evaluation purposes, in Tbl. 9.3, the changed object categories are mapped to 9 different classes, specifically (1) seating such as different chairs, stools, and benches, (2) tables, cabinets, commodes, and shelves, (3) beds and sofas and other upholstery, (4) appliances or similar sanitary equipment, (5) cushions including pillows, bean bags, and ottomans, (6) small and portable items such as boxes and (7) scene structure including windows and doors. We compare our method against classical point cloud descriptors available in PCL [234] such as SHOT [279] and FPFH [7, 232] as well as the learned feature descriptor, 3DMatch [324]. 3DMatch was trained with a contrastive loss using a volumetric volume of $30 \times 30 \times 30$, of $1\,\mathrm{cm}$ voxels.

The baseline methods are processed similarly by first extracting 3D key points on the source object and the target scene, followed by the extraction of the respective feature vector for each point, and finally matching the produced vectors to obtain correspondences which are further processed with RANSAC [76].

Notably, our method has the best performance with a large margin across all object categories, even when trained with static data but especially when utilizing dynamic training patches. Finally, we report some qualitative results of our alignment in Fig. 9.4 and 9.5.

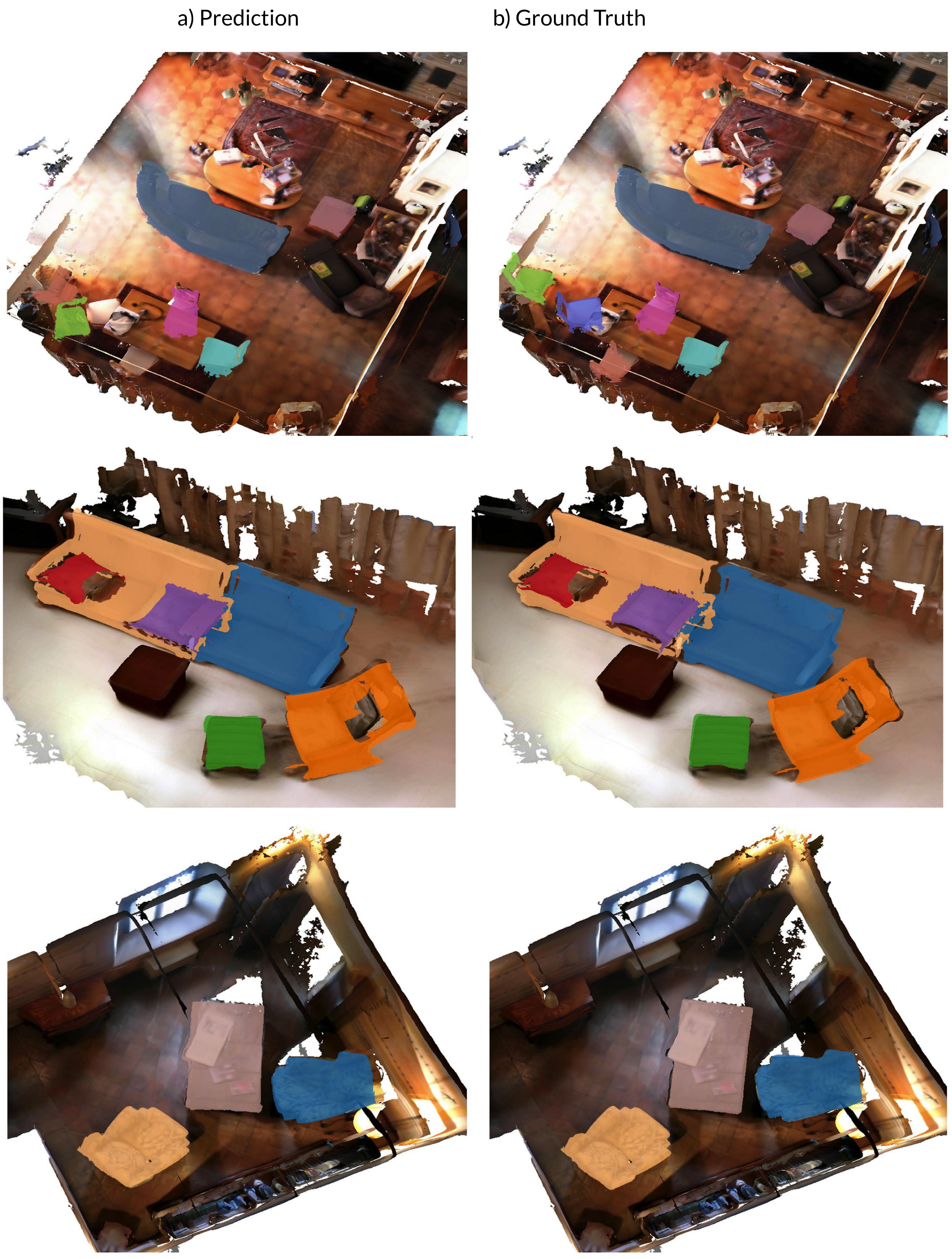


Fig. 9.4. Qualitative object instance re-localization results using learned correspondences on *RIO*. © 2019 IEEE.

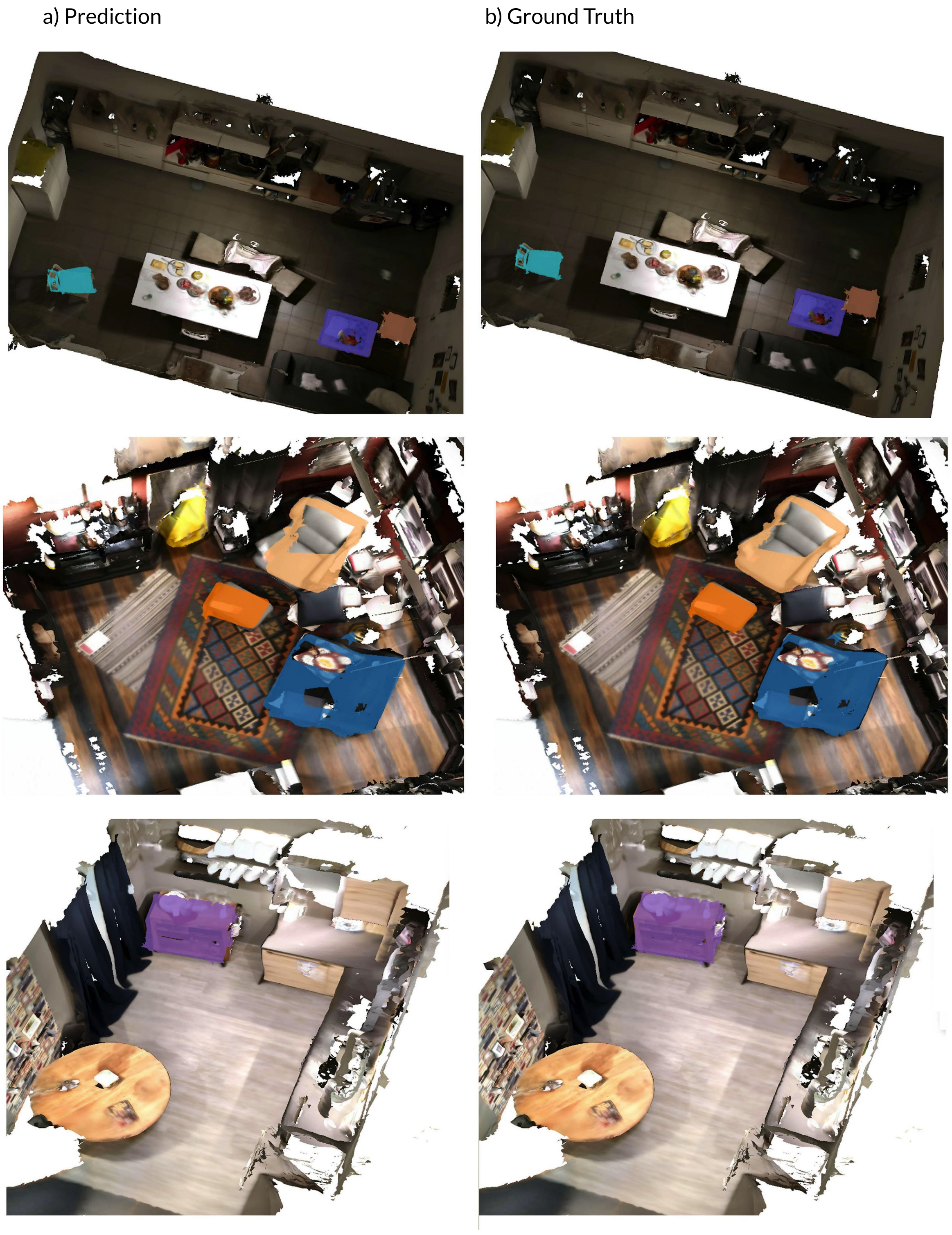


Fig. 9.5. Qualitative object instance re-localization results using learned correspondences on *RIO*. © 2019 IEEE.

9.5 Conclusion

The previous section has introduced a new method for object instance re-localization on the proposed *RIO* dataset, see Sec. 3.2. The most challenging part of determining the 6DoF pose of a changed object is finding stable, change-invariant correspondences across re-scans. Varied scanning patterns and changing geometry and context make this task challenging but utilizing a large-scale dataset and a network to learn these encodings seems to be a promising direction.

Re-localization of objects is an exciting research direction that might profit from being part of higher-level tasks *e.g.* persistent SLAM since it enables updating a potentially outdated map. Therefore, it might be beneficial *e.g.* when estimating the camera's pose in a tracking or re-localization framework.

While this presented method assumes a full 3D scan of both reference and re-scan, it is often more realistic in *e.g.* robotic application to only assume a single image or a short sequence of the re-visited environment. However, the so-called 2D-3D or 2D-2D object instance re-localization is quite challenging since it requires detecting change-invariant features on 2D images and associating them with features stored in a 3D map and was explored in two student projects. In one particular work [2] objects are detected and re-identified using a 2D network, similarly to re-OBJ [23]. Contrary to re-OBJ [23], 2D object-level encodings are learned in changing setups combining higher level and lower level features instead of relying on the background as done in [23], which is unreliable in dynamically changing environments. Such an approach could be extended within an online framework that jointly detects and re-associates objects in a sequence of images to ultimately update the 3D reference map in real-time.

[2] https://github.com/lukasHoel/3rscan-triplet-dataset-toolkit

Part V

Conclusion and Outlook

Conclusion

10

In the following, we shortly summarize the content of this thesis (see Sec. 10.1), then highlight some of the limitations of the presented works and discuss possible directions of future research (see Sec. 10.2).

10.1 Summary

While most state-of-the-art vision systems assume static setups, indoor environments are highly dynamic. Indoor scenes naturally change since they are designed for interaction where objects of various sizes are relocated, added, and removed. To perform dynamic 3D scene understanding requires a rich high-level knowledge from the estimation of geometric and semantic properties of the objects as well as their intrinsic relations.

In this thesis, we have contributed several datasets and benchmarks and novel methods and applications for 3D scene understanding beyond static setups, incorporating scene dynamics and changes.

A major contribution discussed in Sec. 3.2 is *RIO*, the first large-scale real world RGB-D indoor dataset that features re-scans of naturally changing indoor environments. Chapter 4 introduces 3D semantic scene graphs as a representation for scene understanding and in Sec. 4.2 describes *3DSSG* [296] a scene graph dataset based on the 3D data of *RIO* [295].

Part III discussed the first methodological contributions and focuses on the extraction of scene semantics. Efficient methods for 3D semantic segmentation have been described [222, 299] including, *FISS*, Fully Incremental Semantic Segmentation, an online method that runs in real-time on embedded hardware [299] and *FCPN*, a fully-convolutional point network for large-scale 3D semantic segmentation. Then, in Sec. 6.3, 6.4 and 6.6, various methods for 3D scene graph prediction were covered [296, 297, 308] which is a well-known 2D task transferred into the 3D domain. Specifically, the first learned 3D scene graph prediction method is proposed based on a segmented point cloud of the scene. Later, this method is extended by learning the segmentation automatically and jointly estimating instances, and predicting relations.

The second part of the methodological contributions focused on scene changes and consists of three chapters that describe the re-localization of scenes (see Chapter 7), cameras (see Chapter 8) as well as objects (see Chapter 9). Scene re-localization is addressed in Sec. 7.2 where scene graphs are applied to retrieve a target scene given data in different domains [296]. Then, *RIO10*, a benchmark of changing indoor environments, is used to evaluate camera re-localization based on 10 scenes of *RIO*. The conducted experiments show that state-of-the-art camera re-localization methods are highly affected by scene changes, and further research is required to operate in changing setups robustly [298]. To analyze the effect of changes on re-localization performance, novel ways to measure scene change in an image frame are introduced, including a measure for geometric, semantic, and visual change. In the final chapter 9, we describe the task of re-localization of objects in those changing setups [295] where a 6DoF transformation is computed based on learned features that find correspondences between rigidly moved objects given a reference scene and its re-scan, taken at a different point in time.

In summary, this thesis explored scene understanding from static to dynamic indoor environments and played its part in closing the gap between theoretical static setups and real-world, dynamic environments. We showed the importance of sophisticated semantics and described several re-localization algorithms to enforce persistence across time. While this dissertation ultimately only scratched the surface of long-term scene understanding, we believe our contributions brought us a tiny bit closer to operate in real-world changing 3D indoor scenes autonomously.

10.2 Limitations and Future Work

This thesis explored a wide range of interdependent scene understanding tasks and applications. Some of the methods assume prior knowledge about the scene, so does *e.g.* the scene graph prediction method in Sec. 6.3 require a scene segmentation, similarly to the object instance re-localization described in chapter 9. While it is reasonable to expect some prior information – especially given the complexity of some of the tackled tasks – it prevents the proposed algorithms from being applicable to real-world setups since the semantic knowledge is often unavailable or limited in resolution and accuracy.

Furthermore, we have shown that camera re-localization on images of rigidly moving objects potentially causes misalignments where the camera is localized with respect to the object in view rather than the global scene. This problem could be reduced by updating the reference scene as new observations arrive. Ideally, this updating procedure could be done with a single 2D image of the re-scan, but this requires not only to associate objects in changing context but also to handle non-rigid changes and variation in illumination – all of which are quite challenging by themselves. Many, if not all, of the tasks covered in this thesis, could ideally be solved jointly as they might benefit from one other.

This leads us to *persistent SLAM*, a computational problem that was continuously brought up in several research discussions in the last years. It is the real-time SLAM counterpart of long-term/dynamic 3D scene understanding where a 3D map of a known space is constructed and updated while the camera is simultaneously localized. Since the reference map is an earlier observation of the scene, other challenges arise that do not exist in the classical SLAM setup. We believe collecting and storing semantic instance knowledge is an essential process in mapping a dynamic environment. Since scene graphs are a promising representation for 3D scene understanding, as they are compact and rich, they might be well suited for other high-level tasks and ultimately be an excellent representation for *persistent SLAM*.

The scene graphs discussed in our dissertation are, by purpose, quite general and ambiguous, which potentially is not descriptive enough for some tasks *e.g*. scene synthesis due to *e.g*. the lack of some geometric properties of the objects. Therefore, augmenting the 3D scene graph with canonical 6DoF object poses and associating nodes across scans is a research direction worth exploring. A localization algorithm would then need to jointly detect changes and estimate the camera pose given this dynamic setup. This long-term knowledge could be stored as temporal connections in a graph structure. Along with comprehensive semantics beyond binary class labels, 3D temporal scene graphs with 6DoF poses could provide the representation needed for robust and efficient *persistent SLAM*.

Finally, it is worth mentioning that most of the work described in this thesis heavily relies on a vast amount of annotations. Even though some of the main contributions of the thesis are large-scale data collections, this type of datasets are often not available. While it is crucial to have annotated data, especially for evaluation purposes, we believe unsupervised or semi-supervised learning is an auspicious research direction. As humans segment foreground objects from the background via motion, a method could similarly also segment object instances automatically by associating the different regions of multiple scene observations. This requires a rich encoding of the environment's geometry to robustly identify the connected components, which undoubtedly is a pretty challenging mission.

Ultimately, finding an unsupervised and generic solution for *persistent SLAM* would solve many of the challenges long-term/dynamic 3D scene understanding faces and could bring many of the theoretical models into practice.

Part VI

Appendix

List of Authored and Co-authored Publications

A

2021

[207] Evin Pinar Örnek, Shristi Mudgal, **Johanna Wald**, Yida Wang and Federico Tombari. "From 2D to 3D: Re-thinking Benchmarking of Monocular Depth Prediction". *under submission*.

[297] **Johanna Wald**, Nassir Navab and Federico Tombari. "Learning 3D Semantic Scene Graphs with Instance Embeddings". *Springer International Journal of Computer Vision (IJCV), 2022*.

[308] Shun-Cheng Wu, **Johanna Wald**, Keisuke Tateno, Nassir Navab and Federico Tombari. "SceneGraphFusion: Incremental 3D Scene Graph Prediction from RGB-D Sequences". *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2021*.

2020

[298] **Johanna Wald**, Torsten Sattler, Stuart Golodetz, Tommaso Cavallari and Federico Tombari. "Beyond Controlled Environments: 3D Camera Re-Localization in Changing Indoor Scenes". *Springer European Conference on Computer Vision (ECCV), 2020*.

[296] **Johanna Wald***, Helisa Dhamo*, Nassir Navab, and Federico Tombari. "Learning 3D Semantic Scene Graphs from 3D Indoor Reconstructions". *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020*.

2019

[295] **Johanna Wald**, Armen Avetisyan, Nassir Navab, Tombari Federico* and Matthias Nießner*. "RIO: 3D Object Instance Re-Localization in Changing Indoor Environments". Oral, *IEEE International Conference on Computer Vision (ICCV), 2019*.

2018

[222] Dario Rethage, **Johanna Wald**, Jürgen Sturm, Nassir Navab and Federico Tombari. "Fully-Convolutional Point Networks for Large-Scale Point Clouds". *Springer European Conference on Computer Vision (ECCV), 2018*.

[299] **Johanna Wald**, Keisuke Tateno, Jürgen Sturm, Nassir Navab and Federico Tombari. "Real-Time Fully Incremental Scene Understanding on Mobile Platforms". In *Robotics and Automation Letters (RAL)*, presented at *IEEE International Conference on Intelligent Robots and Systems (IROS), 2018*.

Bibliography

[1] H. Abdul-Rashid, J. Yuan, B. Li, Y. Lu, S. Bai, X. Bai, N.-M. Bui, M. N. Do, T.-L. Do, A.-D. Duong, X. He, T.-K. Le, W. Li, A. Liu, X. Liu, K.-T. Nguyen, V.-T. Nguyen, W. Nie, V.-T. Ninh, Y. Su, V. Ton-That, M.-T. Tran, S. Xiang, H. Zhou, Y. Zhou, and Z. Zhou. "2D Image-based 3D Scene Retrieval". In: *Eurographics Workshop on 3D Object Retrieval*. 2018 (cit. on p. 105).

[2] H. Abdul-Rashid, J. Yuan, B. Li, Y. Lu, T. Schreck, N.-M. Bui, T.-L. Do, M. Holenderski, D. Jarnikov, K. T. Le, V. Menkovski, K.-T. Nguyen, T.-A. Nguyen, V.-T. Nguyen, T. V. Ninh, P. Rey, M.-T. Tran, and T. Wang. "Extended 2D Scene Image-Based 3D Scene Retrieval". In: *Eurographics Workshop on 3D Object Retrieval*. 2019 (cit. on p. 105).

[3] D. Acharya, K. Khoshelham, and S. Winter. "BIM-PoseNet: Indoor camera localisation using a 3D indoor model and deep learning from synthetic images". In: *Journal of Photogrammetry and Remote Sensing (P&RS)* 150 (2019), pp. 245–258 (cit. on p. 113).

[4] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. "Building Rome in a Day". In: *Commun. ACM* 54.10 (2011), pp. 105–112 (cit. on p. 119).

[5] A. Agudo, J. Montiel, L. Agapito, and B. Calvo. "Modal Space: A Physics-Based Model for Sequential Estimation of Time-Varying Shape from Monocular Video". In: *Journal of Mathematical Imaging and Vision* (2016), pp. 75–98 (cit. on p. 129).

[6] A. Aldoma, F. Tombari, R. B. Rusu, and M. Vincze. "OUR-CVFH - Oriented, Unique and Repeatable Clustered Viewpoint Feature Histogram for Object Recognition and 6DOF Pose Estimation". In: *Pattern Recognition Proceedings*. 2012 (cit. on pp. 59, 73–75).

[7] A. Aldoma, M. Vincze, N. Blodow, D. Gossow, S. Gedikli, R. B. Rusu, and G. R. Bradski. "CAD-model recognition and 6DOF pose estimation using 3D cues". In: *International Conference on Computer Vision (ICCV)*. 2011 (cit. on pp. 59, 73–75, 135).

[8] L. Alexandre. "3D Descriptors for Object and Category Recognition: a Comparative Evaluation". In: *International Conference on Intelligent Robots and Systems*. Vol. Workshop on Color-Depth Camera Fusion in Robotics. 2012, pp. 1–6 (cit. on p. 74).

[9] A. Anoosheh, T. Sattler, R. Timofte, M. Pollefeys, and L. V. Gool. "Night-to-Day Image Translation for Retrieval-based Localization". In: *International Conference on Robotics and Automation*. 2019 (cit. on pp. 19, 20, 105).

[10] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. "NetVLAD: CNN architecture for weakly supervised place recognition". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2016 (cit. on pp. 105, 112, 120, 122).

[11] S. Ardeshir, A. R. Zamir, A. Torroella, and M. Shah. "GIS-Assisted Object Detection and Geospatial Localization". In: *European Conference on Computer Vision (ECCV)*. 2014 (cit. on p. 114).

[12] I. Armeni, Z.-Y. He, J. Gwak, A. R. Zamir, M. Fischer, J. Malik, and S. Savarese. "3D Scene Graph: A Structure for Unified Semantics, 3D Space, and Camera". In: *International Conference on Computer Vision (ICCV)*. 2019 (cit. on pp. 42, 43, 47, 86).

[13] I. Armeni*, A. Sax*, A. R. Zamir, and S. Savarese. “Joint 2D-3D-Semantic Data for Indoor Scene Understanding”. In: *arXiv:1702.01105* (2017) (cit. on p. 20).

[14] D. Arthur and S. Vassilvitskii. “k-means++: The Advantages of Careful Seeding”. In: *SODA ’07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035 (cit. on p. 93).

[15] O. Ashual and L. Wolf. “Specifying Object Attributes and Relations in Interactive Scene Generation”. In: *International Conference on Computer Vision (ICCV)* (2019) (cit. on p. 83).

[16] N. Atanasov, M. Zhu, K. Daniilidis, and G. J. Pappas. “Localization from Semantic Observations via the Matrix Permanent”. In: *International Journal of Robotics Research* (2016) (cit. on p. 114).

[17] A. Avetisyan, M. Dahnert, A. Dai, M. Savva, A. X. Chang, and M. Nießner. “Scan2CAD: Learning CAD Model Alignment in RGB-D Scans”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2019 (cit. on pp. 29, 32, 105).

[18] A. Avetisyan, T. Khanova, C. Choy, D. Dash, A. Dai, and M. Nießner. “SceneCAD: Predicting Object Alignments and Layouts in RGB-D Scans”. In: *European Conference on Computer Vision (ECCV)*. 2020 (cit. on pp. 42, 86, 105).

[19] H. Badino, D. Huber, and T. Kanade. “Visual Topometric Localization”. In: *Intelligent Vehicles Symposium (IV)*. 2011 (cit. on pp. 19, 20, 112).

[20] H. Bae, M. Walker, J. White, Y. Pan, Y. Sun, and M. Golparvar-Fard. “Fast and Scalable Structure-from-Motion for High-precision Mobile Augmented Reality Systems”. In: *The Journal of Mobile User Experience* 5.1 (2016), pp. 1–21 (cit. on p. 7).

[21] V. Balntas, D. Frost, R. Kouskouridas, A. Barroso-Laguna, A. Talattof, H. Heijnen, and K. Mikolajczyk. *(SILDa): Scape Imperial Localisation Dataset*. https://image-matching-workshop.github.io/challenge/. 2019 (cit. on pp. 19, 20).

[22] V. Balntas, S. Li, and V. Prisacariu. “RelocNet: Continuous Metric Learning Relocalisation using Neural Nets”. In: *European Conference on Computer Vision (ECCV)*. 2018 (cit. on p. 112).

[23] V. Bansal, S. James, and A. D. Bue. “re-OBJ: Jointly Learning the Foreground and Background for Object Instance Re-identification”. In: *International Conference Image Analysis and Processing (ICIAP)*. Springer, 2019 (cit. on p. 138).

[24] A. Benbihi, M. Geist, and C. Pradalier. “ELF: Embedded Localisation of Features in Pre-Trained CNN”. In: *International Conference on Computer Vision (ICCV)*. 2019 (cit. on p. 19).

[25] I. Biederman, R. J. Mezzanotte, Jan, and C. Rabinowitz. *Scene Perception: Detection and Judging Objects Undergoing Relational Violations*. 1982 (cit. on pp. 4–6).

[26] M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, and A. Davison. “CodeSLAM—learning a compact, optimisable representation for dense visual SLAM”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2560–2568 (cit. on p. 129).

[27] E. Brachmann, A. Krull, S. Nowozin, J. Shotton, F. Michel, S. Gumhold, and C. Rother. “DSAC – Differentiable RANSAC for Camera Localization”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2017 (cit. on p. 114).

[28] E. Brachmann, F. Michel, A. Krull, M. Y. Yang, S. Gumhold, and C. Rother. “Uncertainty-Driven 6D Pose Estimation of Objects and Scenes from a Single RGB Image”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2016 (cit. on p. 114).

[29] E. Brachmann and C. Rother. “Expert Sample Consensus Applied to Camera Re-Localization”. In: *International Conference on Computer Vision (ICCV)*. IEEE, 2019 (cit. on p. 114).

[30] E. Brachmann and C. Rother. “Learning Less is More – 6D Camera Localization via 3D Surface Regression”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2018 (cit. on p. 114).

[31] E. Brachmann and C. Rother. “Neural-Guided RANSAC: Learning Where to Sample Model Hypotheses”. In: *International Conference on Computer Vision (ICCV)*. IEEE, 2019 (cit. on p. 114).

[32] E. Brachmann and C. Rother. “Visual Camera Re-Localization from RGB and RGB-D Images Using DSAC”. In: *arXiv:2002.12324* (2020) (cit. on p. 114).

[33] S. Brahmbhatt, J. Gu, K. Kim, J. Hays, and J. Kautz. “Geometry-Aware Learning of Maps for Camera Localization”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2018 (cit. on p. 113).

[34] L. Breiman. “Random Forests”. In: *Mach. Learn.* 45.1 (2001), pp. 5–32 (cit. on p. 61).

[35] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser. “Simultaneous Localization And Mapping: A Survey of Current Trends in Autonomous Driving”. In: *Transactions on Intelligent Vehicles* 2.3 (2017), pp. 194–220 (cit. on p. 7).

[36] J. Bruna, W. Zaremba, A. Szlam, and Y. Lecun. “Spectral networks and locally connected networks on graphs”. In: *International Conference on Learning Representations*. 2014 (cit. on pp. 69, 70).

[37] M. Bui, C. Baur, N. Navab, S. Ilic, and S. Albarqouni. “Adversarial Networks for Camera Pose Regression and Refinement”. In: *International Conference on Computer Vision Workshops* (2019) (cit. on p. 113).

[38] Z. Cai, J. Han, L. Liu, and L. Shao. “RGB-D datasets using Microsoft Kinect or Similar Sensors: a Survey”. In: *Multimedia Tools and Applications* 76 (2016), pp. 4313–4355 (cit. on p. 18).

[39] N. Carlevaris-Bianco, A. K. Ushani, and R. M. Eustice. “The University of Michigan North Campus Long-Term Vision and LIDAR Dataset”. In: *International Journal of Robotics Research* 35.9 (2016), pp. 1023–1035 (cit. on pp. 20, 21).

[40] R. Castle, G. Klein, and D. W. Murray. “Video-rate Localization in Multiple Maps for Wearable Augmented Reality”. In: *International Symposium on Wearable Computers*. IEEE, 2008, pp. 15–22 (cit. on p. 7).

[41] T. Cavallari*, L. Bertinetto, J. Mukhoti, P. Torr, and S. Golodetz*. “Let’s Take This Online: Adapting Scene Coordinate Regression Network Predictions for Online RGB-D Camera Relocalisation”. In: *International Conference on 3D Vision*. 2019 (cit. on p. 114).

[42] T. Cavallari, S. Golodetz, N. A. Lord, J. P. C. Valentin, L. di Stefano, and P. H. S. Torr. “On-the-Fly Adaptation of Regression Forests for Online Camera Relocalisation”. In: *IEEE Conference on Computer Vision and Pattern Recognition* (2017) (cit. on pp. 68, 114, 119, 120, 122).

[43] T. Cavallari*, S. Golodetz*, N. A. Lord*, J. Valentin*, V. A. Prisacariu, L. D. Stefano, and P. H. S. Torr. “Real-Time RGB-D Camera Pose Estimation in Novel Scenes using a Relocalisation Cascade”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019) (cit. on pp. 114, 120, 122).

[44] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Nießner, M. Savva, S. Song, A. Zeng, and Y. Zhang. “Matterport3D: Learning from RGB-D Data in Indoor Environments”. In: *International Conference on 3D Vision*. 2017 (cit. on pp. 19, 20).

[45] D. M. Chen, G. Baatz, K. Köser, S. S. Tsai, R. Vedantham, T. Pylvänäinen, K. Roimela, X. Chen, J. Bach, M. Pollefeys, B. Girod, and R. Grzeszczuk. “City-Scale Landmark Identification on Mobile Devices”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2011, pp. 737–744 (cit. on pp. 19, 20).

[46] W. Choi, Y. W. Chao, C. Pantofaru, and S. Savarese. “Understanding Indoor Scenes Using 3D Geometric Phrases”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2013 (cit. on p. 86).

[47] C. Choy, J. Gwak, and S. Savarese. “4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2019 (cit. on p. 61).

[48] Ö. Çiçek, A. Abdulkadir, S. Lienkamp, T. Brox, and O. Ronneberger. “3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. 2016 (cit. on pp. 61, 77, 92).

[49] R. Clark, S. Wang, A. Markham, N. Trigoni, and H. Wen. “VidLoc: A Deep Spatio-Temporal Model for 6-DoF Video-Clip Relocalization”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6856–6864 (cit. on p. 113).

[50] L. Cosmo, A. Kazi, S. Ahmadi, N. Navab, and M. M. Bronstein. “Latent Patient Network Learning for Automatic Diagnosis”. In: *CoRR* abs/2003.13620 (2020) (cit. on p. 84).

[51] J. M. Coughlan and A. L. Yuille. “Manhattan World: compass direction from a single image by Bayesian inference”. In: *International Conference on Computer Vision (ICCV)*. 1999 (cit. on p. 5).

[52] B. Curless and M. Levoy. “A Volumetric Method for Building Complex Models from Range Images”. In: *SIGGRAPH*. ACM, 1996, pp. 303–312 (cit. on p. 24).

[53] M. Dahnert, A. Dai, L. Guibas, and M. Nießner. “Joint Embedding of 3D Scan and CAD Objects”. In: *International Conference on Computer Vision (ICCV)*. 2019 (cit. on p. 105).

[54] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. “ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2017 (cit. on pp. 18–20, 27, 42, 43, 59–61, 63, 67–70, 73, 76, 77).

[55] A. Dai and M. Nießner. “3DMV: Joint 3D-Multi-View Prediction for 3D Semantic Scene Segmentation”. In: *European Conference on Computer Vision (ECCV)*. 2018 (cit. on pp. 18, 61, 82, 85, 129).

[56] A. Dai, M. Nießner, M. Zollöfer, S. Izadi, and C. Theobalt. “BundleFusion: Real-time Globally Consistent 3D Reconstruction using On-the-fly Surface Re-integration”. In: *ACM Transactions on Graphics (TOG)* (2017) (cit. on pp. 8, 18, 68, 71, 72).

[57] A. Dai, D. Ritchie, M. Bokeloh, S. Reed, J. Sturm, and M. Nießner. “ScanComplete: Large-Scale Scene Completion and Semantic Segmentation for 3D Scans”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2018 (cit. on p. 70).

[58] A. J. Davison. “FutureMapping: The Computational Structure of Spatial AI Systems”. In: *CoRR* abs/1803.11288 (2018). arXiv: 1803.11288 (cit. on p. 7).

[59] A. J. Davison, I. D. Reid, N. Molton, and O. Stasse. “MonoSLAM: Real-Time Single Camera SLAM”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2007) (cit. on p. 59).

[60] H. Deng, T. Birdal, and S. Ilic. “PPF-FoldNet: Unsupervised Learning of Rotation Invariant 3D Local Descriptors”. In: *European Conference on Computer Vision (ECCV)*. 2018 (cit. on pp. 128, 130).

[61] L. Deng, Z. Chen, B. Chen, Y. Duan, and J. Zhou. “Incremental image set querying based localization”. In: *Neurocomputing* (2016) (cit. on pp. 105, 114).

[62] D. DeTone, T. Malisiewicz, and A. Rabinovich. “SuperPoint: Self-Supervised Interest Point Detection and Description”. In: *IEEE Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, 2018 (cit. on p. 114).

[63] H. Dhamo, A. Farshad, I. Laina, N. Navab, G. D. Hager, F. Tombari, and C. Rupprecht. “Semantic Image Manipulation Using Scene Graphs”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2020 (cit. on p. 83).

[64] N.-D. Duong, A. Kacete, C. Sodalie, P.-Y. Richard, and J. Royan. “xyzNet: Towards Machine Learning Camera Relocalization by Using a Scene Coordinate Prediction Network”. In: *International Symposium on Mixed and Augmented Reality*. IEEE, 2018 (cit. on p. 114).

[65] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler. “D2-Net: A Trainable CNN for Joint Detection and Description of Local Features”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2019 (cit. on pp. 19, 113, 119, 120, 122).

[66] M. Dzitsiuk, J. Sturm, R. Maier, L. Ma, and D. Cremers. “De-noising, Stabilizing and Completing 3D Reconstructions On-the-go using Plane Priors”. In: *International Conference on Robotics and Automation*. 2017 (cit. on p. 24).

[67] D. Eigen, C. Puhrsch, and R. Fergus. “Depth Map Prediction from a Single Image Using a Multi-scale Deep Network”. In: *International Conference on Neural Information Processing Systems*. 2014 (cit. on p. 18).

[68] F. Engelmann, M. Bokeloh, A. Fathi, B. Leibe, and M. Nießner. “3D-MPA: Multi Proposal Aggregation for 3D Semantic Instance Segmentation”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2020 (cit. on pp. 18, 60, 85).

[69] F. Engelmann, T. Kontogianni, A. Hermans, and B. Leibe. “Exploring Spatial Context for 3D Semantic Segmentation of Point Clouds”. In: *International Conference on Computer Vision Workshops*. 2017 (cit. on p. 60).

[70] R. Epstein. “Neural Systems for Visual Scene Recognition”. In: 2014, pp. 105–134 (cit. on p. 5).

[71] G. Fanelli, M. Dantone, J. Gall, A. Fossati, and L. van Gool. “Random Forests for Real Time 3D Face Analysis”. In: *International Journal Computer Vision* 101 (2013), pp. 437–458 (cit. on p. 18).

[72] M. Fehr, F. Furrer, D. Ivan, J. Sturm, I. Gilitschenski, R. Siegwart, and C. Cadena. “TSDF-based Change Detection for Consistent Long-Term Dense Reconstruction and Dynamic Object Discovery”. In: *International Conference on Robotics and Automation*. 2017 (cit. on pp. 19, 130).

[73] C. Fellbaum, ed. *WordNet: an electronic lexical database*. MIT Press, 1998 (cit. on pp. 43, 44).

[74] P. F. Felzenszwalb and D. P. Huttenlocher. “Efficient Graph-Based Image Segmentation”. In: *International Journal Computer Vision* 59 (2004), pp. 167–181 (cit. on p. 27).

[75] M. Firman. “RGBD Datasets: Past, Present and Future”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2016 (cit. on p. 18).

[76] M. A. Fischler and R. C. Bolles. “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography”. In: *Commun. ACM* 24.6 (1981), pp. 381–395 (cit. on pp. 59, 113, 114, 123, 130, 133, 135).

[77] M. Fisher, M. Savva, and P. Hanrahan. “Characterizing structural relationships in scenes using graph kernels”. In: *ACM Transactions on Graphics (TOG)* (2011) (cit. on p. 86).

[78] M. M. Franco Manessi Alessandro Rozza. “Dynamic Graph Convolutional Networks”. In: *Elsevier Pattern Recognition* (2019) (cit. on pp. 76, 84).

[79] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao. “Deep Ordinal Regression Network for Monocular Depth Estimation”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2018 (cit. on p. 18).

[80] F. Furrer, T. Novkovic, M. Fehr, A. Gawel, M. Grinvald, T. Sattler, R. Siegwart, and J. I. Nieto. “Incremental Object Database: Building 3D Models from Multiple Partial Observations”. In: *International Conference on Intelligent Robots and Systems*. 2018 (cit. on p. 60).

[81] D. Gálvez-López and J. D. Tardós. “Real-Time Loop Detection with Bags of Binary Words”. In: *International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 51–58 (cit. on pp. 105, 112).

[82] R. Garg, A. Roussos, and L. Agapito. “Dense Variational Reconstruction of Non-rigid Surfaces from Monocular Video”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2013 (cit. on p. 129).

[83] P. Gay, J. Stuart, and A. Del Bue. “Visual Graphs from Motion (VGfM): Scene understanding with object geometry reasoning”. In: *Asian Conference on Computer Vision*. Springer. 2018, pp. 330–346 (cit. on pp. 42, 86).

[84] A. P. Gee and W. Mayol-Cuevas. “6D Relocalisation for RGBD Cameras Using Synthetic View Regression”. In: *British Machine Vision Conference*. BMVA, 2012, pp. 1–11 (cit. on p. 112).

[85] H. Germain, G. Bourmaud, and V. Lepetit. “Sparse-To-Dense Hypercolumn Matching for Long-Term Visual Localization”. In: *International Conference on 3D Vision*. 2019 (cit. on pp. 19, 113, 114).

[86] J. J. Gibson. *The Ecological Approach to Visual Perception*. Boston: Houghton Mifflin, 1979 (cit. on p. 47).

[87] B. Glocker, S. Izadi, J. Shotton, and A. Criminisi. “Real-time RGB-D camera relocalization”. In: *International Symposium on Mixed and Augmented Reality*. IEEE, 2013 (cit. on pp. 20, 35).

[88] B. Glocker, J. Shotton, A. Criminisi, and S. Izadi. “Real-Time RGB-D Camera Relocalization via Randomized Ferns for Keyframe Encoding”. In: *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 21.5 (2015) (cit. on pp. 105, 112).

[89] S. Golodetz*, T. Cavallari*, N. A. Lord*, V. A. Prisacariu, D. W. Murray, and P. H. S. Torr. “Collaborative Large-Scale Dense 3D Reconstruction with Online Inter-Agent Pose Optimisation”. In: *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 24.11 (2018), pp. 2895–2905 (cit. on pp. 20, 114).

[90] S. Golodetz*, M. Sapienza*, J. P. C. Valentin, V. Vineet, M.-M. Cheng, V. A. Prisacariu, O. Kähler, C. Y. Ren, A. Arnab, S. L. Hicks, D. W. Murray, S. Izadi, and P. H. S. Torr. “SemanticPaint: Interactive Segmentation and Learning of 3D Worlds”. In: *ACM SIGGRAPH Emerging Technologies*. 2015, p. 22 (cit. on p. 7).

[91] G. H. Golub and C. Reinsch. “Singular value decomposition and least squares solutions”. In: *Linear Algebra* (1971), pp. 134–151 (cit. on p. 133).

[92] M. A. Goodale and A. D. Milner. “Separate visual pathways for perception and action”. In: *Trends in Neurosciences* 15 (1992) (cit. on p. 6).

[93] *Google Research/tf3d*. https://github.com/google-research/google-research. 2021 (cit. on p. 92).

[94] B. Graham, M. Engelcke, and L. van der Maaten. “3D Semantic Segmentation with Submanifold Sparse Convolutional Networks”. In: *IEEE Conference on Computer Vision and Pattern Recognition* (2018) (cit. on pp. 61, 83, 91).

[95] M. Grewal and A. Andrews. “Kalman Filtering: Theory and Applications”. In: *IEEE Press* (1985) (cit. on p. 64).

[96] M. Grinvald, F. Furrer, T. Novkovic, J. J. Chung, C. Cadena, R. Siegwart, and J. Nieto. “Volumetric Instance-Aware Semantic Mapping and 3D Object Discovery”. In: *Robotics and Automation Letters* (2019), pp. 3037–3044 (cit. on p. 60).

[97] A. Guzman-Rivera, P. Kohli, B. Glocker, J. Shotton, T. Sharp, A. Fitzgibbon, and S. Izadi. “Multi-Output Learning for Camera Relocalization”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2014, pp. 1114–1121 (cit. on p. 114).

[98] M. Halber, Y. Shi, K. K. Xu, and T. Funkhouser. “Rescan: Inductive Instance Segmentation for Indoor RGBD Scans”. In: *International Conference on Computer Vision (ICCV)*. 2019 (cit. on p. 19).

[99] L. Han, T. Zheng, L. Xu, and L. Fang. “OccuSeg: Occupancy-Aware 3D Instance Segmentation”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2020 (cit. on pp. 18, 85, 92).

[100] A. Handa, T. Whelan, J. McDonald, and A. J. Davison. “A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM”. In: *International Conference on Robotics and Automation*. 2014 (cit. on p. 18).

[101] B. Hariharan, P. A. Arbeláez, R. B. Girshick, and J. Malik. “Hypercolumns for Object Segmentation and Fine-Grained Localization”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2015 (cit. on p. 114).

[102] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003 (cit. on p. 7).

[103] K. He, G. Gkioxari, P. Dollár, and R. Girshick. "Mask R-CNN". In: *International Conference on Computer Vision (ICCV)*. 2017 (cit. on pp. 61, 85, 129).

[104] K. He, Y. Lu, and S. Sclaroff. "Local Descriptors Optimized for Average Precision". In: *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2018, pp. 596–605 (cit. on p. 114).

[105] M. A. Hearst. "Support Vector Machines". In: *IEEE Intelligent Systems* 13.4 (1998), pp. 18–28 (cit. on p. 61).

[106] A. Hermans, G. Floros, and B. Leibe. "Dense 3D Semantic Mapping of Indoor Scenes from RGB-D Images". In: *International Conference on Robotics and Automation*. 2014 (cit. on pp. 69, 73).

[107] R. Herzig, M. Raboh, G. Chechik, J. Berant, and A. Globerson. "Mapping Images to Scene Graphs with Permutation-Invariant Structured Prediction". In: *International Conference on Neural Information Processing Systems*. 2018 (cit. on pp. 83, 84).

[108] J. Hou, A. Dai, and M. Nießner. "3D-SIS: 3D Semantic Instance Segmentation of RGB-D Scans". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2019 (cit. on pp. 18, 60, 82, 85, 129).

[109] B.-S. Hua, Q.-H. Pham, D. T. Nguyen, M.-K. Tran, L.-F. Yu, and S.-K. Yeung. "SceneNN: a Scene Meshes Dataset with aNNotations". In: *International Conference on 3D Vision*. IEEE, 2016 (cit. on p. 20).

[110] J. Huang, H. Zhang, L. Yi, T. Funkhouser, M. Nießner, and L. J. Guibas. "TextureNet: Consistent Local Parametrizations for Learning from High-Resolution Signals on Meshes". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2019 (cit. on pp. 60, 61).

[111] S. Huang, S. Qi, Y. Zhu, Y. Xiao, Y. Xu, and S.-C. Zhu. "Holistic 3D Scene Parsing and Reconstruction from a Single RGB Image". In: *ECCV*. 2018 (cit. on p. 86).

[112] J. F. Hughes, A. van Dam, M. McGuire, F. D. Sklar, J. D. Foley, S. Feiner, and K. Akeley. In: *Computer Graphics: Principles and Practice*. Addison-Wesley, 2013 (cit. on p. 85).

[113] A. Irschara, C. Zach, J.-M. Frahm, and H. Bischof. "From Structure-from-Motion Point Clouds to Fast Location Recognition". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2009 (cit. on pp. 19, 20, 113).

[114] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. "KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera". In: *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. 2011 (cit. on pp. 18, 59, 61).

[115] H. Izadinia, Q. Shan, and S. M. Seitz. "IM2CAD". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2017 (cit. on p. 105).

[116] C. Jiang, S. Qi, Y. Zhu, S. Huang, J. Lin, L. Yu, D. Terzopoulos, and S. Zhu. "Configurable 3D Scene Synthesis and 2D Image Rendering with Per-pixel Ground Truth Using Stochastic Grammars". In: *International Journal Computer Vision* (2018) (cit. on p. 86).

[117] L. Jiang, H. Zhao, S. Shi, S. Liu, C.-W. Fu, and J. Jia. "PointGroup: Dual-Set Point Grouping for 3D Instance Segmentation". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2020 (cit. on pp. 18, 85, 92).

[118] A. E. Johnson and M. Hebert. "Using Spin Images for Efficient Object Recognition in Cluttered 3D Scenes". In: *Pattern Anal. Mach. Intell.* (1999) (cit. on p. 58).

[119] J. Johnson, R. Krishna, M. Stark, L. Li, D. A. Shamma, M. S. Bernstein, and L. Fei-Fei. "Image Retrieval Using Scene Graphs". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2015 (cit. on pp. 41, 42, 49, 83).

[120] J. Johnson, A. Gupta, and L. Fei-Fei. “Image Generation from Scene Graphs”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2018 (cit. on pp. 83, 89).

[121] I. T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer-Verlag, 2002 (cit. on p. 100).

[122] W. Kabsch. “A discussion of the solution for the best rotation to relate two sets of vectors”. In: *Acta Crystallographica Section A* 34 (1978), pp. 827–828 (cit. on p. 114).

[123] A. Kacete, T. Wentz, and J. Royan. “Decision Forest For Efficient and Robust Camera Relocalization”. In: *International Symposium on Mixed and Augmented Reality*. IEEE, 2017 (cit. on p. 113).

[124] L. Kavan, S. Collins, C. O’Sullivan, and J. Zara. *Dual Quaternions for Rigid Transformation Blending*. Tech. rep. TCD-CS-2006-46. Trinity College Dublin, 2006 (cit. on p. 123).

[125] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb. “Real-Time 3D Reconstruction in Dynamic Scenes Using Point-Based Fusion”. In: *International Conference on 3D Vision*. 2013 (cit. on pp. 18, 59, 65, 71, 72).

[126] A. Kendall, V. Badrinarayanan, and R. Cipolla. “Bayesian SegNet: Model Uncertainty in Deep Convolutional Encoder-Decoder Architectures for Scene Understanding”. In: *British Machine Vision Conference*. 2017 (cit. on p. 58).

[127] A. Kendall and R. Cipolla. “Geometric Loss Functions for Camera Pose Regression with Deep Learning”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2017 (cit. on p. 113).

[128] A. Kendall and R. Cipolla. “Modelling Uncertainty in Deep Learning for Camera Relocalization”. In: *International Conference on Robotics and Automation*. IEEE, 2016 (cit. on p. 113).

[129] A. Kendall, M. Grimes, and R. Cipolla. “PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization”. In: *International Conference on Computer Vision (ICCV)*. 2015 (cit. on pp. 19, 20, 113, 117).

[130] A. C. Kilgour. “A Hierarchical Model of a Graphics System”. In: *SIGGRAPH Comput. Graph.* 15.1 (1981), pp. 35–47 (cit. on p. 85).

[131] D. P. Kingma and J. Ba. “Adam: A Method for Stochastic Optimization”. In: *International Conference on Learning Representations*. 2015 (cit. on pp. 90, 132).

[132] T. N. Kipf and M. Welling. “Semi-Supervised Classification with Graph Convolutional Networks”. In: *International Conference on Learning Representations*. 2017 (cit. on pp. 83, 89).

[133] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollar. “Panoptic Segmentation”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2019 (cit. on p. 60).

[134] G. Klein and D. Murray. “Parallel Tracking and Mapping on a Camera Phone”. In: *International Symposium on Mixed and Augmented Reality*. 2009 (cit. on p. 59).

[135] R. Klokov and V. Lempitsky. “Escape from Cells: Deep Kd-Networks for the Recognition of 3D Point Cloud Models”. In: *International Conference on Computer Vision (ICCV)*. 2017 (cit. on p. 61).

[136] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalanditis, L.-J. Li, D. A. Shamma, M. Bernstein, and L. Fei-Fei. “Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations”. In: *International Journal Computer Vision* (2017) (cit. on pp. 41–43, 51, 84, 85).

[137] Z. Kukelova, J. Heller, and A. Fitzgibbon. “Efficient Intersection of Three Quadrics and Applications in Computer Vision”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2016 (cit. on p. 123).

[138] N. Kulkarni, I. Misra, S. Tulsiani, and A. Gupta. “3D-RelNet: Joint Object and Relational Network for 3D Prediction”. In: *International Conference on Computer Vision (ICCV)* (2019) (cit. on pp. 84, 86).

[139] A. Kundu, X. Yin, A. Fathi, D. Ross, B. Brewington, T. Funkhouser, and C. Pantofaru. “Virtual Multi-view Fusion for 3D Semantic Segmentation”. In: *European Conference on Computer Vision (ECCV)*. 2020 (cit. on p. 61).

[140] J. Lahoud, B. Ghanem, M. Pollefeys, and M. R. Oswald. “3D Instance Segmentation via Multi-Task Metric Learning”. In: *International Conference on Computer Vision (ICCV)*. 2019 (cit. on pp. 18, 60, 82, 85, 92).

[141] K. Lai, L. Bo, X. Ren, and D. Fox. “A Large-Scale Hierarchical Multi-View RGB-D Object Dataset”. In: *International Conference on Robotics and Automation*. 2011, pp. 1817–1824 (cit. on p. 18).

[142] K. Lai, L. Bo, and D. Fox. “Unsupervised Feature Learning for 3D Scene Labeling”. In: *International Conference on Robotics and Automation*. 2014 (cit. on p. 61).

[143] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. “Deeper depth prediction with fully convolutional residual networks”. In: *International Conference on 3D Vision*. 2016 (cit. on p. 18).

[144] M. Larsson, E. Stenborg, C. Toft, L. Hammarstrand, T. Sattler, and F. Kahl. “Fine-Grained Segmentation Networks: Self-Supervised Segmentation for Improved Long-Term Visual Localization”. In: *International Conference on Computer Vision (ICCV)*. 2019 (cit. on pp. 19, 113).

[145] Z. Laskar*, I. Melekhov*, S. Kalia, and J. Kannala. “Camera Relocalization by Computing Pairwise Relative Poses Using Convolutional Neural Network”. In: *International Conference on Computer Vision Workshops*. IEEE, 2017 (cit. on p. 112).

[146] D. Lee and H. Myung. “Solution to the SLAM Problem in Low Dynamic Environments Using a Pose Graph and an RGB-D Sensor”. In: *Sensors* 14.7 (2014), pp. 12467–12496 (cit. on p. 130).

[147] G. H. Lee, B. Li, M. Pollefeys, and F. Fraundorfer. “Minimal solutions for the multi-camera pose estimation problem”. In: *International Journal of Robotics Research* 34 (2015) (cit. on p. 123).

[148] C. Li, H. Xiao, K. Tateno, F. Tombari, N. Navab, and G. D. Hager. “Incremental Scene Understanding on Dense SLAM”. In: *International Conference on Intelligent Robots and Systems*. 2016 (cit. on p. 60).

[149] J. Li, D. Meger, and G. Dudek. “Semantic Mapping for View-Invariant Relocalization”. In: *International Conference on Robotics and Automation*. IEEE, 2019 (cit. on p. 114).

[150] K. Li, M. Rünz, M. Tang, L. Ma, C. Kong, T. Schmidt, I. Reid, L. Agapito, J. Straub, S. Lovegrove, and R. A. Newcombe. “FroDO: From Detections to 3D Objects”. In: *IEEE Conference on Computer Vision and Pattern Recognition* (2020) (cit. on p. 60).

[151] M. Li, A. Gadi Patil, K. Xu, S. Chaudhuri, O. Khan, A. Shamir, C. Tu, B. Chen, D. Cohen-Or, and H. Zhang. “GRAINS: Generative Recursive Autoencoders for Indoor Scenes”. In: *ACM Transactions on Graphics (TOG)* (2018) (cit. on p. 86).

[152] Q. Li, J. Zhu, R. Cao, K. Sun, J. M. Garibaldi, Q. Li, B. Liu, and G. Qiu. “Relative Geometry-Aware Siamese Neural Network for 6DOF Camera Relocalization”. In: *arXiv:1901.01049v2* (2019) (cit. on p. 113).

[153] W. Li, S. Saeedi, J. McCormac, R. Clark, D. Tzoumanikas, Q. Ye, Y. Huang, R. Tang, and S. Leutenegger. “InteriorNet: Mega-scale Multi-sensor Photo-realistic Indoor Scenes Dataset”. In: *British Machine Vision Conference*. 2018 (cit. on pp. 19, 20).

[154] X. Li and S. Jiang. “Know More Say Less: Image Captioning Based on Scene Graphs”. In: *IEEE Transactions on Multimedia* (2019) (cit. on p. 86).

[155] X. Li, J. Ylioinas, and J. Kannala. “Full-Frame Scene Coordinate Regression for Image-Based Localization”. In: *Robotics: Science and Systems*. 2018 (cit. on p. 114).

[156] X. Li, J. Ylioinas, J. Verbeek, and J. Kannala. “Scene Coordinate Regression with Angle-Based Reprojection Loss for Camera Relocalization”. In: *European Conference on Computer Vision (ECCV)*. 2018 (cit. on p. 114).

[157] X. Li, C. Wu, C. Zach, S. Lazebnik, and J. Frahm. “Modeling and Recognition of Landmark Image Collections Using Iconic Scene Graphs”. In: *European Conference on Computer Vision (ECCV)*. 2008 (cit. on pp. 19, 20).

[158] Y. Li, T. Zhang, Y. Nakamura, and T. Harada. “SplitFusion: Simultaneous Tracking and Mapping for Non-Rigid Scenes”. In: *International Conference on Intelligent Robots and Systems*. 2020 (cit. on p. 129).

[159] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen. “PointCNN: Convolution On X-Transformed Points”. In: *Advances in Neural Information Processing Systems*. 2018 (cit. on p. 62).

[160] Y. Li, S. Pirk, H. Su, C. R. Qi, and L. J. Guibas. “FPNN: Field Probing Neural Networks for 3D Data”. In: *International Conference on Neural Information Processing Systems*. 2016 (cit. on p. 77).

[161] Y. Li, W. Ouyang, X. Wang, and X. Tang. “ViP-CNN: Visual Phrase Guided Convolutional Neural Network”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2017 (cit. on pp. 83, 84).

[162] Y. Li, W. Ouyang, B. Zhou, J. Shi, C. Zhang, and X. Wang. “Factorizable Net: An Efficient Subgraph-Based Framework for Scene Graph Generation”. In: *European Conference on Computer Vision (ECCV)*. 2018 (cit. on pp. 83, 84).

[163] Y. Li, N. Snavely, D. Huttenlocher, and P. Fua. “Worldwide Pose Estimation using 3D Point Clouds”. In: *European Conference on Computer Vision (ECCV)*. 2012 (cit. on pp. 19, 20).

[164] H. Lim, S. N. Sinha, M. F. Cohen, and M. Uyttendaele. “Real-Time Image-Based 6-DOF Localization in Large-Scale Environments”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012 (cit. on p. 7).

[165] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. “Focal Loss for Dense Object Detection”. In: *International Conference on Computer Vision (ICCV)*. 2017 (cit. on p. 90).

[166] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár. “Microsoft COCO: Common Objects in Context”. In: 2014 (cit. on p. 42).

[167] T. Liu, S. Chaudhuri, V. Kim, Q. Huang, N. Mitra, and T. Funkhouser. “Creating Consistent Scene Graphs Using a Probabilistic Grammar”. In: *ACM Transactions on Graphics (TOG)* (2014) (cit. on p. 86).

[168] Y. Liu, D. Zhang, G. Lu, and W.-Y. Ma. “A Survey of Content-based Image Retrieval With High-level Semantics”. In: *Pattern Recognition* (2007) (cit. on p. 83).

[169] Y. Liu, L. Yi, S. Zhang, Q. Fan, T. A. Funkhouser, and H. Dong. “P4Contrast: Contrastive Learning with Pairs of Point-Pixel Pairs for RGB-D Scene Understanding”. In: *CoRR* abs/2012.13089 (2020). arXiv: 2012.13089 (cit. on p. 95).

[170] W. E. Lorensen and H. E. Cline. “Marching Cubes: A High Resolution 3D Surface Construction Algorithm”. In: *SIGGRAPH Comput. Graph.* 21.4 (1987) (cit. on p. 24).

[171] C. Lu, R. Krishna, M. Bernstein, and L. Fei-Fei. “Visual Relationship Detection with Language Priors”. In: *European Conference on Computer Vision (ECCV)*. 2016 (cit. on pp. 83, 84, 88, 94).

[172] G. Lu, Y. Yan, A. Kolagunda, and C. Kambhamettu. “A Fast 3D Indoor-Localization Approach Based on Video Queries”. In: *MultiMedia Modeling*. 2016 (cit. on pp. 105, 114).

[173] S. Lynen, T. Sattler, M. Bosse, J. Hesch, M. Pollefeys, and R. Siegwart. “Get Out of My Lab: Large-scale, Real-Time Visual-Inertial Localization”. In: *Robotics: Science and Systems*. 2015 (cit. on pp. 7, 24).

[174] R. Ma, A. G. Patil, M. Fisher, M. Li, S. Pirk, B.-S. Hua, S.-K. Yeung, X. Tong, L. Guibas, and H. Zhang. “Language-Driven Synthesis of 3D Scenes from Scene Databases”. In: *SIGGRAPH Asia, Technical Papers*. 2018 (cit. on p. 86).

[175] L. van der Maaten and G. Hinton. “Visualizing Data using t-SNE”. In: *Journal of Machine Learning Research* 9 (2008), pp. 2579–2605 (cit. on p. 98).

[176] W. Maddern, G. Pascoe, C. Linegar, and P. Newman. “1 year, 1000km: The Oxford RobotCar Dataset”. In: *International Journal of Robotics Research* 36.1 (2017) (cit. on pp. 19, 20, 112).

[177] M. Malinowski and M. Fritz. “A Multi-World Approach to Question Answering about Real-World Scenes based on Uncertain Input”. In: *International Conference on Neural Information Processing Systems*. 2014 (cit. on p. 6).

[178] D. Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. San Francisco: W. H. Freeman and Company, 1982 (cit. on p. 7).

[179] D. Massiceti, A. Krull, E. Brachmann, C. Rother, and P. H. S. Torr. “Random Forests versus Neural Networks – What’s Best for Camera Localization?” In: *International Conference on Robotics and Automation*. IEEE, 2017 (cit. on p. 114).

[180] D. Maturana and S. Scherer. “VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition”. In: *International Conference on Intelligent Robots and Systems*. 2015 (cit. on p. 61).

[181] J. McCormac, R. Clark, M. Bloesch, A. Davison, and S. Leutenegger. “Fusion++: Volumetric Object-Level SLAM”. In: *International Conference on 3D Vision*. 2018 (cit. on pp. 60, 129).

[182] J. McCormac, A. Handa, A. Davison, and S. Leutenegger. “SemanticFusion: Dense 3D Semantic Mapping with Convolutional Neural Networks”. In: *International Conference on Robotics and Automation*. 2017 (cit. on pp. 60, 69, 73, 75, 76, 129).

[183] J. McCormac, A. Handa, S. Leutenegger, and A. Davison. “SceneNet RGB-D: Can 5M Synthetic Images Beat Generic ImageNet Pre-training on Indoor Segmentation?” In: *International Conference on Computer Vision (ICCV)*. 2017 (cit. on pp. 19, 60, 62).

[184] D. Meagher. *Octree Encoding: a New Technique for the Representation, Manipulation and Display of Arbitrary 3-D Objects by Computer*. 1980 (cit. on pp. 59, 61).

[185] I. Melekhov, J. Ylioinas, J. Kannala, and E. Rahtu. “Image-based Localization using Hourglass Networks”. In: *International Conference on Computer Vision Workshops*. IEEE, 2017 (cit. on p. 113).

[186] L. Meng, J. Chen, F. Tung, J. J. Little, J. Valentin, and C. W. de Silva. “Backtracking Regression Forests for Accurate Camera Relocalization”. In: *International Conference on Intelligent Robots and Systems*. IEEE, 2017 (cit. on p. 114).

[187] L. Meng, J. Chen, F. Tung, J. Little, and C. Silva. “Exploiting Random RGB and Sparse Features for Camera Pose Estimation”. In: *British Machine Vision Conference*. BMVA, 2016 (cit. on p. 114).

[188] L. Meng, F. Tung, J. J. Little, J. Valentin, and C. W. de Silva. “Exploiting Points and Lines in Regression Forests for RGB-D Camera Relocalization”. In: *International Conference on Intelligent Robots and Systems*. IEEE, 2018 (cit. on p. 114).

[189] F. Milletari, N. Navab, and S. Ahmadi. “V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation”. In: *CoRR* (2016) (cit. on p. 61).

[190] G. Mittal, S. Agrawal, A. Agarwal, S. Mehta, and T. Marwah. “Interactive Image Generation Using Scene Graphs”. In: *ICLR Deep Generative Models for Highly Structured Data Workshop* (2019) (cit. on p. 83).

[191] K. Mo, P. Guerrero, L. Yi, H. Su, P. Wonka, N. Mitra, and L. Guibas. “StructureNet: Hierarchical Graph Networks for 3D Shape Generation”. In: *ACM Transactions on Graphics (TOG)* (2019) (cit. on p. 86).

[192] J. P. S. do Monte Lima and V. Teichrieb. “An Efficient Global Point Cloud Descriptor for Object Recognition and Pose Estimation”. In: *2016 29th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)* (2016) (cit. on p. 74).

[193] R. Mur-Artal and J. D. Tardós. “ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras”. In: *IEEE Transactions on Robotics* 33.5 (2017), pp. 1255–1262 (cit. on p. 19).

[194] M. Najibi, G. Lai, A. Kundu, Z. Lu, V. Rathod, T. Funkhouser, C. Pantofaru, D. Ross, L. S. Davis, and A. Fathi. “DOPS: Learning to Detect 3D Objects and Predict Their 3D Shapes”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2020 (cit. on pp. 86, 92).

[195] Y. Nakajima, K. Tateno, F. Tombari, and H. Saito. “Fast and Accurate Semantic Mapping through Geometric-based Incremental Segmentation”. In: *International Conference on Intelligent Robots and Systems*. 2018 (cit. on pp. 60, 129).

[196] R. Nakashima and A. Seki. “SIR-Net: Scene-Independent End-to-End Trainable Visual Relocalizer”. In: *International Conference on 3D Vision*. IEEE, 2019 (cit. on p. 114).

[197] G. Narita, T. Seno, T. Ishikawa, and Y. Kaji. “PanopticFusion: Online Volumetric Semantic Mapping at the Level of Stuff and Things”. In: *International Conference on Intelligent Robots and Systems*. IEEE. 2019, pp. 4205–4212 (cit. on p. 60).

[198] M. Naseer, S. Khan, and F. Porikli. “Indoor Scene Understanding in 2.5/3D for Autonomous Agents: A Survey”. In: *IEEE Access* 7 (2019), pp. 1859–1887 (cit. on p. 6).

[199] P. K. Nathan Silberman Derek Hoiem and R. Fergus. “Indoor Segmentation and Support Inference from RGBD Images”. In: *European Conference on Computer Vision (ECCV)*. 2012 (cit. on pp. 18–20, 42, 48, 51, 68–70, 73, 85).

[200] E. D. Nerurkar, K. J. Wu, and S. I. Roumeliotis. “C-KLAM: Constrained keyframe-based localization and mapping”. In: *International Conference on Robotics and Automation*. 2014, pp. 3638–3643 (cit. on p. 24).

[201] R. A. Newcombe, D. Fox, and S. M. Seitz. “DynamicFusion: Reconstruction and Tracking of Non-Rigid Scenes in Real-Time”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2015 (cit. on p. 129).

[202] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. “KinectFusion: Real-Time Dense Surface Mapping and Tracking”. In: *International Symposium on Mixed and Augmented Reality*. 2011 (cit. on pp. 8, 18, 24, 59).

[203] A. Newell and J. Deng. “Pixels to Graphs by Associative Embedding”. In: *International Conference on Neural Information Processing Systems*. 2017 (cit. on pp. 83, 84).

[204] Y. Nie, X. Han, S. Guo, Y. Zheng, J. Chang, and J. J. Zhang. “Total3DUnderstanding: Joint Layout, Object Pose and Mesh Reconstruction for Indoor Scenes From a Single Image”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2020 (cit. on p. 86).

[205] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. “Real-time 3D Reconstruction at Scale using Voxel Hashing”. In: *ACM Transactions on Graphics (TOG)*. 2013 (cit. on pp. 8, 18, 59, 60).

[206] P. Núñez, P. Drews, A. Bandera, R. Rocha, M. Campos, and J. Dias. “Change detection in 3D environments based on Gaussian Mixture Model and robust structural matching for autonomous robotic applications”. In: *International Conference on Intelligent Robots and Systems*. 2010 (cit. on p. 130).

[207] E. P. Örnek, S. Mudgal, J. Wald, Y. Wang, and F. Tombari. “From 2D to 3D: Re-thinking Benchmarking of Monocular Depth Prediction”. In: vol. abs/2203.08122. 2021. arXiv: 2203.08122 (cit. on pp. 17, 39, 147).

[208] S. Papert. “The Summer Vision Project”. In: (July 1966) (cit. on p. 6).

[209] J. J. Park, P. Florence, J. Straub, R. A. Newcombe, and S. Lovegrove. “DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2019 (cit. on p. 60).

[210] R. Paucher and M. Turk. "Location-based augmented reality on mobile phones". In: *IEEE Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, 2010 (cit. on p. 7).

[211] J. Peyre, I. Laptev, C. Schmid, and J. Sivic. "Weakly-supervised learning of visual relations". In: *International Conference on Computer Vision (ICCV)*. 2017 (cit. on pp. 83, 84).

[212] Q.-H. Pham, B.-S. Hua, T. Nguyen, and S.-K. Yeung. "Real-time Progressive 3D Semantic Segmentation for Indoor Scene". In: *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2019 (cit. on p. 60).

[213] R. Pless. "Using Many Cameras as One". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2003 (cit. on p. 123).

[214] H. Porav, W. Maddern, and P. Newman. "Adversarial Training for Adverse Conditions: Robust Metric Localisation Using Appearance Transfer". In: *International Conference on Robotics and Automation*. IEEE, 2018 (cit. on pp. 19, 20).

[215] C. R. Qi, O. Litany, K. He, and L. J. Guibas. "Deep Hough Voting for 3D Object Detection in Point Clouds". In: *International Conference on Computer Vision (ICCV)*. 2019 (cit. on pp. 60, 85).

[216] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". In: *IEEE Conference on Computer Vision and Pattern Recognition* (2017) (cit. on pp. 18, 59–61, 69, 70, 75–77, 82, 83, 88, 91, 129, 130).

[217] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. Guibas. "Volumetric and Multi-View CNNs for Object Classification on 3D Data". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2016 (cit. on pp. 61, 82, 94).

[218] C. R. Qi, L. Yi, H. Su, and L. Guibas. "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space". In: *International Conference on Neural Information Processing Systems*. 2017 (cit. on pp. 18, 60, 62, 69, 70, 75–78, 82, 129).

[219] M. Qi, W. Li, Z. Yang, Y. Wang, and J. Luo. "Attentive Relational Networks for Mapping Images to Scene Graphs". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2019 (cit. on pp. 83, 84).

[220] N. Radwan, A. Valada, and W. Burgard. "VLocNet++: Deep Multitask Learning for Semantic Visual Localization and Odometry". In: *Robotics and Automation Letters* 3.4 (2018), pp. 4407–4414 (cit. on p. 113).

[221] S. Ren, K. He, R. Girshick, and J. Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *International Conference on Neural Information Processing Systems*. 2015 (cit. on p. 84).

[222] D. Rethage, J. Wald, J. Sturm, N. Navab, and F. Tombari. "Fully-Convolutional Point Networks for Large-Scale Point Clouds". In: *European Conference on Computer Vision (ECCV)*. 2018 (cit. on pp. 11, 18, 57–60, 62, 69, 70, 76–79, 82, 129, 141, 147).

[223] G. Riegler, O. Ulusoy, and A. Geiger. "OctNet: Learning Deep 3D Representations at High Resolutions". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2017 (cit. on p. 61).

[224] N. L. Rodas, F. Barrera, and N. Padoy. "Marker-less AR in the Hybrid Room using Equipment Detection for Camera Relocalization". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2015 (cit. on p. 7).

[225] O. Ronneberger, P.Fischer, and T. Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2015 (cit. on p. 92).

[226] A. Rosinol, M. Abate, Y. Chang, and L. Carlone. "Kimera: an Open-Source Library for Real-Time Metric-Semantic Localization and Mapping". In: *International Conference on Robotics and Automation*. 2020 (cit. on p. 62).

[227] A. Rosinol, A. Gupta, M. Abate, J. Shi, and L. Carlone. “3D Dynamic Scene Graphs: Actionable Spatial Perception with Places, Objects, and Humans”. In: *rss*. 2020 (cit. on p. 42).

[228] M. Rünz, M. Buffier, and L. Agapito. “MaskFusion: Real-Time Recognition, Tracking and Reconstruction of Multiple Moving Objects”. In: *International Symposium on Mixed and Augmented Reality*. 2018 (cit. on p. 129).

[229] M. Rünz and L. Agapito. “Co-Fusion: Real-time Segmentation, Tracking and Fusion of Multiple Objects”. In: *International Conference on Robotics and Automation*. 2017 (cit. on p. 129).

[230] R. Russell Chrisand Yu and L. Agapito. “Video Pop-up: Monocular 3D Reconstruction of Dynamic Scenes”. In: *European Conference on Computer Vision (ECCV)*. 2014 (cit. on p. 129).

[231] R. B. Rusu. “Andreas Nüchter (2009): 3D Robotic Mapping: The Simultaneous Localization and Mapping Problem with Six Degrees of Freedom”. In: *KI* (2010) (cit. on p. 59).

[232] R. B. Rusu, N. Blodow, and M. Beetz. “Fast Point Feature Histograms (FPFH) for 3D Registration”. In: *International Conference on Robotics and Automation*. 2009 (cit. on pp. 58, 73, 75, 127, 130, 134, 135).

[233] R. B. Rusu, G. R. Bradski, R. Thibaux, and J. M. Hsu. “Fast 3D recognition and pose using the Viewpoint Feature Histogram”. In: (cit. on pp. 59, 73–75).

[234] R. B. Rusu and S. Cousins. “3D is here: Point Cloud Library (PCL)”. In: *International Conference on Robotics and Automation*. 2011 (cit. on p. 135).

[235] S. Saeedi, E. D. C. Carvalho, W. Li, D. Tzoumanikas, S. Leutenegger, P. H. J. Kelly, and A. J. Davison. “Characterizing Visual Localization and Mapping Datasets”. In: *International Conference on Robotics and Automation*. 2019 (cit. on p. 20).

[236] R. F. Salas-Moreno, B. Glocker, P. H. J. Kelly, and A. J. Davison. “Dense planar SLAM”. In: *International Symposium on Mixed and Augmented Reality*. 2014 (cit. on p. 59).

[237] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison. “SLAM++: Simultaneous Localisation and Mapping at the Level of Objects”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2013 (cit. on pp. 59, 60, 114, 129).

[238] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich. “SuperGlue: Learning Feature Matching with Graph Neural Networks”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2020 (cit. on pp. 59, 84).

[239] P.-E. Sarlin, C. Cadena, R. Siegwart, and M. Dymczyk. “From Coarse to Fine: Robust Hierarchical Localization at Large Scale”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2019 (cit. on pp. 19, 113, 114, 119, 120, 122).

[240] T. Sattler, B. Leibe, and L. Kobbelt. “Efficient & Effective Prioritized Matching for Large-Scale Image-Based Localization”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 9 (2017) (cit. on pp. 113, 120, 122).

[241] T. Sattler, B. Leibe, and L. Kobbelt. “Fast Image-Based Localization using Direct 2D-to-3D Matching”. In: *International Conference on Computer Vision (ICCV)*. 2011 (cit. on pp. 119, 124).

[242] T. Sattler, B. Leibe, and L. Kobbelt. “Improving Image-Based Localization by Active Correspondence Search”. In: *European Conference on Computer Vision (ECCV)*. 2012 (cit. on p. 113).

[243] T. Sattler, W. Maddern, C. Toft, A. Torii, L. Hammarstrand, E. Stenborg, D. Safari, M. Okutomi, M. Pollefeys, J. Sivic, F. Kahl, and T. Pajdla. “Benchmarking 6DOF Outdoor Visual Localization in Changing Conditions”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2018 (cit. on pp. 19, 20, 112, 113).

[244] T. Sattler, T. Weyand, B. Leibe, and L. Kobbelt. “Image Retrieval for Image-Based Localization Revisited”. In: *British Machine Vision Conference*. BMVA, 2012 (cit. on pp. 19, 20, 112).

[245] T. Sattler, Q. Zhou, M. Pollefeys, and L. Leal-Taixé. “Understanding the Limitations of CNN-based Absolute Camera Pose Regression”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2019 (cit. on p. 113).

[246] J. L. Schönberger, M. Pollefeys, A. Geiger, and T. Sattler. “Semantic Visual Localization”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2018 (cit. on pp. 113, 114).

[247] J. L. Schönberger and J.-M. Frahm. “Structure-from-Motion Revisited”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2016 (cit. on p. 20).

[248] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm. “Pixelwise View Selection for Unstructured Multi-View Stereo”. In: *European Conference on Computer Vision (ECCV)*. 2016 (cit. on p. 20).

[249] P. Schönemann. “A generalized solution of the orthogonal procrustes problem”. In: (1966) (cit. on p. 31).

[250] T. Schöps, T. Sattler, C. Häne, and M. Pollefeys. “3D Modeling on the Go: Interactive 3D Reconstruction of Large-Scale Scenes on Mobile Devices”. In: *International Conference on 3D Vision*. 2015 (cit. on p. 64).

[251] E. Shelhamer, J. Long, and T. Darrell. “Fully Convolutional Networks for Semantic Segmentation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2016) (cit. on pp. 58, 59).

[252] Y. Shi, A. X. Chang, Z. Wu, M. Savva, and K. Xu. “Hierarchy Denoising Recursive Autoencoders for 3D Scene Layout Prediction”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2019 (cit. on p. 86).

[253] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon. “Scene Coordinate Regression Forests for Camera Relocalization in RGB-D Images”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2013 (cit. on pp. 19, 20, 111, 114).

[254] I. Sipiran and B. Bustos. “Harris 3D: a robust extension of the Harris operator for interest point detection on 3D meshes”. In: *Vis. Comput.* (2011) (cit. on p. 132).

[255] M. Slavcheva, M. Baust, D. Cremers, and S. Ilic. “KillingFusion: Non-Rigid 3D Reconstruction Without Correspondences”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2017 (cit. on p. 129).

[256] K. Sohn. “Improved Deep Metric Learning with Multi-class N-pair Loss Objective”. In: *International Conference on Neural Information Processing Systems*. Vol. 29. Curran Associates, Inc., 2016, pp. 1857–1865 (cit. on p. 92).

[257] S. Song, S. Lichtenberg, and J. Xiao. “SUN RGB-D: A RGB-D Scene Understanding Benchmark Suite”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2015 (cit. on pp. 19, 20, 82, 86).

[258] S. Song and J. Xiao. “Deep Sliding Shapes for Amodal 3D Object Detection in RGB-D Images”. In: *CoRR* (2015) (cit. on p. 61).

[259] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser. “Semantic Scene Completion from a Single Depth Image”. In: *IEEE Conference on Computer Vision and Pattern Recognition* (2017) (cit. on pp. 18, 19, 59, 61, 76, 132).

[260] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma, A. Clarkson, M. Yan, B. Budge, Y. Yan, X. Pan, J. Yon, Y. Zou, K. Leon, N. Carter, J. Briales, T. Gillingham, E. Mueggler, L. Pesqueira, M. Savva, D. Batra, H. M. Strasdat, R. D. Nardi, M. Goesele, S. Lovegrove, and R. Newcombe. “The Replica Dataset: A Digital Replica of Indoor Spaces”. In: *arXiv preprint arXiv:1906.05797* (2019) (cit. on p. 19).

[261] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. “A Benchmark for the Evaluation of RGB-D SLAM Systems”. In: *International Conference on Intelligent Robots and Systems*. 2012, pp. 573–580 (cit. on p. 18).

[262] H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller. “Multi-view Convolutional Neural Networks for 3D Shape Recognition”. In: *International Conference on Computer Vision (ICCV)*. 2015 (cit. on p. 61).

[263] E. Sucar, K. Wada, and A. Davison. “NodeSLAM: Neural Object Descriptors for Multi-View Shape Reconstruction”. In: *International Conference on 3D Vision*. 2020 (cit. on p. 129).

[264] X. Sun, J. Wu, X. Zhang, Z. Zhang, C. Zhang, T. Xue, J. B. Tenenbaum, and W. T. Freeman. “Pix3D: Dataset and Methods for Single-Image 3D Shape Modeling”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2018 (cit. on p. 105).

[265] C. Sweeney, V. Fragoso, T. Höllerer, and M. Turk. “gDLS: A Scalable Solution to the Generalized Pose and Scale Problem”. In: *European Conference on Computer Vision (ECCV)*. 2014 (cit. on p. 123).

[266] H. Taira, M. Okutomi, T. Sattler, M. Cimpoi, M. Pollefeys, J. Sivic, T. Pajdla, and A. Torii. “InLoc: Indoor Visual Localization with Dense Matching and View Synthesis”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2018 (cit. on pp. 19, 20, 113, 114).

[267] H. Taira, I. Rocco, J. Sedlar, M. Okutomi, J. Sivic, T. Pajdla, T. Sattler, and A. Torii. “Is This the Right Place? Geometric-Semantic Pose Verification for Indoor Visual Localization”. In: *International Conference on Computer Vision (ICCV)*. 2019 (cit. on pp. 113, 114).

[268] A. Taneja, L. Ballan, and M. Pollefeys. “Geometric Change Detection in Urban Environments Using Images”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.11 (2015), pp. 2193–2206 (cit. on p. 130).

[269] K. Tateno, F. Tombari, I. Laina, and N. Navab. “CNN-SLAM: Real-Time Dense Monocular SLAM with Learned Depth Prediction”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2017 (cit. on p. 60).

[270] K. Tateno, F. Tombari, and N. Navab. “Real-Time and Scalable Incremental Segmentation on Dense SLAM”. In: *International Conference on Intelligent Robots and Systems*. 2015 (cit. on pp. 60, 63–65, 68, 71, 72, 100).

[271] K. Tateno, F. Tombari, and N. Navab. “When 2.5D is not enough: Simultaneous Reconstruction, Segmentation and Recognition on dense SLAM”. In: *International Conference on Robotics and Automation* (2016) (cit. on pp. 59, 60, 62).

[272] G. Te, W. Hu, A. Zheng, and Z. Guo. “RGCNN: Regularized Graph CNN for Point Cloud Segmentation”. In: *International Conference on Multimedia*. 2018 (cit. on p. 86).

[273] D. Teney, L. Liu, and A. Van Den Hengel. “Graph-Structured Representations for Visual Question Answering”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2017 (cit. on p. 83).

[274] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas. “KPConv: Flexible and Deformable Convolution for Point Clouds”. In: *International Conference on Computer Vision (ICCV)*. 2019 (cit. on p. 82).

[275] B. Thomee, D. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li. “The New Data and New Challenges in Multimedia Research”. In: (2015) (cit. on p. 42).

[276] C. Toft, C. Olsson, and F. Kahl. “Long-term 3D Localization and Pose from Semantic Labellings”. In: *International Conference on Computer Vision (ICCV)*. 2017 (cit. on pp. 19, 113).

[277] C. Toft, E. Stenborg, L. Hammarstrand, L. Brynte, M. Pollefeys, T. Sattler, and F. Kahl. “Semantic Match Consistency for Long-Term Visual Localization”. In: *European Conference on Computer Vision (ECCV)*. 2018 (cit. on pp. 19, 113).

[278] C. Tomasi and R. Manduchi. “Bilateral Filtering for Gray and Color Images”. In: *International Conference on Computer Vision (ICCV)*. 1998 (cit. on p. 64).

[279] F. Tombari, S. Salti, and L. Di Stefano. “Unique Signatures of Histograms for Local Surface Description”. In: *European Conference on Computer Vision (ECCV)*. 2010 (cit. on pp. 58, 128, 130, 134, 135).

[280] A. Torii, R. Arandjelović, J. Sivic, M. Okutomi, and T. Pajdla. “24/7 place recognition by view synthesis”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2015 (cit. on pp. 105, 112, 120, 122).

[281] A. Torii, J. Sivic, and T. Pajdla. “Visual localization by linear combination of image descriptors”. In: *International Conference on Computer Vision Workshops*. IEEE, 2011 (cit. on p. 112).

[282] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. “Bundle Adjustment - A Modern Synthesis”. In: International Conference on Computer Vision (ICCV). Springer-Verlag, 1999 (cit. on p. 24).

[283] M. Ulrich, B. Drost, N. Navab, and S. Ilic. “Model globally, match locally: Efficient and robust 3D object recognition”. In: *IEEE Conference on Computer Vision and Pattern Recognition* (2010) (cit. on pp. 58, 59).

[284] A. Valada*, N. Radwan*, and W. Burgard. “Deep Auxiliary Learning for Visual Localization and Odometry”. In: *International Conference on Robotics and Automation*. IEEE, 2018 (cit. on p. 113).

[285] J. P. C. Valentin, V. Vineet, M. Cheng, D. Kim, J. Shotton, P. Kohli, M. Nießner, A. Criminisi, S. Izadi, and P. H. S. Torr. “SemanticPaint: Interactive 3D Labeling and Learning at your Fingertips”. In: *ACM Trans. Graph.* (2015) (cit. on pp. 7, 60, 62, 68, 129).

[286] J. Valentin, A. Dai, M. Nießner, P. Kohli, P. Torr, S. Izadi, and C. Keskin. “Learning to Navigate the Energy Landscape”. In: *International Conference on 3D Vision*. 2016 (cit. on pp. 19, 20, 114).

[287] J. Valentin, M. Nießner, J. Shotton, A. Fitzgibbon, S. Izadi, and P. Torr. “Exploiting Uncertainty in Regression Forests for Accurate Camera Relocalization”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2015 (cit. on pp. 68, 114).

[288] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. “Attention is all you need”. In: *Advances in neural information processing systems*. 2017 (cit. on p. 84).

[289] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. “Graph Attention Networks”. In: *International Conference on Learning Representations*. 2018 (cit. on p. 84).

[290] J. Ventura, C. Arth, G. Reitmayr, and D. Schmalstieg. “A Minimal Solution to the Generalized Pose-and-Scale Problem”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2014 (cit. on p. 123).

[291] B. Vijay, K. Alex, and C. Roberto. “SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017) (cit. on p. 58).

[292] N. Wade and M. Swanston. “Visual perception: An introduction”. In: *third edition*. 2013, pp. 1–322 (cit. on p. 6).

[293] F. Walch, C. Hazırbaş, L. Leal-Taixé, T. Sattler, S. Hilsenbeck, and D. Cremers. “Image-based localization using LSTMs for structured feature correlation”. In: *International Conference on Computer Vision (ICCV)*. 2017 (cit. on p. 113).

[294] A. Walcott-Bryant, M. Kaess, H. Johannsson, and J. J. Leonard. “Dynamic pose graph SLAM: Long-term mapping in low dynamic environments”. In: *International Conference on Intelligent Robots and Systems*. 2012 (cit. on p. 130).

[295] J. Wald, A. Avetisyan, N. Navab, F. Tombari, and M. Nießner. “RIO: 3D Object Instance Re-Localization in Changing Indoor Environments”. In: *International Conference on Computer Vision (ICCV)*. 2019 (cit. on pp. 17, 21, 22, 35, 36, 42, 43, 51, 59, 94, 95, 98, 105, 106, 127, 128, 131, 135, 141, 142, 147).

[296] J. Wald, H. Dhamo, N. Navab, and F. Tombari. “Learning 3D Semantic Scene Graphs from 3D Indoor Reconstructions”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2020 (cit. on pp. 11, 41, 62, 81–83, 87, 90, 91, 93–96, 101, 105, 106, 108, 141, 142, 147).

[297] J. Wald, N. Navab, and F. Tombari. “Learning 3D Semantic Scene Graphs with Instance Embeddings”. In: *International Journal Computer Vision*. 2022 (cit. on pp. 11, 81, 83, 87, 90, 91, 93, 99, 101, 141, 147).

[298] J. Wald, T. Sattler, S. Golodetz, T. Cavallari, and F. Tombari. “Beyond Controlled Environments: 3D Camera Re-Localization in Changing Indoor Scenes”. In: *European Conference on Computer Vision (ECCV)*. 2020 (cit. on pp. 17, 20, 21, 35, 105, 111, 117–126, 142, 147).

[299] J. Wald, K. Tateno, J. Sturm, N. Navab, and F. Tombari. “Real-Time Fully Incremental Scene Understanding on Mobile Platforms”. In: *Robotics and Automation Letters*. 2018 (cit. on pp. 11, 57, 58, 60, 62, 69, 70, 73, 78, 129, 141, 147).

[300] K. Wang, M. Savva, A. Chang, and D. Ritchie. “Deep convolutional priors for indoor scene synthesis”. In: *ACM Transactions on Graphics (TOG)* (2018) (cit. on p. 86).

[301] M. Wang, Y.-K. Lai, Y. Liang, R. R. Martin, and S.-M. Hu. “BiggerPicture: Data-driven Image Extrapolation Using Graph Matching”. In: *ACM Transactions on Graphics (TOG)* (2014) (cit. on p. 83).

[302] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. “Dynamic Graph CNN for Learning on Point Clouds”. In: *ACM Transactions on Graphics (TOG)* (2019) (cit. on p. 62).

[303] T. Whelan, S. Leutenegger, R. S. Moreno, B. Glocker, and A. Davison. “ElasticFusion: Dense SLAM Without A Pose Graph”. In: *Robotics: Science and Systems*. 2015 (cit. on pp. 8, 18, 19, 59, 60).

[304] A. R. Widya, A. Torii, and M. Okutomi. “Structure from Motion Using Dense CNN Features With Keypoint Relocalization”. In: *Transactions on Computer Vision and Applications* 10.1 (2018), p. 6 (cit. on p. 113).

[305] E. Wijmans and Y. Furukawa. “Exploiting 2D Floorplan for Building-scale Panorama RGBD Alignment”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2017 (cit. on p. 21).

[306] J. Wu, L. Ma, and X. Hu. “Delving Deeper into Convolutional Neural Networks for Camera Relocalization”. In: *International Conference on Robotics and Automation*. 2017 (cit. on p. 113).

[307] S.-C. Wu, K. Tateno, N. Navab, and F. Tombari. “SCFusion: Real-time Incremental Scene Reconstruction with Semantic Completion”. In: *International Conference on 3D Vision*. 2020 (cit. on p. 18).

[308] S.-C. Wu, J. Wald, K. Tateno, N. Navab, and F. Tombari. “SceneGraphFusion: Incremental 3D Scene Graph Prediction from RGB-D Sequences”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2021 (cit. on pp. 11, 62, 78, 81, 83, 87, 100, 101, 141, 147).

[309] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. “3D ShapeNets: A deep representation for volumetric shapes.” In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2015 (cit. on pp. 60, 61).

[310] F. Xia, A. R. Zamir, Z.-Y. He, A. Sax, J. Malik, and S. Savarese. “Gibson Env: Real-World Perception for Embodied Agents”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2018 (cit. on p. 42).

[311] F. Xia, A. R. Zamir, Z.-Y. He, A. Sax, J. Malik, and S. Savarese. “Gibson Env: Real-World Perception for Embodied Agents”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2018 (cit. on pp. 44, 47).

[312] B. Xu, W. Li, D. Tzoumanikas, M. Bloesch, A. Davison, and S. Leutenegger. “MID-fusion: octree-based object-level multi-instance dynamic SLAM”. In: *International Conference on Robotics and Automation*. 2019 (cit. on p. 129).

[313] D. Xu, Y. Zhu, C. Choy, and L. Fei-Fei. “Scene Graph Generation by Iterative Message Passing”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2017 (cit. on pp. 83, 84, 88, 94).

[314] J. Yang, J. Lu, S. Lee, D. Batra, and D. Parikh. “Graph R-CNN for Scene Graph Generation”. In: *European Conference on Computer Vision (ECCV)*. 2018 (cit. on pp. 83, 84, 88, 94).

[315] L. Yang, H. Uchiyama, J. Normand, G. Moreau, H. Nagahara, and R. Taniguchi. “Real-Time Surface of Revolution Reconstruction on Dense SLAM”. In: *International Conference on 3D Vision*. 2016 (cit. on p. 60).

[316] L. Yang*, Z. Bai*, C. Tang, H. Li, Y. Furukawa, and P. Tan. “SANet: Scene Agnostic Network for Camera Localization”. In: *International Conference on Computer Vision (ICCV)*. 2019 (cit. on p. 114).

[317] X. Yang, K. Tang, H. Zhang, and J. Cai. “Auto-Encoding Scene Graphs for Image Captioning”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2019 (cit. on p. 83).

[318] L. Yi, V. G. Kim, D. Ceylan, I.-C. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, and L. Guibas. “A Scalable Active Framework for Region Annotation in 3D Shape Collections”. In: *SIGGRAPH Asia* (2016) (cit. on p. 77).

[319] L. Yi, H. Su, X. Guo, and L. Guibas. “SyncSpecCNN: Synchronized Spectral CNN for 3D Shape Segmentation”. In: *IEEE Conference on Computer Vision and Pattern Recognition* (2016) (cit. on p. 77).

[320] L. Yi, W. Zhao, H. Wang, M. Sung, and L. J. Guibas. “GSPN: Generative Shape Proposal Network for 3D Instance Segmentation in Point Cloud”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2019 (cit. on pp. 18, 82).

[321] A. R. Zamir and M. Shah. “Accurate Image Localization Based on Google Maps Street View”. In: *European Conference on Computer Vision (ECCV)*. Springer, 2010 (cit. on p. 112).

[322] A. Zareian, S. Karaman, and S.-F. Chang. “Bridging Knowledge Graphs to Generate Scene Graphs”. In: *European Conference on Computer Vision (ECCV)*. 2020 (cit. on pp. 83, 84).

[323] R. Zellers, M. Yatskar, S. Thomson, and Y. Choi. “Neural Motifs: Scene Graph Parsing with Global Context”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2018 (cit. on p. 83).

[324] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser. “3DMatch: Learning Local Geometric Descriptors from RGB-D Reconstructions”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2017 (cit. on pp. 59, 128, 130, 132, 134, 135).

[325] C. Zhang, Z. Cui, Y. Zhang, B. Zeng, M. Pollefeys, and S. Liu. “Holistic 3D Scene Understanding from a Single Image with Implicit Representation”. In: *arXiv preprint arXiv:2103.06422* (2021) (cit. on p. 86).

[326] J. Zhang, K. J. Shih, A. Elgammal, A. Tao, and B. Catanzaro. “Graphical Contrastive Losses for Scene Graph Parsing”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2019 (cit. on p. 84).

[327] J. Zhang, C. Zhu, L. Zheng, and K. Xu. “Fusion-Aware Point Convolution for Online Semantic 3D Scene Segmentation”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2020, pp. 4534–4543 (cit. on p. 60).

[328] W. Zhang and J. Kosecka. “Image based Localization in Urban Environments”. In: *3DIM-PVT*. IEEE, 2006 (cit. on p. 112).

[329] H. Zhao, L. Jiang, J. Jia, P. Torr, and V. Koltun. “Point Transformer”. In: *CoRR*. 2020 (cit. on p. 61).

[330] Y. Zhao and S.-c. Zhu. “Image Parsing with Stochastic Scene Grammar”. In: *International Conference on Neural Information Processing Systems*. 2011 (cit. on p. 86).

[331] Y. Zhao, T. Birdal, H. Deng, and F. Tombari. "3D Point Capsule Networks". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2019 (cit. on p. 61).

[332] S. Zhi, M. Bloesch, S. Leutenegger, and A. J. Davison. "SceneCode: Monocular Dense Semantic Reconstruction Using Learned Encoded Scene Representations". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2019 (cit. on p. 129).

[333] Zhiyuan Zhang and Binh-Son Hua and Sai-Kit Yeung. "ShellNet: Efficient Point Cloud Convolutional Neural Networks using Concentric Shells Statistics". In: *International Conference on Computer Vision (ICCV)*. 2019 (cit. on p. 62).

[334] Q. Zhou, T. Sattler, M. Pollefeys, and L. Leal-Taixe. "To Learn or Not to Learn: Visual Localization from Essential Matrices". In: *International Conference on Robotics and Automation*. 2020 (cit. on p. 112).

[335] Y. Zhou and O. Tuzel. "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection". In: *IEEE Conference on Computer Vision and Pattern Recognition* (2017) (cit. on p. 82).

[336] M. Zollhöfer, P. Stotko, A. Görlitz, C. Theobalt, M. Nießner, R. Klein, and A. Kolb. "State of the Art on 3D Reconstruction with RGB-D Cameras". In: *Computer Graphics Forum* 37 (2018) (cit. on p. 129).

[337] M. Zollhöfer, M. Nießner, S. Izadi, C. Rhemann, C. Zach, M. Fisher, C. Wu, A. Fitzgibbon, C. Loop, C. Theobalt, and M. Stamminger. "Real-time Non-rigid Reconstruction using an RGB-D Camera". In: *ACM Transactions on Graphics (TOG)* (2014) (cit. on p. 129).

[338] C. Zou, Z. Li, and D. Hoiem. "Complete 3D Scene Parsing from Single RGBD Image". In: *International Journal Computer Vision* (2019) (cit. on p. 86).

List of Figures

List of Tables

Acronyms

3DSSG	3D Semantic Scene Graphs.
AR	Augmented Reality.
BA	Bundle Adjustment.
CAD	Computer Aided Design.
DCRE	Dense Correspondence Re-Projection Error.
DoF	Degree of Freedom.
ER	Error Rate.
FCN	Fully Convolutional Network.
FCPN	Fully Convolutional Point Network.
FISS	Fully Incremental Semantic Segmentation.
FoV	Field of View.
FPR	False Positive Rate.
FPS	Frames Per Second.
GAN	Generative Adversarial Network.
GCN	Graph Convolutional Network.
GCN	Graph Neural Network.
ICP	Iterative Closest Point.
IMU	Inertial Measurement Unit.
InSeg	Incremental Segmentation.
kNN	k-Nearest Neighbor.

LSTM	Long Short-Term Memory.
MDP	Monocular Depth Prediction.
MLP	Multilayer Perceptron.
NLP	Natural Language Processing.
NSSD	Normalized Sum of Squared Differences.
RANSAC	Random Sample Consensus.
ReLu	Rectified Linear Unit.
RIO	Object Instance Re-Localization.
RNN	Recurrent Neural Network.
SGD	Stochastsic Gradient Descent.
SLAM	Simultaneous Localization and Mapping.
SoR	Surface of Revolution.
SVD	Singular Value Decomposition.
SVM	Support Vector Machine.
TSDF	(Truncated) Signed Distance Function.
VAE	Variational Auto-Encoder.
VIO	Visual Inertial Odometry.
VoL	Variance of Laplacian.
VQA	Visual Question Answering.
VR	Virtual Reality.
VRD	Visual Relationship Detection.