Technische Universität München
Fakultät für Elektrotechnik und Informationstechnik
Lehrstuhl für Medientechnik

Semantic Understanding of 3D Point Clouds of Indoor Environments

Dmytro D. Bobkov, M.Sc.

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Die Dissertation wurde am 17.01.2019 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 15.07.2019 angenommen.

Abstract

There has been an advent of affordable mobile sensors that can capture the geometry of indoor environments, such as LiDAR and depth sensors. Thanks to this, the digital revolution changes the way how indoor environments are perceived. In particular, it becomes possible to obtain a high-quality digital representation of the indoor environment, a so-called "digital twin", which can be used for a variety of applications, such as facility management, architecture, and robot navigation. These sensors typically generate 3D point cloud (PC) data as an output. The PC representation poses a number of challenges due to its irregularity and large size. In particular, such data is hard to visualize, handle, store and transmit to different clients. To address these issues and to enable new applications, it is vital to extract semantic information from the PC data, e.g., what room is this, or is this object a chair or a table.

To this end, this thesis investigates the problem of extracting semantic information from large-scale 3D PCs of indoor environments. "Large-scale" refers to the data spanning one or multiple buildings. For semantic understanding, a top-down approach is employed. Thus, the building data is first segmented into rooms, and the rooms are segmented into objects. Finally, the resulting objects are classified according to their semantic category. In particular, three scientific contributions are presented. **The first contribution** addresses the problem of segmenting the building PC data into rooms, while not making the usual assumption of a Manhattan world or requiring sensor pose information. In particular, a room segmentation algorithm using a new volumetric signature with anisotropic potential fields is presented. **The second contribution** focuses on the problem of unsupervised object segmentation using local geometric properties. For this, a novel noisy point removal step is proposed. This step is particularly robust to artifacts in real PC data and takes specific properties of large-scale PC data collected using moving LiDAR sensors into account. To illustrate such properties, a new LiDAR dataset and its semantic annotation are presented. The introduced multi-scale evaluation metric allows considering different objects scales, e.g., chair and its parts. **The third contribution** addresses the problem of object classification by using point pair features combined with a neural network. By employing 4D point pair-based feature representation as input to the neural network, it becomes possible to reliably classify objects despite high levels of noise and occlusion that are commonly observed for PC data of indoor environments. The experimental evaluation using a number of large-scale indoor PC datasets containing semantic annotation confirms that the proposed approaches to semantic understanding including room segmentation, object segmentation and classification outperform existing approaches on such data in terms of segmentation and classification accuracy.

Kurzfassung

Mittlerweile gibt es ein breites Angebot an kostengünstigen mobilen Sensoren, welche die Geometrie von Innenumgebungen genau erfassen können, beispielsweise Laserscanner und Tiefensensoren. Die digitale Revolution verändert dadurch die Art und Weise, wie Innenräume wahrgenommen werden. Insbesondere wird es möglich, eine digitale Darstellung der Umgebung von hoher Qualität, den sogenannten "digitalen Zwilling", zu erstellen. Diese kann für eine Vielzahl der Anwendungen wie Facility-Management, Architektur und Robot-Navigation verwendet werden. Diese Sensoren erzeugen typischerweise eine 3D-Punktewolke (engl. "Point cloud" oder PC) als Ausgabe. Die PC-Darstellung kommt aufgrund ihrer Unregelmäßigkeit und Größe mit einer Reihe von Herausforderungen. Insbesondere sind solche Daten schwer zu visualisieren, zu handhaben, zu speichern und an verschiedene Clients zu übertragen. Um dieses Problem zu lösen und eine Reihe von neuen Anwendungen zu ermöglichen, wird es zunehmend wichtig, semantische Informationen aus den PC-Daten zu extrahieren, z. B. um welchen Raum es sich handelt oder ob das betrachtete Objekt ein Stuhl oder ein Tisch ist.

Zu diesem Zweck wird in dieser Arbeit die Aufgabe untersucht, semantische Informationen aus Large-scale 3D PCs von Innenräumen zu extrahieren. "Large-scale" bezieht sich in diesem Zusammenhang auf Daten, die sich über mehrere Gebäude erstrecken. Für das semantische Verständnis wird ein Top-Down Ansatz verfolgt. So werden die Gebäudedaten zunächst in Räume unterteilt und anschließend die Räume in Objekte partitioniert. Schließlich werden die Objekte nach ihrer semantischen Bedeutung klassifiziert. Insbesondere werden die drei folgenden wissenschaftlichen Beiträge vorgestellt. **Der erste Beitrag** befasst sich mit der Aufgabe, die PC-Daten der Gebäude in Räume und Korridore zu unterteilen, ohne Annahmen über eine Manhattan-Welt zu machen oder Informationen über bekannte Sensorposen vorauszusetzen. Dazu wird ein Algorithmus zur Raumsegmentierung unter Verwendung einer neuen volumetrischen Signatur mit anisotropen Potentialfeldern vorgestellt. **Der zweite Beitrag** geht das Problem der unbeaufsichtigten Objektsegmentierung anhand von lokalen geometrischen Eigenschaften an. Insbesondere wird ein neues Konvexitätskriterium vorgeschlagen. Es ist besonders robust gegenüber Artefakten in realen PC-Daten und berücksichtigt die spezifischen Eigenschaften von Large-scale PC-Daten, die mittels LiDAR erfasst werden. Zur Veranschaulichung dieser Eigenschaften werden ein neuartiger LiDAR-Datensatz und seine semantische Annotation vorgestellt. Die eingeführte Multiskalen-Bewertungsmetrik ermöglicht die Berücksichtigung verschiedener Objektskalen, z. B. eines Stuhls und seiner Teile. **Der dritte Beitrag** befasst sich mit der Herausforderung der Objektklassifizierung. Hier werden Punktpaarmerkmale mit einem neuronalen Netzwerk kombiniert. Durch die Verwendung eines 4D-Punktpaar-basierten Deskriptors als Eingabe in das neuronale Netzwerk wird es möglich, Objekte trotz hoher Geräuschpegel und Verdeckungen, die üblicherweise für die PC-Daten von Innenumgebungen beobachtet werden, zuverlässig zu klassifizieren. Die experimentelle Auswertung unter Verwendung von großen PC-Innendatensätzen mit semantischer Annotationen bestätigt, dass der vorgeschlagene Ansatz zum semantischen Verständnis, einschließlich Raumsegmentierung, Objektsegmentierung und Klassifizierung, die in der Literatur existierenden Ansätze hinsichtlich der Segmentierung- und Klassifizierungsgenauigkeit übertrifft.

Acknowledgements

The work presented in this dissertation was carried out as a member of the academic staff at the Chair of Media Technology (LMT) at the Technical University of Munich. My research was in part supported by the space agency of the German Aerospace Center (DLR) with funds from the Federal Ministry of Economics and Technology on the basis of a resolution of the German Bundestag under the reference 50NA1307.

After finishing a long way of the PhD thesis, I came to realize how much support and cooperation is needed on the way in order to achieve the required goal.

First of all I would like to express my sincere gratitude to my supervisor Prof. Dr. Eckehard Steinbach for his kind support and excellent scientific guidance. He was always able to find time for fruitful discussions despite of his busy schedule. I would also like to thank Prof. Dr.-Ing. Klaus Diepold and Prof. Dr. Gerhard Rigoll for taking the time out of their busy schedule to review this thesis. I would like to thank Prof. Dr. Bernd Girod for inviting me as a research visitor in Information Systems Lab of Stanford University in spring/summer 2018. I learned a lot from this visit and enjoyed the good team.

My sincere gratitude also goes to the NavVis team, including Dr. Georg Schroth, Sebastian Hilsenbeck, Robert Huitl, Adrian Garcea, and Dr. Dominik van Opdenbosch. They truly showed scientific excellence uniquely combined with willingness to help. Moreover, my deep appreciation goes to my LMT colleagues, Dr. Tamay Aykut, Dr. Christoph Bachhuber, Dr. Rahul Chaudhari, Dr. Clemens Schuwerk, Dr. Anas Al-Nuaimi, Jingyi Xu, Matti Strese, among others. They were kind and eager to help in any situation. It was a lot of fun and I learned a lot from them. I also want to acknowledge Martin Kiechle, with whom I had many brainstorming sessions on different aspects of computer vision, which contributed to this thesis. My appreciation also goes to the professional administrative support given by Dr. Martin Maier, Marta Giunta, Martina Schmidt, Brigitte Vrochte, and Simon Krapf. They were always eager to help in any situation.

Clearly, this work would not have been possible without my students who contributed significantly to this work and assisted with implementation and evaluation. In particular, Sili Chen, Ruiqing Jian, Priyamvadha Krishnakumar, Octavio Rodriguez, Bhushan Chaudhari, Marco Falke, Tu Cheng, Saumil Patel, Jianyu Zhao, and Zirong Chen.

I also want to thank my parents who always supported and believed in me. Without their support and valuable advice I would not be where I am right now. Finally, I also would like to thank my girlfriend Graciela for support and patience.

Munich, September 28, 2019

Contents

Notation

Abbreviations

Abbreviation	Description	Definition
1D	One-Dimensional	page 94
2D	Two-Dimensional	page 17
3D	Three-Dimensional	page 1
4D	Four-Dimensional	page 90
ARI	Adjusted Rand Index	page 29
ASCII	American Standard Code for Information Interchange	page 10
CAD	Computer-Aided Design	page 31
CNN	Convolutional Neural Network	page 26
CPU	Central Processing Unit	page 9
CVFH	Clustered Viewpoint Feature Histogram	page 41
EPPF	Enhanced Point Pair Function	page 95
ESF	Ensemble of Shape Functions	page 41
GOOD	Global Orthographic Object Descriptor	page 41
GPU	Graphics Processing Unit	page 104
GRSD	Global Radius-based Surface Descriptor	page 41
GT	Ground Truth	page 72
HOG	Histogram Of Gradients	page 41
LCCP	Locally Convex Connected Patches	page 71
LiDAR	Light Detection and Ranging	page 1
MMF	Mixture of Manhattan Frames	page 51
MRF	Markov Random Field	page 20
MST	Minimum Spanning Tree	page 22
MVPC	Multi-view Point Cloud	page 3
OS	Oversegmentation	page 71
OSD	Object Segmentation Dataset	page 77
OUR-CVFH	Oriented Unique and Repeatable Clustered Viewpoint Feature Histogram	page 41
PC	Point Cloud	page 1
PCA	Principal Component Analysis	page 14
PCL	Point Cloud Library	page 2
PF	Potential Field	page 49
PFH	Point Feature Histogram	page 41
PP	Point Pair	page 90
PPF	Point Pair function	page 90
RAM	Random Access Memory	page 9
RANSAC	Random Sample Consensus	page 39
ReLU	Rectified Linear Unit	page 26
RGB	Red Green Blue	page 2

Abbreviation	Description	Definition
RGBD	Red Green Blue Depth	page 38
RRHT	Robust Randomized Hough Transform	page 17
SFM	Structure From Motion	page 5
SHOT	Signature of Histograms of Orientations	page 41
SLAM	Simultaneous Localization and Mapping	page 5
US	Undersegmentation	page 71
VFH	Viewpoint Feature Histogram	page 41
WO	Weighted Overlap	page 29

Mathematical notation

x	scalar
$\mathbf{x}$	vector
$\boldsymbol{X}$	matrix
$\lvert x \rvert$	absolute value of scalar x
$\lVert \mathbf{x} \rVert_2$	Euclidean (L2) norm of vector $\mathbf{x}$
$\mathbf{x} \cdot \mathbf{y}$	dot product of $\mathbf{x}$ and $\mathbf{y}$
p_i	ith component of vector $\mathbf{p}$
$\mathbb{P}$	set
$\lvert \mathbb{P} \rvert$	size of set $\mathbb{P}$
$\mathbb{R}^n$	Euclidean space of dimensionality n
$\mathbb{N}(\mathbf{p_i})$	Points that lie in the neighborhood of point $\mathbf{p_i}$
$\mathcal{G}$	Graph
$\overline{x}$	mean of random variable x
$\overline{\mathbf{x}}$	mean vector
$\hat{x}$	estimated/predicted value of x

Chapter 1

Introduction

1.1 Motivation

Nowadays, humanity experiences the era of digital revolution. More and more aspects of human lives become digitized, i.e., transformed into a digital representation. This fact became possible due to the development of affordable mobile sensors that can capture the appearance and geometry of the environment, such as laser scanners and depth sensors. As humans spend most of their lives in indoor environments, it becomes especially useful to obtain an accurate digital representation of an indoor environment, a so-called “digital twin”. As indoor environments can have very complex 3D geometry counting multiple floors and various 3D objects, a suitable digital representation should be able to reflect all these aspects accurately. Out of possible 3D representations, a point cloud (PC) is closest to the sensor output and does not require additional complex processing steps, like voxelization or meshing. PC is a set of points usually describing occupied space. Each point can have a number of attributes, the most important ones are X, Y and Z coordinates in a particular coordinate system. Such representation can be obtained using various sensors: an RGB camera that captures an image sequence in a specific trajectory combined with a structure from motion technique, a depth sensor or light detection and ranging (LiDAR) sensor.

While a PC is a powerful representation, it is not designed for human interaction. A typical PC of a building contains millions of points and can easily require hundreds of megabytes of storage and even more (see Figure 1.1). Such data is hard to visualize, handle, store and transmit to different clients. Moreover, it cannot directly help humans in their everyday life, e.g., answering questions like “where is the next door, or where is the closest bathroom”. For robotic systems, this issue becomes even more challenging as robotic agents need to make specific decisions and cannot directly handle sensor data. To address this problem, the extraction of semantic information from the PC data is required. Here, semantic information refers to information that primarily describes the meaning of a certain object, e.g., is it a table, door, room. Such information is directly understandable to humans or robotic agents and is critical for various high-level tasks like navigation, object grasping, and others. While indoor environments can have rich semantic content on various levels, the focus of this work is on the level of objects and rooms of the buildings. The term “indoor environments” in this work refers to human-made buildings and similar structures, e.g., residential houses, universities,

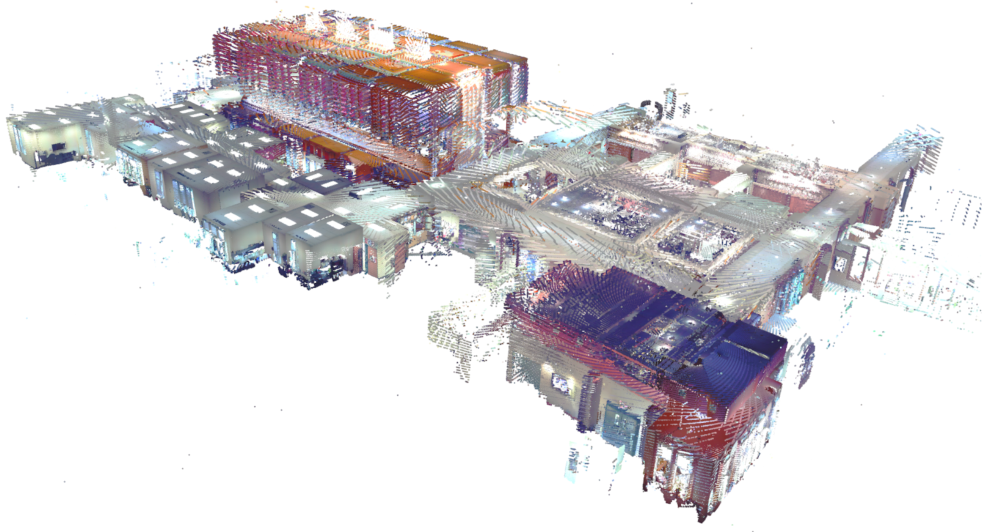


Figure 1.1: Illustration of a PC of an indoor environment counting several floors and over 21 million points (when using PCD format of Point Cloud Library). Only a very modern computer can visualize such PC at the original point resolution without running out of memory. Dataset from [10].

offices and similar.

In order to extract semantic information in indoor environments, a top-down approach is employed, where, in the beginning, the PC data of an entire building is considered (see Fig. 1.2). In the first step, the building PC data (A) is partitioned into the first level of semantic entities, i.e., rooms (B). By splitting the building into rooms, there is an advantage of being able to process each room in parallel in the consecutive processing steps, without any deterioration in segmentation performance. After the rooms have been identified, the PC of every room is partitioned into semantic elements, such as objects, including chairs, tables, and other everyday indoor objects. While PC data contains a significant amount of information, it is also subject to substantial levels of noise. In particular, typical PC data obtained with RGB cameras or laser scanners suffers from sensor noise, subsampling, occlusion, and other artifacts. Such effects are typically unavoidable as the data needs to be captured with affordable sensors in limited time.

Within this work, only unorganized PCs are considered, where the point index within the point set is not related to the spatial relationship. This is done for the sake of generalization, as organized PCs (e.g., depth images) can be considered as a particular case of unorganized PCs. The term "unorganized PC" was first introduced in the Point Cloud Library (PCL) [12] and it typically applies to all data obtained with laser scanners. This is in contrast to depth images that have a regular image-like structure, where the neighbors of each pixel can be determined using a predefined lookup table. Thus, a PC is an unordered set of points having specific attributes. Clearly, dealing with unorganized PCs is more challenging, as such a simple problem as determining spatial neighbors of a particular point becomes a computationally complex task with the worst case time complexity $O(N)$ (for a kd-tree with unbalanced branches), where N is the number of points in the PC [13]. The more difficult tasks, consequently, require larger computational complexity. Furthermore, even though points can also

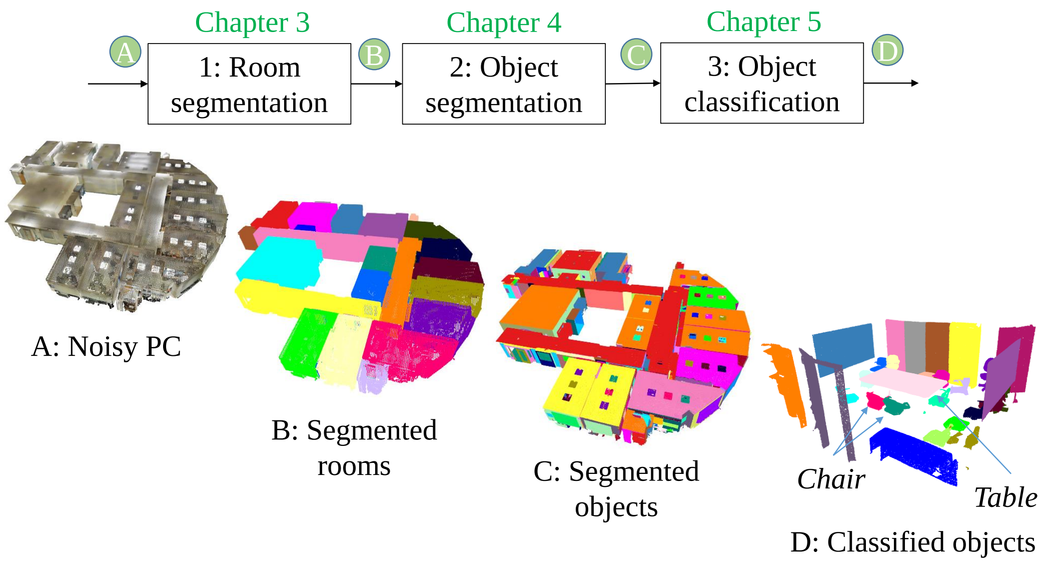


Figure 1.2: Illustration of the processing steps in the semantic understanding pipeline. Results of every step of the pipeline shown in the circled letters are illustrated in the bottom. As input, a noisy PC is used. Given the noisy PC, as a result of the room segmentation step, segmented rooms are obtained (B). The room segments are given to the object segmentation step, which produces object segments (C). This result is provided to the object classification step that performs object classification (D). The output of the pipeline is PC segments and their labels (e.g., a chair or table). Dataset of [11].

have color information, the focus of this thesis is on geometry-based methods. Hence, RGB information is not used by any of the presented methods. This allows for larger applicability of the given methods as it typically requires significant effort to obtain RGB information that is accurately registered to a 3D PC.

1.2 Major Contributions

The goal of this thesis is to address the challenges in semantic understanding in 3D PC data of indoor environments that is subject to various measurement artifacts. To this end, the thesis describes the following scientific contributions:

1. A novel unsupervised approach to **room segmentation** in 3D PCs of indoor environments using anisotropic potential fields is proposed, see step 1 in Fig. 1.2 and Chapter 3. In related work, it was typically assumed that the indoor environment follows Manhattan assumptions, i.e., vertical walls oriented at right angles to each other [14], [11]. Other works assumed a low level of noise [15] or availability of a sensor trajectory [16]. In contrast, the proposed solution does not make such assumptions and works for any room layouts. This solution also shows superior performance for PCs with high levels of noise and occlusion, which is typical for the PC data collected in real-world scenarios using affordable sensors subject to time constraints.

2. A new unsupervised approach to **object segmentation** in 3D PCs using a concavity-convexity criterion is suggested, see step 2 in Fig. 1.2 and Chapter 4. It performs well on multi-view point cloud (MVPC) data. Here, MVPC data refers to the data obtained from multiple viewpoints using simultaneous localization and mapping (SLAM) or a structure from motion (SFM) system. Previously available concavity/convexity criteria exhibit inferior performance on such data [17], as they do not take important effects of MVPC into account, such as high levels of noise and occlusion. A realistic indoor MVPC dataset counting more than 500 object parts is collected and annotated to illustrate these effects accurately. Furthermore, a new multi-scale metric for object segmentation evaluation is proposed.

3. A novel approach to **object classification using point pair features** in 3D PCs in combination with deep learning is presented, see step 3 in Fig. 1.2 and Chapter 5. Typical PC data has high levels of noise, and this leads to the fact that point-based object classification approaches achieve inferior performance as compared to synthetic data [18]. To mitigate this issue, a novel 4D point pair descriptor is proposed. Furthermore, it is shown how 4D point pair descriptors can be combined with deep learning techniques to obtain a superior object classification accuracy.

1.3 Thesis Organization

This thesis is organized as follows. First, an overview of the background and related work is given in Chapter 2. Here, the typical data processing pipeline is described, and background knowledge is explained. Furthermore, a review of related work on room segmentation, object segmentation, and object classification is provided. The task of room segmentation and the proposed methodology to address this problem are discussed in Chapter 3. After that, the task of object segmentation and the suggested object segmentation method are described in Chapter 4. Here, also the new laser scanner dataset and the proposed evaluation metric are discussed. In Chapter 5 the problem of object classification using point pair features is described in detail. Chapter 6 concludes with the results, provides limitations of the presented work and gives an outlook for future work.

Parts of the work, presented in this thesis, have been published in international peer-reviewed journals [1] and conferences [4], [5], [6]. Outside of the scope of this thesis, a number of publications on the topic of data fusion, SLAM and indoor navigation have been presented in [7], [8], [2], [9], [3].

Chapter 2

Background and Related Work

This chapter establishes the basic terms and background knowledge that are used throughout this thesis. Furthermore, the typical data processing pipeline is described. Finally, a review of related work on room segmentation, object segmentation, and object classification is given.

2.1 Background

At first, the steps of data acquisition and preprocessing are discussed as these are the essential steps to obtain a suitable data representation. After that, a short overview of the relevant segmentation algorithms, convolutional neural networks, and evaluation metrics is given.

2.1.1 PC Data Acquisition and Representation

Mapping is the process of creating a digital map of the environment. This process is most often done in the context of the simultaneous localization and mapping (SLAM) problem [2]. Here, a sensor is simultaneously localized within the map that is being built at the same time. Out of different mapping solutions, laser scanner-based mapping systems can provide the highest density of the data in a short time when mapping large-scale indoor environments. In contrast to multi-view structure from motion (SFM) reconstruction pipelines [19], the laser scanners are less sensitive to lighting variations and can provide very high point density even in low-texture environments assuming a sufficiently high sampling rate of the laser scanner. Compared to Kinect-based solutions[1], laser scanners have a clear advantage as they provide a larger scanning range (typically more than 30 meters) and wider angle of view. With such a system, it is possible to scan an area of ten thousand square meters within a day, which is practically impossible using any Kinect-like sensor.

Various laser scanner-based mapping systems exist, such as backpack-based [20] or trolley-based [21]. The sensors progressively take measurements and integrate them into a 3D model using a SLAM system, while the sensor platform is moved through the indoor space.

[1] Matterport sensor https://matterport.com/. Accessed: 2018-12-16.

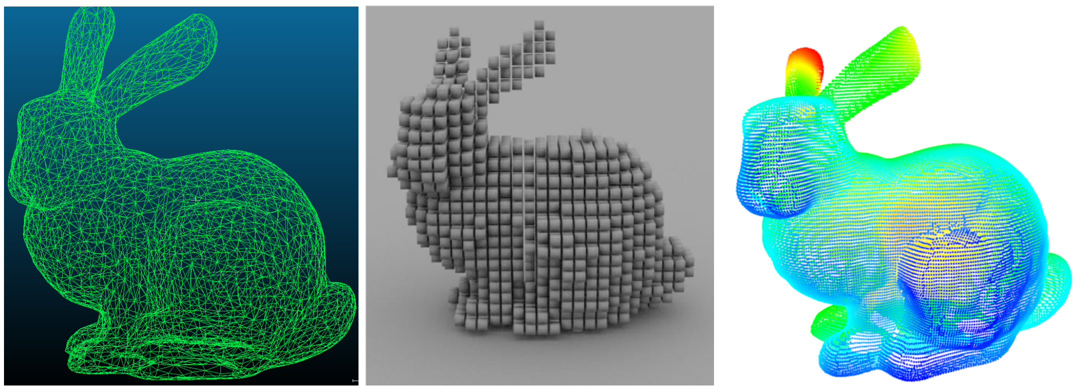

(a) Mesh. (b) Volumetric grid. (c) Point cloud.

Figure 2.1: Illustration of different 3D representations of the Stanford bunny. While the underlying geometry is the same, the three representations have different properties and require different algorithms for processing. Data from the Stanford 3D scanning repository[2].

The mapping result is the environment map that can be represented in different forms, such as point cloud, mesh or grid-based representation. All three **representations** are commonly used in different areas of data processing and have their advantages:

- **Mesh** is a boundary representation of objects using geometric polygonal primitives (see Fig. 2.1a). Essentially, meshes are approximations of smooth surfaces of objects, where the faces (polygons) contain vertices that are connected using edges. Within the mesh, connectivity information between different meshes is stored by defining common edges to different faces. This way, it is possible to encode planar area of the object boundary using a single primitive requiring a limited number of parameters given as vertex coordinates. Furthermore, it is easy to fulfill the requirements on the regularity of the representation by splitting larger primitives into smaller ones. Mesh faces can also have RGB and other information. Normal vector orientation can be easily computed from the face vertex coordinates. Mesh representation has the following advantages: it requires lower computational complexity for rendering (most rendering engines, nowadays, support primitive-based meshes on a hardware level), lower storage and transmission requirements. Furthermore, various operations such as neighbor search, visibility calculation and surface normal computation can be significantly speeded-up. The main disadvantage of this representation is that it is challenging to obtain a high-quality mesh from noisy sensor data.

- **Volumetric grid** is a regular 3D grid representation, where each grid element (voxel) can be "occupied" if there is occupied space inside or "free" if space inside is empty, see Fig. 2.1b. In addition to occupancy information, volumetric grid elements can have other attributes, such as color and occupancy probability. The grid elements can be obtained by thresholding the signed distance function obtained from Kinect-based reconstruction approaches [22] or from the scanned points directly. Volumetric grids have

[2] https://graphics.stanford.edu/data/3Dscanrep/. Accessed: 2018-10-02.

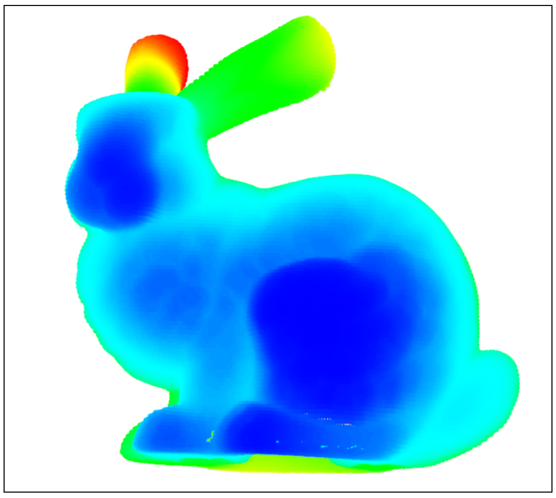
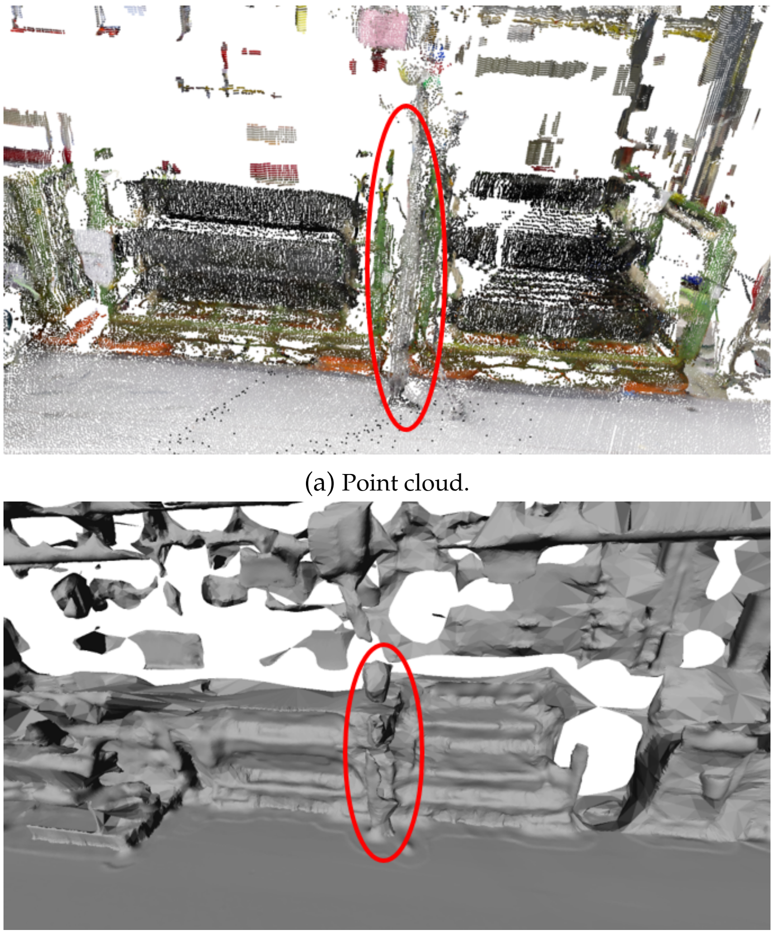


Figure 2.2: Illustration of a depth image for the Stanford bunny. Depth values depicting closer located parts of the scene are shown in blue, whereas further located parts are shown in red. A single depth image can only depict a limited part of a 3D scene and does not allow to consider the entire 3D geometry at once. Data from Stanford 3D scanning repository.

an advantage of having a regular structure, which makes spatial neighborhood computation, visibility check, and other operations significantly faster. Their disadvantage lies in the large storage requirements: all voxels within a certain volume need to be stored. Thus, a large number of the voxels correspond to empty space, hence carry no information. Furthermore, the same voxel size is used in the entire grid, which results in uniform spatial resolution even though the actual object geometry can have different levels of detail in distinct object parts.

- **Point cloud** is an irregular 3D representation containing a set of points corresponding to occupied space. Each point can have a number of attributes, in the simplest case X, Y and Z coordinates given in a certain coordinate system (see Fig. 2.1c). The PC is an unordered set, which means that by shuffling the points within the PC, e.g., changing their order, the described geometry remains the same. Such representation typically corresponds to the output of a laser scanner and multi-view SFM-based reconstruction approaches. PC has an advantage that it directly corresponds to the sensor output. Thus, no data conversion is required once the data arrives from the sensor and the data can be directly used for further tasks. Furthermore, no information loss occurs when the data is stored in this form. The main disadvantage of PCs is that various operations such as point neighbor search or visibility calculation require high complexity using, e.g., kd-tree or other spatial structures. Furthermore, it is hard to process the PCs in many machine learning algorithms due to their inherent data irregularity.

Furthermore, depth images are also commonly used as a pseudo-3D representation. In a depth image, which is a 2D representation, each pixel has a floating-point value that stores the distance from the camera center to the corresponding part of the 3D scene visible at this pixel. An illustration of a depth image is shown in Fig. 2.2. A single depth image can only depict a limited part of a 3D scene, e.g., the back side of the bunny is occluded by its


(a) Point cloud.

(b) Mesh reconstruction result.

Figure 2.3: Illustration of the real sensor output in the form of a PC (a) and the corresponding result of the mesh reconstruction [23] (b). It can be observed that for the areas with low point sampling density (encircled region in (a)), the resulting mesh cannot accurately represent the underlying geometry (encircled region in (b)). Many erroneous faces can be observed in the encircled area. This issue is particularly pronounced for thin and long structures, which are commonly present in indoor environments.

front part. Therefore, this representation does not allow to consider the entire 3D geometry at once. By having an image-like structure, this representation offers lower computational complexity. Furthermore, many of the methods that have been developed for RGB images can be applied to depth images without the need for major modification.

Out of the above mentioned 3D representations, mesh offers the advantage for processing and recognition, but it remains a challenging research problem on how to convert real sensor output in the form of a PC to a mesh representation. This area of research is called surface reconstruction [23]. In particular, it is challenging to obtain the mesh representation without introducing significant artifacts (wrong vertices and false connectivity) from the PC data. The noise and holes are often present in real sensor output, see Fig. 2.3a. These effects inevitably result in the artifacts in the corresponding mesh, as shown in Fig. 2.3b. Here it is possible to observe that for the areas with low point sampling density (encircled region), the resulting mesh cannot accurately represent the underlying geometry. Furthermore, many erroneous faces can be observed. This issue is particularly pronounced for thin and long structures, as shown in Fig. 2.3b.

Of the above-mentioned representations, using a volumetric grid for semantic understanding poses significant challenges due to high computational complexity requirements. For example, to store a volumetric grid for the PC shown in Fig. 1.1 at the voxel resolution of 2 cm (sufficient to describe fine object geometry), ca. 14 Gigabytes of storage are needed. This fact means that it is not feasible to keep the entire PC in random access memory (RAM) of a typical desktop computer (e.g., Quad-core i7 central processing unit (CPU) with 8 GB of RAM) for fast processing, not speaking of mobile devices with even lower amounts of available memory. Due to the mentioned limitations of other representations, PC is used as a suitable representation for processing and semantic understanding. PC data considered in this work is usually obtained from laser scanners, such as Hokuyo[3], Velodyne[4] or Kinect-like sensors[5].

2.1.2 PC Data Description

PC data can have various **attributes**:

- X, Y and Z coordinates of points are provided in a certain coordinate system, typically given in meters. These attributes are obligatory and are always present in the PC data.
- R, G and B color information for each point, typically given as integer scalar values in the range $0..255$ (optional).
- Surface normal vectors are given in the form of a 3D vector (optional).
- Surface curvature value is provided in the form of a floating-point scalar value (optional).
- Point labels are given in the form of an integer indicating to which segment a certain point belongs (optional).
- Laser ray intensity values are measured after the ray has been sent and received. This only applies to the data measured with LiDAR sensors. It is typically a floating-point scalar value (optional).

PC data can be stored in different **file formats**:

- PCD (Point Cloud Data) is a format that was introduced in the Point Cloud Library (PCL) [12]. This file format typically contains a header that describes the size of the PC

[3] Hokuyo laser scanner specification https://www.hokuyo-aut.jp/search/single.php?serial=169. Accessed: 2018-12-20.

[4] Velodyne laser scanner https://velodynelidar.com/. Accessed: 2018-12-21.

[5] Kinect-like scanner specification https://www.asus.com/3D-Sensor/Xtion_PRO/specifications/. Accessed: 2019-09-12

and the present attributes. It is supported by PCL, CloudCompare[6] and other software libraries.

- PLY is a data format that is commonly used in different areas of robotics, semantic understanding, indoor reconstruction, architecture, and indoor modeling. Similarly to PCD, it also contains a header that describes the PC attributes. It supports PC and mesh representations. This file format is supported by PCL, MeshLab and other standard software tools.
- American Standard Code for Information Interchange (ASCII) is a human-readable format where the point attributes are stored as plain text. This file format does not offer any compression but can be easily inspected by humans. Furthermore, loading and writing can also be done in a straightforward way.
- LAS is an industry-standard binary format for storing airborne LiDAR data. This format is not common for storing the PC data of indoor environments.

In the area of semantic understanding, robotics and computer vision, the most common file formats for PC data are PCD and PLY. No common file format is used throughout the different communities, e.g., architecture and robotics. This is in contrast to image formats, where JPEG and PNG are widely adopted.

2.1.3 Basic Terms

Here, the basic terms are clarified that are commonly used in this thesis, in particular, *segmentation*, *clustering*, *semantic segmentation*, *instance segmentation*, *labeling*, *classification*, and *unsupervised algorithm*. It has been observed that in the literature some of these terms are used interchangeably without paying particular attention to their meaning. Due to this shortcoming, there exists a need to formally define these terms in this work.

In the context of this thesis, i.e., in 3D computer vision, *segmentation* has the primary goal of assigning the points to a number of segments, without providing any knowledge what each segment represents. *Clustering* is an equivalent concept, where each of the points is assigned to one of the clusters. In contrast, *semantic segmentation* has the goal of assigning each of the points to one of the semantic categories. Here, distinct instances of the same category will be assigned to the same label as no distinction between instances is typically made, e.g., two different chairs have the same assignment. In contrast, *instance segmentation* assigns each of the points of different objects to a distinct instance. Thus, two points belonging to different chairs will be assigned to distinct labels even though they still represent chairs. *Classification* has the goal of assigning a given segment to one of the labels, e.g., chair or table. It typically assumes that the segmentation has been done beforehand. Finally, *labeling* means the assignment of each segment (or point herein) to a specific label. Labeling is often used for either semantic segmentation, instance segmentation or classification, depending on the context. *Unsupervised algorithm* (in the context of machine learning) refers to the algorithm that does not require annotated data.

[6] CloudCompare software framework http://www.cloudcompare.org/. Accessed: 2018-12-20

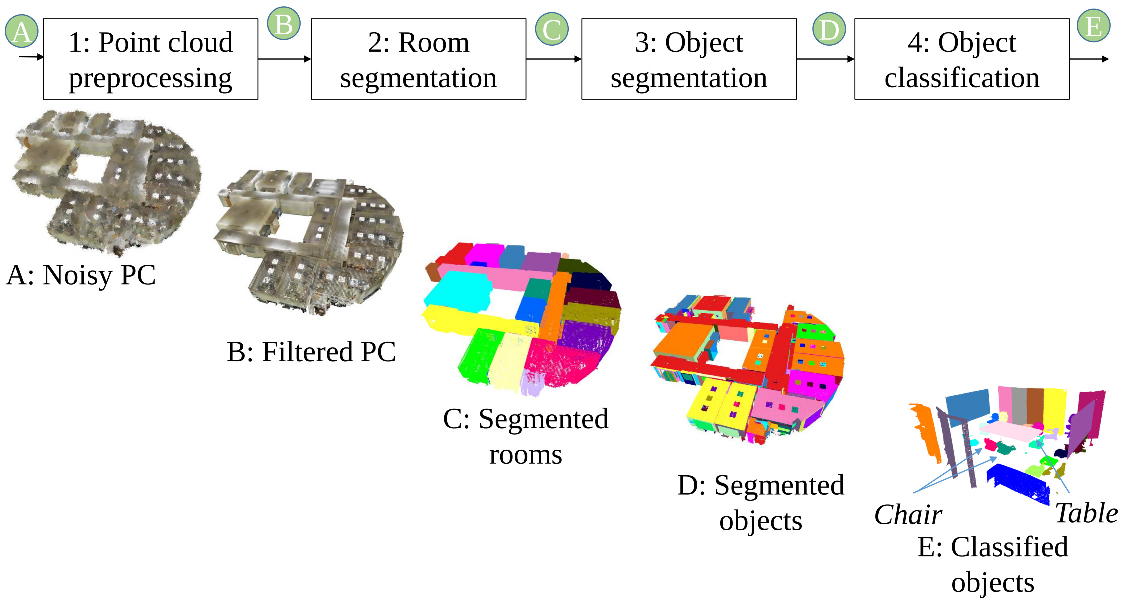


Figure 2.4: Processing steps in the semantic understanding pipeline. The result of every step highlighted with the circled letter in the top is shown below. As input (A), a noisy PC is used. Within the PC preprocessing step, noise and outliers are removed. Furthermore, surface normal directions and principal surface curvatures are estimated. When providing a filtered PC (B) into the room segmentation step, the segmented rooms are obtained (C). In the following, the PCs of the rooms are given to the object segmentation step, which generates segmented objects (D). The object classification step performs classification of the segments into objects (E). The output of the pipeline are points corresponding to the object segments and their semantic labels (such as a chair or a table).

2.1.4 PC Pipeline Overview

An overview of the typical semantic understanding processing pipeline is given in Fig. 2.4. A noisy PC is used as input (A). For this purpose, typically multiple sensor scans that are spatially registered to each other are used. They can originate from a LiDAR sensor or other SLAM or SFM-based systems. Due to estimation inaccuracies within the SLAM or SFM process, noise and registration artifacts are introduced in the PC data. Within the PC preprocessing step, the noise and outliers are partially removed. Furthermore, a local surface is estimated, in particular, surface normal direction and the principal surface curvature. When providing a filtered PC (B) to the room segmentation step, room segments are obtained (C). In the following step, the PCs of the rooms are given to the object segmentation step, which generates a segmentation of objects (D). This result is provided to the object classification step that performs classification of the objects (E). The output of the pipeline is point segments representing objects and their semantic labels (such as chair or table).

2.1.5 PC Preprocessing

The typical sensor output is the noisy PC data, which contains points influenced by sensor noise, registration artifacts, and reflection effects. The reflection effects happen when scanning reflective surfaces. There are so-called "ghost points" that correspond to non-existing geometry at this part of space. This noise does not only influence the point coordinates but also affects the later processing steps, such as surface estimation. Surface information is heavily used by many object recognition and semantic understanding algorithms. To obtain surface information, PC preprocessing is performed before the following steps of segmentation and recognition. In this step, the PC data is filtered, and the local surface is estimated. The filtering step is essential to remove the noisy points present in the scan as these points would deteriorate the performance of recognition algorithms.

Before the PC filtering, the concept of the point neighborhood is described. To determine neighbors of the point $\mathbf{p}$ from the PC, a straightforward solution requires iterating over all N points in the PC represented as set $\mathbb{P}$, which would result in the time complexity $\mathcal{O}(N)$. This is not acceptable as the number of points can reach millions. Therefore, spatial search structures are usually employed for this purpose instead, such as octree or kd-tree [13]. These structures, essentially, compute tree-based representations, where a particular part of the tree corresponds to the specific part of space. Thus, to determine neighbors of the point $\mathbf{p}$ it is needed to only propagate towards a certain branch of the tree. In case the tree is balanced, the average time complexity is $\mathcal{O}(\log_{BF} N)$, where BF is the branching factor of the tree. Of the two spatial structures, the kd-tree recursively splits the space along the direction of the largest variance with a branching factor of 2. In contrast, the octree uses the branching factor of 8. The octree has a specific advantage over the kd-tree, i.e., insertion and deletion operations have lower complexity. In contrast, kd-trees have lower worst time complexity for neighbor query [24].

2.1.5.1 PC Filtering

For PC data filtering, an analysis of the point neighborhood is performed in order to remove isolated points that result from reflections and sensor noise. For this task, a number of approaches exist. One of the most efficient and simple approaches is statistical analysis [25]. In particular, for each point $\mathbf{p_i} \in \mathbb{P}$, the mean distance $\overline{d_i}$ to its k closest neighbors is computed. It is formally defined as follows:

$$\overline{d_i} = \frac{1}{|\mathbb{N}(\mathbf{p_i})|} \cdot \sum_{\mathbf{p_j} \in \mathbb{N}(\mathbf{p_i})} \|\mathbf{p_i} - \mathbf{p_j}\|_2, \tag{2.1}$$

where $\mathbb{N}(\mathbf{p_i})$ is the neighborhood of point $\mathbf{p_i}$ containing k closest points. Radius-based neighborhood of point $\mathbf{p_i}$ with radius value of R is defined as follows:

$$\mathbb{N}(\mathbf{p_i}) = \{\mathbf{p_j} \in \mathbb{P}, \text{ where } \|\mathbf{p_i} - \mathbf{p_j}\|_2 \leq R\}. \tag{2.2}$$

Based on the mean distance for each point, the distribution of distances over the entire PC is computed. From this distribution, parameters of the normal distribution, such as mean μ_k and standard deviation σ_k can be estimated. The points for which the mean distance value to its neighbors $\overline{d_i}$ does not deviate from the corresponding value for the rest of the points within a certain threshold are preserved. The points not satisfying this condition are removed from the PC. This step is typically combined with a simple thresholding step. Thus, the points that do not have at least a certain number of points in their radius neighborhood are removed. This technique helps to remove isolated clusters caused by reflections and other measurement artifacts.

2.1.5.2 Surface Normal Estimation

Once the neighboring points of the query point $\mathbf{p_i}$ are filtered and the number of outliers and noisy points is substantially reduced, the following steps of surface normal direction estimation $\mathbf{n_i}$ can be performed. The normal information is one of the essential properties of the surface, as it abstracts the surface properties, such as principal directions of a tangential plane to the surface, into a single 3D vector. More details on the tangential place are given in the next paragraph. The normal vector information is commonly used to identify boundaries between different objects, compute the surface illumination for correct shading, and other tasks in recognition and computer graphics. Once information about tangential plane of the geometric surface is available, it is usually straightforward to infer the direction of the normal at a certain point on the surface. In such case, the normal direction is usually defined as the vector perpendicular to the surface at that point. Estimation of the surface properties based on the PC data is a challenging task. Two solutions to this problem exist. The first solution is to reconstruct the mesh using mesh reconstruction techniques and leverage the obtained face orientation to determine the normal vector direction. This approach has the disadvantage that the required step of mesh reconstruction is at least equally challenging (see Fig. 2.3b). Instead, a second approach to this problem is to estimate the surface normals from the PC data directly by analyzing the point neighborhood. This approach is typically more straightforward as compared to the first one. It is described in the following.

Surface estimation using principal component analysis. Determining the normal to the point $\mathbf{p_i}$ on the surface can be approximated by estimating the normal of the plane K tangential to the surface S [26]. The surface estimation task can be postulated as a least-squares plane fitting problem in the point neighborhood $\mathbb{N}(\mathbf{p_i})$, see Fig. 2.5. In particular, let us assume that there is a local surface S consisting of a set of points and a curve L lies in this surface. A right-handed orthogonal frame at point $\mathbf{p_i}$ is commonly called the Darboux frame and is given as $E = (\mathbf{e_1}, \mathbf{e_2}, \mathbf{n_i})$ [27]. Principal curvature vectors of the surface are given as $\mathbf{e_1}$ and $\mathbf{e_2}$. The above mentioned tangential plane K is defined as a plane that goes through yet unknown point $\mathbf{x_i}$ with a normal vector $\mathbf{n_i}$. The distance from point $\mathbf{p_i}$ to the tangential plane can be defined as $d = (\mathbf{p_i} - \mathbf{x_i}) \cdot \mathbf{n_i}$. The values of $\mathbf{x_i}$ and $\mathbf{n_i}$ are computed in a least-squares sense so that $d = 0$ [28]. The point $\mathbf{x_i}$ is equal to centroid $\overline{\mathbf{p}}_i$ of the point neighborhood $\mathbb{N}(\mathbf{p_i})$ and the latter is computed as follows:

$$\mathbf{x_i} = \overline{\mathbf{p}}_i = \frac{1}{|\mathbb{N}(\mathbf{p_i})|} \cdot \sum_{\mathbf{p_j} \in \mathbb{N}(\mathbf{p_i})} \mathbf{p_j}. \tag{2.3}$$

The solution for $\mathbf{n_i}$ is found by performing the principal component analysis (PCA), i.e., considering the eigenvalues and eigenvectors of the covariance matrix $\boldsymbol{C}(\mathbf{p_i}) \in \mathbb{R}^{3\times3}$. The

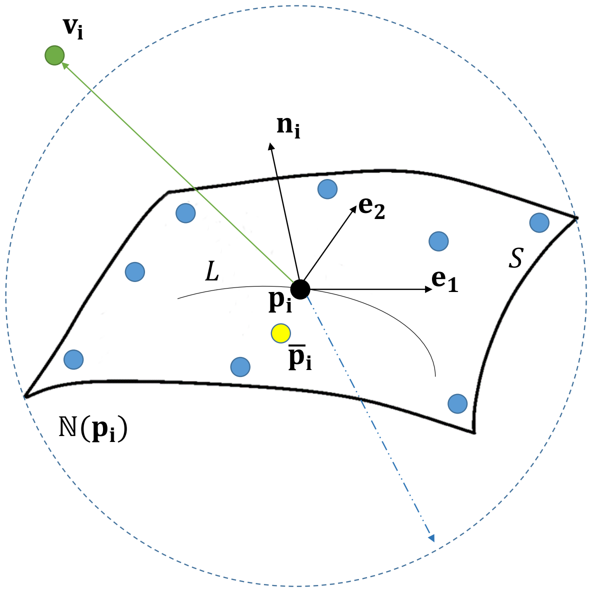


Figure 2.5: Illustration of the Darboux frame $E = (\mathbf{e_1}, \mathbf{e_2}, \mathbf{n_i})$ that is a right-handed orthogonal frame at point $\mathbf{p_i}$. Surface S consists of the points neighboring to $\mathbf{p_i}$ given as $\mathbb{N}(\mathbf{p_i})$ and shown in blue. L is an oriented curve that lies in S. Principal curvature vectors of the surface are given as $\mathbf{e_1}$ and $\mathbf{e_2}$. The normal direction to the tangential plane defined by E at the given point is shown as $\mathbf{n_i}$. The point from which the given point $\mathbf{p_i}$ has been observed is shown in green. The centroid of the point neighborhood $\overline{\mathbf{p_i}}$ is shown in yellow. The corresponding vector from the given point to the observation point is called viewpoint vector $\mathbf{v_i}$.

covariance matrix is formally expressed as follows:

$$\boldsymbol{C}(\mathbf{p_i}) = \frac{1}{|\mathbb{N}(\mathbf{p_i})|} \sum_{\mathbf{p_j} \in \mathbb{N}(\mathbf{p_i})} \psi_j \cdot (\mathbf{p_j} - \overline{\mathbf{p_i}}) \cdot (\mathbf{p_j} - \overline{\mathbf{p_i}})^T, \tag{2.4}$$

where the term ψ_j represents a weighting factor for $\mathbf{p_j}$. This term usually equals 1 as all points are considered equally important when computing neighborhood statistics. The eigenvector equation is defined as follows:

$$\boldsymbol{C}(\mathbf{p_i}) \cdot \mathbf{v_l} = \lambda_l \cdot \mathbf{v_l},\ l \in \{0, 1, 2\}, \tag{2.5}$$

where $\boldsymbol{C}(\mathbf{p_i})$ is symmetric and positive semi-definite, and its eigenvalues are real numbers $\lambda_l \in \mathbb{R}$. The eigenvectors $\mathbf{v_l}$ form an orthogonal frame, corresponding to the principal components of $\mathbb{N}(\mathbf{p_i})$. If $0 \leq \lambda_0 \leq \lambda_1 \leq \lambda_2$, the eigenvector $\mathbf{v}_0$ corresponding to the smallest eigenvalue λ_0 is, therefore, the approximation of $+\mathbf{n_i} = (n_x, n_y, n_z)^T$ or $-\mathbf{n_i}$.

In general, both solutions $\mathbf{n_i}$ and $-\mathbf{n_i}$ to Equation 2.5 are valid. Hence the normal orientation computed using the aforementioned method of PCA is ambiguous. This can lead to the fact that the normal vectors are inconsistently oriented over the PC set, resulting in sudden changes in the directions of the normal orientation from one point to the other without a significant change in geometry. This effect is undesirable and presents a challenge to many recognition algorithms. In case a point $\mathbf{o_i}$ from which the given point $\mathbf{p}_i \in \mathbb{P}$ has been observed is known (as a by-product result of SLAM or SFM procedure), it is possible to compute viewpoint vector as $\mathbf{v_i} = \mathbf{o_i} - \mathbf{p_i}$. Hence, a straightforward solution to the problem of determining a sign of orientation exists. For this, all normals need to satisfy the condition:

$$\mathbf{n_i} \cdot (\mathbf{v_i} - \mathbf{p_i}) \geq 0. \tag{2.6}$$

For the normals that do not satisfy the given condition, the normal direction vector $\mathbf{n_i}$ needs to be inverted, i.e., multiplied by -1. In case this information about the viewpoint is not available, it is possible to formulate the problem of normal consistency modeling as a graph optimization problem [29]. The main idea is to consider that the two data points $\mathbf{p_k}$ and $\mathbf{p_j}$ that belong to a smooth surface typically have their normal directions consistently oriented so that:

$$\mathbf{n_k} \cdot \mathbf{n_j} \approx 1, \tag{2.7}$$

where $\mathbf{n_k}$ and $\mathbf{n_j}$ are the normal vectors corresponding to these points. This assumption holds for densely sampled PC datasets, which is the case for PC data collected with laser scanners. This way, each point is modeled as a node in the graph with edge costs being set to Euclidean distances between the neighboring points. By performing graph cut on the given graph using a binary graph cut formulation (see Section 2.1.6.1 for more details), it is possible to obtain the binary label for each point that indicates whether the corresponding normal direction vector has to be inverted or preserved.

An important result from plane fitting is the curvature value, which is a measure for how significantly this surface geometrically deviates from the plane. It can be approximated [30] by the ratio of the smallest eigenvalue to the sum of the three eigenvalues obtained from the covariance matrix $\boldsymbol{C}(\mathbf{p_i})$ as follows:

$$\sigma = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2}, \tag{2.8}$$

where $0 \leq \lambda_0 \leq \lambda_1 \leq \lambda_2$. This measure has an important property: invariance to scaling, translation, and rotation. This means that the curvature of the region does not change if the region is scaled and points are pushed apart or closer together. It also does not change if the region is rotated or translated. The surface curvature lies in the range $\sigma = [0, 1/3]$, and it is equal to 0 if the surface is planar, which results in the fact that the covariance matrix $\boldsymbol{C}(\mathbf{p_i})$ has rank 2. The surface curvature achieves the maximum value of $1/3$ for the case of isotropic point sets, as in this case $\lambda_0 = \lambda_1 = \lambda_2$. Isotropic point set is a point set where the points are uniformly distributed within the volume of a sphere.

An important issue for normal estimation is the definition of the point neighborhood. One option is to use neighborhood search of k nearest neighbors. In practice, however, using a fixed number of nearest neighbors could result in the effect that the point neighbors located far away from the given point $\mathbf{p}$ are used, which would result in the fact that the computed statistics are not representative. This happens for PCs having areas with low point sampling density. Due to these effects, fixed radius search is advantageous to other techniques. This approach has, however, an important effect that needs to be considered when dealing with

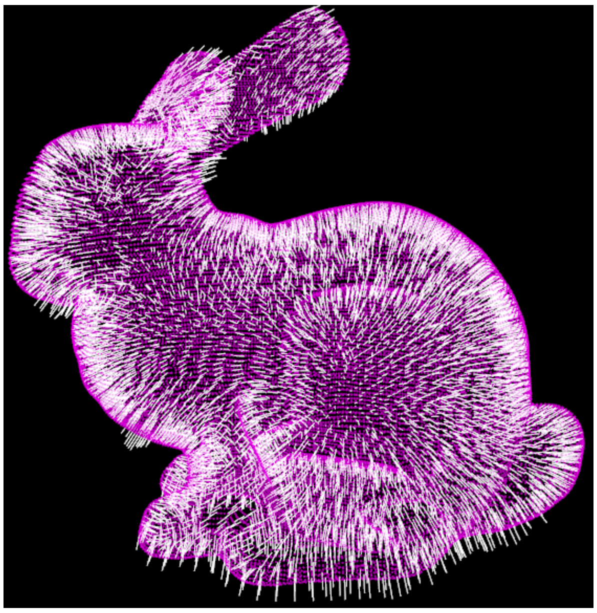


Figure 2.6: Illustration of the normal direction estimation result for the PC data of bunny from Fig. 2.1. The white arrows originating at the corresponding points of the PC indicate normal directions. It can be observed that the normal directions follow the local surface orientations.

the sparse PC areas. In particular, fixed radius search with too small radius value possibly considers a very low number of neighbors. Hence, the computed statistics based on these points would not be representative of the true point statistics. As the closed form solution does not exist, a rule of thumb is used. It helps to choose the right value and to avoid the large effort of parameter tuning per dataset. In particular, radius R is set to the multiple values of the mean point resolution within this PC:

$$R = 2.5 \cdot \sum_{i=1}^{|\mathbb{P}|} \overline{d_i}(\mathbf{p_i}), \tag{2.9}$$

where $\overline{d_i}(\mathbf{p_i})$ is the average distance from point $\mathbf{p_i}$ to its closest neighbors, from Equation 2.1. The value of 2.5 has been chosen experimentally after considering a number of PC datasets. The PCs that are considered in this work have an average point resolution of 2 cm, i.e., the average distance between neighboring points is 2 cm. Therefore, the radius for surface normal estimation is set to 5 cm. An example of the normal estimation result is shown in Fig. 2.6. It can be observed that the normal direction follows the local surface orientation.

It has been observed that the previously described approach to normal estimation using PCA has the disadvantage of not preserving a sufficient level of detail on the edges of objects (see the left part in Fig. 2.7). The normals of points lying close to the object edge (top-left corner) are smoothly changing their direction to accommodate for the geometry change (edge). For many object segmentation algorithms, such smoothly changing normal directions lead to deterioration of segmentation performance. In particular, with a lower level of geometric and surface details in the boundary regions, it is harder to make a correct decision on the object boundary. Furthermore, PCA-based methods cannot reliably estimate the surface

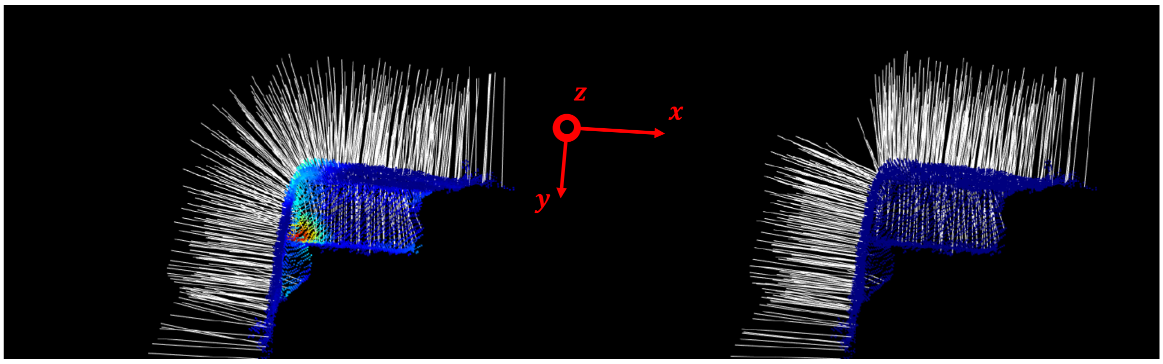


Figure 2.7: Left: PCA normal estimation result, right: Robust Randomized Hough Transform (RRHT) normal estimation result for the edge area of an object. The white arrows originating at the corresponding points of the PC indicate normal directions. The left figure shows that the normals corresponding to the points lying close to the object edge (top-left corner) smoothly change their direction to accommodate for the change in the object geometry due to the edge. This can lead to loss of information on the object boundary for object recognition algorithms. In contrast, in the right figure, it can be observed that the normal vectors are oriented in vertical and horizontal directions, thus creating a clear, sharp transition between the two sides of the object. Such sharp transitions in normal vectors corresponding to the change in geometry allow many object recognition algorithms to improve their performance.

properties in the presence of point sampling anisotropy, e.g., when the point density significantly varies across the PC, as shown in Fig. 2.8. To mitigate this effect, there exists a multi-scale modification to PCA that performs normal estimation in several iterations with the radius value being adapted depending on the curvature of the surface estimated in the first step with a fixed radius. It has been observed that this approach performs better than plain PCA, but it requires a considerable effort of parameter tuning that needs to be done for every dataset depending on the present point resolution and the level of noise.

Robust Randomized Hough Transform. There are alternative methods to normal estimation using parametric representations that achieve superior performance at the cost of higher computational complexity. In particular, Boulch *et al*. [31] proposed the Robust Randomized Hough Transform (RRHT) method to address this problem. The main idea of the approach is to first select a suitable neighborhood for each point and then randomly pick three points out of the neighborhood of the corresponding point. Using the picked three points, the plane can be estimated, which in turn can be used to calculate the plane normal. This process is repeated iteratively, each time with new randomly picked points. The computed plane normal direction that was obtained in every iteration is accumulated in a 2D histogram of directions. The most voted bin with associated normal direction is then used to compute the plane normal. To mitigate the discretization issues, the plane normal vectors corresponding to the most voted bin are averaged to obtain the final normal vector. An example of normal estimation for the edge area is shown in the right side of Fig. 2.7 and for a low point density area in the right side of Fig. 2.8.

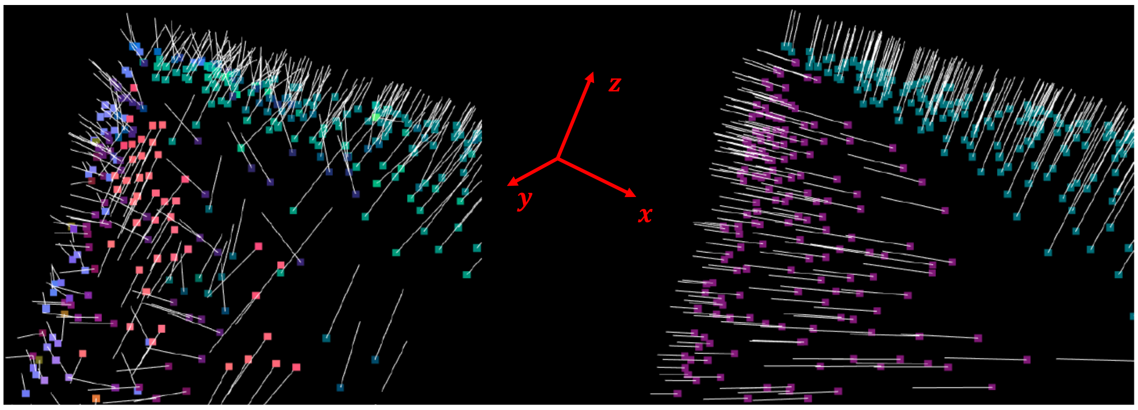


Figure 2.8: Left: PCA normal estimation result, right: RRHT normal estimation result. The color of the point corresponds to the unique colormap scheme, where each normal vector direction is mapped to a unique color on a sphere. Similar to Fig. 2.7, it can be seen in the left that the PCA normal estimation leads to noisy normal vectors, especially in the areas with low point density. In contrast, in the right figure it can be seen that the normal vectors are oriented in vertical and horizontal directions, thus creating a clear, sharp transition in the normal vectors between the two sides of the object.

2.1.6 Unsupervised Segmentation Algorithms

The primary goal of segmentation is to assign each of the points to a specific cluster. In the general case, the clusters do not need to have any meaning, most importantly, the similar points have to be grouped. There exist many unsupervised segmentation algorithms that are commonly used in 3D computer vision. They consequently serve as building blocks of modern complex segmentation algorithms. A number of employed segmentation algorithms are described in the following.

2.1.6.1 Min-cut Algorithm

One of the most common algorithms for segmentation is the min-cut algorithm, which formulates the segmentation problem in a graph [32]. As input to segmentation, there are a number of points that need to be assigned to one of $k = 2$ segments (binary segmentation). Within graph-based formulation, each point corresponds to a node in the graph. Now, the connections between nodes are added as edges to the graph with the associated edge weights. Thus, the graph $\mathcal{G} = \{\mathbb{V}, \mathbb{E}\}$ consists of a set of nodes $\mathbb{V}$ and a set of directed edges with associated weights $\mathbb{E}$. The edge weights can be set to arbitrary values, but they need to be non-negative. There are also two special nodes that do not correspond to any real points and are purely virtual. They are called terminals and typically consist of the source s and the sink t. Thus $\mathbb{V} = \{s, t\} \cup \mathbb{P}$, with $\mathbb{P}$ being non-terminal nodes. A simple example of a graph consisting of a source and a sink is shown in Fig. 2.9. In order to assign each point to s or t, there exist edges that connect each point to source and sink, shown in blue and red, respectively. Some edges connect non-terminal nodes, commonly denoted as smoothness edges. Now, by performing the min-cut on this graph, it is possible to determine the assignment of the point to s or t. Thus, each non-terminal node is either connected to source or sink. The

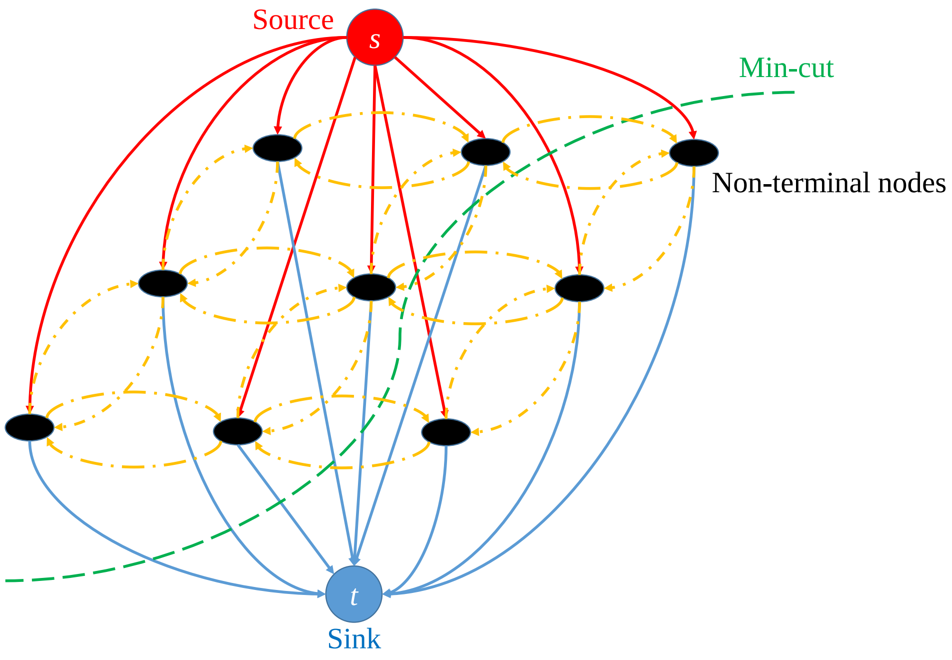


Figure 2.9: Illustration of a directed graph for the min-cut algorithm. Non-terminal nodes are shown in black. Edges to terminal nodes source s and sink t are shown in red and blue, respectively. Smoothness edges connecting non-terminal nodes are shown in yellow. The min-cut partitioning is shown in green.

connection defines the node's label assignment.

The graph formulation is particularly useful because it allows for an efficient algorithm to find a min-cut solution. For this, the max-flow min-cut theorem [33] states that the minimum s/t cut problem can be reformulated as the problem of finding a maximum flow from s to t. In other words, the maximum flow is the maximum amount of "water" that can be sent from s to t when interpreting the graph edges as directed pipes with specific capacities that are equal to the edge weights. Hence, the min-cut and the max-flow problems along their solutions are equivalent [33]. The maximum flow value is equal to the cost of the minimum cut. In a max-flow graph, each edge now contains capacity, which is the maximum possible flow on this edge. The edge assignment of flow needs to be less or equal to the edge capacity.

There are a number of methods to solve the maximum flow problem in polynomial time. They can typically be divided into two main groups: push-relabel and augmenting path-based algorithms. The push-relabel algorithm maintains active nodes that have a positive "flow excess" [34]. Thus, the algorithm keeps track of the node labeling, hence providing a lower bound estimate on the distance to t along non-saturated edges. The algorithm pushes excess flows towards nodes that have the smallest estimated distance to t. In contrast, the augmenting path algorithm pushes flow along non-saturated paths (e.g., paths with flow less than capacity) from the source to the sink until the maximum flow on the graph is reached. In the general case, the push-relabel algorithm performs better regarding segmentation performance. The graphs used in this work, however, have a certain structure, i.e., they are grid graphs. This means that the graph nodes are organized in a dense grid, such as pixels of

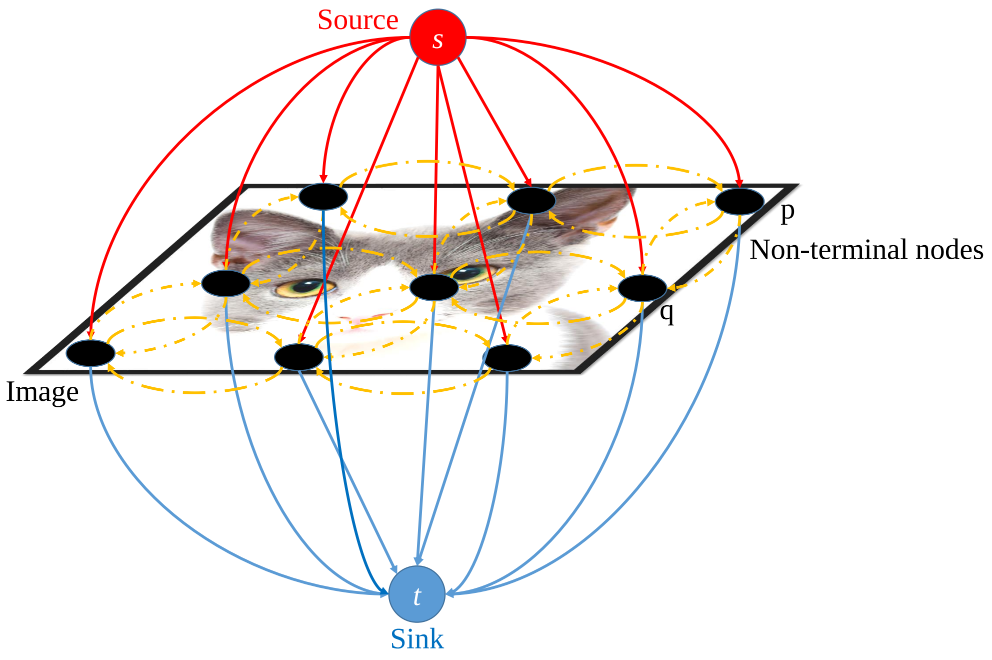


Figure 2.10: Illustration of a grid graph for image pixels. Non-terminal nodes are pixels from the image. For illustration purposes, only nine pixel nodes are shown. Background image is CC0 Creative Common license.

voxels. This way, the grid can be either 2D (pixels in an image grid) or 3D (voxels in a voxel grid). In such case, every node is connected to four, six or eight nearest neighbors based on their spatial neighborhood. For example, for a pixel, four nearest neighbors correspond to the pixels located above, below, left and right from the given pixel. For a voxel, six nearest neighbors correspond to the voxels located above, below, in front, behind, left and right from the current voxel. Such neighbors are connected to the current pixel or voxel through edges. See Fig. 2.10 for illustration of such a grid graph on image pixels. Grid graphs are different from general graphs, where any node can be connected to any other node in an arbitrary configuration. For grid graphs, Boykov and Kolmogorov [32] introduced a fast augmenting path-based algorithm that leverages this grid structure. It typically outperforms the push-relabel algorithms and has linear running time [32].

The min-cut solution is also a solution to the Markov random field (MRF) formulation using a certain global energy function [35]. MRF is an undirected graphical model representing a set of random variables that satisfy the Markov property [36]. The energy function for the MRF is formulated as follows:

$$E(\mathbb{L}) = \sum_{p \in \mathbb{P}} E_p(L_p) + \sum_{e_{p,q} \in \mathbb{E}} E_{p,q}(L_p, L_q), \tag{2.10}$$

where $\mathbb{L} = \{L_p | p \in \mathbb{P}\}$, $L_p = \{s, t\}$ is the assignment of the non-terminal node $p \in \mathbb{P}$. $E_p(L_p)$ is a data penalty function for node p assuming a certain labeling L_p. $E_{p,q}$ is an interaction potential between nodes p and q. $\mathbb{E}$ is a set of all edges between neighboring nodes. To achieve equivalent formulation in the graph cut, the data penalty term is typically represented as an edge between this non-terminal node p and source s or sink t with corresponding weight. The interaction potential between nodes p and q is represented as an edge with a corresponding weight equal to a certain non-negative constant value if they have the same label, and 0 otherwise.

2.1.6.2 Random Walker Algorithm

Another important algorithm for segmentation is the Random Walker algorithm [37]. This algorithm is primarily used in the context of image segmentation. Image is given as a 2D matrix $\boldsymbol{I}$. In contrast to min-cut algorithms, it requires initial labeling of a number of pixels, the so-called seeds. Seeds represent an assignment of certain pixels in the image to foreground or background. Similarly to the graph cut algorithm, it also uses a graph $\mathcal{G} = \{\mathbb{V}, \mathbb{E}\}$, where neighboring pixels are represented by nodes $\mathbb{V}$ that are connected by directed edges with certain weights $\mathbb{E}$. There are no terminal nodes in this graph, i.e., no nodes s or t. The edge weight between nodes i and j is defined as follows:

$$w_{i,j} = \exp(-\beta(I_i - I_j)^2), \tag{2.11}$$

where I_i is the image intensity at node i and β is a propagation coefficient.

The random walker algorithm optimizes the energy, which can be expressed as follows:

$$Q(\mathbf{x}) = \mathbf{x}^T \boldsymbol{L} \mathbf{x} = \sum_{e_{i,j} \in \mathbb{E}} w_{i,j} \left(x_i - x_j\right)^2, \tag{2.12}$$

where x_i is a real-valued variable associated with node $v_i \in \mathbb{V}$ in the graph, and $\boldsymbol{L}$ is the graph Laplacian. Optimization is constrained by $x_i = 1$ for $i \subset \mathbb{F}$ and $x_i = 0$ for $i \subset \mathbb{B}$, where $\mathbb{F}$ and $\mathbb{B}$ represent the sets of foreground and background seeds, respectively. To find the optimal solution, anisotropic diffusion is performed from the seeded pixels to the remaining pixels in the image. There exists an analytical solution to the anisotropic diffusion problem by solving a sparse, positive-definite system of linear equations with the graph Laplacian matrix [37]. The graph Laplacian matrix is defined as follows:

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}, \tag{2.13}$$

where $\boldsymbol{D}$ is the graph degree matrix and $\boldsymbol{A}$ is the adjacency matrix of the graph.

2.1.6.3 Felzenszwalb-Huttenlocher Algorithm

This is a heuristic segmentation algorithm, which uses an undirected graph [38]. Similarly to previously presented graph-based algorithms, the graph $\mathcal{G} = \{\mathbb{V}, \mathbb{E}\}$ has a set of vertices $\mathbb{V}$ and a set of undirected edges $\mathbb{E}$. There are no seeds, hence no initial labeling is required. The edge weight $w_{i,j}$ is a non-negative measure of dissimilarity between nodes i and j. Now, a component in a graph is a subgraph $\mathcal{G}' = \{\mathbb{V}' \subset \mathbb{V}, \mathbb{E}' \subset \mathbb{E}\}$ where each node is connected and the nodes are assigned to the same segment. Thus, segmentation $\mathbb{S}$ can be presented by a number of components, each $\mathbb{C} \in \mathbb{S}$. The internal difference of a component $\mathbb{C} \subseteq \mathbb{V}$ is defined to be the largest edge weight of the minimum spanning tree $MST(\mathbb{C}, E)$ of the component.

The MST of the graph $\mathcal{G}$ is defined as the tree graph $\mathcal{G}'$ that contains the same vertices as the original graph $\mathcal{G}$ and has the smallest possible sum of edge weights so that the vertices in $\mathcal{G}'$ remain connected. Formally, the internal difference is computed as follows:

$$Int(\mathbb{C}) = \max_{e_{i,j} \in MST(\mathbb{C}, E)} w_{i,j}. \tag{2.14}$$

The difference between the components is defined as follows:

$$Dif(\mathbb{C}_1, \mathbb{C}_2) = \min_{\substack{v_i \in \mathbb{C}_1 \\ v_j \in \mathbb{C}_2 \\ e_{i,j} \in \mathbb{E}}} w_{i,j}. \tag{2.15}$$

In particular, given an assignment of a number of nodes to two connected components $\mathbb{C}_1 = \{v_i \subset \mathbb{V}\}$ and $\mathbb{C}_2 = \{v_j \subset \mathbb{V}\}$, the algorithm uses an adaptive metric to decide whether these connected components are merged. The components $\mathbb{C}_1$ and $\mathbb{C}_2$ are merged if the following condition is satisfied:

$$Dif(\mathbb{C}_1, \mathbb{C}_2) \leq MInt(\mathbb{C}_1, \mathbb{C}_2). \tag{2.16}$$

The minimum internal difference $MInt$ is defined as follows:

$$MInt(\mathbb{C}_1, \mathbb{C}_2) = \min(Int(\mathbb{C}_1) + \tau(\mathbb{C}_1), Int(\mathbb{C}_2) + \tau(\mathbb{C}_2)). \tag{2.17}$$

Here $\tau(\mathbb{C}) = \frac{k}{|\mathbb{C}|}$ is a threshold function that depends on the size of the component, i.e., its value is larger if the component size is smaller, and larger otherwise.

In essence, this algorithm performs heuristic merging of nodes by keeping track of the internal variance of each component. By comparing the internal variance value to the difference between the two components, a decision on a merging of two components is made. As this decision depends on the particular component statistics, this algorithm can adapt to specific statistics in a particular graph area. In practice, the algorithm's running time is much lower as compared to the Random Walker or the min-cut algorithm [37]. This property allows the algorithm's broad adoption in practice. Moreover, the algorithm has two hyperparameters that are directly linked to the preferred size of the obtained components. By setting the hyperparameter values accordingly, it is possible to obtain larger or smaller components, depending on the application.

2.1.6.4 HDBSCAN

HDBSCAN is a density-based clustering algorithm based on hierarchical density estimates [39]. In fact, HDBSCAN is a hierarchical extension of DBSCAN [40]. At first, the main idea of the DBSCAN algorithm is described. Given a set of points that need to be assigned to one of the segments, DBSCAN groups together the points that are closely located to each other, i.e., points with many nearby neighbors. These points are located in the areas of high point density. The remaining points are located in the low-density areas, hence they are marked as outliers. By operating solely based on density, the DBSCAN algorithm does not assume a specific shape of the segments and can even cluster non-convex shapes in multi-dimensional spaces. The underlying segmentation algorithm is based on region growing defined on k nearest neighbor graph. Here, undirected edges connect each point p to the k most similar points to p, based on the similarity matrix. Similarity can be defined by the user, hence various features are supported. After the similarity matrix and nearest neighbor graph are computed, a minimum spanning tree (MST) that connects all points is found.

Illustration of the DBSCAN performance on a challenging dataset with varying point density is shown in the top part of Fig. 2.11. It can be observed that clusters of non-convex shape are identified by the algorithm, despite a varying point density. The clustered points are illustrated in the corresponding color. The black points are not assigned to any cluster. It can also be observed that not all clusters have been correctly identified by DBSCAN. This is due to the erroneously set parameter of the neighborhood size that is used for point density computation. The choice of this parameter is particularly challenging as the best parameter value depends on the data. In contrast, the HDBSCAN algorithm does not require this parameter, and, instead, automatically derives the correct value of the neighborhood radius from the data. In particular, when the algorithm considers the resulting clusters based on the MST, it attempts to obtain stable clusters. Stable clusters are the clusters, which have large inter-cluster distances, whereas intra-cluster distances are small. Furthermore, the clusters are chosen in such a way that they do not change when a slightly different intra-cluster distance is selected. As a result of this specific procedure within HDBSCAN, its performance is typically superior as compared to the DBSCAN algorithm, see the bottom part of Fig. 2.11.

2.1.6.5 Mixture of Manhattan Frames

The buildings and objects within human-made environments typically have a limited number of major structures in the form of orthogonal and parallel planes. The traditional concept of Manhattan world [41] assumes that the planes are orthogonal to one of the axes of the single coordinate system. This assumption is highly restrictive as there may be multiple planes that are not orthogonal to each other. Straub *et al*. [42] observed that the number of such planes could be larger than two, but still limited to a number of the so-called Manhattan frames. For estimation of the Manhattan frames, it is beneficial to consider normal vectors of the points in the PC data. In particular, the normal vectors are located in a 3D manifold of rotations SO(3) due to the fact that the normal vectors are given in 3D space but have unit norm. For estimation of the Manhattan frames, a formulation of the clustering algorithm using spherical data has been given. The algorithm is called Mixture of Manhattan Frames. The algorithm explores the fact that there exist a limited number of dominant plane directions in indoor environments. To find these, an efficient clustering algorithm on spherical data has been formulated. In particular, the proposed approach uses an adaptive Markov chain Monte-Carlo sampling algorithm [43] with Metropolis-Hastings split/merge moves that allow inferring the unknown number of a mixture of clusters [44]. The main difference as compared to the common clustering algorithms, such as k-means and similar algorithms [45] is that the distance is defined in spherical and not Euclidean space and that the number of clusters does not need to be known beforehand. This formulation allows estimating dominant directions using normal directions. By doing so, dominant planes of the indoor

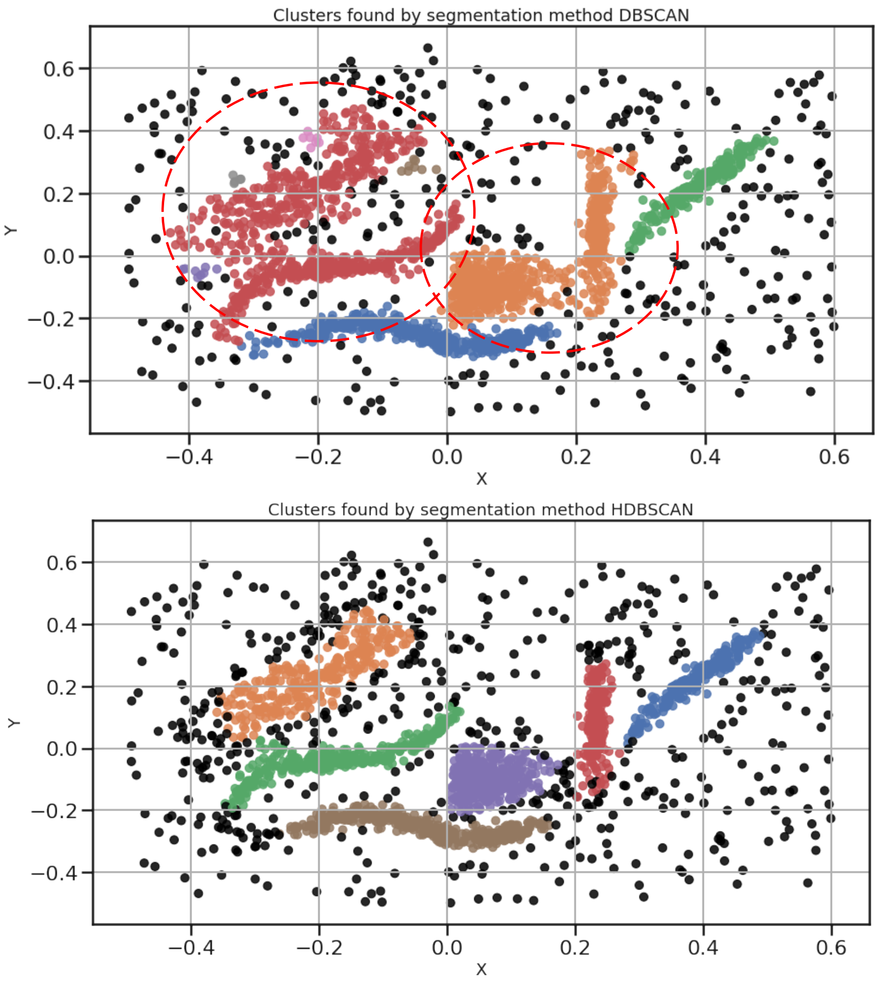


Figure 2.11: Illustration of the differences between DBSCAN (top) and HDBSCAN (bottom) clusterings on a point dataset with varying point density and non-convex cluster shapes. Colored points are assigned to the corresponding clusters so that different colors denote distinct clusters. Outlier points that are not assigned to any cluster are shown in black. HDBSCAN is able to distinguish the two distinct clusters in the top left area and two different clusters in the middle region (red and violet), whereas DBSCAN erroneously merges them together (these erroneous clusters are shown in encircled areas).

environment can be identified accurately and in short time.

2.1.6.6 Supervoxel Clustering

A supervoxel clustering algorithm has the goal of grouping points in the PC data into perceptually meaningful segments that conform to object boundaries. When performing segmentation in a PC data or an image, there is a common problem of a considerable computational complexity, e.g., a typical PC can contain hundreds of thousands of points and more. In particular, many common segmentation algorithms based on graph cuts have the property that the computational cost of inference grows sharply with the increasing number of nodes. To address this problem, it is possible to apply a preprocessing step by performing overseg-mentation of the points. This means that the points that are unlikely to belong to different

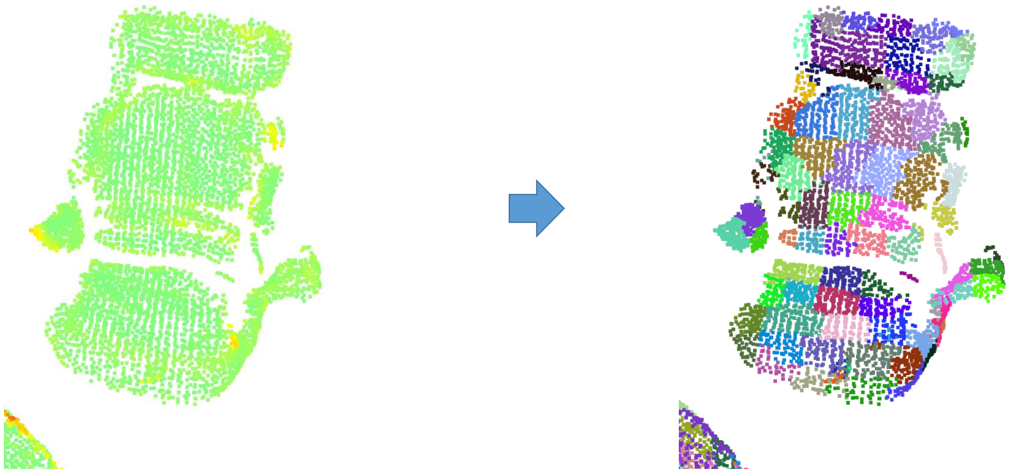


Figure 2.12: Illustration of the PC data (left) and the corresponding segmentation into supervoxels (right). In the right, different colors of points correspond to different segments. Here 1000 points in the PC (left) are replaced by 50 clusters (right). This supervoxel clustering step results in a significant reduction of computational complexity when performing object segmentation.

segments are grouped. Thus, this group of points can be replaced by their cluster center with the corresponding feature that abstracts the feature distribution of all the points within the cluster. As a result of this procedure, a significant reduction in computational complexity can be achieved by reducing the number of nodes in the graph. By carefully choosing the features, no deterioration in segmentation performance is observed. There exist a number of other supervoxel algorithms designed for various data. For more details, the interested reader can refer to the recent survey [46].

Out of different supervoxel algorithms, the Voxel Cloud Connectivity Segmentation algorithm [47] has the important advantage of being explicitly designed for PC data. The algorithm essentially segments the 3D PC data into surface patches called supervoxels (Fig. 2.12). The supervoxels are designed not to extend over the object boundaries. At first, the PC data is voxelized at a certain resolution R_{voxel}. This means that the points are replaced by the centroid of the resulting voxel grids. After that, the seeding of supervoxel clusters is done by partitioning the PC data at the resolution R_{seed}. These seeds are used to initialize the supervoxel centers. This step is followed by the computation of the supervoxel features. Each supervoxel $\mathbf{p}$ has an associated feature in a 39-dimensional space, defined as follows:

$$\boldsymbol{f} = [x, y, z, L, a, b, \mathbf{f_{fpfh}}], \tag{2.18}$$

where x, y, z are spatial coordinates, L, a, b are color components in CIELab space and $\mathbf{f_{fpfh}}$ represents the first 33 components of the Fast Point Feature Histogram feature [48]. Based on this supervoxel feature, k-means clustering is performed in order to group similar supervoxels together. After the clustering is finished, each resulting supervoxel has the corresponding information $\mathbf{p}_i = (\mathbf{x}_i, \mathbf{n}_i, \mathbf{Lab_i}, \mathbb{E}_i)$, with centroid $\mathbf{x_i}$, normal vector $\mathbf{n_i}$, corresponding color $\mathbf{Lab_i}$ in CIELab space and edges to adjacent supervoxels $e \in \mathbb{E}_i$. CIELab color space expresses color as three numerical values, L for the lightness, a and b for the green-red and blue-yellow color components, respectively. This color space was designed to be perceptually uniform with respect to human color vision, i.e., its definition closely matches human perception. The graph edge weight between neighboring supervoxels $\mathbf{p}_1$ and $\mathbf{p}_2$ is computed as follows:

$$D(\mathbf{p}_1, \mathbf{p}_2) = \frac{\|\mathbf{x}_1 - \mathbf{x}_2\|_2}{R_{seed}} \cdot w_s + (1 - |\cos(\mathbf{n}_1, \mathbf{n}_2)|) \cdot w_n + D_c(\mathbf{Lab_1}, \mathbf{Lab_2}) \cdot w_c, \tag{2.19}$$

where $\|\mathbf{x}_1 - \mathbf{x}_2\|_2$ is the Euclidean distance between the centroids of the corresponding nodes of supervoxel patches, R_{seed} is the seed radius, $\mathbf{n}_1$ and $\mathbf{n}_2$ are normals of the corresponding supervoxels. $\mathbf{Lab_1}$ and $\mathbf{Lab_2}$ are color components in CIELab space of the corresponding supervoxels and $D_c(\cdot)$ is their color distance. w_s, w_n and w_c are spatial, normal and color weights, respectively. Note that this formula for the graph edge weight is given in the PCL[7] and it is different from the one presented in the paper [47]. This difference is due to the fact that the authors later discovered that the above given formula is more robust on real data as compared to the original one.

Supervoxel clustering is the only existing algorithm that can work on large-scale PC data and generate meaningful feature clusters, thus reducing volume and complexity of the data, while keeping the information required for segmentation intact. This way, depending on the size of the desired cluster, 5–100 points can be replaced with one supervoxel. Illustration of a typical result of supervoxel clustering is given in Fig. 2.12. The running time of the algorithm is low, e.g., hundreds of milliseconds for PC data containing ten thousand points with typical parameter values. The running time depends on the chosen parameter value for R_{seed} and the level of connectivity in the adjacency graphs of the PC data.

2.1.7 Convolutional Neural Networks

An artificial neural network is a computing network structure that is loosely inspired by the neural connections in an animal brain, therefore it is called "neural" [49]. It is yet unclear how exactly the neural networks in the brain work. In computer vision, the most common neural networks are convolutional neural networks (CNN). A CNN is a feedforward neural network that contains multiple layers, where at least one of the layers is a convolutional layer. "Feedforward" means that the signal within the network only flows from input to output layers, i.e., the output of the network is not propagated back to the previous layers. For illustration of a typical CNN see Fig. 2.13.

In particular, a convolutional layer is based on convolution, i.e., a filter kernel is applied to the image (or matrix in general) in a sliding window fashion using convolution operation. The sliding window can be applied with a certain displacement along the data, this is commonly called *stride*. Thus, stride of 1 denotes that the displacement of 1 element is used in different dimensions when applying the filter. The filter weights within the same layer remain the same irrespective of the employed data region. A max-pooling layer applies maxima operation over a certain region, which means that only the maximum value within that region is propagated onto the next layer. A fully connected layer is a layer, in which the neurons are connected to each activation in the previous layer, therefore it is called fully

[7] Point Cloud Library http://pointclouds.org/. Accessed: 2018-12-16.

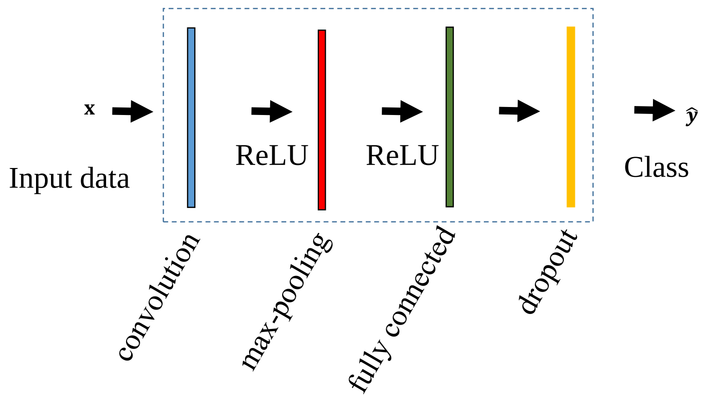


Figure 2.13: Illustration of a typical convolutional neural network architecture. It consists of convolutional, max-pooling and fully connected layers, followed by dropout. At the output $\hat{\mathbf{y}}$, the class of the input data $\mathbf{x}$ is predicted.

connected. This transformation can be formulated as an affine transformation. For example, given the input data $\mathbf{x}$, the weight matrix of fully connected layer $\boldsymbol{W}$ and bias factor $\mathbf{b}$, the output of the layer is computed as follows:

$$\mathbf{y} = \boldsymbol{W} \cdot \mathbf{x} + \mathbf{b}. \tag{2.20}$$

Thus, given the input data $\mathbf{x}$ of size 3072×1 and the weight matrix $\boldsymbol{W}$ of size 10×3072, bias factor $\mathbf{b}$ of size 1×10, the output $\mathbf{y}$ has dimensions 1×10. Between the layers, a non-linear mathematical operation is typically applied. This is done with the goal of easier approximation of a non-linear function that transforms the input data into such high-dimensional space, where points corresponding to same labels have a low distance to each other. In contrast, points having different labels have high distance from each other. This helps to classify the input data into one of the labels correctly. One of the common non-linear functions is the rectified linear unit (ReLU). Given scalar input x, the output of the ReLU function is defined as follows:

$$y(x) = \max(0, x). \tag{2.21}$$

As the output of each layer is typically a high-dimensional tensor, the non-linearity is applied to each entry independently $y_i = ReLU(x_i)$. The ReLU function is illustrated in Fig. 2.14. In essence, ReLU represents a linear operation in the positive range of the input data. In case the input data becomes negative, it "shuts off", hence a value of 0 is output.

Once the architecture of the CNN is defined, its weights need to be trained, which means that the weights of the particular layers are changed in such a way that the network can predict correct labels based on the input data. To quantify inaccuracy of the network prediction and other performance metrics of the network, a loss function is used. Based on the predicted labels, a certain scalar value is computed, which is then used to compute the gradient of the loss function with respect to the weights. By using the negative gradient as a direction of the layers' weight change at each step, the network can improve its performance over multiple iterations, in a so-called back-propagation operation. The most common loss function for

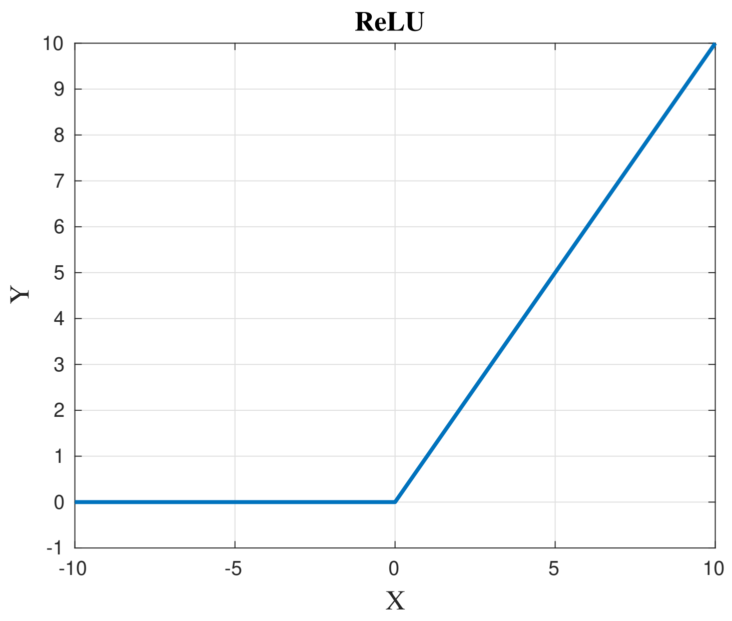


Figure 2.14: Illustration of ReLU function output vs. input. In essence, ReLU represent a linear operation in the positive range of the input data. In case the input data becomes negative, it "shuts off", hence a value of 0 is output.

classification is a cross-entropy loss. Given label predictions of the network defined in the form of the vector as $\hat{\mathbf{y}}$ and true object labels as $\mathbf{y}$, the loss is computed as follows:

$$L = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log \hat{y}_i, \tag{2.22}$$

where N is the number of instances. This loss function quantifies the difference between the true distribution $\mathbf{y}$ and the estimated distribution $\hat{\mathbf{y}}$ of labels.

In order to improve the generalization property of the network, regularization is typically applied. One of the most efficient regularization techniques is a dropout layer that is typically placed after a fully connected layer [50] (as shown in Fig. 2.13). The dropout layer with a dropout probability p sets neural weights of the layer located before it with probability p to zero. In other words, at every iteration of the training procedure, a number of neurons are disabled, which leads to the phenomenon that the network is forced to use all its weights for classification as a certain path is blocked at each iteration. This helps the network to improve its generalization property by using all possible neural paths. The dropout is not applied during testing, but only during training.

2.1.8 Evaluation Metrics

There are a number of metrics that are typically used for quantitative evaluation of segmentation and classification performance. They are described in the following.

2.1.8.1 Segmentation Metrics

ARI. For evaluation of segmentation performance, the Rand Index (RI) metric is commonly used. It was introduced in [51] for general clustering evaluation. This metric compares the compatibility of assignments between the point pairs within the segments. The RI between segmentation A and B is given by the sum of the number of point pairs that have the same label in A and B and those that have different labels in both segmentation, divided by the total number of point pairs. A further extension of RI has been developed, called Adjusted Rand Index (ARI) metric [52]. The main difference of ARI as compared to RI is that the former metric has been corrected with respect to the probability of a random guess. This has been done in order to mitigate the effect that some random guess-based clusterings can generate the value of RI larger than 0. In essence, the ARI metric compares how many clusterings have similar cluster assignments to generate a quantitative measure of similarity.

Richtsfeld et al.. Another common metric of segmentation is the metric of Richtsfeld *et al*. [53]. It includes oversegmentation (OS) and undersegmentation (US) errors. The OS error metric quantifies how many true segments are preserved, e.g., not oversegmented (split) into smaller subsets. The US error quantifies how many of the segments that do not have the same label have been erroneously merged into a segment with the same label. In particular, the OS error for the predicted segmentation and the true segmentation is given as follows:

$$E_{os} = 1 - \frac{N_{true}}{N_{all}}, \tag{2.23}$$

where N_{true} is the number of correctly assigned segment points, N_{all} is the number of all points. The US error is defined as follows:

$$E_{us} = \frac{N_{false}}{N_{all}}, \tag{2.24}$$

where N_{false} is the number of incorrectly assigned segment points.

Weighted overlap. This is a segmentation evaluation metric that weighs segmentation accuracy for segments according to their size [54]. In particular, let $\mathbb{G} = \{\mathbb{G}_1, \mathbb{G}_2, ..., \mathbb{G}_{|\mathbb{G}|}\}$ be a set of ground truth segments and $\mathbb{S} = \{\mathbb{S}_1, \mathbb{S}_2, ..., \mathbb{S}_{|\mathbb{S}|}\}$ be a set of predicted segments each containing indices of assigned points. For a given pair of segments $\mathbb{G}_j$ and $\mathbb{S}_k$, the overlap OV between them is defined as follows:

$$OV(\mathbb{G}_j, \mathbb{S}_k) = \frac{\mathbb{G}_j \cap \mathbb{S}_k}{\mathbb{G}_j \cup \mathbb{S}_k}. \tag{2.25}$$

The weighted overlap (WO) is a segmentation similarity metric that measures the overlap between two segmentations. It is defined as follows:

$$WO(\mathbb{G}, \mathbb{S}) = \frac{1}{|\mathbb{G}|} \sum_{j=1}^{|\mathbb{G}|} |\mathbb{G}_j| \max_{k=1..|\mathbb{S}|} OV(\mathbb{G}_j, \mathbb{S}_k), \tag{2.26}$$

where $|\mathbb{S}|$ denotes the cardinality of the set.

2.1.8.2 Classification Metrics

Precision and recall. In the context of classification, the most important evaluation metric is precision and recall. Precision specifies how many of predicted instances are classified correctly. It is defined as follows:

$$P = \frac{tp}{tp + fp}, \tag{2.27}$$

where tp denotes the number of true positives, fp is the number of false positives. True positive occurs when the classifier correctly predicts the label, and false positive happens when the classifier predicts the label for this example, even though this example is not relevant and does not contain a relevant label. Recall is defined as follows:

$$R = \frac{tp}{tp + fn}, \tag{2.28}$$

where fn is the number of false negatives. False negative happens when the classifier misses to predict the relevant label for this example, even though this example is relevant and should be detected. Precision and recall can be defined per category or over all categories. Their values lie in the range between 0 and 1.

F1-score. In an attempt to combine both measures of precision and recall into a single value, the F1-score was proposed [55]. It is computed as a harmonic average of precision and recall. Formally, F1-score is defined as follows:

$$F1 = 2 \cdot \frac{P \cdot R}{P + R}, \tag{2.29}$$

where P is precision, and R is recall. F1-score value lies in the range between 0 and 1. Higher values reflect better classification performance.

Confusion matrix. Another important metric of classification is the confusion matrix. It is a matrix that is used to describe the performance of the classifier when true and predicted labels are available [56]. An example is given in Table 2.1. Here, the value in the first row and the second column shows how many times the classifier predicted label 2 for the point whose true label is 1. Thus, the matrix rows represent the true labels, and the columns represent the predicted labels. The counts are typically normalized so that the sum of counts in each row is equal to 100%. Clearly, the error-free classifier that predicts the labels correctly produces a confusion matrix with diagonal-only values equal to 100%, whereas non-diagonal values are equal to 0%.

2.2 Related Work

In this section, a review of related work on room segmentation, object segmentation, and object classification is given. Furthermore, the main requirements for segmentation algorithms are formulated and discussed.

2.2.1 3D Room Segmentation in PCs

After the filtered PC data and the surface normal orientation are obtained, it is possible to proceed to the step of inferring semantic meaning from the data. The building data can

Table 2.1: Example of a confusion matrix for a classifier with 3 labels. Matrix rows represent the true labels and the columns represent the predicted labels.

True label	Predicted label: 1	Predicted label: 2	Predicted label: 3
1	83.2%	9.0%	7.8%
2	9.9%	84.8%	5.2%
3	3.8%	4.1%	92.1%

constitute millions or even billions of points, which requires a large amount of storage. Processing such amount of data at once would be practically impossible due to memory and complexity constraints in many recognition algorithms. This is because many of the algorithms were designed to operate on a certain size of the input data and would fail if the data volume is too large. Instead, it is beneficial to partition the data into semantically meaningful parts. This way, it is possible to process each part of the PC data separately using segmentation algorithms. After that, the segmentation result can be merged into a consistent output without any negative impact on the total segmentation accuracy. A reasonable way to define semantic parts for building data is to split the PC data into rooms. Thus, each semantic element (room) can be processed separately without the negative impact on object recognition performance, as the same object is unlikely to span multiple rooms at once. This way it is possible to leverage the benefits of parallelization, which can be significant for large indoor environments, e.g., $50\times$ for a typical office environment with 50 offices. There is no clear consensus on the definition of "room" in the literature. Some consider that a room is an enclosed indoor space [16], however, it remains unclear whether a long corridor consisting of several right-angle sections (see sections shown in orange, violet and green colors in Fig. 2.15) should be labeled as one room or as several. This issue is discussed in depth in Chapter 3.

The following **requirements** for a room segmentation algorithm are identified:

- **No assumptions on the wall structure**. Assumptions of Manhattan world, vertical or not curved (planar) walls significantly limit the applicability of a room segmentation approach. An illustration of an environment violating these assumptions is shown in Fig. 2.16. The Manhattan world refers to environments where the dominant planes (walls, floor, and ceiling) are oriented at right angles to each other [41].

- **No knowledge on the sensor trajectory is required**. Sensor trajectory information is often unavailable, for example for CAD models or when the sensor pose information is not stored after the SLAM process has been completed. It has been observed that this information is rarely provided for public PC datasets, e.g., [58], [11], [59].

- **No assumptions on the room layout**. Rooms can often be oriented in random patterns with respect to each other, and it is not reasonable to make assumptions about a specific room layout or whether a room represents a concave region (when considering a top-down view of the room). In the middle of Fig. 2.15 a corridor is shown. It consists

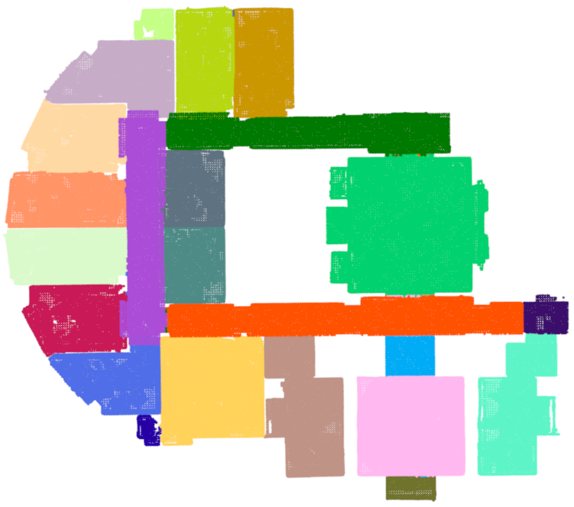


Figure 2.15: Illustration of the PC data segmented into rooms (top-view). Different colors of the points correspond to different room segments. Observe the u-shaped corridor in the middle that is denoted here as three separate rooms (green, purple and orange colors), whereas it is also correct to denote it as one large room. Dataset of [11].

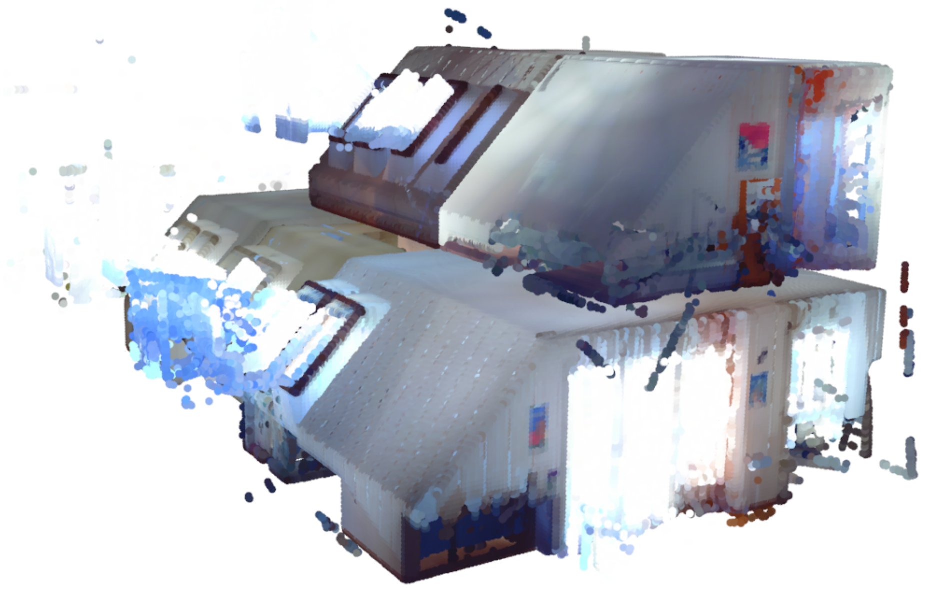


Figure 2.16: Illustration of the PC dataset with tilted walls violating the Manhattan world assumption. Many existing room segmentation methods would fail on such data. Dataset from [57].

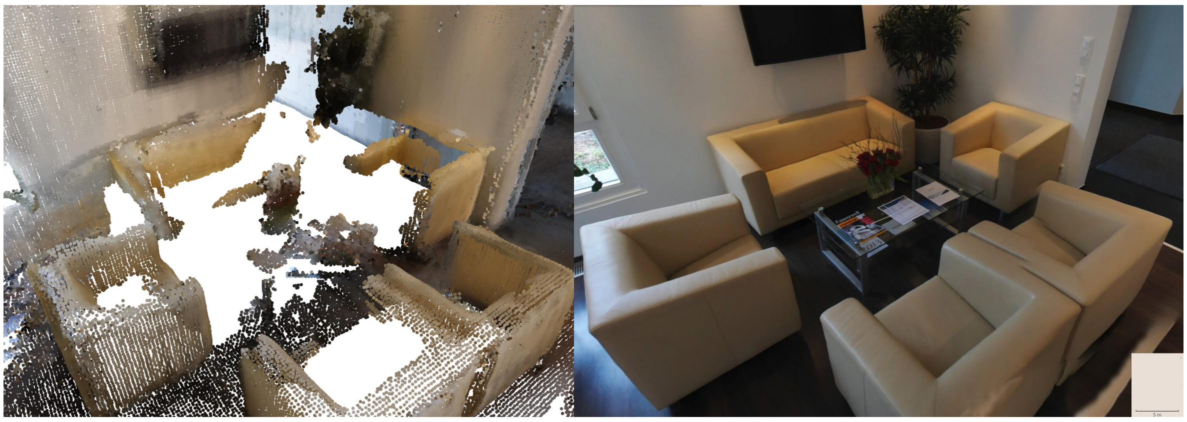


Figure 2.17: The PC of a room is shown on the left. The corresponding RGB image is shown on the right. It can be observed that significant parts of the object geometry are missing due to the fast mapping procedure - denoted as white areas in the middle of the left image. Due to limited scanning time, the objects cannot be scanned from all sides. Dataset of [4].

of three sections (shown in orange, purple and green colors). If labeled as one room, it represents a non-convex shape.

- **Robustness to noise, occlusion, and clutter**. Noise and occlusion are often present in sensor data of indoor environments, in particular when the data is obtained while mapping under time constraints using noisy sensors, see Fig. 2.17.

Although the problem of 3D room segmentation has been studied actively in recent years, related work only partially addressed the requirements mentioned before. In particular, this problem has been considered in the literature from the perspective of the robotics community, as well as from the view of computer graphics and vision.

Robotics. Most of the relevant previous work in the area of robotics has dealt with room segmentation in 2D data, in particular, with occupancy grid maps captured with a mobile robotic platform [60]–[67]. The focus of related work on 2D data is because obtaining large-scale 3D PC data of indoor environments still constitutes a significant effort when using mobile robotic platforms. In particular, [61] proposed a feature-based approach with Voronoi segmentation to represent different types of rooms, e.g., rooms, hallways, door, and crossings. This requires a significant amount of labeled data. The authors in [62] suggested a graph partitioning combined with spectral clustering to solve the problem of room segmentation. Spectral clustering has the disadvantage of having large computational complexity [68]. The authors in [60] provided an extensive survey on the topic of room segmentation in 2D maps and performed a comparative evaluation of various algorithms on a benchmark consisting of 20 datasets. The main insight from the published analysis is that machine learning approaches achieve superior performance. From the recent work, [65] proposed a convolution-based method to room segmentation using grid occupancy maps. In particular, a circular kernel is used to detect ripple-like patterns with large variance values in the maps. The neighboring regions with similar values are then merged. Furthermore, the authors also improved the evaluation metric of room segmentation in 2D maps. Fermin *et* *al.* [63] suggested an alternative to the common approaches of topological segmentation in structured and unstructured environments using a contour-based segmentation. Kleiner *et al.* [64] proposed a watershed-based method for segmenting occupancy grid maps into regions that represent rooms and corridors in the real world. These segmented regions are further merged into rooms using semantic decision rules. Fleer [66] approached the task of room segmentation by formulating it as a supervised learning task, where the classifier uses obstacle measurements and camera images to cluster the map into room-border edges. These edges are then segmented using graph clustering. Furthermore, [67] formulated the room segmentation task as a problem of automatic generation of an accessible graphic from a floor plan image. This work assumed that the clean and clutter-free floor plan image of the building is available, which is hardly the case for most indoor environments.

It has been observed that the presence of clutter in the indoor environment significantly deteriorates the performance of the segmentation approaches that use occupancy grid maps [61]–[67]. Furthermore, the approaches of [61] and [62] require information on the sensor trajectory and a significant amount of labeled data. In contrast, the approach of [16] does not require the viewpoint information and achieves high segmentation performance. It, however, relies on parametric plane detection, which limits its applicability for buildings with curved walls. For such walls, plane detection algorithms often fail, which would result in missing regions [69].

Computer vision and graphics. In the area of computer vision and graphics, the task of 3D room segmentation has been addressed by [11], [14], [15], [69]–[74]. The authors in [69] proposed a method that finds candidate permanent components by reasoning on a graph-based scene representation. This representation is then used to build a 3D linear cell complex that is partitioned into separate rooms through a multi-label energy minimization formulation. For this, the scan poses for every point in the PC are required in order to initialize the segmentation problem. The authors in [70] addressed the task of room segmentation by first detecting walls and room separations. The room separations were then used to generate 2D floor plans of the indoor environment. The authors of [15] applied a parametric approach to the problem of room segmentation and detected planes in indoor environments, which are then used to extract the segmented floor plan. Ochmann *et al.* [71] proposed to solve this problem by assuming a piece-wise planar structure of walls. The laser scanner capture locations are then used as prior information for the room segmentation task. The obtained segmentation is then refined in the global energy minimization problem. In the more recent work of [72], the vertical walls are projected onto the floor. After this step, all openings (e.g., doorways) are closed (removed) using morphological processing using the simplified (skeletonized) map. Mura and Pajarola [73] identified the fact that the viewpoint locations are important for clustering. Instead of using the viewpoint information from SLAM or SFM, a set of synthetic viewpoints is generated by embedding the bounding rectangles in an octree of the PC and using its leaf cells. This way, a clustering algorithm, like the Markov cluster algorithm [75], can be used to obtain an estimate of the room boundary [69]. Markov cluster algorithm is an unsupervised clustering algorithm based on flow simulation in graphs. In [11], the authors exploited the Manhattan world assumption. In particular, the dominant directions of the indoor environment are used as prior information for wall estimation problem. This information is then used to obtain an estimation of the room boundaries. Li *et al*. [74] proposed to first extract wall lines and use this information to erode the floor space. After that, this space is segmented into rooms. This approach is interesting in that it is one of the first approaches to multi-floor room segmentation. For this, the authors employed peak-bottom-peak strategy in the distribution of points along the z-axis in order to extract connected areas across multiple floors, which carries a resemblance to the method in [11] that is used to identify dominant vertical directions.

None of these approaches satisfies all of the requirements outlined above. Some assume that the data is not subject to significant levels of occlusion [15], which is rarely the case for an indoor environment. Other methods assume a Manhattan world structure of the building [11], [14]. The following methods [16], [72], [74] relax this assumption from the walls oriented at right angles to each other (Manhattan world) to vertical walls only. Nonetheless, this assumption still does not include the tilted walls, which are often observed in indoor environments (see Fig. 2.16). Some approaches further require the scan poses for every measured point in the PC [14], [69]–[71]. This information is difficult to obtain in practice, because the 3D PC data often represents the result of a fusion of data from multiple sensors, such as RGBD and LiDAR scanners.

Based on the fact that some of the above-mentioned requirements have not been addressed by the existing approaches, a novel approach to room segmentation is presented in Chapter 3.

2.2.2 Unsupervised 3D Object Segmentation in PCs

Once the PC data of indoor environments has been segmented into rooms, it is possible to proceed to the subsequent task of object segmentation. Similarly to the room segmentation, the object segmentation constitutes an ambiguous problem as the definition of the object can vary depending on the context and there is no single correct definition. For example, it is correct to say that the chair in the left side of Fig. 2.18 is a single object, but at the same time, it is also correct to claim that the chair consists of chair parts, such as chair arm, chair back and chair seat - see the right side of Fig. 2.18. The approach to semantic understanding involving segmentation and subsequent classification is not the only possible way to address the task. Some works ([11] among others) approached the problem of semantic understanding assuming that the segmentation is not available. Instead, a sliding volume method is employed, where a sliding cube is moved through the PC data in three directions along X, Y and Z axes. For each possible cube position, an object label is predicted [76]. Afterward, a non-maxima suppression method is used to merge neighboring predictions to a consistent result. Whereas such approach avoids the problem of segmentation, the resulting computational complexity of the sliding cube approaches is cubic in the dimensions of the PC data, as each possible position with a certain volumetric stride (step) needs to be evaluated. For example, it is possible to use a cube of size $1\text{ m} \times 1\text{ m} \times 1\text{ m}$ in a PC having dimensions of $100\text{ m} \times 50\text{ m} \times 5\text{ m}$, which are exemplary dimensions for one floor of the indoor environment in the dataset of [11]. When using the cube overlap of 0.25 m, this results in the following

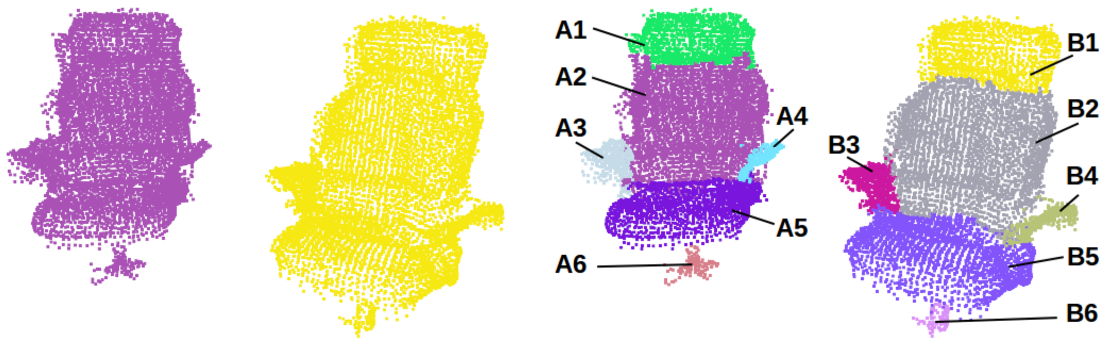


Figure 2.18: Illustration of the object segmentation for the chair. Left: the entire chair is treated as one object. Right: the chair parts are treated as separate objects. Here, different colors correspond to distinct segments. It can observed that it is meaningful to consider multiple segmentation scales at once. Dataset of [4].

number of possible cube positions:

$$N = N_x \cdot N_y \cdot N_z = \left(\frac{100-1}{0.25}+1\right) \cdot \left(\frac{50-1}{0.25}+1\right) \cdot \left(\frac{5-1}{0.25}+1\right) = 1{,}329{,}553. \tag{2.30}$$

Such a large number of computations is prohibitive. Moreover, the resulting object localization accuracy is limited to a specific volumetric overlap (in this case 0.25 m). Furthermore, this issue becomes even more pronounced, when the environment does not satisfy the Manhattan world assumption, as in this case different possible orientations need to be evaluated, resulting in additional computational complexity.

Ideally, an object segmentation approach has to meet the following **requirements**:

- **Robust to noise and occlusions**. Noise and occlusion are commonly present in PC data of indoor environments, in particular, in multi-view PC (MVPC) data that has been obtained using a moving sensor, such as laser scanner, depth sensor (Kinect) or RGB camera when moved along a certain trajectory and combined with the SFM approach.

- **Unsupervised**. The algorithm has to exhibit a good segmentation performance without the need to obtain a large amount of annotated data. This step can require large effort due to significant variations in objects geometry and many possible labels.

- **No assumptions of planarity or specific object geometry**. The assumptions of planarity or specific object geometry are often violated for curved objects or objects with varying geometry - see Fig. 2.18.

- **Low computational complexity**. It is essential for the segmentation algorithm to have low computational complexity in order to enable the application of the algorithm on large-scale indoor scenes.

The problem of object segmentation is a well-researched topic. However, the existing approaches were mostly designed with single-view and small-scale PC data in mind. Single-view refers to the fact that the PC data has been captured from a single viewpoint. This is

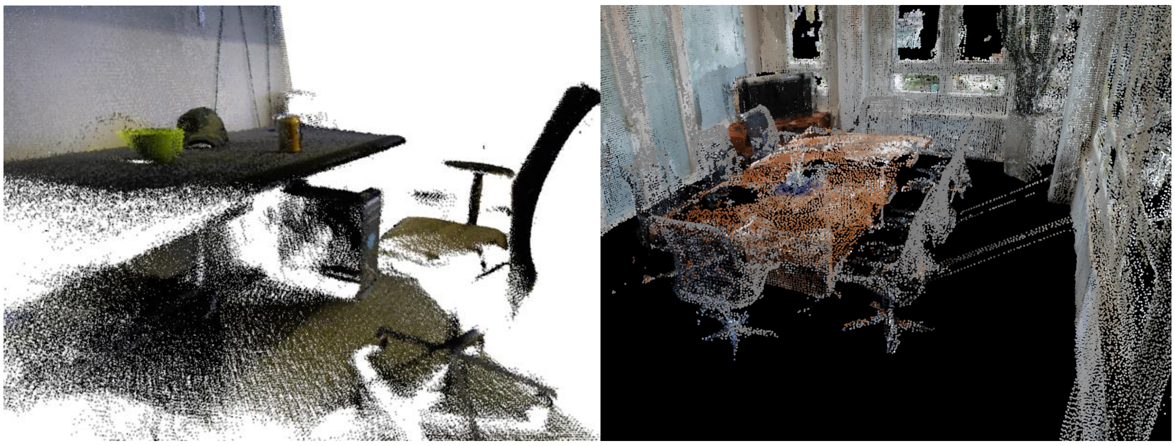


Figure 2.19: Left: illustration of a typical PC dataset obtained using handheld sensors depicting a small part of an indoor environment (dataset of [59]). A low level of registration artifacts and high point density can be observed. Occlusions, however, still exist. Right: typical MVPC dataset captured using a laser scanner in large-scale scenario (dataset of [4]). Higher levels of registration artifacts, reflections and occlusions are present.

in contrast to multi-view data when the sensor has been moved in a trajectory, thus resulting in a number of viewpoints. "Small-scale" term refers to the data depicting one room of the building or a part of it. A focus of previous work on small-scale data is because, until recently, the most extensive available indoor PC datasets were mostly captured with depth sensors with limited scanning range, such as Kinect [58], [59], [77] and [76]. They typically contain scans depicting small parts of indoor scenes. This is due to the high effort involved in recording PC data of large-scale environments using handheld sensors. In particular, this poses a challenge for object segmentation in large-scale indoor environments due to differences between two scenarios and the resulting data. When recording large indoor environments, the operational costs and time constraints become more important as the environment has to be free of moving objects during the time of scanning. This poses significant challenges as many environments cannot be made inaccessible for a long time, e.g., public environments including railway stations and similar. Compared to Kinect-based solutions, laser scanners have a clear advantage as they provide a larger scanning range, as discussed in Section 2.1.1.

As a result of the specific scanning procedure required for large indoor environments (e.g., building floors or even entire buildings), MVPC data acquired using a moving platform tends to have the following **differences** when compared to single-view data:

- **Unreliable surface normal information**, which is caused by registration artifacts. These artifacts are mostly due to inaccuracies when registering multiple range scans into a single 3D map. In particular, so-called "double wall" artifacts are caused by iterative point registration algorithms such as Iterative Closest Point (ICP) [78]. Such artifacts occur in any MVPC data, but they are most pronounced in large datasets because the registration noise tends to accumulate over time [79]. This results in deterioration of the object segmentation performance for algorithms that use normal information.

- **Varying point density** within different parts of the dataset. This is caused by a large scanner setup and strict time constraints, so that it is, sometimes, simply impossible to scan the objects from various directions (see the holes in the chairs and the table in Fig. 2.19).

Unsupervised small-scale object segmentation. Review of the entire literature on the topic of object segmentation can take significant space. For the sake of space constraints, the review is limited to the most relevant works. The interested reader can refer to [80] for a thorough, in-depth analysis. Most of the existing approaches for 3D PC segmentation were tested on single-view datasets. Hence, they do not consider the important peculiarities of MVPC data and tend to perform poorly on such datasets. In particular, [81] proposed to find approximate convex 3D shapes from a single RGBD image. At first, the image is segmented into superpixels, which are groups of pixels in the image that do not cross the object boundaries. Based on a large set of candidates generated from these superpixels, the selection of the candidates is optimized, so that they fulfill a number of properties. These properties include convex shape, small intersection value, and large scene coverage. Similarly, [82] proposed several objectness measures to be used in a graph-based mesh segmentation approach from [38]. The suggested measures include compactness, convexity, and volumetric extent. In [83], the authors proposed a method to infer the object segments based on the image. For this task, the pixels are grouped into planes. The orientation of planes and the edge convex/concave nature are inferred. The Manhattan world assumption has been leveraged in this work to reduce computational complexity. The entire scene interpretation problem is solved jointly by using superpixels. In the end, the problem of segmentation is formulated as a binary quadratic optimization problem. It takes the cues and geometric constraints into account to produce a final interpretation of the scene. In the method proposed in [84], the object relationships between different object categories are learned directly from the annotated data. The learned object relationships are then used to perform segmentation in the RGBD images using conditional random field employing mutex constraints. Van Kaick *et al.* [85] proposed to perform unsupervised object segmentation by decomposing the object into weakly convex parts. These decomposed neighboring weakly convex parts are merged in case they have similar properties and comparable volumetric properties. The authors in [86] suggested performing segmentation in parallel to the SLAM process in an incremental procedure using depth maps. The segmentation of the depth maps is then propagated onto the 3D PC data.

Unsupervised large-scale object segmentation. Handling large-scale PC data poses an important challenge concerning computational complexity. To mitigate this issue, [47] proposed to segment the PC data into a volumetric variant of superpixels called supervoxels. Here, a supervoxel is a group of neighboring PC points that belong to the same object and represent a surface. Instead of considering all points in the PC, it is possible to consider the extracted supervoxels that have associated features, such as 3D coordinates, surface normal, curvatures and shape descriptions. By reducing the number of nodes to be segmented, it is possible to significantly accelerate the process of object segmentation. For more details, see Section 2.1.6.6. For example, instead of considering 50 points, 1 supervoxel and the associated feature descriptor can be used. In the follow-up work of [17], the authors exploited the previously made observation in psychophysiology [87], [88]. This observation suggests that the transition between convex and concave image parts can be indicative of the separation between objects and their parts. In particular, the authors showed that this criterion could be applied to 3D object segmentation by distinguishing between convex and concave relationships. Given neighboring supervoxels, a graph can be created. This graph can be used to perform region growing, thus achieving segmentation. Before segmentation, it is important to remove noisy edges that are commonly present in noisy PC data. For this, convex and concave relationships between the supervoxels and certain heuristics, such as extended convexity criterion and sanity criterion, have been used. In the follow-up work of [89], it was proposed to use the RANdom SAmple Consensus (RANSAC) algorithm in combination with the convexity criterion to find the best planar cut separating the object parts. A number of methods exploited planarity and symmetry constraints. In particular, the approach of Mattausch *et al*. [90] leverages planarity assumptions in the indoor environments and represents an object as a collection of planar patches. Such simplified representation is then used to perform clustering in the indoor environment to performance object segmentation. Ecins et al. [91] proposed to leverage the symmetry constraint to improve the scene segmentation. Of the most recent work on unsupervised object segmentation, [92] proposed not only to employ the scene geometry but also reason about unknown objects using scene semantics. In particular, a CNN operating on RGB images detects object boundaries. The detected boundaries are then used to construct region hierarchies. The novelty of this approach is that the employed CNN does not need to be trained on the given environment and can be, instead, pre-trained on a different dataset. This allows this approach to be used to discover novel objects.

Supervised object segmentation. Due to the repetitive nature of indoor environments and the objects therein, it is expected that supervised methods can achieve good performance by learning typical patterns specific to certain object categories, e.g., tables normally have planar horizontal regions. Due to an extensive amount of related work on the subject, only the most relevant works are described here. For instance, [93] proposed to use depth images in combination with a supervised classification algorithm to perform object segmentation. The authors in [76] leverage existing 3D CAD models to render depth images from multiple viewpoints. These depth images are then used to create artificial PC data measurements, based on which various PC-based features are computed. These features are then given to the support vector machine (SVM) classifier to distinguish different categories. For object localization, a 3D detection window in 3D space is moved in an exhaustive search. Soni *et al*. [94] proposed to directly learn the object boundary in the depth image using a CNN. Of the most recent work, [95] proposed to use 3D fully convolutional neural network in combination with a recurrent neural network that implements conditional random fields for the task of semantic segmentation given PC data. Fully convolutional neural network denotes a network that contains only convolutional layers and does not have fully connected or other layers. The used data representation is a volumetric grid that is computed based on the PC data. After the class predictions on the grid are obtained, they are transferred onto the PC data using trilinear interpolation. In the recent work in [96], Li *et al.* showed how an analysis-by-synthesis strategy can be used to generate proposals for object instances, thus solving a problem of object segmentation. This proposal generation step can then be combined with instance segmentation approaches based on deep learning for the task of object detection.

The above-mentioned approaches that work on single-view data do not exhibit similarly good performance for MVPC data due to the previously described differences between MVPC and single view small-scale datasets. In particular, the convexity criteria used in [17], [83], [85], [89] as the primary evidence for an object boundary are negatively influenced by noise in concave object regions. This effect leads to deterioration of segmentation performance on MVPC data. Many approaches assume that the data is given in the form of a depth image [76], [81], [83], [84], [86]. This assumption increases the computational complexity for the recognition problem due to the additional step of depth rendering. Supervised approaches can improve segmentation performance [76], [82], [94], [95] but they require a massive amount of labeled examples. A significant effort is required to record and annotate MVPC datasets due to the 3D nature of the data. Other methods make unreasonable assumptions on scene planarity [90] or Manhattan world structure [83].

To address the above-mentioned challenges, a specific approach to unsupervised object segmentation exhibiting low complexity and able to handle large-scale PC data needs to be developed. The details of the proposed approach are given in Chapter 4.

2.2.3 3D Object Classification in PCs

Once the objects are segmented, classification of the object segments is performed. It is done to determine to which semantic elements these segments correspond, i.e., is it a chair or a table. This information can further be used in some applications, such as in robotics for object grasping and manipulation, in architecture for indoor modeling, in augmented and virtual reality for content creation and efficient visualization.

Descriptors are usually used for object classification. The descriptors take PC data as input and generate a multi-dimensional feature representation of the object. It is essential that the descriptor signatures are similar for objects of the same category and significantly different for objects of distinct categories. Depending on the considered scale, PC descriptors can be divided into global and local ones [97]. *Local* descriptors describe the geometry of the object in the local point neighborhood of the object part, e.g., a sphere of radius 2–10 cm. To describe the entire object, a large number of such descriptors is needed, which leads to the problem of high computational complexity. This complexity is not only due to the descriptor computation step, but also due to descriptor matching, as multiple descriptor pairs need to be matched to each other. Instead, *global* descriptors describe the geometry of the entire object using a single descriptor representation. As a single descriptor is computed per object, computational complexity in description and matching is typically significantly lower. Furthermore, the storage requirements for global descriptors are also significantly lower. It is desirable that the global descriptor signature does not change if the object is rotated. Otherwise, distinct descriptor signatures for rotated versions of the same object are obtained, which would lead to increased complexity in object matching. To mitigate this issue, it would be necessary to perform an additional step of object pose estimation, which presents additional complexity, especially in a large-scale setup.

Thus, the following **main requirements** for the PC object descriptor are identified:

- **Robust to noise and occlusion**. MVPC data typically has high levels of noise and occlusion due to registration artifacts. Hence, it is important that the descriptor signature does not significantly change when the data is subject to high levels of noise and occlusion.
- **Robust to changes in point sampling density**. MVPCs have non-uniform point density due to the inherent property of laser scanners. The descriptor signature should not significantly change when such effects occur.
- **Invariant to rotations**. Objects can be arbitrarily oriented in 3D space, and it is undesirable that the object descriptor values change when the object is rotated.
- **Low computational complexity**. It is essential that the descriptor has low computational complexity so that the descriptor is applicable for large-scale indoor environments. This requirement is even more important in case the object classification algorithm has to be deployed on a mobile robotic platform with limited computational resources.

A large number of global descriptors were proposed in the literature. The review of the most relevant descriptors is provided in Table 2.2. First, the handcrafted descriptors are reviewed, followed by learning-based object description approaches.

Point pair-based handcrafted feature descriptors. Majority of global descriptors employ point pair features to achieve rotation-invariance and robustness to occlusion. The first to propose randomly sampled point tuples for shape description were Osada *et al.* [109]. Later, [103] evaluated this approach for noisy PC data and confirmed its superior performance. Given the sampled point pairs, the features are computed. These features are further quantized in a 4D histogram. The used features employ the point distances and normal directions of the corresponding points. For matching, a simple approach of L_N distance using the histograms of two objects can be used. In later work, [110] proposed to use relative angles of the normal vectors of the corresponding point pairs to compute a Point Feature Histogram (PFH). The authors also evaluated the distances between the points in the point pair but came to a surprising conclusion that the distance feature is not very discriminative. The distances between point normals are then binned into a histogram. In the follow-up work, [98] introduced Viewpoint Feature Histogram (VFH) that is related to the PFH feature. In particular, based on the point normal and the normal of the PC centroid, the relative angles are computed. The combination of these features allows achieving higher robustness to noise. Furthermore, a viewpoint-dependent signature component was introduced. It is a histogram of the angles with respect to the viewpoint direction. The presence of viewpoint-dependent information allows the descriptor to be used not only for object classification but

Table 2.2: Review of the most relevant global PC descriptors. *Var.* refers to the fact that the descriptor size varies and no standard size is used by the authors (in principle, sizes for all descriptors can be changed).

Descriptor name	Descriptor size	Required information
VFH [98]	308	Normals and point coordinates
CVFH [99]	308	
OUR-CVFH [100]	308	
GRSD [101]	20	
SHOT [102]	135	
Wahl *et al*. [103]	625	
Drost *et al*. [104]	Var.	
Kasaei *et al*. [105]	25	
ESF [106]	640	Point coordinates
3D HOG [107]	Var.	
Lima *et al*. [108]	16	

also for pose estimation. Aldoma *et al*. [99] proposed Clustered Viewpoint Feature Histogram (CVFH) that is an extension of the VFH. This approach leverages the observation that the objects typically consist of a number of smooth regions. By splitting the object into several such regions and describing each of them separately using the VFH histogram, the algorithm can achieve higher robustness to occlusion and noise. In the later work, [106] proposed a descriptor called Ensembles of Shape Functions (ESF). In particular, ESF is an ensemble of multiple 64-bin-sized histograms of shape functions describing characteristic properties of the PC. The shape functions include angle, point distance, and area shape-based functions. To accelerate descriptor computation, a voxel grid is used to approximate the real surface so that the visibility calculations can be performed quickly. Depending on whether a point pair lies completely on the surface, partially on the surface or completely off the surface, the feature value is aggregated into a different part of the histogram. Thus, higher descriptive ability can be achieved as more information about global and local object geometry is stored. For an extensive quantitative comparison of point pair features, the interested reader can refer to the current review in [111]. The authors studied different point pair features and concluded that distance and normal-based point pair features show the best performance in case RGB information is not available. Once RGB information is available, it is beneficial to augment the descriptor with RGB information.

Object detection using point pair features. The authors in [104] showed how point pair features can be used for object detection when no prior segmentation is available. For point pair features, the features similar to [103] and [48] were used. They include distance and relative angles of normal vectors to the direction vector connecting two points in the point pair.

For matching, a fast voting scheme was employed. In the follow-up work [18], the authors showed how such an approach can be enhanced in order to tolerate higher levels of noise and occlusion by using Hough voting for verification. Furthermore, the steps of segmentation and pose verification are performed jointly. For this, the same point pair features have been used. Recently, [112] showed that by sampling points in a specific fashion (in contrast to previously used random sampling), it is possible to improve the descriptive performance of point pair features and thus enhance matching performance.

Handcrafted descriptors that do not use point pair features. There are also a number of approaches that do not use point pair features. For instance, [102] proposed Signature of Histograms of OrienTations (SHOT) as a local descriptor for PC data that uses normal information only. In particular, the 3D volume of the object is first divided into a spherical grid. For each of the bins in the grid, a local histogram of normal orientations is computed. Within the histogram, normal directions of the corresponding points and their relative orientations to the dominant orientation of the part of the spherical grid are used. The authors in [101] introduced a Global Radius-based Surface Descriptor (GRSD). Within this descriptor, a radius value of the approximating sphere is estimated for each point. Based on the radius value, the surface type for each point can be inferred, such as plane, cylinder, corner, sphere or edge. After this step, the adjacency between different surface types is accumulated by iterating over all points. The resulting descriptor histogram size is only 20 bins. Pedersoli and Tuytelaars [107] suggested a 3D Histogram of Gradients (HOG). This descriptor extends the 2D version of the HOG descriptor to 3D representations. For this, the PC data is voxelized into a voxel grid, and this representation is used to compute a statistical histogram of oriented gradients. The sensitivity to noise and occlusion remains an unsolved issue for this descriptor. To mitigate rotation issues, [108] proposed to transform the PC data into a canonical coordinate system. After this step, the point distribution is described using a histogram. This approach has the advantage that the RGB color distribution can be supported similarly. Analogously, [105] proposed a global 3D shape descriptor called Global Orthographic Object Descriptor (GOOD) that also employs projection. In particular, it uses orthographic projections along three main directions that are computed using the PCA (as explained in Section 2.1.5). The projected points are partitioned into spatial bins so that the point density is quantized in each of them.

Learning-based object description. The PC object description has also been studied in the context of a supervised formulation. In particular, [113] proposed a deep learning approach called Deep Local feature Aggregation Network (DLAN). This approach extracts rotation-invariant 3D local features and then aggregates these features in a neural network. For a description of local 3D regions, the DLAN uses a set of 3D geometric features that are invariant to local rotation. The resulting descriptor represents an aggregation of a set of features into a (global) rotation-invariant and compact feature.

Previously, deep learning approaches were limited in their performance on PC data due to the issue of point set order. In particular, a point set contains a number of points describing object geometry. This set is invariant to permutations of its members. The different permutations of the set members result in different input data for the network, which makes training of the neural network challenging. This issue is also known as symmetrization problem. The first to solve this issue, [114] proposed to apply max-pooling operation on the input point set in the PointNet deep learning architecture. This operation allows learning of the best features on the PC data directly in an end-to-end fashion. A number of alternative deep learning architectures working on PC data appeared after that. In the recent work, [115] recognized that the PCs originating from laser scanner sensors usually have 2.5D structure, i.e., they represent surfaces and not solely 3D volume. Hence, instead of performing convolution on point sets independently, it is possible to perform so-called tangential convolutions and, hence, improve the classification performance. The authors in [116] proposed graphCNN to operate on PCs and explore neighborhoods more efficiently than PointNet using a new module called EdgeConv. The authors in [117] formulated a general-purpose, CNN architecture to efficiently process large-scale 3D data using fully convolutional neural network. Li *et al.* [118] proposed a new convolution operator that works directly on the PC data. In particular, the authors suggested learning the point transformation from the PCs. The obtained operator is used to weigh the input features associated with the points and permute them into a latent canonical order. After that, the element-wise product and sum operations are applied. The authors in [119] proposed a pointwise convolution, which is a new convolution operator that can be applied at each point of a PC. For every point, nearest neighbors are queried on the fly and binned into kernel cells before the points are convolved with kernel weights. By stacking pointwise convolution operators together, the authors achieved superior results for scene segmentation and object recognition in PC data. Deng *et al.* [120] showed how point pair features can be used in a deep learning network to create a 3D descriptor. This descriptor can then be used to find correspondences between parts of PC data. This method uses n-tuple loss function and an architecture that can consider local and global object geometry at once. In the later work, Deng *et al.* [121] proposed an extension of this architecture for unsupervised learning of 3D descriptors. In particular, encoder-decoder architecture is used in combination with 4D point pair features.

Whereas deep learning methods show large potential for PC description, they are still not robust enough to be applied in large-scale scenarios. Their segmentation and classification performance is particularly low in case high levels of noise and occlusion are observed.

From the review of the related work on object classification, it can be concluded that despite very active research in this area, most of the descriptors do not show good performance on the MVPC data or do not achieve low computational complexity. To address these challenges, a novel approach to object description using point pairs is proposed in Chapter 5.

2.3 Chapter Summary

In this chapter, background knowledge on semantic understanding of indoor environments using PC data was reviewed. In particular, acquisition and representation methods for PC data were discussed in Section 2.1.1. Among different acquisition devices, laser scanners offer the fastest mapping procedure to capture large indoor areas in a short time. Out of different representations, the PC is the most suitable representation as it is the closest to the output from a laser scanner. Furthermore, this representation requires a little amount of storage, and it does not introduce additional artifacts to the PC data. Moreover, an overview of the typical semantic understanding pipeline has been given along the specific processing steps. In the first step, the PC of the indoor environment is segmented into rooms. This step is followed by the object segmentation step, where objects within each room are partitioned. Finally, a semantic label for each segment is determined using the object classification step.

PC preprocessing was reviewed in Section 2.1.5. It was identified that the RRHT method for normal estimation provides the highest robustness to noise and non-uniform point density. Sections 2.1.6-2.1.8 gave an overview of background knowledge that is essential for this thesis.

In Section 2.2.1, the desired properties of the room segmentation algorithm were discussed. This was followed by the review of the related work on room segmentation. It was identified that very few of the state-of-the-art room segmentation approaches could handle non-Manhattan structures, such as tilted or curved walls. Furthermore, the requirement of the available sensor trajectory information is very restrictive, especially for public PC datasets. None of the available room segmentation approaches satisfy both requirements.

In Section 2.2.2, requirements for object segmentation algorithms were described. It was identified that an object segmentation method needs to be unsupervised, make no assumptions about scene planarity and be robust to noise and occlusion. Based on the review of the related work, it has been identified that none of the available methods address all these requirements.

Similarly, in Section 2.2.3 the desired properties for an object classification approaches have been discussed. It was concluded that the PC-based object descriptor has to be rotation-invariant, exhibit low computational complexity and be robust to noise and occlusion. To address these requirements, it is beneficial to use point pairs. From the review of the related work, it has been identified that the existing methods exhibit inferior performance when dealing with non-uniform point density and high levels of occlusion.

Chapter 3

3D Room Segmentation in Point Clouds

In this chapter, the problem of room segmentation in PCs is presented. Furthermore, the proposed method for room segmentation using large-scale PC data of buildings is described. After that, the experimental evaluation of the proposed approach for a number of datasets is provided and discussed.

Parts of this chapter have been published in [5].

3.1 Problem Statement

As described before, room segmentation is an important step in the pipeline of semantic understanding of indoor environments. Furthermore, room segmentation is closely connected to the problem of indoor reconstruction that deals with the task of reconstructing a simplified representation of the environment using geometric primitives, e.g., meshes [14]. Each of the tasks benefits from the results of the other one, e.g., by having a room segmentation, a more

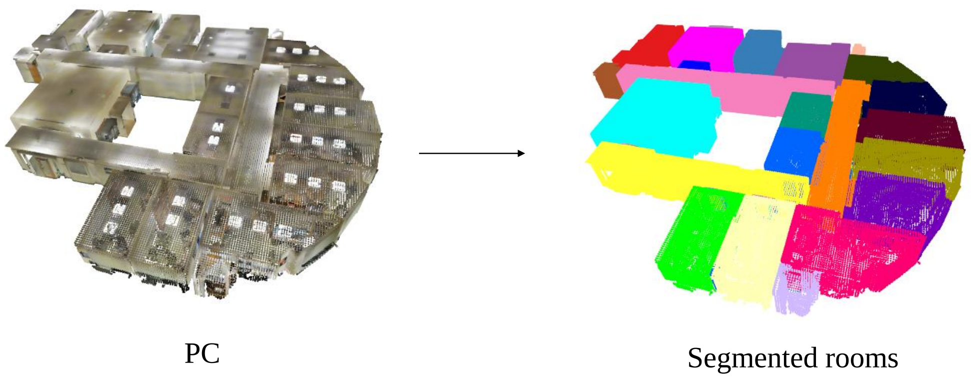


Figure 3.1: Illustration of the desired result of room segmentation (right) based on the PC data (left), where the points corresponding to different rooms are assigned to distinct segments. In the right part, different colors denote different rooms.

accurate primitive extraction is possible and vice versa. Both of the tasks become increasingly important because of the need for automatically generated semantic models of buildings from 3D data. Potential applications of these include virtual reality and architecture, to name a few. Before further processing, due to a large volume of PC data, it is important to partition the data into semantically meaningful parts, e.g., rooms (as shown in Fig. 3.1). This task, however, is made difficult by a number of factors, such as clutter, occlusion and a large volume of data. As reviewed in Section 2.2.1, previous work has only partially addressed these problems. In particular, the unsolved challenges are:

- The assumption of precise knowledge of the sensor poses [69], [70], [14]. By using this information, the task of interior space estimation becomes easier. Such information, however, is often not available. Furthermore, the sensor pose information is highly specific to the used acquisition sensor. Therefore, any algorithm that uses this information would need to be adapted to a specific scanning procedure (e.g., RGBD and LiDAR sensors).

- The assumption of the Manhattan world structure for the building parts, such as walls and other architectural elements that are oriented at right angles to each other [11], [69]. Clearly, this assumption does not hold for a general indoor environment exhibiting curved or tilted walls, see Fig. 2.15 for an illustration.

To tackle these limitations, the definition of a room has to be reviewed. As no precise mathematical formulation exists, an architectural formulation is used instead. According to the dictionary of architecture [122], a room is "In a building, a particular portion, an enclosure or division separated from other divisions by partitions". In other words, rooms are bigger (in volume) enclosed free spaces that are connected to each other through a smaller (in volume) free space, such as a door or an arch (see the top part in Fig. 3.2). This formulation follows the human understanding of a room having a certain homogeneous volumetric signature within its boundaries. To address this formulation, a new way to compute an anisotropic potential field (PF) for enclosed (inner) free space in 3D is presented. It is robust to the negative impact of clutter and occlusion. In particular, a number of objects can be observed in the top part of Fig. 3.2, such as a chair, a table and a dresser that are located on the left. Despite these objects, it can be seen in the bottom part of Fig. 3.2 that the PF values are increasing towards the center of the room. Thus, the PF values do not reflect the existing furniture.

For internal space representation, a voxel grid computed from the PC data is used. The voxel grid has the advantage of regularity that is quite important for segmentation algorithms. It also plays a vital role in the subsequent step of 3D to 2D projection so that 2D pixels can be computed directly from the 3D grid values without the need for extra processing steps. It was indeed discussed in Section 2.2.2 that a volumetric (voxel) grid requires a large amount of storage and complexity and it is, therefore, inferior to a PC representation for a number of applications requiring fine details. Nonetheless, this is not a major issue for this specific case due to the much coarser level of details that is sufficient for room segmentation.

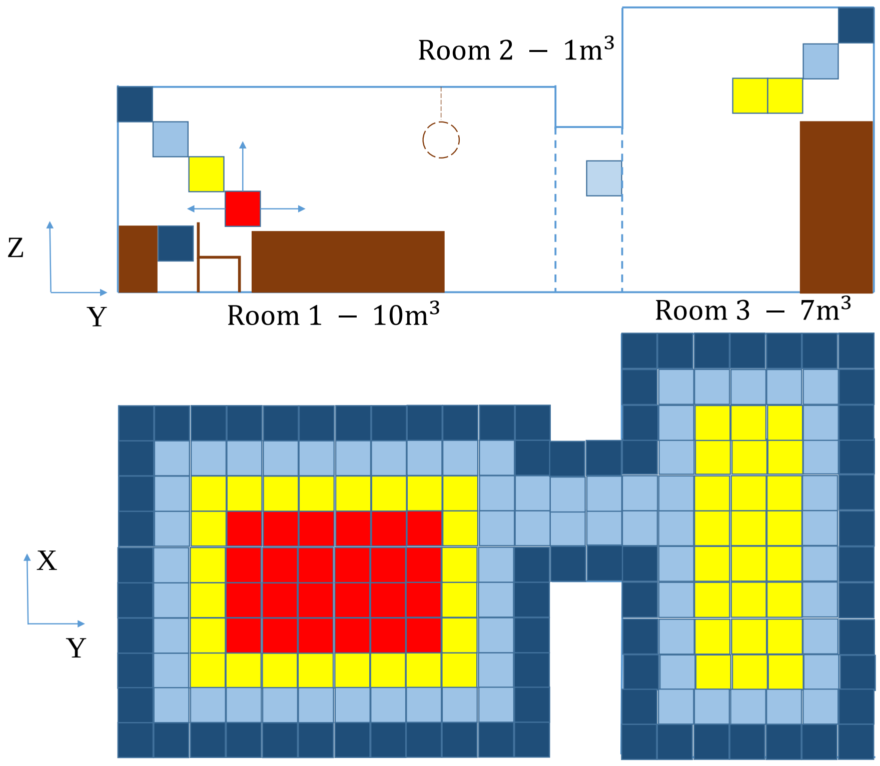


Figure 3.2: Illustration of rooms as inner free spaces separated by openings with different volumetric signature. Top: side view of an indoor environment with two rooms separated by a smaller room (corridor denoted as room 2). Furniture is shown in brown, several free voxels with the corresponding potential field (PF) values are also shown. Bottom: top-down view with proposed anisotropic PF maximum values along the vertical voxel stack. For voxels, red corresponds to high PF values and dark blue to low. Adapted from [5], ©2017 IEEE.

In particular, it is possible to use large voxels (20 cm) for this task as this size is adequate to capture building geometry and perform accurate room segmentation.

In the following, it will be described how such a representation can be used for room segmentation in a general indoor scene. After that, the described framework will be evaluated qualitatively and quantitatively on PC data of multiple buildings.

3.2 Method

An overview of the proposed method is given in Fig. 3.3. As input, a 3D PC (A) is used. Based on the 3D PC, interior free space voxels are detected (B). These interior free voxels are the free voxels that are positioned in the interior of the building, in other words, enclosed by the architectural elements of the buildings. After that, 3D anisotropic PF values for free voxels are computed, which is followed by maxima detection in the PF values within each vertical voxel stack (C). The maximum PF value along each stack is stored into a 2D PF map (D). This step is done to reduce the complexity of further processing. Given the PF image, clustering can be performed using the information about the PF values as well as the visibility between voxels (E). Finally, the labeling of the free space is mapped back to the 3D PC (F).

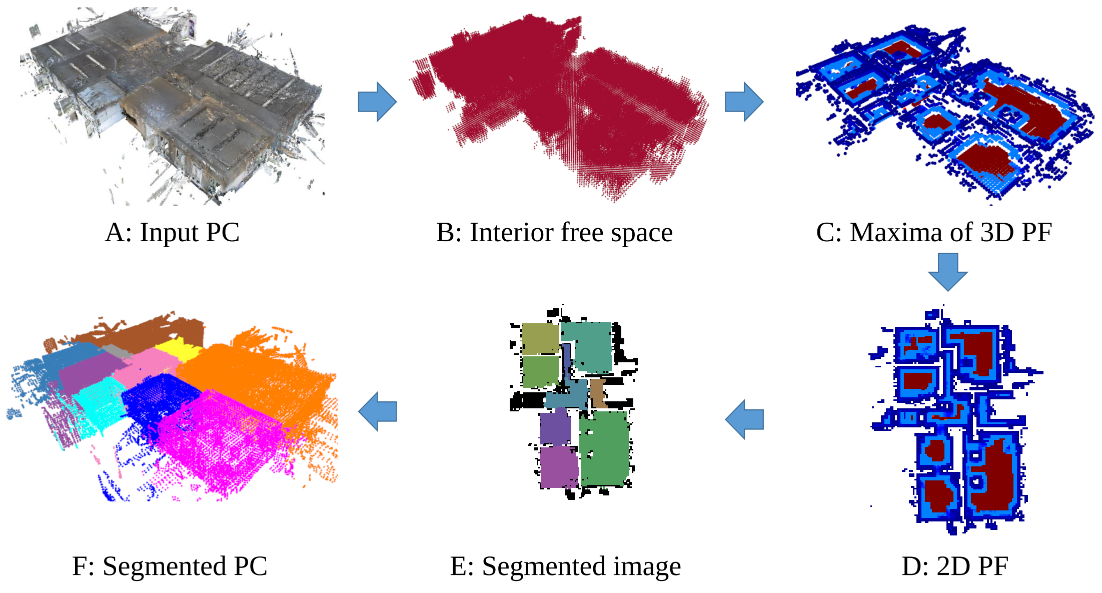


Figure 3.3: Overview of the proposed room segmentation method on the dataset of [4]. Input PC (A) is used to estimate the voxels that are inside the building (also called as interior free space) (B). Given the interior free space, the 3D potential field (PF) for every voxel is computed. The maxima of 3D PF along each vertical voxel stack are shown in (C). The maxima are projected onto a 2D PF image (D), which is then used to partition voxels (now pixels) into separate rooms resulting in a segmented image (E). In the final step, the 2D segmentation is projected onto the 3D PC producing a segmented PC (F).

As input data, unstructured 3D PC data is used, which is acquired using either RGBD sensors [14], [11] or LiDAR scanners [69]. No RGB information is used for any of the algorithms as geometry is more informative for the task of room segmentation. Further improvements can be achieved by incorporating RGB information, but this lies outside of the scope of this work. Furthermore, this would limit the applicability of the proposed method as not all PC datasets contain RGB information.

3.2.1 Interior Free Space Classification

In the first step, the interior free step needs to be identified, i.e., the free space inside the building interior. This step addresses tasks of free space classification and interior space recognition.

3.2.1.1 Free vs. Occupied Space Classification

As a room encompasses free space, the free space has to be recognized, as compared to the occupied space occupied by objects and architectural parts. For this, the 3D bounding box enclosing the PC data is voxelized, i.e., a 3D voxel grid is overlaid over the 3D extent of the PC. Each voxel that contains at least one point is labeled as occupied, and free otherwise. Hence, the voxels corresponding to furniture and other indoor objects are initially labeled as occupied even though they represent the inner volume of the room. In the worst case, when the room is entirely occupied with furniture, the entire inner volume will be labeled as occupied. As the goal is to reconstruct the inner volume of rooms and compute its volumetric signature, such voxels should be labeled as free. For this, [123] proposed an approach to classify voxels into free and occupied using a graph cut operating on the volumetric voxel grid with occupied/free labels. This method, however, requires camera poses and such information is often unavailable. Therefore, a different approach is selected. First, binary 3D morphological operations are applied to the 3D occupancy grid along the vertical direction (see the top part in Fig. 3.2). In particular, isolated occupied voxels surrounded by free voxels are identified and subsequently labeled as free. In essence, the pattern "occupied"-"free"-"occupied"-"free"-"occupied" along the vertical direction is detected. The central, occupied voxels matching such pattern will be labeled as free. For example, a lamp in the top part of Fig. 3.2 is removed as a result of such operation. Such voxel operations on large-scale datasets can result in high computational complexity. Therefore, a relatively large voxel size is chosen, e.g., with a side length of 20 cm. In order to further reduce computational complexity, neighbors of every voxel are precomputed and stored in a lookup table. This way, the constant time complexity for neighbor search is achieved at the cost of slightly increased storage requirements.

3.2.1.2 Interior vs. Exterior Free Space Classification

Now, the identified free space voxels need to be classified into the interior (inside the building) and exterior (outside the building). The theoretical solution to this problem could be formulated as a point in polygon enclosing check using raycasting or the winding number algorithm [124]. This is challenging, however, as it is difficult to estimate the enclosing polygon. Furthermore, visibility calculations have high computational complexity when done for all free voxels. Instead, [123] proposed to check the condition if the free space is enclosed by the occupied space using visibility checks. This is typically followed by formulating this task as a Markov random field (MRF) problem, which can be efficiently solved using graph cuts. See Section 2.1.6.1 for more details on the graph cut formulation. Performing such visibility checks in 3D for large-scale PCs, however, would result in a prohibitively high computational complexity. Therefore, it is, instead, checked whether this free voxel is placed between two occupied voxels along certain directions (so-called enclosing). To keep computational complexity low, the enclosing condition is not verified in all possible directions in 3D space. Instead, the properties of large-scale building datasets are leveraged, i.e., buildings typically have a limited and small number of dominant directions, which are most often occurring orientations of the plane normals. It has been observed that indoor environments have a small number of such plane orientations [42]. Hence, enclosing checks are performed for every free voxel only along the dominant directions of the indoor environment.

In particular, when estimating the dominant directions, it cannot be assumed that the environment is orthogonal and its dominant directions are axis-aligned to the global coordinate system, therefore the dominant directions in 3D space are estimated using the Mixture of Manhattan Frames (MMF) algorithm [42]. See Fig. 3.4 for an illustration of the dominant directions of the indoor environment. Refer to Section 2.1.6.5 for more details. It is, essen-

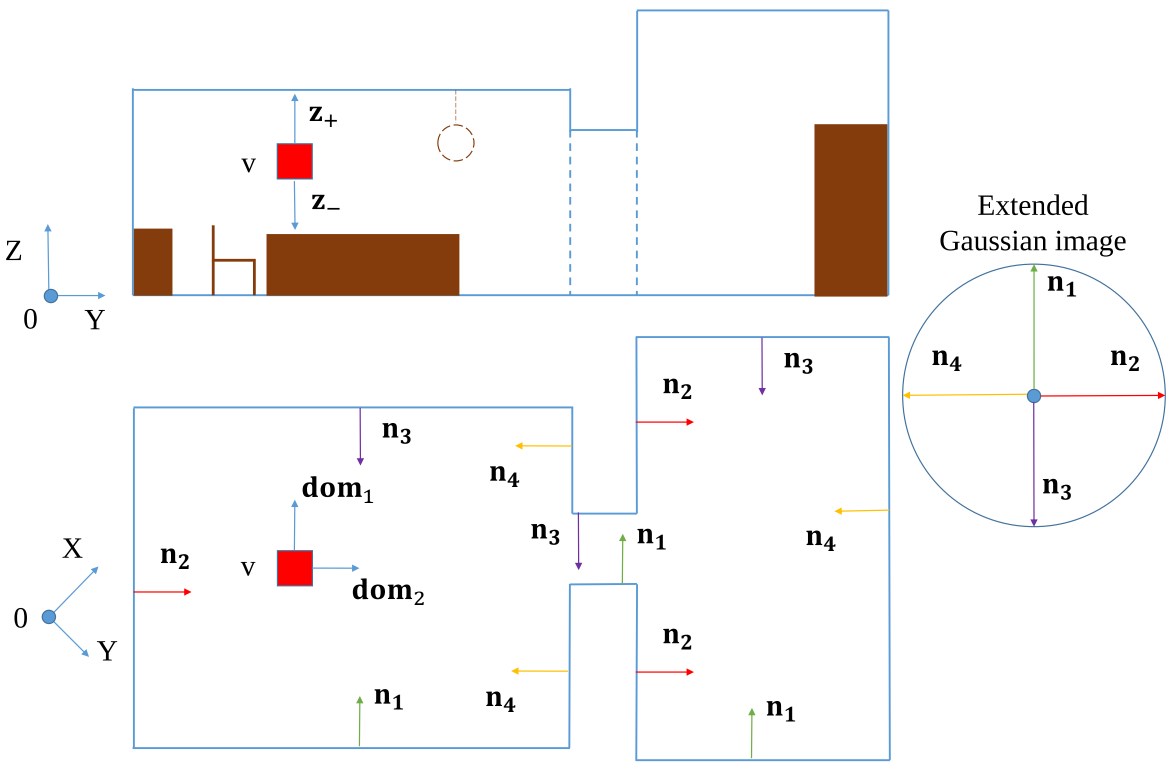


Figure 3.4: Illustration of dominant directions for the voxel v in an indoor environment. The dominant directions coincide with normal directions of the largest planes and in general do not coincide with directions of X and Y axes, as shown here. A dominant direction consists of two co-planar directions, e.g., $\mathbf{z}_+$ and $\mathbf{z}_-$ or $\mathbf{n_1}$ and $\mathbf{n_3}$. Normals for the corresponding planes $\mathbf{n_i}$ are also shown on the Extended Gaussian image on the right. It can be observed that the number of dominant directions for this environment is small (3).

tially, a formulation of a clustering algorithm for spherical data, which allows us to estimate dominant directions using normal directions. It has an advantage compared to PCA, as it allows to estimate more than three principal directions. Thus, instead of performing costly geometric checks in 3D space, it is possible to precompute the directions along which neighboring occupied voxels need to be checked. It has been observed that the level of occlusion is different in various parts of the indoor space due to the scanning procedure, while the scanning device is moved around using a moving platform. This in contrast to a flying platform, such as a drone, where a sensor observes the environment from above and below directions. Furthermore, it has been commonly noted that upper parts of the environment (e.g., ceiling and elevated parts) are less likely to be occluded during the scanning procedure as compared to the floor and lower parts of the environment due to the fact that the scanning platform is typically located on the floor [90]. Therefore, when accumulating evidence for a free voxel to be exterior, different weight values are chosen for various directions of enclosing. In particular, the data term $E_v(L_v)$ for voxel v, which serves as evidence for enclosing, is computed as follows:

$$\begin{aligned} E_v(L_v) = w_1 \cdot \mathbf{1}(v, \mathbf{z}_-) + w_2 \cdot \mathbf{1}(v, \mathbf{z}_+) + w_3 \cdot \mathbf{1}(v, \mathbf{z}) + w_4 \cdot \mathbf{1}(v, \mathbf{dom_1}) + \\ + w_5 \cdot \mathbf{1}(v, \mathbf{dom_2}), \end{aligned} \tag{3.1}$$

where $\mathbf{1}(v, \mathbf{z}_-)$ is equal to 1 in case this free voxel v has occupied voxels below it along the z-axis, and 0 otherwise, $\mathbf{1}(v, \mathbf{z}_+)$ is equal to 1 in case there are occupied voxels above the voxel v along the z-axis, and 0 otherwise. See Fig. 3.4 for an illustration of the various directions of enclosing. Here, also the Extended Gaussian image is shown for the plane normals [125], which provides an illustration of normal vectors placed in the unit sphere in such a way that each normal originates at the sphere center and ends in the sphere surface according to the vector orientation. $\mathbf{1}(v, \mathbf{z})$ is 1 in case there are occupied voxels above and below the voxel v. $\mathbf{1}(v, \mathbf{dom_1})$ is 1 in case there are occupied voxels in the positive and negative direction starting from voxel v along the first dominant direction $\mathbf{dom_1}$. It typically lies in the horizontal plane. Similarly, $\mathbf{dom_2}$ refers to the second dominant direction that also lies in the horizontal plane. Here, a higher weight is chosen for the case of occupied voxels located below voxel v as such voxels indicate a higher probability of this space being interior: $w_1 = 1.5$. In contrast, the remaining weights are set to $w_2 = w_3 = w_4 = w_5 = 0.5$. Even though the number of dominant directions is not strictly limited to two and can be larger, it has been observed that two dominant directions suffice for the considered datasets. $E_v(L_v)$ is used as the data term for a MRF formulation. As mentioned in Section 2.1.6.1, the MRF formulation, which is equivalent to the graph cut formulation, also requires a smoothness term. To recapitulate, the global energy function for the MRF is formulated as follows:

$$E(\mathbb{L}) = \sum_{v \in \mathbb{P}} E_v(L_v) + \sum_{e_{v,u} \in \mathbb{E}} E_{v,u}(L_v, L_u), \tag{3.2}$$

where $\mathbb{L} = \{L_v | v \in \mathbb{P}\}$, $L_v = \{s, t\}$ is the assignment of the non-terminal node $v \in \mathbb{P}$. $E_v(L_v)$ is a data penalty function for node v assuming a certain labeling L_v. $\mathbb{E}$ is a set of all edges between neighboring nodes. $E_{v,u}$ is an interaction potential between nodes v and u. It is defined as follows:

$$E_{v,u}(L_v, L_u) = \begin{cases} 0.6, & \text{if } L_v = L_u, \\ 0, & \text{if } L_v \neq L_u, \end{cases}$$

where L_v denotes the label of voxel v. The value of 0.6 has been experimentally verified to obtain a proper regularization of the indoor voxels, while still accurately following the computed free space evidence. Furthermore, a 6-neighborhood connected graph is built that is spanning all free voxels. In order to find interior free voxels, the graph cut using the Boykov-Kolmogorov min-cut algorithm is computed [32]. In particular, OpenGM-based implementation of the algorithm is used [126].

3.2.2 Anisotropic Potential Field Computation

Once the interior free space has been determined (see (B) in Fig. 3.3), the room segmentation step can be performed. So far, many voxels within the room volume have been labeled as occupied due to the presence of furniture and other objects. For the most accurate depiction of the interior space, it would be best to separate indoor objects from architectural elements of the building, but this remains a challenging problem in indoor reconstruction [69], [70]. Instead, it has been observed that PF-based approaches for room segmentation have shown good performance for 2D scenarios, e.g., when segmenting 2D occupancy grid maps [60]. The PF value of the free voxel is normally defined as the distance of the free voxel to the closest occupied voxel. A straightforward formulation of PF in 3D space would result in significant variations of its values due to clutter and indoor objects that are commonly located on the floor. These objects typically have little in common with room boundaries (see *table* and *chair* in the top part of Fig. 3.2). Therefore, the nearest neighbor search is instead performed in the half-space spanning the positive z-direction. This way, every voxel stores the squared L_2-distance to the closest occupied voxel lying in the half-space spanning positive z values, so-called anisotropic PF value. It is called anisotropic as the term "anisotropy" commonly refers to the property of being directionally dependent, i.e., the field properties vary in different directions.

Given the 3D PF map, it is possible to formulate clustering as an MRF problem with the PF gradient as a data term, thus enforcing smoothness. It was observed, however, that the maximum PF value along the vertical voxel stack is descriptive enough to detect rooms (see Fig. 3.3 (C)). This has the further advantages of low computational complexity and the ability to provide a simple visualization (see the lower part in Fig. 3.2). The resulting maxima of PF values from each vertical stack are now stored in a 2D image, which is used for further processing (see (D) in Fig. 3.3). In order to enhance the robustness of the method, visibility checks between pairs of voxels are performed. For instance, given a pair of voxels v_1 and v_2, a value of 1 is stored in case the v_2 voxel is visible from voxel v_1, and 0 otherwise. The v_2 voxel is visible from voxel v_1 iff no occupied voxels are located along the ray starting from the first voxel and terminating in the second voxel. Here, instead of performing visibility checks for every voxel, the visibility checks are only performed for the highest located free voxel within the vertical stack. It has been observed that due to variations in the ceiling profile the visibility of the highest voxel is more informative for space partitioning as compared to including the voxel with the maximum PF value.

3.2.3 Segmentation

Free voxels. Now, given the 2D PF map, discontinuities need to be identified, as these indicate room boundaries. Voronoi graphs combined with merging heuristics [60] can be employed for this. This solution would, however, impose constraints on the room layout and shape, e.g., in the general case rooms can be oriented in an arbitrary layout forming non-convex shapes, for example, corridors enclosing the rooms. Common clustering methods, such as spectral clustering [127] and k-means clustering, suffer from certain limitations: clusters need to have a similar number of points or clusters need to form convex shapes. Furthermore, graph cut algorithms typically suffer from the erroneous merging of smaller clusters into the neighboring bigger clusters. In contrast, density-based clustering algorithms (e.g., DBSCAN [40]) have an advantage as they do not assume any specific cluster shape, but, instead, perform region growing solely based on density. Furthermore, they can use a general distance metric, thus being able to incorporate other distance measures besides the Euclidean distance. It has been observed, however, that DBSCAN performance is highly sensitive to the specific parameter of the chosen neighborhood radius. To avoid the high effort of parameter tuning, its extension, called HDBSCAN, is considered [39]. This algorithm employs a specific measure to maximize the stability of the selected clusters, which improves the result. This way, no extensive manual search for optimal hyperparameters needs to be done as the algorithm does this automatically. For more details on the HDBSCAN algorithm, see Section 2.1.6.4.

Prior to clustering, the PF values projected onto a 2D map $\boldsymbol{I}$ are reshaped into a 1D histogram. This step is followed by peak detection in the histogram. As a result of the peak detection, a set of peak thresholds is obtained: $thr_0 \leq thr_1 \leq thr_2 ... \leq thr_k$. Furthermore, the maximum value of the PF map $\boldsymbol{I}$ is stored in a. Now, iterative thresholding of the 2D PF map $\boldsymbol{I}$ is performed starting from the highest intensity peak with index k in descending order of the peak values. The distance matrix $\boldsymbol{D^{PF}}$ is updated for each pair of pixels. In the beginning, all values in the distance matrix $\boldsymbol{D^{PF}}$ are set to a. At each iteration k, given threshold thr_k, $D_{i,j}^{PF}$ is updated for the pair of pixels (i, j) as follows:

$$D_{i,j}^{PF} = \begin{cases} \max(a - thr_k, D_{i,j}^{PF}), & \text{if pixels } i \text{ and } j \text{ become connected,} \\ D_{i,j}^{PF}, & \text{otherwise.} \end{cases} \tag{3.3}$$

Here "connected" denotes pixels for which there is a sequence of neighboring non-zero pixels containing a path from one pixel to the other. The combined distance matrix for the voxels (now pixels) for HDBSCAN clustering is defined as follows:

$$\boldsymbol{D} = \boldsymbol{D^{vis}} \cdot w_{vis} + \boldsymbol{D^{eucl}} \cdot w_{eucl} + \boldsymbol{D^{PF}} \cdot w_{PF}, \tag{3.4}$$

where $\boldsymbol{D^{eucl}}$ is the distance matrix that describes the Euclidean distance between the corresponding voxels. The entry at position i, j describes the Euclidean distance between voxels i and j. $\boldsymbol{D^{PF}}$ is the previously computed PF-based distance matrix. $\boldsymbol{D^{vis}}$ is the distance matrix based on mutual visibility. For voxels p and q, the corresponding entry in $\boldsymbol{D^{vis}}$ is computed as the normalized Hamming distance between their visibility vectors [14]:

$$D_{p,q}^{vis} = \frac{d(\mathbf{s}(p), \mathbf{s}(q))}{\sum_i s_i(p) + s_i(q)}, \tag{3.5}$$

where $\mathbf{s}(p)$ is the visibility vector of voxel p, such that $s_i(p) = 1$, if voxel i is visible from voxel p, and 0 otherwise. The Hamming distance d is defined as follows: $d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{N} \mathbf{1}(x_i = y_i)$, where $\mathbf{1}(\cdot)$ is an indicator function that is 1 in case the enclosed condition is true, and 0 otherwise. In order to take into account information on the PF difference as well as the change of visibility within different parts of the environment, the weight parameters are chosen as follows: $w_{vis} = 0.05$, $w_{eucl} = 0.7$, $w_{PF} = 0.25$.

After clustering, there exist a number of points that have not been assigned to any segment during the thresholding operation. To identify such non-assigned points, local peaks are detected. For this, peak detection using the smallest detected histogram peak threshold followed by connected component segmentation [128] are employed. Finally, to assign the remaining non-assigned points, the Random Walker algorithm [37] is used. The previously segmented pixels serve as seeds for this segmentation algorithm. For more details on the algorithm see Section 2.1.6.2.

Occupied voxels. As the goal of the approach is 3D segmentation, the segmented 2D PF map (see Fig. 3.3) needs to be propagated onto 3D occupied voxels. For this, the segmentation of the (segmented) free voxels is propagated onto unlabeled free voxels in the vertical stack along the vertical direction. Afterward, for each occupied voxel, its nearest ten neighboring free voxels that contain a segment assignment are determined. The most often occurring label of the labeled free voxels defines the labeling of the considered occupied voxel.

3.3 Experimental Evaluation

In this section, the proposed approach is experimentally evaluated and compared to a number of existing methods.

3.3.1 Evaluation Setup

For evaluation, several datasets were used. First, the performance was verified on the unlabeled laser scanner PC dataset that violates the Manhattan world assumption as described in Mura *et al.* [69], i.e., exhibiting tilted ceilings and curved walls. Furthermore, the evaluation was performed on the labeled indoor dataset of Armeni *et al.* [11] spanning 4 buildings and counting in total 175 rooms. For qualitative evaluation, the unlabeled dataset of Ikehata *et al.* [14] was used. The only algorithm parameters that were changed across datasets are the parameters of the volumetric graph cut w_1 and w_2, which were set depending on the level of occlusion in the datasets. In particular, $w_1 = 0.5$ for the dataset of Ikehata *et al.* because it exhibits a higher level of occlusion in the floor voxels. $w_2 = 1.5$ for the dataset of Mura *et al.* as it exhibits a lower level of occlusion in the ceiling voxels as compared to the other datasets. No further parameter tuning was performed on the dataset basis, as this would limit the generality of the proposed algorithm.

3.3.1.1 Results on the non-Manhattan Dataset of Mura [69]

The experimental results using the described method on the non-Manhattan dataset are shown in Fig. 3.5 along with the result of [69]. The proposed approach performed similarly to [69], even though it did not use the information on the scanner poses. In particular, it can be observed that all rooms have been correctly segmented. Notably, the segmentation result illustrates that the proposed approach is able to segment the building data exhibiting tilted walls and ceilings correctly.

3.3.1.2 Results on the Dataset of Armeni [11]

The experimental results using the proposed method along with the method of [11] on the dataset of [11] are given in Fig. 3.6. The authors of the paper used the Adjusted Rand Index (ARI) metric to measure the quality of segmentation. The main difference of ARI as compared to RI is that the former metric has been corrected with respect to the probability of a

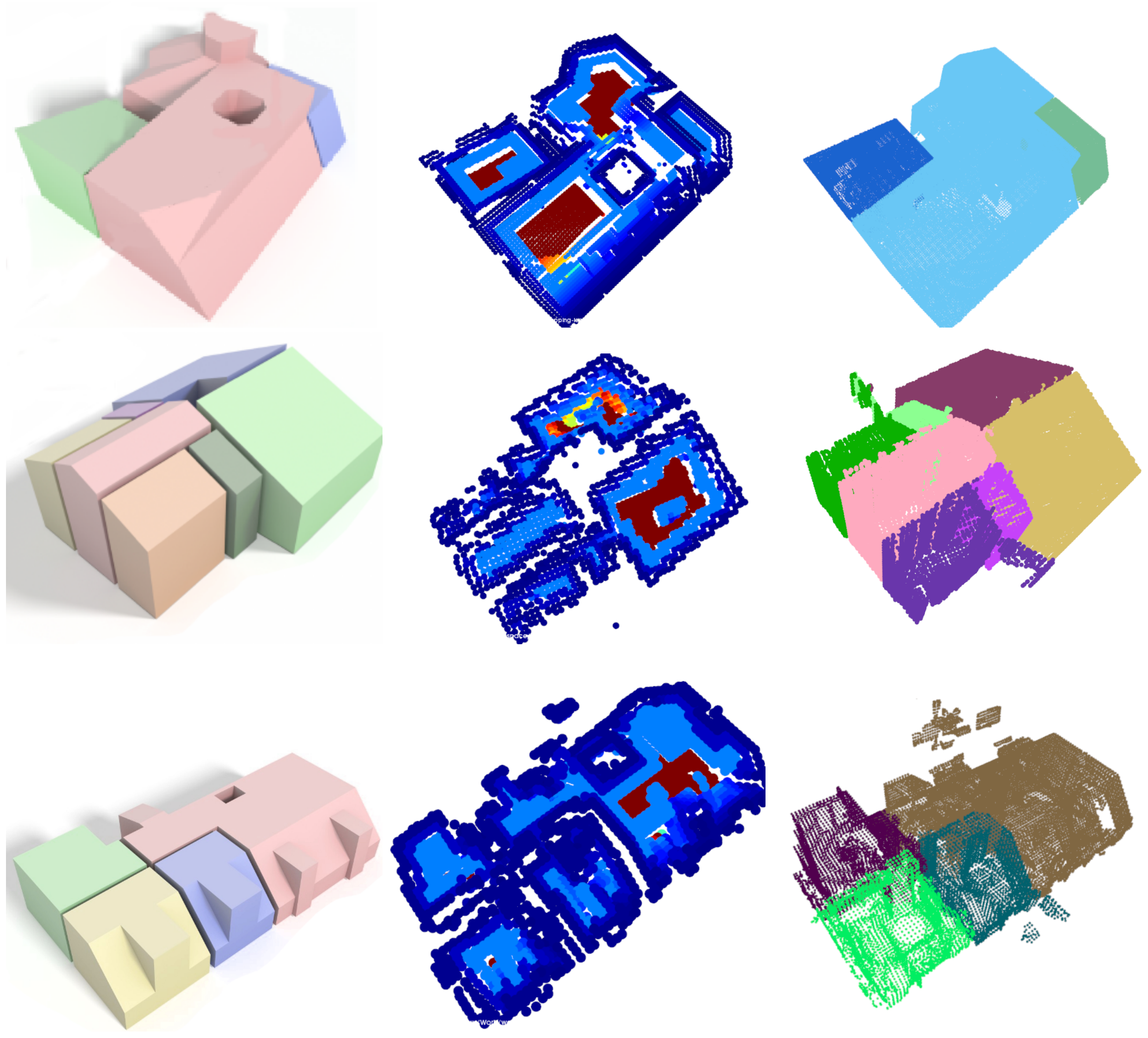


Figure 3.5: Room segmentation results for the dataset violating the Manhattan world assumption [69]. Different buildings are shown in different rows. Top row: Modern, middle row: Cottage, bottom row: Penthouse. Left column: reconstruction result of [69], where different colors correspond to different room segments. Middle column: PF map, where low values of the PF field are shown in dark blue and high values in red. Right column: room segmentation result of the proposed method. In the right column, different colors correspond to different room segments. It can be observed that all rooms have been correctly segmented by the proposed segmentation method, despite the fact that the sensor pose information was not used. The method of Mura *et al*. does not generate segmentation on the point basis, therefore only the reconstruction result is provided. Reproduced from [5], ©2017 IEEE.

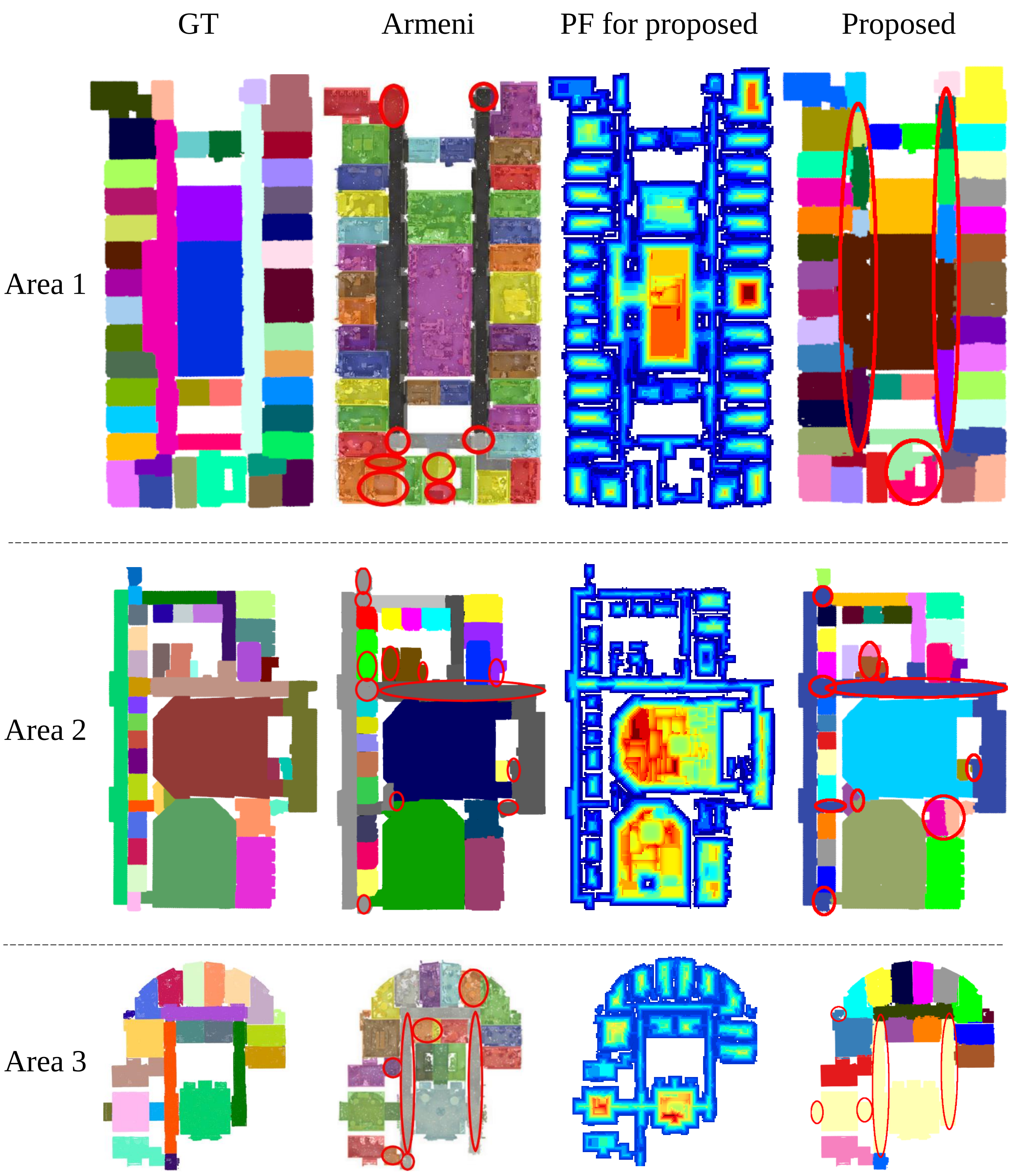


Figure 3.6: Room segmentation results for the large-scale dataset of Armeni *et al.* [11]. From left to right column: ground truth, segmentation result of [11], PF map for the proposed method, segmentation result of the proposed method. Here, in the two left columns and the rightmost column, different colors correspond to different room segments. In the second from right column, low values of the PF map are shown in dark blue, whereas the high values as shown in red. Here, red ellipses denote erroneously segmented rooms. Adapted from [5], ©2017 IEEE.

Table 3.1: Room segmentation results on the dataset from [11]. Numbers in the two right columns show the number of incorrectly segmented rooms (lower is better). It can be observed that the proposed method outperforms the method of Armeni *et al*. on all datasets. Adapted from [5], ©2017 IEEE.

Area	Number of rooms	Results for proposed	Results for Armeni [11]
1	44	**3**	8
2	40	**10**	12
3	23	**5**	7
5	68	**7**	13
Total	175	**25**	40

random guess. This has been done in order to mitigate the effect that some random guess-based clusterings can generate the value of RI larger than 0, which would lead to overoptimistic evaluation. In essence, the ARI metric compares how many clusterings have similar cluster assignments to generate a quantitative measure of similarity. For more details on the metric see Section 2.1.8. Using this metric required the segmentation result of [11], which was not made available by the authors. Furthermore, as ARI operates on points, instead of rooms, it is biased towards larger rooms. E.g., it inadequately measures the incorrect labeling of smaller rooms. Because of these reasons, a quantitative evaluation was not possible. A qualitative evaluation was performed, instead, and the erroneously labeled rooms were counted (shown as red ellipses in Fig. 3.6). The quantitative evaluation is given in Table 3.1. Such metric also allowed for a meaningful evaluation and could adequately represent the segmentation performance.

From Fig. 3.6 it can be observed that the approach in [11] does not perform well for the rooms that are not aligned with the main walls of the building, e.g., the top part of Area 1, the top part of Area 2 and the right part of Area 3. This is a critical conceptual limitation of the algorithm in [11]. Furthermore, smaller rooms are often erroneously merged into the neighboring bigger ones - see the top left and bottom parts in Area 1 and the middle part of Area 2. In contrast, the proposed approach does not assume the Manhattan world structure, therefore, it can label such rooms correctly. The proposed method still incorrectly labels several rooms due to irregularities in the PF map, which might be due to imperfections when estimating the interior free space. In particular, observe the corridors and stairs in Area 1. Furthermore, the proposed algorithm incorrectly segmented the smaller rooms in Area 2, which is due to the limitations of the proposed image segmentation algorithm. Finally, in Area 3 the proposed algorithm merged several corridors due to large entrance areas connecting them to the lobby area in the middle. This evaluation is somewhat conservative because the dataset labeling in [11] is inconsistent across different buildings: in some buildings, the corridor is labeled as several parts, while in others it is labeled as one.

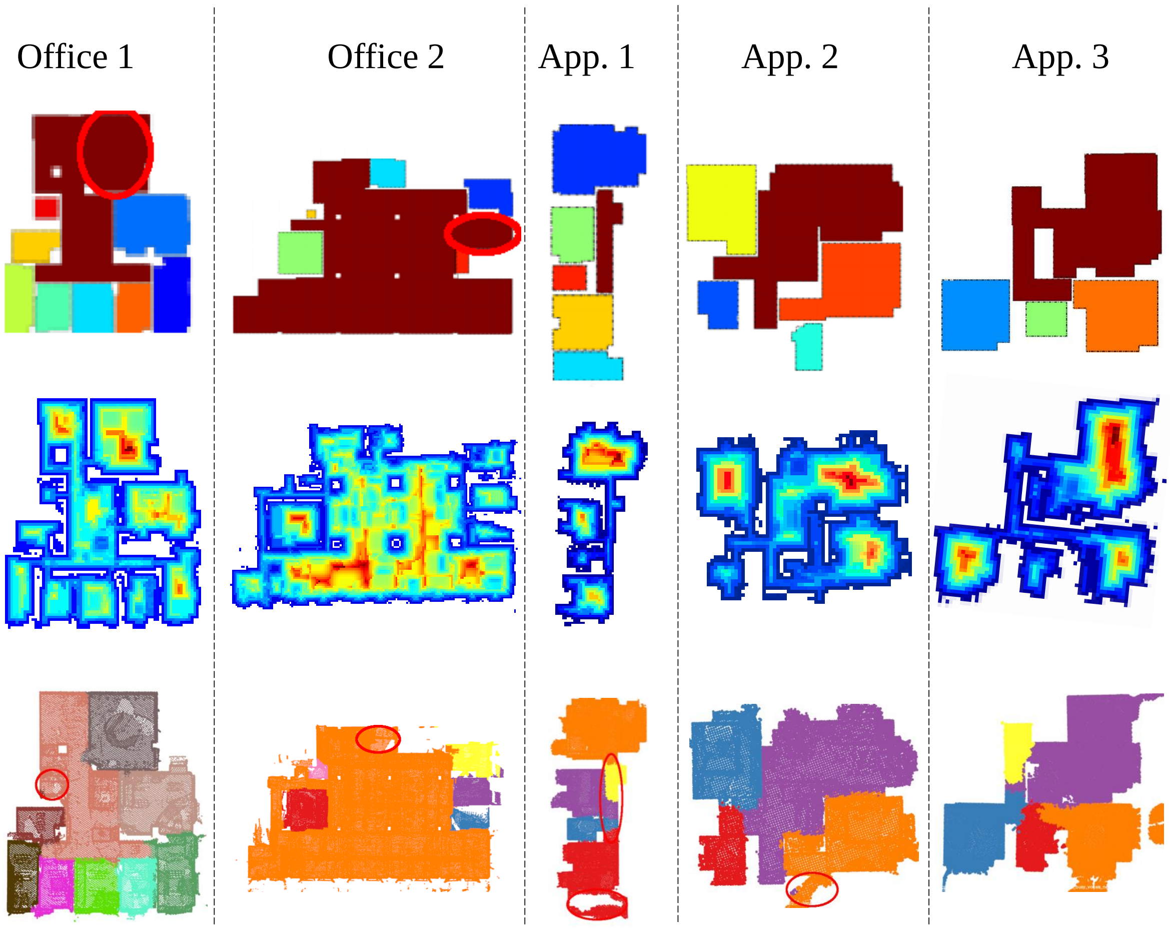


Figure 3.7: Results for the dataset of [14] for different buildings. From left to right column: Office 1, Office 2, Apartment 1, Apartment 2, Apartment 3. Top row: results of [14], middle row: PF map of the proposed method, bottom row: segmentation result for the proposed method. Here, in the top and bottom rows, different colors correspond to different room segments. In the middle row, low values of the PF map are shown in dark blue, whereas the high values as shown in red. Red ellipses denote erroneously segmented rooms. Reproduced from [5], ©2017 IEEE.

3.3.1.3 Results on the Dataset of Ikehata [14]

The results for room segmentation on the dataset of Ikehata *et al*. [14] are given in Fig. 3.7. It can be observed that the proposed approach outperforms [14] in several places: the room in the top-right part of Office 1 and the right part of Office 2. This is despite the fact that the information about scanner poses was not used as compared to [14]. Furthermore, parts of the outdoor space are also correctly segmented - see the bottom part of Apartment 1. The inferior segmentation performance of [14] on the rooms in Office 1 and 2 is due to the inadequate merging heuristics of free space voxels, which results in the erroneous merging of two rooms. Note that the proposed approach does not perform very well in the parts of the PC, where the PC data is very sparse, such as the left part of Apartment 1, the top part of Office 2 and the bottom part of Apartment 2 in Fig. 3.7. In such cases, the algorithm of [14] heavily relies on the scanner pose information to discard such voxels before segmentation, whereas

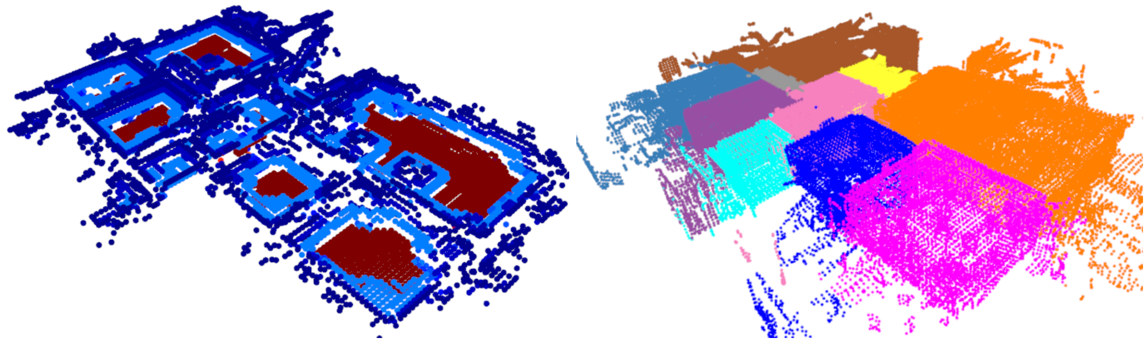


Figure 3.8: Illustration of room segmentation result (right) for the proposed method on the dataset of [4]. The corresponding PF maxima in 3D are shown on the left. It can be observed that the proposed method can correctly segment all rooms.

the proposed method does not require such information. Furthermore, the approach of [14] erroneously segmented two rooms, while the proposed approach incorrectly segmented five rooms.

3.3.1.4 Results on the Dataset of Bobkov [4]

Additionally, the proposed method has been evaluated on the dataset of Bobkov *et al.* [4]. It represents an office environment with vertical walls. A result of room segmentation for the proposed method is given in Fig. 3.8. As the source code of other room segmentation methods was not made available by the authors, only evaluation of the proposed method has been performed on this dataset. From the given results it is possible to see that the proposed method can segment all rooms correctly.

3.3.2 Discussion

The derived PF map as a volumetric signature of inner free space serves as a powerful descriptor for inner spaces. By performing the projection of the maximum PF value along each vertical voxel stack onto the 2D map, it becomes possible to significantly simplify the clustering problem. It has been observed that for most environments the PF information is the most important feature for clustering. Nonetheless, in certain cases, such as transitions between corridors (e.g., in the middle part of Area 1 in Fig. 3.6), it is essential to include visibility to detect changes of the spatial signature. Furthermore, in some situations, as in the case of detecting a long rectangular-shaped corridor, it can be disadvantageous to use visibility for clustering. This is because the corridor parts located far away from each other are not mutually visible, but they still belong to the same corridor. In contrast, PF values become more important in such situations. Furthermore, PF maps can provide a good illustration of the room layout, which can be helpful for visual inspection by humans. In order not to limit the generality of the algorithm, parameter tuning on a specific dataset has been intentionally omitted. Otherwise, practical usage of the algorithm would be limited as the algorithm parameters would need to be adjusted depending on the dataset.

It has also been observed that decreasing the voxel size leads to the effect that finer details on the objects located within rooms are captured. In particular, when using the voxel size of 5 cm, the PF map would reflect finer details and larger variations due to the small objects, e.g., lamps. As such objects are not directly correlated to the object boundaries, using smaller voxel size would only lead to deterioration of the room segmentation performance. A possible way to mitigate this issue is to adapt the values of the volumetric graph cut accordingly. Another important disadvantage of using smaller voxel size is a significantly increased computational complexity without any noticeable gains in the segmentation accuracy. The optimal values of the voxel size typically lie in the range $10-20$ cm for most indoor environments.

3.3.3 Limitations

The proposed room segmentation approach has average performance on very sparse PC data, as in this case, it becomes very challenging to estimate interior free space of the building. This can be mitigated by extending the criteria of the interior space, e.g., including priors regarding common orientations or dominant planes. Another significant limitation is low segmentation performance on long narrow corridors, where the method for volumetric signature estimation remains sensitive to clutter and objects. To mitigate this issue, an additional step of furniture recognition can be employed. This would allow removing such furniture from the interior space, thus improving the overall performance. Finally, the voxel grid requires large storage in case of buildings with substantial inner volume (foyer or atrium spanning large volume). To address this limitation, a voxel grid with adaptive voxel size can be used. Thus, the voxel size can be adapted to the level of details in corresponding regions, which would allow to segment rooms of significantly different sizes. By carefully adapting the voxel size it becomes possible to preserve the same level of room segmentation performance within different parts of the building while reducing computational complexity and required storage.

3.4 Chapter Summary

In this chapter, a novel framework for room segmentation in 3D PCs of indoor environments has been presented. It satisfies the requirements mentioned in Section 2.2.1 and addresses the unsolved challenges described in Section 3.1, such as arbitrary room layout, no requirements on the sensor pose information and buildings violating the Manhattan world assumption. To this end, a number of tasks have been solved, such as the computation of the interior free space of indoor environments without assuming a knowledge of scanner poses or the Manhattan world structure. By using a volumetric grid combined with the graph cut algorithm, it is possible to accurately estimate inner free space, hence no information on sensor poses is required. As the proposed voxel grid is formulated in 3D, buildings violating the Manhattan world structure are also supported. Based on the inner free space, it has become possible to derive a general volumetric signature using the anisotropic PF, which made the problem of room segmentation much easier. This signature leverages the intuitive definition of the room as inner free space separated from the other rooms via a free space with a different volumetric signature. Most importantly, the presented PF signature is robust to indoor clutter and occlusion that is common in indoor environments. The clustering algorithm operating on such a signature needs to be able to segment non-convex clusters, as the rooms can often represent non-convex shapes, as commonly observed for corridors of indoor environments. A good choice of such algorithm, supporting general room layouts and having a small set of hyper-parameters, is HDBSCAN density-based clustering. Furthermore, by using a parameter-free clustering method, it is not required to specify how many rooms within the building exist, as the algorithm can define this automatically. The source code of the proposed method has been made publicly available[1] to facilitate further progress in the scientific community.

[1] https://github.com/DBobkov/room_segmentation_icme2017

Chapter 4

Unsupervised 3D Object Segmentation in Point Clouds

In this chapter, the problem of unsupervised 3D object segmentation in PCs is described. Furthermore, an object segmentation method to address this problem is proposed. Finally, an experimental evaluation using a number of indoor PC datasets is presented and discussed.

Parts of this chapter have been published in [4].

4.1 Problem Statement

Once the room segmentation has been done, the next step in semantic understanding is to reason about the objects that are contained within the room (see Fig. 4.1). To extract semantic information about the objects, it is first necessary to segment 3D indoor scenes into objects. As discussed in Section 2.2.2, an object segmentation algorithm operating on large-scale data needs to be unsupervised - that is the data does not have to be annotated - and achieve low complexity without assuming abundant planar structures. This task remains challeng-

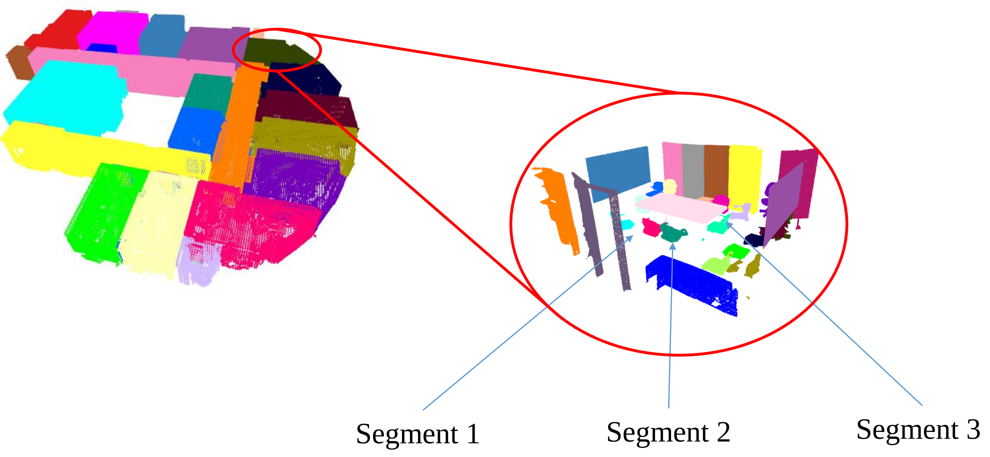


Figure 4.1: Once the PC data of the building has been segmented into rooms (left), one can process every room separately and perform the task of object segmentation (right), where the segments within the room that correspond to different objects need to be determined. Here, different colors denote different room and object segments in the left and right parts, respectively.

ing despite many years of research. Object segmentation operating on the data obtained using handheld depth sensors, e.g., Kinect RGBD sensor, is a well-studied research topic. These depth sensors are low-cost and flexible, hence they have been frequently used for indoor scanning [81], [82], [94]. Most existing datasets [58], [129] included only single-view data and focused on small environments (part of a room) due to the high effort involved in recording building-scale environments using handheld sensors. When capturing data in large indoor environments, the operational costs and time constraints become more important as the environment has to be free of moving objects during the time of scanning, e.g., people. Compared to Kinect-based solutions, laser scanners have a clear advantage in this context, as they provide a larger scanning range (typically more than 30 meters) and a wider viewing angle. With these systems, it is possible to scan an area of $10{,}000\ m^2$ within a day, which is practically impossible using any Kinect-like sensor due to limited range. A number of mapping platforms equipped with laser scanners have been developed using either a wearable backpack [20] or a moving trolley [21]. The sensors progressively take measurements and integrate them into a 3D model using a SLAM system, while the platform is moved through the indoor space.

Due to the specific scanning procedure required for large indoor environments, e.g., floors and buildings, multi-view PC (MVPC) data, acquired using a moving platform, tends to have certain drawbacks as compared to single-view data (see Section 2.2.2). State-of-the-art algorithms tend to exhibit inferior segmentation performance on MVPC data, as shown later in the experimental evaluation. To address these limitations, a new approach for object segmentation is proposed. In particular, the **contributions** of the approach are the following:

- A method to robustly detect and remove high-noise regions is proposed. This method helps to overcome limitations of state-of-the-art object segmentation algorithms that perform poorly on MVPC datasets due to the high noise and occlusion properties of such data. By removing such regions, the existing graph segmentation algorithms can be applied without major modifications, while showing superior segmentation performance.

- A new MVPC evaluation framework is presented. Within this framework, a new MVPC dataset was created that captures six indoor scenes (rooms) with multiple objects. Furthermore, it was augmented with ground truth annotation for objects and their parts to reflect multiple object scales. Finally, a new evaluation metric for under- and oversegmentation error is presented that considers objects and their relation to the object parts for the most accurate evaluation.

4.2 Method

To illustrate the improvements achievable with the proposed method, a typical unsupervised 3D object segmentation pipeline is considered as baseline [17]. Preprocessing on the input PC data (step 1 in Fig. 4.2) is performed to remove outliers and other artifacts (step 2 in Fig. 4.2). This is done in combination with normal and curvature calculations. For more details on

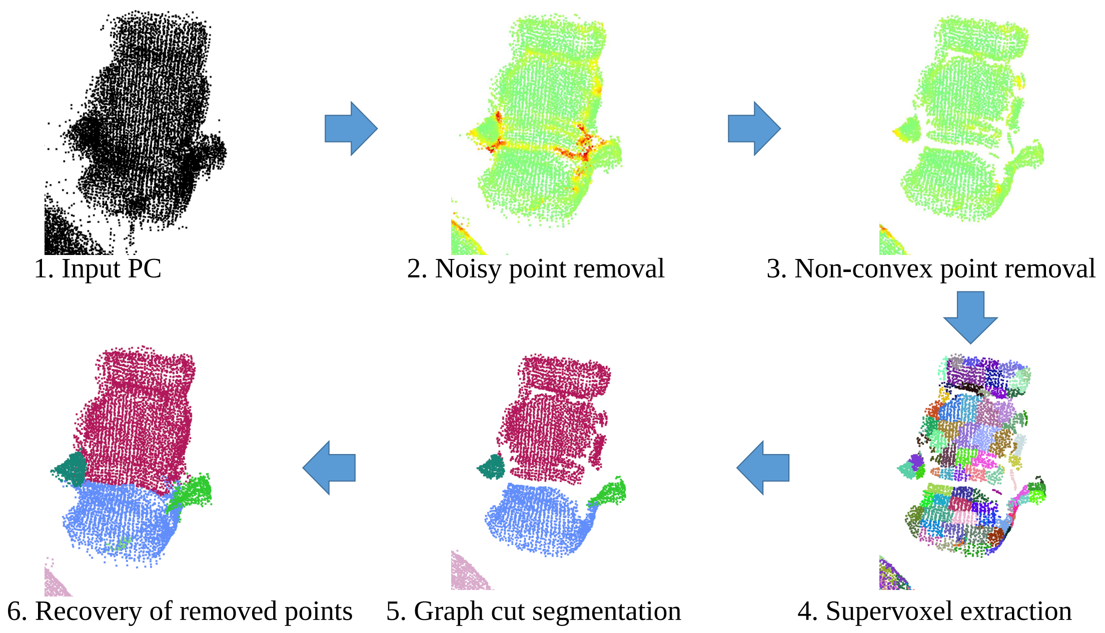

1. Input PC
2. Noisy point removal
3. Non-convex point removal
4. Supervoxel extraction
5. Graph cut segmentation
6. Recovery of removed points

Figure 4.2: Processing steps of the analyzed segmentation pipeline with the result of the corresponding step shown above it. Input PC (1) is first preprocessed. This is followed by a noisy point removal step (2). After that, the non-convex points satisfying the proposed concavity criteria are discarded (3). In the following, supervoxels are extracted from the remaining points (4). The supervoxels and the edges connecting them are used in the graph cut segmentation (5). Finally, the removed points are recovered (6). Here, in Subfigures (4-6) different colors correspond to different segments. In Subfigures 2 and 3, curvature values of the corresponding points are color coded as follows: low values are shown in green and high values in red. Reproduced from [4] with permission, © 2017 INSTICC.

preprocessing, see Section 2.1.5. Afterwards, the supervoxel (surface-patch) adjacency graph is computed based on the PC data, e.g., using the supervoxel clustering approach of [47] (see Section 2.1.6.6). The supervoxel graph is used to reduce the complexity of the input data (step 4 in Fig. 4.2). After this step, segmentation is performed on the given graph using a state-of-the-art graph partitioning method, e.g., the graph cut algorithm (step 5 in Fig. 4.2). For more details on the graph cut algorithm, see Section 2.1.6.3. It is proposed to augment the segmentation pipeline by the steps of curved non-convex point removal (step 3 in Fig. 4.2) and the recovery of the removed points (step 6 in Fig. 4.2). The details of each of these steps are presented, and the limitations of the state-of-the-art approaches are further discussed. The PCs considered in this work contain viewpoint information, that is the direction from which the range sensor has detected the corresponding point. For preprocessing (PC smoothing, curvature and normal estimation) the PCA-based method for normal estimation of Rusu *et al*. [130] was used, see Section 2.1.5 for more details.

4.2.1 Classification into Convex and Non-convex Points

Limitation of supervoxels and normals. In order to reduce the computational complexity, the points of the PC are grouped into segments using the supervoxel clustering algorithm of Papon *et al*. [47]. The algorithm partitions 3D PC data into surface patches called supervoxels (part 4 in Fig. 4.2). This algorithm can generate the clusters using the large-scale PC data, thus

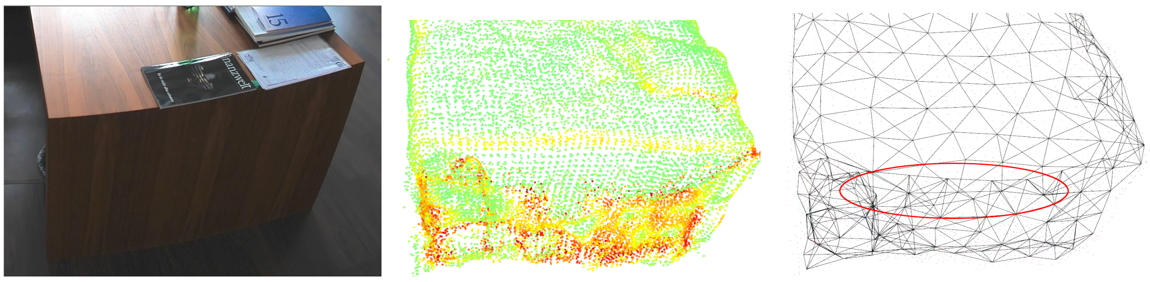


Figure 4.3: To illustrate the influence of noisy regions on the surface graph, a table region in the office in dataset [4] is shown. From left to right: RGB image, PC, surface graph. In the PC, points are color coded according to the corresponding curvature value: low values are shown in green and high value in red. Erroneous connections in the surface graph that result from noise in a planar region are encircled in red. After the non-convex region removal step, these noisy points in the planar region along the erroneous connections are removed. Reproduced from [4] with permission, © 2017 INSTICC.

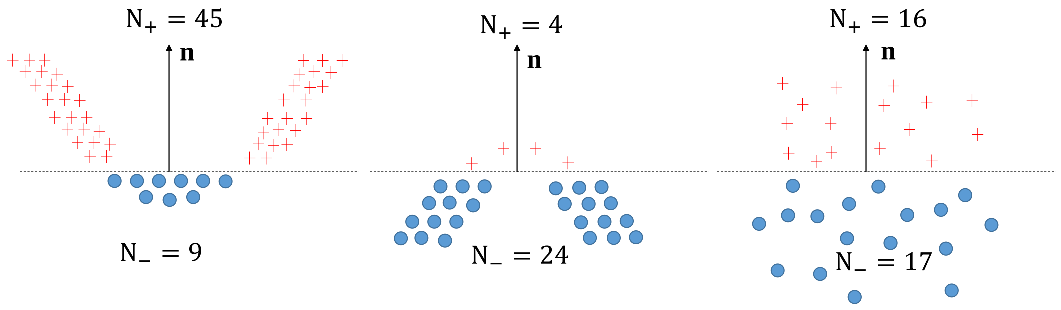


Figure 4.4: Illustration of high-curvature regions with normal **n** and the neighboring points. The points located in the positive half-space (same as the normal vector) are shown in red, whereas the points located in the negative half-space are shown in blue. Left: concave region. Middle: convex region. Right: ambiguous region. The numbers N_+ and N_- indicate the number of points in positive and negative half-space, respectively. By using this number, convex, concave and ambiguous regions can be easily distinguished. Adapted from [4], ©2017 INSTICC.

reducing the volume and complexity of the data, while keeping the most important information intact. Depending on the size of the desired cluster, 5–100 points can be replaced with one such supervoxel. The supervoxels are designed not to span boundaries across objects. In practice, this is not the case for noisy, highly-curved, concave regions, which often coincide with object boundaries. This effect is illustrated in Fig. 4.3. Note the false connections (circled in red) in the concave high-curvature region within the table side in the middle of the scene. In case all high-curvature regions are simply removed, many important intra-object connections would be discarded as well. Hence, the segmentation performance would be severely deteriorated. This effect is not specific to supervoxels only and also occurs in other patch-based surface representations, as surface estimation is negatively influenced by noise. For more details on the clustering algorithm, see Section 2.1.6.6.

Noise-resilient convexity/concavity criterion. In order to cope with the aforementioned inferior performance in the presence of noise, a novel convexity/concavity criterion is derived. It employs the PC statistics and is robust to noise in the normal estimation. The convexity criterion is defined for a given point **p** and its radius neighborhood $\mathbb{N}(p)$, as explained in Section 2.1.5. In particular, for the given point $\mathbf{p} = (p_x, p_y, p_z)^T$ and the corresponding normal vector $\mathbf{n} = (n_x, n_y, n_z)^T$ (see Fig. 4.4), it is possible to define a plane having the same normal vector as $\mathbf{n}$ and containing point $\mathbf{p}$. The plane equation is given in Hessian normal form [131] as:

$$n_x \cdot x + n_y \cdot y + n_z \cdot z + d = 0, \tag{4.1}$$

where d is the distance to the origin and can be computed from Equation (4.1) by plugging in p_x, p_y, p_z instead of x, y, z into this equation. This tangent plane divides the 3D space into two half-spaces. To determine in which half-space a particular point is located, Equation (4.1) is used. In particular, if the value of d computed from this equation is positive it indicates that this point is in the positive half-space, or negative otherwise. By analyzing convex and concave neighborhood regions, the points within R are typically located within the same half-space as the normal direction $\mathbf{n}$ for the concave regions, and for convex ones in the other half-space. The number of points within each half-space is compared to each other in order to determine whether the given point neighborhood $\mathbb{N}$ is non-convex. If it is indeed non-convex, it will be removed according to the following equation:

$$m(\mathbf{p}, \mathbb{N}) = \begin{cases} \text{true} & \text{, if } N_+ \geq \alpha_t \cdot N_- \\ \text{false} & \text{, if } N_+ < \alpha_t \cdot N_-, \end{cases} \tag{4.2}$$

where N_+ is the number of points in the neighborhood $\mathbb{N}$ lying within the same half-space as the normal vector of the local surface $\mathbf{n}$, and N_- is the number of neighboring points lying within the opposite half-space and α_t is the threshold to detect noisy regions. It lies in the range between 0 and 1. The choice of the value of α_t is illustrated in Fig. 4.4. Concave and ambiguous regions need to be removed for the best segmentation performance. In contrast, convex regions have to be preserved as they belong to the object. For $\alpha_t = 0.1$, the convex region in Fig. 4.4 is classified as non-convex and removed. Its removal leads to oversegmentation of the object. For $\alpha_t = 1.0$, the ambiguous region (that mostly consists of measurement noise) will be classified as convex and thus is preserved. Experiments on laser scanner data indicate that with $\alpha_t = 0.2$ such regions are correctly classified for the indoor datasets (see further results in Section 4.3.4). The point neighborhoods satisfying the non-convex condition in Equation (4.2) and with a curvature value $\theta > \theta_t$ will be temporarily removed for the following processing steps (see Fig. 4.2). From this moment on, concave and ambiguous regions are denoted as non-convex for the sake of simplicity.

4.2.2 Supervoxel Clustering and Graph Partitioning

After noisy, high-curvature, non-convex regions have been removed, edge weights between neighboring supervoxels can be adequately computed. To reduce the complexity and volume of the PC data, the supervoxel clustering algorithm of Papon *et al.* [47] is used. In particular, supervoxel $\mathbf{p_i} = (\mathbf{x_i}, \mathbf{n_i}, \mathbb{E}_i)^T$ is considered, with the centroid $\mathbf{x_i}$, the normal vector $\mathbf{n_i}$ and the edges to adjacent supervoxels $e \in \mathbb{E}_i$. It is not desired to strictly enforce the condition of concave object boundaries, as no boundaries might be present in such areas.

Instead, concavity has to serve as just one indicator for the object boundary combined with the Euclidean distance and surface normals, which still provide evidence for object boundaries in case of non-concave regions. Therefore, the graph edge weight between neighboring supervoxels $\mathbf{p_1}$ and $\mathbf{p_2}$ is computed as follows:

$$w_e(\mathbf{p_1}, \mathbf{p_2}) = \begin{cases} a \cdot D(\mathbf{p}_1, \mathbf{p}_2)^2 & \text{, if convex edge,} \\ D(\mathbf{p}_1, \mathbf{p}_2) & \text{, if concave edge.} \end{cases} \tag{4.3}$$

$D(\mathbf{p}_1, \mathbf{p}_2)$ is the definition of the edge weight described in [47]. This weight equation, however, was implemented differently in the Point Cloud Library:

$$D(\mathbf{p_1}, \mathbf{p_2}) = \frac{\|\mathbf{x_1} - \mathbf{x_2}\|_2}{R_{seed}} \cdot w_s + (1 - |\cos(\mathbf{n_1}, \mathbf{n_2})|) \cdot w_n, \tag{4.4}$$

where $\|\mathbf{x}_1 - \mathbf{x}_2\|_2$ is the Euclidean distance between two nodes (centroids of supervoxel patches), R_{seed} is the seed radius, $\mathbf{n}_1$ and $\mathbf{n}_2$ are normals of the corresponding supervoxels. $\cos(\mathbf{n}_1, \mathbf{n}_2)$ describes the angle between the corresponding normals. w_s and w_n are spatial and normal weights, respectively. Note that the color difference term is omitted, compared to the original formulation, as it does not necessarily improve the segmentation results. This effect was also observed in [82]. This is due to the registration inaccuracies between RGB images and PC data. The parameter a denotes the weight of concavity criterion when objects are partitioned. A lower value for a increases the weight of the concavity criterion in the segmentation process. Based on the experimental results it has been observed that in order to segment various objects a trade-off needs to be achieved when selecting a. Hence, the value of a is set to 0.25 for all experiments. It can be seen from Equation (4.3) that for similar weights, concave edges are preferred as object boundaries to convex ones. Nonetheless, in case a convex edge connects two remotely located regions with drastically different surface properties, the spatial and normal distances can serve as evidence for object partitioning. Similarly to [17], $R_{seed}/R_{voxel} = 4$ is chosen for all datasets, where R_{voxel} is the voxel radius. In the experiments, the parameters are set as follows: $w_s = 0.2$ and $w_n = 0.5$ for all datasets. This choice of parameter values is due to the fact that normal information is more characteristic when describing the surface geometry as compared to the Euclidean distance. The latter is subject to sensor noise that affects point coordinates.

When partitioning the extracted graph for scenes with complex geometry, it has been observed that simple region growing algorithms do not perform well. Therefore, an adaptive statistics and graph-based segmentation algorithm Felzenszwalb-Huttenlocher [38] is employed (step 5 in Fig. 4.2). This method has the advantage that it does not require to set the number of clusters in advance, i.e., the algorithm derives this value from the data directly. Furthermore, this algorithm has low computational complexity. For more details, see Section 2.1.6.3. In contrast, other graph partitioning algorithms, such as spectral clustering and normalized min-cut [127] do not achieve such a good trade-off between accuracy and speed. Furthermore, the latter algorithms require to manually set the number of clusters, which is often a non-trivial task for complex indoor environments with multiple objects.

4.2.3 Recovery of Previously Removed Noisy Non-convex Points

After the noisy non-convex points have been removed in step 3, they need to be recovered in a later step. These points need to be assigned to correct segments based on the segmented preserved points. While assigning discarded points, it is important to determine similar points in the near vicinity that have already been assigned to a label. Especially, the local surface geometry is informative for this purpose. Hence, the graph edge weight defined in Equation (4.4) is used as a similarity metric. It has been observed that simple region growing algorithms based on seeds (i.e., known segments) are sensitive to outliers, which often occur at such highly-curved regions. To overcome this problem, the number of propagated segments is constrained to a certain value per iteration. Furthermore, the algorithm starts with processing the connections with lower weights as such weights exhibit a higher similarity. After this step, the distance metric is computed for each point in the radius R_{voxel}. During one iteration, the number of currently recovered points is limited to a percentage $P_r = 80\%$ of presently unassigned points that have assigned neighbors. It has been experimentally found that $S = 20$ such iterations are sufficient to recover non-convex points (observe an example of the restored segments in Fig. 4.2). The pseudocode for the algorithm is given in Algorithm 1. In line 6 of Algorithm 1, $AssignedPointsWithinRadius(P, R)$ returns assigned neighboring points around P within the search radius R. In line 10, $AssignmentOf(P)$ returns the point assignment to one of the clusters. $\lfloor\rceil$ denotes a rounding operation of the floating number to the closest integer. Note that W denotes a triplet consisting of a point, a corresponding distance, and a weight.

4.3 Experimental Evaluation

In this section, a laser scanner PC dataset, its semantic annotation, and a new multi-scale evaluation metric are presented. This is followed by the quantitative evaluation of this dataset. Furthermore, experimental results for a number of Kinect datasets are also provided. The results are compared to state-of-the-art geometry-based unsupervised segmentation algorithms of Locally Convex Connected Patches (LCCP) [17] and Van Kaick *et al.* [85]. For this, the publicly accessible algorithm implementations provided by the authors are used. For some of Kinect datasets, the results of other algorithms are also provided for comparison, when available.

4.3.1 Laser Scanner Dataset and Evaluation Metric

For rigorous evaluation of segmentation approaches and due to the lack of publicly available MVPC datasets, a novel MVPC dataset is presented. For an illustration of the building within the dataset, see Fig. 4.5. The PC data was acquired using a mobile mapping platform[1] with three Hokuyo UTM-30LX laser scanners. While the analysis can be done on the PC data acquired from any range sensor, laser scanners are advantageous in this regard as they

[1] NavVis M3 mapping trolley https://www.navvis.com/m3. Accessed: 2018-12-20.

Algorithm 1: Assign Removed Non-convex Points

```
Input  : Assigned points Q, unassigned points U
Output: Assigned points Q
1  Q ← $assigned\ points$
2  U ← $unassigned\ points$
3  S ← 20
4  for $s = 0$ to $S - 1$ do
5      W ← {} /*Init to empty set*/
6      for $P_n \in U$ do
           /*for each non-assigned point*/
7          M ← $AssignedPointsWithinRadius(P_n, R_{Voxel})$
8          if $M \neq \emptyset$ then
               /*if there exist assigned points within the given radius to the point*/
9              $j_{min} \leftarrow \arg\min_{\forall M_j \in M} D(P_n, M_j)$ /* Find closest point */
10             $D_{min} \leftarrow D(P_n, M_{j_{min}})$ /*Compute edge weight according to
               Equation (4.4) */
11             $L_{min} \leftarrow AssignmentOf(M_{j_{min}})$ /*Obtain label*/
12             $W \leftarrow W \cup \{M_{j_{min}}, D_{min}, L_{min}\}$ /*Append this point to the considered
               set*/
13             if $s \neq S - 1$ then
14                 Sort W with ascending order of D
15                 $N_{Preserve} \leftarrow \lfloor Length(W) \cdot P_r \rceil$
16                 for $i = 0$ to $N_{Preserve} - 1$ do
17                     $Q \leftarrow Q \cup W_i$ /*Mark this point as labeled*/
                   end
               end
           end
       end
   end
18 Return Q
```

offer a fast acquisition procedure in large indoor environments. As the mapping platform was moved through the indoor environment, its laser scanners performed range measurements in one horizontal and two intersecting vertical planes, thus incrementally building a 3D map. The average scanning time per room constituted several minutes. The captured scenes represent typical office environments with various objects.

4.3.1.1 Dataset Annotation

For evaluation purposes, six indoor scenes were manually labeled. Before object labeling, RANSAC-based plane segmentation was employed to remove architectural parts of buildings, such as walls and floor. Furthermore, due to the rather coarse resolution of PCs, small objects that are not distinguishable from noise were not labeled, e.g., the pen lying on the table. When labeling, some may regard a chair as a whole object, while others may regard it as a collection of parts, such as a chair back, chair leg, etc. It is unclear which of these label-

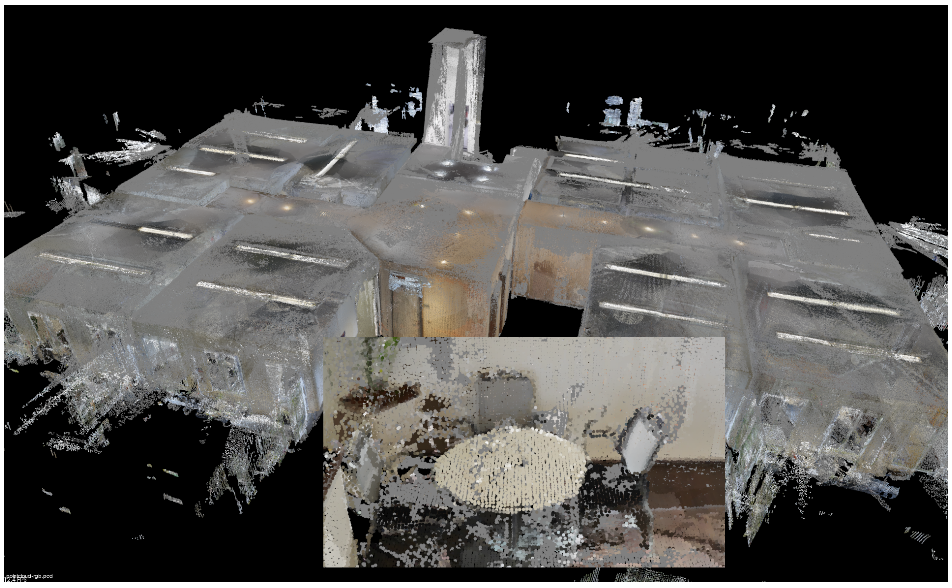


Figure 4.5: Illustration of the PC showing one floor of an office environment of the captured laser scanner dataset (see Section 4.3.1.1). Zoomed in area of a specific room is shown in the bottom. It can be observed that there is a significant amount of noise and occlusions resulting from the rapid mapping procedure and reflections from glass.

ings is correct. To address this ambiguity, labeling was considered on several object levels, e.g., fine and coarse ground truth (GT). Fine GT includes object parts, while coarse GT corresponds to objects themselves. The object categories within the considered scenes correspond to *chair*, *table*, *television*, *wardrobe*, *whiteboard* and similar objects that are commonly observed in office environments. See Fig. 2.18 for an illustration of this annotation result. The total number of objects is 156, which contain 452 semantic object parts (e.g., *chair* contains such parts: *chair back*, *leg*, *arm*, *seat*). For illustration of the object PC data, see Fig. 4.6.

4.3.1.2 Multi-scale Evaluation Metric

For the evaluation of the segmentation results, Richtsfeld *et al.* [53] proposed undersegmentation (US) ME_{US} and oversegmentation (OS) ME_{OS} errors (described in Section 2.1.8.1). The main limitation of this metric is that it considers only a single scale of the segment. This would result in an incorrect evaluation of the given labeling. To address this limitation, a multi-scale extension to the metric of Richtsfeld *et al.* is proposed. The new metric evaluates the segmentation result not only concerning an object but also with respect to its parts so that the most appropriate scale of GT is taken into account. In particular, the proposed multi-scale GT evaluation approach works as follows. First, each meaningful object part is labeled as a separate semantic object within fine GT. In order to generate coarse GT, object parts that are semantically related are merged into one segment. For this, a specific approach to establishing correspondences between points and appropriate GT segments is given in Algorithm 2. As input to the algorithm, several mapping data structures are provided: IL describes a mapping from points to fine GT segments. FC specifies a mapping from fine to coarse GT segments. FP describes correspondences from fine GT segments to

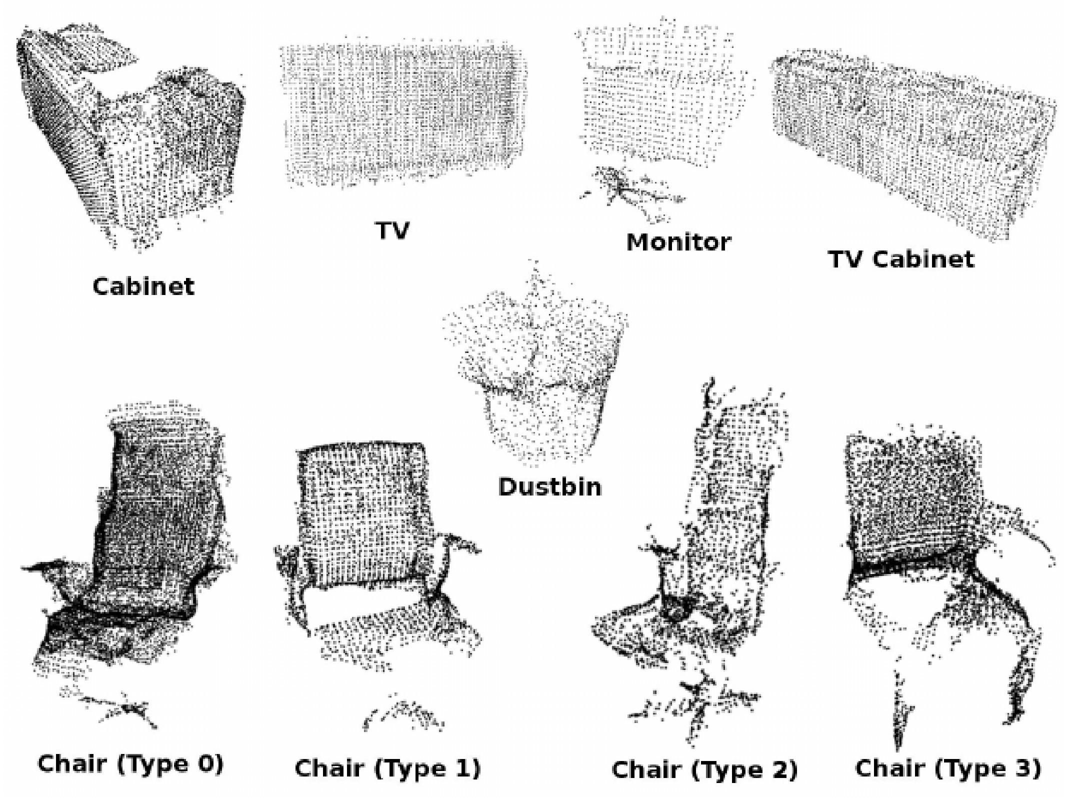


Figure 4.6: Illustration of object appearance for different categories in the annotation of the captured laser scanner dataset. The PC data exhibits significant level of occlusion and undersampling due to the rapid scanning procedure. Chairs have large intra-class geometry variance.

predicted segments, where a predicted segment assignment corresponding to a fine segment is the majority of predicted assignments of this part. The important task is to establish correspondences between points and appropriate fine GT (possibly merged) segments, which are returned in the form of the map IM after the algorithm's execution.

A simple example of input data for Algorithm 2 is given in Fig. 4.7. Here, 10 points need to be evaluated according to the given predicted segments, coarse and fine GT data. FM and IM are empty maps before the algorithm's execution. After the algorithm has finished, the FC and FP have not changed (see Fig. 4.7). Furthermore, the returned structure IM describes a mapping from point indices to merged fine segments.

Now, when proper GT data has been generated, the metric of Richtsfeld *et al.* [53] can be directly extended to multi-scale objects. The multi-scale OS and US errors can be calculated as follows:

$$ME_{OS} = 1 - \frac{\sum_{i=1}^{n} PT_i}{\sum_{i=1}^{n} M_i}, \tag{4.5}$$

$$ME_{US} = \frac{\sum_{i=1}^{n}(P_i - PT_i)}{\sum_{i=1}^{n} M_i}, \tag{4.6}$$

where M_i is the number of points with the ith merged fine segment. P_i is the number of points with the predicted segment that after the mapping from merged fine segment to predicted segment is corresponding to the ith merged fine segment. Furthermore, PT_i is the

Algorithm 2: Generate GT Segments of the Correct Scale

```
Input  : Map $IL$ $< PointIndex, FineLabel >$
Map $FC$ $< FineLabel, CoarseLabel >$
Map $FP$ $< FineLabel, PredLabel >$
Output: GT segments $IM < PointIndex, MergedFineSegment >$
1  $keyArray \leftarrow keys(FP)$
2  $K \leftarrow keyArray.size$
3  Map $FM$ $< FineL, MergedFineL >$
4  Map $IM$ $< PointId, MergedFineL >$
5  $flagArray \leftarrow new\_array(K)$
6  $flagArray \leftarrow 0$
7  for $k = 0$ to $K - 1$ do
8      if $flagArray[k] == 0$ then
9          for $j = k + 1$ to $K - 1$ do
10             $Flag1 \leftarrow (FP[keyArray(k)] == FP[keyArray(j)])$
11             $Flag2 \leftarrow (FC[keyArray(k)] == FC[keyArray(j)])$
12             $RFlag \leftarrow Flag1 \wedge Flag2$
13             if $RFlag$ then
14                 $flagArray[j] \leftarrow -1$
15                 $FM[keyArray(j)] \leftarrow keyArray(k)$
16 for $p \in keys(IL)$ do
17     if $IL[p] \in keys(FM)$ then
18         $IM[p] \leftarrow FM[IL[p]]$
       else
19         $IM[p] \leftarrow IL[p]$
20 Return: $IM$
```

number of correctly assigned point segments within the ith merged fine segment. Finally, n is the number of distinct merged fine segments.

For the example given in Fig. 4.7, the values of OS and US errors can be computed as follows:

$$ME_{OS} = 1 - \frac{5+1+2+2}{5+1+2+2} = 0, \tag{4.7}$$

$$ME_{US} = \frac{(6-5)+(6-1)+(2-2)+(2-2)}{5+1+2+2} = 0.6. \tag{4.8}$$

4.3.2 Results on the Laser Scanner SLAM Dataset

The quantitative results are presented for each scene in Table 4.1. Here, the results for the two aforementioned algorithms (LCCP and Van Kaick), as well as a combination of the proposed criterion (non-convex region removal and recovery) with the LCCP segmentation algorithm ("Proposed+LCCP") are given. For all laser scanner dataset scenes, the same parameters for the proposed method were used, in particular $R_{seed} = 12cm$, $C = 3$, $\theta_t = 0.03$, $k = 3$, thus no parameter tuning for a particular scene was performed. For LCCP, the same R_{voxel} and

R_{seed} were used, while other parameters were described in [17]. In particular, the concavity tolerance angle was set to $10°$, filter number n_{filter} was set to 3, and the threshold angle was set to $60°$.

The illustrations of the segmentation results on scenes 1 and 3 along GT data for the analyzed algorithms are provided in Fig. 4.8. Further segmentation results on scenes 2 and 4 are given in Fig. 4.9. Finally, segmentation results on scenes 5 and 6 are given in Fig. 4.10.

It can be seen for scene 3 (the right column of Fig. 4.8) that LCCP oversegments the objects behind the cupboard in the upper part of the scene (zoomed), whereas the proposed method correctly segments such parts, and thus has a lower OS error. Furthermore, for scene 1 (left column in Fig. 4.8), the LCCP and Van Kaick *et al*. methods oversegment the tables in the right part and the left part fo the scene. In contrast, the proposed method segments the tables correctly.

Observe for scene 4 (the right column of Fig. 4.9) the case when LCCP segmentation performance deteriorates due to noisy normals. The high US error of LCCP stems from the fact that the algorithm has merged the chair in the top part of the scene with the table. The method of Van Kaick *et al*. [85] also shows limited performance on partitioning the table from the adjacent chairs. In contrast, the proposed method has produced better results by separating the chairs from the table. The limited LCCP performance on this scene is mostly due to noisy normals and low-density regions in the neighborhood of chairs. The method of Van Kaick *et al*. is limited on such scenes as it cannot handle sparsity in the PC data. On the other hand, the proposed method is more robust with respect to such regions. Furthermore, LCCP, as well as the proposed method, have oversegmented the left part of the scene containing kitchen cupboards and objects on the table. Finally, in the right part of the scene, both algorithms are unable to correctly segment the corner table, thus increasing OS error. It can be seen for scene 2 (the left column of Fig. 4.9) that due to the sparse PC data in the table region, none of the algorithms can correctly segment the table into a single region. Moreover, the proposed approach correctly segments the chairs, whereas LCCP and Van Kaick *et al*. oversegment them into non-meaningful parts.

When analyzing the performance on scene 5 (the left column of Fig. 4.10), it is possible to see that all of the algorithms oversegment the table and chair regions due to sparsity in the bottom right. For scene 6 (the right column of Fig. 4.10), the proposed method can correctly segment the table and chair area, as compared to LCCP and Van Kaick *et al*. methods.

From the quantitative results in Table 4.1, it is possible to observe that the proposed algorithm significantly outperforms LCCP as well as the approach of Van Kaick *et al*. [85] for both multi-scale US and OS errors. In particular, the method of Van Kaick *et al*. [85] shows particularly high US error on scene 6, which is due to oversegmentation of the chair region. Also note that the proposed convex/concave criterion combined with LCCP algorithm ("Proposed+LCCP") gives a clear improvement in segmentation accuracy, as compared to LCCP only. This justifies that the proposed noisy point removal criterion makes a significant contribution to the improved segmentation accuracy. By including a robust weight metric and more sophisticated graph segmentation algorithm, it is further possible to improve the performance by ca. 7% and 14% of OS and US errors, respectively (refer to mean OS and

Table 4.1: Performance comparison of the segmentation methods on the laser scanner dataset. The multi-scale over- and undersegmentation errors are used as error metrics. The top value is ME_{OS} and the bottom value is ME_{US}. Bold entries indicate best performance per scene. It can be observed that the proposed method outperforms the existing methods on all scenes. Moreover, a combination of the proposed concavity criterion with LCCP leads to an improvement as compared to LCCP only. This confirms that the proposed convexity criterion can be combined with existing methods in a modular fashion leading to superior segmentation performance. Reproduced with permission from [4], ©2017 INSTICC.

Scene	Proposed	LCCP [17]	Proposed+LCCP	Van Kaick [85]
1	**15.3%**	35.8%	23.8%	37.1%
	5.6%	12.2%	6.5%	16.3%
2	**6.2%**	30.4%	20.2%	25.6%
	0.6%	9.0%	6.1%	23.6%
3	**10.9%**	20.5%	17.3%	17.7%
	8.9%	9.7%	**6.1%**	78.7%
4	**8.8%**	18.8%	11.0%	32.9%
	17.5%	143.7%	88.3%	647.8%
5	**6.6%**	29.6%	22.3%	27.8%
	8.7%	23.2%	**4.3%**	80.8%
6	**15.1%**	21.2%	17.7%	37.6%
	12.5%	36.9%	28.8%	104.4%
Mean	**11.4%**	26.0%	18.8%	29.8%
	8.9%	39.1%	23.3%	158.6%

US errors for LCCP and "Proposed+LCCP" columns in Table 4.1). The average processing time of the proposed approach is less than 10 s per scene containing ca. 100,000 points on a modern desktop computer with Quadcore i7 with 16 GB RAM.

4.3.3 Results on Kinect Datasets

For benchmarking purposes, the evaluation is also performed on a number of Kinect datasets. As no multi-scale GT data is available, single-scale GT is considered. As an evaluation metric, US and OS errors are used. Furthermore, the weighted overlap metric is also employed. For more details on these evaluation metrics, see Section 2.1.8.1.

Object Segmentation Dataset. It is a single-view small-scale depth image-based object dataset, also denoted as Object Segmentation Dataset (OSD) (adopted from [53]). It was captured with a Kinect sensor in a table-top setting. The OSD dataset contains 111 scenes. For each of them, the depth image, the RGB image, and the GT annotation are provided. The segmentation results are shown in Fig. 4.11. The quantitative results for the proposed method along LCCP and Richtsfeld *et al*. [53] method are given in Table 4.2. It can be observed that the method of Richtsfeld *et al*. achieves the best OS error at the cost of undersegment-

Table 4.2: Performance comparison of different segmentation methods on the OSD dataset with respect to under- and oversegmentation errors (smaller is better). It can be observed that the method of Richtsfeld *et al.* achieves the best OS error at the cost of undersegmenting the scenes, which leads to higher US error. The method of LCCP achieves the average performance of all three methods. Finally, the proposed method strikes a trade-off in terms of OS and US errors. Reproduced with permission from [4], ©2017 INSTICC.

Method	Learned features	F_{OS}	F_{US}
Proposed	No learning	6.8%	**2.6%**
LCCP [17]	No learning	7.4%	4.7%
Richtsfeld *et al.* [53]	RGBD and geometry	**4.5%**	7.9%

Table 4.3: Performance of segmentation methods on the NYU dataset using weighted overlap (WO) (larger value is better). The proposed method achieves reasonable performance in spite of being learning-free, as compared to the other learning-based methods. Reproduced with permission from [4], ©2017 INSTICC.

Method	Learned features	WO
Proposed	No learning	58.0%
LCCP [17]	No learning	57.6%
Silberman *et al.* [58]	Depth	53.7%
Gupta *et al.* [93]	RGBD	62.0%

ing the scenes, which leads to higher US error. The method of LCCP achieves the average performance of all three methods. From Fig. 4.11 it is possible to observe that the proposed method performs erroneous segmentation for the top-right part of the scene in the book area. Furthermore, the cup is oversegmented into two areas due to concavity. Finally, the fully occluded book is oversegmented into two parts, even though it represents one object. It is important to note that this is a particularly challenging task for any method using geometry only.

NYU Dataset. For further benchmarking, the algorithms were also evaluated on a much larger Kinect dataset, as compared to OSD. For this, the single-view NYUv2 Kinect dataset [58] was used. This dataset contains 1449 scenes with realistic cluttered conditions, captured from a single viewpoint. Quantitative evaluation on 654 test scenes is provided in Table 4.3. Parameter values are set as follows: $\theta_t = 0.02$, $C = 3$, $k = 5$, $R_{seed} = 16cm$ for all scenes. For comparison, the performance of LCCP and training-based methods of [58] and [93] are also provided. The proposed method achieves reasonable performance despite being learning-free, as compared to [93].

Washington Dataset. For the sake of extensive evaluation, the Kinect dataset covering larger indoor scenes was also included. For this, the non-annotated Washington dataset of Lai *et al.* [59] was used. Due to the absence of annotation, only qualitative evaluation was done. The dataset has been captured in indoor environments spanning multiple rooms and contains 300 objects organized into 51 categories. The illustration of segmentation results is given in Fig. 4.12. The proposed method achieves good segmentation performance by accurately segmenting objects on the table. The major segmentation mistakes correspond to the floor area of scene 14 and the wall area of scene 6. This is due to the curved surface properties, which serve as an indication of the object boundary.

SUN Dataset. Even though the SUN dataset of [129] has been commonly used in a number of related works, multiple issues with the PC data quality have been observed. In particular, there are multiple registration inaccuracies across different scans. These inaccuracies accumulate and result in the total PC data that is very noisy. An illustration of these effects is given in Fig. 4.13. Due to these reasons, it has been decided not to perform evaluation on this dataset.

4.3.4 Parameter Sensitivity

It is important to study the influence of the parameter values on the resulting segmentation performance, the so-called parameter sensitivity. In particular, an illustration of the influence of the curvature threshold θ_t on the number of inter-object vs. intra-object edges that are removed for the laser scanner dataset is shown in Fig. 4.14. By choosing its value in the range 0.02 to 0.03 a significant number of inter-object connections are removed (68.31%), whereas most of the intra-object connections are preserved (77.21%). This allows to significantly simplify the segmentation task while achieving even better performance. Please note that α_t value has also been varied. From Fig. 4.14, the value of $\alpha_t = 0.2$ results in the best performance for the laser scanner dataset. Due to a marginal improvement, parameter k is set to 3 for all scenes in the laser scanner dataset. The parameter k offers the trade-off between US and OS errors, in particular, higher k would result in lower OS and higher US errors, respectively. In case the low US error is more important, the parameter k has to be reduced. The parameter for graph partitioning C has to be chosen jointly with seed resolution R_{seed}, depending on the desired size of the smallest segment. Finally, R_{seed} should be greater than the average point cloud resolution, as indicated in [47].

4.3.5 Limitations

Removing high-curvature, non-convex regions can, sometimes, result in the situation that the regions become too sparse. Therefore, no connections within the object remain. This, apparently, will lead to erroneous oversegmentation of the object. To mitigate this issue, it is possible to keep such low-density regions intact by using machine learning techniques for a data-driven criterion. Furthermore, it is essential to acknowledge the simplicity of the used criterion of a concave edge, which can fail in some cases (e.g., TV set in scene 1 in Fig. 4.8). Moreover, it is clear that in case RGB information is accurately registered to the PC data, RGB modality can be crucial to improve segmentation performance. Finally, the captured dataset can be compared to the recently presented large datasets containing multi-scale annotation, e.g., PartNet [132].

4.4 Chapter Summary

A new approach for object segmentation in multi-view indoor PCs has been presented. To address the particular properties of the MVPC datasets, such as non-uniform point density and high levels of noise, a novel noise-resilient criterion for the detection of noisy non-convex regions has been proposed. This step makes the graph partitioning (and thus segmentation) problem simpler and reduces the number of erroneous connections that appear due to noise. By combining the proposed point removal step with state-of-the-art segmentation algorithms, their segmentation performance can be significantly improved. Despite the fact that the algorithm has been designed for MVPC data, it also achieves state-of-the-art performance on single-view PC data. For realistic evaluation, a new laser scanner dataset along with multi-scale object annotation was presented. The MVPC dataset helped to experimentally illustrate that there is a discrepancy between single-view and multi-view PCs in terms of noise level, especially at high-curvature regions. The proposed laser scanner dataset spans 6 rooms within an office environment and contains 452 object parts. Even though there are a number of larger public indoor PC datasets available, this dataset can still be valuable to the scientific community as the moving laser scanner-based approaches are particularly suitable for the rapid mapping and 3D reconstruction of large indoor environments. The source code of the proposed evaluation metric and the labeled dataset have been made publicly available[2].

[2] https://github.com/DBobkov/segmentation

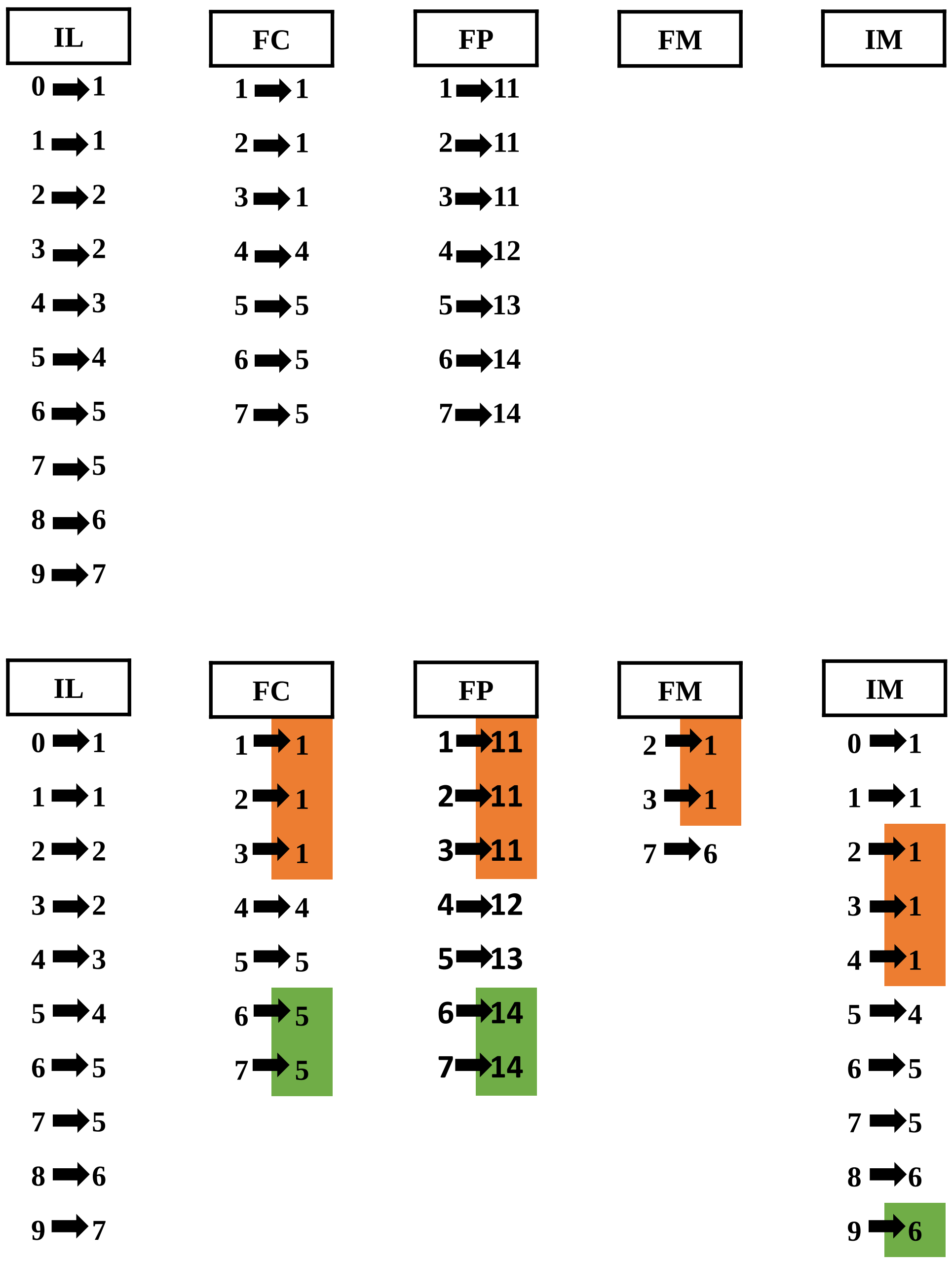


Figure 4.7: Illustration of values within input and output variables for the proposed algorithm to find GT segments of the correct scale. Top: input data. Bottom: output data after the algorithm's execution. The merged fine segments of IM and the corresponding point groups are highlighted in color. It can be observed that the proposed evaluation metric is able to correctly compute a suitable object scale for accurate evaluation.

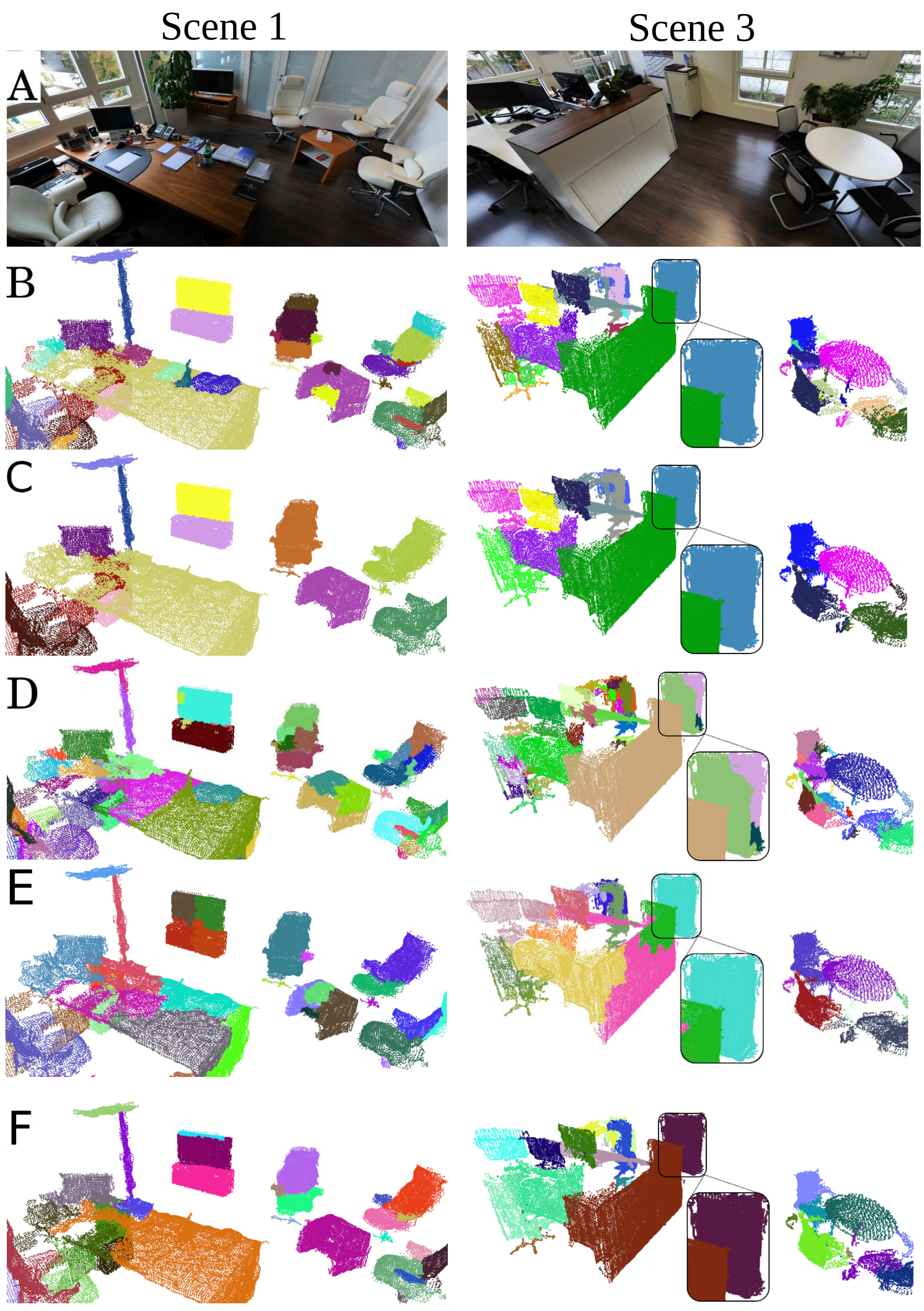


Figure 4.8: Segmentation results for the laser scanner dataset scenes 1 and 3 (left and right column). Here, row A shows RGB information that is given for illustration, but not used by any of the algorithms. B is the fine GT. C illustrates coarse GT. D represents LCCP segmentation results. E shows segmentation results of the approach of Van Kaick *et al.* [85]. F corresponds to segmentation results of the proposed method. Observe that the proposed method correctly segments the chairs and table areas in scene 1. In contrast, Vak Kaick and LCCP oversegment the table area and the chair backs. Reproduced with permission from [4], ©2017 INSTICC.

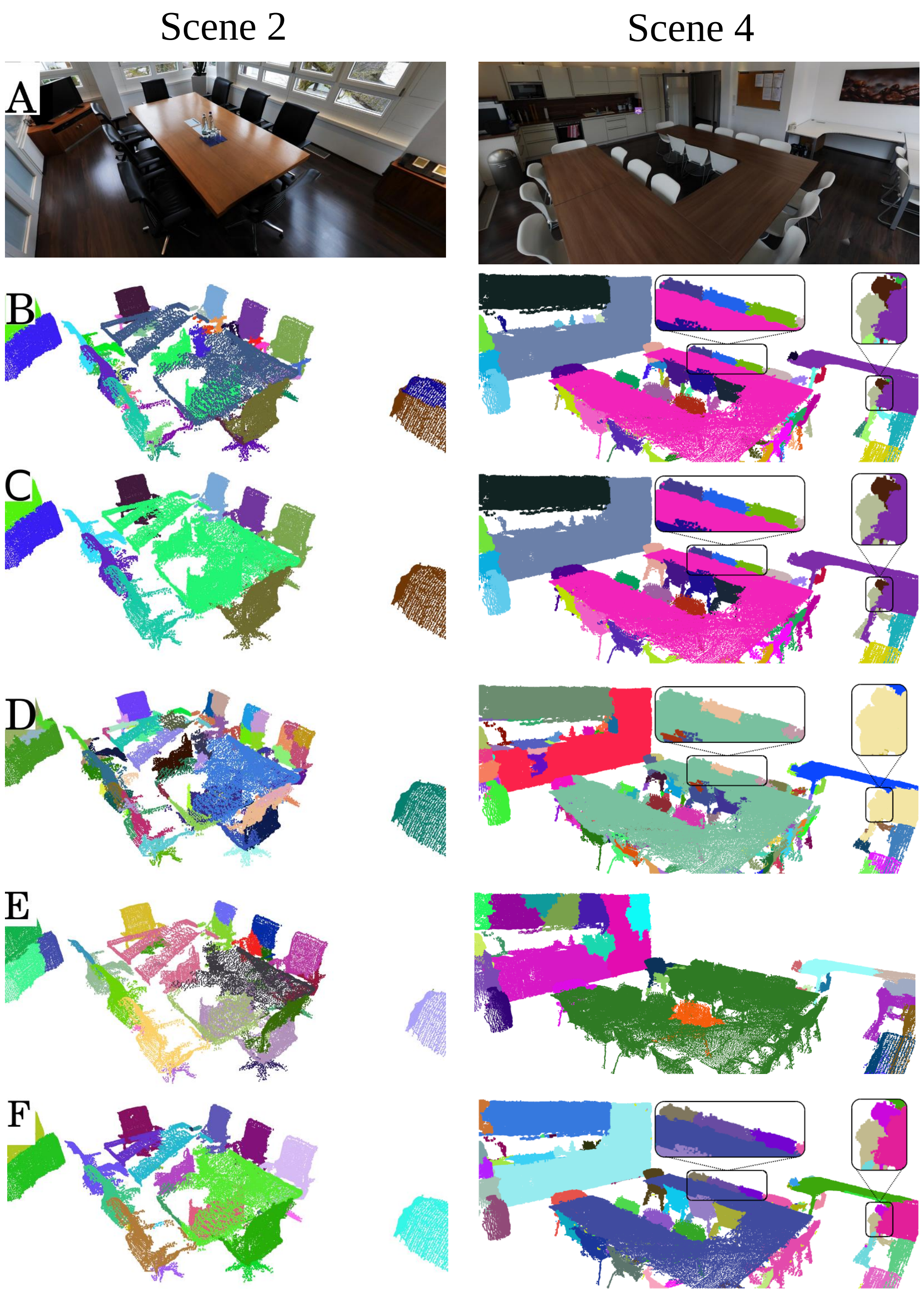


Figure 4.9: Segmentation results for the laser scanner dataset scenes 2 and 4 (left and right column). Here, row A shows RGB information that is given for illustration, but not used by any of the algorithms. B is the fine GT. C illustrates coarse GT. D represents LCCP segmentation results. E shows segmentation results of the approach of Van Kaick *et al.* [85]. F corresponds to segmentation results of the proposed method. Observe that the proposed method correctly segments the chairs placed around the table in scene 2. Due to the sparse PC data in the table region of scene 2, none of the algorithms can correctly segment the table into a single region. Moreover, the segmentation performance of the proposed method on the chairs in scene 4 is superior as compared to the other methods, where Vak Kaick and LCCP methods oversegment the table area and the chair backs. Reproduced with permission from [4], ©2017 INSTICC.

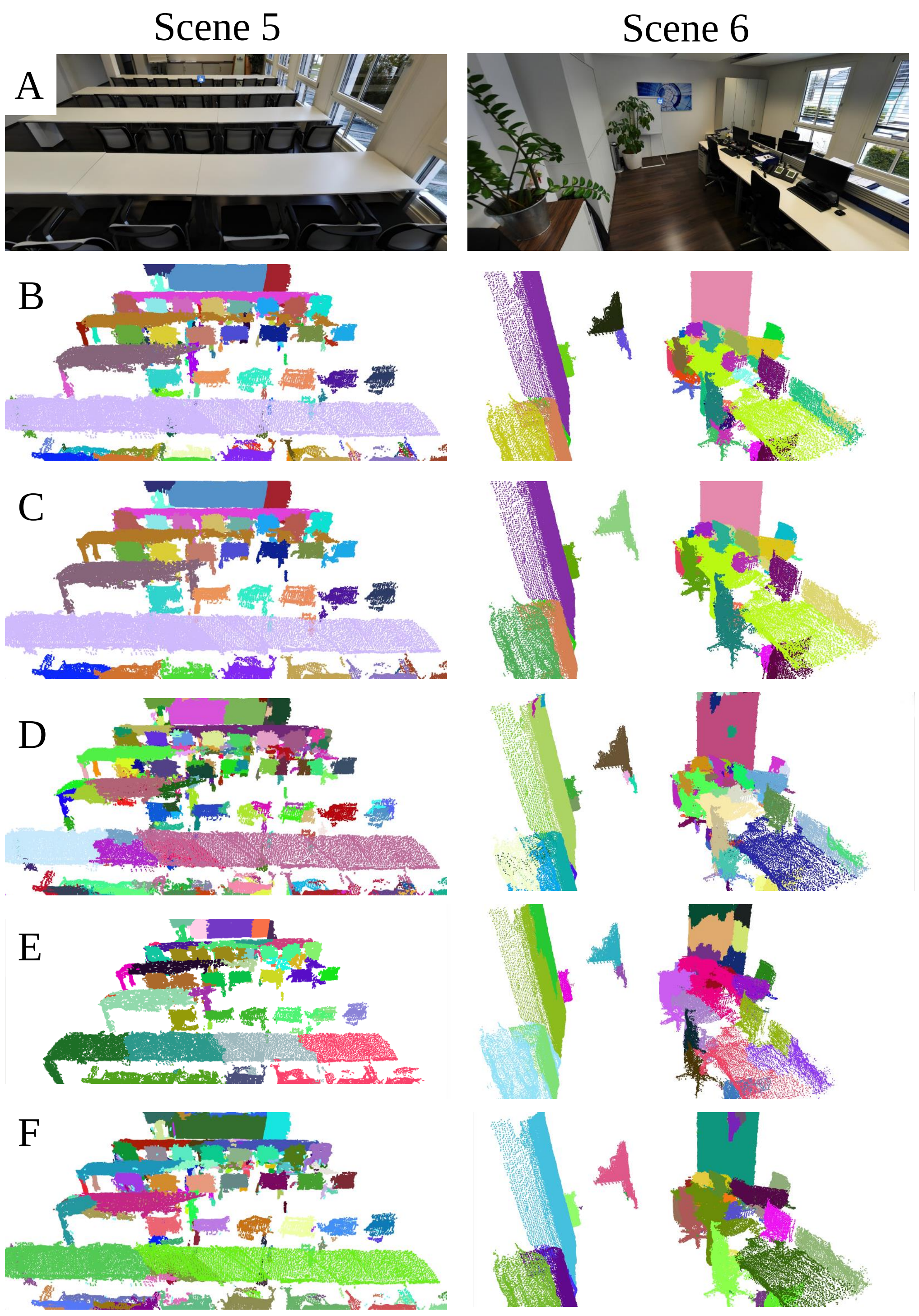


Figure 4.10: Segmentation results for the laser scanner dataset scenes 5 and 6 (left and right column). Here, row A shows RGB information that is given for illustration, but not used by any of the algorithms. B is the fine GT. C illustrates coarse GT. D represents LCCP segmentation results. E shows segmentation results of the approach of Van Kaick *et al.* [85]. F corresponds to segmentation results of the proposed method. It can be seen that all of the algorithms oversegment the table and chair regions in in the bottom right of scene 5 due to PC data sparsity. For scene 6, the proposed method can correctly segment the table and chair area, as compared to LCCP and Van Kaick *et al.* methods. Reproduced with permission from [4], ©2017 INSTICC.

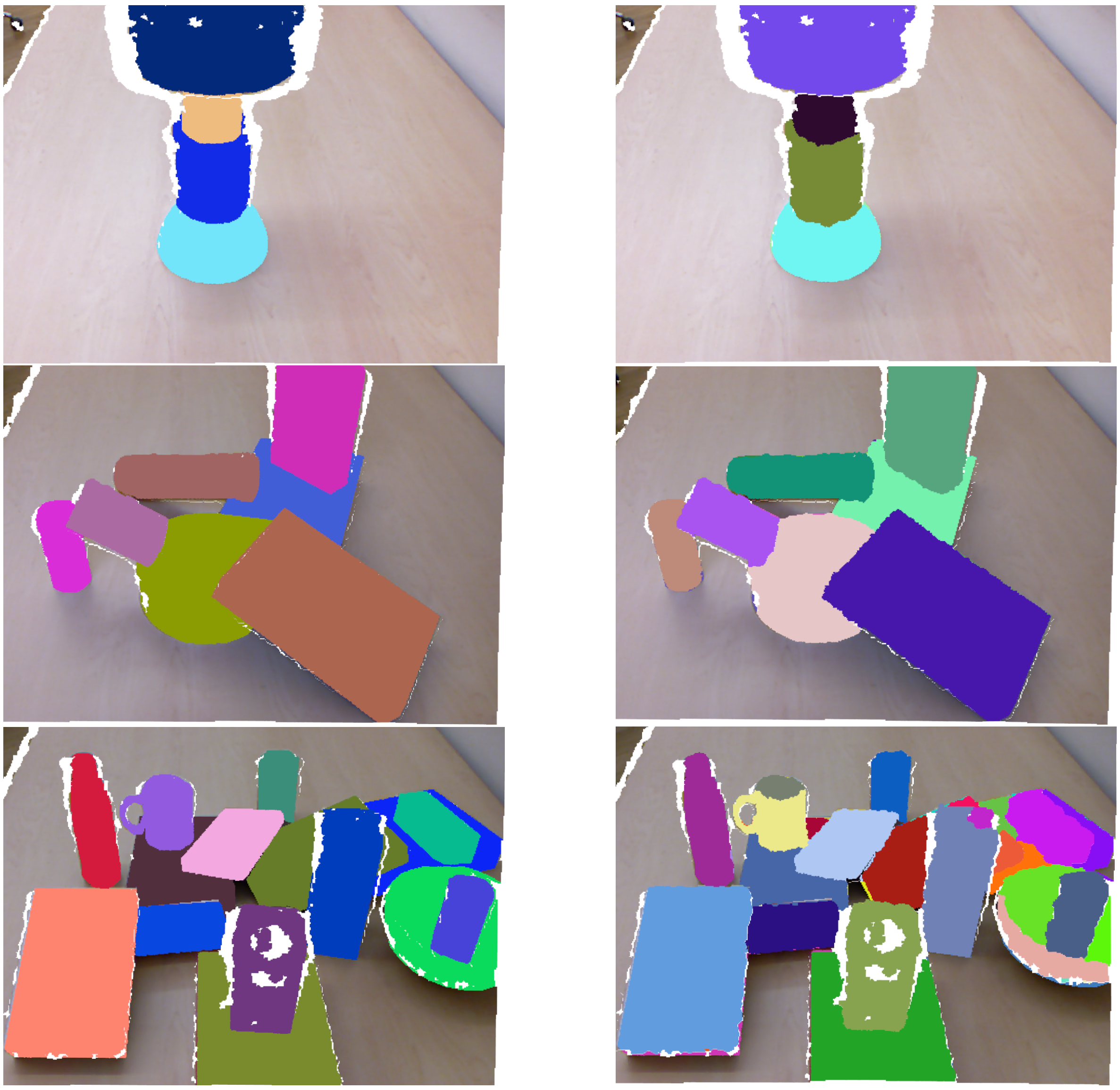


Figure 4.11: Exemplary segmentation results for the OSD dataset [53]. Top to bottom row: scene 42, scene 51 and scene 63. Within each row from left to right column: GT labeling and segmentation results of the proposed algorithm. Different colors of the objects correspond to different labels. The proposed method performs erroneous segmentation for the top-right part of the scene in the book area. Furthermore, the cup is oversegmented into two areas due to concavity. Finally, the book that is fully occluded in the middle part is segmented into two parts, despite that it represents one object. This is a particularly challenging segmentation task for any method using geometry only.

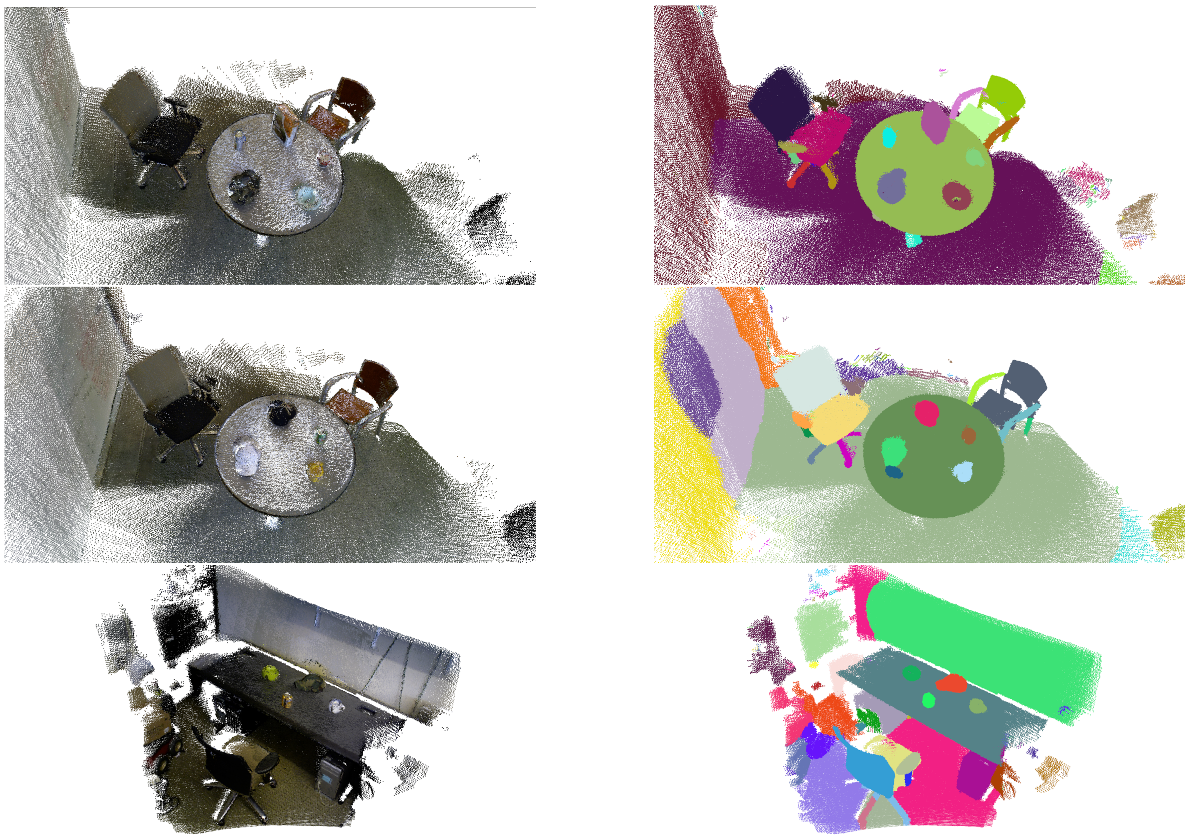


Figure 4.12: Exemplary object segmentation results for the dataset of Lai *et al.* [59]. Left column: PC data. Right column: segmentation result of the proposed method. From top to bottom rows: scene 6, 8 and 14. In the right column, different colors correspond to the different object segments. The proposed method achieves good segmentation performance by accurately segmenting objects on the table. The major segmentation mistakes correspond to the floor area of scene 14 and the wall area of scene 6. This is due to the curved surface properties, which serve as indication of the object boundary.

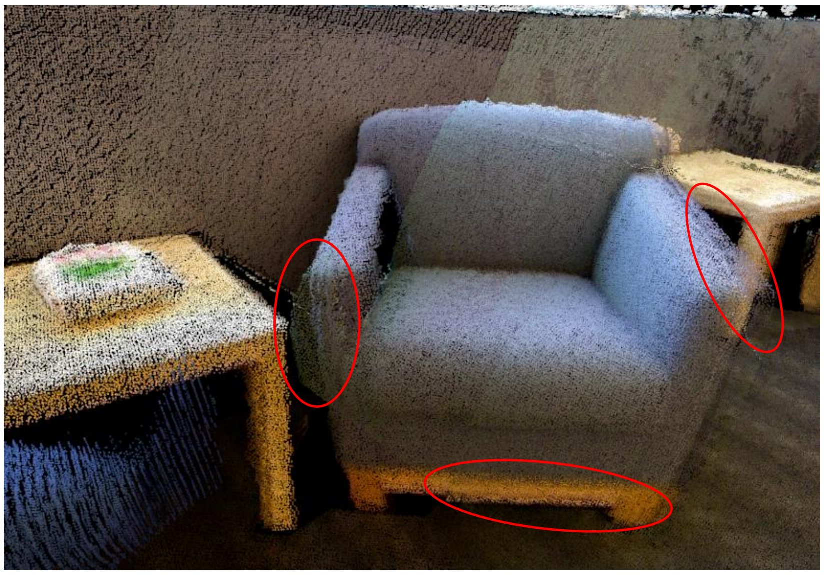


Figure 4.13: Illustration of the registration inaccuracies for the indoor SUN dataset of Song *et al*. [129]. Observe that there are several hypotheses for points describing the chair arm, which result in unclear object boundaries and high point noise (encircled areas).

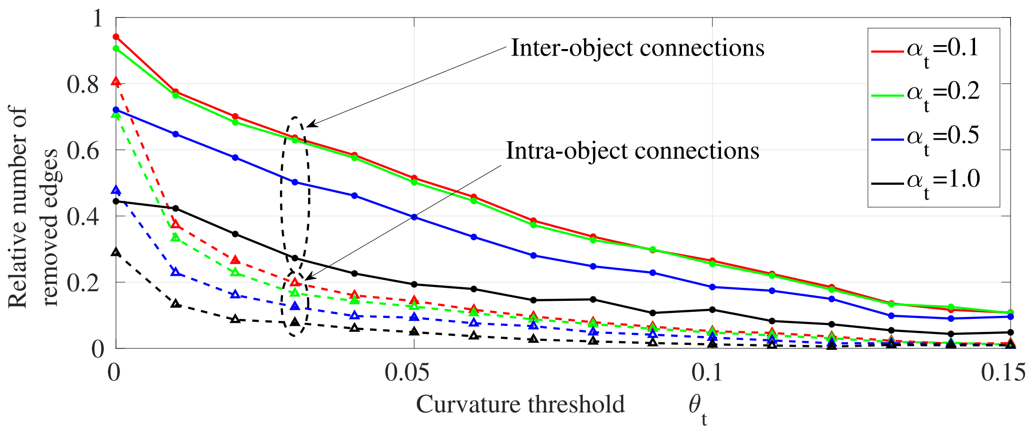


Figure 4.14: Relative number of removed supervoxel graph edges for intra- and inter-object connections vs. curvature threshold θ_t. Inter-object connections are shown in solid lines, whereas intra-object ones are shown in dashed lines. By choosing the threshold value in the range 0.02 to 0.03 a significant number of inter-object connections are removed (68.31%), whereas most of the intra-object connections are preserved (77.21%). This allows to significantly simplify the segmentation task while achieving even better performance. Note the low number of removed intra-object connections. Reproduced with permission from [4], ©2017 INSTICC.

Chapter 5

3D Object Classification in Point Clouds using Point Pair Features

In this chapter, the problem of object classification in PCs is presented. Furthermore, the proposed method to object classification using point pair features in combination with deep learning is described. After that, the experimental evaluation of a number of PC indoor datasets is provided and discussed.

Parts of this chapter have been published in [1].

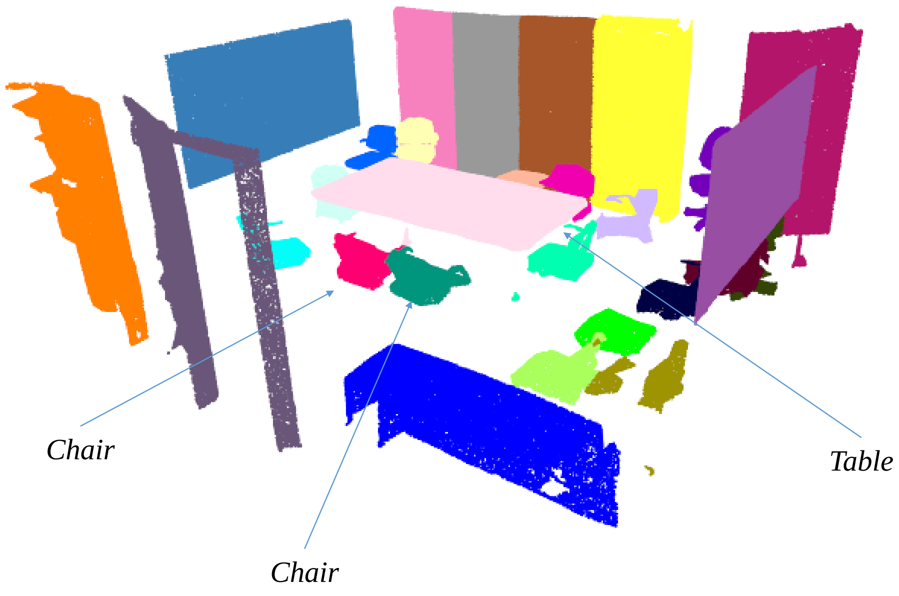


Figure 5.1: Once the PC data of the rooms has been segmented into objects, each object has to be classified according to its semantic meaning. Different colors correspond to distinct segments that need to be assigned to one of the semantic categories. Dataset of [11].

5.1 Problem Statement

After the object segmentation step is finished, each object has to be classified according to its semantic meaning, such as a *chair* or *table*, see Fig. 5.1 for illustration of the object classification step. Of all 3D data representations that can be used to address this problem, PCs are closest to the output from LiDAR and depth sensors. This representation is challenging due to its irregular data structure and large data size. Because of this, many methods first convert this representation into 3D voxel grids or multi-view rendered images [133]. While convenient for data handling, this preprocessing step introduces additional computational complexity and makes the resulting data representation unnecessarily voluminous. For this reason, this work focuses on PC-based approaches. As discussed in Section 2.2.3, the main requirements for an object classification algorithm are robustness to noise and occlusion, low computational complexity, rotation-invariance, and high classification accuracy. Out of the existing methods, none completely satisfies these requirements, mostly due to the challenging requirement of robustness to noise and occlusion. In particular, once the level of noise increases, it becomes challenging to distinguish small variations in PC data due to fine details in object geometry from variations due to noise. To address these limitations, a novel method for object classification using PC data is presented.

Contributions of the proposed approach are the following:

- A new handcrafted point pair function-based 4D descriptor exhibiting high robustness for realistic, noisy PC data is proposed. Its superior performance has been confirmed with a number of experimental evaluations.
- Combination of the presented descriptor with a 4D CNN architecture using the proposed 4D descriptor as input. It outperforms existing deep learning approaches on real PC datasets obtained using LiDAR or RGBD sensors.

5.2 Method

An overview of the proposed approach is given in Fig. 5.2. Based on the input PC data of the object, the point pair-based descriptor in the form of a 4D histogram is computed. This histogram is then given to the CNN that outputs the object class (e.g., *chair* or *table*).

5.2.1 Point Pair Descriptors

To achieve rotation-invariance, point pair (PP)-based descriptors are typically preferred to the descriptors that use points directly for geometry description as the PP-based descriptors show the best performance [18], [103], [106], [112]. Usually, point pairs (or point n-tuples in the general case) are randomly sampled from the point set. Based on the sampled pairs, the PP functions map them to scalar values, which are then quantized into a histogram that describes the statistics of the shape. Such point sampling step leads to certain randomness in the result, but also enhances robustness to noise and occlusion as instead of geometry itself,

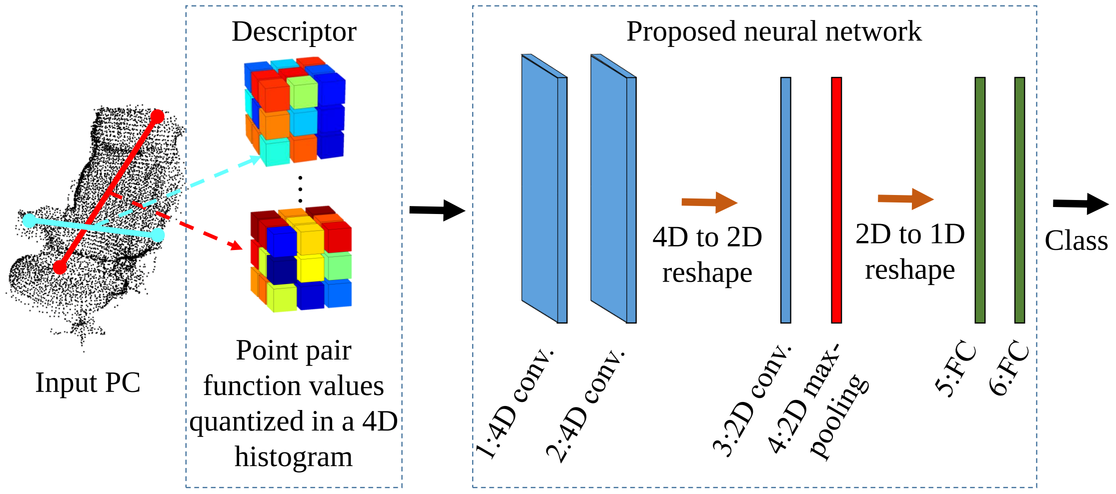


Figure 5.2: Overview of the proposed object classification pipeline that is a combination of a novel handcrafted descriptor and a 4D convolutional neural network (CNN). For details on the network architecture and layer dimensions, see Fig. 5.7. Here, FC denotes a fully connected layer. Reproduced from [1], ©2018 IEEE.

a statistical description of the object shape is used. A further advantage of this approach is its rotation-invariance.

Point pair function (PPF) f is defined as the mapping of a PP (containing a pair of point coordinates and normal vectors) to a scalar value as follows:

$$f : (\mathbb{R}^3, \mathbb{R}^2) \times (\mathbb{R}^3, \mathbb{R}^2) \rightarrow \mathbb{R}^1, \tag{5.1}$$

with $\mathbb{R}^3$ being a Euclidean space and $\mathbb{R}^2$ denoting a manifold of surface normal orientations in 3D space. Due to the fact that the normal vectors have unit norm, they lie in $\mathbb{R}^2$ instead of $\mathbb{R}^3$. The following **functions** f_1 to f_4 are employed in the proposed PP-based descriptor:

1. Euclidean distance between the points in the PP f_1 [103].
2. The maximum angle between the corresponding surface patches of the points and direction vector $\mathbf{d}$ connecting the points f_2.
3. Distance between the normal vectors of the corresponding points in the PP f_3.
4. Occupancy ratio along the line connecting the points f_4 [106].

Euclidean distance. The function value f_1 is the Euclidean distance between two points $\mathbf{p_1}$ and $\mathbf{p_2}$:

$$f_1(\mathbf{p_1}, \mathbf{p_2}) = \|\mathbf{d}\|_2 = \|\mathbf{p_2} - \mathbf{p_1}\|_2. \tag{5.2}$$

The statistics of the distances between point pairs represents both the geometry and the size of the object (see $\mathbf{d}$ in Fig. 5.3). Thus, objects having large dimensions, such as elongated tables, typically have a larger number of high distance samples, whereas smaller objects, such as short tables, have a higher occurrence of smaller distance values.

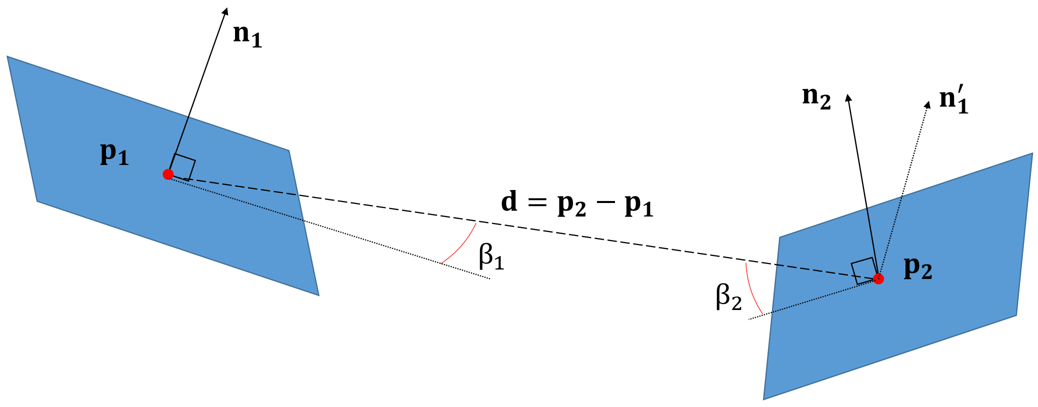


Figure 5.3: Illustration of the points $\mathbf{p_1}$, $\mathbf{p_2}$, their normal vectors $\mathbf{n_1}$, $\mathbf{n_2}$ and the Euclidean distance $f_1 = \|\mathbf{d}\|_2$. $\mathbf{n_1}'$ denotes $\mathbf{n_1}$ that originates at $\mathbf{p_2}$. The resulting angles between vectors $\mathbf{d}$ and the tangent patches of points $\mathbf{p_1}$ and $\mathbf{p_2}$ are shown as β_1 and β_2, respectively. Tangential planes of surfaces at points $\mathbf{p_1}$ and $\mathbf{p_2}$ are shown in blue. Adapted from [1], ©2018 IEEE.

The maximum surface angle. Function value f_2 describes the surface patch orientation with respect to the line $\mathbf{d}$ connecting both points $\mathbf{p_1}$ and $\mathbf{p_2}$. It is defined as follows:

$$f_2(\mathbf{p_1}, \mathbf{n_1}, \mathbf{p_2}, \mathbf{n_2}) = \max(\beta_1, \beta_2), \tag{5.3}$$

with β_1 and β_2 being the angles between vector $\mathbf{d}$ and the tangent patches of points $\mathbf{p_1}$ and $\mathbf{p_2}$, respectively (see Fig. 5.3). They are defined as follows:

$$\beta_1 = \arccos(\mathbf{n_1} \cdot \mathbf{d}) - \pi/2, \tag{5.4}$$

$$\beta_2 = \arccos(\mathbf{n_2} \cdot \mathbf{d}) - \pi/2. \tag{5.5}$$

$$f_2 = \max(\beta_1, \beta_2), \tag{5.6}$$

The direction vector is computed as follows: $\mathbf{d} = \mathbf{p}_2 - \mathbf{p}_1$. f_2 lies in the range between $-\pi/2$ and $\pi/2$. Function f_2 is important in the cases, when the Euclidean distance f_1 and the normal-based distance f_3 are not very descriptive, see Fig. 5.4 for illustration. Here, the point pairs in the left and the right have the same Euclidean distance and the same normal distance. In contrast, function f_2 has significantly different values for both pairs, hence this function allows us to distinguish these cases.

Although β_1 and β_2 may take on different values and, hence, provide an informative surface description, only one value is chosen. Thus, a trade-off between compactness and accuracy is achieved. To compute a descriptive value from both values, a number of mathematical functions can be used. The ablation study that compares different functions, such as maximum, minimum or mean, is given in Fig. 5.12 and discussed in Section 5.3.2. It has been observed that the maximum operation is more descriptive for noisy datasets as compared to other functions.

Normal distance. Function value f_3 describes the similarity of surface orientations of the corresponding point neighborhoods. It is defined as follows:

$$f_3(\mathbf{n_1}, \mathbf{n_2}) = \arccos(|\mathbf{n_1} \cdot \mathbf{n_2}|), \tag{5.7}$$

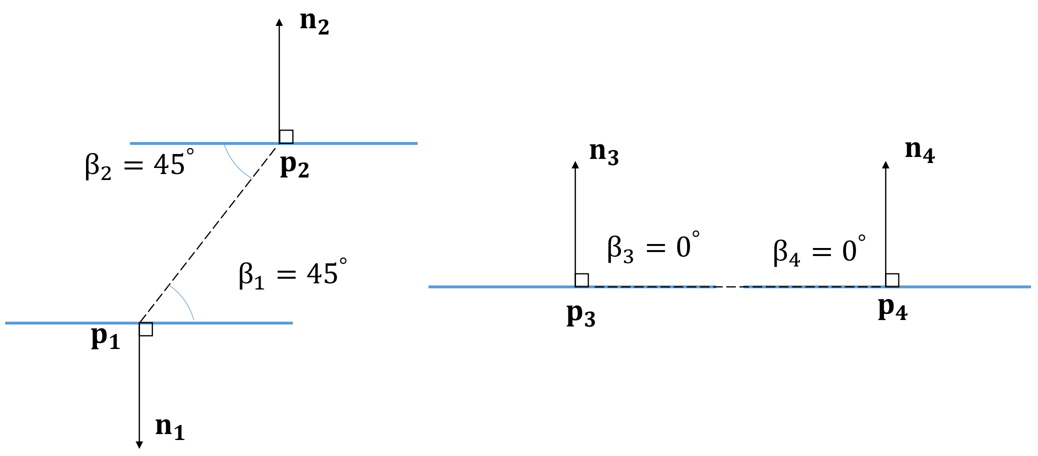


Figure 5.4: Illustration of the case with point pairs $(\mathbf{p_1}, \mathbf{p_2})$ (left) and $(\mathbf{p_3}, \mathbf{p_4})$ (right) that have similar Euclidean (f_1) and normal distances (f_3), but still describe significantly different shapes. The maximum angle between patches and direction f_2 is an important feature for such case. Adapted from [1], ©2018 IEEE.

which lies in the range from 0 to π. The absolute value in the expression of the dot product is used to eliminate the influence of the viewpoint. In particular, the viewpoint information can often be unreliable in MVPCs due to scanning the same surface from various directions [4]. In essence, this feature describes the relative orientation of two local surfaces.

Occupancy ratio. The feature f_4 is used to describe the global object geometry. In particular, it encodes the information about joint visibility of the points in the PP. For fast visibility computations, the object volume is voxelized using a 3D voxel grid of dimensions $N_X \times N_Y \times N_Z$. This way, fast lookup and occupancy check computations can be achieved [106]. For the given datasets, the same number of voxels in three dimensions are used as the indoor objects often have cube-like shape. The values are set to $N_X = N_Y = N_Z = 64$. In particular, the value of f_4 is defined as follows:

$$f_4(\mathbb{P}, \mathbf{p}_1, \mathbf{p}_2) = \frac{N_{occ}}{N_{total}}, \tag{5.8}$$

with $\mathbb{P} \in \mathbb{R}^3$ being the set of points in the considered PC, N_{occ} is the number of occupied voxels intersected by the 3D line $\mathbf{d}$ connecting two points $\mathbf{p_1}$ and $\mathbf{p_2}$, and N_{total} is the total number of voxels intersected by the line (see Fig. 5.5). The voxel is classified as occupied if at least one point is contained inside. Even though a higher threshold for voxel occupancy decision can be chosen, it would disregard areas with a low number of points, e.g., low point density areas. To consider all such areas, a conservative value of 1 is chosen as a threshold for voxel occupancy. This function f_4 describes the global object geometry, because the voxel grid occupancy is computed based on all points. For the example with $\mathbf{p}_1$ and $\mathbf{p}_2$ in Fig. 5.5, the values are $N_{occ} = 8$ and $N_{total} = 13$, which leads to $f_4 = 0.615$. In contrast, for point pair $(\mathbf{p}_1, \mathbf{p}_3)$ $f_4 = 1$, as all intersected voxels are occupied. Feature f_4 lies in the range from 0 to 1.

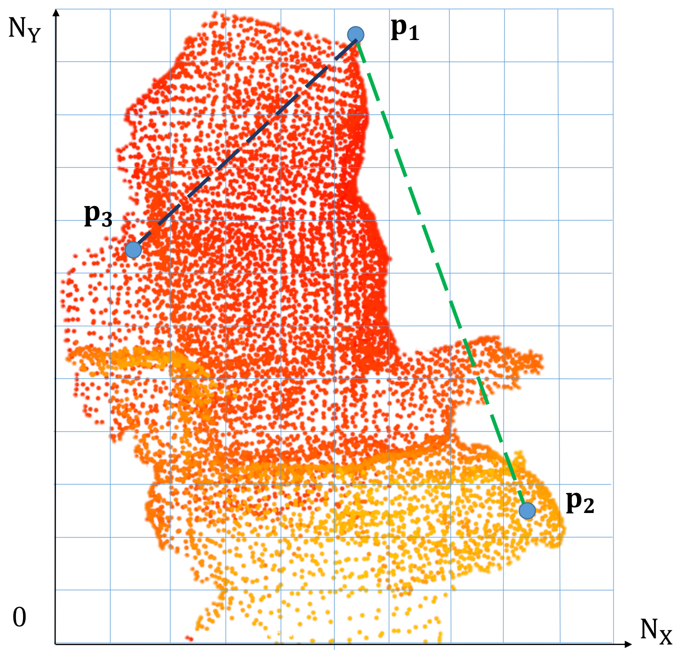


Figure 5.5: 2D illustration of the grid used for performing voxel occupancy checks for the voxels lying along the line (dashed line) connecting a given point pair $\mathbf{p}_1$ and $\mathbf{p}_2$ and another point pair $\mathbf{p}_1$ and $\mathbf{p}_3$. Adapted from [1], ©2018 IEEE.

5.2.2 Feature Statistics

In order to accurately describe geometric statistics of complex shapes, a number of PPs need to be drawn at random from the point set. In this work, $20{,}000$ randomly sampled PPs are used, as this number has been observed to be sufficient to describe complex object geometry. After the PPF values are computed for these pairs, they need to be aggregated into a descriptor histogram. Many approaches (among others [106] and [100]) assume that the different function values are uncorrelated with each other. Therefore, these function values were previously discretized into bins and concatenated into a 1D histogram. It has been observed in this work that the feature quantization in a 1D histogram leads to a significant loss of performance because information on co-occurrences of different function values is neglected. To verify this, the Pearson correlation coefficient is computed between each pair of features. The Pearson correlation between vectors $\mathbf{x}$ and $\mathbf{y}$ is defined as follows [134]:

$$r(\mathbf{x},\mathbf{y}) = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i-\bar{y})^2}}, \tag{5.9}$$

where x_i and y_i are the ith values of $\mathbf{x}$ and $\mathbf{y}$, respectively. n is the size of the vectors. $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ is the sample mean of the values in the vector $\mathbf{x}$. The correlation coefficient r has a value in the range between -1 and 1. The average computed Pearson correlation value over all PPF values and all datasets is equal to 0.424. This value confirms a significant level of correlation. Thus, to avoid loss of information on 4D co-occurrences, similar to [103], a 4D histogram of function value occurrences is computed instead (shown in Fig. 5.6). The

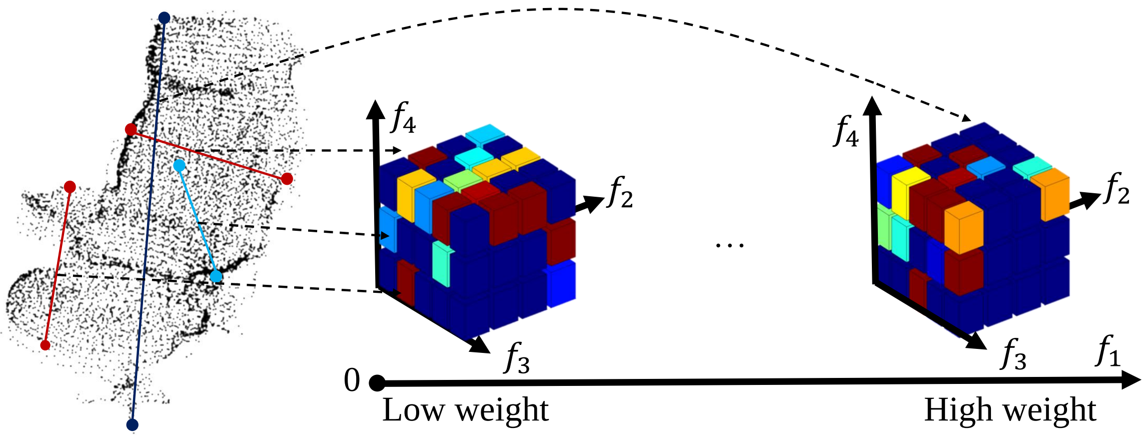


Figure 5.6: 4D histogram (right) that is used to discretize the aggregated counts of sampled PPF values into a descriptor. Blue color denotes the bins with low number of counts, whereas red corresponds to high. The corresponding object PC data is shown in the left. It can be seen that the different function values are aggregated in different parts of the resulting histogram. Reproduced from [1], ©2018 IEEE.

histogram can be expressed as:

$$\boldsymbol{F} = (\mathbf{w_1}, \mathbf{w_2}, \mathbf{w_3}, \mathbf{w_4}), \tag{5.10}$$

where $\mathbf{w_i}$ are bin counts of ith feature.

The proposed descriptor is denoted as Enhanced Point Pair Function (EPPF). A straightforward extension into a 4D histogram with a large and equal number of bins along each dimension would result in an exponential increase of computational complexity [103], [104]. Instead, it is possible to leverage the observation that not all PPF values are equally informative for the description of the object geometry. Hence, a different number of bins can be chosen for different dimensions. In the later Section 5.3.2 an experimental study of the feature contribution to the overall performance (ablation study) is provided. In particular, function f_1 helps to distinguish objects of different sizes, therefore a relatively large number of bins is chosen, $N_{f_1} = 20$.

As the values of f_1 can significantly vary for the objects with different physical dimensions, scaling of function f_1 is required. An ablation study investigating the performance using different scaling strategies is given in Fig. 5.12 and discussed in Section 5.3.2. In particular, the objects can be scaled to fit into unit cube, thus the information about absolute physical dimensions of each object is lost. Alternatively, the objects can be scaled according to the maximum or median of sizes of all objects. Contrary to scale-invariant approaches [114], the object is not scaled to fit into a unit sphere, as it has been observed that in indoor environments the size of the object provides important information about its semantic label. For example, *monitor* and *whiteboard* can have similar geometric shape, but different physical dimensions. To preserve the information on dimensions, the descriptor is designed to be scale-variant. Thus, all objects are scaled so that the largest one fits into a unit cube. In case a certain object has dimensions that are much larger than the dimensions of the rest of the object, this could lead to inefficient quantization of the feature f_1. In other words, a number of bins would not be used for the description of the remaining smaller objects. In practice, however, it has been observed that this does not lead to significant negative effects as long as the object have physical dimensions in the same order of magnitude, see Fig. 5.12 and Section 5.3.2 for more details.

For f_2, it has been observed that the its descriptive ability is relatively low for noisy and occluded data. Therefore, a relatively small number of bins was chosen $N_{f_2} = 4$. Furthermore, for f_3, the number of bins was set to $N_{f_3} = 5$. Finally, for f_4, it has been observed that $N_{f_4} = 3$ is sufficient. A number of experiments with larger numbers of bins have illustrated that there was no significant performance improvement. Thus, a trade-off between complexity and accuracy of the descriptor has been achieved.

Experiments have also shown that point pairs with larger Euclidean distances usually have higher discriminative power as compared to those with smaller Euclidean distances. This effect is due to the fact that every object has point pairs with small distance values, but only big objects have point pairs with larger distances. To improve the discriminative ability of objects of different sizes and to suppress the influence of noise in low distance regions, a bin weighting factor is used. For the bin located at index i, j, k, l, the factor is computed as follows:

$$\alpha_i = \ln(i/N_{f_1} + c), \tag{5.11}$$

where i is the bin index of the Euclidean feature f_1. α_i is used to compute the bin count as follows: $w^{new}_{i,j,k,l} = \alpha_i \cdot w_{i,j,k,l}$. c is a constant value greater than 1, which is used to guarantee that weights are positive and to mitigate noise for point pairs with smaller Euclidean distances. Based on the experimental validation, the value is set to $c = 1.2$. Finally, every weighted descriptor histogram is normalized. The total number of bins of the resulting 4D histogram is set to $N = 20 \cdot 4 \cdot 5 \cdot 3 = 1{,}200$. For matching, different distance metrics can be used, such as L_1 or L_2 norms. It has been observed that they do not allow to accurately describe the distance between the underlying feature distributions. To address this limitation, a probabilistic distance metric is used, i.e., symmetrized version of Kullback-Leibler divergence [135]. It is defined as follows:

$$d(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{N} |a_i - b_i| \ln \frac{a_i}{b_i}, \tag{5.12}$$

where $\mathbf{a}$ and $\mathbf{b}$ are the histogram counts for object 1 and 2, respectively. Similarly to [103], all zero bins of a histogram are set to a common minimum value that is twice smaller than the smallest observed bin value in this dataset. By setting the zero bins to a non-zero minimum value, it is possible to avoid numerical issues when having certain bin counts equal to zero and performing division by zero. The zero bin counts can occur due to an insufficient number of drawn random samples or due to specific properties of the geometry of this object.

5.2.3 4D Deep Learning Architecture

The previously computed 4D descriptor is rotation-invariant, which means that the descriptor values do not change in case the object is rotated. The rotation-invariance also resolves the issue of symmetry of point sets in neural networks. This issue is because the PC is a set of

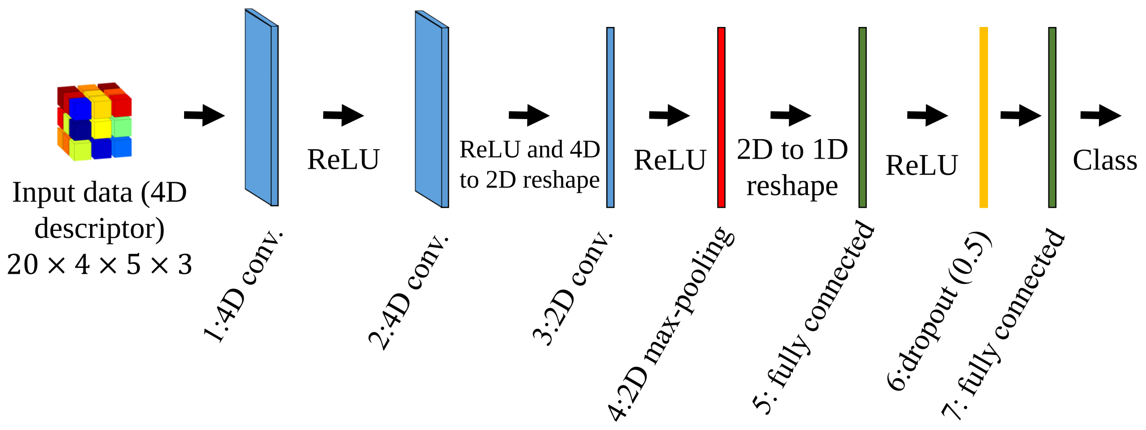


Figure 5.7: Architecture of the proposed 4D neural network. Input data is a 4D descriptor computed from the PC data. The descriptor histogram is provided to the neural network for object classification. "Conv." denotes convolutional layer. ReLU stands for rectified linear unit (ReLU). Table 5.1 provides more details on the dimensions. Adapted from [1], ©2018 IEEE.

points that describes underlying geometry. By performing permutations of the points in the PC, the underlying geometry does not change, but the data values look different for many algorithms, such as CNNs. By using the EPPF descriptor, it is, thus, possible to feed this representation into a neural network for the task of object classification. For more details on CNNs, see Section 2.1.7.

To preserve information about 4D co-occurrences of the function values, 4D convolution is used in the input layers of the neural network (see Fig. 5.7). 4D convolution has already been successfully applied for the task of material recognition by Wang *et al.* [136], where it outperformed other architectures. In this work, 4D convolution is applied as a stacked 3D convolution, following the implementation available at GitHub[1]. Within the neural network, 4D convolutional blocks are used in the first and second layers, respectively. Details on the dimensions are given in Table 5.1. Afterward, the resulting network responses are reshaped into a 2D structure and input into a 2D convolutional block. Furthermore, 2D max-pooling is performed to pool features from a spatial neighborhood and thus achieve spatial robustness. This step is followed by reshaping from 2D to 1D representation, which is input into a fully connected layer. Afterward, a dropout is applied to the fully connected layer with probability 0.5 during the training procedure. The dropout is used to achieve regularization and enhance the generalization property of the network. After this layer, another fully connected layer is used. At the output of the network, a class prediction for the object is provided. For training, a cross-entropy loss function was used. For more details on the loss function, see Section 2.1.7.

For comparison, 2D and 3D convolution-based networks for object classification are also designed, trained and evaluated. This way, it is possible to verify whether 4D convolution is essential for good classification performance. For a fair comparison, the dimensions of

[1] Stacked 4D convolution https://github.com/mhuen/TFScripts/blob/master/tfscripts/conv.py. Accessed: 2018-12-22.s

Table 5.1: Layer dimensions for 2D, 3D and 4D-variants of the network. N_f denotes the number of filters. "conv." denotes convolutional layer. Reproduced from [1], ©2018 IEEE.

Layer	2D network	3D network	4D network	N_f
Input	40×30	$20 \times 4 \times 15$	$20 \times 4 \times 5 \times 3$	-
1	2D conv.: $5 \times 5 \times 1$	3D conv.: $5 \times 5 \times 1 \times 1$	4D conv.:$5 \times 2 \times 2 \times 1 \times 1$	32
2	2D conv.: $5 \times 5 \times 32$	3D conv.: $5 \times 5 \times 1 \times 32$	4D conv.: $5 \times 2 \times 2 \times 1 \times 32$	64
3	2D conv.: $5 \times 5 \times 64$	3D conv.: $5 \times 5 \times 1 \times 64$	2D conv.: $5 \times 5 \times 64$	48
4	2D max-pooling: 2×2			1
5	fully connected: 192×1024			1
6	dropout with probability 0.5			1
7	fully connected: $1024 \times N_{classes}$			1

equivalent 2D and 3D convolution-based network variants are chosen in such a way so that the number of parameters of all three networks is comparable to each other. The dimensions of the single layers are given in Table 5.1. Thus, for the 2D-variant of the network, the input 4D descriptor was first reshaped into 2D data with dimensions (40×30) and then processed with three 2D convolutional layers. For the 3D-variant of the network, the input 4D descriptor is reshaped into 3D data with dimensions $(20 \times 4 \times 15)$ and then processed with three 3D convolutional layers. The number of filters is chosen to be the same for all three networks. Filter stride value is set to 1.

5.3 Experimental Evaluation

For experimental comparison with state-of-the-art approaches, the best performing descriptors according to various benchmarks [97] were chosen. In particular, OUR-CVFH [100], ESF [106] and Wahl *et al.* [103] descriptors were considered. For OUR-CVFH and ESF, the Point Cloud Library 1.8 implementations were used [97]. For Wahl *et al.*, due to unavailability of the open source code, own version has been implemented in C++ following the description in the original paper. Furthermore, tuning of the descriptor parameters was performed to obtain optimal performance. For comparison with deep learning approaches, PointNet [114] was used as it is one of the most accurate approaches up-to-date that can directly work on 3D point sets without additional operations of multi-view projection [133] or voxelization [137]. For PointNet, the author's implementation was used[2].

The proposed descriptor EPPF has a larger number of bins as compared to OUR-CVFH and ESF. This fact raises the question, whether a larger number of bins has an impact on description performance. To answer this question, the version of the EPPF descriptor with a fewer number of bins was also evaluated. The bin number was chosen in such a way that it

[2] https://github.com/charlesq34/pointnet

is comparable to the other descriptors. In particular, $N_{f_1} = 15$, $N_{f_2} = 3$, $N_{f_3} = 4$, $N_{f_4} = 3$. This choice results in a total number of bins of $N = 540$. This value is comparable to 640 in ESF and 308 in OUR-CVFH. This descriptor is denoted as "EPPF short" from now on. For evaluation metrics, total accuracy was chosen, which is the accuracy value divided by the total number of objects. Also used were mean accuracy and mean recall. Mean accuracy and mean recall are accuracy and recall values, respectively, that are averaged over all classes. Furthermore, F1-score was also used as a single measure of classification performance. For more details, see Section 2.1.8.2.

5.3.1 Datasets

For evaluation, the Stanford PC dataset [11], ScanNet mesh dataset [138] and ModelNet40 CAD dataset [137] were used. These are the most recent and largest datasets of indoor objects that also contain semantic and instance annotation.

Stanford dataset. The Stanford dataset in [11] contains RGB and depth images and has been captured in six office areas within three different buildings, using structured-light sensors during a $360°$ rotation at each scanning location. Due to sensor noise and limited scanning time, point density significantly varies throughout the scene. Furthermore, there is a high level of occlusion. The annotation has been done using humans that labeled triangles in the mesh accordingly to semantic categories. This annotation was then projected onto the PC data. The authors in [11] proposed a training/testing split of data according to buildings. This split cannot be used for this evaluation, because in this case some objects never occur in the testing or training sets. This fact would make the evaluation of object classification less meaningful, therefore a different split of $60/40$ was derived. Here, 60% of object instances per category constitute the training set and the remaining 40% represent the testing set. The category *clutter* was omitted, as it contains multiple categories. Furthermore, architectural element categories with a high level of planarity were also removed, such as *floor*, *ceiling* and *wall*, as they can easily be classified using normal direction of the PC data. The presence of these objects would make the object classification task unnecessarily complex. Thus, there are 10 classes in total with $3{,}735$ objects. The categories are as follows: *beam, board, bookcase, chair, column, door, sofa, stairs, table, window*. For illustration of intra-object variance, the PC data for two instances of category *table* is shown in Fig. 5.8. It is possible to observe that there are significant variations in geometry between the two instances of the same category.

ScanNet dataset. The ScanNet dataset [138] is a large-scale mesh dataset containing a semantic annotation of indoor scenes. The mesh data exhibits a high level of occlusion and noise, as it is collected using a commodity low-cost RGBD sensor. Annotation was done using humans that labeled planar groups of mesh triangles. This step was followed by annotation verification to ensure consistency of labeling. For classification, the training/testing split specified by the authors [138] was used. Out of all categories, only the categories that are compatible with ShapeNet-55 dataset from [138] were used. To avoid an unbalanced training set, category *laptop* was removed, as it contains only 18 instances (as compared to the other categories with at least 50 instances). Thus, the used dataset contains 14 categories:

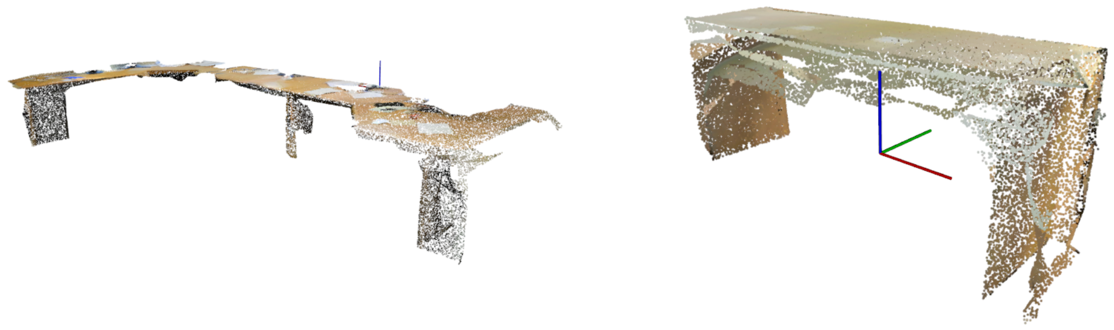


Figure 5.8: Illustration of the PC data for two instances (left and right) of object *table* from the Stanford dataset [11]. Significant variations in geometry between the two instances can be observed.

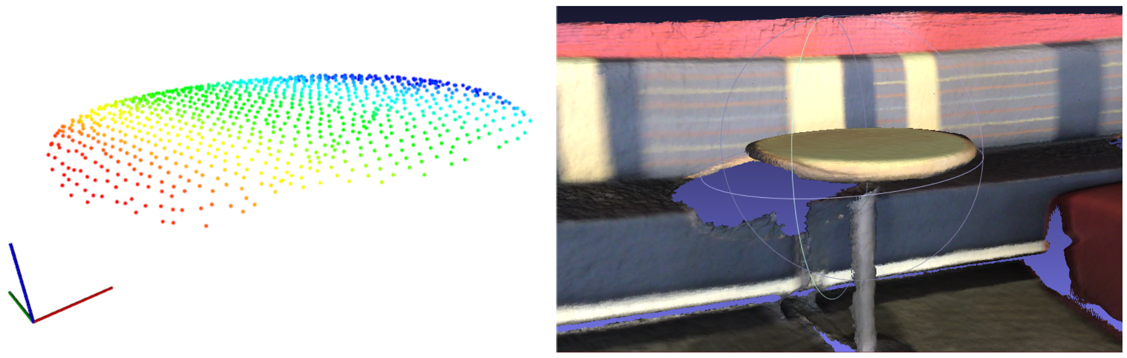


Figure 5.9: Illustration of the PC data (left) for object *table* from the ScanNet dataset [138]. Colored mesh is shown on the right. The table leg was erroneously labeled as not belonging to the table object, as can be seen from the PC data.

basket, *bathtub*, *bed*, *cabinet*, *chair*, *keyboard*, *lamp*, *microwave*, *pillow*, *printer*, *shelf*, *stove*, *table* and *tv*. This choice of categories results in 5,203 objects in the training set and 1,699 objects in the testing set. An example of a table's PC and mesh data is shown in Fig. 5.9. It can be observed that the provided annotation has certain inaccuracies, in particular, the table leg was erroneously labeled as not belonging to the table.

ModelNet40 dataset. The ModelNet40 (also often denoted as M40) dataset [137] is a large-scale CAD model dataset of objects. The CAD models have been manually cleaned, thus containing practically no noise or occlusion. There are 12,311 CAD models from 40 categories, separated into 9,843 instances for training and 2,468 instances for testing sets. The categories correspond to common objects, such as airplane, table, chair and other.

ModelNet40 and ScanNet datasets contain mesh models, which need to be converted into a PC representation. For this, the mesh sampling approach from the Point Cloud Library [97] was employed with a resolution of 1 cm. Because EPPF, Wahl *et al.*, and OUR-CVFH descriptors require normal information, the mesh sampling step was followed by normal estimation using the method of Boulch and Marlet [31]. For illustration, PC data for two instances of category *table* for the ModelNet40 dataset is shown in Fig. 5.10. It can be observed that

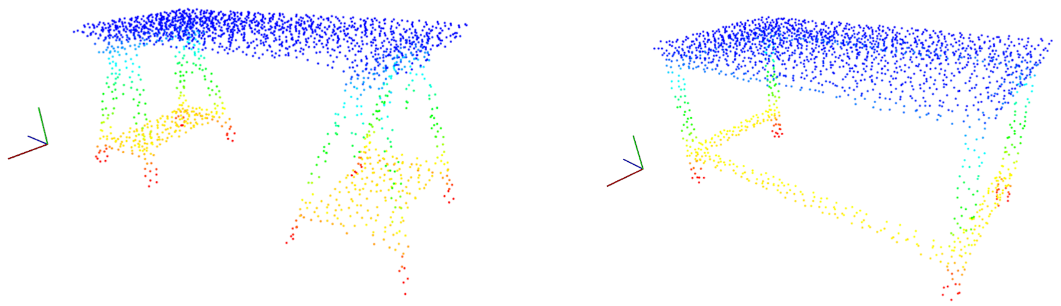


Figure 5.10: Illustration of the PC data for two instances (left and right) of object *table* from the ModelNet40 dataset [137]. The variations in geometry between the two instances are much lower as compared to other datasets. There is practically no occlusion or noise in this data.

the variations in geometry between the two instances are much lower as compared to other datasets.

5.3.2 Object Retrieval using Handcrafted Descriptors

For the evaluation of the descriptor performance, an object retrieval task was chosen. In particular, leave-one-out cross-validation of retrieval was performed. Thus, a descriptor for every object is computed. Given the object's descriptor, the distance to the descriptors of all other objects is computed. The descriptor with the smallest distance is the closest match. When the closest match is of the same category as the query object, it is considered as a correct retrieval, and incorrect otherwise. This step is repeated, each time changing the query object to the next one in the dataset.

Because the ESF, Wahl *et al*. and EPPF descriptors contain the step of random sampling of point pairs from the point set, there are variations in descriptor performance as each time different PPs are chosen. To mitigate this effect, the experiments were repeated ten times, and the mean and the standard deviation values of the metrics were recorded. The retrieval performance is given in Table 5.2. It is possible to observe in Table 5.2 that the proposed EPPF descriptor (in full and short versions) outperforms ESF and OUR-CVFH on all datasets. Furthermore, the EPPF descriptor outperforms the Wahl descriptor on the Stanford and ScanNet datasets but shows comparable performance on the M40 dataset. The superior performance is because of the low level of noise in this dataset. The PPFs employed in the Wahl descriptor are less robust to high levels of noise, but with lower noise levels the PPFs can provide a higher descriptive ability, as compared to the EPPF descriptor. Notably, there is a big difference in total and mean accuracy values for all descriptors. This difference is because the datasets are unbalanced, i.e., some categories happen more often than others, therefore matching to a category with more instances is more likely. Thus, the approaches perform correct retrieval for categories with more objects, which increases the total accuracy, but results in smaller mean accuracy.

Ablation study of feature removal. To gain further insights on the influence of various

Table 5.2: Retrieval performance for the tested handcrafted descriptors. Results averaged over ten iterations. The mean value is given in the corresponding column, while the standard deviation of the measured value is given in brackets. Best performance is shown in bold. The EPPF descriptors outperform the other methods on the Stanford and ScanNet benchmarks. The descriptors show good performance on the ModelNet40 dataset, as compared to Wahl *et al*. Reproduced from [1], ©2018 IEEE.

Dataset	Metric	Descriptor				
		OUR-CVFH [100]	ESF [106]	Wahl [103]	EPPF Short	EPPF
	N_{bins}	308	640	625	540	1200
Stanford [11]	Total accuracy (%)	62.79	71.34 (±0.82)	75.13 (±0.35)	77.26 (±0.40)	**79.18** (±0.40)
	Mean accuracy (%)	42.91	54.54 (±1.10)	57.00 (±1.26)	60.53 (±1.21)	**62.51** (±0.66)
	Mean recall (%)	49.90	52.28 (±1.01)	57.45 (±2.57)	60.16 (±2.61)	**62.58** (±0.59)
	F1-score	0.437	0.530 (±0.011)	0.567(±0.017)	0.601 (±0.017)	**0.625** (±0.007)
ScanNet [138]	Total accuracy (%)	56.23	53.41 (±0.60)	63.72 (±0.32)	63.49 (±0.20)	**65.29** (±0.39)
	Mean accuracy (%)	39.83	33.69 (±0.82)	**45.40** (±0.65)	42.02(±0.49)	44.95 (±0.69)
	Mean recall (%)	38.21	32.72 (±0.98)	45.94 (±0.46)	45.17 (±0.80)	**47.54** (±1.00)
	F1-score	0.382	0.327 (±0.008)	0.444(±0.005)	0.430 (±0.006)	**0.457** (±0.008)
ModelNet40 [137]	Total accuracy (%)	53.22	65.87 (±0.37)	**74.41** (±0.24)	73.00 (±0.21)	73.68 (±0.21)
	Mean accuracy (%)	46.43	58.91 (±0.51)	**67.50** (±0.31)	65.79 (±0.26)	66.43 (±0.31)
	Mean recall (%)	49.26	59.96 (±0.68)	**70.33** (±0.33)	69.12 (±0.36)	69.79 (±0.30)
	F1-score	0.465	0.588 (±0.005)	**0.680** (±0.003)	0.666 (±0.003)	0.671 (±0.003)

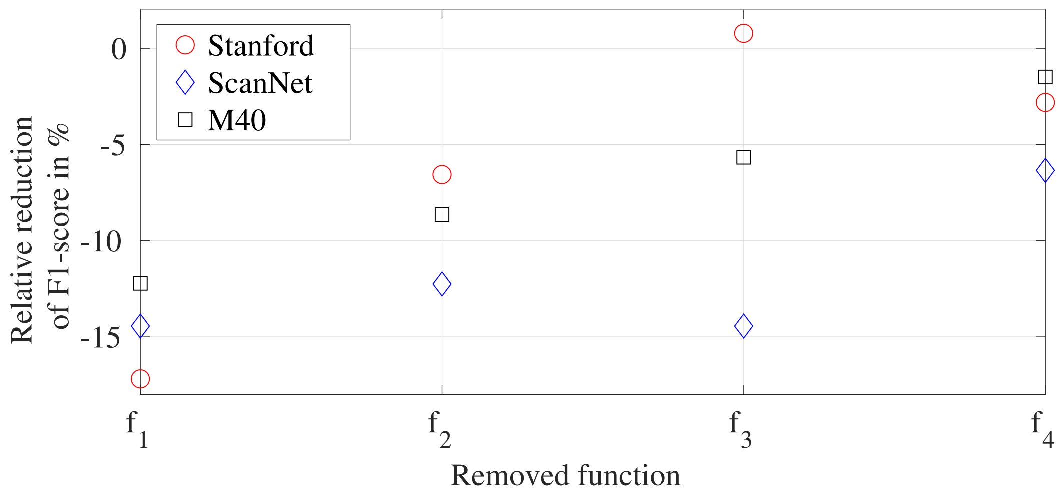


Figure 5.11: Ablation study of various features. Influence of function removal on the retrieval performance (F1-score) is shown for the proposed descriptor. One function is removed at a time. Averaged over ten runs. It can be seen that the most important feature for retrieval is Euclidean distance f_1, whereas occupancy ratio f_4 shows the smallest impact on the result. In case normal information is noisy (f_3 on the Stanford dataset), the removal of feature f_3 can lead to improved performance. Adapted from [1], ©2018 IEEE.

functions on the resulting performance, one PPF was disabled at a time, and retrieval experiments were repeated. As evaluation metric, F1-score was used. For more details see Section 2.1.8. The results for EPPF are given in Fig. 5.11. Here it is possible to see that the largest drop in retrieval performance ($13 - 17\%$) is observed when the Euclidean distance feature f_1 is removed. The performance drop resulting from removing the surface angle function f_2 is lower than from f_1 ($6 - 12\%$). Interestingly, the normal distance function f_3 performs differently on various datasets. In particular, on the Stanford dataset, which exhibits high levels of noise in normal orientation, removal of the normal distance function leads to a performance improvement of 1%. In contrast, on the other datasets, this effect does not happen, and there is a significant drop by up to 15%. Finally, the occupancy ratio function f_4 contributes the least to the overall performance on all datasets and results in a drop of $2 - 6\%$. This ablation study justifies the chosen number of bins for every dimension.

Ablation study of f_1 scaling. To study the effects of different scaling strategies for feature f_1 an ablation study has been performed. In particular, the experimental results of the ablation study for the Stanford dataset are given in Fig. 5.12. It can be seen that the maximum scaling achieves the highest F1-score from the different scaling strategies. Intuitively, by scaling each object into unit cube information on the absolute object dimensions is lost, which leads to deterioration of retrieval performance.

Ablation study of f_2 functions. In order to gain additional insights about the choice of particular function for f_2, another ablation study has been performed. In particular, for function f_2 not only maximum of β_1 and β_2, but also other functions have been evaluated. These functions include: maximum, minimum, and mean of the two values. Additionally, it has been evaluated if a random selection of one of the two values (β_1, β_2) provides a better per-

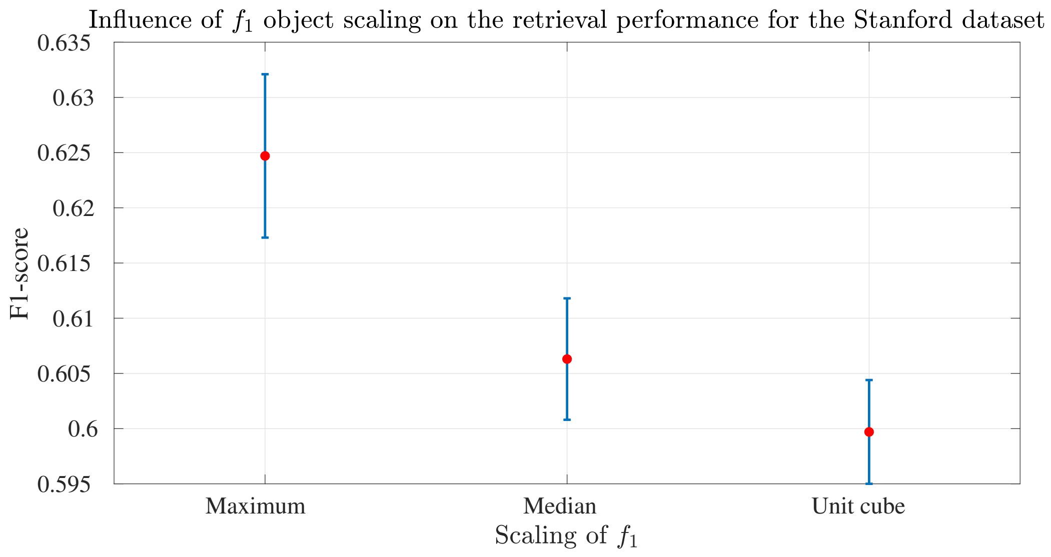


Figure 5.12: Influence of scaling for feature f_1 on the object retrieval performance (F1-score) on the Stanford dataset. The retrieval experiments have been repeated five times. In red, the average value of F1-score is shown. In blue, the standard deviation of the corresponding measured value of F1-score is shown. It can be seen that the maximum object scaling achieves the highest retrieval performance. This can be explained by the fact that scaling each object into unit cube leads to loss of information about the absolute object dimensions. This, in turn, leads to deterioration of retrieval performance.

formance. Furthermore, it has been evaluated if instead of the occupancy ratio function f_4, a modified descriptor is used. It employs β_1, β_2, Euclidean and normal distance features and omits the occupancy ratio feature. The descriptor 4D point pair function is formally defined as follows:

$$\mathbf{f} = (f_1, \beta_1, f_3, \beta_2), \tag{5.13}$$

where β_1 is computed using Equation 5.4 and β_2 is defined in Equation 5.5. f_3 is computed using Equation 5.7 and f_1 is the Euclidean distance that is computed using Equation 5.2. The PPFs are aggregated into a histogram in the same manner as the originally described EPPF descriptor. The experimental results of the ablation study for the Stanford dataset are given in Fig. 5.13. It can be seen that the maximum function achieves the highest F1-score from the different functions. This justifies the choice of the maximum function for f_2. It can also be observed that using β_1 and β_2 and omitting visibility ratio function f_4 leads to the deterioration of the retrieval performance.

5.3.3 Object Classification using Deep Learning Approaches

Evaluation setup. In this step, object classification was evaluated for deep learning approaches. Similar to the previous evaluation, the Stanford, ScanNet and M40 datasets were used. The proposed 4D CNN network was used in combination with the handcrafted feature descriptor. For comparison, 2D and 3D convolution-based networks were also evaluated (denoted as 2D and 3D, respectively). For optimization, Adam optimizer was employed with a learning rate of $5 \cdot 10^{-4}$ and 0.5 dropout probability. Training was performed for $2{,}000$

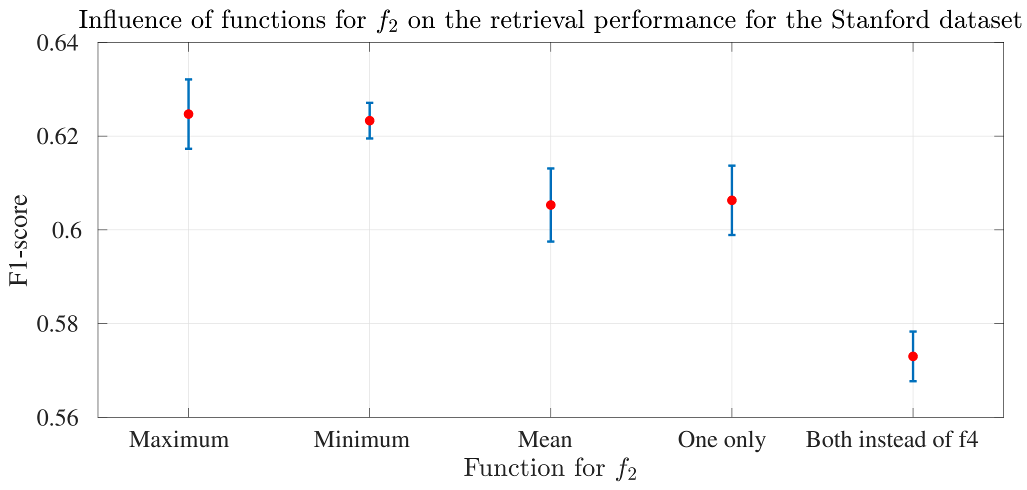


Figure 5.13: Influence of different functions for feature $f_2 = f(\beta_1, \beta_2)$ on the object retrieval performance (F1-score) on the Stanford dataset. The retrieval experiments have been repeated five times. In red, the average value of F1-score is shown. In blue, the standard deviation of the corresponding value of F1-score is shown. “Maximum” denotes maximum function, “Minimum” denotes minimum function. “Mean” denotes average of two values. “One only” denotes randomly selected one of the two values. “Both instead of f4” denotes a modified version of the descriptor described in Equation 5.13. It can be seen that the maximum function achieves the highest retrieval performance.

epochs. Training on ScanNet took between one to three hours to converge with Tensorflow [139] and Nvidia Titan XP graphics processing unit (GPU). For comparison, the method that works directly on the point set (PointNet [114]) was also evaluated. The PointNet network was trained on the given objects while taking into account normalization into a unit cube as advised by the authors[3]. The standard parameter values were used. The input PC to the network contains 2,048 points. Similarly to the proposed approach, training was also performed for 2,000 epochs.

Evaluation results. In Table 5.3 object classification results for both approaches are given. Here, EPPF 4D denotes the proposed 4D convolutional network. The 4D convolution-based network performs better than the 2D- and 3D-variants. This is thanks to the fact that 4D co-occurrences between various dimensions are recorded. In contrast, by reshaping into 2D and 3D, such information is lost. On the Stanford and ScanNet datasets, the 3D network performs better than the 2D-based one. It can be observed that the proposed approach outperforms PointNet on the first two datasets. This can be explained by the fact that the proposed network can easily learn noise-resistant class-specific patterns based on handcrafted descriptors as compared to feeding the point sets directly in PointNet. Notably, PointNet outperforms the proposed approach on the M40 dataset. Here, the obtained PointNet object classification result is different from the one reported by authors in [114] (87.01% vs. 89.2%), because network training has random behavior depending on the chosen random seed when performing optimization. With the lower level of noise in the M40 dataset, PointNet can learn more descriptive representation for object classification. The lower performance of the proposed network is due to the loss of information when operating on PPFs instead of point sets.

[3] https://github.com/charlesq34/pointnet/issues/39

Table 5.3: Classification performance of deep learning approaches using 2D, 3D and 4D convolutional layers on indoor 3D datasets concerning accuracy and F1-score. Greater is better. Best value is shown in bold. EPPF 4D outperforms the other methods on the Stanford and ScanNet datasets. On the ModelNet40 dataset PointNet shows superior performance to EPPF because PointNet can learn more characteristic features from point sets directly when lower levels of noise are observed. Reproduced from [1], ©2018 IEEE.

Dataset	Metric	PointNet [114]	EPPF 2D	EPPF 3D	EPPF 4D
Stanford [11]	Total accuracy (%)	64.30	82.01	81.94	**83.22**
	Mean accuracy (%)	42.48	64.26	**66.37**	65.11
	Mean recall (%)	40.47	70.88	60.94	**72.13**
	F1-score	0.395	0.652	0.665	**0.672**
ScanNet [138]	Total accuracy (%)	63.04	70.39	70.57	**72.10**
	Mean accuracy (%)	37.50	38.98	44.35	**45.70**
	Mean recall (%)	19.53	**63.52**	54.53	56.58
	F1-score	0.209	0.433	0.472	**0.488**
ModelNet40 [137]	Total accuracy (%)	**87.01**	81.64	81.15	82.13
	Mean accuracy (%)	**82.08**	76.37	75.87	77.05
	Mean recall (%)	**83.48**	77.30	77.51	76.99
	F1-score	**0.824**	0.765	0.762	0.769

For better illustration, the confusion matrix of the proposed approach is given in Fig. 5.14. The proposed method has problems distinguishing microwave from lamp and chair, which might be due to its small physical dimensions. Similarly, TV and chair are also often confused. The other categories can be classified by the proposed method much more accurately. For comparison, the confusion matrix for PointNet is given in Fig. 5.15. From the confusion matrix for PointNet it can be observed that the network often confuses various categories with *cabinet*. This effect can be explained by the fact that the network has difficulties learning generalizable and robust features to distinguish different categories.

Noise influence experiment. To investigate the influence of noise on the total accuracy, zero-mean Gaussian random noise with various standard deviation values is added onto 3D coordinates of point sets. This is followed by the re-training of the network using the noisy examples. The results for the proposed 4D approach and PointNet are given in Fig. 5.16. Even though PointNet outperforms the proposed approach on lower levels of noise, with increasing noise levels, the proposed approach suffers no significant decrease in total accuracy. In contrast, PointNet performance starts to drastically deteriorate already at standard deviation values of 0.06 (e.g., 6% of the unit cube size). This can be explained by the fact that the proposed PPFs are more robust to noise as compared to the network architectures trained on point sets directly.

Network response visualization on different layers. To gain further insights about the transformation learned by the network, the network responses are shown for an exemplary object. For this, the object *table* in ScanNet is chosen (see Fig. 5.9) and the corresponding de-

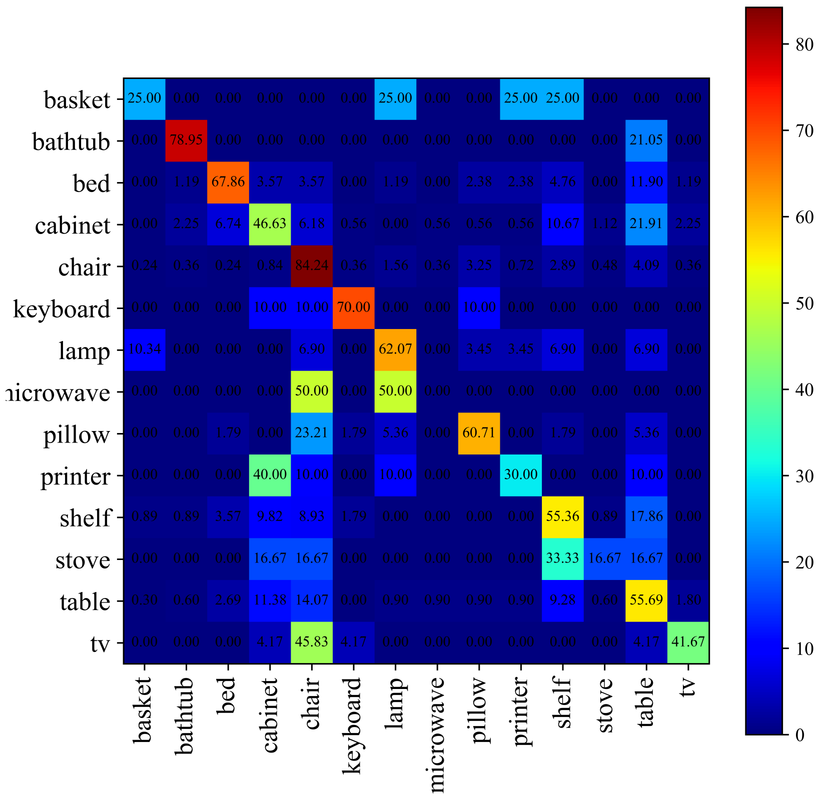


Figure 5.14: Illustration of the confusion matrix for the proposed method on the ScanNet dataset. The proposed method has problems distinguishing microwave from lamp and chair, which might be due to its small physical dimensions. Similarly, TV and chair are also often confused. The other categories can be classified by the proposed method much more accurately.

scriptor values and responses of the first filter in the first two layers are visualized in Fig. 5.17. Observe that the descriptor is very sparse, e.g., a large part of the quantized space takes zero values. Curse of dimensionality is not a big issue here, as the dimensionality of the proposed function space is low (4D) and the space is strongly quantized. 20,000 4D PPF values are aggregated into 1,200 histogram bins. Hence, there are 16.67 counts per bin on average, which further confirms that space is sufficiently sampled. The goal of the descriptor histogram is not to accurately represent the true distribution, but to achieve robustness to variations due to noise and occlusion. Furthermore, when feeding this descriptor into the first 4D convolutional layer, it is possible to observe that the network has smeared this signal in the 4D space. Finally, in the second layer, the signal is even further spread across different dimensions. This is followed by a max-pooling layer that helps to achieve certain invariance to spatial shift and abstract the information. The transformation learned by the network does not only perform simple Gaussian smoothing but, more importantly, it amplifies the signal in certain regions and suppresses the signal in the other regions. This special perturbation benefits the generalization of the proposed network, as the first 4D convolutional layer can learn the fine

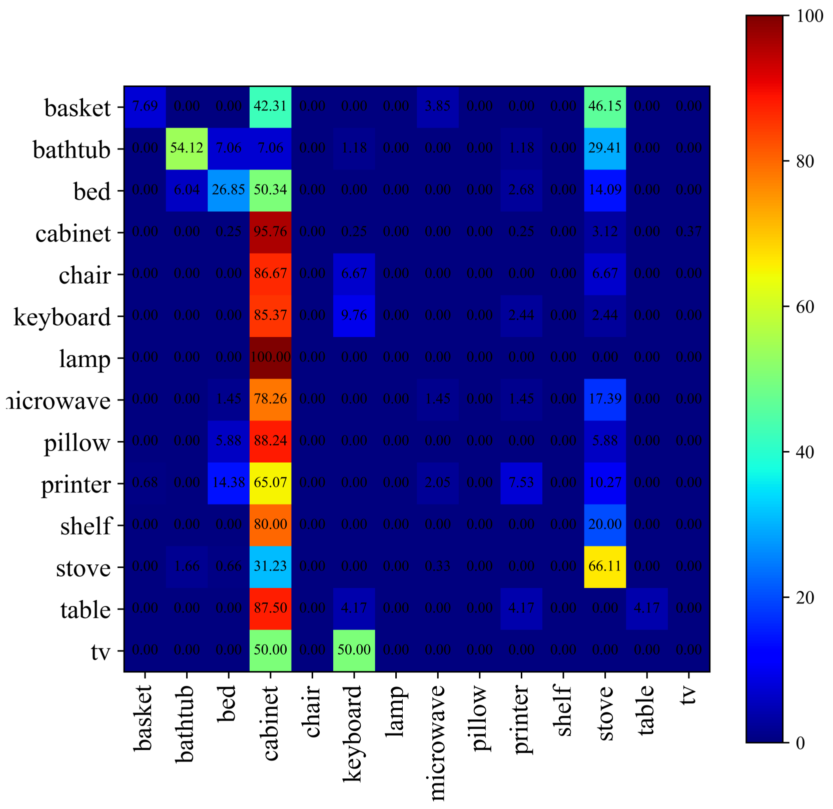


Figure 5.15: Illustration of the confusion matrix for PointNet on the ScanNet dataset. From the confusion matrix for PointNet it can be observed that the network often confuses various categories with *cabinet*. This might be due to the fact that the network is unable to learn the learn generalizable and robust features to distinguish different categories.

features, which are characteristic for certain object categories while suppressing occlusion and noise.

Runtime analysis. The runtime performance of the proposed descriptor was reviewed. The descriptor was implemented in C++ with OpenMP[4] parallelization. For evaluation, desktop personal computer Intel i7 with 24 GB RAM was used. According to performed timings, the descriptor computation took 8 ms per object on average. This value is comparable to runtime performance of the ESF, Wahl *et al*. and OUR-CVFH descriptors. This performance still allows using such descriptor in real-time operation in robotics for perception tasks. As the proposed descriptor provides fixed feature size irrespective of the object dimensions, relatively constant runtime is expected when using the neural network for object classification.

Further insights. Experiments with a number of network architectures for object classification have been performed. Nonetheless, no significant classification performance im-

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

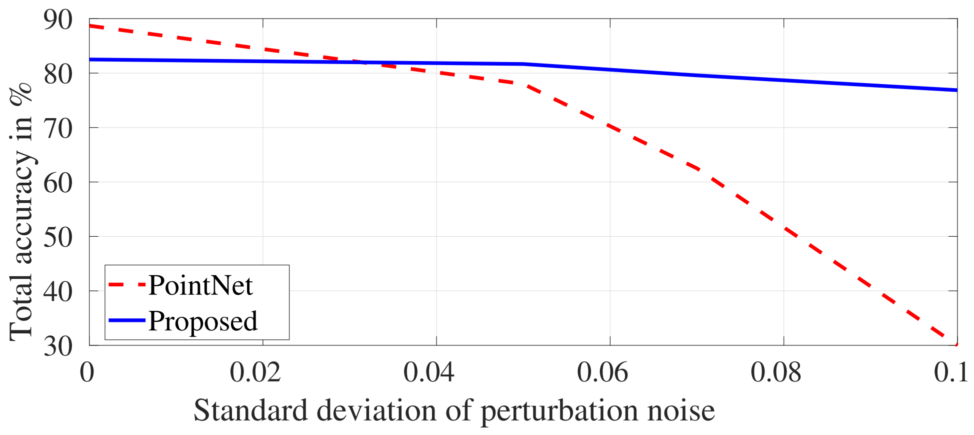


Figure 5.16: Illustration of the influence of standard deviation values for zero-mean Gaussian random noise on the classification accuracy for the M40 dataset using 1024 points. The noise is added to each point coordinate independently. PointNet results taken from [114]. Even though PointNet outperforms the proposed approach on lower levels of noise, with increasing noise levels the proposed approach suffers no significant decrease in accuracy. In contrast, PointNet performance starts to drastically deteriorate already at standard deviation values of 0.06 (e.g., 6% of the unit cube size). Adapted from [1], ©2018 IEEE.

provement has been achieved when using larger architectures, which can be explained by the limited size of the training data. Intuitively, reshaping operations performed in the proposed neural network should remove information about the structure and feature co-occurrences. However, it has been observed that 2D reshaping gave higher classification performance than using 4D blocks directly. This observation could be explained by the fact that the category-specific clusters learned by the network are spatially separated in all dimensions. Alternatively, some other strategies have been considered, such as stacking the dimensions into a 2D representation and global max-pooling. None of them led to any performance improvement.

PointNet generally took much longer to converge as compared to the proposed approach on all datasets. For PointNet, a voting scheme has also been evaluated. This scheme applies multiple perturbations to the PC data and outputs predictions on each of them. The resulting predictions are combined into a single result using a majority vote. No significant performance improvement has been observed. Furthermore, experiments with feeding point pairs directly to PointNet have also been performed. This approach indeed slightly improved performance on noisy datasets, however, only by a small margin. To make sure no local optima influenced the evaluation, the training was repeated several times, and the best test accuracy was reported. Most of the considered datasets have unbalanced categories, e.g., some categories (such as *chair* in ScanNet) occur much more often than others (*lamp*). This fact led to the effect that the network can learn very complex and well generalizable patterns for often occurring objects, while the learned patterns for rarely occurring objects cannot be well generalized. In the future, weighted loss function can be used to take the class imbalance into account. This way, loss value on each class is weighted according to the class occurrence.

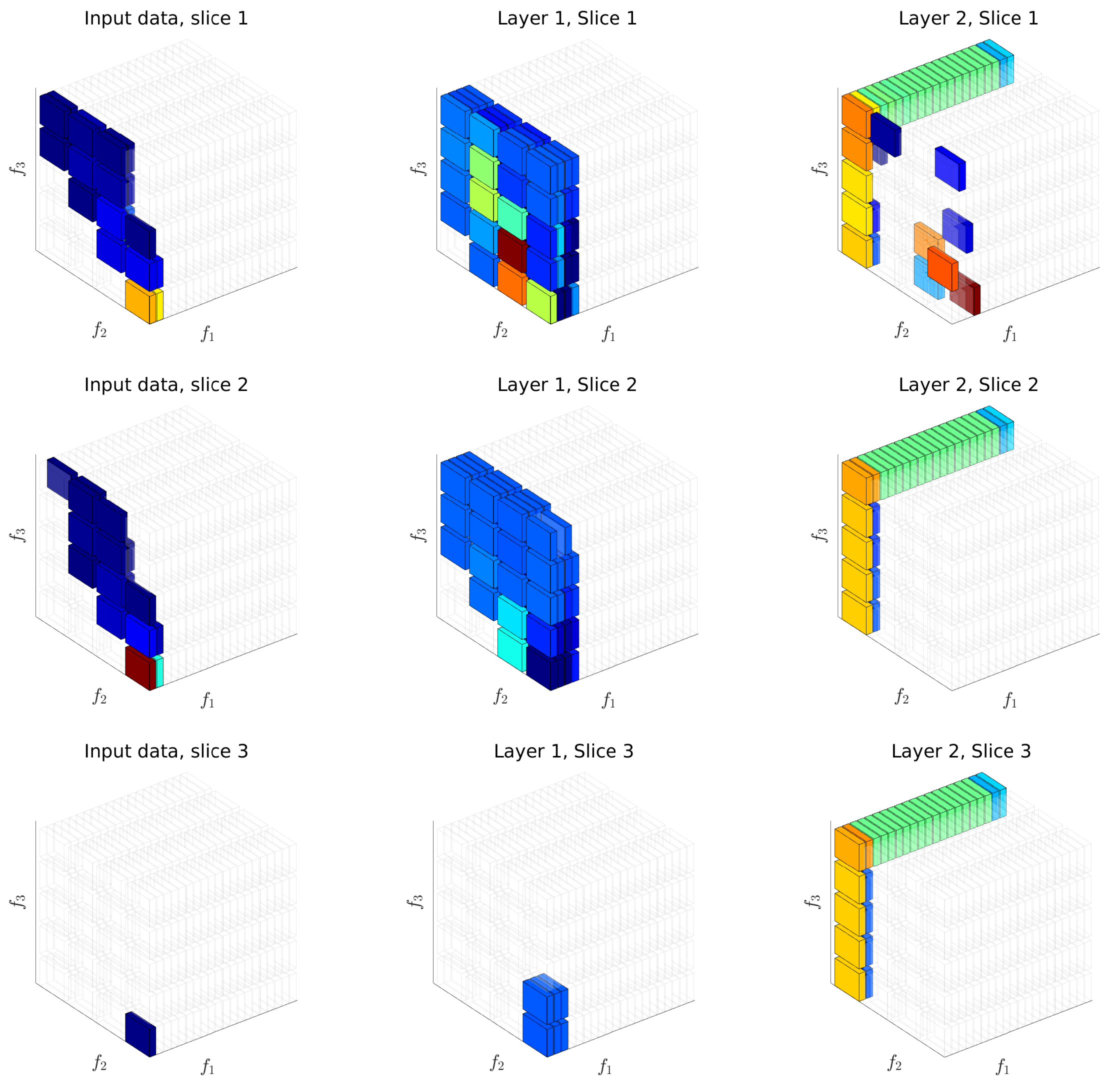


Figure 5.17: 4D descriptor values and the corresponding CNN filter responses for the object *table* (see Fig. 5.9 for PC data) in the ScanNet dataset. Left column: descriptor values. Middle column: responses of the first filter in the first layer. Right column: responses of the first filter in the second layer. The rows show slices of the fourth dimension for descriptor and filter values. Transparent bins correspond to constant offset values for the response (or 0 for the descriptor values), colored bins - to varying values. The bins are colored so that low values are shown in blue color, while high in red.

5.3.4 Limitations

Tuning the hyper-parameters of the network could bring further improvements. In particular, the point sampling strategy can be improved: a straightforward random sampling has been used for this work. It is expected that by performing non-random point sampling, one could further improve the classification performance. For this, the techniques similar to the ones explained by Birdal and Ilic [140] can be applied. Finally, end-to-end learning with the goal of identifying more descriptive point pair and point n-tuple functions could bring further improvements in classification performance, e.g., as in the approach of [120].

5.4 Chapter Summary

In this chapter, a novel method to object classification has been described. To overcome the limitations of related work, including limited performance in the presence of noise and occlusion, a number of improvements have been presented. First, careful choice of PP features for object description is essential for object classification performance. By further combining the 4D histogram of feature co-occurrences into a simple neural network architecture, it is possible to achieve superior object classification performance. Whereas a number of neural network architectures showed high classification performance, 4D convolutional layers outperformed 2D and 3D convolutional layers for this task. Experimental results on 3 benchmark datasets confirmed the superiority of such design in a high noise and occlusion scenario. In particular, an improvement of $9\% - 19\%$ in accuracy and $32\% - 37\%$ in recall can be achieved for object classification on a number of noisy large indoor benchmarks as compared to existing methods. By providing a compact description as input data into a neural network, the learning problem can be simplified and faster convergence can be achieved. The source code of the descriptor is made publicly available[5].

[5] https://github.com/DBobkov/object-descriptor

Chapter 6

Conclusion and Outlook

Semantic understanding of indoor environments allows augmenting a large-volume 3D representation, such as a PC, with semantic labels corresponding to understandable entities, such as *room, table, chair*. By extracting semantic labels, the indoor data becomes searchable, i.e., it is possible to find all elements corresponding to a particular entity within an indoor environment. Such approach allows to significantly simplify handling of data for humans or robotic agents. This fact, in turn, enables multiple applications, such as digital facility management, automated robotics, virtual reality or context-aware indoor navigation.

To this end, this dissertation proposes a top-down approach to the semantic understanding of indoor environments, where the large-scale PC data of buildings is first partitioned into rooms. The PC data corresponding to rooms is further segmented into objects, and each of the objects is classified into a particular semantic category. Such approach gives a scalable solution to semantic understanding by significantly reducing the computational complexity, while keeping the segmentation performance unchanged.

Existing methods to segmentation primarily focused on PC datasets having low levels of occlusion and noise. For example, the ModelNet40 dataset [137], which was most commonly used for evaluation of object classification methods, was generated based on large effort in human manual annotation and refinement of the resulting models. This dataset does not have the same properties as the PC data captured with laser scanners or RGBD sensors in indoor environments under time constraints. In particular, a rapid mapping procedure results in significant occlusion and undersampling artifacts in the collected PC data, which were not adequately addressed by existing methods.

6.1 Summary of the Results

Chapter 3 described the problem of room segmentation in 3D PCs of indoor environments. To address the problem, a human intuition of the room was used to derive a novel volumetric signature for rooms that is based on inner free space. In particular, Section 3.2 described the proposed room segmentation method. To this end, a number of tasks have been solved, such as the computation of the interior free space of indoor environments without assuming a knowledge of scanner poses or the Manhattan world structure. By using a volumetric grid combined with the graph cut algorithm it was possible to estimate inner free space accurately. Hence, no information on sensor poses was required. As the proposed voxel grid was formulated in 3D, buildings violating the Manhattan world structure are also supported. The presented PF signature is robust to object clutter and occlusion that are common in PC data of indoor environments. The experimental evaluation of a number of datasets verified that the proposed method achieves superior performance on a number of datasets collected with LiDAR or RGBD sensors in environments exhibiting curved walls and tilted ceilings.

Chapter 4 discussed the problem of unsupervised object segmentation. To address the particular properties of these datasets, such as non-uniform density and high levels of noise, a typical object segmentation pipeline employing supervoxels and graph cuts was augmented with a novel noise-resilient criterion for the detection of noisy non-convex regions, as described in Section 4.2. By combining the proposed point removal step with state-of-the-art segmentation algorithms, their performance was significantly improved. It was observed that there is a discrepancy between single-view and multi-view PCs in terms of noise level, especially at high-curvature regions. To illustrate this, a new LiDAR PC dataset and its semantic annotation given on several scales were described in Section 4.3. The algorithm showed superior performance on MVPC data, outperforming the existing methods by 20% in terms of OS and US errors. Although the algorithm has been designed for MVPC data, it also achieved state-of-the-art performance on single-view PC data.

Chapter 5 described the problem of object classification. In particular, it has been observed that the object classification is particularly challenging for existing methods when PC data is subject to high levels of noise and occlusion. To address this issue, Section 5.2 discussed the choice of the point pair features that are most descriptive for this task. Furthermore, an approach combining point pair features with a simple 4D deep learning architecture was presented. It allows achieving superior classification performance. The experimental results on three benchmark datasets given in Section 5.3 confirmed the superiority of such design in a high noise and occlusion scenario. In particular, the proposed approach achieved 9%–19% improvement in classification accuracy and $32\% - 37\%$ improvement in recall on two indoor benchmarks.

With the presented methods to the semantic understanding of indoor environments in noisy PC data, it becomes possible to apply automated semantic extraction methods. As human involvement is not strictly required anymore, it becomes feasible to perform automated large-scale understanding of indoor spaces with the goal of creating a searchable digital representation of an indoor environment.

6.2 Limitations and Outlook

In this section, the limitations of the given methods are discussed, and an outlook for the future work is given.

6.2.1 Limitations

Whereas the proposed methods to semantic understanding achieved superior performance on noisy data as compared to existing methods, a number of limitations yet remain:

- The room segmentation approach that was described in Chapter 3 exhibits moderate performance on very sparse PC data. This can be mitigated by extending the criteria of the interior space, e.g., including prior information regarding common orientations or dominant planes in this indoor environment. Another limitation is that the voxel grid requires large storage in case of buildings with huge inner volume (foyer or atrium spanning great volume). To address this limitation, a voxel grid with adaptive voxel size can be used. By carefully adapting the voxel size it might be possible to preserve the same level of room segmentation performance within different parts of the building while reducing computational complexity and storage requirements.
- The proposed method to object segmentation that was described in Chapter 4 exhibited a number of limitations. In particular, removing high-curvature non-convex regions can occasionally result in the situation that the regions become too sparse, thus leading to erroneous oversegmentation of the object. To mitigate this issue, it might be possible to use machine learning techniques for a data-driven convexity criterion. Thus, very sparse noisy regions can be preserved to avoid pruning of intra-object geometry. This step would also prevent the erroneous object undersegmentation. Furthermore, the used criterion of a concave edge is relatively simple and can fail in certain cases of complex object geometry. Finally, the designed method does not take the RGB information into account. Once accurate RGB information is available, object segmentation can significantly benefit from it.
- The method to object classification that was given in Chapter 5 also exhibited a number of restrictions. In particular, instead of a random point sampling, a more sophisticated sampling strategy can be used, thus resulting in improved performance. Furthermore, the chosen point pair functions have been designed by hand and not learned from the data. By learning the best point features from the data directly, the performance can be further improved.

6.2.2 Outlook

Several important directions for future work in the area of semantic understanding of indoor environments can be identified.

Data-based learning. The above-described methods to semantic understanding were in part handcrafted, resulting from the careful design of the most suitable PC-based features. This was due to a limited amount of annotated PC data of indoor environments. With the recent appearance of large semantically annotated indoor PC datasets, such as Matterport3D [141], ScanNet, Stanford3D, and others, it becomes possible to learn the best point-based features using end-to-end learning for these tasks. In particular, there is a promising research direction on how to learn best room features from the formulated volumetric grid of interior free space to estimate the room boundary correctly. It might even be possible to predict whether the given room is a bedroom or an office based on the given free space signature. Similarly, for object classification, the most promising research direction is a development of the most suitable deep learning architectures that can consume PCs directly. The first one to propose a neural network architecture for this task was the PointNet method [114]. Whereas a number of methods working on similar research problem appeared afterward, it remains an important research question what deep learning architecture is most suitable for semantic understanding. Towards this goal, the network should be able to take not only global but also local geometric features into account when performing segmentation, as both of them are crucial for precise semantic segmentation. Furthermore, by using learning-based methods, it might be possible to develop generic methods for recognition, which do not need to be entirely retrained on another dataset. Instead, only the last several layers of the network (typically responsible for classification based on previously extracted features) would need to be fine-tuned on another dataset. This would allow to significantly reduce the required amount of annotated data while producing generalizable models.

Context information. Another important research direction is how to consider context information when performing semantic segmentation. The existing deep learning methods typically apply convolutional layers with a certain receptive field that considers only part of the object geometry at once. In contrast, people strongly leverage context information (e.g., nearby objects, room, building, city) when performing semantic segmentation. For example, a chair is likely to be located in an office environment, whereas jacuzzi is more likely to appear in a spa area of a hotel. Hence, it is important to develop models that are inherently able to include this context information to improve the resulting segmentation performance.

Model-based learning. Many objects in indoor environments typically exhibit certain shapes that can be easily described by humans in a couple of words by following our intuition. For example, a chair is "an object on which one sits, it has a leg (or legs), a seat and a back". Modern supervised learning methods cannot include such object description into their learning. Instead, the methods learn this (or perhaps similar) structure by looking at thousands of chair instances in a cumbersome learning procedure. Instead of doing this in a purely data-driven fashion, it might be beneficial to develop methods that can include model assumptions into their training procedure. By including additional requirements into the optimization procedure a smaller amount of annotated data might be necessary to obtain the same segmentation performance.

Bibliography

Publications by the author

Journal publications

[1] D. Bobkov, S. Chen, R. Jian, M. Z. Iqbal, and E. Steinbach. “Noise-resistant deep learning for object classification in three-dimensional point clouds using a point pair descriptor.” *IEEE Robotics and Automation Letters*, pp. 865–872. 2018. DOI: 10.1109/LRA.2018.2792681.

[2] L. Han, L. Xu, D. Bobkov, E. Steinbach, and L. Fang. “Real-time global registration for globally consistent RGBD SLAM.” *IEEE Transactions on Robotics (T-RO)*, pp. 1–11. 2019. DOI: 10.1109/TRO.2018.2882730.

[3] Z. Jin, M. Z. Iqbal, D. Bobkov, W. Zou, X. Li, and E. Steinbach. “A flexible deep CNN framework for image restoration.” *IEEE Transactions on Multimedia*, pp. 865–872. 2019. DOI: 10.1109/TMM.2019.2938340.

Conference publications

[4] D. Bobkov, S. Chen, M. Kiechle, S. Hilsenbeck, and E. Steinbach. “Noise-resistant unsupervised object segmentation in multi-view indoor point clouds.” In *Proceedings of the International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP)*. 2017, pp. 149–156. DOI: 10.5220/0006100801490156.

[5] D. Bobkov, M. Kiechle, S. Hilsenbeck, and E. Steinbach. “Room segmentation in 3d point clouds using anisotropic potential fields.” In *Proceedings of the International Conference on Multimedia and Expo (ICME)*. 2017, pp. 727–732. DOI: 10.1109/ICME.2017.8019484.

[6] J. Boin, D. Bobkov, E. Steinbach, and B. Girod. “Efficient panorama database indexing for indoor localization.” In *in Proceedings of the International Conference on Content-Based Multimedia Indexing (CBMI)*. 2019.

[7] S. Hilsenbeck, D. Bobkov, G. Schroth, R. Huitl, and E. Steinbach. “Graph-based data fusion of pedometer and wifi measurements for mobile indoor positioning.” In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*. Seattle, Washington: ACM, 2014, pp. 147–158. DOI: 10.1145/2632048.2636079.

[8] D. Bobkov, F. Grimm, E. Steinbach, S. Hilsenbeck, and G. Schroth. “Activity recognition on handheld devices for pedestrian indoor navigation.” In *2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2015, pp. 1–10. DOI: 10.1109/IPIN.2015.7346945.

[9] M. Iqbal, D. Bobkov, and E. Steinbach. “Adaptive fusion-based 3d keypoint detection for rgb point clouds.” In *in Proceedings of the IEEE International Conference on Image Processing (ICIP)*. 2019, pp. 3711–3715. DOI: 10.1109/ICIP.2019.8803680.

General publications

[10] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 7199–7209. DOI: 10.1109/CVPR.2018.00752.

[11] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese. “3d semantic parsing of large-scale indoor spaces.” In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 1534–1543. DOI: 10.1109/CVPR.2016.170.

[12] R. B. Rusu and S. Cousins. “3d is here: Point Cloud Library (pcl).” In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2011, pp. 1–4. DOI: 10.1109/ICRA.2011.5980567.

[13] J. L. Bentley. “Multidimensional binary search trees used for associative searching.” *ACM Communications*. Vol. 18. No. 9, pp. 509–517. 1975. ISSN: 0001-0782. DOI: 10.1145/361002.361007.

[14] S. Ikehata, H. Yang, and Y. Furukawa. “Structured indoor modeling.” In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1323–1331. DOI: 10.1109/ICCV.2015.156.

[15] J. Xiao and Y. Furukawa. “Reconstructing the world’s museums.” *International Journal of Computer Vision (IJCV)*. Vol. 110. No. 3, pp. 243–258. 2014. DOI: 10.1007/s11263-014-0711-y.

[16] R. Ambruş, S. Claici, and A. Wendt. “Automatic room segmentation from unstructured 3d data of indoor environments.” *IEEE Robotics and Automation Letters (RAL)*. Vol. 2. No. 2, pp. 749–756. 2017. DOI: 10.1109/LRA.2017.2651939.

[17] S. Stein, M. Schoeler, J. Papon, and F. Worgotter. “Object partitioning using local convexity.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 304–311. DOI: 10.1109/CVPR.2014.46.

[18] T. Birdal and S. Ilic. “Point pair features based object detection and pose estimation revisited.” In *Proceedings of the International Conference on 3D Vision (3DV)*. 2015, pp. 527–535. DOI: 10.1109/3DV.2015.65.

[19] A. Akbarzadeh, J. Frahm, P. Mordohai, B. Clipp, C. Engels, D. Gallup, P. Merrell, M. Phelps, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewenius, R. Yang, G. Welch, H. Towles, D. Nister, and M. Pollefeys. “Towards urban 3d reconstruction from video.” In *Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT)*. 2006, pp. 1–8. DOI: 10.1109/3DPVT.2006.141.

[20] T. Liu, M. Carlberg, G. Chen, J. Chen, J. Kua, and A. Zakhor. “Indoor localization and visualization using a human-operated backpack system.” In *Proceedings of the International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2010, pp. 1–10. DOI: 10.1109/IPIN.2010.5646820.

[21] R. Huitl, G. Schroth, S. Hilsenbeck, F. Schweiger, and E. Steinbach. “TUMindoor: An extensive image and point cloud dataset for visual indoor localization and mapping.” In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*. Orlando, FL, USA, 2012. DOI: 10.1109/ICIP.2012.6467224.

[22] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. “Kinectfusion: Real-time dense surface mapping and tracking.” In *10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 2011, pp. 127–136. DOI: 10.1109/ISMAR.2011.6092378.

[23] M. Kazhdan and H. Hoppe. “Screened poisson surface reconstruction.” *ACM Transactions on Graphics*. Vol. 32. No. 3, 29:1–29:13. 2013. ISSN: 0730-0301. DOI: 10.1145/2487228.2487237.

[24] M. Blum, R. W. Floyd, V. Pratt, R. L. Rivest, and R. E. Tarjan. “Time bounds for selection.” *Journal of Computer and System Sciences*. Vol. 7. No. 4, pp. 448–461. 1973. ISSN: 0022-0000. DOI: 10.1016/S0022-0000(73)80033-9.

[25] R. B. Rusu, Z.-C. Marton, N. Blodow, M. E. Dolha, and M. Beetz. “Towards 3d point cloud based object maps for household environments.” *Robotics and Autonomous Systems*. Vol. 56, pp. 927–941. 2008. DOI: 10.1109/IROS.2007.4399309.

[26] C. M. Shakarji. “Least-squares fitting algorithms of the nist algorithm testing system.” *Journal of Research of the National Institute of Standards and Technology*, pp. 633–641. 1998. DOI: 10.6028/jres.103.043.

[27] H. Guggenheimer. *Differential Geometry*. McGraw-Hill, 1963. DOI: 10.4310/jdg/1214429071.

[28] R. B. Rusu. “Semantic 3D object maps for everyday manipulation in human living environments.” PhD Thesis. Technische Universität München. München. 2009.

[29] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. “Surface reconstruction from unorganized points.” In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*. 1992, pp. 71–78. ISBN: 0-89791-479-1. DOI: 10.1145/133994.134011.

[30] M. Pauly, M. Gross, and L. Kobbelt. “Efficient simplification of point-sampled surfaces.” In *Proceedings of the IEEE Conference on Visualization*. 2002, pp. 163–170. DOI: 10.1109/VISUAL.2002.1183771.

[31] A. Boulch and R. Marlet. “Fast and robust normal estimation for point clouds with sharp features.” *Computer Graphics Forum*. Vol. 31. No. 5, pp. 1765–1774. 2012. DOI: 10.1111/j.1467-8659.2012.03181.x.

[32] Y. Boykov and V. Kolmogorov. “An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision.” *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*. Vol. 26. No. 9, pp. 1124–1137. 2004. DOI: 10.1109/TPAMI.2004.60.

[33] G. Dantzig, D. Fulkerson, and R. Corporation. *On the Max Flow Min Cut Theorem of Networks*. Rand Corporation, 1964.

[34] A. V. Goldberg and R. E. Tarjan. “A new approach to the maximum-flow problem.” *Journal of the ACM*. Vol. 35. No. 4, pp. 921–940. 1988. ISSN: 0004-5411. DOI: 10.1145/48014.61051.

[35] D. M. Greig, B. T. Porteous, and A. H. Seheult. “Exact maximum a posteriori estimation for binary images.” *Journal of the Royal Statistical Society. Series B (Methodological)*. Vol. 51. No. 2, pp. 271–279. 1989. ISSN: 00359246.

[36] A. A. Markov and N. M. Nagorny. *The Theory of Algorithms*. 1st edition. 2010.

[37] L. Grady. “Random walks for image segmentation.” *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*. Vol. 28. No. 11, pp. 1768–1783. 2006. DOI: 10.1109/TPAMI.2006.233.

[38] P. F. Felzenszwalb and D. P. Huttenlocher. “Efficient graph-based image segmentation.” *International Journal on Computer Vision (IJCV)*. Vol. 59. No. 2, pp. 167–181. 2004. ISSN: 0920-5691. DOI: 10.1023/B:VISI.0000022288.19776.77.

[39] R. Campello, D. Moulavi, A. Zimek, and J. Sander. “Hierarchical density estimates for data clustering, visualization, and outlier detection.” *ACM Transactions on Knowledge Discovery from Data*. Vol. 10. No. 1, pp. 1–51. 2015. DOI: 10.1145/2733381.

[40] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. “A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise.” In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. 1996, pp. 226–231.

[41] J. M. Coughlan and A. L. Yuille. "Manhattan world: Compass direction from a single image by bayesian inference." In *Proceedings of the International Conference on Computer Vision (ICCV)*. 1999, pp. 941–947. DOI: 10.1109/ICCV.1999.790349.

[42] J. Straub, G. Rosman, O. Freifeld, J. J. Leonard, and J. W. Fisher III. "A mixture of manhattan frames: Beyond the manhattan world." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 3770–3777. DOI: 10.1109/CVPR.2014.488.

[43] W. K. Hastings. "Monte carlo sampling methods using markov chains and their applications." *Biometrika*. Vol. 57. No. 1, pp. 97–109. 1970. DOI: 10.2307/2334940.

[44] A. Nobile and A. T. Fearnside. "Bayesian finite mixtures with an unknown number of components: The allocation sampler." *Statistics and Computing*. Vol. 17. No. 2, pp. 147–162. 2007. DOI: 10.1007/s11222-006-9014-7.

[45] E. Forgy. "Cluster analysis of multivariate data: Efficiency versus interpretability of classification." *Biometrics*. Vol. 21. No. 3, pp. 768–769. 1965.

[46] D. Stutz, A. Hermans, and B. Leibe. "Superpixels: An evaluation of the state-of-the-art." *Computer Vision and Image Understanding*. Vol. 166, pp. 1–27. 2018. ISSN: 1077-3142. DOI: 10.1016/j.cviu.2017.03.007.

[47] J. Papon, A. Abramov, M. Schoeler, and F. Worgotter. "Voxel cloud connectivity segmentation - supervoxels for point clouds." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2013, pp. 2027–2034. DOI: 10.1109/CVPR.2013.264.

[48] R. B. Rusu, N. Blodow, and M. Beetz. "Fast point feature histograms (FPFH) for 3d registration." In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2009, pp. 3212–3217. DOI: 10.1109/ROBOT.2009.5152473.

[49] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[50] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. "Dropout: A simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*. Vol. 15, pp. 1929–1958. 2014.

[51] W. M. Rand. "Objective criteria for the evaluation of clustering methods." *Journal of the American Statistical Association*. Vol. 66. No. 336, pp. 846–850. 1971. ISSN: 01621459.

[52] L. Hubert and P. Arabie. "Comparing partitions." *Journal of Classification*. Vol. 2. No. 1, pp. 193–218. 1985. ISSN: 1432-1343. DOI: 10.1007/BF01908075.

[53] A. Richtsfeld, T. Morwald, J. Prankl, M. Zillich, and M. Vincze. "Segmentation of unknown objects in indoor environments." In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2012, pp. 4791–4796. DOI: 10.1109/IROS.2012.6385661.

[54] N. Silberman, D. Sontag, and R. Fergus. "Instance segmentation of indoor scenes using a coverage loss." In *Proceedings of the European Conference on Computer Vision (ECCV)*. 2014, pp. 616–631. DOI: 10.1007/978-3-319-10590-1_40.

[55] C. J. V. Rijsbergen. *Information Retrieval*. 2nd edition. Butterworth-Heinemann, 1979. DOI: 10.1002/asi.4630300621.

[56] S. V. Stehman. "Selecting and interpreting measures of thematic classification accuracy." *Remote Sensing of Environment*. Vol. 62. No. 1, pp. 77 –89. 1997. ISSN: 0034-4257. DOI: 10.1016/S0034-4257(97)00083-7.

[57] C. Mura, O. Mattausch, A. Jaspe Villanueva, E. Gobbetti, and R. Pajarola. "Automatic room detection and reconstruction in cluttered indoor environments with complex room layouts." *Computers and Graphics*. Vol. 44, pp. 20–32. 2014. ISSN: 0097-8493. DOI: 10.1016/j.cag.2014.07.005.

[58] P. K. N. Silberman D. Hoiem and R. Fergus. "Indoor segmentation and support inference from RGBD images." In *Proceedings of the European Conference on Computer Vision (ECCV)*. 2012, pp. 746–760. DOI: 10.1007/978-3-642-33715-4_54.

[59] K. Lai, L. Bo, X. Ren, and D. Fox. "RGB-D object recognition: Features, algorithms, and a large scale benchmark." In *Consumer Depth Cameras for Computer Vision: Research Topics and Applications*. 2013, pp. 167–192. DOI: 10.1007/978-1-4471-4640-7_9.

[60] R. Bormann, F. Jordan, W. Li, J. Hampp, and M. Hägele. "Room segmentation: Survey, implementation, and analysis." In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 1019–1026. DOI: 10.1109/ICRA.2016.7487234.

[61] S. Friedman, H. Pasula, and D. Fox. "Voronoi random fields: Extracting topological structure of indoor environments via place labeling." In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 2007, pp. 2109–2114.

[62] E. Brunskill, T. Kollar, and N. Roy. "Topological mapping using spectral clustering and classification." In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2007, pp. 3491–3496. DOI: 10.1109/IROS.2007.4399611.

[63] L. Fermin-Leon, J. Neira, and J. A. Castellanos. "Incremental contour-based topological segmentation for robot exploration." In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 2554–2561. DOI: 10.1109/ICRA.2017.7989297.

[64] A. Kleiner, R. Baravalle, A. Kolling, P. Pilotti, and M. Munich. "A solution to room-by-room coverage for autonomous cleaning robots." In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 5346–5352. DOI: 10.1109/IROS.2017.8206429.

[65] M. Mielle, M. Magnusson, and A. J. Lilienthal. "A method to segment maps from different modalities using free space layout MAORIS: map of ripples segmentation." In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 4993–4999. DOI: 10.1109/ICRA.2018.8461128.

[66] D. Fleer. "Human-like room segmentation for domestic cleaning robots." *Robotics*. Vol. 6. No. 4. 2017. DOI: 10.3390/robotics6040035.

[67] A. Madugalla, K. Marriott, and S. Marinai. "Partitioning open plan areas in floor plans." In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Vol. 1. 2017, pp. 47–52. DOI: 10.1109/ICDAR.2017.17.

[68] A. Y. Ng, M. I. Jordan, and Y. Weiss. "On spectral clustering: Analysis and an algorithm." In *Advances in Neural Information Processing Systems (NIPS)*. 2001, pp. 849–856.

[69] C. Mura, O. Mattausch, and R. Pajarola. "Piecewise-planar reconstruction of multi-room interiors with arbitrary wall arrangements." *Computer Graphics Forum*. Vol. 35. No. 7, pp. 179–188. 2016.

[70] E. Turner, P. Cheng, and A. Zakhor. "Fast, automated, scalable generation of textured 3d models of indoor environments." *IEEE Journal of Selected Topics in Signal Processing*. Vol. 9. No. 3, pp. 409–421. 2014. DOI: 10.1109/JSTSP.2014.2381153.

[71] S. Ochmann, R. Vock, R. Wessel, and R. Klein. "Automatic reconstruction of parametric building models from indoor point clouds." *Computers & Graphics*. Vol. 54, pp. 94–103. 2016. Special Issue on CAD/Graphics 2015. ISSN: 0097-8493. DOI: doi:10.1016/j.cag.2015.07.008.

[72] J. Jung, C. Stachniss, and C. Kim. "Automatic room segmentation of 3d laser data using morphological processing." *ISPRS International Journal of Geo-Information*. Vol. 6. No. 7. 2017. DOI: 10.3390/ijgi6070206.

[73] C. Mura and R. Pajarola. "Exploiting the room structure of buildings for scalable architectural modeling of interiors." In *ACM SIGGRAPH 2017 Posters*. 2017, pp. 1–2. DOI: 10.1145/3102163.3102213.

[74] L. Li, F. Su, F. Yang, H. Zhu, D. Li, X. Zuo, F. Li, Y. Liu, and S. Ying. "Reconstruction of three-dimensional (3d) indoor interiors with multiple stories via comprehensive segmentation." *Remote Sensing*. Vol. 10. No. 8, p. 1281. 2018. DOI: 10.3390/rs10081281.

[75] S. M. Van Dongen. "Graph clustering by flow simulation." PhD Thesis. University of Utrecht. Utrecht. 2000.

[76] S. Song and J. Xiao. "Sliding shapes for 3d object detection in depth images." In *Proceedings of the European Conference on Computer Vision (ECCV)*. 2014, pp. 634–651. DOI: 10.1007/978-3-319-10599-4_41.

[77] J. Xiao, A. Owens, and A. Torralba. "Sun3d: A database of big spaces reconstructed using sfm and object labels." In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2013, pp. 1625–1632. DOI: 10.1109/ICCV.2013.458.

[78] P. J. Besl and H. D. McKay. "A method for registration of 3-d shapes." *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Vol. 14. No. 2, pp. 239–256. 1992. ISSN: 0162-8828. DOI: 10.1109/34.121791.

[79] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat. "Comparing icp variants on real-world data sets." *Autonomous Robots*. Vol. 34. No. 3, pp. 133–148. 2013. ISSN: 0929-5593. DOI: 10.1007/s10514-013-9327-2.

[80] M. Naseer, S. H. Khan, and F. Porikli. "Indoor scene understanding in 2.5/3d: A survey." *CoRR*. Vol. abs/1803.03352. 2018. arXiv: 1803.03352. [Online]. Available: http://arxiv.org/abs/1803.03352.

[81] H. Jiang. "Finding approximate convex shapes in RGBD images." In *Proceedings of the European Conference on Computer Vision (ECCV)*. Vol. 8691. 2014, pp. 582–596. DOI: 10.1007/978-3-319-10578-9_38.

[82] A. Karpathy, S. Miller, and L. Fei-Fei. "Object discovery in 3D scenes via shape analysis." In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2013, pp. 2088–2095. DOI: 10.1109/ICRA.2013.6630857.

[83] D. Fouhey, A. Gupta, and M. Hebert. "Unfolding an indoor origami world." In *Proceedings of the European Conference on Computer Vision (ECCV)*. Vol. 8694. 2014, pp. 687–702. DOI: 10.1007/978-3-319-10599-4_44.

[84] Z. Deng, S. Todorovic, and L. Jan Latecki. "Semantic segmentation of RGBD images with mutex constraints." In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1733–1741. DOI: 10.1109/ICCV.2015.202.

[85] O. van Kaick, N. Fish, Y. Kleiman, S. Asafi, and D. Cohen-Or. "Shape segmentation by approximate convexity analysis." *ACM Transactions on Graphics*, pp. 1–11. 2014. DOI: 10.1145/2611811.

[86] F. T. K. Tateno and N. Navab. "Real-time and scalable incremental segmentation on dense SLAM." In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2015, pp. 4465–4472. DOI: 10.1109/IROS.2015.7354011.

[87] J. J. Koenderink. "What does the occluding contour tell us about solid shape." *Perception*. Vol. 13. No. 3, pp. 321–330. 1984. DOI: 10.1068/p130321.

[88] M. Bertamini and J. Wagemans. "Processing convexity and concavity along a 2-d contour: Figure–ground, structural shape, and attention." *Psychonomic bulletin & review*. Vol. 20. No. 2, pp. 191–207. 2013.

[89] M. Schoeler, J. Papon, and F. Worgotter. "Constrained planar cuts - object partitioning for point clouds." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 5207–5215. DOI: 10.1109/CVPR.2015.7299157.

[90] O. Mattausch, D. Panozzo, C. Mura, O. Sorkine-Hornung, and R. Pajarola. "Object detection and classification from large-scale cluttered indoor scans." *Computer Graphics Forum*. Vol. 33. No. 2, pp. 11–21. 2014. DOI: 10.1111/cgf.12286.

[91] A. Ecins, C. Fermüller, and Y. Aloimonos. "Cluttered scene segmentation using the symmetry constraint." In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 2271–2278. DOI: 10.1109/ICRA.2016.7487376.

[92] T. Pham, T.-T. Do, N. Sünderhauf, and I. Reid. "SceneCut: Joint Geometric and Object Segmentation for Indoor Scenes." In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 1–9. DOI: 10.1109/ICRA.2018.8461108.

[93] S. Gupta, P. Arbeláez, and J. Malik. "Perceptual organization and recognition of indoor scenes from RGB-D images." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2013, pp. 564–571. DOI: 10.1109/CVPR.2013.79.

[94] N. Soni, A. M. Namboodiri, C. Jawahar, and S. Ramalingam. "Semantic classification of boundaries of an RGBD image." In *Proceedings of the British Machine Vision Conference (BMVC)*. 2015, pp. 1–12. DOI: 10.5244/C.29.114.

[95] L. P. Tchapmi, C. B. Choy, I. Armeni, J. Gwak, and S. Savarese. "Segcloud: Semantic segmentation of 3d point clouds." In *International Conference on 3D Vision (3DV)*. 2017, pp. 537–547. DOI: 10.1109/3DV.2017.00067.

[96] L. Yi, W. Zhao, H. Wang, M. Sung, and L. J. Guibas. "GSPN: generative shape proposal network for 3d instance segmentation in point cloud." *CoRR*. Vol. abs/1812.03320. 2018.

[97] A. Aldoma, Z.-C. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R. B. Rusu, S. Gedikli, and M. Vincze. "Tutorial: Point cloud library: Three-dimensional object recognition and 6 dof pose estimation." *IEEE Robotics and Automation Magazine*. Vol. 19, pp. 80–91. 2012. DOI: 10.1109/MRA.2012.2206675.

[98] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu. "Fast 3d recognition and pose using the viewpoint feature histogram." In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2010, pp. 2155–2162. DOI: 10.1109/IROS.2010.5651280.

[99] A. Aldoma, M. Vincze, N. Blodow, D. Gossow, S. Gedikli, R. B. Rusu, and G. Bradski. "Cad-model recognition and 6dof pose estimation using 3d cues." In *Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. 2011, pp. 585–592. DOI: 10.1109/ICCVW.2011.6130296.

[100] 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." *Proceedings of the Joint 34th DAGM and 36th OAGM Symposium on Pattern Recognition*, pp. 113–122. 2012. DOI: 10.1007/978-3-642-32717-9_12.

[101] Z.-C. Marton, D. Pangercic, N. Blodow, and M. Beetz. "Combined 2d-3d categorization and classification for multimodal perception systems." *International Journal on Robotics Research*. Vol. 30. No. 11, pp. 1378–1402. 2011. ISSN: 0278-3649. DOI: 10.1177/0278364911415897.

[102] F. Tombari, S. Salti, and L. Di Stefano. "Unique signatures of histograms for local surface description." In *Proceedings of the 11th European Conference on Computer Vision Conference on Computer Vision (ECCV)*. 2010, pp. 356–369. DOI: 10.1007/978-3-642-15558-1_26.

[103] E. Wahl, U. Hillenbrand, and G. Hirzinger. "Surflet-pair-relation histograms: A statistical 3d-shape representation for rapid classification." In *Proceedings of the IEEE International Conference on 3-D Digital Imaging and Modeling (3DIM)*. 2003, pp. 474–481. DOI: 10.1109/IM.2003.1240284.

[104] B. Drost, M. Ulrich, N. Navab, and S. Ilic. "Model globally, match locally: Efficient and robust 3d object recognition." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2010, pp. 998–1005. DOI: 10.1109/CVPR.2010.5540108.

[105] S. H. Kasaei, L. S. Lopes, A. M. Tomé, and M. Oliveira. "An orthographic descriptor for 3d object learning and recognition." In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 4158–4163. DOI: 10.1109/IROS.2016.7759612.

[106] W. Wohlkinger and M. Vincze. "Ensemble of shape functions for 3d object classification." In *Proceedings of the IEEE International Conference on Robotics and Biomimetics*. 2011, pp. 2987–2992. DOI: 10.1109/ROBIO.2011.6181760.

[107] M. Pedersoli and T. Tuytelaars. "A scalable 3d hog model for fast object detection and viewpoint estimation." In *Proceedings of the International Conference on 3D Vision (3DV)*. Vol. 1. 2014, pp. 163–170. DOI: 10.1109/3DV.2014.82.

[108] J. P. S. d. M. Lima and V. Teichrieb. "An efficient global point cloud descriptor for object recognition and pose estimation." In *29th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. 2016, pp. 56–63. DOI: 10.1109/SIBGRAPI.2016.017.

[109] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin. "Shape distributions." *ACM Transactions in Graphics*. Vol. 21. No. 4, pp. 807–832. 2002. ISSN: 0730-0301. DOI: 10.1145/571647.571648.

[110] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz. "Aligning point cloud views using persistent feature histograms." In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2008, pp. 3384–3391. DOI: 10.1109/IROS.2008.4650967.

[111] L. Kiforenko, B. Drost, F. Tombari, N. Krüger, and A. G. Buch. "A performance evaluation of point pair features." *Computer Vision and Image Understanding (CVIU)*. Vol. 166, pp. 66 –80. 2018. ISSN: 1077-3142. DOI: 10.1016/j.cviu.2017.09.004.

[112] S. Hinterstoisser, V. Lepetit, N. Rajkumar, and K. Konolige. "Going further with point pair features." In *Proceedings of the European Conference on Computer Vision (ECCV)*. 2016, pp. 834–848. DOI: 10.1007/978-3-319-46487-9_51.

[113] T. Furuya and R. Ohbuchi. "Deep aggregation of local 3d geometric features for 3d model retrieval." In *Proceedings of the British Machine Vision Conference (BMVC)*. 2016, pp. 1–12. DOI: 10.5244/C.30.121.

[114] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. "Pointnet: Deep learning on point sets for 3d classification and segmentation." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 77–85. DOI: 10.1109/CVPR.2017.16.

[115] M. Tatarchenko, J. Park, V. Koltun, and Q.-Y. Zhou. "Tangent convolutions for dense prediction in 3d." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 3887–3896. DOI: 10.1109/CVPR.2018.00409.

[116] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. "Dynamic graph CNN for learning on point clouds." *arXiv preprint arXiv:1801.07829*. 2018.

[117] D. Rethage, J. Wald, J. Sturm, N. Navab, and F. Tombari. "Fully-convolutional point networks for large-scale point clouds." In *Proceedings of the IEEE European Conference on Computer Vision (ECCV)*. 2018, pp. 625–640. DOI: 10.1007/978-3-030-01225-0_37.

[118] 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 (NIPS)*. 2018, pp. 828–838.

[119] B.-S. Hua, M.-K. Tran, and S.-K. Yeung. "Pointwise convolutional neural networks." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 984–993. DOI: 10.1109/CVPR.2018.00109.

[120] H. Deng, T. Birdal, and S. Ilic. "PPFNet: Global context aware local features for robust 3d point matching." In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 195–205. DOI: 10.1109/CVPR.2018.00028.

[121] H. Deng, T. Birdal, and S. Ilic. "PPF-FoldNet: Unsupervised learning of rotation invariant 3d local descriptors." In *Proceedings of European Conference on Computer Vision (ECCV)*. 2018, pp. 620–638. DOI: 10.1007/978-3-030-01228-1_37.

[122] C. Harris. *Dictionary of architecture & construction*. McGraw-Hill, 2006. DOI: 10.1036/0071452370.

[123] Y. Furukawa, B. Curless, S. Seitz, and R. Szeliski. "Reconstructing building interiors from images." In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2009, pp. 80–87. DOI: 10.1109/ICCV.2009.5459145.

[124] I. E. Sutherland, R. F. Sproull, and R. A. Schumacker. "A characterization of ten hidden-surface algorithms." *ACM Computing Survey*. Vol. 6. No. 1, pp. 1–55. 1974. ISSN: 0360-0300. DOI: 10.1145/356625.356626.

[125] B. K. P. Horn. "Extended gaussian images." *Proceedings of the IEEE*. Vol. 72. No. 12, pp. 1671–1686. 1984. ISSN: 0018-9219. DOI: 10.1109/PROC.1984.13073.

[126] B. Andres, T. Beier, and J. Kappes. "OpenGM: A C++ library for discrete graphical models." *CoRR*. Vol. abs/1206.0111. 2012. [Online]. Available: http://arxiv.org/abs/1206.0111.

[127] A. Y. Ng, M. I. Jordan, and Y. Weiss. "On spectral clustering: Analysis and an algorithm." In *Proceedings of the 14th International Conference on Neural Information Processing Systems (NIPS)*. 2001, pp. 849–856.

[128] C. Fiorio and J. Gustedt. "Two linear time union-find strategies for image processing." *Theoretical Computer Science*. Vol. 154. No. 2, pp. 165–181. 1996. ISSN: 0304-3975. DOI: 10.1016/0304-3975(94)00262-2.

[129] S. Song, S. P. Lichtenberg, and J. Xiao. "SUN RGB-D: A RGB-D scene understanding benchmark suite." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 567–576. DOI: 10.1109/CVPR.2015.7298655.

[130] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz. "Towards 3D point cloud based object maps for household environments. robotics and autonomous systems." *Journal Robotics and Autonomous Systems*. Vol. 56. No. 11, pp. 927–941. 2008. DOI: 10.1016/j.robot.2008.08.005.

[131] W. Gellert, H. Kustner, M. Hellwich, and H. Kaestner. *The VNR concise encyclopedia of mathematics*. 2nd edition. Van Nostrand Reinhold New York, 1989. ISBN: 0442205902. DOI: 10.1007/978-94-011-6982-0.

[132] K. Mo, S. Zhu, A. X. Chang, L. Yi, S. Tripathi, L. J. Guibas, and H. Su. "Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding." *CoRR*. Vol. abs/1812.02713. 2018. arXiv: 1812.02713. [Online]. Available: http://arxiv.org/abs/1812.02713.

[133] H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller. "Multi-view convolutional neural networks for 3d shape recognition." In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 945–953. DOI: 10.1109/ICCV.2015.114.

[134] K. Pearson. "Note on regression and inheritance in the case of two parents." *Proceedings of the Royal Society of London*. Vol. 58, pp. 240–242. 1895. DOI: 10.1098/rspl.1895.0041.

[135] S. Kullback and R. A. Leibler. "On information and sufficiency." *The Annals of Mathematical Statistics*. Vol. 22. No. 1, pp. 79–86. 1951. DOI: 10.1214/aoms/1177729694.

[136] T.-C. Wang, J.-Y. Zhu, E. Hiroaki, M. Chandraker, A. A. Efros, and R. Ramamoorthi. "A 4d light-field dataset and CNN architectures for material recognition." In *Proceedings of the European Conference on Computer Vision (ECCV)*. 2016, pp. 121–138. DOI: 10.1007/978-3-319-46487-9_8.

[137] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. “3d shapenets: A deep representation for volumetric shapes.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1912–1920. DOI: 10.1109/CVPR.2015.7298801.

[138] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. “ScanNet: Richly-annotated 3d reconstructions of indoor scenes.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 2432–2443. DOI: 10.1109/CVPR.2017.261.

[139] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, and M. D. et al. “Tensorflow: Large-scale machine learning on heterogeneous distributed systems.” *arXiv preprint arXiv:1603.04467*. 2016.

[140] T. Birdal and S. Ilic. “A point sampling algorithm for 3d matching of irregular geometries.” In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 6871–6878. DOI: 10.1109/IROS.2017.8206609.

[141] 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.” *International Conference on 3D Vision (3DV)*, pp. 667–676. 2017. DOI: 10.1109/3DV.2017.00081.

List of Figures

List of Tables