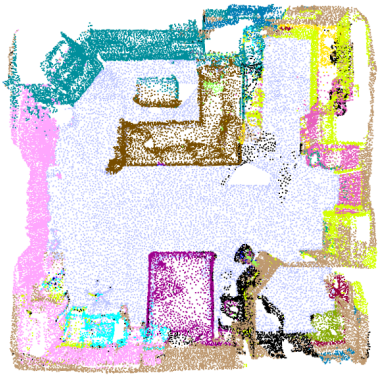
Learned 3D Semantic Segmentation using Point Clouds

Groundtruth



Prediction

Michael Seeber

Bachelor Thesis

October 2018

Prof. Dr. Marc Pollefeys

Prof. Dr. Konrad Schindler

Advisors:

Ian Cherabier, Audrey Richard, Martin Oswald

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CVG Computer Vision and Geometry Lab

PRS Photogrammetry Remote Sensing

Abstract

3D reconstruction tries to model the environment with the highest accuracy possible. Recent research has shown, that the introduction of class-specific shape priors (e.g. the ground is mostly flat, while walls are vertical and reach the ground...) can greatly improve the quality of reconstructions. However, this technique relies on good and efficient semantic segmentation of a scene. Existing semantic 3D reconstruction methods apply classic 2D image segmentation to obtain this information. On the contrary, our thesis explores the idea of using point clouds as inputs and doing the segmentation step directly on them. Point clouds are intrinsically sparse and therefore not as memory intensive as discretized representations like voxel grids. However, because of their unstructured nature new challenges arise. Moreover, we implemented a pipeline that converts the segmentation results into a datacost for a state-of-the-art shape prior reconstruction algorithm. From our experiments we obtained an overall segmentation accuracy of 71.4 % on the ScanNet dataset, although important classes like "floor" got predicted more precisely at 94.4%. This results in 3D reconstructions that more or less correctly represent the scene, but struggle with fine grained detail in densely packed areas.

Zusammenfassung

Das Ziel der 3D-Rekonstruktion ist die möglichst präzis Darstellung der Umgebung. Erst kürzlich hat die Forschung ergeben, dass klassenspezifische Annahmen (z.B. der Boden ist flach, während Wände vertikal darauf stehen) die Qualität erheblich verbessern können. Diese Methode beruht jedoch auf einer akkuraten und effizienten semantischen Segmentierung. Bestehende semantische 3D-Rekonstruktionsmethoden wenden klassische 2D-Bildsegmentierung an, um diese Information zu erhalten. Im Gegensatz dazu, versuchen wir in dieser Arbeit den Segmentierungsschritt direkt auf Point Clouds auszuführen. Point Clouds können anhand einer Liste von Koordinaten sehr effizient repräsentiert werden, weshalb sie im Gegensatz zu Voxelgittern nicht sehr speicherintensiv sind. Wegen ihrer Unstrukturiertheit ergeben sich jedoch neue Probleme. Zusätzlich haben wir eine Pipeline implementiert, welche die Segmentierungsergebnisse so umwandelt kann, dass sie von einem Rekonstruktionsalgorithmus weiterverwendet werden können. Dieser optimiert anschliessend sowohl die Geometrie als auch die Semantik. Unsere Experimente auf dem ScanNet-Datensatz haben eine Genauigkeit von 71,4% für die Segmentierung ergeben. Wichtige Klassen, wie beispielsweise der "Boden" konnten allerdings zu 94,4% Prozent korrekt erkannt werden. Dadurch war es für die Rekonstruktionsmethode möglich, die Szenen mehrheitlich korrekt darzustellen. Nur in sehr dichten Bereichen mit vielen Objekten kam es zu Komplikationen mit den Details.

Acknowledgments

I want to thank my supervisors Ian Cherabier and Audrey Richard for their enormous support, advice and valuable input during all the different stages of this thesis. Without their assistance and dedicated involvement, this thesis would have never been accomplished. Furthermore, I would like to thank Martin Oswald for giving me the opportunity to work on this interesting project.

Contents

List of Figures

List of Tables

1

Introduction

Ever since the rise of computer vision, 3D reconstruction has been a core problem. The goal is to create an as close as possible digital representation of the real world. This is useful for a wide range of applications. Varying from photo-realistic visual reproduction of cultural heritage to recently being used even for the authentication of users on mobile devices. The understanding of the surrounding 3D geometry of an environment is also of high importance for robots, UAVs like drones or self-driving cars, because they navigate and execute critical decisions based on the data provided. Thus, having high quality reconstructions at hand is crucial for a reliable operation.

However, because of the greater availability of 2D sensors like classical cameras, most of the research focused on multi-view reconstruction. The idea behind is to infer the 3D geometry of objects from images that are taken in different positions by fusing them together. Despite its long history in computer vision, many problems remain unsolved with this approach. In particular, textureless or reflective regions as well as image noise lead to ambiguities and therefore introduce uncertainty. This lack of information emerges in artefacts on the reconstructed scenes. For example, flat surfaces like floors have "holes" or furniture is hanging midair instead of being anchored to the floor. To further improve the results, strong additional priors are needed. Recent advances in semantic image classification methods led to the development of methods

that jointly optimize the geometry and the semantics of scenes. This allows to solve problems like the ones mentioned above. Because of prior knowledge we know that floors need to be more or less a horizontal plane in space and potentially missing areas can be filled accordingly. This approach was first introduced in [HZCP17] where they used depth maps together with 2D semantic segmentation to receive a volumetric semantic reconstruction of scenes with higher quality. They formulate the task as a variational multi label problem, where each voxel is either freespace or belongs to one of the semantic classes. Although they obtained impressive results, their work still relies on hand tuned regularizers that are based solely on the surface normal directions. Thus, they are overly simplistic and cannot fully capture the complex semantic and geometric dependencies of our 3D world. The next logical step introduced by [CSO+18] was to learn these regularizers. They propose to express the gradient through a more general matrix, which enables to model every directional and semantic interaction. Their work will serve as baseline for our reconstruction approach.

So far, both of these methods require a voxel grid as input. Voxel grids are usually generated by segmenting multi-view images and fusing them together with the depth maps. This yields a base volumetric representation of the scenes with segmentation labels assigned to each voxel [SCD+06]. However, there is a dilemma with voxel grids, as high resolutions result in a very large memory footprint. This is due to the rasterization of the whole area, even if there is no useful information present (e.g. in the free space area). To tackle this problem, more efficient data structures like octrees [BVR+16] or tetrahedralization [RVB+17] can be used instead of regular fixed-size voxel grids. Nevertheless, all of these methods still rely on an arbitrary discretization which leads to a loss of information and thereby limits the possible performance gains.

A more natural representation of 3D environments are point clouds. Point clouds are a set of data points located in 3D space that resemble objects by covering their surface areas. It is also the representation that gets produced by most 3D scanners, as it comes closest to their principle of operation. LIDARs and most other 3D sensors emit rays of light and measure distances for a large number of points on the external surfaces of surrounding objects. Even smartphones started to feature depth sensors, thereby sparking more interest for this kind of representation in recent years.

The reason why people have not worked intensively with point clouds before is mainly due

to their lack of structure. Most of the recent advances in computer vision rely on deep neuronal architectures, all of which require highly regular input. While images and their 3D counterpart voxel grids fulfill this requirement, point clouds cannot be used as input directly. With [QSMG17], the foundation work was laid out on how to handle point clouds in deep neuronal networks.

Therefore, the idea behind our thesis was to exploit the fact that point clouds are a more efficient representation of the environment and provide enough information for meaningful semantic segmentation. Receiving this information directly from captured point clouds would allow to get rid of the 2D image segmentation that most state-of-the-art reconstruction algorithms rely on. In this thesis, we propose a pipeline that takes a point cloud as input, processes it with a PointNet fashioned architecture and encodes the results in a volumetric datacost. Then, our pipeline feeds this datacost into a reconstruction network like the one proposed by [CSO+18]. That network has learned shape priors for each semantic class during training. Using these priors, it is able to infer a good 3D reconstruction of the original point cloud by combining the geometric as well as the semantic information from the first network. In contrast to most current 3D reconstruction approaches, our pipeline solely relies on a captured rgb-colored point cloud and doesn't require a valid semantic segmentation of the scene, that gets mostly extracted through 2D image segmentation.

Thesis Organization

This thesis is structured into a total of 5 chapters, including this introduction. In Chapter 2, we examine existing work related to semantic segmentation and 3D reconstruction. In Chapter 3, we describe the theory and implementation details of our segmentation and reconstruction methods and proceed with explaining our pipeline that connects them together. In Chapter 4, we present the results from our experiments and discuss them. In the 5th and last chapter we conclude our findings and provide an outlook on future work.

2

Related Work

This thesis has been inspired by several other papers in the field of both semantic segmentation as well as 3D reconstruction. Both fields have been thoroughly studied, so a lot of concepts are reused and combined in order to explore a novel approach. In the following two sections, the most relevant papers for this thesis are briefly described.

2.1. Semantic Segmentation

Semantic segmentation is seen as one of the key problems of computer vision because of its wide range of application. Besides its huge role in medical imaging, it is also used for autonomous driving and virtual reality systems to name a few.

Before the introduction of deep learning, most methods included some kind of feature extraction. The simplest image segmentation method is thresholding. Values above and below a specific clip-level are mapped correspondingly to 1 and 0, which results in a binary image. More sophisticated classical image segmentation approaches rely on clustering methods or region extraction [FM81]. Nowadays nearly every approach is based on certain deep neuronal networks because of their superior performance. Examples are GoogLeNet [SLJ+15] or AlexNet

[KSH12], which was the pioneering deep convolutional neuronal network (CNN) for semantic 2D image segmentation [GGOEO+17].

Since we live in a 3D world, most applications include some kind of 3D segmentation tasks. Therefore, many different approaches have been studied for the segmentation of 3D space. In [WSK+15] the authors explore 3D deep believe networks that use voxel grids as input. Although it seems reasonable to consume a 3D representation when trying to segment 3D space, this method is only feasible for very small voxel grids due to performance issues. Other papers like [SMKLM15] tried to solve this issue by recognizing 3D shapes through their 2D views. A 2D view is just an image of the scene from a specific viewpoint. Multiple segmented images from different viewpoints can be fused together by multi-view CNNs in order to receive a semantic 3D segmentation. Intuitively volumetric representations feature more information, thus volumentric CNNs should outperform multi-view CNN approaches, which have to make decisions from 2D data. However, the contrary is the case. Given the fact that 2D image sensors are more widely available, multi-view segmentation was state-of-the art for a long time.

The primary geometric representation captured by 3D scanners, as mentioned in the introduction, are point clouds. Consequently, the input needs to be converted into multi-view images or voxel grids, before the mentioned methods can be applied. As 3D scanners get more common and start to appear in consumer hardware, it would be favorable to work directly with a more natural and efficient representation such as point clouds. The main drawback of point clouds is their absence of structure, hence classical convolutional architectures don't work for them, as they require highly regular input formats. Pioneering work on how to handle point clouds was done in PointNet [QSMG17]. In this paper the authors propose to achieve input order invariance by the use of symmetric functions over the inputs. Building up on that idea, a more sophisticated version named PointNet++ [QYSG17] was published that takes spatial localities into account, which yields slightly better results. In PointCNN [LBSC18] they introduce a different approach for working with point clouds, namely using χ-Convolutions in the local feature extraction stage, instead of being max-pooling based like PointNet.

To conclude, each method has its own advantages and limitations. Because multi-view CNNs work so well, most 3D reconstruction algorithms utilize it for obtaining semantic information. One goal of this thesis is to get rid of this semantic image segmentation and process a 3D data format, namely point clouds, instead.

2.2. 3D Reconstruction

The completion of 3D shapes has a long history in geometry processing and is one of the core interests of Computer Vision. Related work on this topic either focuses on the aspect of receiving high quality reconstruction regardless of run times or has been devoted to achieving real-time performance.

At the heart of many classical reconstruction algorithms lies the concept of inferring 3D geometry from a collection of images. This so called multi-view reconstruction attracted a lot of research interest because 2D sensors were widely available. It works by tracking features like edges or corners throughout multiple images, which were taken from different viewpoints. This makes it possible to approximate their positions in space, which can be subsequently used to reconstruct the scene. The vast majority of multi-view 3D reconstruction algorithms rely on photo-consistency measures [KS00] to find correlations between frames. Over time, many sophisticated photo-consistency metrics have been devised, although in transparent or low textured areas they all fail. To compensate for this lack of accurate information additional priors are required, such that the precise geometry can be recovered.

To fix this issue, many techniques started to utilize total variation (TV) as regularizer. The TV-norm seeks to reduce the surface area of 3D shapes [KKBC09], however this often leads to overly smooth reconstructions. In [HZCP17] they introduce semantic segmentation as an additional metric, which can be used besides photo-consistency. Indeed they showed that the semantic segmentation of a scene has the potential to vastly improve the reconstruction results, by jointly optimizing the geometry and semantics.

Deep learning methods have recently allowed to infer such shape priors directly from data, instead of manually crafting them as in [HZCP17]. However, a drawback is the excessive amount of parameters involved in such approaches, due to being based on generic convolution neuronal network architectures. As a consequence, enormous amounts of training data are required for obtaining satisfying results [DRB$^+$17].

The work done in Learning Priors for Semantic 3D Reconstruction [CSO$^+$18] was very important for this thesis as it tries to tackle the problems of generic CNN approaches. The idea of the paper is to embed variational regularization into a neuronal network. In contrast to previ-

ously mentioned methods, they incorporate structural constraints via unrolled variational inference. This limits the amount of parameters and therefore makes the approach lightweight. It is also end-to-end trainable, hence it is able to capture more complex dependencies which results in visually more appealing reconstructions. Furthermore, it is magnitudes faster than previous methods and therefore a first step in the direction to close the gap to real-time reconstruction algorithms.

One approach that focused on real-time performance is KinectFusion [NIH+11]. They were able to achieve real-time performance by fusing all of the depth data streamed from a Kinect sensor into a single global implicit surface model. In the last years research has increased considerably in this area, as real-time performance is crucial for a lot of applications such as AR, self driving cars and a variety of other uses.

3

Method

This chapter presents the theoretical foundation and methodological approach employed throughout this thesis. Section 3.1 introduces the theory behind PointNet and also provides an insight into our own implementation of the PointNet approach. The network for learning shape priors and further implementation details are elaborated in Section 3.2 . Finally, a thorough outline on the proposed pipeline for connecting the two parts is provided in Section 3.3 . This leads to a method that is able to infer 3D reconstructions from incomplete point clouds. To fully understand this chapter, basic knowledge of neuronal networks and deep learning is required.

3.1. PointNet

PointNet is a deep learning architecture that is capable of reasoning about 3D geometries directly from point clouds. It either predicts a class label for a point cloud as a whole or through a small extension for each single point. This can be used for object classification, part segmentation or semantic scene parsing, which is the application we are interested in.

3.1.1. Theory

Intuitively the high level idea behind the PointNet architecture is that it learns to summarize a shape by a spare set of key points. In figure 3.1 one can see that the key points resemble the skeleton of an object and therefore serves as an abstraction. This allows the network to draw conclusions for completely new point clouds, which feature similar key points.

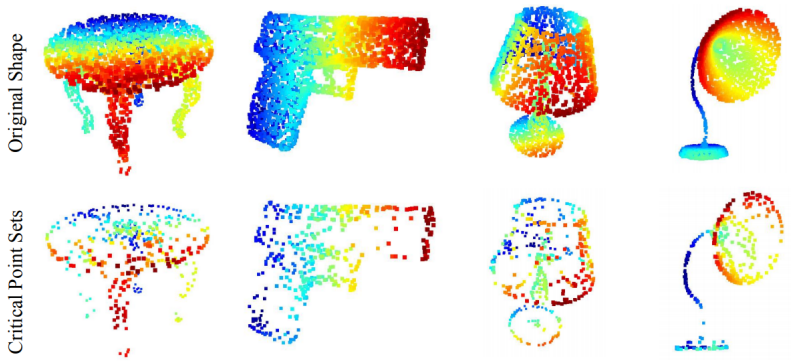


Figure 3.1.: *Visualization of key points [QSMG17]*

The PointNet architecture is surprisingly straight forward as in the inital stages each point is processed identically and independently. The key concept in the approach is the introduction of a single symmetric function for achieving input order invariance. This is essential, because we always want to receive the same result, no matter in which order we input the points of our point cloud. Mathematically this can be expressed as:

$$f(x_1, \ldots, x_n) \approx g(h(x_1), \ldots, h(x_n)) \tag{3.1}$$

$$\begin{aligned} \text{where} \quad & h : \mathbb{R}^N \to \mathbb{R}^K \\ & f : 2^{\mathbb{R}^N} \to \mathbb{R} \\ & g : \underbrace{\mathbb{R}^K \times \ldots \times \mathbb{R}^K}_{n} \to \mathbb{R} \text{ is a symmetric function} \end{aligned}$$

The idea is to approximate a general function f on a point set by applying a symmetric function g on elements, which got transformed by h beforehand. Translated to the actual implementation within a network, h gets approximated through a multi-layer perceptron net (MLP) and g by a composition of a single variable function and a max-pooling layer as visualized in Figure 3.2.

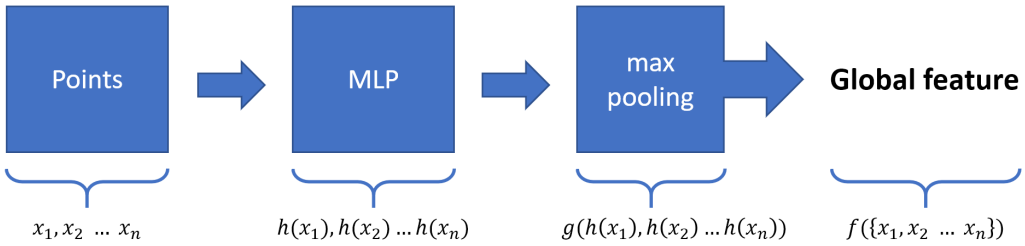


Figure 3.2.: *Visualization of the PointNet function*

In other words, this means that the network learns criterias for selecting interesting and informative points and also encodes the reason for choosing them. The max-pooling layer yields a global feature vector that would need to be further processed by fully connected layers for receiving a classification score. This can be seen in Figure 3.3 which shows the complete PointNet architecture. However, as we are only interested in semantic segmentation, we focus on how the network can be extended to handle this very similar task.

For semantic point segmentation, not only local information but also the combination with global knowledge is essential. This is because we have to predict a label for each single point of the input, instead of a global prediction that can be assigned to the whole point cloud, as in object classification tasks. Without additional knowledge of the surrounding points, it is impossible to provide accurate predictions, as they are more or less all the same. In PointNet they achived this in a simple yet highly effective manner. Instead of generating a score from the global feature vector they concatenate it with each point after the pre-processing stage, which we will describe in more detail shortly. Through the concatenation the global context got encoded into each single point. Afterwards they extract new per point features, which are this time aware of both the local and global information. With this extension, illustrated in Figure 3.3 (Segmentation Network), we are able to predict per point characteristics that rely on local geometry and global semantics.

Concerning the pre-processing, we expect the semantic labeling of a point cloud to be invariant, even if the cloud undergoes certain geometric transformations like translation, rotation or scaling. A straight forward solution would be to align all the inputs to a canonical space before extracting features. However, in [QSMG17] they propose a much simpler idea that has a similar outcome, namely T-Nets. The idea behind T-Nets is to predict an affine transformation matrix

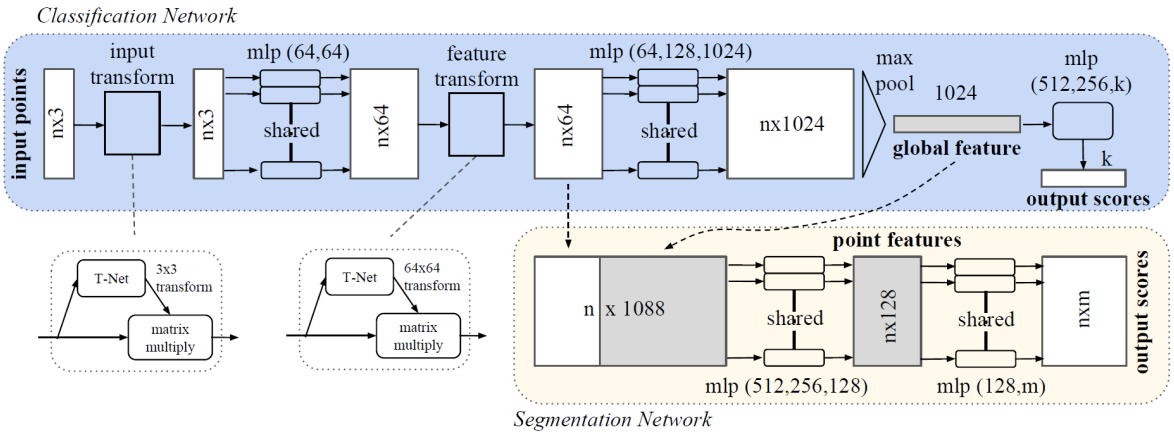


Figure 3.3.: *PointNet architecture [QSMG17]*

that can be directly applied to the coordinates of the input. This yields a representation that is invariant to the previously mentioned rigid transformations. However, our experiments have shown that these T-Nets are only suited for object classification tasks. In segmentation tasks we have to deal with point clouds that consist of multiple objects, thus making it more difficult for a T-Net to find a canonical pose.

3.1.2. Implementation

Our implementation of a point cloud segmentation network is closely related to the architecture proposed in PointNet [QSMG17]. In the following section we will outline the most important differences between our approach and the vanilla PointNet architecture. At the end of this section you can find an illustration of our network in Figure 3.5 together with some additional information.

- In PointNet each point is represented by a 9 dimensional vector. Beside XYZ and RGB, they also provide the normalized location relative to the room. We only use a colored point cloud as input, therefore each point is represented by only 6 values.
- We assign each point to one of 21 semantic classes (NYU labels) that are listed in the Appendix A.1. The authors of PointNet used only 13 classes for their experiments, which should be taken into account when comparing the performance.
- Instead of extracting $1m^3$ blocks from the scenes like in the original paper, we use a

block size of $1.5m$ x $1.5m$ x $3m$ that spans from the floor to the ceiling of the scene. This increase helps in capturing a more expressive global feature vector.

- Furthermore we are not using a T-Net as described in section 3.1.1, since they are not able to come up with a canonical pose for point clouds that consist of multiple objects. Instead, we randomly rotate the blocks before we feed them into the network.

The most important addition, that made the results usable, was the introduction of a weighted loss. For the evaluation of PointNet for semantic segmentation the authors used the Standford Indoor Dataset [ASZS17]. It turned out to be quite uniform regarding the presence of semantic classes as it includes just office scenes. In contrast the ScanNet dataset [DCS+17], which we use for this thesis covers a very broad area, ranging from gyms over bathrooms and apartments to classrooms. Additionally, the ScanNet dataset has huge differences in number of points per class within a scene, which implies that not every class is present in each scan. Therefore using the original loss function resulted in really bad segmentations as visualized in Figure 3.4 when trained on the whole dataset.

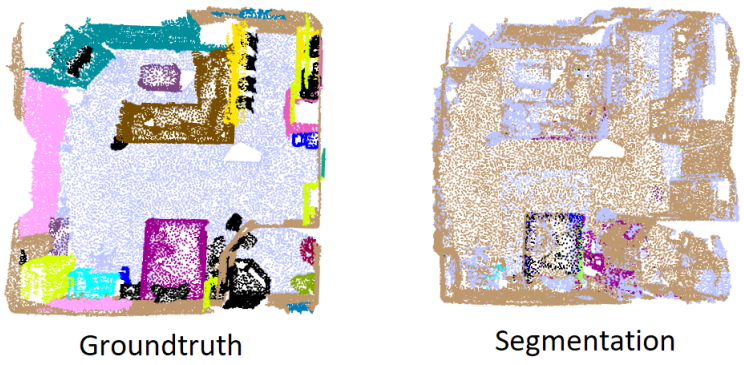


Figure 3.4.: *Visualization of segmentation results with old loss function*

It turned out to primarily predict either wall or floor, as these two semantic classes are present throughout all scenes. Interestingly even these two classes got mixed up a lot. Although this was improved with the new loss function, the fact that the network struggled learning the orientation of classes so heavily, inspired another idea. Instead of feeding randomly rotated blocks during training, we started to limit the rotations to be around the z-axis only. For our dataset this limitation makes sense, as the orientation of entities can only vary along the z-axis throughout

different scenes. In Chapter 4 we discuss more about the huge impact this small adoption had on our segmentation performance. In the end, the new weighted cross entropy loss can be expressed through:

$$\mathcal{L} = \sum_{c=1}^{21} w_c \cdot y_{x,c} \cdot \log(p_{x,c}) \quad \text{where}$$

$$w_c \text{ denotes the weight for class c and } \sum_{c=1}^{21} w_c = 1$$

$$y_{x,c} \in \{0, 1\} \text{ binary indicator if c is the correct predicted class for point x}$$

$$p_{x,c} \text{ probability of x being predicted to be class c}$$

In Figure 3.5 you can see our network architecture. As we work on colored point clouds our input consists of 6 values for each point. Besides the geometric coordinates (x, y, z), there are three additional fields containing the rgb-color value. After they passing through 5 convolutional layers the values get saved, such that they can be later concatenated to the global feature vector. As described in Figure 3.2 the max-pool layer corresponds to the symmetric function h in Equation 3.1. After concatenation we apply two more convolution. During training, an additional drop-out layer is active with a probability of 0.7 to fight over-fitting. In a final step we convolve everything to the amount of semantic classes, in order to receive a probability for each class. One limitation of using fully connected layers is their obligation for a fixed input size. For our experiments we fed 32 batches of 8192 points per epoch during training. Because we use 21 semantic classes the last convolutional layer matches this number in size, as visible in Figure 3.5. More information about the experiments can be found in Chapter 4.

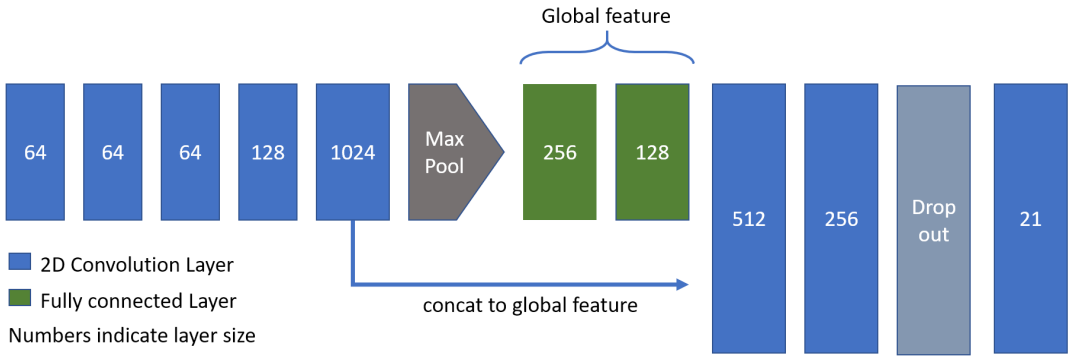


Figure 3.5.: *Model of our segmentation network*

3.2. Learned Priors for Semantic 3D Reconstruction

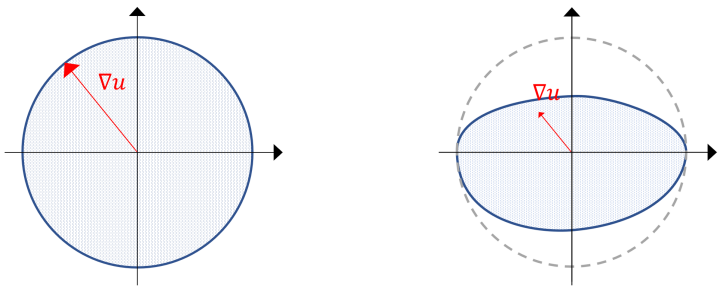
Objects are very hard to reconstruct when the data is inaccurate or tenuous. This is predominantly caused by reflections, transparency or lack of texture. A possible solution is to infer the missing geometry from other objects of the same class, as they have similar shapes. Early approaches manually expressed such shape priors through Wulff shapes, though recently efforts were made to learn them directly from data. We relied on the work done by [CSO+18] since their method enables good results, while keeping the amount of parameters low.

3.2.1. Theory

The traditional variational approach minimized the following energy function to obtain the best labeling u for each point in space.

$$\underset{u}{minimize} \int_{\Omega} (\underbrace{\phi_x(u)}_{regularization} + \underbrace{fu}_{data\ fidelity}) \, dx \quad \text{subject to} \quad \forall x \in \Omega : \sum_{\ell} u_\ell(x) = 1 \tag{3.2}$$

The regularization term is in charge of dealing with noise and missing data. Without one the solution would have a lot of incomplete areas and would not be smooth. There are various handcrafted regularizers available, from total variation (TV) as the simplest to more sophisticated ones, that can be expressed through Wulff shapes. Total variation $\phi_x(u) = \lambda g(x) \|\nabla u(x)\|_2$ corresponds to just minimizing the surface area of a 3D object. The smoothness cost $\phi_x(u)$ for a gradient direction $\nabla u(x)$ can be visualized by a unit circle as in Figure 3.6(a).



(a) TV-Norm (b) Wulff-Shape

Figure 3.6.: *Possible regularizers*

To encode more complex shape priors Wulff shapes can be used. Figure 3.6(b) shows the smoothness cost of a Wulff shape $\phi_x(u) = \lambda \max_{\xi \in \mathcal{W}_\phi} \langle \xi, \nabla u \rangle$. The biggest advantage of the introduced regularizers is their characteristic of being convex. Therefore it is possible to efficiently compute a global minimizer to the previously stated energy minimization problem 3.2. However this useful property also limits them in their expressiveness. Hence they are not powerful enough to correctly represent the true statistics of the underlying problem [KKHP17] In [CSO+18] it is proposed to express the gradient operator through a more general matrix W to overcome the limitations of hand-crafted regularizers. For W they choose to use a 6-dimensional matrix such that every directional and semantic interaction in 3D can be modeled. Mathematically this can be expressed as:

$$\phi_x(u) = \|Wu\|_2 \quad \text{with} \quad W \in \mathbb{R}^{2\times2\times2\times|\mathcal{L}|\times|\mathcal{L}|\times3} \tag{3.3}$$

$2 \times 2 \times 2$ models forward-backward differences for computing the gradients
$|\mathcal{L}| \times |\mathcal{L}|$ models interaction between any combination of semantic label
3 models the spatial dimensions

Because W is now directly represented, fewer parameters than with Wulff shapes are required. This leads to significantly less memory usage as they showed in their paper [CSO+18]. From 3.3 follows the new energy minimization problem 3.4.

$$\underset{u}{minimize} \int_\Omega (\|Wu\|_2 + fu)\, dx \quad \text{subject to} \quad \forall x \in \Omega : \sum_\ell u_\ell(x) = 1 \tag{3.4}$$

In their work they used a first-order primal-dual (PD) algorithm [18] to find a minimum to the above problem. However, this requires to reformulate 3.4 into a saddle point problem. More information on how to accomplish this can be found directly in their paper [CSO+18]. After all, they obtained 4 update equations, which can be applied iteratively to approach the solution.

3.2.2. Implementation

The network architecture consists of 3 stages, namely an encoder, the optimization component and a decoder at the end. Furthermore, the whole network consists of 3 levels, with each one reasoning at a different scale, particular at full, half and quarter resolutions. In other words,

the third layer works on $4 \times 4 \times 4$ voxel blocks and passes the more coarse grained knowledge onto the second layer. This one works on $2 \times 2 \times 2$ blocks and can therefore further refine the information before it gets passed to the full resolution level. This arrangement helps to model semantic interactions in a more global context, which reflects on the reconstruction performance. Additionally, it allows information to propagate faster which leads to faster convergence. An illustration of the just described general network architecture can be found in Figure 3.7. Throughout the following paragraphs we provide additional insights into each of the three stages.

The encoding stage (yellow in Figure 3.7) works directly on the input datacost and serves several purposes. The previously described multi scale architecture relies on down sampled inputs. By using separate encoders for each level of the network, this automatically happens through average pooling. Furthermore, the encoding process reduces low level noise and normalizes the influence of different semantic classes with respect to each other.

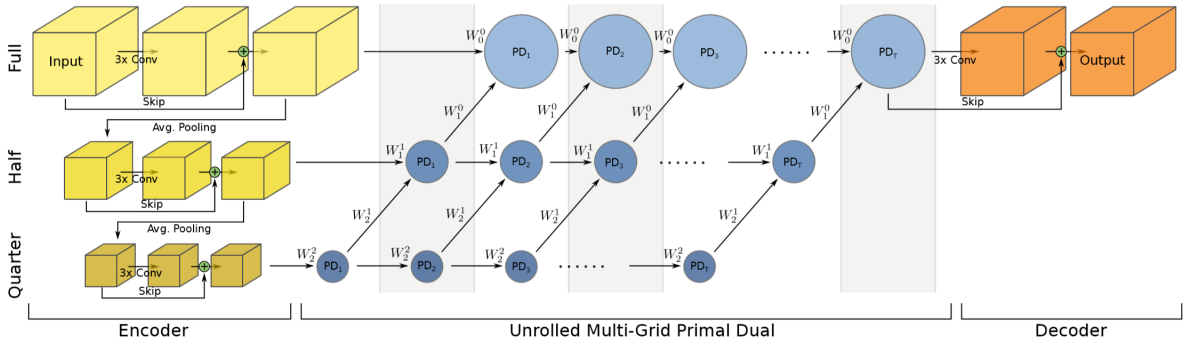


Figure 3.7.: *Reconstruction network architecture [CSO+18]*

The most important component is the optimization stage. As described in Section 3.2.1, the energy minimization problem 3.4 can be approximated through primal-dual processing steps. This comes in handy because the iterations of the primal-dual algorithm can be unrolled for a fixed number of steps. Compared to using a classical 3D convolutional neural network this approach is more efficient, because weight sharing allows for fewer parameters. In our implementation we unroll the primal-dual for 50 iterations. In Figure 3.7, each iteration is represented through a blue circle with the iteration number as subscript. The unrolling is done for each scale level separately. Because higher levels rely on information from lower scale levels we obtain a cascading information flow, where dual-primal updates on the same layer are processed concurrently.

As a last step the solution from the final dual primal iteration gets fed into the decoding stage. Similar to the encoding stage it consists of ReLu activated convolutions. Their purpose is to smooth and increase contrast, allowing the final softmax activated convolution to make more accurate decisions.

Additionally, we had to incorporate a weight factor similar to Section 3.1.2 into the cross entropy loss in order to avoid predicting mainly freespace.

3.3. Proposed Pipeline

To connect the semantic segmentation with the reconstruction approach mentioned above, we propose a pipeline as illustrated in Figure 3.8. In the following subsections we provide additional implementation details on parts of the pipeline that have not been described in Section 3.1 or 3.2 yet.

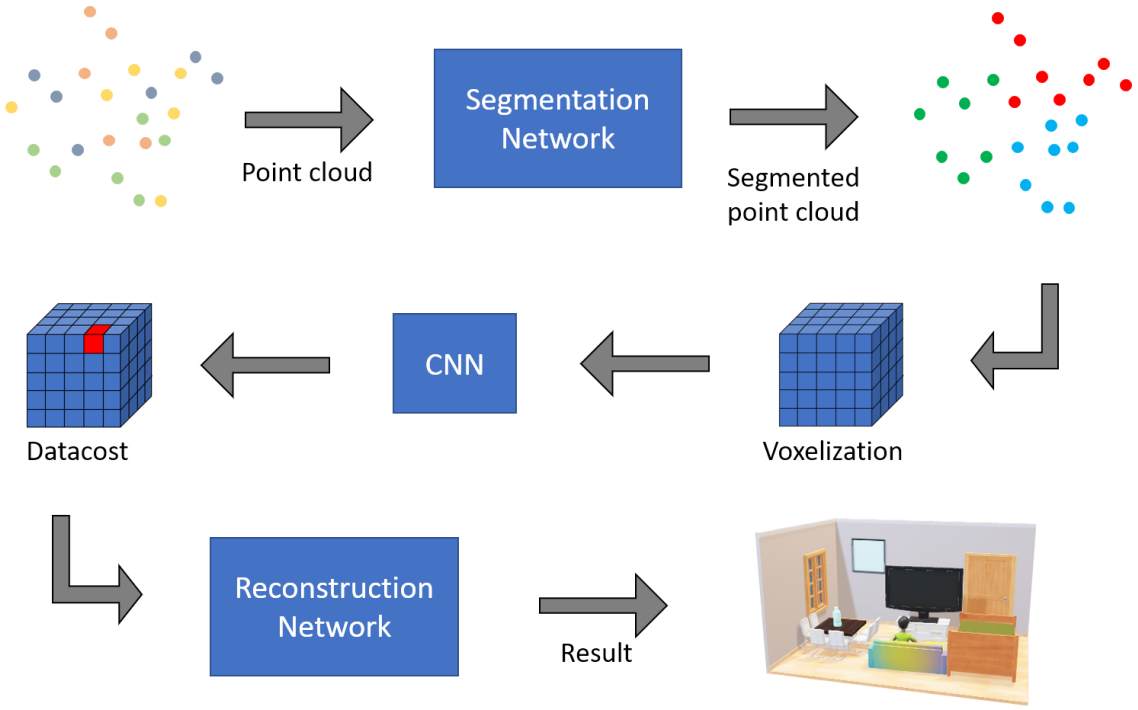


Figure 3.8.: *Proposed Pipeline*

3.3.1. Voxelization

After receiving a semantic labeled point cloud by a previously trained segmentation network we need to create an informative datacost out of it. This datacost can be used to train our reconstruction network as well as for the evaluation of new scenes. Because the reconstruction approach we use only works on voxelgrids, the first step consists in converting the segmented point cloud into one. This is done by rasterizing the whole scene with a resolution of $5cm$. In the next step each voxel gets assigned an array containing 23 values. The first 21 values represent the probabilities of the possible classes we predict in the segmentation step. As point clouds are sparse not every voxel contains a point, therefore the 22nd entry is set to 1 if the voxel belongs to the new class "freespace". In the last entry we encode the amount of points that lie within a voxel. This allows us to express "density" in our datacost. To keep the density value in the same range as the other 22 entries we normalize it to $[0, 1]$ by dividing through the maximal amount of points inside a single voxel. Figure 3.9 shows a visualization of the voxelgrid.

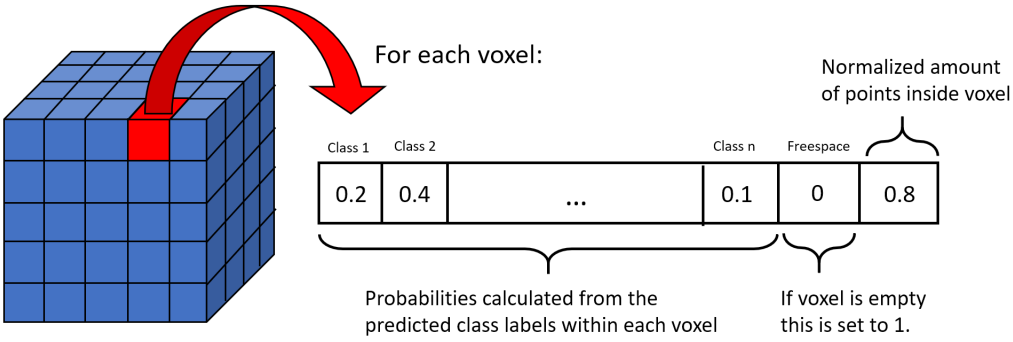


Figure 3.9.: *Voxelgrid from segmented point cloud*

3.3.2. Generation of datacost

To generate our final datacost, we apply a small CNN on the previousy created voxelgrid. The main purpose of this CNN is to reduce the dimensions from $W \times H \times D \times 23$ to $W \times H \times D \times 22$ and to combine semantic information from nearby voxels. We do this by applying 3D-convolutions, such that the 23 channel values get compressed into 22 at the end. In particular it consists of a $\tanh$ activated point-wise 3D convolution followed by two additional $\tanh$ activated 3D convolutions that operate on 3x3x3 and 6x6x6 blocks respectively. The last step is again a $\tanh$

point-wise convolution that compresses everything back to 22 channels. Figure 3.10 illustrates the just described model. The final datacost is then written to disk as a numpy array for further processing in the reconstruction step.

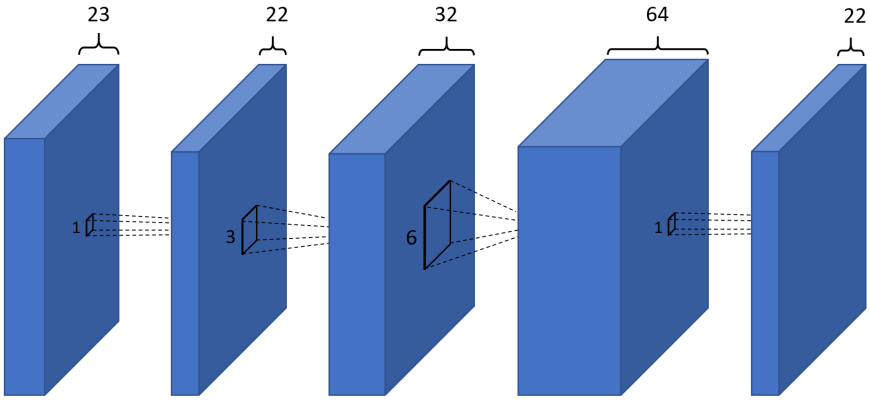


Figure 3.10.: *Visualization of intermediate CNN*

We have also experimented with solely a single point-wise convolution as well as more complex models, however the impact of such were neglectable. The above model is a good balance of both worlds, as it is simple, yet able to yield better results than a sole point-wise convolution.

3.3.3. Extracting mesh

The reconstruction network yields a voxel grid as output where each voxel belongs to one class. To receive a surface mesh we replace the value of all voxels that got classified as freespace with 0 and everything else with 1. We use "Lewiner marching cubes" algorithm [LLVT03] to find surfaces within our predicted voxelgrid, such that we receive our final mesh.

4

Experiments

In this chapter we initially provide some information about the data we used for our experiments, as well as other experiment setup details. In the subsequent sections we discuss our results.

4.1. Dataset

We use ScanNet [DCS+17] as dataset. It provides annotated RGB-D scans of various real world indoor environments. The whole dataset features 2.5M RGB-D images that got acquired from 1513 scans. Each scan also includes estimated camera parameters and is annotated with semantic classes, which we used for our experiments.

Performance in early tests of PointNet on the ScanNet dataset differed considerably, depending on the amount and types of scene we used. With the vanilla PoinNet implementation as in [QSMG17], satisfying results were only achievable when using very similar scenes like apartments. Therefore, we decided to create 5 different subsets of ScanNet, each containing more scenes than the previous one. The composition of the subsets can be found in Table 4.1.

Set	Categories of scans	Total Number
Set 0	Apartments	40 scenes
Set 1	Apartments + Living room/Lounge	264 scenes
Set 2	Apartments + Living room/Lounge + Bedroom/Hotel	537 scenes
Set 3	Apartments + Living room/Lounge + Bedroom/Hotel + Office	710 scenes
full	all	1513 scenes

Table 4.1.: *Overview of ScanNet subsets*

4.1.1. Preprocessing for segmentation

For the training of our PointNet adoption we only need the colored point cloud with the corresponding semantic labels for each point. ScanNet provides us the colored point cloud as .ply file and also two .json files from which the labels of each point can be inferred.
To save on computation we extract the needed information beforehand and saved it to disk, such that we can use it directly for training and testing. We do this by loading and extracting all the needed information for the scenes belonging to one of the above listed sets. Afterwards we randomly shuffle the scenes and create a training split containing 80% of the scenes as well as a test split containing the remaining 20%. These splits are saved to disk and reused for all the following experiments. Table 4.2 summarizes the amount of scenes, available in each split. Furthermore the label distribution for each class for the train splits is visualized in Figure 4.1. A similar graph for the test splits can be found in Figure A.1. The Appendix also includes Table A.1 which lists the corresponding semantic class for each label id.

Set	Number of training scenes	Number of testing scenes
Set 0	33	7
Set 1	212	52
Set 2	430	107
Set 3	569	141
full	1211	302

Table 4.2.: *Summary of train and test splits*

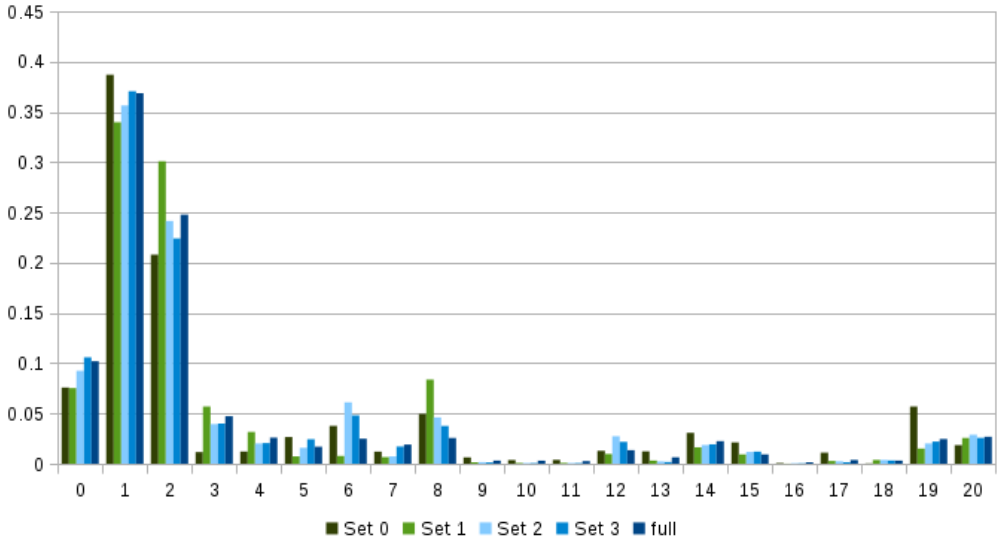


Figure 4.1.: *Label distribution for train splits*

4.1.2. Preprocessing for reconstruction

To generate the groundtruth voxel grid for training the network proposed in [CSO+18] we follow the same steps as them. This includes re-integrating the provided depth maps and semantic segmentation using TSDF fusion [HZC+13] based on the provided camera poses to get a datacost with very strong evidence. With the use of multi-label TV-L1 optimization on that datacost with $W = \nabla$ we finally receive our groundtruth voxel grid for training. Since our final segmentation network was able to achive similar accuracy scores to the previously tested subsets, we just use the full dataset for the reconstruction experiments. In order not to bias the results, we can only segment the 302 testing scenes and convert them into datacosts. As before with segmentation, we use 80% of the scenes for training and 20% for evaluation. This is summarized in Table 4.3:

Set	Number of training scenes	Number of testing scenes
full	242	60

Table 4.3.: *Train and test split for reconstruction*

4.2. Computing Environment

We used *Python 3.6* and *Tensorflow 1.11* for the implementation of our PointNet adoption as well as for the reconstruction part. Other libraries that were intensely used are *PyntCloud, Numpy* and *Pandas*. The code of this project can partly be found on GitHub[1].

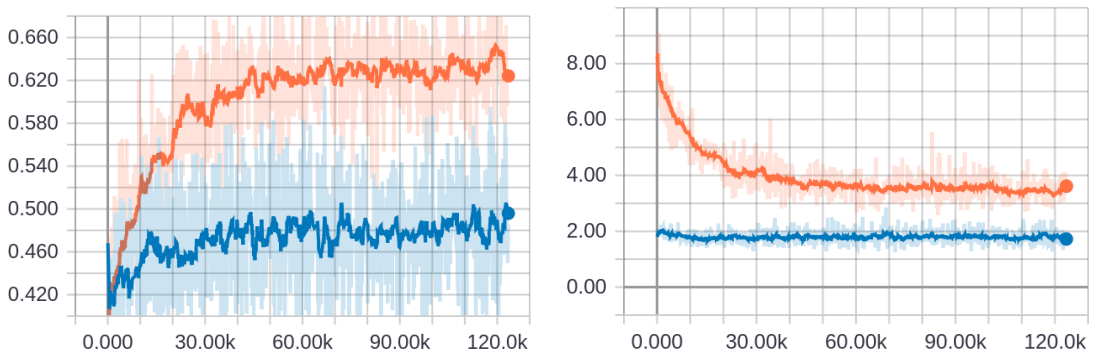
The training as well as the execution of the experiments were done on a workstation provided by *ETH Zurich* with the following hardware details:

CPU: Intel Xeon CPU E5-1650 v3 @ 3.5 GHz
Memory: 64 GB
GPU: Nvidia GTX TITAN X with 12GB memory

4.3. Results and Discussion

In this section we will present the results from the experiments as well as discuss them. We start with the segmentation results in Section 4.3.1 and continue with the reconstruction results in Section 4.3.2.

4.3.1. Segmentation results



(a) Accuracy *(b) Loss*

Figure 4.2.: *Accuracy and loss during training with random rotations orange: train split, blue: test split*

[1] `https://github.com/michaelseeber`

The initial experiments were carried out with our PointNet adoption as described in 3.1.2, with randomly rotating the extracted blocks before feeding them into the network. Figure 4.2 shows the convergence of the accuracy as well as the loss for the full dataset during training. The approximately 60k steps required for convergence corresponded to around 24h of training on the setup mentioned in 4.2.

For each subset we trained the model until convergence and subsequently evaluated the corresponding testing split. The values in Table 4.4 were obtained by evaluating each scene as a whole, in contrast to testing single blocks, as we did during training. Furthermore, each scene was weighted equally for the average.

Sets	Accuracy	Weighted class accuracy	Average class accuracy
Set 0	50.7	45.9	14.9
Set 1	59.7	45.3	11.1
Set 2	56.0	45.8	11.5
Set 3	56.5	47.9	11.1
full	53.0	42.9	9.9

Table 4.4.: *Results for segmentation of whole scenes with random block rotations*

Figure 4.3 visualizes the per class accuracy for the full dataset as well as the amount of test scenes which include the label. Similar graphs for the other subsets can be found in Figure A.2.

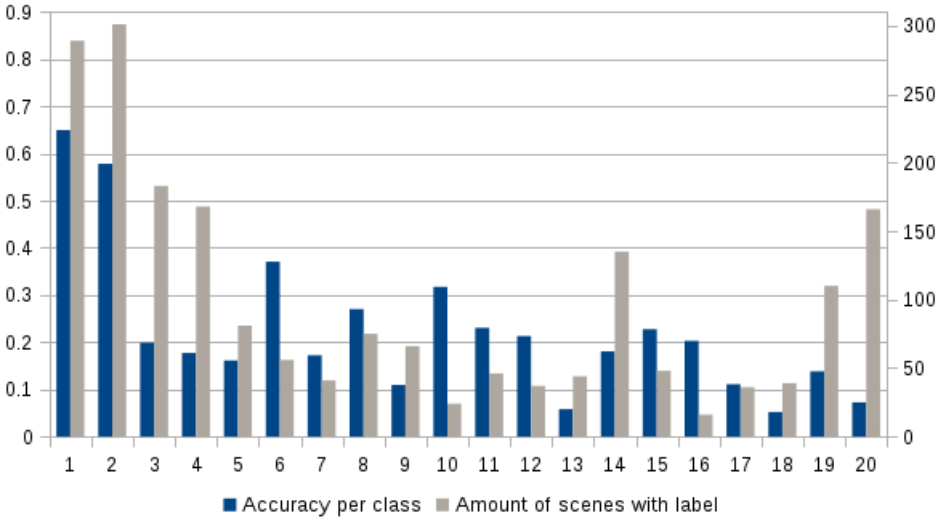


Figure 4.3.: *Per class accuracy for full dataset with random rotations*

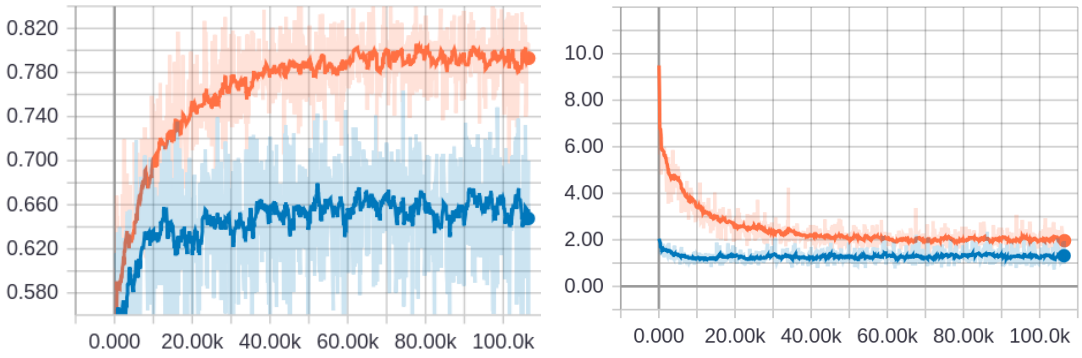
In contrast to our early tests, we were able to achieve comparable performance between the subsets in the end. This is largely due to the new weighted loss function as well as the bigger block size. Hence, all the following experiments were carried out only on the full dataset.

As described in Section 3.1.2 we continued our experiments with limiting the random rotations of the blocks to be only around the z-axis, as this captures every meaningful transformation for our data. That small change boosted the accuracy score by more than 20% when we evaluated our model on the whole scenes, as visible in Table 4.5.

Sets	Accuracy	Weighted class accuracy	Average class accuracy
full	71.4	58.8	16.1

Table 4.5.: *Results with limited rotations on blocks*

Like before, Figure 4.4 shows the accuracy and the loss during training for both splits. Also with this modification the network converges after 60k steps which equals again roughly 24h of training.



(a) Accuracy *(b) Loss*

Figure 4.4.: *Accuracy and loss during training with limited rotations orange: train split, blue: test split*

As before, Figure 4.5 visualizes the accuracy for each single class. This time "floor" stands out as not only being the most prominent class, but also having an accuracy of 94.4%. Additionally, both "wall" and "bed" burst through the 70% per class accuracy barrier. In comparison to Figure 4.3 the general distribution on the labels is still similar.

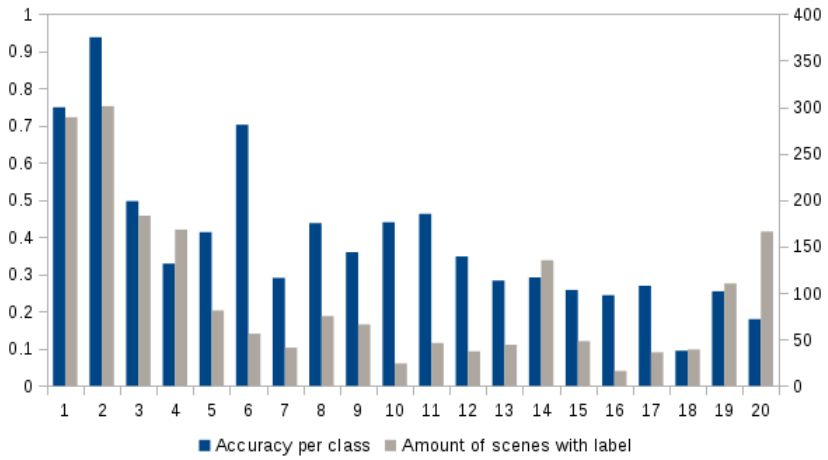
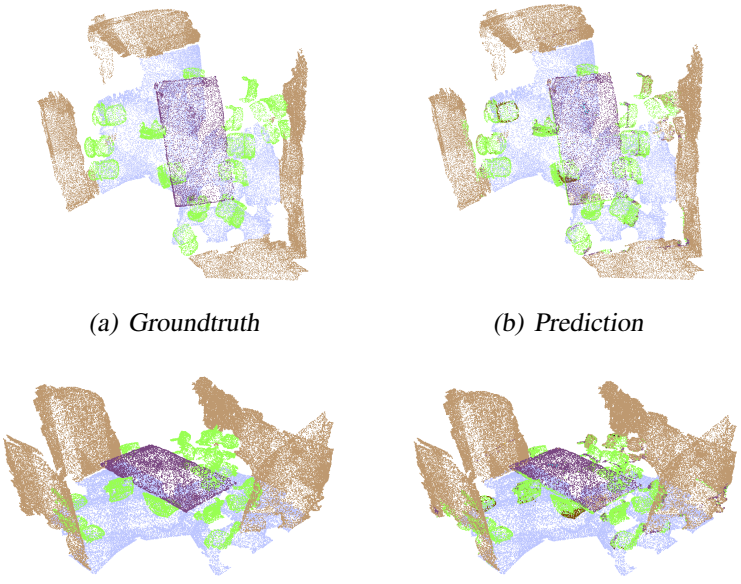


Figure 4.5.: *Per class accuracy for full dataset with rotations only around z-axis*

After inspecting the visualization of various scenes with very high and low accuracy scores we came to the conclusion that our segmentation network is doing really well on smaller scenes that contain few labels. In Figure 4.6 a very good segmentation example can be found. Although this meeting room scene contains many chairs in direct vicinity, our segmentation network is able to correctly predict the class for nearly every point. We assume this is due to the low amount of semantic classes present in the scene as well as the frequent appearance of chairs and tables throughout the ScanNet dataset.



(a) Groundtruth *(b) Prediction*

(c) Groundtruth *(d) Prediction*

Figure 4.6.: *Segmentation example with high accuracy*

In more complex scene the overall semantic segmentation is satisfying, though lacking accuracy in densely backed areas. One such example is the apartment in Figure 4.7, where the representation of the kitchen area in the upper right is messed up.

(a) Groundtruth *(b) Prediction*

Figure 4.7.: *Segmentation of more complex apartment scene*

During our inspections, we also came across very interesting examples like the one in Figure 4.8. The groundtruth wrongly suggests that there is only one chair in the scene. Additionally also the big quadratic table is labeled as "ununnotated". In contrast, our segmentation is not only classifying wall and floor correctly, but also recognizes the other chairs. For the table our network predicts both "table" and "desk", which is a very subtle difference. Presumably the shape priors for these two categories should be very similar, so this is more an issue with the NYU labels than with the segmentation.

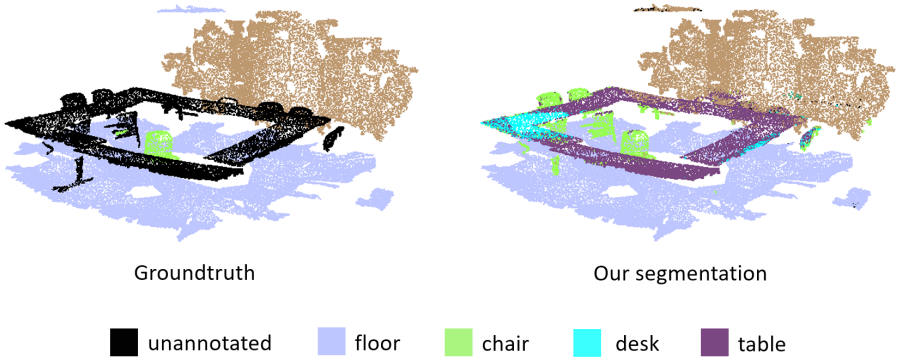


Figure 4.8.: *Our segmentation on the right is more accurate than what the GT suggests*

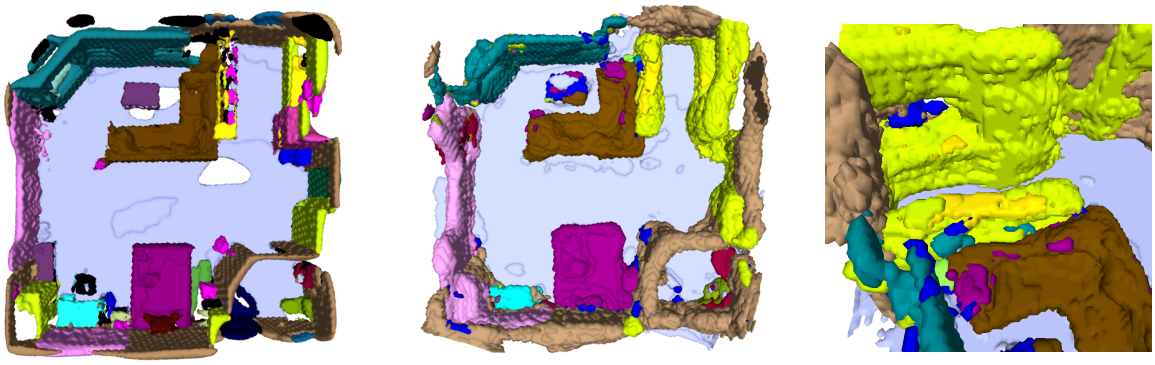
4.3.2. Reconstruction results

We trained several different models for the reconstruction and also tested many different architectures for the intermediate CNN, which creates the datacost for the reconstruction algorithm [CSO+18]. During our experiments, we noticed that there is trade off between the actual accuracy numbers and visually pleasing reconstructions. Models with lower occupied accuracy and higher freespace accuracy resulted in cleaner reconstructions while models with balanced accuracies tend to look overly smooth. Our final reconstruction network with the visually most pleasing results yielded the numbers shown in Table 4.6.

Accuracy	Freespace accuracy	Occupied accuracy	Semantic accuracy
65.2 %	84.4%	64.1%	44.4%

Table 4.6.: *Results of the model with best looking reconstructions*

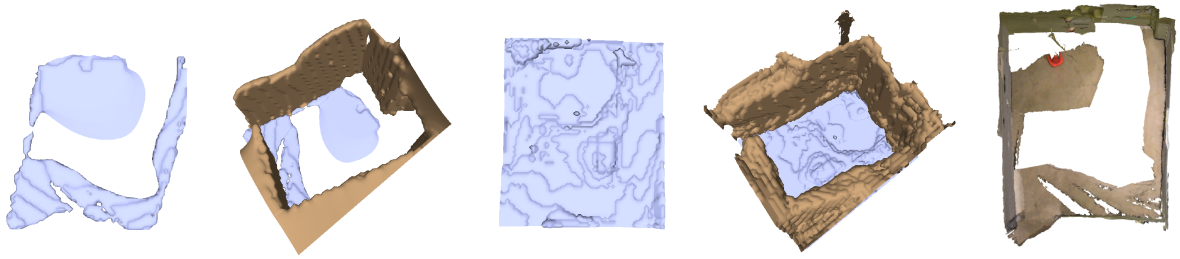
Figure 4.9 visualizes the reconstruction of the relatively complex apartment scene. The prediction is able to represent the underlying geometry in an acceptable manner and even surpasses the groundtruth with its floor reconstruction. Although the segmentation struggled in the kitchen area, the reconstruction network still managed to come up with a reasonable prediction.



(a) Groundtruth *(b) Prediction* *(c) Detail view of kitchen*

Figure 4.9.: *Reconstruction of apartment scene*

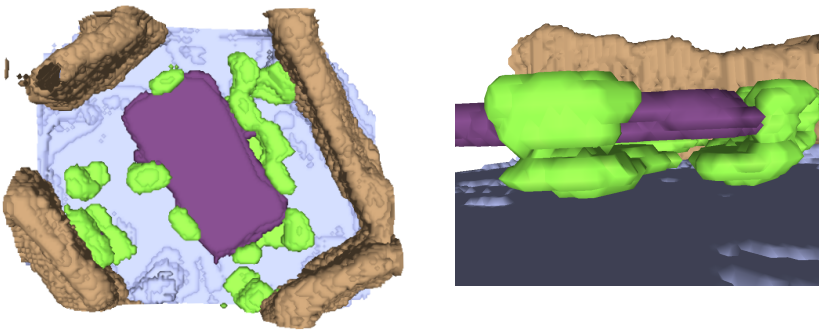
A more drastically example of superior floor reconstruction is visible in Figure 4.10. While the groundtruth resembles more or less the floor from the original scan, our prediction yields a completely closed surface, that is representing the reality more accurately.



(a) Groundtruth *(b) Prediction* *(c) Original scan*

Figure 4.10.: *Example of superior floor reconstruction*

Figure 4.11 shows the reconstruction of the meeting room scene. Although the chairs are overly smooth, the general geometry is very well visible. Moreover, it correctly inferred the free space between the chairs and the table, indicating once again that the network was able to learn reasonable shape priors.



(a) Top view *(b) Detail view of chairs and table*

Figure 4.11.: *Reconstruction of meeting room scene*

As discussed in Section 4.3.1 we obtain inferior segmentation results with completely random block rotations. Furthermore, we also expected a high correlation between reconstruction performance and segmentation accuracy. To verify our assumption, we trained the same network with the "bad" semantic segmentations from Table 4.4. After convergence of the network, we obtained the values shown in Table 4.7 during evaluation.

Accuracy	Freespace accuracy	Occupied accuracy	Semantic accuracy
67.1%	87.7%	54.8%	38.7%

Table 4.7.: *Results when trained and evaluated on "bad" segmentations*

To our surprise the impact on the reconstruction results was lower than anticipated. As illustrated in Figure 4.12, the overall and freespace accuracy are even higher, while the occupied and semantic accuracies lack behind. This implies, that a model trained on "bad" semantic segmentations tends to predict freespace, because this is the predominant class throughout the ScanNet dataset. As a result, the reconstructions look more sparse and incomplete. However, when taking into account that the semantic segmentation accuracy differed by nearly 20%, these results are more than satisfying. Furthermore this proves, that our intermediate CNN is really successful at integrating the information from the segmentation into a meaningful datacost for the reconstruction algorithm.

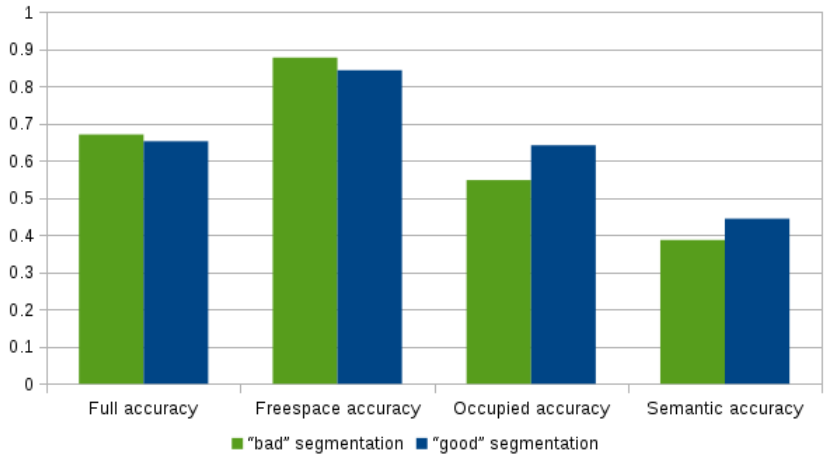


Figure 4.12.: *Impact of segmentation accuracy on reconstruction*

We also tested how well a model, which was trained on good semantic segmentations, can cope with inputs of worse quality. Therefore, we fed the "bad" segmentations (Table 4.4) into our best model, which yielded the numbers from Table 4.6. The obtained results are visualized in Figure 4.13.

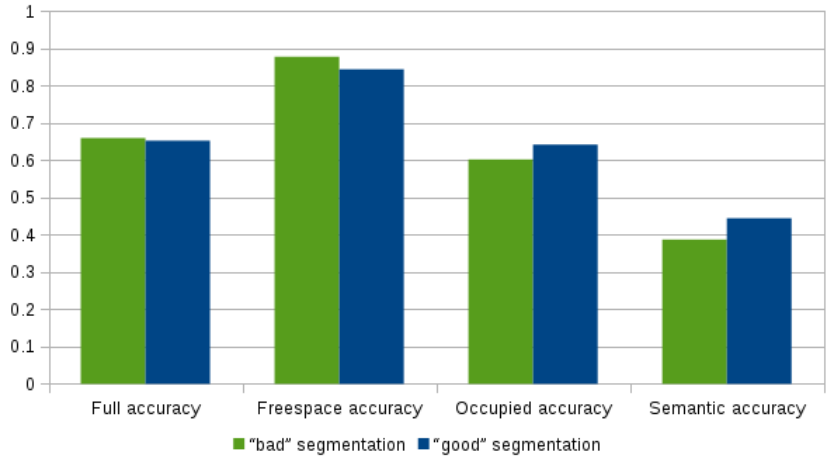


Figure 4.13.: *Ability of our best model to deal with bad segmentations*

While the semantic accuracy is lacking behind, the other metrics only differ by a very small margin. This further strengthens our conclusion that the intermediate CNN is able to abstract the essential information form the voxelgrids very well. Furthermore, it is also able to to correct segmentation errors, by combining the information from adjacent voxels in a sensible way.

5

Conclusion and Outlook

In this thesis we implemented a point cloud segmentation network that is able to correctly infer the semantic class for most of the points. Especially important classes like floor or wall stand out with high accuracy scores. The good segmentation results allowed for the creation of a pipeline that encodes the semantic segmentation into a datacost, from which a state-of-the-art reconstruction network is able to produce satisfying 3D reconstructions. Although the final results tend to be overly smooth in areas with high object density like kitchens and bathrooms, the general scene layout is reconstructed very well. Interestingly, floors were most of the time better reconstructed than what the groundtruth would suggest.

However, not only the reconstruction struggled with the fine grained details but also the semantic point cloud segmentation failed in that area. Because we were able to show that the segmentation performance correlates with the reconstruction performance, although less than previously anticipated, we suggest to further improve the accuracy of the semantic segmentation. A first step in that direction was already made with PointNet++, which processes point clouds on different scales levels and is therefore able to better handle non-uniform scenes, like the ones provided by ScanNet. Another idea would be to segment the scenes in fewer, but more general classes, that share similar shape priors. During our experiments we came across this problem for class "table" and "desk". Furthermore, we were able to show that the intermediate

CNN is able to abstract the input very well, which allows for the correction of semantic prediction errors. Further engineering on this part of the pipeline has therefore also the potential to improve the reconstruction results, as it is able to compensate for not perfectly segmented pointclouds. Additionally, our approach did not got rid of the voxelization step completely, as we only replaced the 2D image segmentation. Performance-wise it would be favorable to not voxelize the scene at all. This should be the long term goal of further work in this direction.

A

Appendix

A.1. Dataset

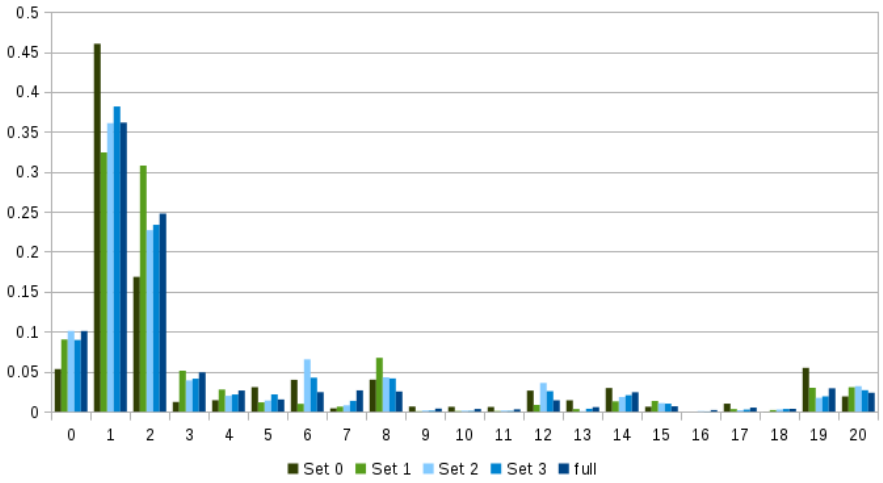
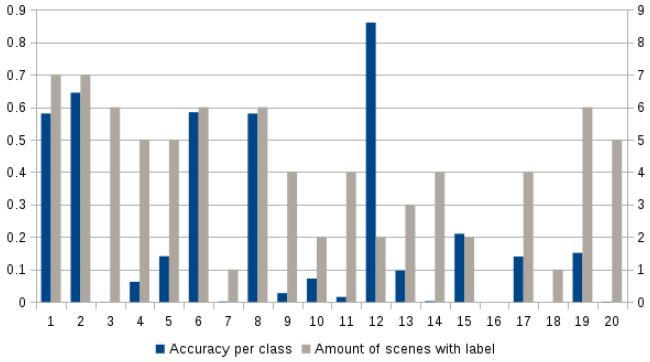


Figure A.1.: *Label distribution for test splits*

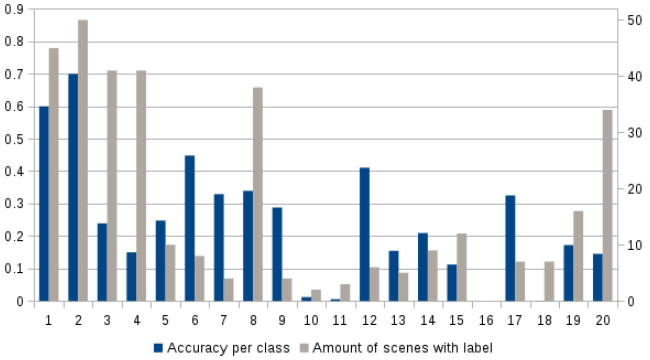
A.2. Segmentation

Table A.1.: *Semantic class labels with distribution across datasets*

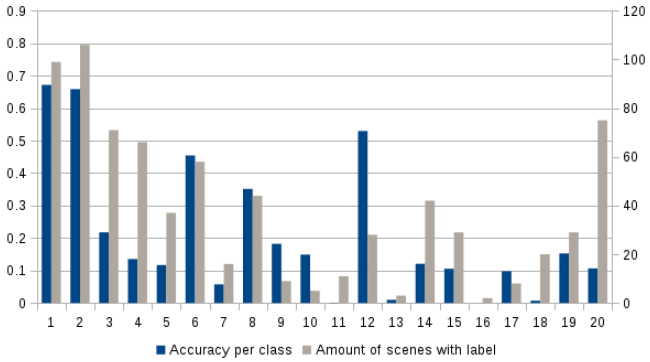
Label id	Semantic class	full	Set 3	Set 2	Set 1	Set 0
0	unannotated	10.20%	10.59%	9.24%	7.53%	7.58%
1	wall	36.87%	37.07%	35.65%	33.98%	38.70%
2	floor	24.79%	22.42%	24.15%	30.09%	20.81%
3	chair	4.73%	4.00%	3.95%	5.70%	1.17%
4	table	2.61%	2.06%	2.02%	3.17%	1.22%
5	desk	1.70%	2.44%	1.59%	0.73%	2.66%
6	bed	2.50%	4.82%	6.11%	0.79%	3.77%
7	bookshelf	1.91%	1.72%	0.74%	0.66%	1.20%
8	sofa	2.57%	3.76%	4.61%	8.38%	4.96%
9	sink	0.32%	0.13%	0.18%	0.15%	0.64%
10	bathtub	0.31%	0.06%	0.07%	0.08%	0.38%
11	toilet	0.27%	0.06%	0.07%	0.09%	0.40%
12	curtain	1.36%	2.18%	2.75%	0.98%	1.30%
13	counter	0.65%	0.18%	0.24%	0.32%	1.24%
14	door	2.26%	1.95%	1.86%	1.64%	3.08%
15	windows	0.93%	1.21%	1.18%	0.93%	2.12%
16	shower curtain	0.16%	0.05%	0.06%	0.02%	0.08%
17	refridgerator	0.38%	0.16%	0.24%	0.28%	1.10%
18	picture	0.32%	0.33%	0.38%	0.39%	0.03%
19	cabinet	2.47%	2.21%	2.03%	1.52%	5.70%
20	other furniture	2.69%	2.58%	2.89%	2.56%	1.86%



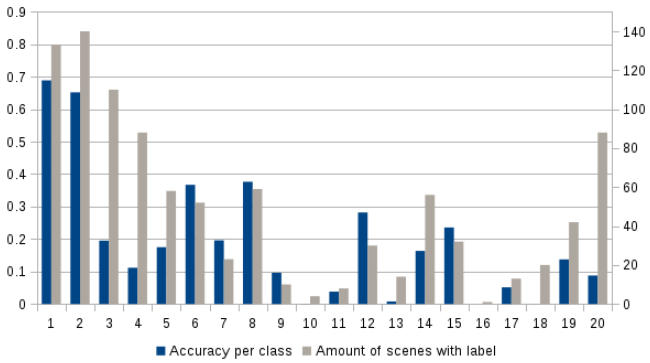
(a) Set 0



(b) Set 1



(c) Set 2



(d) Set 3

Figure A.2.: *Per class accuracy for each subset when trained with random block rotations*

Bibliography

[ASZS17] Iro Armeni, Sasha Sax, Amir R Zamir, and Silvio Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105*, 2017.

[BVR+16] MAROŠ Blahá, CHRISTOPH Vogel, AUDREY Richard, Jan D Wegner, Thomas Pock, and Konrad Schindler. Towards integrated 3d reconstruction and semantic interpretation of urban scenes. *Dreiländertagung der SGPF, DGPF und OVG: Lösungen für eine Welt im Wandel: Vorträge*, 25:44–53, 2016.

[CSO+18] Ian Cherabier, Johannes L Schönberger, Martin R Oswald, Marc Pollefeys, and Andreas Geiger. Learning priors for semantic 3d reconstruction. In *European Conference on Computer Vision (ECCV)*, 2018.

[DCS+17] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas A Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, volume 2, page 10, 2017.

[DRB+17] Angela Dai, Daniel Ritchie, Martin Bokeloh, Scott Reed, Jürgen Sturm, and Matthias Nießner. Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans. In *Proc. Conference on Computer Vision and Pattern*

Recognition (CVPR), 2017.

[FM81] King-Sun Fu and JK Mui. A survey on image segmentation. *Pattern recognition*, 13(1):3–16, 1981.

[GGOEO+17] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, and Jose Garcia-Rodriguez. A review on deep learning techniques applied to semantic segmentation. *arXiv preprint arXiv:1704.06857*, 2017.

[HZC+13] Christian Hane, Christopher Zach, Andrea Cohen, Roland Angst, and Marc Pollefeys. Joint 3d scene reconstruction and class segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[HZCP17] Christian Haene, Christopher Zach, Andrea Cohen, and Marc Pollefeys. Dense semantic 3d reconstruction. *IEEE transactions on pattern analysis and machine intelligence*, 39(9):1730–1743, 2017.

[KKBC09] Kalin Kolev, Maria Klodt, Thomas Brox, and Daniel Cremers. Continuous global optimization in multiview 3d reconstruction. *International Journal of Computer Vision*, 84(1):80–96, 2009.

[KKHP17] Erich Kobler, Teresa Klatzer, Kerstin Hammernik, and Thomas Pock. Variational networks: connecting variational methods and deep learning. In *German Conference on Pattern Recognition*, pages 281–293. Springer, 2017.

[KS00] Kiriakos N Kutulakos and Steven M Seitz. A theory of shape by space carving. *International journal of computer vision*, 38(3):199–218, 2000.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[LBSC18] Yangyan Li, Rui Bu, Mingchao Sun, and Baoquan Chen. Pointcnn. *arXiv preprint arXiv:1801.07791*, 2018.

[LLVT03] Thomas Lewiner, Hélio Lopes, Antônio Wilson Vieira, and Geovan Tavares. Efficient implementation of marching cubes' cases with topological guarantees. *Journal of graphics tools*, 8(2):1–15, 2003.

[NIH+11] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011.

[QSMG17] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017.

[QYSG17] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017.

[RVB+17] Audrey Richard, Christoph Vogel, Maros Blaha, Thomas Pock, and Konrad Schindler. Semantic 3d reconstruction with finite element bases. *arXiv preprint arXiv:1710.01749*, 2017.

[SCD+06] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *null*, pages 519–528. IEEE, 2006.

[SLJ+15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[SMKLM15] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015.

[WSK+15] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Learned 3D Semantic Segmentation using Point Clouds

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Seeber	Michael

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Zurich, 9.10.2018	Michael Seeber

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.