RWTH AACHEN UNIVERSITY

Diese Arbeit wurde vorgelegt am
Lehr- und Forschungsgebiet Informatik 8 (Computer Vision)
Fakultät für Mathematik, Informatik und Naturwissenschaften
Prof. Dr. Bastian Leibe

Master Thesis

3D Instance Semantic Segmentation on Point Clouds

vorgelegt von

Cathrin Elich
Matrikelnummer: 317928
2019-12-11

Erstgutachter: Prof. Dr. Bastian Leibe
Zweitgutachter: Prof. Dr. Leif Kobbelt

Eidesstattliche Versicherung

Cathrin Elich	317928
Name	Matrikelnummer

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Masterarbeit mit dem Titel

3D Instance Semantic Segmentation on Point Clouds

selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Aachen, 2019-12-11	
Ort, Datum	Unterschrift

Belehrung:

§ **156 StGB: Falsche Versicherung an Eides Statt**
Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ **161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt**

(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.

(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten dementsprechend.

Die vorstehende Belehrung habe ich zur Kentnis genommen:

Aachen, 2019-12-11	
Ort, Datum	Unterschrift

Contents

1 Introduction **1**

2 Related Work **3**
2.1 Semantic Segmentation . . . 3
2.1.1 2D Images . . . 3
2.1.2 RGB-D Data . . . 4
2.1.3 3D Data . . . 4
2.2 Object Detection . . . 7
2.3 Instance Segmentation . . . 7
2.3.1 Proposal-Based Methods . . . 7
2.3.2 Proposal-Free Methods . . . 8

3 Preliminaries **11**
3.1 Problem Formulation . . . 11
3.1.1 Semantic Segmentation . . . 12
3.1.2 Instance Segmentation . . . 13
3.2 Semantic Segmentation on 3D Point Clouds . . . 17
3.2.1 PointNet . . . 18
3.2.2 PointNet++ . . . 20
3.2.3 Dynamic Graph CNN (DGCNN) . . . 23
3.3 Similarity Group Proposal Network (SGPN) . . . 26
3.3.1 Network Architecture . . . 27
3.3.2 Group Merging . . . 30
3.3.3 Block Merging . . . 32
3.3.4 Drawbacks . . . 34

4 3D-BEVIS **37**
4.1 Global Instance Features from Bird's-Eye View . . . 39
4.2 Propagation of BEV Features to the Point Cloud . . . 43
4.3 Instance Grouping . . . 45

5 Experiments **49**
5.1 Datasets . . . 49
5.2 Evaluation Metrics . . . 53
5.3 SGPN . . . 56

5.3.1 Implementation Details 56
5.4 3D-BEVIS 60
5.4.1 2D BEV-FN 60
5.4.2 3D P-FN 62
5.4.3 Clustering 63
5.5 Results 64
5.5.1 S3DIS 64
5.5.2 ScanNet 67
5.5.3 Discussion 70

6 Conclusion **75**

Bibliography **77**

1 Introduction

In this thesis, we explore a novel approach for instance semantic segmentation on 3D point clouds by leveraging recent deep learning approaches for point clouds and a novel mechanism to incorporate global context information.

Scene understanding has become one of the most popular research fields in computer vision. Starting from self-driving vehicles [GLU12, NBG+17] over mobile robots [RMB+08] to the usage in medicine [RFB15] and augmented reality [PLW08], this topic offers a broad field of applications.
Depending on the respective demands of each challenge, one normally concentrates on a more specialized subtask of the extensive problem of scene analysis. For instance, much work has been done in the fields of object detection [RHGS15] and semantic segmentation [SLD15] for 2D image data. The task of instance semantic segmentation describes the combination of these [HGDG17, CHP+17]. Here, we aim to segment single objects and infer their category. Compared to plain semantic segmentation, additionally differentiating between instances of the same class makes this problem noticeable more difficult. Reasons for this are the unknown number of objects and the arbitrary indexing order. As a result, a simple transfer of semantic segmentation approaches is not possible.

Since the introduction of convolutional neural networks (*CNNs*), deep learning methods showed dominant performance for the named scene understanding tasks for 2D image data. In comparison to this, the analysis of 3D data has been studied relatively little. Two major reasons can be named for this.
First, contrary to the large range of 2D datasets (e.g. [LMB+14, COR+15]) the amount of publicly available 3D was rather limited for a long time. However, as measuring devices like LIDAR sensors, Microsoft Kinect, or stereo sensors have become more easily accessible lately, the number of suitable large-scale datasets has experienced a strong growth (e.g. [ASZ+16, DCS+17]).
The second reason is correlated with the issue of data representation. Whereas the choice is simple for 2D data to use regular pixel grids as input, there are sev-

eral possibilities to represent 3D data like point clouds [QSMG17a], volumetric grids [MS15] or multiple 2D views [DN18]. Each of these structures has its own advantages and disadvantages. Point clouds are obtained from the raw output of most 3D sensors. It is thus attractive to use them as no prior data formation is required. This also ensures relatively small memory requirements compared to the available information. However, due to the characteristic of point clouds to be unordered, a direct transfer of successful approaches in 2D is not possible as these require highly structured data.
Leveraging 3D data to examine the surrounding environment is nonetheless desirable as this kind of data includes much more information of the real setting compared to 2D images. This presently motivates a lot of research work in this field. However, there is still a lot of potential to improve the performance of current methods.

Our work is mainly motivated by the recently proposed *SGPN* model for 3D instance semantic segmentation on point clouds by Wang et al. [WYHN18]. The main idea of the named approach is to learn a point-wise feature representation that can be used to group points belonging to the same instance. However, due to the proceeding of most current deep learning models on point clouds like the applied *PointNet* [QSMG17a], these features are only valid within a small sub-block of the complete scene. Thus, Wang et al. further present a heuristic merging algorithm to receive a segmentation of the full point cloud.
In this thesis, we aim to learn globally consistent features relevant to instance segmentation. By doing this, we eliminate the necessity of an additional post-merging procedure. As a result of our work, we present a new framework *3D-BEVIS* which combines information from fine-detailed 3D point clouds with the global context received from a bird's-eye view representation. We utilize methods for 2D instance segmentation to learn a suitable feature encoding for the subset of points that were projected into the bird's-eye view. The inferred features are subsequently transferred to the complete point cloud. Following this, we can use any off-the-shelf clustering method to find the single objects within a scene. We evaluate our method on the popular 3D indoor datasets S3DIS [ASZ+16] and ScanNet [DCS+17] and compare our results to those received from SGPN.

The thesis is structured as follows: We first present related work in Chapter 2. In particular, we take a closer look on semantic segmentation methods on 3D data and instance segmentation approaches for 2D images. Afterwards, the problems of semantic and instance segmentation are described in more detail (Chapter 3). Furthermore, we explain current semantic segmentation models for point clouds as well as the SGPN framework. Following this, we present our own model 3D-BEVIS in Chapter 4. We conclude with an experimental evaluation in Chapter 5.

2 Related Work

In this chapter, we will survey previous work related to the topic of this thesis. First, we will have a look on the progress that has been achieved for the related tasks of semantic segmentation and object detection with a major focus on methods for 3D data. Afterwards, different approaches tackling the actual problem of instance segmentation are listed.
The surpassing achievements of deep learning methods in several computer vision tasks lead to a promising outlook for future work using such techniques. Thus, we will not consider traditional methods here but focus on those approaches only.

2.1 Semantic Segmentation

Semantic segmentation represents a task from the field of scene understanding where a class label prediction is made for each pixel or point of the examined scene. Hence, a partition of the scene with respect to the various classes is obtained.
Earlier successes were achieved mainly for 2D images. The growing availability of 3D datasets during the last few years motivated increased research on these more recently.

2.1.1 2D Images

A major breakthrough for semantic segmentation by applying deep learning techniques was achieved using the *Fully Convolutional Network* by Long et al. [SLD15]. They successfully integrate CNNs from existing classification approaches (e.g. VGG [SZ14]) for hierarchical feature learning. Then, the features are upsampled using deconvolutional layers and skip connections into heatmaps for the distinct categories. From these, a dense pixel-wise prediction from inputs of arbitrary size is obtained.
Afterwards, many enhanced deep learning techniques based on this encoder-

decoder architecture were proposed, e.g., SegNet [BKC15] altered the decoder to transfer max-pooling indices from the respective stage of the encoder for upsampling. Improved dense prediction were achieved using dilated convolutions which increase the receptive field without lowering the resolution [YK15,CPK+16, CPSA17]. Another noteworthy proposed network is U-Net [RFB15], which had been presented together with a training strategy based on strong usage of data augmentation to learn from a relatively small dataset.
A survey of several deep learning methods for semantic segmentation was provided by Garcia et al. [GOO+17]. Another overview can be found in [Chi17].

2.1.2 RGB-D Data

Image data with an additional depth map provide further knowledge about the geometric structure of the real-world scene. The question remains how to use this new information.
Qi et al. [QLJ+17] construct a k-nearest neighbor graph based on pixels' coordinates and depth. Pixel features are initially obtained by considering only their RGB values and then iteratively updated from messages of corresponding neighbors from the graph. An alternative approach using adapted CNN operators was proposed by Wang et al. [WN18]. They weight the contribution of neighboring pixels with respect to their depth similarity during both convolution and pooling operations.

2.1.3 3D Data

Working with 3D data proved to be more challenging compared to the handling of planar views. Here, differences between various methods arise regarding the used data representation. While the data is commonly received as a raw point cloud from the sensors, it is often pre-processed into an intermediate representation which allows an easier adaptions of existing algorithms for images.
In the following, we review different options for 3D data representation and corresponding approaches.

Voxelized Volumes

To apply traditional convolutional operators, a highly structured data format is required. For 2D images, a pixel-grid structure is used to satisfy this condition. The equivalent of this in the 3D space are voxelized grids. The conversion is done by carrying out a quantization on the original point cloud.
This proceeding enables an easy transfer of approaches from 2D to 3D data. However, there are several disadvantages using this data format. They mainly result from the sparsity of point clouds. After the transformation, most voxels are not occupied due to the empty spaces in the scene. This results in an inefficient format where a huge amount of memory is wasted, as the empty voxels are still

part of the representation. An important parameter is the resolution of the grid which determines the voxel size and thus the total number of resulting voxels. Therefore, it determines the trade-off between capturing fine details in the scene and a computationally efficient handling of the data structure due to its cubic complexity.

Nevertheless, voxelized volumes have been used repeatedly. Several approaches came up with different ideas to deal with the previous named limitations: Tchapmi et al. [TCA+17] refined their coarse prediction on the subsampled voxel grid to the original resolution on a point-wise level. For this, the 3D convolutions are followed by trilinear interpolation and conditional markov random fields. The scan completion approach by Dai et al. [DRB+18], which also yields a semantic labeling, is likewise based on voxel grids. Instead of an occupancy state, they store a TSDF value in each voxel which encodes the distance to the nearest surface. Moreover, their framework is designed in a multi-resolution hierarchical structure that allows for the processing of large-scale scenes more efficiently. Other approaches benefit from an improved hierarchical structure like octrees [RUG16]. These require less memory with regard to the provided resolution.
Besides the work on semantic segmentation, voxel grids have often been used for object recognition tasks (e.g. [MS15, WSK+15]). As only a single object needs to be represented, the required scale is much smaller compared to a complete scene for our task.

Multi-View Renderings

Another way to profit from CNNs is to project the 3D point cloud data into a set of images depicting the scenes from different perspectives. Resulting pixel-wise features or labels for the single renderings can later be aggregated into a common point. This kind of approach requires less computational costs compared to work on the previous presented volumetric structure. However, this data representation does not consider the underlying 3D geometric structure.

One of the earlier approaches apply Bayesian updates and dense pairwise 3D Conditional Random Fields to transfer semantic labels predicted from 2D images to 3D reconstructed point clouds [HFL14]. This method works directly on RGB-D data received from a sensor. In contrast to this, Boulch et al. [BGLSA17] generate 2D snapshots from a point cloud on which they apply a CNN for semantic segmentation. Hence, inferred labels can be easily back projected as pixel-point correspondences are known. Another promising usage of 2D data has been presented in [DN18]. Here, features extracted from multiple views are combined with geometric features received from a voxel grid within a joint end-to-end framework.

Point Clouds

It is preferable to work on point clouds for several reasons: First, this kind of data format is the raw output of 3D data sensors. Therefore, there is no need to generate an intermediate representation, which might result in loss of details. Nevertheless, there is one major difficulty: As a point cloud presents a set of points and is hence by definition invariant to permutation, this data format is unstructured. Due to this irregular format, reliable convolutional methods for 2D data cannot be applied directly.

A pioneering deep learning approach for several scene understanding tasks was proposed by Qi et al. [QSMG17a]. The presented *PointNet* learns a symmetric function over all points in the cloud to achieve input order invariance. For this, a local feature is computed for each point independently. Afterwards, a common global signature is received from a simple pooling function to encode global context. The concatenation of local and global features is then used to predict point-wise labels.
A downside of this approach is that no local structure of nearby points is taken into account when determining a point's feature. Later methods were built up on the general idea and aimed to improve it by modeling local dependencies. In [QYSG17a], a hierarchical network architecture is presented, which uses PointNet recursively on nested partitions. Similarly, feature learning and aggregation in [ZG17] is also done hierarchically by applying a k-d structure on the points. Two alternative mechanisms to get larger-scale spatial context are proposed in [EKHL17]. Here, local context is obtained by considering either point blocks of different scale or neighboring blocks simultaneously.
Alternatively, several operators were presented which aimed to be equivalent to convolutions and were usable on point clouds. The approach in [HTY18] applies a kernel function over neighboring points. In [LBSC18], an additional transformation is discussed that guarantees invariance with respect to the neighboring points.Wang et al. [WSL$^+$18] introduce a novel *EdgeConv* operator, which first builds up a k-nearest neighbor graph based on the points' features and then learns edge features. From these, the point features are obtained.
All the previous mentioned approaches require the point cloud to be split up in smaller blocks due to memory limitations. These blocks are processed independently and are subsequently merged into a common output point cloud.

A number of other approaches works on the complete point cloud. Landrieu et al. [LS18] determine a set of superpoints based on handcrafted features which form geometrical homogenous elements. From these, a graph is generated and a contextual segmentation is received via graph convolutions. Another method transforms the point cloud into an ordered sequence of features corresponding to slices [HWN18]. Recurrent neural networks are then applied on these sequences. In [TPKZ18], convolutions are performed on virtual tangent planes.

2.2 Object Detection

When performing object detection, we aim to predict oriented bounding boxes around all objects existing in the scene. Furthermore, the found objects are categorized.
As we are mainly interested in working on 3D data, we will concentrate on such methods in the following.

So far, most approaches relied on either multiple 2D views or a discretized voxel representation of the scene.
The voxelized format is for example used in [Li16], where a FCN for object detection is extended to 3D, and more recently in [ZT18]. In the latter, a voxel is not described by statistical values like the mean over all points it contains, but results from learned features of these points.
In contrast, using 2D image representations allows the application of 2D CNN object detectors as done in [QLW+17]. Here, the resulting 2D proposals are subsequently lifted into 3D. The inferred amodal bounding box encloses the entire object even though only parts of it are visible in the image. Frequently, the used views are limited to a single bird's-eye view (BEV) rendering [SMAG18, YLU18]. Their common aim is to achieve real-time efficiency to make the models usable for autonomous driving. The framework in [LYU18] additionally deals with tracking and motion forecasting. In [CMW+16], object proposals received from the BEV are combined with features learned on multiple RGB images.

2.3 Instance Segmentation

Instance segmentation can be seen as the intersection of semantic segmentation and object detection. Here, the task is to determine a labeling which differentiates between the single objects of the same category on a pixel- or point-level.
So far, approaches have been merely performed on 2D images. In general, there are two main basic concepts to tackle the problem. *Proposal-based* methods look for interesting regions first and then segment the main object in the detected proposal. Alternatively, *proposal-free* approaches learn a feature embedding space for the pixels within the image. The pixels are subsequently grouped according to their feature vector. In the following, we will present models from both.

2.3.1 Proposal-Based Methods

This type of method first follows the idea of object detection. This means, a set of region proposals in form of bounding boxes is predicted which are likely to contain objects. Afterwards, the primary found object is classified and segmented

to split the foreground object from the background.

Several approaches focused on finding appropriate instance candidates [PCD15, DHL+16]. One of the first successful end-to-end frameworks for the whole task of instance segmentation was proposed in [DHS16]. The proposed network has a cascaded structure build up of branches that correspond to the sub-tasks of detecting instances, segmenting a mask, and classifying the object. The branches depend on each other regarding both their input and output as well as their respective losses. In [HGDG17], the efficient Faster-RCNN network for object proposal detection [RHGS15] is extended by an additional branch. Thus, not only a class label is predicted as in the original network, but also an segmentation mask. Similarly, the approach in [CHP+17] also relies on Faster-RCNN. In addition to this, they learn semantic and direction predictions which they combine to obtain a segmentation of the proposed window. More recently, the propagation of features along the different levels was proposed in [LQQ+18].

Although many state-of-the-art methods for instance segmentation follow this procedure, there are some drawbacks which gain in significance for instance segmentation on 3D data. As segmentation is only performed on the detected proposal regions, this type of method heavily relies on getting good proposals. This does not only mean that for each object a proposal needs to be found, but also, that the received bounding boxes are a good approximation of the respective shapes. This works well for compact objects but leads to difficulties for elongated or entangled instances. However, in 3D space, the latter case becomes more likely, which leads to an increased probability that bounding boxes of two different objects overlap. This also means that more than one object is present in the proposal box, which possibly affects the following segmentation process. Because of this difficulty, we decided against a proposal-based approach for our work.

2.3.2 Proposal-Free Methods

An alternative to the previous instance segmentation approaches are proposal-free methods. The idea is to learn an embedding for all the pixels which can be used directly for instance segmentation. For most recent approaches, the learned feature vectors are supposed to be similar for pixels belonging to the same object but lie far apart in the feature space for a pair of pixels from different instances. Thus, in a post-processing step, pixels can be clustered with respect to their feature representation into instance groups.

First approaches that followed the idea of learning instance-relevant features aimed to learn specific information like the location of the object to which a pixel belongs [LWS+15] or the direction towards the center of the object [UCFB16].

Later on, features were more commonly learned based on a similarity measure between pairs of pixels by applying a semantic segmentation network. One example for such a similiarity measure is the discriminative loss function proposed in [BNG17]. This loss draws all feature vectors of points from the same instance close together in the embedding space but pushes the means according to the individual instances away from each other. Newell et al. [ND16] use a one-dimensional feature to distinguish between objects within a predicted segmented mask corresponding to all objects of a common class. In [FWR+17], additional seed points are learned. Starting from these, groups are build up by associating the remaining pixels to one of the seeds based on their similarity in the feature space. Whereas previous methods based their similarity metric mainly on the L2-norm, another option is the cosine distance which results in hyper-spherical embedding space [KF17, PSN+18]. One of the advantages of this function is that it forces the model to learn feature vectors with different orientations instead of focusing only on the magnitude. In [KF17], a differentiable mean shift operation is repeatedly applied to help the model to focus on aspects which are later hard to be segmented in the post-processing clustering. Payer et al. [PSN+18] integrate a recurrent network into their segmentation architecture to not only apply instance segmentation but also tracking over time. Instead of learning features, Hsu et al. [HXKH18] aim to directly predict an instance index for each pixel. They deal with the problem of an unknown number of objects by considering a fixed number and regarding the assignment task as a graph coloring problem. Thus, only nearby instances are required to be labeled differently.

Recently, a first framework was proposed for instance segmentation on 3D point clouds [WYHN18]. The introduced framework *SGPN* learns point-wise features according to a similarity metric. From this, a proposal is received for each point. These proposals are afterwards merged into final objects following a heuristic procedure. More specifically, following the idea of non-maximum suppression, two proposals are merged if their overlap exceeds some threshold.

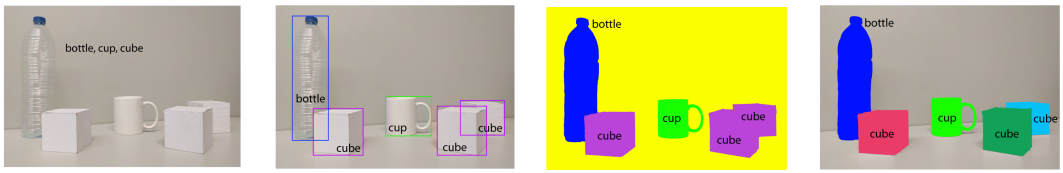
3 Preliminaries

In this chapter, background knowledge about the general problem formulation is stated and a selection of specific deep learning approaches for segmentation on point clouds is presented. We will start by defining the problems of semantic segmentation as well as instance segmentation (Section 3.1). Following this, we will present several successful deep learning frameworks for semantic segmentation on 3D point cloud data in Section 3.2. Afterwards, a recently proposed framework for instance segmentation in 3D space is explained (Section 3.3). The last named approach will be used as the baseline for this thesis.

3.1 Problem Formulation

The general challenge of scene understanding can be subdivided into more distinct tasks (Figure 3.1). Classification predicts the category of an object. This can be extended to a list of inferred labels in the case of several present objects. As a next stage, when performing *Object Detection*, we not only want to name all seen objects but also locate them within the scene by predicting centroids or surrounding bounding boxes. *Semantic Segmentation* yields an even denser prediction: For each point of a scene, a class label is predicted to describes the object's category to which the point belongs. For the task of *Instance Segmentation*, a point-wise labeling is performed to assign it to both a class and an instance. A further variant of semantic segmentation is *Part-Based Segmentation* where a single object is split up into separate components.

In this section, we will review the scene analysis tasks semantic segmentation and instance segmentation. The problem description considers 2D data but can easily be transferred to a higher dimensional space.



(a) Classification (b) Object Detection (c) Semantic Segmentation (d) Instance Segmentation

Figure 3.1: **Different vision tasks for scene understanding.** Figure extracted from [GOO+17].

3.1.1 Semantic Segmentation

Semantic segmentation is the high-level task of partitioning an image into meaningful, distinct regions that correspond to a certain category. For this, a class label is assigned to every pixel within the picture. Formally, given an image with $W \times H = N$ pixels, each pixel $p_i, i = 1, ..., N$, is assigned a label ℓ_i from $\mathcal{L}_{sem} = \{1, ..., K\}$. Each label corresponds to one of the $K \in \mathbb{N}$ classes.

The common way of deep learning approaches for semantic segmentation is the estimation of pixel-wise feature vectors. These encodings are used to predict a probability distribution over all classes. This is typically realized with an encoder-decoder architecture [SLD15].
The standart loss function for the semantic segmentation task is the pixel-wise multi-class cross-entropy loss

$$L(\hat{y}, y) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{K} y_{i,c} \log \hat{y}_{i,c} \tag{3.1}$$

for predicted class scores $\hat{y}_i$ with $\sum_{c=0}^{K} \hat{y}_{i,c} = 1$ and k-dimensional one-hot encoded ground truth labels y_i for a point p_i. Additionally, weights w_c can be applied with respect to every class. This helps to balance their contributions to the loss in case of having overrepresented categories.
By the design of this loss function, a pixel's embedding feature is pushed towards the unit vector corresponding to its class. Thus, for the final result, the predicted class label ℓ_i is determined as

$$\ell_i = \underset{c}{\operatorname{argmax}}\, \hat{y}_{i,c}. \tag{3.2}$$

For an overview about several existing deep learning methods we refer the reader to Section 2.1.

3.1.2 Instance Segmentation

Instance segmentation expands the pixel labeling problem of semantic segmentation. Besides distinguishing between classes, we also aim to identify single objects belonging to the same category and predict a seperate segmentation mask for each of these instances.
There are several aspects of this problem which require special consideration. First, the maximum number of instances within a scene is unknown and might vary a lot within a dataset. Even more, the number of images containing a high number of instances is expected to be rather small. Thus, higher indices have only a small number of corresponding training data. Second, the order of the instance labels can be permuted arbitrary. Therefore, there are no fixed target indices to associate the pixels with specific objects. Instead, the relationship between pixel is known, i.e., whether two pixels belong to the same instance.

Given an image, we let C denote the number of instance clusters. Furthermore, for each instance $c = 1, ..., C$, N_c is the number of pixels belonging to the object group c.
Similar to the problem formulation for semantic segmentation, a label ℓ_i from $\mathcal{L}_{inst} = \{1, ..., C\}$ is selected for each pixel $p_i,\ i = 1, ..., N$ such that

$$\begin{aligned} \ell_i &= \ell_j \qquad \text{if } p_i, p_j \text{ belong to the same object,} \\ \ell_i &\neq \ell_j \qquad \text{if } p_i, p_j \text{ belong to different objects.} \end{aligned} \tag{3.3}$$

Especially, swapping the indices of all points from any two instance groups still yields a valid solution. Thus, the predicted instance labels do not need to be the same as stated in the ground truth. Instead, they only have to satisfy the relationship stated in Equation 3.3.
For this reason, it is not possible to simply transfer basic semantic segmentation methods which aim to learn fixed categories from a known limited set of possible classes.

Another aspect that needs to be considered is that computing the loss for all pairs of pixels would be computationally infeasible. A common way to handle this problem is to consider only a subset of points $S = \bigcup_{c=1}^{C} S_c$. A set S_c contains M points that are randomly sampled from an instance c. The loss function is then evaluated only on pixel pairs from this subset.

Among the different categories of instance segmentation approaches presented in Section 2.3, in this thesis, we will focus on the idea of proposal-free methods. They rely on learning a representation x_i for each pixel and perform a clustering of the pixels with respect to a feature embedding space.
In the following part, we will present different options for objective functions to learn an appropriate embedding. Afterwards, we will outline the clustering method Mean Shift for the final assignment of instance labels.

Loss Functions

The main idea for this problem is to learn an embedding space such that pixels from the same instance are embedded closely together whereas the respective feature vectors from pixels belonging to different instances are far apart. This leads to the general loss function:

$$L = L_{var} + L_{dist} + L_{reg}. \tag{3.4}$$

The *variance* loss L_{var} pulls embeddings from pixels assigned to the same group together. In contrast, embeddings from pixels with different instance labels are pushed apart by the *distance* loss L_{dist}. A regularization term L_{reg} can optionally be added to prevent the model from overfitting.

The named loss terms depend on the pairwise distance between the embedded points. There are different possible metrics to evaluate the similarity $s_{i,j}$ of features x_i, x_j corresponding to two pixels.
Most commonly, the L2 norm is applied (e.g. [ND16,FWR$^+$17,WYHN18]). Straightforward, this results in

$$s_{i,j} = \|x_i - x_j\|_2 \tag{3.5}$$

This can be combined with the loss terms

$$L_{var} = \sum_{c=1}^{C} \sum_{x_i, x_j \in S_c} s_{i,j}, \qquad L_{dist} = \sum_{\substack{c,c'=1 \\ c \neq c'}}^{C} \sum_{\substack{x_i \in S_c \\ x_j \in S_{c'}}} [\delta_{dist} - s_{i,j}]_+ \tag{3.6}$$

for some constant margin δ_{dist}. $[x]_+$ denotes the hinge $\max(0, x)$.

In [FWR$^+$17], Fathi et al. modify the similarity measure by using an exponential function:

$$s_{i,j} = \frac{2}{1 + \exp(\|x_i - x_j\|_2^2)} \tag{3.7}$$

The resulting similarity values $s_{i,j}$ are within the range $[0, 1]$ with scores close to 1 indicating similar features. Thus, the loss terms can be changed to

$$L_{var} = -\sum_{c=1}^{C} \sum_{x_i, x_j \in S_c} \log(s_{i,j}), \qquad L_{dist} = -\sum_{\substack{c,c'=1 \\ c \neq c'}}^{C} \sum_{\substack{x_i \in S_c \\ x_j \in S_{c'}}} \log(1 - s_{i,j}). \tag{3.8}$$

Notably, these do not depend on the choice of the hyper-parameter δ_{dist} any more.

An alternative to the Euclidean distance is the cosine similarity as used by Kong et al. [KF17]:

$$s_{i,j} = \frac{1}{2}\left(1 + \frac{x_i^T x_j}{\|x_i\|_2 \|x_j\|_2}\right) \tag{3.9}$$

Here, the distance between two points correlates to the angle between their vectors. Due to scaling and the addition of an offset, this again yields values between 0 and 1. They use this metric for

$$L_{var} = -\sum_{c=1}^{C} \sum_{x_i, x_j \in S_c} 1 - s_{i,j}, \qquad L_{dist} = -\sum_{\substack{c,c'=1 \\ c \neq c'}}^{C} \sum_{\substack{x_i \in S_c \\ x_j \in S_{c'}}} [s_{i,j} - \alpha]_+. \tag{3.10}$$

The hyper-parameter α controls the minimum angular distance between points belonging to different groups. Kong et al. argue in favor for the cosine measure that it is invariant to the scale of the feature vectors and, hence, make the usage of the regularization loss insignificant.

Furthermore, it is possible to add weights $w_{i,j}$ depending on the pixels' instance combination. For example, instance pairs that get assigned different class labels can be weighted differently compared to objects belonging to the same category [WYHN18].

De Brabandere et al. propose another variation of the stated loss terms [BNG17]. Instead of performing pairwise comparison, they consider the current mean μ_c of all features associated with an instance c. For the variance loss, they penalize all features that are not within a margin δ_{var} of the corresponding instance mean. To distinguish between instances, means of distinct groups are pushed away from each other up to a certain distance δ_{dist}. Formally, this yields

$$\begin{aligned} L_{var} &= \frac{1}{C} \sum_{c=1}^{C} \frac{1}{N_c} \sum_{i=1}^{N_c} [\|\mu_c - x_i\| - \delta_{var}]_+^2, \\ L_{dist} &= \frac{1}{C(C-1)} \sum_{\substack{c_A, c_B = 1 \\ c_A \neq c_B}}^{C} [\delta_{dist} - \|\mu_{c_A} - \mu_{c_B}\|]_+^2, \\ L_{reg} &= \frac{1}{C} \sum_{c=1}^{C} \|\mu_c\|. \end{aligned} \tag{3.11}$$

Contrary to the previous functions, all points are considered instead of only a subset.

Clustering Methods

Having learned an embedding, the subsequent step involves a clustering algorithm to group the pixels regarding their feature representation.
There are some aspects that restrain the choice for a suitable algorithm: Firstly, the number of instances is unknown and varies for different images. Therefore, it is preferable to not depend on a hyper-parameter which specifies the number

of clusters. This makes several methods like k-Means unfavorable for this task. Secondly, the algorithm needs to be scalable as we have a high number of pixels. Lastly, the size of the clusters might vary making it necessary to detect point groups of different scales. This is important in case that the objects in the scene significantly differ in their size.

One possible clustering algorithm that meets the demands is *Mean Shift* [CMM02]. Thus, this technique has alreaday been utilized for several instance segmentation proceedings (e.g. [BNG17, KF17]). We will describe the basic operating principle in the following.
For a set of points embedded in some feature space, Mean Shift looks for positions with a high density of points. This is done in an iterative process where a current location is updated according to the points being situated in a surrounding area. The found positions correspond to cluster centers. Each point is eventually associated to one of these centroids.

Starting with some randomly sampled point x_i^0, its position is step-wise shifted towards the direction of increasing density of nearby present points. For an iteration step t, the neighborhood $N(x_i^t)$ of current candidate position x_i^t is regarded to determine the shift. In general, the neighborhood $N(x)$ is composed of points within a certain distance from x. This distance is set by the *bandwidth* parameter of the algorithm. The direction towards the area with high density within this neighborhood is computed by the mean shift vector

$$m(x) = \frac{\sum_{x_j \in N(x)} \mathcal{K}(x_j - x) x_j}{\sum_{x_j \in N(x)} \mathcal{K}(x_j - x)} \tag{3.12}$$

where $\mathcal{K}(\cdot)$ denotes some kernel function.
The updating step of x_j results from

$$x_i^{t+1} = x_i^t + m(x_i^t) \tag{3.13}$$

where the new position x_i^{t+1} is equivalent to the mean of the points within the neighborhood $N(x_i^t)$. The iteration stops when x_i^t converges at some location.

There are different possibilities to assign the points to the single clusters. Ideally, the iteration process is run for every point individually. Each point is then assigned to the outcome of the procedure. The resulting set is subsequently pruned for local maximums. However, this is not reasonable for a large set of points. Considering N points and K iteration steps, the resulting runtime complexity is $\mathcal{O}(KN^2)$. A faster version is to either assign points that occur in some window along the way of an iteration run to the resulting cluster center or assign each point in a final processing step to the respective closest centroid.

In conclusion, the Mean Shift algorithm proposes a useful grouping algorithm for finding clusters within an embedding space. It only depends on the bandwidth parameter that describes the window size according to which the mean shift step is computed. Therefore, this clustering method is suitable for the task of instance segmentation based on point features.

3.2 Semantic Segmentation on 3D Point Clouds

The recent increment of publicly available 3D datasets (e.g. [ASZ+16, DCS+17]) has increased the demand to perform scene analysis tasks in the spatial space as well. In contrast to the treatment of 2D images, the best way of structuring the data is not clear here. In Section 2.1.3, different options for 3D data representation together with their respective advantages and drawbacks were named. In this thesis, we will focus on approaches that work on raw point clouds directly. Recent achievements for this kind of data promise capabilities for further work (e.g. [QSMG17a, QYSG17a, WSL+18]).
Working directly on a point set received from a sensor is preferable compared to converting the data into some other format as volumetric grids or multiple views. By using the original data, we neither rapidly increase the complexity nor loose details about the geometrical structure.

By definition, a point cloud $P = \{p_i \mid i = 1, ..., N\} \in \mathbb{R}^F$ is an unordered set of N points. Each point p_i is given by a tuple of F features consisting of the point's coordinates (x_i, y_i, z_i) and optional additional information like RGB color values. A network that receives such a point cloud as input needs to be invariant to any permutation of the points.
However, most of previous successful deep learning approaches for 2D image tasks like convolutional networks require structured data as input like, for example, pixel grids. Thus, the question how point clouds could be handled for neural networks has been unexplored until recently.
One novel approach to deal with this problem was presented by Qi et al. [QSMG17a]. In contrast to conventional methods, it does not rely on any ordering of the points as it first treats all points independently before aggregating them all by applying a symmetric function. With their method, they achieved state-of-the-art results and motivated others to use and expand their idea (e.g. [QYSG17a, WSL+18]).

In the following part, we will first explain the structure of PointNet. Afterwards, we will present the two extensions PointNet++ and Dynamic Graph CNN (DGCNN) that aim to deal with the drawbacks of PointNet and achieved improved performance.

3.2.1 PointNet

Recently, Qi et al. presented a new neural network architecture they named *PointNet* [QSMG17a]. Their pioneering framework is able to work directly with an unordered set of data points. The authors applied their proposed network for several 3D analysis tasks like object recognition, part segmentation of an object and semantic segmentation of a scene. In the following, we will focus on the network model that was applied for semantic segmentation.
The key idea is to learn a high dimensional representation for each point. For this, each point is considered independently. The resulting features are subsequently aggregated into a common global feature. The class label for a point is afterwards determined from the combination of its individual representation and the global feature.

The network architecture is depicted in Figure 3.2. As input, PointNet receives a list of N points and corresponding F_{in} features. The order of the points in this (N, F_{in}) matrix is arbitrary. The output of the network is a (N, K) matrix with K denoting the number of semantic classes. Each entry of this matrix denotes a score that indicates how likely a point belongs to a certain class.
The whole pipeline can be subdivided into two parts:
In the first part, the feature encodings of the distinct points are learned by a multilayer perceptron (MLP). The weights of these layers are shared along all points. Starting with a relatively low number of input features, the feature dimension of a point is highly increased by this.
They also propose in their paper to add joint alignment networks ($T\text{-}net$). The idea is to predict a transformation matrix that aligns the input points' coordinates as well as their predicted features into a canonical space. Doing this ensures that the learned features are invariant to any affine transformation applied to the input point cloud. However, this extension is only used for the point cloud classification and the part segmentation task as it did not show improvements for semantic segmentation. Hence, we will ignore it from now on.
In the second part, the individual point features are aggregated into a global feature. This global feature takes some kind of interaction between all points into account and hence, holds information about the whole input set. The received input points are subject to a random permutation. Any possible aggregation function therefore is required to be invariant to the input order. One option is the usage of a simple symmetric function. Examples for this are the operators $+$, $*$, and max. The authors argue that by combining such a symmetric function with the output of a MLP, they are technically able to learn any potentially more complicated symmetric function regarding the original input point set. In their proposed network, they use a max pooling function.
The resulting global feature is concatenated to each of the previous point features. Another MLP predicts the final classification score for each point.

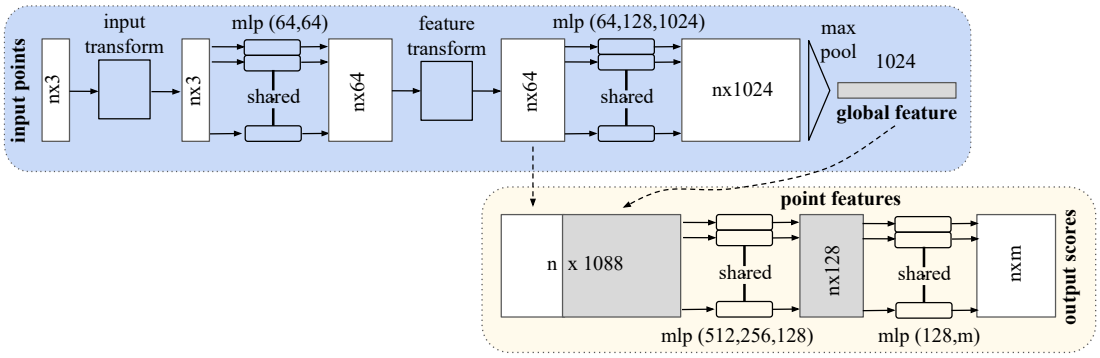


Figure 3.2: **PointNet architecture.** Given an input set of points, feature representations are determined independently for each point using a MLP. These are subsequently aggregated into a global feature. After concatenating the global feature to each point's feature representation, class labels are predicted for the point set using another MLP with shared weight along all points. Figure adapted from [QSMG17a].

A limitation of this approach is the high memory requirement resulting from the high dimensional features computed for every point. Thus, it is often not possible to process the whole point cloud at once. To deal with this, the full scene is subdivided into blocks which are processed separately. The individual results are subsequently merged to receive a segmentation for the whole scene. A fixed number N of points is sampled from each block.

For their experiments on the S3DIS dataset [ASZ+16], Qi et al. split each room into blocks with an area $(1m \times 1m)$ along the ground. Each point is described by a tuple $(x, y, z, r, g, b, \hat{x}, \hat{y}, \hat{z})$. $x,\ y,\ z$ represent the point's coordinates, $r,\ g,\ b$ its color and $\hat{x},\ \hat{y},\ \hat{z}$ the normalized position of the point with respect to the room size.

A fundamental problem of PointNet, however, is the weakness in processing local geometric structure. For the main part, each point is handled separately without having any local context at all. The only time the interaction among the points is considered, is the application of the aggregation function. That means, we receive only global context information regarding all points within an input block. This prevents us from gaining any knowledge about both the geometric structure of the whole scene and finer details indicated by nearby points.

3.2.2 PointNet++

Following the success PointNet had achieved for deep learning on unstructured point sets, several methods were proposed that build on top of this idea. Especially the aspect of exploiting local structure at different scales has been shown to be essential [EKSL18].

One improved network was proposed by Qi et al. themselves [QYSG17a]. Their presented *PointNet++* uses a hierarchical structure to learn features recursively at different scales from a nested partitioning of the input point set.
Starting with small regions, low-level features are extracted from these. These features hold information about fine geometric structures. Following this, several regions and their corresponding features are grouped together to form higher level features. By doing this, the receptive field is enlarged to handle more complex structures.

Qi et al. [QYSG17a] propose the concept of a local feature learner. The purpose is to both determine features for the small neighborhoods and aggregate low-level features to higher ones when grouping regions. PointNet was chosen for this task.
Another main contribution of their work is the generation of an appropriate partition of a point set. The demand for such a partition is to have overlapping regions that distribute the point set evenly. Each region can be defined by their centroid and scale. The associated points are chosen with respect to the underlying Euclidean metric space of the point set.

The hierarchical structure of PointNet++ is depicted in Figure 3.3. It can be split up into two phases. In the first phase, point set features are learned for varying hierarchical levels. In the second stage, the features from before are upsampled respective to the hierarchical levels to provide per-pixel scores for semantic segmentation.

The first phase consists of a series of *set abstraction* levels. Each such level receives a set of points with corresponding feature vectors as input. Then, it determines a subsampled set of points with new features corresponding to the next higher scale.
For this, a subset of points is sampled at first and specified as the centroids for the new regions (*Sampling Layer*). Next, the remaining points are grouped with respect to these centroids (*Grouping Layer*). Finally, a lighter version of PointNet computes the feature encoding for each region (*PointNet Layer*). The single phases of the abstraction layer are explained in more detail in the following:
The input for the Sampling Layer is a $(N, (d + C))$ matrix with N points and corresponding features. A point feature consist of the d-dimensional coordinates and further C feature channel. For the first abstraction layer, these are the points

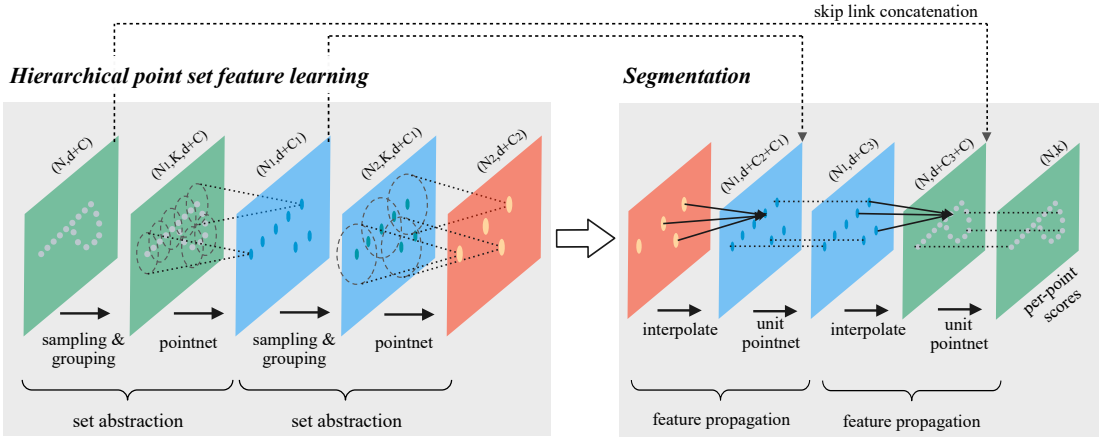


Figure 3.3: **PointNet++ architecture.** The pipeline can be split into two parts: In the phase for *hierarchical feature learning*, high dimensional point representations are learned by a sequence of set abstractions. Each set abstraction level first samples a subset of points from its input as centroids and then groups the remaining points towards these centroids. Subsequently, PointNet is applied to each group to determine a feature for the whole group. The group features and the centroids are then passed to the next level. In the *segmentation* phase, centroid features are upsampled to all corresponding points in the group. Skip link connections are used to consider features from all levels. Figure adapted from [QYSG17a].

from the original input point cloud. Afterwards, the output of the previous abstraction layer is passed forward to the successive layer. From this input point set, a subset of N' points is sampled to be used as centroids. To ensure a consistent allocation of the resulting regions, iterative farthest point sampling is applied. The points derived from this method should optimally be those that have the largest distance from each other.

The Grouping Layer receives both the $(N, (d + C))$ point matrix from before as well as the (N', d) matrix containing the sampled centroid candidates and their position. In this stage, K neighbor points are determined for each centroid. Thus, the output of this stage is a $N', K, (d + C)$ matrix describing N' overlapping regions.

Using ball query, all points within a radius are detected. From these, K points are selected. This method guarantees a fixed region scale along the current level.

In the PointNet Layer, the N' local regions of shape $(K, d+C)$ are encoded in an abstract feature representation of the respective neighborhood. The result from this is a $(N', d + C')$ matrix that consists of the centroid points with respective coordinates as well as the new region feature of dimension C'.

PointNet is used to abstract the local information from the sampled points within a region. For this, the coordinates of the points within a region are translated

such that the corresponding centroid is at the origin. The resulting normalized coordinates are then used together with the C-dimensional feature vectors of the points as input for PointNet.

Another difficulty that has not been considered so far is that point clouds have no uniform density in general. That means that some areas within the point cloud are likely to have far more points than others. This leads to the problem that we either miss fine structures in dense region in case the used scale is to large or receive unreliable results for sparse areas when choosing a scale that is to small.
Qi et al. propose two proceedings to deal with this. The general idea is to execute the grouping and feature extraction process not only on a single but multiple scales. Features from different scales are combined adaptively to the density of the current region into a single representation.
The first variant is called *Multi-scale grouping* (Figure 3.4 (a)). Several features are extracted separately for different scales. The network is trained to learn how to combine these features. During training, variation of the sparsity and uniformity among the point clouds is achieved by randomly dropping out input points.
A computationally cheaper method is *Multi-resolution grouping*. Here, only two part vectors are computed as shown in Figure 3.4 (b). For the first vector, the abstract features from the subregions of the previous level are considered. Given a dense point set as input, this vector is able to catch fine geometrical details as it leverages the results from higher resolutions. However, for sparse point sets, the abstract features from the subsets are likely not very reliably as they are based on even less points. The second vector results from regarding all raw points within the local region. It is therefore able to also handle sparse input. On the other hand, it does not detect the same amount of detail in dense point sets as the first vector.
Depending on the density of the point set, one of these vectors is weighted higher.

For semantic segmentation, an output score needs to be predicted for every point of the original input set. The single features from the different hierarchical levels of the previous stage therefore need to be upsampled and combined. As a result, each point gets an individual feature representation holding information about fine structures in the small local neighborhood as well as the geometry on a larger scale. These features can then be used to predict scores for the semantic classes.
The upsampling process is done by *feature propagation* layers. Each such layer receives as input an $(N, d + C)$ matrix which holds the feature information of the N points corresponding to the centroids of the l-th layer of the feature extraction stage. From this, N_{l-1} features are computed for the next upper level. The features are first upsampled via distance based interpolation: The k nearest neighbors of a point are sampled and a weighted average of their features is com-

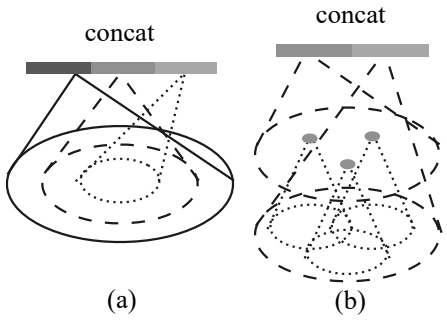


Figure 3.4: **Grouping Variations** (a) Multi-scale grouping: Features are determined for various scales. (b) Multi-resolution grouping: Both raw points within a small region and subregion features computed in the previous level are considered to extract a respective feature. Figure extracted from [QYSG17a].

puted. The weights for this are based on the inverse distance between the current point and the sampled subset. Next, the features resulting from the hierarchical layer at the same scale are concatenated to the respective interpolated features by using skip link connections. The resulting features are then refined for each point separately by a MLP with shared weights among the points (*unit pointnet*).

With their new architecture, Qi et al. achieve state-of-the-art results on various 3D analysis challenges. It is notable that they accomplish this by using only the coordinates as input features. In contrast, PointNet also utilized color information as well as a normalized position of a point. They do not, however, present semantic segmentation results on the S3DIS dataset for comparison with PointNet.

3.2.3 Dynamic Graph CNN (DGCNN)

Another extension of PointNet is *Dynamic Graph CNN* (DGCNN) by Wang et al. [WSL+18]. The key component of their network is the operator *EdgeConv*. This module takes the local neighborhood of a point into account instead of processing the point independently from the others. By doing this, the geometric structure of a point's local surrounding contributes to its established feature representation. An advantage of this module is the possibility to integrate it into an existing network architecture. Wang et al. chose PointNet as basic network and showed improved performance on classification and semantic segmentation tasks.

The idea of the EdgeConv operator is to first construct a local k-nearest neighbors graph based on the feature space and then generate edge features for a point and each of its neighbor. Such a feature represent the relationship between a pair of points. Edge features corresponding to a point are subsequently aggregated

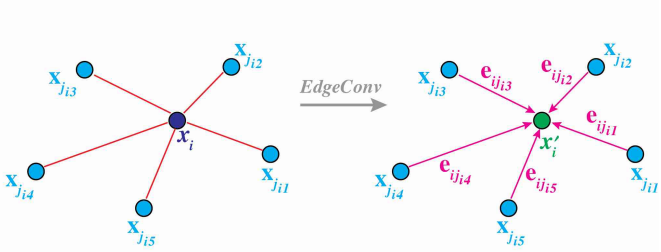


Figure 3.5: **Idea of EdgeConv.** Given a point x_i, edge features are determinded for each of its neighbors. These are then aggregated into a single point representation x_i'. Figure extracted from [WSL+18].

into a feature representation of this point (Figure 3.5). The name EdgeConv is derived from *edge convolutions* due to its performing of convolution-like operations.

For the full DGCNN segmentation model by Wang et al., several of these modules are stacked together (Figure 3.6, top row). The network's name results from the property of the EdgeConv operator to dynamically construct a new neighborhood graph $\mathcal{G}^{(l)} = (\mathcal{V}^{(l)}, \mathcal{E}^{(l)})$ based on the points' feature representation at the current layer l. By doing this, the receptive field of a point is broadened not primary over spatial space but the predicted feature space of the proceeding layers. Hence, in deeper EdgeConv modules, points with similar properties according to the previous part of the network will be the contributing neighbors instead of those that only lie close together in the original 3D input space. A visualization for the distance development in feature space along a series of edge convolutions is provided in Figure 3.7.

Each EdgeConv operator (Figure 3.6, bottom row) at some level l receives as input an $F^{(l)}$-dimensional point cloud $P^{(l)}$ consisting of N points and yields a new $F^{(l+1)}$-dimensional representation for each point. The input point features are provided by the previous layer $l-1$. For the first layer, these correspond to the input features, namely the spatial coordinates of the points. Based on these features, it constructs a k-nearest neighbor graph $\mathcal{G}^{(l)} = (\mathcal{V}^{(l)}, \mathcal{E}^{(l)})$ with vertices $\mathcal{V} = \{1, ..., N\}$ and edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. For a point $p_i^{(l)}$ with corresponding closest points $p_{j_i1}^{(l)}, ..., p_{j_ik}^{(l)}$, this yields the directed edges $(i, j_{i1}), ..., (i, j_{ik})$.
A non-linear function $h_\Theta^{(l)} : \mathbb{R}^{F^{(l)}} \times \mathbb{R}^{F^{(l)}} \to \mathbb{R}^{F^{(l+1)}}$ is applied to determine the edge features

$$e_{ij} = h_\Theta^{(l)}(p_i^{(l)}, p_j^{(l)}). \tag{3.14}$$

Θ denotes the set of learnable parameters for this function. The parameters are shared within a layer l.

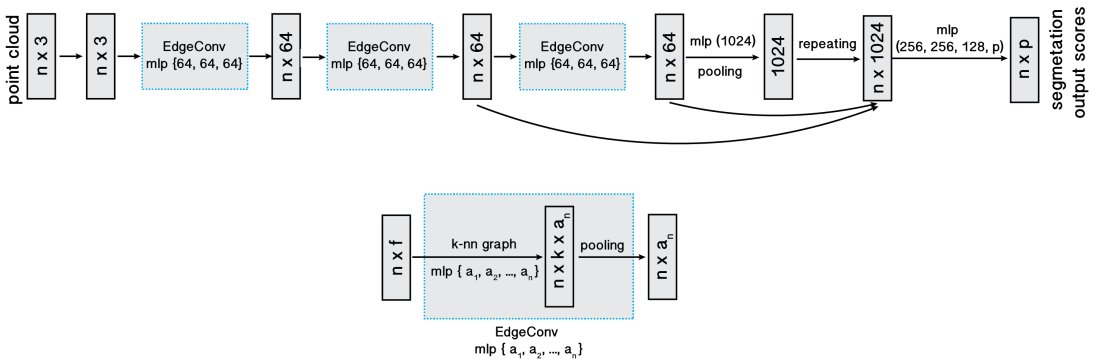


Figure 3.6: **DGCNN architecture:** The input point cloud is passed through a sequence of EdgeConv operators. Thereafter, a max pooling operation is applied to compute a global descriptor. For each point, the global descriptor and all intermediate features from the EdgeConv layers are concatenated. The final output scores for the semantic segmentation prediction result from a MLP.
EdgeConv: The edge convolution operator receives a matrix of shape (N, F) with points and corresponding features as input and determines new feature points (N, F') regarding the neighbor relationship between the points. Regarding some point, edge features are computed for all its neighbors separately using a MLP with shared weights. The number of neurons in each of the n layers in the MLP are denoted as $\{a_1, a_2, ..., an_n\}$. Hence, the MLP yields a (N, k, a_n) matrix containing the edge features of the k neighbors of a point. The edge features associated to a point are subsequently aggregated into a single feature by applying max pooling. This results in new features for each point in form of a (N, a_n) matrix.
Figure adapted from [WSL+18].

The resulting edge features e_{ij} associated with the i-th point are then aggregated into its new feature representation

$$p_i^{(l+1)} = \underset{j:(i,j)\in\mathcal{E}^{(l)}}{\square} h_\Theta^{(l)}(p_i^{(l)}, p_j^{(l)}). \tag{3.15}$$

$\square$ indicates any channel-wise symmetric aggregation function. For their implementation, the authors decided for the max operation.

By applying the EdgeConv subsequently, the neighborhood of the points changes with respect to the learned features. Thus, semantically similar points are taken into account to refine a point's features. This idea of enlarging the receptive field to obtain geometric information is the major difference compared to the previous presented models PointNet and PointNet++.
Wang et al. showed that their DGCNN model outperforms PointNet significantly.

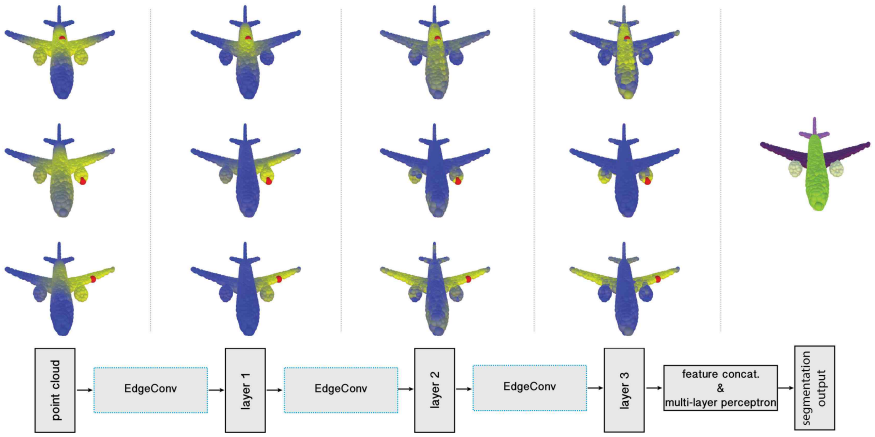


Figure 3.7: **Alteration of feature space resulting from concatenated EdgeConv layers**. The shown features were learned for the part segmentation task. For each EdgeConv layer, the distance between some point (red) and all other points is presented. The points' colors depict the distance, ranging from yellow (nearby point) to blue (distant point). It is noteworthy that, for deeper layers, points belonging to semantically similar parts (e.g. wings, turbines) obtain features that are close in feature space even if the points themselves are not close in the original spatial space. Figure extracted from [WSL+18].

3.3 Similarity Group Proposal Network (SGPN): Instance Segmentation in 3D

Recently, Wang et al. proposed a framework to approach the task of instance-aware semantic segmentation on point clouds [WYHN18]. Inspired by the usage of the popular PointNet for semantic segmentation on point clouds [QSMG17a] as well as present impressive results for instance segmentation on 2D images (e.g. [DHS16, HGDG17]), they examined the combined task. As a result, they presented *SGPN (Similarity Group Proposal Network)* which is the first deep learning end-to-end trainable framework that can be applied for instance segmentation on point clouds.

Their basic idea for the task of instance segmentation follows the strategy of clustering the points in a feature embedding space. They use PointNet to learn an appropriate point representation. Additional to the original task of receiving class information for each point, the learned feature vectors for points corresponding to the same object should be more similar as opposed to the ones of points from different instances. From this embedding the authors aim to find group proposal matching the object instances.

Analogously to the proceeding of semantic segmentation with PointNet, blocks of points are considered separately instead of processing the complete point cloud at once. Therefore, the whole process can be split up into two phases: Initially,

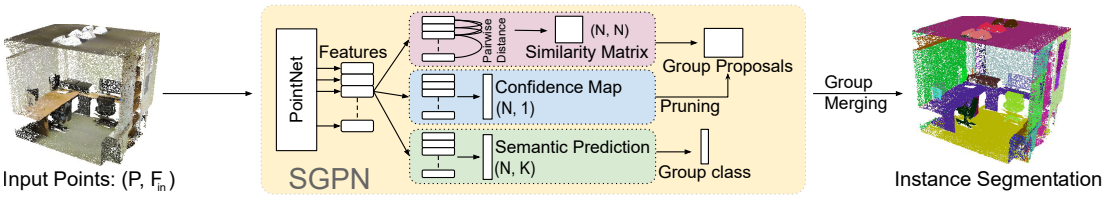


Figure 3.8: **Framework of the Similarity Group Proposal Network.** Figure adapted from [WYHN18]

they look for group proposals within each block of points independently. For the subsequent combining of the partial outcomes, the authors present a block merging algorithm.

In the following part of this section we will describe the proposed method in more detail regarding the network architecture, the clustering procedure, and the block merging method. Moreover, we will use this approach as a baseline to compare with our own results. Details about our implementation as well as the results from experiments can be found in later chapters.

3.3.1 Network Architecture

Given a point cloud as input, the SGPN infers three different outputs: First, we get a *similarity matrix* measuring the affinity between each pair of points and thus indicating how likely they belong to the same group. More specifically, each row of the matrix can be considered as a group proposal belonging to a specific point. Second, each point obtains a *confidence* value implying the quality of the point's proposal. These scores are later used for pruning the set of group proposals. Finally, we also receive class labels for the points in the form of a *segmentation map*.

The complete pipeline is depicted in Figure 3.8. At the beginning, a feature vector is computed for every point of the input point cloud P of size N by using the basic PointNet architecture reduced by the last two layers. The resulting feature matrix F is then passed to the three branches each corresponding to one of the subtasks. In each branch, the point features are refined for the specific task using an additional PointNet layer. This yields the new feature matrices X_{sim}, X_{cf} and X_{sem} with shape (N, F_{out}) respectively.
For each of the branches we get a separate loss function. The total loss is defined as the sum of these losses $L = L_{sim} + L_{cf} + L_{sem}$.

Similarity Matrix

Resulting from the first branch, the proposed similarity matrix S of shape (N, N) represents a set of N group proposals. Each proposal corresponds to a point of

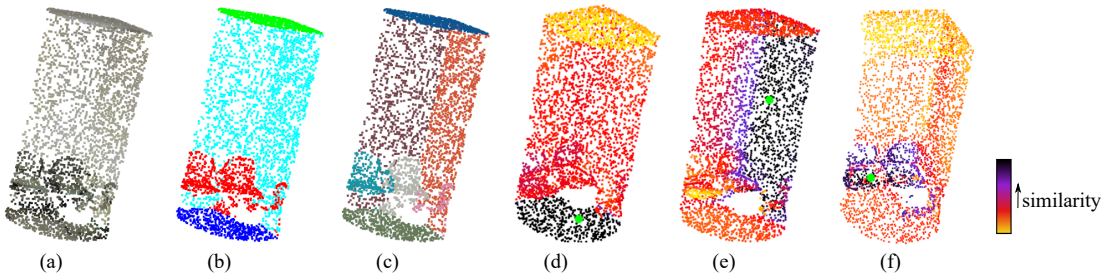


Figure 3.9: **Visualization of similarity measure.** For a pointcloud (a) with ground truth semantic (b) and instance (c) labels, we compute similarity scores for each pair of points (d-f). For a fixed point (green) the distance to every other point is displayed by color. The block presents the corner of a room with two chairs.

the input cloud. The associated potential instance can be obtained by applying a threshold to the proposal.
The objective is to have points from the same instance lie close together in the task specific feature space whereas two points belonging to different point groups should be further apart. The similarity $S_{i,j}$ between each pair of points (p_i, p_j) with $i, j \in [N]$ is given by the L2 distance in the task specific feature space, i.e. $S_{i,j} = \|X_{sim_i} - X_{sim_j}\|_2$. Hence, a small value $S_{i,j}$ implicates a strong similarity, indicating that these points presumably belong to the same object. Figure 3.9 visualize some own example results of this similarity measurement.

For the similarity loss L_{sim}, a double hinge loss is defined. Each pair of points (p_i, p_j) is assigned to one of the following categories:

1. (p_i, p_j) belong to the same object
2. (p_i, p_j) are associated with the same class but correspond to different instances
3. (p_i, p_j) have different semantic labels

The assignment of a pixel pair to one of theses categories is denoted as $C_{i,j}$. The loss function is then defined as

$$L_{sim} = \sum_{i}^{N} \sum_{j}^{N} l(i,j)$$
$$l(i,j) = \begin{cases} \|X_{sim_i} - X_{sim_j}\|_2 & C_{i,j} = 1 \\ \alpha \max(0, K_1 - \|X_{sim_i} - X_{sim_j}\|_2) & C_{i,j} = 2 \\ \max(0, K_2 - \|X_{sim_i} - X_{sim_j}\|_2) & C_{i,j} = 3 \end{cases} \tag{3.16}$$

The constants K_1 and K_2 are chosen such that $K_2 > K_1$. By doing so, we allow points from the same class but different instances to be more similar compared to

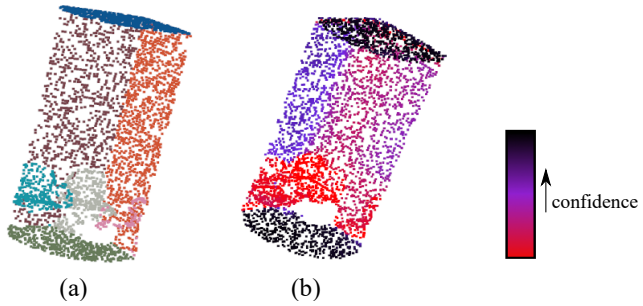


Figure 3.10: **Visualization of confidence scores.** For a pointcloud with ground truth instance labels (a), a confidence score is computed for every point (b). A high value implies that the associated proposal of that point likely is a good instance candidate.

points from different classes. Thus, the network is supported in learning meaningful features corresponding to the semantic segmentation task. To make sure that we can still distinguish between instances from the same class, the weight parameter $\alpha > 1$ is used. Therefore, even if feature point pairs from this category do not lie as far apart, it is penalized more strongly if they are too close.

The resulting similarity matrix is used to indicate relationships between points. We get S' from S by using a category dependent threshold Th_{S_c} such that $S'_{i,j} = 1$ if $S_{i,j} \leq Th_{S_c}$ where c is the predicted class of point p_i.

Confidence Map

The second branch of the network outputs a confidence map CM that indicates the expectation about the quality of each point's proposal. Specifically, we get a $(N, 1)$ vector which provides a confidence score for each point. These scores are learned from the features X_{cf} with an additional PointNet layer.
An example for such a resulting confidence map is depicted in Figure 3.10. It can be seen that the network is certain about the proposals of most points along the ceiling and floor. This is expectable as nearly all rooms have a single ceiling and floor object respectively. Thus, the network can merely focus on classifying these points directly instead of also learning to distinguish between instances. It becomes more interesting for the wall instances. Here, we see that the proposals of the points in the middle of the wall are probably better than those at the corners. This is also preferable as the assignment of a point at the border between two instances of the same class is more difficult compared to a point at the center of an object. However, it can also be seen that for more difficult objects (e.g. chairs), the confidence scores decrease in general.

For computing the loss L_{cf}, meaningful confidence values are required that can be learned to be regressed. For estimating such values, the predicted similarity

matrix is compared to the ground truth groups. For this, a binary group matrix G of shape (N, N) is generated with $G_{i,j} = 1$ if points p_i, p_j belong to the same instance. Furthermore, the binarized similarity matrix S' is taken into account. For each point p_i, we compute the intersection over union between the corresponding rows G_i and S'_i. The resulting value represents the quality of the similarity prediction for a point and is used as the expected confidence score.
The L2 loss between the described expected confidence map and the inferred result is the loss of this branch.

Semantic Segmentation Map

The third branch yields class label predictions in the form of a (N, K) matrix M_{sem}. Similar as it is done for the confidence map, an additional PointNet layer with K output channels is appended to the branch specific feature layer. K corresponds to the number of classes and depends on the used data. Each entry $M_{sem_{i,c}}$ expresses the probability that point p_i should be labeled with class c. From this, the final class label prediction for a point p_i can be obtained by $\ell_{sem_i} = \text{argmax}_c \, M_{sem_i}$.

A weighted cross entropy loss function L_{sem} is used for this branch. For the weights, we first compute the frequency for each class c as $freq(c) = N_c / N_{c_p}$ where N_c is the number of points of class c along all given point clouds and N_{c_p} is the number of point clouds where points from class c appear. The weight for each class c is set to $w_c = median(freq)/freq_c$.
The combination of this branch and the common feature network actually matches the complete PointNet architecture.

3.3.2 Group Merging

Given the partial outputs presented in the previous section, Wang et al. present a method to group the points of the current block into instance proposals. The output from the network consisting of the similarity matrix S, the confidence map CM, and the semantic segmentation map M_{sem} are the inputs for Algorithm 1. As a result, we receive an instance segmentation PL for the current block where a group label is assigned to each point.
The following explanation is based on the description in [WYHN18]. We will discuss variations in the available code by the authors thereafter [Wan18a].

To start with, we get N group proposals from the similarity matrix S. Ideally, points from the same instance yield the same proposal and there is no overlap between proposals from points belonging to different objects. Obviously, there will be noise in the data as well as inaccuracy in the learned features. This results in the need of a process for pruning and merging the proposals.
As a first requirement, we only consider those proposals for which the confidence

Algorithm 1 Group Merging Algorithm (based on [WYHN18, Wan18a]

Input S, CM, M_{sem} for point block P
Output Instance segmentation PL for block P

procedure GROUP MERGING(S, CM, M_{sem})
  for every class c **do**
    $proposals,\ P_{valid_c} \leftarrow []$
    for point index i in P **do**
      $S'_i \leftarrow$ apply class specific threshold Th_{S_c} on S_i,
          remove points from S'_i with diff. predicted class label
      if $\text{argmax}_c\, M_{sem_i} = c,\ CM_i \geq Th_c$, and $size(S'_i) \geq Th_{M_{num}}$ **then**
        append i to P_{valid_c}.
    if $len(P_{valid_c}) = 0$ **then**
      $proposals \leftarrow$ single proposal of all points with class label c
    else ▷ Non-Maximum Suppression
      for proposal idx i in P_{valid_c} and respective S'_i **do**
        for S'_j in $proposals$ **do**
          if $IoU(S'_i, S'_j) > Th_{M_{iou}}$ or $Subset(S'_i, S'_j) > Th_{M_{sub}}$ **then**
            if $size(S'_i) > size(S'_j)$ **then**
              replace S_j in $proposals$ by S'_i
            mark S'_i as merged
            continue
        if S'_i was not merged **then**
          append S'_i to $proposals$
    for S'_j in $proposals$ **do**
      $PL_i \leftarrow$ new instance label I for all points p_i in S'_j
  for every instance I **do**
    if number of points assigned to $I < Th_{M_{num}}$ **then**
      $PL_i \leftarrow$ unlabeled, where $PL_i = I$
      remove I
  for every point p_i without instance label **do**
    $PL_i \leftarrow$ instance that appears most often in own proposal S'_i
  return PL

score of the corresponding point is above some threshold Th_c. Moreover, the number of points in the proposed group needs to be higher than another threshold $Th_{M_{num}}$.
Afterwards, a Non-Maximum Suppression procedure is applied on the remaining set of valid proposals. For this, we compute the *Intersection over Union* (IoU) for pairs of proposals. In case the resulting value is higher than some threshold $Th_{M_{iou}}$, the proposal with the lower cardinality is discarded. This yields the final set of instance proposals. However, there might still be some points which are

assigned to several proposals. The authors expect the proportion of these to be rather low and propose to assign them randomly to one of the associated proposals. Finally, for each instance we select the most common class label from all corresponding points regarding the semantic prediction from M_{sem}.

The authors also published their code [Wan18a]. There are some minor differences and expansions compared to the proposed explanation in the paper. These changes were probably done to reduce the computational effort and improve the reliability of the result.
First, the predicted class labels are already used for a pre-segmentation of the points. In case the confidence score is to low for all points within a class, all points are assigned to a single instance.
Instead of comparing all pairs of valid proposals, for a current instance candidate, we only consider the previous proceeded, kept proposals. Furthermore, not only is the IoU computed but it is also checked whether a large part of the current proposal is a subset of a previous proposal.
The case that a point appears in several proposals is merely ignored in the authors' code. By simply iterating over all proposals and labeling all associated points with a new instance label, each point gets assigned to the instance of the last proposal where it appears. By doing this, it might happen that the number of points within an instance decreases below the minimum required number Th_{min}. In this case, the instance proposal is subsequently removed.
Finally, there are still some points that were not assigned to any instance so far. For these, their corresponding binary proposal is considered. From all points within this proposal, the instance label that appears most often is selected. It is possible that a point now gets assigned to a group corresponding to a different class label. However, due to noise in the semantic prediction, this is reasonable and sometimes even helps to smooth both the semantic and instance segmentation.
We will present more details regarding among others the choice of the threshold later in Section 5.3.1.

3.3.3 Block Merging

The group merging algorithm yields a final instance labeling for a single block. Thus, a merging of these block results is required to get the instance segmentation of the complete point cloud.
For this, the authors discretize a scene into a grid V of fixed size. While iterating over the blocks, instance labels are determined for the cells V_k of the grid. Afterwards, all points that lie in a cell are assigned the corresponding label.

The main reason for requiring a separate merging algorithm for the predicted blocks is that we have labeled the instances independently of their appearance

in different blocks. That means, an object that appears in more than one block might get assigned different instance labels. Moreover, different objects from two blocks might have the same instance label. It should be noted that overlapping blocks are used during testing.
Expectedly, there will be noise in the instance prediction. On the one hand, instance parts in different blocks belonging to the same object might not fully match in the overlapping sections between blocks. Hence, these need to be merged. On the other hand, there might be overlap between predicted instance parts that belong to different objects which thus need to be differentiated.

Algorithm 2 Block Merging (based on [WYHN18])

Input Instance segmentation PL^b for every block P^b of point cloud P,
merged semantic segmentation L_{sem} for P
Output Instance prediction L and refined semantic labels L'_{sem} for P
procedure BLOCK MERGING($[PL^b], [P^b]$)
$V \leftarrow$ unlabeled voxelgrid of size $(400, 400, 400)$
$InstCount \leftarrow 0$
for every block P^b with predicted instance labels PL^b **do**
for every instance I_j that is present in PL^b **do**
$V_{I_j} \leftarrow V_{k_i}$ for all k_i where $PL^b_i = I_j$,
k_i defines the cell where point p^b_i is located
$V_t \leftarrow$ all cells in V_{k_i} that were already assigned a label
$I_m \leftarrow mode(V_t)$
if number of cells in V_t with value $I_m > Th_G$ **then**
$V_{I_j} \leftarrow I_m$
else
$V_{I_j} \leftarrow InstCount$
$InstCount \leftarrow InstCount + 1$
for every point p_i in P **do**
$L_i \leftarrow V_{k_i}$, k_i defines the cell where p_i is located
for every instance I in L **do**
$l \leftarrow$ mode of semantic prediction L_{sem} for points in instance I
$L'_{sem_i} \leftarrow l$ for all i where $L_i = I$
return L, L'_{sem}

Algorithm 2 describes the block merging procedure from [WYHN18].
It receives a sequence of block predictions from the grouping algorithm as input and returns a final instance segmentation L of the full point cloud. The single blocks are processed in a snake pattern as shown in Figure 3.11. This always yields the currently highest possible overlap to the previous handled blocks.

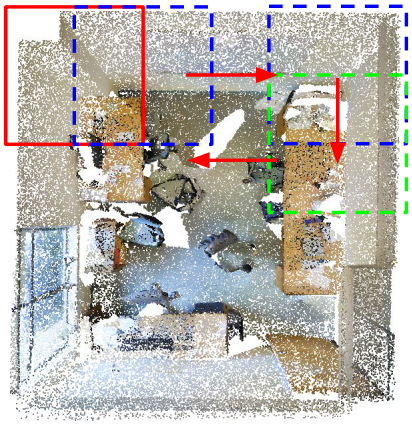


Figure 3.11: Order in which point blocks are processed. Figure extracted from [WYHN18]

For a block P^b and predicted instance labels PL^b, the discretized cell positions k_i are computed for all points p_i in P^b. Next, we iterate over all instances that were detected in the current block. For each instance I_j, V_{I_j} is the set of voxel cells in which the points from this instance are located. Furthermore, V_t is determined as the subset of V_{I_j} containing all cells that already got a label assigned from a previous block. These are the cells that require a merging with the new proposed instance I_j. There are two options: Either the cells V_{I_j} are associated with an already existing instance from V_t or V_{I_j} gets assigned a new instance label. The choice depends on the frequency regarding the mode of the instance labels in V_t. If some instance I_m is highly present in the overlapping region, all points of the new proposal obtain the existing group label. Otherwise, a new instance is defined by all cells in V_{I_j}.
Finally, a category is assigned to each instance by taking the mode of the predicted semantic labels from the associated points. The algorithm therefore additionally yields a refined semantic segmentation L'_{sem}.

3.3.4 Drawbacks

There is one major limitation of the proposed SGPN framework. As the instance segmentation is determined on subblocks of the point cloud independently, an additional merging algorithm is required. This involves the estimation and fine-tuning of several threshold parameters.
Moreover, instance features X_{sim} computed in the similarity branch are not expected to be consistent over several blocks. This means, that a point which appears in multiple overlapping blocks might get assigned completely different instance embedding vectors. The reason for this is that, during training, the network only learns to comply with the pairwise relationship within a small area corresponding to a single block but not for the whole point cloud. Hence, a

transfer of these features into a common point cloud as done for the semantic segmentation task is not possible. Distinct instances that do not appear in at least one common block might get assigned similar feature vectors. Therefore, if these features are considered in a global view, the corresponding points would get assigned to the same instance group. Another problem results from the fact that there is no restriction that a single point obtains the same feature vector within the context of different blocks.
Besides this problem, the usage of the similarity matrix which is quadratic in the number of considered points results in relatively high computational effort and memory requirement. This negatively affects both training and inference time.

4

3D Bird's Eye View Instance Segmentation (3D-BEVIS)

As the main contribution of this thesis, we present *3D-BEVIS* (*3D Bird's Eye View Instance Segmentation*) for instance segmentation on point clouds.
Similar to what was done in the SGPN framework (Section 3.3), we want to learn point-wise features according to which we are able to identify instances. However, our method differs from the existing approach in that the learned feature embedding space is globally coherent. This means we are able to perform clustering within a complete scene instead of single blocks. Hence, we do not rely on a heuristic merging algorithm.

We achieve this by utilizing additional 2D input data. Inspired by [LYU18], we make use of bird's-eye views (BEV) from the entire scene. Applying a simple 2D instance segmentation architecture, we learn an instance feature embedding space for the subset of points that is visible in the BEV images (Section 4.1). These representations can be considered as global feature representations as they were predicted with respect to the whole scene.
Given a set of predicted global instance features, we transfer these to the respective 3D points in the point cloud (Section 4.2). As many points are not seen in the BEV rendering, not all points get an instance feature assigned. Hence, we apply the architecture of a point cloud feature extraction network (Section 3.2) to propagate the learned global instance features to the points that were not considered before.
The resulting features for the individual blocks can be merged into a common point cloud. The points of the whole scene are subsequently grouped according to a clustering algorithm into instance proposals (Section 4.3). After a post-processing step where instances with an inconsistent semantic labeling are split up, the method yields the final instance segmentation.

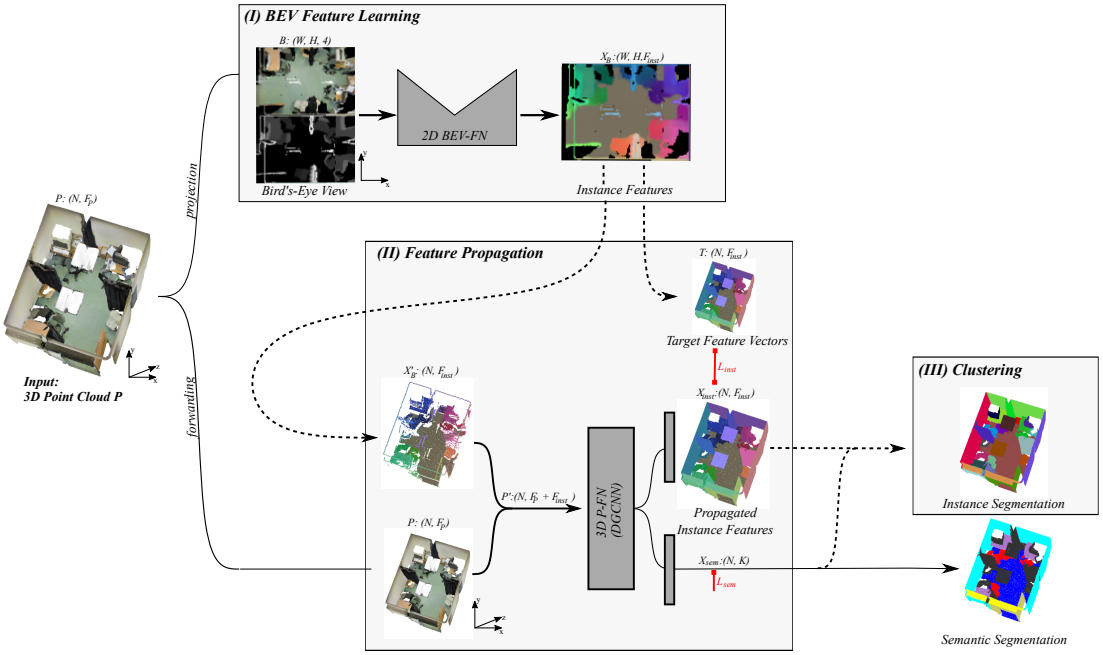
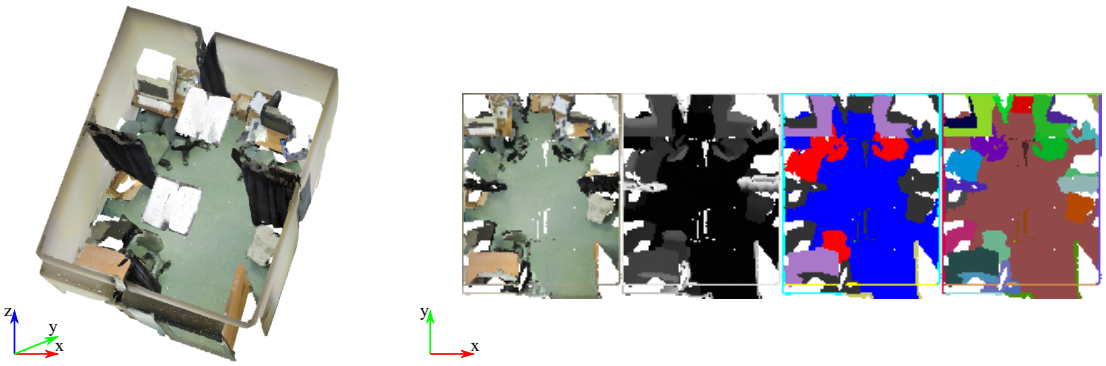


Figure 4.1: **Pipeline of the 3D-BEVIS Network.** The framework comprises three stages: (I) Given an input point cloud P which consists of N points described by F_P dimensional features, a BEV image B is rendered. A 2D feature extraction network (*2D BEV-FN*) is applied to learn global instance features X_B (Section 4.1). (II) The instance features are propagated to the whole point cloud. For this, the predicted features are used as additional input channels X_P besides the original point cloud data P. By applying a 3D feature network (*3D P-FN*), we simultaneously learn to transfer the partially present instance features to all points (X_{inst}) of the point cloud and to predict semantic logits X_{sem}. During training, fixed target features T are specified. These result for each instance by averaging over all corresponding pixels in X_B (Section 4.2). (III) While we can directly transfer a semantic segmentation from the semantic features, an additional clustering method needs to be applied to obtain an instance segmentation (Section 4.3). For visualization, instance features are embedded into RGB space by applying PCA. Dashed lines represent the transitions between the single stages.

The complete framework is depicted in Figure 4.1. Overall, for an input point cloud P, both semantic and instance point-wise labels are predicted.
In the following part of this section, we will have a look on the single components of the proposed framework in more detail.



(a) (b)

Figure 4.2: **Bird's-Eye View renderings from a point cloud.** (a) Original 3D point cloud (b) BEV, Left to Right: Color values, depth map, ground truth semantic segmentation and instance segmentation

4.1 Global Instance Features from Bird's-Eye View

We aim to learn globally consistent instance features with respect to the complete point cloud. This enables a grouping of the points by regarding the overall context of the scene.

In the first phase of our pipeline for *2D BEV Feature Learning*, we learn a globally consistent embedding space for the entire scene (Figure 4.1, Phase (I)). For this, we leverage an additional, intermediate representation of the point cloud to process the entire scene at once. In particular, the entire scene is regarded from a BEV perspective (Figure 4.2). This involves the following advancements: Firstly, the resulting regular structure of the data allows the application of CNNs to learn instance features (see Section 2.3.2). Secondly, the number of points that is considered at once is decreased as not all points are projected into the image due to occlusion. The limited number results in a reduced computational complexity.

Given a 3D point cloud P, we generate a BEV rendering B from this input. This image is passed through a 2D BEV-Feature Network (*2D BEV-FN*) which yields pixel-wise instance features X_B.
The BEV image B is described by a $(W, H, F_{B,in})$ matrix where W, H denote its shape resulting from the discretized maximum position values of the points along the x- and y-axis respectively. $F_{B,in}$ is the number of input feature channels for each pixel.
For generating the BEV image, the points are projected onto a grid on the ground plane. If several points fall into the same cell, only the highest point above the ground plane is taken into account. We denote the subset of points that are

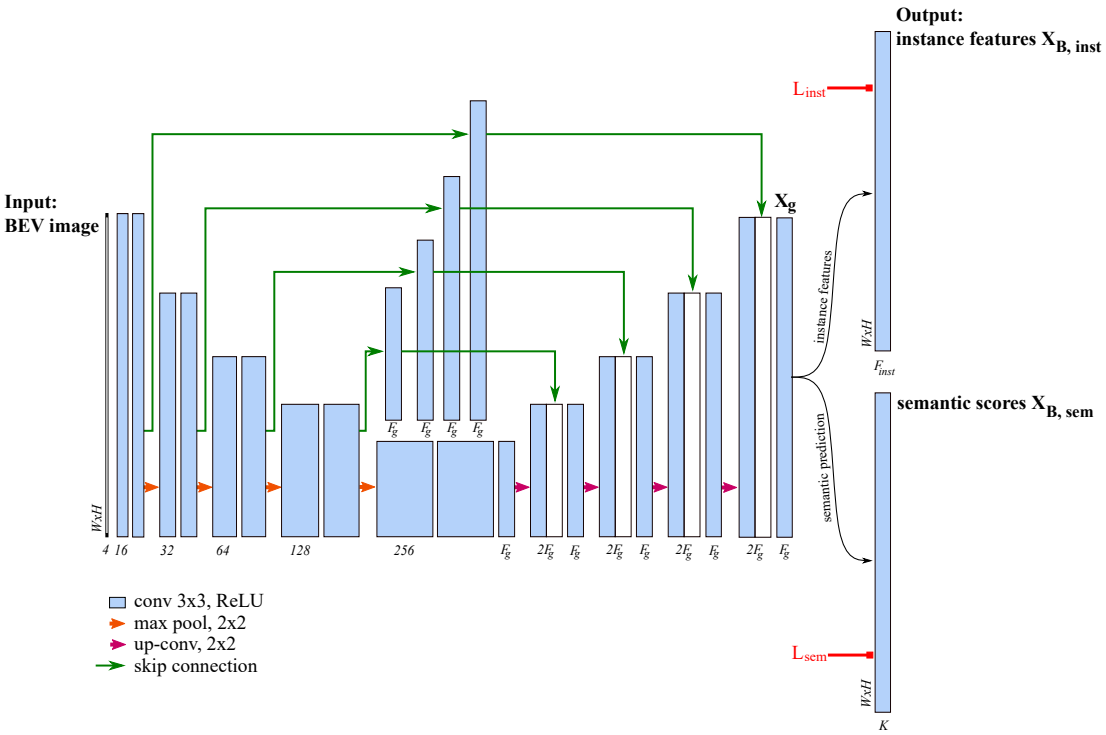


Figure 4.3: **Instance feature learning from bird's eye view renderings.** The BEV image is passed through a simple encoder-decoder architecture to receive point-wise features X_g. Using additional convolutional layer respective, task specific features for instance and semantic segmentation are obtained. The network is trained regarding the losses for each of the named tasks. However, only the set of instance features is utilized for the subsequent phase of the pipeline.

projected onto the image as $P_B \subset P$. The resolution is chosen such that the finest possible detail is retained but also a high proportion of the cells is occupied. For this work, each pixel occupies an area of 3×3cm. Besides the original color value of a point, we also keep its height above the ground within an additional depth channel. Thus, we get a RGBD feature for every occupied pixel. During the training of the 2D BEV-FN, the ground truth semantic and instance labels of the projected points are required. To allow for a later back mapping of the estimated feature vector, point indices are saved as well.

For the input BEV image B, the 2D BEV-FN yields features $X_B = (W, H, F_{inst})$ (Figure 4.3). These encodings differentiate between pixels belonging to distinct object instances. Besides this, the network is encouraged to also take semantic relevant aspects into account. For this, the model first learns general features $X_g = (W, H, F_g)$ which are afterwards passed through two separate branches. In the upper branch, the general features are refined for the instance segmentation

task ($X_{B,inst}$) whereas the lower branch is used to predict class labels ($X_{B,sem}$).

We utilize a simple encoder-decoder architecture in the style of FCN [SLD15] to learn a feature encoding for each pixel of the BEV rendering. A learned embedding resulting from this kind of network has already been succesfully applied for instance segmentation on 2D data in previous work like [HXKH18, BNG17, FWR$^+$17]. Our network architecture as depicted in Figure 4.3 was mainly inspired by the works in [RFB15, HXKH18].
However, due to the kind of input data, we need to consider some additional factors. First of all, our BEV images differs a lot from photographs that are commonly regarded for the instance segmentation task. For this reason, the usual approach of adopting the weights of a pretrained feature extractor network like ResNet [HZRS16] seemed inappropriate in our case. Furthermore, the available amount of input data is rather limited even after involving data augmentation. Therefore, we prefer to limit the number of convolutional layers and hence the trainable parameters to prevent our model from overfitting.

Our network consists of a sequence of convolutional and deconvolutional blocks in correlation with skip connections.
A convolutional block is composed of two convolutional layers with kernel size (3, 3) followed by the ReLU non-linear activation function. Afterwards, a pooling layer with a (2, 2) window and stride size 2 is applied. The number of feature channels is doubled within the first convolutional layer of a block. This means a block on the lth level of the network receives the input $B_{enc}^{(l)} = (W^l, H^l, F^l)$. From this, we first receive $B_{enc}^{\prime(l)} = (W^l, H^l, 2F^l)$ following the convolutions and $B_{enc}^{(l+1)} = (\frac{W^l}{2}, \frac{W^l}{2}, 2F^l)$ after applying the pooling layer. An exception is made for the first block where we start with a feature dimension of 16.
Subsequently, we add the same number of deconvolutional blocks. Each of these receives the output of the previous block from the lower level $B_{dec}^{(l)} = (W^l, H^l, F_g)$ as input and computes upsampled features for the next higher level $B_{dec}^{(l-1)} = (2W^l, 2H^l, F_g)$. Similar to the network architecture used in [HXKH18], we keep the number of feature channels constant for the deconvolutional part of the network. Moreover, batch normalization is applied after each convolution layer in the decoder. Within each block, we first apply a transposed convolutional layer to the output of the block from the previous layer $B_{dec}^{(l)}$ to obtain the spatial dimensions of the next upper level. In addition, skip-connections are inserted between the encoder and decoder. For this, another convolution is applied on the output of the respective level of the encoder part $B_{enc}^{\prime(l-1)}$. The outcome is concatenated to the current decoder feature. The combined feature is then passed through another convolution layer to obtain the general encoding X_g with the feature dimension F_g.
Next, the network is split into two branches: one for refining the current encodings into features relevant for instance segmentation and one for learning a

Figure 4.4: **Visualization of an instance embedding.** Left to Right: Input BEV B, ground truth semantic segmentation and instance segmentation, instance features X_B with color labeling regarding either semantic or instance annotation. The dimension of the instance features was reduced to two by applying PCA. We can clearly distinguish groups of points belonging to different instances. Especially, we are interested in being able to differ between clusters of points corresponding to different instances but the same class as point groups from different classes can also be seperated by their semantic prediction.

semantic segmentation. Each branch thus receives a (W, H, F_g) matrix as input and applies one additional convolutional layer. Within the instance branch, we get an output tensor $X_{B,inst}$ of dimension (W, H, F_{inst}) where F_{inst} denotes the dimension of the global feature instance space that we aim to learn (Figure 4.4). The second branch estimates semantic scores $X_{B,sem}$ with a (W, H, K) matrix for K different classes.

Both instance and semantic features contribute to the objective loss function, but only the instance features $X_{B,inst}$ are passed on to the next stage of the pipeline. The applied loss function is therefore composed of

$$L = L_{inst} + L_{sem}. \tag{4.1}$$

As stated in Section 3.2, we apply the multi-class cross-entropy loss function for the semantic loss L_{sem}. It should also be noted that only those pixels that are occupied by a point contribute to this loss.
The instance loss L_{inst} is based on a similarity measure (Section 3.1.2). This ensures feature vectors of points belonging to the same object to be similar while encouraging a large distance in the feature space between features corresponding to different instances. Following the result regarding the usage of different loss functions in our ablation study (Section 5.4.1), we decided for the L2 loss described in Section 3.1.2:

$$L_{var} = \sum_{c=1}^{C} \sum_{x_i, x_j \in S_c} s_{i,j}, \qquad L_{dist} = \sum_{\substack{c,c'=1 \\ c \neq c'}}^{C} \sum_{\substack{x_i \in S_c \\ x_j \in S_{c'}}} [\delta_{dist} - s_{i,j}]_+ \tag{4.2}$$

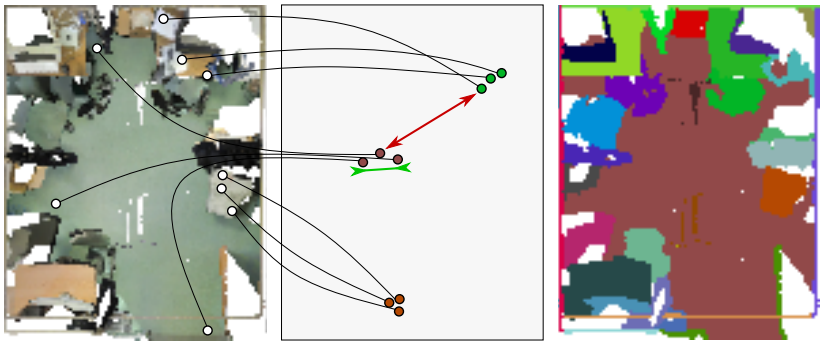


Figure 4.5: **Sampling strategy for evaluating L_{inst}.** A fixed number M of pixels is sampled per instance for computing the instance loss. The corresponding features are either pulled towards each other if belonging to the same instance or pushed apart otherwise. For visualization, we choose $M = 3$ and only display embeddings for three objects.

To compute the instance loss, we use the same sampling strategy as applied in [FWR+17, ND16]. Instead of comparing all pairs of feature vectors, we sample a subset S_c containing M pixels for each instance c (Figure 4.5).

Summarizing, we obtain instance features corresponding to a subset of points distributed over the whole scene. These features are globally consistent and can thus be used as a basis for later grouping.

4.2 Propagation of Bird's-Eye View Features to the Point Cloud

From the 2D Feature Learning phase, we obtain instance features $X_B = (N_B, F_{inst})$ for those points that are projected into the BEV image B. These features can be considered to be globally consistent across a scene as they were trained to distinguish between every pair of objects within the whole scene. We want to propagate these features to obtain representations $X_{inst} = (N, F_{inst})$ for all points of the point cloud. These features can later be used for instance segmentation by applying a clustering algorithm globally.

The main idea is to use the learned features X_B as additional input channels for a point cloud feature network from Section 3.2. This network is trained such that it propagates the predicted encodings to all points for which we have no feature from X_B. Thus, a point $p_i \notin P_B$ that has not been considered so far should obtain a similar feature vector as a point $p_i \in P_B$ from the same instance. Due to this usage, we will refer to this network as *3D Propagation Feature Network (3D P-FN)*.

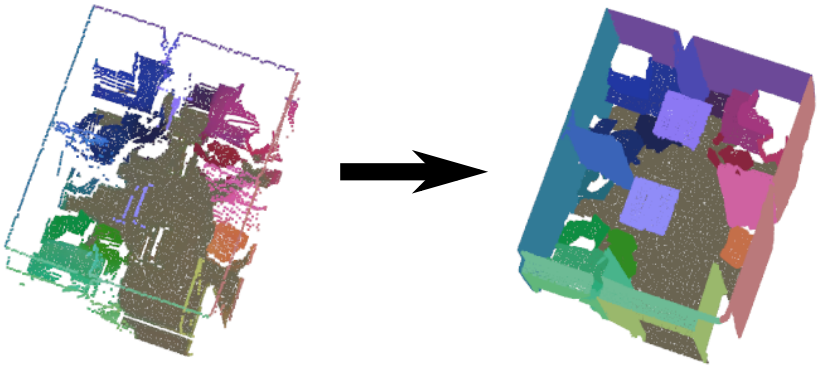


Figure 4.6: **Target features T for instance feature propagation.** For each instance, the mean of all predicted BEV features that belong to the respective group is computed. The specific target feature for a point is chosen according to the point's instance.

This process is depicted in the second stage for *Feature Propagation* in Figure 4.1.

The new input point features P' consist of extended point features P with dimension $(N, F_P + F_{inst})$. For every point that was projected into the BEV image, $p_i \in P_B$, we obtained a predicted instance feature $x_{i,B}$. This feature is stacked to the original point cloud feature $x_{i,P}$ yielding $x'_i = \left[\begin{smallmatrix} x_{i,P} \\ x_{i,B} \end{smallmatrix}\right]$. For the remaining points, zero-padding is applied.
From here, we want to predict the final point-wise instance features X_{inst}. In contrast to the 2D BEV-FN of the pipeline, the 3D P-FN is not free to learn an arbitrary instance embedding under the consideration of a pairwise similarity based loss function. Instead, *target features* $T = (N, F_{inst})$ are specified for all points (Figure 4.6). These target features are determined based on the prediction from the BEV feature extraction phase. Target features might differ slightly compared to the direct BEV prediction X_P. We use the higher amount of details in the point cloud compared to the 2D BEV projection to obtain an improved prediction regarding both smoothness within an instance and distinction between objects.

The target feature vectors that the network aims to learn are determined as follows: For each instance $I \in \mathcal{I}, I \subset P$ we consider all points $P_{B,I}$ that belong to this object and were projected into the BEV image. For these, we got the estimated features $X_{B,I}$. The ground truth feature $t_I \in \mathbb{R}^{F_{inst}}$ is set to the mean of these:

$$t_I = \frac{1}{|I|} \sum_{x_i \in X_{B,I}} x_i. \tag{4.3}$$

As in Section 4.1, we not only want to predict an instance feature but also infer a semantic class label for each point. Therefore, both a feature vector $x_{i,inst} \in \mathbb{R}^{F_{inst}}$ and semantic class scores $x_{i,sem} \in \mathbb{R}^K$ are predicted for each point p_i.
Equal to Equation 4.1, the losses L_{inst} and L_{sem} are summed up for the overall loss function. The computation of L_{inst} differs from the presented instance loss functions in Section 3.1.2 as there are specific target feature vectors. We apply the mean squared error function

$$L_{inst} = \frac{1}{N} \sum_{i=0}^{N} (x_{i,inst} - t_{I(i)})^2. \tag{4.4}$$

As before, L_{sem} is the cross entropy loss.

We use the DGCNN model for semantic segmentation from [WSL+18] for our 3D P-FN. The model is taken up to the second last layer as the base network to learn a meaningful feature embedding for point clouds. As done for the 2D BEV-FN, this yields a general encoding which is passed to two distinct branches where it is refined with another PointNet layer. In the first one, the network learns the transfer of the pre-estimated instance features to the complete set of points X_{inst}, whereas characteristics relevant for semantic segmentation are extracted in the second branch (X_{sem}).
Due to the proceeding of DGCNN, the point cloud is split up into blocks which are handled separately. However, although the instance features are determined for each block individually, the learned representations are still globally applicable for instance segmentation of the complete scene. This results from the use of the global target feature vectors. Therefore, the inferred feature vectors of the single blocks can be merged into a common point cloud. For a point that received multiple feature prediction, the mean of these is used as it is already done for the semantic segmentation scores.

This results in the final output of this part of the framework: globally valid instance features $X_{inst} = (N, F_{inst})$ as well as semantic category scores $X_{sem} = (N, K)$.

4.3 Instance Grouping

After obtaining point-wise feature instances X_{inst} and semantic scores X_{sem} for the whole scene, these are used to segment the set of points into the individual instance groups. We apply the Mean Shift algorithm to cluster the points with respect to the learned instance feature representation. As described in Section 3.1.2, this method does not require the total number of clusters as input and is thus suited for the instance segmentation task.

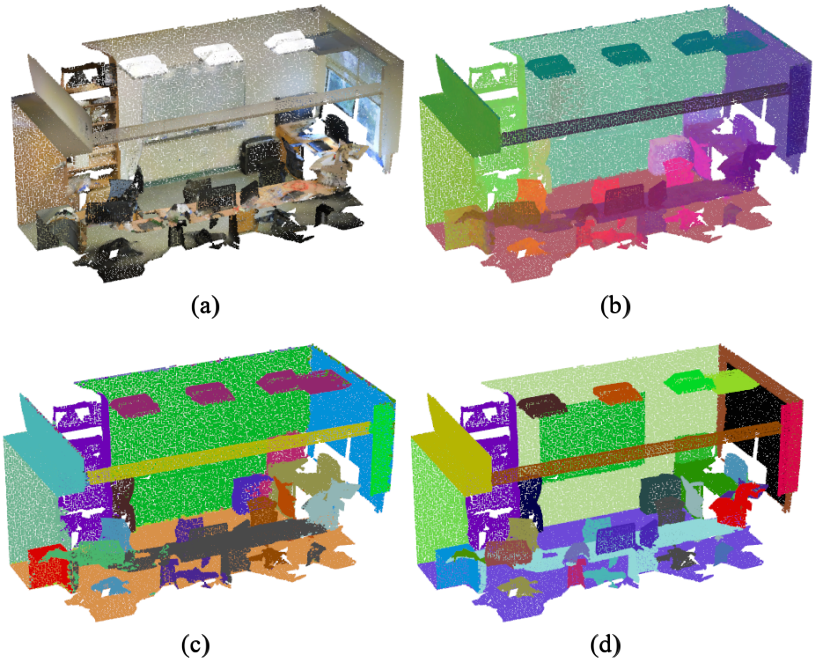


Figure 4.7: **Mean Shift Clustering.** For an input point cloud (a) with predicted point-wise instance features (b), the point cloud is partitioned into instance proposals by utilizing the Mean Shift clustering algorithm (c). Ground truth instance annotations are shown in (d).

An example is shown in Figure 4.7. The learned instance features provide a good basis for segmenting objects that are well viewable from a bird's-eye perspective like chairs or tables. However, difficulties occur for thin objects along the wall like boards.

To deal with the problem that some objects are hardly identified from the bird's-eye view, we also take the semantic prediction into account. From the obtained scores X_{sem} of the previous stage, the final class label ℓ_i of p_i corresponds to the category with highest value in $X_{sem,i}$.
For each proposed instance I, the semantic labels of all points belonging to this proposal are considered. In case that more than one class is strongly represented, the instance is split up (Figure 4.8). More specifically, we obtain a new instance I_c for every class c if at least $Th_{min,c}$ points in I have predicted semantic label c. $Th_{min,c}$ is chosen to be proportional to the average number of points per instance of the respective class, $Th_{min,c} = c\overline{N}_{inst,c},\ c < 1$. Points from I whose predicted class label does not correspond to any of the new sub-instances are afterwards assigned to an instance by regarding the k-nearest neighbors among the labeled points.

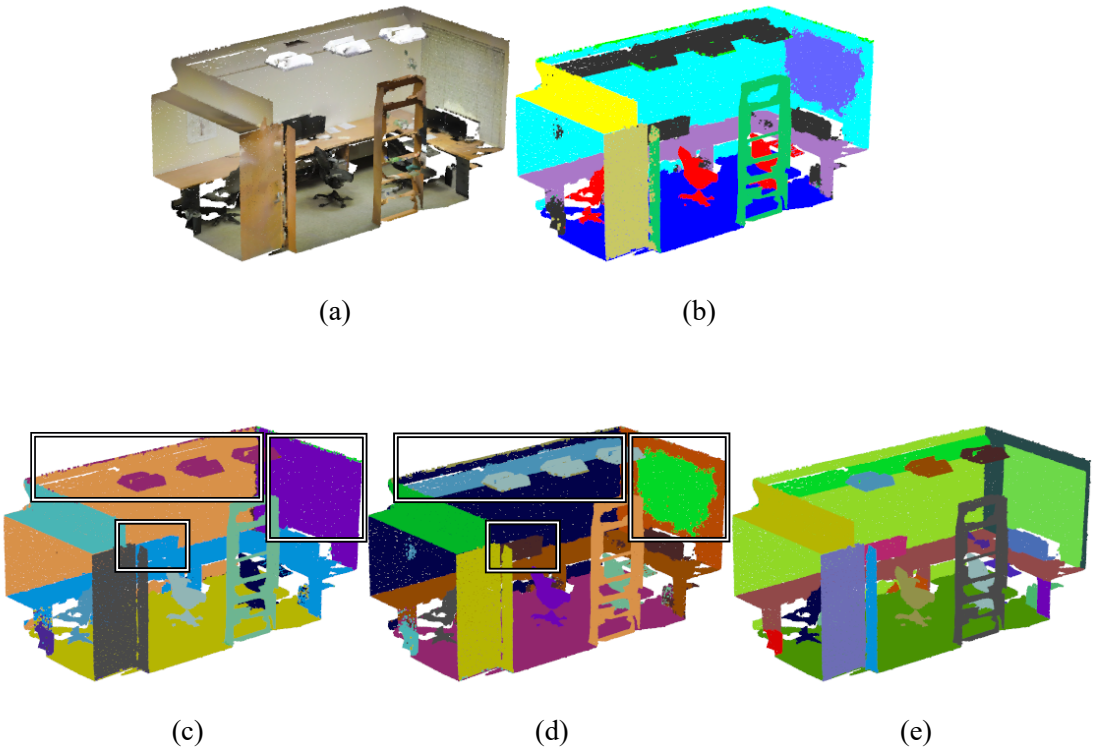


Figure 4.8: **Post-processing of clustered scene.** For an input point cloud (a), the predicted semantic segmentation (b) is used to refine the initial instance segmentation obtained after applying Mean Shift (c). This yields the final instance segmentation (d). Interesting areas are outlined here. For comparison, the ground truth instance segmentation is shown as well (e).

5 Experiments

In this chapter, we present evaluation results for 3D-BEVIS on 3D instance segmentation. Furthermore, we compare our method to SGPN by Wang et al. [WYHN18] which is currently the only other approach for the named task that works directly on point clouds. We examine the performance on the two datasets S3DIS and ScanNet which are described in Section 5.1 in more detail. The underlying evaluation metric that is used for the comparison is outlined subsequently in Section 5.2. We then describe training settings for both SGPN (Section 5.3) and 3D-BEVIS (Section 5.4). Considering our own method, we also explore alternative settings to justify our design choices. Afterwards, we compare quantitative and qualitative evaluation results of both methods (Section 5.5).

5.1 Datasets

Our presented framework 3D-BEVIS as well as SGPN can be applied for instance segmentation on large point cloud scenes. Although the number of 3D datasets for semantic segmentation has largely increased recently, the fraction of those with required instance annotations is still relatively small.
For this thesis, we focus on the large scale indoor datasets S3DIS [ASZ+16] and ScanNet [DCS+17]. Both of them were also considered in the work of Wang et al., thus allowing a fair comparison.

Stanford 3D Indoor Semantics Dataset (S3DIS) This indoor 3D dataset consists of point cloud scans for six indoor areas from three different buildings [ASZ+16]. A total number of 272 rooms is included which can be associated to certain room types like offices, conference rooms, hallways, or open spaces. An example of such a room is depicted in Figure 5.1. The scans were retrieved from Matterport cameras which yield relatively dense point cloud representations. Both semantic and instance annotations are provided for each point. Overall, there are 13 different categories that can be distinguished into structural

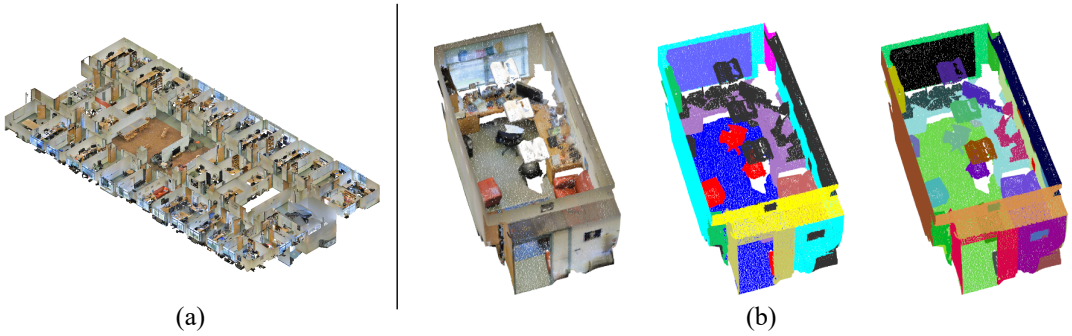


Figure 5.1: **Stanford 3D Indoor Semantics Dataset (S3DIS).** (a) The complete area 6 of the S3DIS dataset composed of 48 rooms. (b) A point cloud scan of a single room with rgb information (left), semantic (middle), and instance (right) label annotations. Points belonging to the ceiling were removed to enable a better visualization.

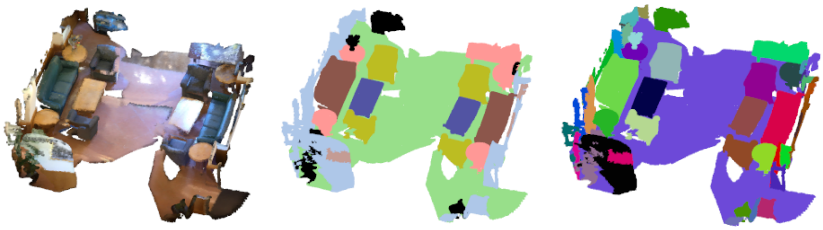


Figure 5.2: **ScanNet.** An example scan of a scene with rgb information (left), semantic (middle), and instance (right) label annotations. Points belonging to the ceiling were removed to enable a better visualization.

building elements (ceiling, floor, wall, beam, column, window, and door) and furniture classes (table, chair, sofa, bookcase, and board). Furthermore, objects that cannot be identified as any of these are labeled as 'clutter'.
As it is commonly done for segmentation evaluation on this dataset, we regard a 6-fold cross validation over the 6 areas and average the single results [QSMG17a].

ScanNet Equivalent to S3DIS, the dataset ScanNet also contains 3D reconstructed scans from indoor scenes [DCS+17]. The original version includes 1513 rooms in total based on approximately 2.5M RGB-D images. An official split of 1201 for training and 312 for testing is proposed. The dataset has recently been extended by 100 additional scans which hold as test set for the new ScanNet Benchmark Challenge. In this thesis, we will follow the original split in our evaluation to enable a valid comparison to [WYHN18].

Semantic annotations are provided with respect to the 40 labels used in the NYU Depth V2 dataset [NSF12]. Usually, though, only a subset of 20 labels is

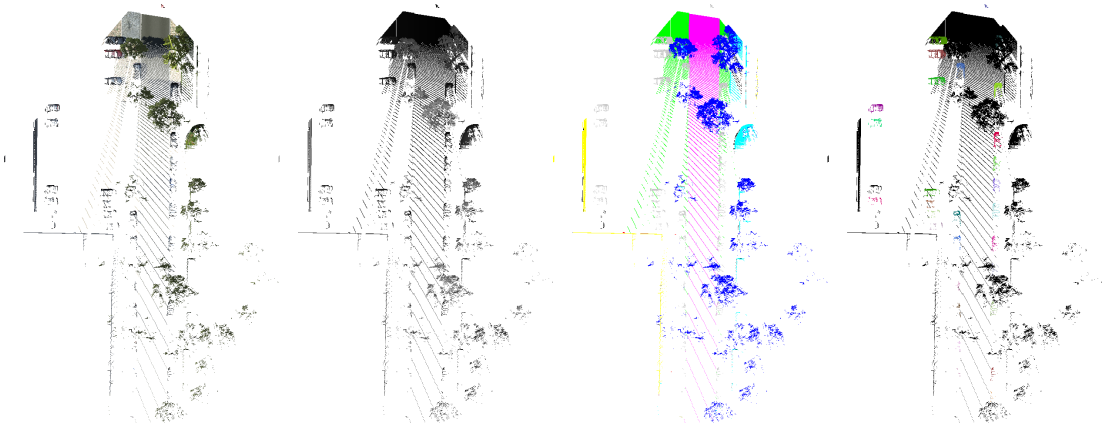


Figure 5.3: **BEV for scene from VKitti dataset.** Left to right: RGB, depth, ground truth semantic segmentation, ground truth instance segmentation for vehicle objects.

considered which is listed in Table 5.9. The classes 'floor' and 'wall' are not taken into account for the instance segmentation task. In contrast to S3DIS, there are regions that remain unannotated.
The dataset is provided by surface mesh files for which we utilize the code used by [QYSG17a] to retrieve the required point cloud format. Furthermore, we also transform the single scenes to axis alignment by making use of the respective matrices that were released in the latest version.

Although we also intended to test our model on an outdoor dataset, we rejected this plan due to lack of appropriate data. To the best of our knowledge, the only potential dataset that offers at least partly object differentiating labels is Virtual Kitti [GWCV16]. However, instance annotations are solely provided for objects of the category 'car' and 'van' which only represent a minor part of the whole point cloud. Furthermore, the density of the points within the whole scene varies significantly. This also heavily influences the bird's eye view rendering as depicted in Figure 5.3. We therefore think that this dataset is not applicable for the current version of our model.

Furthermore, we did not evaluate on NYU Depth V2 [NSF12] as it was done for SGPN. This dataset consists of single RGBD images. These views can be lifted into the 3D space, yielding partial 3D scans. However, this only produces point clouds that model a rather small area. Thus, it does not appear to be of that significance for our approach where we especially want to show that large scale context can be leveraged for instance segmentation on point clouds.

Data Pre-Processing

For our experiments, we consider several steps of pre-processing. Both 2D BEV-FN and 3D P-FN of our pipeline pose certain requirements for the input data. For the 3D feature networks like PointNet [QSMG17a] and DGCNN [WSL+18], we deal with single blocks of points from a scene. This network structure is applied in both SGPN and the second stage of 3D-BEVIS (P-FN). For the 2D instance feature extraction network (BEV-FN) in our approach, we feed pre-rendered BEV images into the pipeline.

Due to the immense number of points within a point cloud, we only consider sampled subsets of points for each scene. Final predictions for these smaller point clouds can be easily expanded to the original scenes by applying a k-nearest neighbor proceeding. We generate an adequate subsampling by laying a grid over a whole scene and determining a representative point for each occupied cell. For each cell, we compute the mean of the x, y, z-position as well as the r, g, b-values of all points that are placed within this cell. The semantic and instance labels are set to the combined majority vote. We choose a cell size of 3cm for our experiments.

Regarding the 3D feature network models, we handle the input data similar to the proceeding in [QSMG17a]. Each point cloud corresponding to a room is split up into blocks of fixed area. This is done by using a sliding window that moves along the groundplane. In contrast to the training procedure of Qi et al., we generate new blocks each epoch which vary slightly due to small random shifts of the centroid position. By doing this, we get additional data augmentation. Moreover, we consider cylindrical blocks. We choose the base area such that it surrounds a square with a side length of 1m for S3DIS and 1.5m for ScanNet. Within each block, a fixed number of points is uniformly sampled. Each point is represented as a 9-dimensional feature vector, including x, y, z-coordinates, r, g, b-color values as well as normalized spatial coordinates x', y', z'. As it is sufficient during training, we only predict the labels for the sampled subset of points and consider the result for each block individually. For testing, we derive labels from neglected points by applying a voting among the k nearest points that obtained a prediction. Moreover, we need to merge the results within overlapping areas. For the semantic segmentation scores, this can simply be done by averaging over the softmax score predictions corresponding to a point. We also apply this technique to our approach for merging a point's instance feature. However, as the similarity features from SGPN are highly block dependent, it is not possible to proceed with them in the same way. Thus, a special block merging algorithm was proposed which we presented in Section 3.3.3.

We refer the reader to Section 4.1 for a general description of generating bird's-eye views from point clouds. The resolution is chosen to be 3cm for the S3DIS dataset and 5cm for ScanNet scenes. Particular attention needs to be paid to

the ceiling of the rooms. Regarding the S3DIS dataset, we have a full ceiling for every room. As this is clearly meaningless in the BEV image, we ignore the highest points within a range of 2.5cm for each cell during the rendering process. By doing this, we are able to deal with noisy points along the ceiling as well as several ceiling elements at different heights in a single room. The scenes from ScanNet are more irregular concerning this aspect. Only some of the rooms contain ceiling components and the represented ceiling is not always complete. We therefore decided to remove all points that are within the highest ten percent of each room. Another problem occurs from the fact that the rooms vary a lot in their size. This leads to different sizes along the BEV images as well. However, for the training, fixed size input data is required. We therefore choose the dimensions regarding the mean of the height and width along the training pictures. For an in-between evaluation on the testing data as well as the final instance feature prediction for the subsequent stage of the pipeline, the whole BEV images are taken into account. During training, we add data augmentation by randomly scaling the image within a small range and flipping it along either of the axes. To obtain images of fixed size, we either randomly crop a fitting section or place the view in an otherwise empty image.

5.2 Evaluation Metrics

The task of instance segmentation on point clouds is largely unexplored so far. Therefore, there is no generally established metric for evaluation yet. To the best of our knowledge, the only other approach on instance segmentation on point clouds is SGPN [WYHN18]. As we want to compare our method with the results achieved by Wang et al., we will follow their used evaluation scheme to allow a fair comparison. However, we also look into shortcomings and possible alternatives. In the following, we present the criterion used for our baseline approach. Moreover, we will later provide an outlook at the evaluation scheme used in the recently published ScanNet Benchmark Challenge [DCS+17].

Wang et al. first decide for each detected instance whether it has a valid counterpart in the ground truth segmentation and thus, labeling it as TP_{inst} (*true positiv*) or FP_{inst} (*false positive*) on the instance level. An instance proposal is marked as TP_{inst} if it has an high overlap with any ground truth object. To formalize this, the intersection over union (IoU) is computed between a predicted instance proposal and each ground truth group. In general, the IoU is defined as

$$IoU = \frac{TP_{pxl}}{TP_{pxl} + FP_{pxl} + FN_{pxl}}. \tag{5.1}$$

Here, TP_{pxl}, FP_{pxl}, and FN_{pxl} are defined on the pixel-level. TP_{pxl} indicates the number of pixels which lie in the overlapping section of both point groups,

whereas FP_{pxl}, FN_{pxl} count those points that only belong to either the predicted or the ground truth instance. In case that the IoU to any ground truth object is greater than a threshold τ, the predicted instance is considered to be a true positive. An additional restriction which matters for threshold smaller than 0.5 is that each ground truth instance can only be associated to a single predicted proposal. Possible other predicted objects that would get assigned to the same ground truth instance are regarded as false positive.
From this, we can determine the *precision* and *recall* for a specific threshold τ with:

$$\begin{aligned} Prec^{\tau} &= \frac{TP^{\tau}_{inst}}{TP^{\tau}_{inst} + FP^{\tau}_{inst}} \\ Rec^{\tau} &= \frac{TP^{\tau}_{inst}}{TP^{\tau}_{inst} + FN^{\tau}_{inst}} \end{aligned} \tag{5.2}$$

Wang et al. propose to calculate the *average precision* (AP) for measuring the performance of their method. They do not make it clear how exactly they define this score. In general, the AP is the average over precision scores for a ranked output and can be interpreted as the area under the precision-recall curve [HAGM14]. In our case, however, there is no ranked output as we get a single instance id for every pixel and final scores are regarded for different IoU thresholds independently. Thus, a direct transfer to our problem is not obvious. We therefore decide to interpret the stated term 'AP' as the average over precision scores for the different categories.

Class associations are set to each predicted instance with respect to a majority voting of the predicted semantic labels among the corresponding pixels. It should be further noted that, in this setting, instance predictions with a sufficient IoU to a ground truth object are always counted as true positives even if their inferred semantic label differs from the correct category.
Wang et al. present detailed category-wise results for the AP based on IoU with a threshold of 0.5. Besides this, they also report the mean AP for the thresholds 0.25 and 0.75. Predicted instances classified as 'clutter' are ignored when computing the mean AP. We will follow this setting in our evaluation.

We could observe several shortcomings when using the proposed metric in our experiments.
To start with, the way the semantic prediction influences the result leaves room for discussion. Firstly, an object might be labeled as true positive, even if its predicted class is not correct. Secondly, objects affect the AP of the category they were assigned to with respect to the prediction. This means our prediction might add a bias to the final mean over all classes. Thirdly, disproportionately large object instances might absorb smaller objects without a negative effect on the AP score.
Regarding the first aspect, this does not seem unreasonable to us. Both SGPN

as well as our 3D-BEVIS depend on a strong backbone point cloud network for inferring semantic labels. Hence, the main focus of both approaches is not to improve the learned semantic features. One could now think of a well segmented object that is wrongly categorized, for example, as a chair instead of a sofa. We think it is meaningful to concentrate on the instance segmentation aspect only. Thus, we will follow the setting by Wang et al.
It is woth to consider, however, whether the obtained instance segmentation can be used to improve the semantic prediction. For this, each point gets assigned the label that is associated with the object it belongs to. We will evaluate on both the original semantic predictions as well as the segmentation resulting after taking instance correspondences into account. For the evaluation of the semantic segmentation, we follow the evaluation in [QSMG17a] and report the mean IoU (mIoU) over all classes as well as the overall accuracy (oA).
The second problem is actually caused by the setting that a wrong semantic prediction is accepted. This makes the assignment to a category during evaluation ambiguous. In the proposed evaluation scheme, an inferred instance increases the number of true or false positives of its predicted class and thus affects the final score of this category. Referring to the previous stated example, one can think of a model that is able to perfectly segment chair objects but is weak for sofa instances. However, while labeling chairs correctly, some sofas are wrongly predicted to be chairs as well. This results in a misleading decrease of the AP score for the chairs category. A possible alternative would be, to use the majority vote based on the ground truth labels of the points within a predicted instance. This would also help to detect the categories where our model is weak.
Finally, the third problem shows that our AP metric might not be very informative by its own as it is biased by methods that mainly segment large objects. This affects the case where the whole point cloud is segmented into a small number of instances. Each segment corresponds to a big object in a scene but besides the correct points, it also contains points from smaller objects. For example, all points belonging to a big table might be grouped together but points from a small table next to it are also incorrectly included. Depending on the threshold for the IoU criterion, the large object might still be considered as correct whereas missing small objects have no negative effect on the score. We therefore propose to also take the averaged recall into account. This measures the ratio of found correct instances respective to the total number of objects. To get valid results, we categorize instances according to the ground truth based majority vote.

The evaluation scheme used in the recently published ScanNet Benchmark Challenge [DCS+17] differs in several aspects. Contrary to our approach, where each point is assigned to exactly one object, a list of instance proposals that are detected in the scene is expected. Each of this proposals is determined by the points that belong to it, the predicted class of the object and a confidence score. This means in particular that a single point might be associated to several

proposals. As the given instances can be ranked according to the confidence score, they are able to determine the AP in its original meaning from the precision-recall curve. Furthermore, only those instance proposals are marked as true positives whose matching ground truth object's label is equal to the predicted class. All in all, it is more related to the evaluation criteria used in popular 2D instance segmentation tasks like PascalVOC [EVGW+10], COCO [LMB+14], and CityScape [COR+15] which are based on common metrics for object detection.
This evaluation scheme is not directly transferable to our setting as we do not aim to get several probably overlapping proposal groups. Instead, we get a complete partition of the point cloud which makes our approach more related to the task of semantic segmentation. Deciding for confidence scores does not seem reasonable. However, we will follow the proceeding in [DCS+17] regarding the 'floor' and 'wall' category in the ScanNet dataset and exclude them from our averaged score.

5.3 SGPN

We consider SGPN from [WYHN18] as our baseline. As described in Section 3.3, this approach aims to partition a point cloud into the individual present objects. Their outcome format matches the result of our own framework and is therefore an obvious choice for our baseline. An implementation of SGPN was published recently by Wang [Wan18a]. However, we decided to put our own version into practice. For one reason, the code was not available when we started this thesis. Another intention was to check how easily the results presented in the paper could be reproduced. Furthermore, we tested the influence of the superior 3D feature extraction networks PointCloud++ [QYSG17a] and DGCNN [WSL+18] which we applied as an alternative to PointNet [QSMG17a] as used in the publication. We implemented our own version of SGPN with Tensorflow following the description in [WYHN18].
In this section, we will outline the implementation setting, including the training proceeding, used hyper-parameters, and the derivation of the various required thresholds.

5.3.1 Implementation Details

As described in Section 3.3, the SGPN framework from [WYHN18] is arranged to have a point cloud embedding network at the beginning whose resulting point features are then used for the three subtasks of similarity measuring, confidence estimation and semantic labeling. We evaluate on three different variants of the SGPN framework which differ in the used point cloud feature networks. The examined networks are PointNet [QSMG17a], PointNet++ [QYSG17a] and DGCNN [WSL+18]. For our implementation, we consider the network architecture of the respective semantic segmentation model up to the second last layer. Independently from the chosen model variant, the resulting features are passed to

each of the three branches where they are forwarded through a single PointNet-layer.

The general training setting when applying PointNet as the 3D feature extraction network is merely adapted from [QSMG17a]. Each point of the input point cloud is characterized by 9 feature channels and a point cloud is split into blocks as described in Section 5.1. We sample 4096 points in each block both at train and test time. During testing, we apply k-nearest neighbor on all points that were not sampled.
As stated in [Wan18a], we pretrain the PointNet-based feature extraction model with a large batch size. More precisely, we choose a batch size of 32 and use the same learning rate as described before. This training is performed for 50 epochs. Afterwards, the training of the complete SGPN model is started using only L_{SIM} for the first 5 epochs. The batch size is decreased to 4 which is in our opinion due to the highly increased memory usage resulting from the similarity matrix. The model is trained for 250 epochs.
Both PointNet and SGPN are trained using the ADAM optimizer [KB15]. At the beginning, the learning rate is set to 0.0005 and divided by 2 every 20 epochs. Regarding the batch normalization, the decay rate is initialized with 0.5 and incrementally increased to 0.99.
According to [WYHN18], the partial loss functions L_{sim}, L_{cf}, L_{sem} are summed up without any weighting. However, we applied a weighting factor of 10 for both L_{cf}, L_{sem} to have the single loss values within a similar magnitude.
Wang et al. provide the specific value for several of their hyper-parameters. For the similarity loss function from Equation 3.16, the constants K_1, K_2 are set to 10 and 80 respectively. The value for α is initially set to 2 and increased by 2 every 5 epochs. By doing this, we force the network over time to concentrate more on separating features of different instances but equal classes. The threshold for the confidence map Th_c is set to 0.1, the minimum number of points in a proposed group Th_{num} is set to 200, and proposals are merged if they have an IoU of at least $Th_{iou} = 0.6$.
We also tested the training with slightly different parameter settings for the weighting of the loss functions, the number of epochs where only L_{sim} is used and the constant K_1. However, as we could not notice a significant improvement of the results and the training process, we remained with the proposed values.

There is a number of thresholds which are estimated based on a validation set. As Wang et al. did not specify how they chose this subset from the training set, we simply sampled a small number of rooms from the training data. We ensured that each class is present across these scenes and took at least one room from each training area.

We use the same setting for training the DGCNN model as it was done in [WSL+18]. However, following [QYSG17a] for PointNet++, we only use the points' coordinates as input. Our implementation adapts available code for PointNet [QSMG17b], PointNet++ [QYSG17b], and DGCNN [Wan18b].

Thresholds

There is a number of threshold parameters for which the authors do not provide any fixed values. On the one hand, we need to determine Th_{S_c} to get point-wise proposals from the similarity matrix. On the other hand, category depending thresholds Th_{num_c} are used during a post-processing step to eliminate proposals that are too small. Both kinds of parameters are estimated for each category separately based on the validation set.

Wang et al. state that they use 'per-category histogram thresholding' [WYHN18]. Hence, we computed category-wise histograms based on the pairwise similarity between points and their correlation as depicted in Figure 5.4. We let h_{c_n} denote the number of point pairs from the same object of class c whose similarity $S_{i,j}$ falls into the n-th bin. Analogously, $\overline{h_{c_n}}$ describes the histogram for point pairs related to different instances. As no further information was available, we decided for two distinct criteria to determine suitable thresholds.
First, we considered the F-measure which is the weighted harmonic mean of precision and recall:

$$F_\beta = (1+\beta^2)\cdot \frac{Prec \cdot Rec}{\beta^2 \cdot Prec + Rec} \tag{5.3}$$

where β is used for prioritizing either precision or recall. In the context of our task, for a class c, different thresholds τ and corresponding $F^\tau_{\beta_c}$, we set

$$Th_{S,\beta_c} = \underset{\tau}{\operatorname{argmax}}\, F^\tau_{\beta_c}. \tag{5.4}$$

We compute thresholds Th_{S,β_c} for each class c and $\beta = \{1.0, 0.75, 0.5\}$.

The second option that we take into account is the intersection of h and $\overline{h}$. That means

$$Th_{S,intersect} = \underset{n}{\operatorname{argmax}}\, h_n, \quad \text{where } h_i < \overline{h_i}. \tag{5.5}$$

Based on experimental results presented in Table 5.1 we decided for threshold $Th_{S,\beta=0.5}$ for the successive part of the evaluation.

The second set of threshold Th_{num_c} is used for validating the results from the proposal merging step of Algorithm 1 based on the number of points within a proposal. The thresholds are related to the mean number of points over all instances from a class:

$$Th_{num_c} = r\cdot \frac{1}{\mid \mathcal{I}_c \mid} \sum_{I\in\mathcal{I}_c} [\text{number of points in } I] \tag{5.6}$$

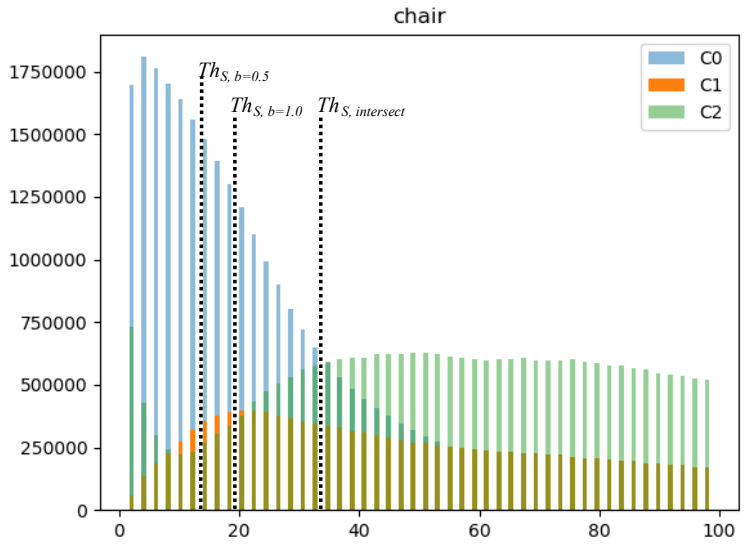


Figure 5.4: **Estimation of similarity thresholds Th_S based on class-wise histograms.** The depicted histogram corresponds to points of chair objects from the S3DIS dataset. For each point belonging to a chair, the distance in the feature space to points of the same object as well as all other points is regarded. The different resulting thresholds $Th_{S,intersect}$, $Th_{S,\beta=1.0}$, and $Th_{S,\beta=1.0}$ are shown as well.

Table 5.1: **Evaluation results for different similarity thresholds Th_S used in the Group Merging algorithm** (Section 3.3.2). Experiments where performed using the basic SGPN implementation with PointNet on area 6 of the S3DIS dataset. Whereas the AP is relatively consistent, we notice an increase of the AR score when enhancing on precision for the F-measure. However, smaller β showed significantly worse results.

		$AP_{0.5}$	$AR_{0,5}$
SGPN, PointNet	$Th_{s,\beta=1}$	40.51	37.56
	$Th_{s,\beta=0,75}$	41.12	41.65
	$Th_{s,\beta=0.5}$	41.16	45.45
	$Th_{s,intersect}$	40.94	42.47

with $r \ll 1$. For our experiments, we set $r = 0.25$. As before, we use the validation set to estimate Th_{num_c}.

Notes about the original code [Wan18a] There is a number of aspects that we noted when we had a closer look on the implementation by Wang and compare

it to our solution. First of all, they still do not provide any information about the choice of the validation set. Moreover, even as they make a pretrained model available, they do not state for which test area they trained it. Furthermore, we were not able to understand their functions about determining thresholds Th_S and computing the evaluation scores. Lastly, their computation of the IoU for the semantic segmentation evaluation differs slightly from the calculation of Qi et al. [QSMG17b] to which they compare their scores. Whereas Qi et al. compute the number of true positive, false positive and false negative point predictions over all rooms before computing the IoU, Wang computed the IoU score for each room separately before averaging over all rooms. We tested both versions in our 3D-BEVIS, PointNet implementation and measured discrepancies of up to 9.4% (27.84% using PointNet evaluation scheme, 37.24% for the scheme used by Wang on area 2) on single areas in favor for the proceeding of Wang.

5.4 3D-BEVIS

In this section, we present an ablation study to justify the design of the network architectures as well as the choices of parameters for the separate parts of our 3D-BEVIS network. Furthermore, we provide further details about our implementation. For this, we sequentially regard the three different parts of the framework, namely BEV-FN, P-FN, and the final clustering stage.

5.4.1 2D BEV-FN

As stated previously, we were motivated to keep the number of trainable parameters in our network limited due to the small set of training data. For this reason, we took inspiration from U-Net [RFB15] which was designed for learning from a small amount of data for the task of 2D semantic segmentation.

The basic network architecture is described in Section 4.1. In our primary version, we consider a sequence of four convolutional blocks followed by two additional convolutional layers. This results in an intermediate representation where the spatial size is decreased to an eighth of the input dimension and the number of feature channels equals 128.
Regarding the instance loss, we use the L2 loss from Equation 3.6 for our basic BEV-FN version with margin $\delta_{dist} = 10$. The influence of the exponential loss variant from Equation 3.8 and the cosine similiarity ($\alpha = 0.8$) measurement introduced in Equation 3.9 are discussed in the following ablation study.

We tested several variations of the proposed network architecture. The effects of these settings are tested on area six of the S3DIS dataset. Training was executed for 2000 epochs. We measure the performance considering the AP and AR with respect to a threshold of 0.5 for the intersection over union. To obtain

Table 5.2: **Ablation study for BEV-FN.** We evaluate different versions of our 2D instance feature network on the sixth area of the S3DIS dataset. We compare their performance regarding the AP and AR with an IoU threshold of 0.5, the semantic IoU and two control loss functions. Unless otherwise stated, we set $F_g = 32$ and use the L2 loss. The setting used for further evaluation are marked in bold letters.

		$AP_{0.5}$	$AR_{0.5}$	IoU (Sem.)	$l_{discr,var}$	$l_{discr,dist}$
BEV-FN	$F_{inst} = 4$	43.78	37.68	34.63	0.67	5.94
	$F_{inst} = 8$	40.93	36.3	**38.65**	0.62	5.59
	$F_{inst} = 16$	43.22	35.6	36.16	**0.6**	5.93
	$F_{inst} = 32, F_g = 64$	40.63	37.46	35.84	0.67	5.6
BEV-FN, without L_{sem}	$F_{inst} = 8$	20.08	34.64		0.69	5.6
	$F_{inst} = 16$	27.86	36.86		0.64	5.4
BEV-FN, $F_g = 128$	$F_{inst} = 8$	35.31	41.94	32.38	0.59	5.18
	$F_{inst} = 16$	43.53	38.86	40.96	0.62	5.57
BEV-FN, deep	$F_{inst} = 8$	**46,08**	**41.25**	35.7	0.61	**5.35**
	$F_{inst} = 16$	45.91	40.18	36.12	0.56	5.36
BEV-FN, deep, $F_g = 128$	$F_{inst} = 8$	42.24	41.48	31.41	0.59	5,14
	$F_{inst} = 16$	48.59	43.26	41.01	0.61	4.69
BEV-FN, cos-loss	$F_{inst} = 16$	29.92	35.17	39.12	0.17	71.44
BEV-FN, exp-loss	$F_{inst} = 16$	41.17	30.59	35.58	0.57	9.3

instances which are required for calculating these scores, we apply Mean Shift as this method is also used for the final output later. However, these scores have to be treated with caution since they depend on an additional clustering algorithm. Thus, we also have a look on the semantic IoU and the outcomes of the alternative discriminative loss functions from Equation 3.11. Detailed results can be found in Table 5.2. We found that for some configurations there were high variances regarding the results from several training runs with the same setting. Thus, we focused on the settings which proved to be more consistent even if scores were a little bit lower.

Firstly, we compared the impact of different feature dimensions. Following the assessment in [ND16], we decided to use a selection of small features numbers $F_{inst} \in \{4, 8, 16, 32\}$. We set $F_g = 32$ for $F_{inst} = 4, 8, 16$. As F_g contains information for both the instance and semantic segmentation task, we expect that F_g should reasonably be higher than F_{inst}. Thus, for $F_{inst} = 32$ we set $F_g = 64$. The resulting scores are around the same magnitude. However, the network for $F_{inst} = 32$ showed a much faster tendency to overfit. The usage of $F_{inst} = 4$ showed most variations over several training procedures. Thus, we limit potential number of feature dimensions to 8 and 16.
Next, we tested the usefulness of the semantic segmentation branch by training the network without L_{sem}. The strong decrease of the AP scores justifies the

integration of the semantic labeling.
Furthermore, we tested the impacts of a high dimension for the general features with $F_g = 128$, a deeper version using an additional convolution block, and the combination of both. Again, we needed to make a trade-off between good scores and stable training. The most promising variant to us seemed to be the deeper version of the network with $F_{inst} = 8$. We did not test for an even deeper network as previous evaluation had shown how easily the training results became unstable for an increase of parameters.
Finally, we also tested alternative loss functions on the basic BEV-FN. Here, we only got useful results for $F_{inst} = 16$. We get a relatively low AP score when using the cosine loss function. This might be due as the applied clustering algorithm is based on the Euclidean distance. Although the scores resulting from the usage of the exponential similarity measure are not overwhelming, we take this loss function in consideration in the next step to explore whether it might be more useful in the later steps.

We decided for a subset of four configurations based on which we make further studies for the P-FN in the second part of our framework. The chosen settings are marked in the table with bold letters. By doing this, we want to ensure that the learned features from BEV-FN are suitable for being transferred to the point cloud.

A fixed dimension for the input images is required during training. We choose a multiple of the total down-scaling factor of the encoder network. Thus, we avoid the problem of inconsistent sizes during the upsampling.
During testing, we only consider the output of the instance feature branch. The resulting matrix (W, H, F_{inst}) is used for the input of the following P-FN.

5.4.2 3D P-FN

The second stage for transferring the instance features from the BEV to the point cloud consists of a 3D feature extractor network. This is partly comparable to the usage of PointNet in SGPN which is used to learn the similarity relationships between points. However, in contrast to this approach, we do not aim to learn a new feature embedding. Instead, we want to learn point-wise representations which match to those retrieved from the bird's-eye view.

We decided for DGCNN as our main point cloud feature extractor. This model looks most promising for our intentions as we do not only regard each point on its own to extract a local descriptor but take neighbors with respect to the current stage of the feature space into account. We expect that this makes it easier for points that were projected to the BEV to delegate their inferred features to

Table 5.3: **Confimation of BEV-FN architecture based on results of the following P-FN.** We compare the capability of promising BEV-FN settings from the previous section. As before, we use $AP_{0.5}$, $AR_{0.5}$ and semantic IoU for comparing the performance.

		$AP_{0.5}$	$AR_{0.5}$	IoU (Sem.)
BEV-FN	$F_{inst} = 8$	66.02	38.82	**76.46**
	$F_{inst} = 16$	65.54	36.08	76.32
BEV-FN, deep	$F_{inst} = 8$	**70.78**	**45.34**	76.01
BEV-FN, exp-loss	$F_{inst} = 16$	65.81	34.13	75.92

unregarded points of the same object. Nevertheless, we also present results from a version which utilizes PointNet for this task for comparison.
Similar as was done in SGPN, we apply the respective network until the second last layer. The resulting point features are then passed again to a semantic and an instance feature branch.
We use the same training setting as descripted for SGPN as these have proven to be appropriate for this kind of network.

We trained the model with the outcome of successful settings from the previous section. We report results in Table 5.3. As before, we consider the AP, AR and semantic IoU.

It can be seen that the deeper version of BEV-FN outperforms the other models by a large margin. In the following part of this thesis, we will keep this architecture and refer to it as BEV-FN.

5.4.3 Clustering

We use a Mean Shift clustering algorithm for grouping the points according to the learned instance features. For our implementation, we utilized the Mean Shift function from scikit-package [PVG+11]. The algorithm requires only a single parameter which determines the bandwidth (see Section 3.1.2). Scikit provides a function for estimating this parameter depending on the current point cloud. However, we found out that we get better results using a constant bandwidth of 1.0 along our test set.

In the following part, we examine two aspects for the clustering part of our network.
First, we execute Mean Shift on different feature settings. Besides the actual instance features $P_{inst} = (N, F_{inst})$, we also considered using only the semantic logits $P_{sem.logits} = (N, K)$ to show that P_{inst} hold additional useful information

Table 5.4: **Different settings for clustering points into instances.** We compare results when using different features as well as the impact of the proposed post-processing step.

	$AP_{0.5}$	$AR_{0.5}$
inst	71.25	50.27
inst, post-process	79.87	62.17
inst+sem.logits	65.05	59.03
inst+sem.logits, post-process	**80.82**	**65.05**
sem.logits	33.78	44.75

for instance segmentation. Moreover, we tested whether a point-wise concatenated feature vector $P_{inst+sem.logits} = (N, F_{inst} + K)$ could lead to improvement. This was motivated by the fact that nearby objects often got a relatively similar instance feature. Nevertheless, when considering the category dependent scores instead of the final class labels only, we also take into account that the semantic prediction of some points might be rather uncertain.
Furthermore, we tested the influence of our post-processing step where we split up those instances, in which more than one class is highly represented.

We show results for different configurations in Table 5.4. It can be seen that applying the post-processing split-up boosts the performance of our method significantly for both the AP and the AR measure. Regarding the choice of features, we see superior performance regarding the AP score when considering the instance features only. However, for the combination of instance features and semantic logits, we observed a large improvement for the achieved recall. Overall, the combination of features and a subsequent post-processing of the received segmentation promises the best results.

5.5 Results

We present quantitative and qualitative results for our proposed 3D-BEVIS framework on the popular datasets S3DIS and ScanNet. Furthermore, we compare the achieved performance to those of our SGPN variations as well as the original SGPN scores reported in [WYHN18].

5.5.1 S3DIS

We present scores for each category regarding the AP and AR for IoU with threshold 0.5 in Table 5.5 and Table 5.6 respectively. We will first focus on the AP scores as these were used in [WYHN18] as well. Besides the mean over all categories excluding 'clutter', we also report the mean regarding only the five

Table 5.5: **Category-wise $AP_{0.5}$ on S3DIS.**

	Mean	$Mean_{obj}$	ceiling	floor	wall	beam	column	window	door	table	chair	sofa	bookcase	board
SGPN* [WYHN18]	54.35	39.44	79.44	66.29	**88.77**	**77.98**	60.71	**66.62**	56.75	46.90	40.77	6.38	47.61	11.05
SGPN, PointNet	42.90	36.70	78.15	80.27	48.90	33.65	16.97	49.63	44.48	30.33	52.22	23.12	28.50	28.62
SGPN, PointNet++	47.44	43.82	68.35	72.23	53.53	49.78	18.62	32.72	43.05	43.07	64.92	**54.73**	43.48	24.75
SGPN, DGCNN	58.56	53.09	**85.85**	83.15	61.65	52.82	47.60	55.12	62.22	34.97	66.02	42.50	55.93	54.85
3D-BEVIS, PointNet	51.66	42.78	77.15	93.98	66.23	41.42	41.63	59.75	53.43	41.65	60.52	15.77	42.87	25.48
3D-BEVIS, DGCNN	**65.66**	**61.62**	71.00	**96.70**	79.37	45.10	**64.38**	64.63	**70.15**	**57.22**	**74.22**	47.92	**57.97**	**59.27**

Table 5.6: **Category-wise $AR_{0.5}$ on S3DIS.**

	Mean	$Mean_{obj}$	ceiling	floor	wall	beam	column	window	door	table	chair	sofa	bookcase	board
SGPN, PointNet	46.37	31.73	74.05	91.17	45.68	43.30	33.08	54.17	51.77	31.22	44.62	22.22	23.95	36.65
SGPN, PointNet++	52.39	40.41	67.25	85.67	50.85	**76.68**	42.58	47.38	56.22	46.82	42.02	30.12	37.77	45.33
SGPN, DGCNN	**61.45**	**50.09**	**75.05**	**95.17**	59.50	50.40	**64.40**	**75.32**	**67.10**	44.85	**62.88**	38.28	**44.48**	**59.97**
3D-BEVIS, PointNet	44.60	33.70	64.52	86.77	49.27	44.07	32.08	48.62	41.42	41.32	45.22	38.13	31.38	12.45
3D-BEVIS, DGCNN	54.74	49.12	65.22	87.50	**60.25**	47.73	49.05	51.08	50.45	**53.13**	56.00	**44.25**	42.23	50.00

furniture object classes (table, chair, sofa, bookcase, board) as we consider these to be more interesting for later application tasks.

Regarding the results of the SGPN approach, it can be observed that applying the superior models PointNet++ and DGCNN with respect to the semantic segmentation task also leads to improved performance for instance segmentation. However, we miss the reported scores by Wang et al. by a large margin. This might be due to a bad choice of the various threshold parameters. As we spent a considerable amount of time on testing several parameter settings, this shows that the estimation of these thresholds complicates the reproduction of the proposed approach. Nevertheless, we were able to attain slightly better results when regarding only object categories.
In comparison, our model outperforms SGPN by a large margin. Even when using the much weaker PointNet for transferring features along the point cloud, we nearly reach the scores presented by Wang et al. and significantly exceed the results of our own PointNet-based SGPN model. Considering only the subset of object classes, we already observe a significant improvement compared to the reported scores of SGPN for 3D-BEVIS network when using PointNet. Our final model even nearly doubles the provided scores.

Furthermore, we examined the category-wise AR scores. Here, we can confirm a superior performance when using better 3D point cloud networks. When comparing SGPN to 3D-BEVIS with respect to the used feature model, our framework shows weaker outcomes. However, the differences become less significant or even reverse when considering only real object classes.

We also present AP and AR scores regarding different thresholds for the IoU in Table 5.7. A general dominance of our 3D-BEVIS model can be observed. However, we noticed that the scores for different thresholds vary much more for our implemented models compared to the reported scores by Wang et al.

Table 5.7: **Instance segmentation results for S3DIS regarding different thresholds for IoU**. Metrics are AP(%) and AR(%). Moreover, we present scores for the setting that only semantically correct labeled instances can be count as true positives.

	$AP_{0.25}$	$AP_{0.5}$	$AP_{0.75}$	$AP_{0.5,\text{ correct class}}$	$AR_{0.25}$	$AR_{0.5}$	$AR_{0.75}$
SGPN* [WYHN18]	59.85	54.35	43.09				
SGPN, PointNet	62.47	42.91	23.89		64.03	46.82	25.52
SGPN, PointNet++	64.27	47.44	27.64		**70.05**	52.39	30.85
SGPN, DGCNN	70.73	58.56	39.73		68.00	**61.45**	**42.56**
3D-BEVIS, PointNet	67,29	51,65	27,36	48.40	60,47	44,61	23,62
3D-BEVIS, DGCNN	**78,45**	**65,66**	**46,72**	**62.47**	67.12	54.75	38.81

Table 5.8: **Semantic segmentation evaluation on S3DIS.** Metrics are mean IoU (%), and overall accuracy (%). As our own experiments were conducted on the downsampled point clouds, small variations compared to the original reported scores are expected.

	Mean IoU (Sem.)	oA (Sem.)
SGPN, PointNet	48.27	71.07
3D-BEVIS, PointNet	46,53	76,88
(refinement)	46.01	76.39
PointNet* [QSMG17a]	47.71	78.62
SGPN, PointNet++	57.68	76.21
SGPN, DGCNN	59.29	80.71
3D-BEVIS, DGCNN	58.37	83.69
(refinement)	58.32	83.19
DGCNN* [WSL^+18]	56.1	84.1

In Table 5.8 we present achieved results of the different models when evaluating the semantic segmentation and compare them to the scores in the corresponding original papers. Besides the original point-wise inference of class labels, we also examine a refined version of the segmented point cloud with respect to the predicted instance segmentation. For this, we assign each point the class label of the instance it belongs to.
In contrast to what Wang et al. stated in their work [WYHN18], we cannot observe an overall improvement of the semantic scores. Whereas we see improvement regarding the mean IoU for most cases, the overall accuracy gets worse. No significant differences can be observed between the original segmentation and the refined version.

Table 5.9: **Category-wise $AP_{0.5}$ on ScanNet.**

	Mean	Mean (*SGPN*)	wall	floor	cabi-net	bed	chair	sofa	table	door	win-dow	book-	pic-ture	coun-ter	desk	cur-tain	fridge	shower curtain	toilet	sink	bath-tub	other furniture
3D-BEVIS	**53.80**	**57.73**	**70.30**	**97.00**	29.70	**78.30**	**75.60**	**65.00**	**68.50**	**36.80**	**37.40**	**65.00**	**21.30**	14.50	**37.50**	57.80	**71.40**	**56.40**	**68.10**	**57.40**	**88.90**	38.80
SGPN* [WYHN18]	31.82	35.09	46.90	79.00	**34.10**	43.80	63.60	36.80	40.70	0.00	0.00	22.40	0.00	**26.90**	22.80	**61.10**	24.50	21.70	60.50	35.80	46.20	-

Table 5.10: **Category-wise $AR_{0.5}$ on ScanNet.**

	Mean	Mean (*SGPN*)	wall	floor	cabi-net	bed	chair	sofa	table	door	win-dow	book-	pic-ture	coun-ter	desk	cur-tain	fridge	shower curtain	toilet	sink	bath-tub	other furniture
3D-BEVIS	51.86	54.75	51.90	92.80	37.40	73.60	64.00	75.30	64.30	37.20	23.80	75.30	5.10	19.20	47.20	44.30	61.40	75.00	82.50	35.70	74.20	38.00

We present qualitative results for both instance and semantic segmentation in Figure 5.5. Different instances are distinguished by their color. The color of a predicted object does not necessarily match to the color of the corresponding instance in the ground truth segmentation. This results from the arbitrary order of the instance indices. Due to the random color mapping and the limited number of colors, it might happen that two close objects are assigned the same color. However, especially for the ground truth representation but also for large objects correctly labeled with respect to their category, it holds that objects belonging to different classes are segmented as different objects.

Furthermore, we show intermediate 2D instance feature results for the same subset of rooms in Figure 5.6.

5.5.2 ScanNet

We also tested 3D-BEVIS on ScanNet. According to previous reported evaluation for semantic models, this dataset is more challenging compared to S3DIS as it contains more classes and the data is more noisy [TPKZ18].
We present achieved scores for $AP_{0.5}$ in Table 5.9. For comparison, we also list the scores for SGPN from [WYHN18]. For this dataset, Wang et al. applied PointNet++ for feature learning. As for S3DIS, our model outperforms SGPN by a large margin. The resulting mean AP score is significantly lower compared to the one from S3DIS.

For completeness, we again present category-wise results regarding the $AR_{0.5}$ measure (Table 5.10) as well as mean scores for both AP and AR when using different thresholds for IoU (Table 5.11).

A selection of qualitative results is shown in Figure 5.7. We present predicted semantic and instance segmentation as well as a visualization of the predicted instance features.

We also evaluated 3D-BEVIS on the current ScanNet Benchmark Challenge [DCS+18]. For this a new test set of 100 scenes was provided. In Table 5.12,

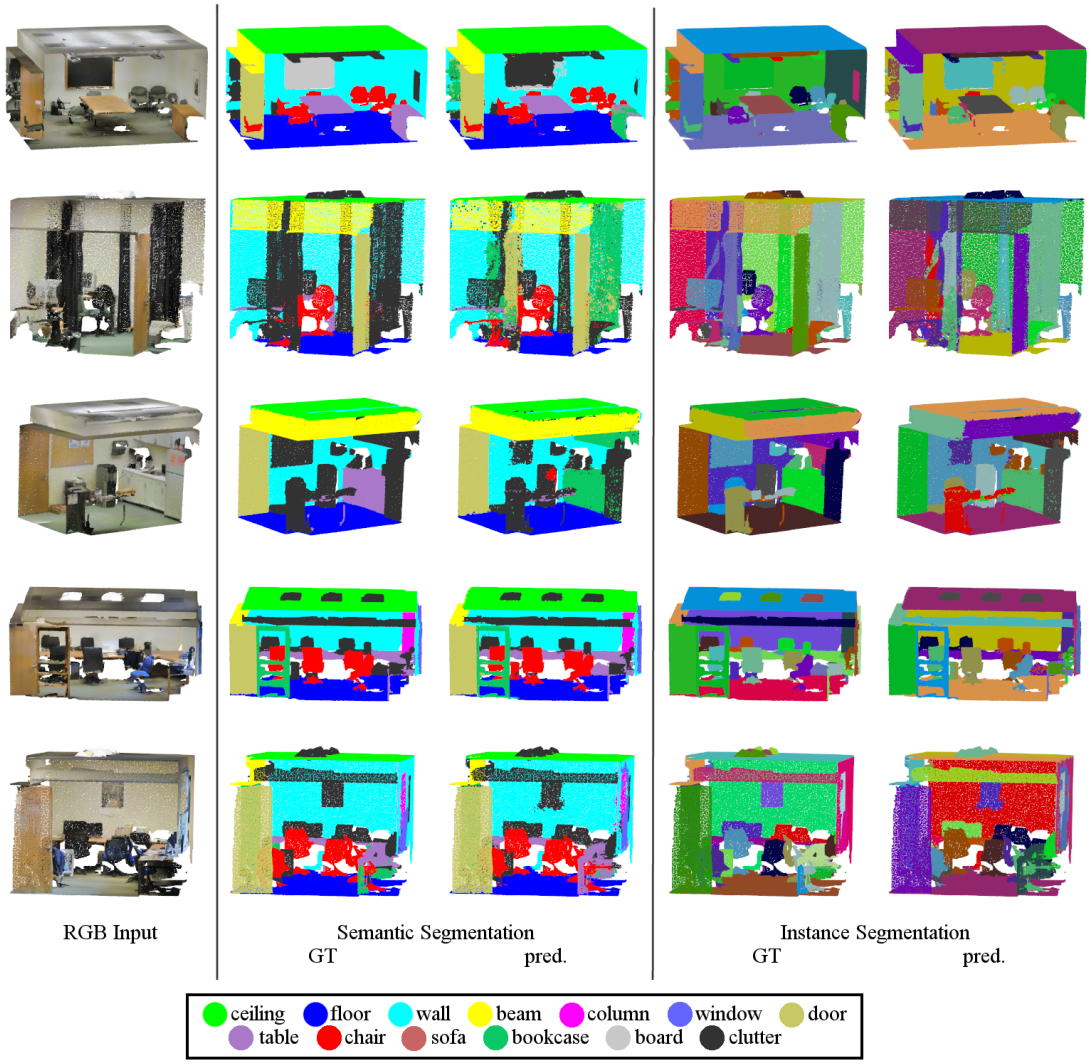


Figure 5.5: **3D-BEVIS results on S3DIS.** Left to right: Input RGB point cloud, semantic segmentation (ground truth, prediction), instance segmentation (ground truth, prediction). While we have a fixed color for each class, the color mapping for the single instances is arbitrary.

we report our received scores and compare them to those of the stated baseline method. The provided baseline approach applies Mask-RCNN on 2D image data and transfers the results onto the 3D point cloud data. Scores are based on the metric of the challenge which differs in some points from our own measure. Thus, a direct comparison to the previous scores is not possible.

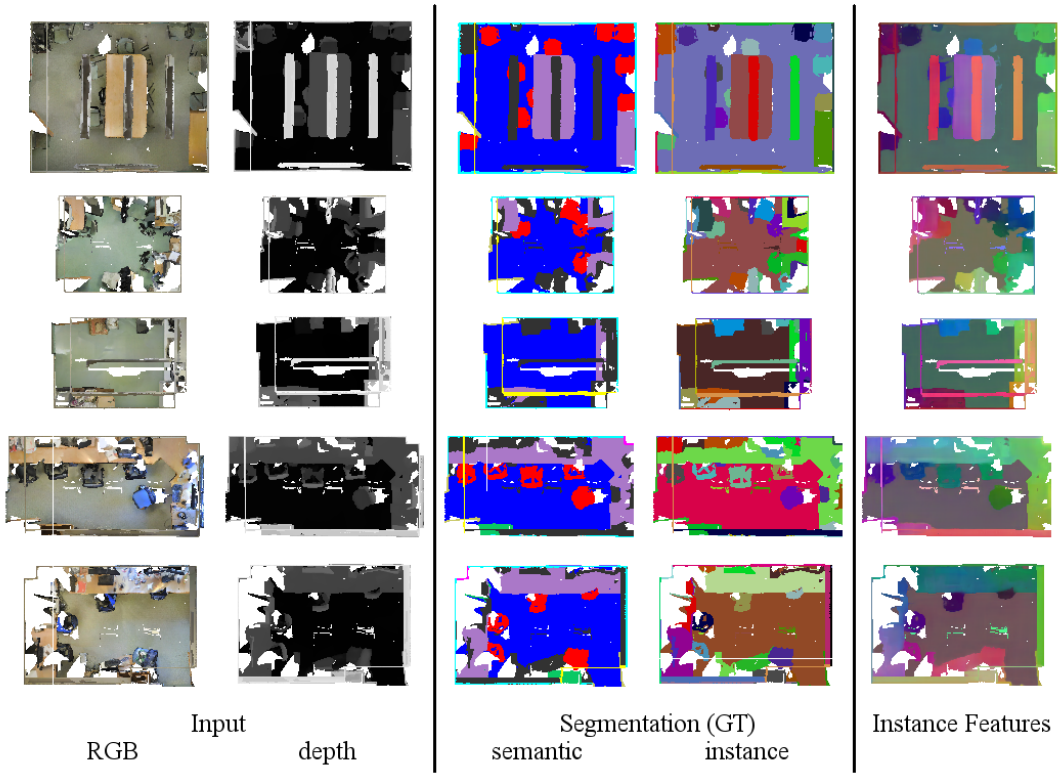


Figure 5.6: **Predicted instance features for 2D BEV.** Left to right: Input RGB and depth images, ground truth semantic and instance segmentation, instance features. Instance features are mapped into RGB space by applying PCA.

Table 5.11: **Instance segmentation results on ScanNet regarding different thresholds for IoU.** Metrics are AP(%) and AR(%). Only the subset of categories with respect to the benchmark challenge is considered.

	$AP_{0.25}$	$AP_{0.5}$	$AP_{0.75}$	$AP_{0.5,\text{ correct class}}$	$AR_{0.25}$	$AR_{0.5}$	$AR_{0.75}$	$AR_{0.5,\text{ correct class}}$
3D-BEVIS	69.50	53.80	23.82	47.71	72.69	51.86	22.50	42.07

Table 5.12: **Benchmark Challenge ScanNet.**

		avg ap	bathtub	bed	bookshelf	cabinet	chair	counter	curtain	desk	door	otherfurniture	picture	refrigerator	shower curt.	sink	sofa	table	toilet	window
AP 0.25	3DBEVIS	0.35	0.556	0.641	0.385	0.112	0.528	0.156	0.176	0.309	0.168	0.079	0.065	0.286	0.4	0.337	0.714	0.28	0.807	0.298
	baseline	0.227	0.85	0.074	0.002	0.191	0.15	0.221	0.103	0.073	0.131	0.147	0.387	0.197	0.143	0.532	0.356	0.117	0.38	0.03
AP 0.5	3DBEVIS	0.225	0.556	0.53	0.074	0.033	0.368	0.026	0.033	0.096	0.084	0.028	0.023	0.095	0.279	0.119	0.575	0.176	0.807	0.153
	baseline	0.053	0.333	0.002	0	0.047	0.002	0.001	0.02	0	0.031	0.021	0.184	0.065	0	0.014	0.107	0.02	0.109	0.004
AP	3DBEVIS	0.11	0.21	0.296	0.02	0.009	0.258	0.006	0.007	0.029	0.033	0.014	0.003	0.035	0.116	0.041	0.37	0.116	0.347	0.063
	baseline	0.021	0.185	0	0	0.014	0	0	0.006	0	0.007	0.005	0.087	0.012	0	0.002	0.027	0.004	0.022	0.001

5.5.3 Discussion

The presented results from the previous section demonstrate superior performance of our 3D-BEVIS model compared to the current state of the art. We showed that we beat SGPN on both S3DIS and ScanNet with respect to their self-chosen evaluation metric.

Furthermore, we compared the interference time when applying the basic SGPN model compared to our 3D-BEVIS model. For this, we measured the total required test time under equal condition on area six of the S3DIS dataset. It took 44 minutes for SGPN to compute the instance segmentation for the 53 rooms. A major time-consumption here is the group merging algorithm where, among other things, an instance proposal is processed for every point and many pairwise comparing operations are executed. It should be noted that the time for computing threshold values under consideration of the validation set is not included here although it takes another considerable amount of time.
In contrast, 3D-BEVIS finishes the same evaluation after 10 minutes (BEV-FN: 2 minutes, T-FN: 8 minutes). The most tedious component for our method is the application of the Mean Shift algorithm for the full point cloud. As the rooms are split up into a total number of 2210 blocks, this results in a average time of 1.19 seconds per block or 49.81 seconds per room for SGPN and 0.22 or 11.32 seconds respectively for 3D-BEVIS.
We suppose that it is possible to speed up the SGPN procedure by considering additional heuristic aspects. However, we show that our model in its basic version is already much faster.

Nevertheless, we also observed some general limitations of our method. Some of these become clear when regarding the visual results (Figure 5.5, 5.7).
One aspect concerns flat objects like walls. These just cover a relatively small area in the BEV and are therefore difficult to distinguish regarding both semantic and instance segmentation. Sometimes, instances do not appear in the rendered image at all. For example, this often holds for boards. In these cases, it is often still possible to segment a board instance due to semantic prediction on the respective point cloud later. However, in case that there are several boards at a single wall instance, there is no chance to distinguish the boards.
Another point is related to high objects like the ventilation system at the ceiling of some example rooms. These are removed from the BEV together with the ceiling. Thus, we never learn instance features for any of the contained points. Similarly, we do not take the case of distinct ceiling instances into consideration. Whereas the model relatively well distinguishes between objects that are scattered around the place, a dense row of similar object leads to difficulties as no boundary can be recognized in the BEV. This problem repeatedly occurs for chair or shelves objects.
Furthermore, our model is not able to handle exceptionally elongated rooms like

hallways. As can be seen in Figure 5.8, it is nearly impossible for our model to distinguish between single wall segments. This is probably due to the fact that these kind of rooms are never seen in full size during training as a fixed image size is used there.
However, we think that many off the named problems are tolerable regarding real world applications. One could for example think of a robot which has to navigate within an indoor environment. There would probably be no benefit for it in distinguishing different parts of the ceiling. Moreover, in some cases, the ground truth instance annotation itself is questionable as in the case of distinct wall elements in hallways.
Overall, this complies with our motivation to put a special focus on the real object categories.

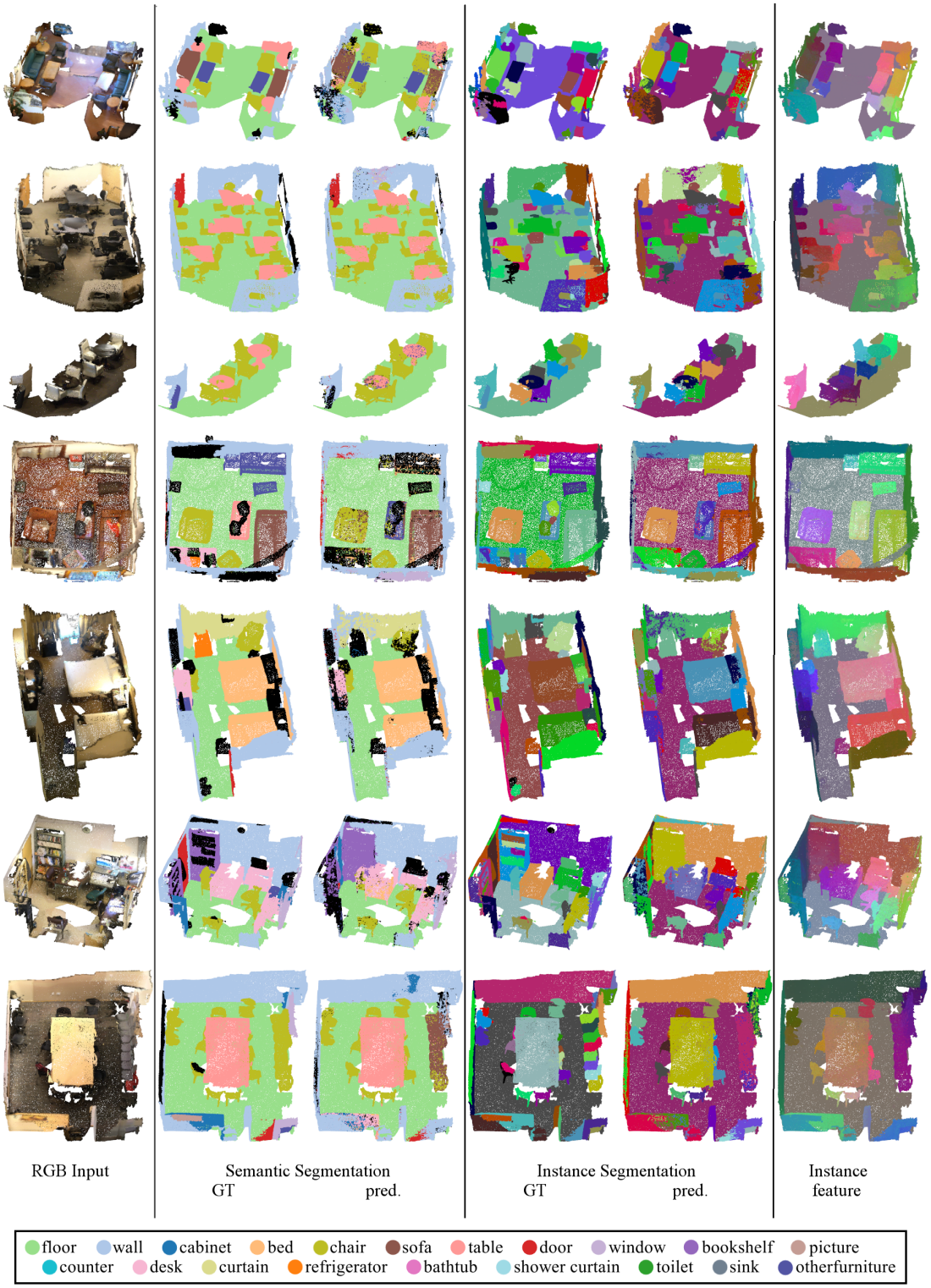


Figure 5.7: **3D-BEVIS results on ScanNet.** Left to right: Input RGB point cloud, semantic segmentation (ground truth, prediction), instance segmentation (ground truth, prediction), predicted instance features.

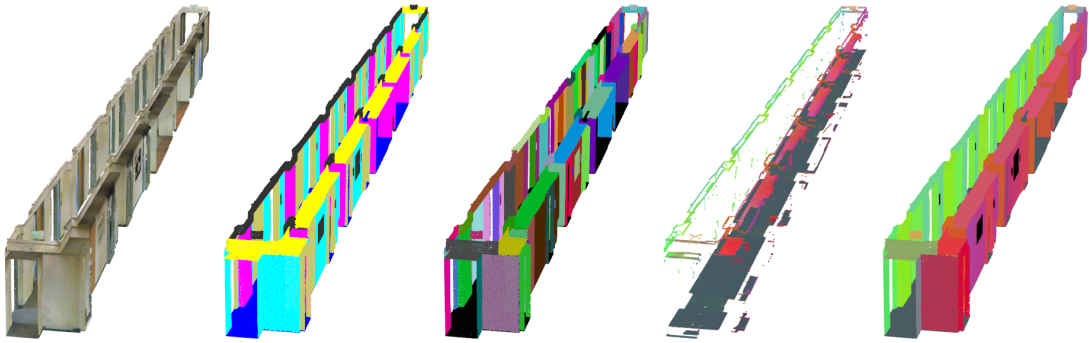


Figure 5.8: **Limitations of our approach.** 3D-BEVIS shows weaknesses regarding long rooms like hallways that have nearly all instances along the wall. For an input point cloud (a) with ground truth semantic segmentation (b), the ground truth instance segmentation contains several wall segments (c). However, the number of points corresponding to these instances which can be seen in the BEV image is relatively small, thus it is difficult to obtain good instance features for these (d). As a result, we are able to distinguish between a single left and right wall but cannot differentiate the single segments (e).

6 Conclusion

In this thesis, we presented 3D-BEVIS which is a novel framework for 3D instance semantic segmentation on point clouds. Our model learns point-wise features, which are used to obtain an instance segmentation by applying a simple clustering algorithm.
To overcome the difficulty of many current state-of-the-art 3D deep learning models for point clouds, which consider only blocks of points separately from each other, we include global context from a bird's-eye view rendering of the entire scene. Regarding this, our model learns a globally consistent feature embedding relevant for instance segmentation which it propagates to the whole point cloud.

We evaluated our method on the two realistic large-scale indoor datasets S3DIS and ScanNet and showed that our model outperforms SGPN which is currently the only other approach for this task. Especially, 3D-BEVIS demonstrates dominant performance when it comes to segmenting instances from real-object categories.

Our approach leaves room for future work: First, we would like to include multiple views with respect to varying axes. For example, we think that a rendering with a projection plane parallel to either the x-z or y-z plane will help to overcome the difficulty arising from flat instances. Next, we plan to include information of finer structures from the point cloud to update the instance feature representation. As the point cloud contains additional information that cannot be observed in the bird's-eye view, e.g. due to occlusion, we expect that this will improve the final instance features. For this, we need to think of ideas for learnable feature extension that do not violate the global consistency of the original instance features. Moreover, we would like to integrate our pipeline into an end-to-end framework, combining the 2D and 3D network.

In conclusion, we are convinced that our proposed approach shows good potential for the considered task and might conduce as a valuable basis for further extensions.

Bibliography

[ASZ+16] Iro Armeni, Ozan Sener, Amir R. Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[BGLSA17] Alexandre Boulch, Joris Guerry, Bertrand Le Saux, and Nicolas Audebert. Snapnet: 3d point cloud semantic labeling with 2d deep segmentation networks. *Computers & Graphics*, 2017.

[BKC15] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *CoRR*, abs/1511.00561, 2015.

[BNG17] Bert De Brabandere, Davy Neven, and Luc Van Gool. Semantic instance segmentation with a discriminative loss function. *CoRR*, abs/1708.02551, 2017.

[CDF+17] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017.

[Chi17] Sasank Chilamkurthy. A 2017 guide to semantic segmentation with deep learning. http://blog.qure.ai/notes/semantic-segmentation-deep-learning-review, 2017. [Online; accessed 7-October-2018].

[CHP+17] Liang-Chieh Chen, Alexander Hermans, George Papandreou, Florian Schroff, Peng Wang, and Hartwig Adam. Masklab: Instance segmentation by refining object detection with semantic and direction features. *CoRR*, abs/1712.04837, 2017.

[CMM02] Dorin Comaniciu, Peter Meer, and Senior Member. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002.

[CMW+16] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. *CoRR*, abs/1611.07759, 2016.

[COR+15] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Scharwächter, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset. In *CVPR Workshop on The Future of Datasets in Vision*, 2015.

[CPK+16] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *CoRR*, abs/1606.00915, 2016.

[CPSA17] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *CoRR*, abs/1706.05587, 2017.

[DCS+17] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[DCS+18] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet benchmark challenge. http://kaldir.vc.in.tum.de/scannet_benchmark/, 2018. [Online; accessed 17-October-2018].

[DHL+16] Jifeng Dai, Kaiming He, Yi Li, Shaoqing Ren, and Jian Sun. Instance-sensitive fully convolutional networks. *arXiv preprint arXiv:1603.08678*, 2016.

[DHS16] Jifeng Dai, Kaiming He, and Jian Sun. Instance-aware semantic segmentation via multi-task network cascades. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[DN18] Angela Dai and Matthias Nießner. 3dmv: Joint 3d-multi-view prediction for 3d semantic scene segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[DRB+18] 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 *The IEEE*

Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[EKHL17] Francis Engelmann, Theodora Kontogianni, Alexander Hermans, and Bastian Leibe. Exploring spatial context for 3d semantic segmentation of point clouds. In *IEEE International Conference on Computer Vision, 3DRMS Workshop, ICCV*, 2017.

[EKSL18] Francis Engelmann, Theodora Kontogianni, Jonas Schult, and Bastian Leibe. Know what your neighbors do: 3d semantic segmentation of point clouds. In *IEEE European Conference on Computer Vision, GMDL Workshop, ECCV*, 2018.

[EVGW+10] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 2010.

[FWR+17] Alireza Fathi, Zbigniew Wojna, Vivek Rathod, Peng Wang, Hyun Oh Song, Sergio Guadarrama, and Kevin P. Murphy. Semantic instance segmentation via deep metric learning. *CoRR*, abs/1703.10277, 2017.

[GLU12] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[GOO+17] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, and José García Rodríguez. A review on deep learning techniques applied to semantic segmentation. *CoRR*, abs/1704.06857, 2017.

[GWCV16] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtualworlds as proxy for multi-object tracking analysis. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[HAGM14] Bharath Hariharan, Pablo Andrés Arbeláez, Ross B. Girshick, and Jitendra Malik. Simultaneous detection and segmentation. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VII*, 2014.

[HFL14] Alexander Hermans, Georgios Floros, and Bastian Leibe. Dense 3D Semantic Mapping of Indoor Scenes from RGB-D Images. In *International Conference on Robotics and Automation*, 2014.

[HGDG17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

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

[HWN18] Qiangui Huang, Weiyue Wang, and Ulrich Neumann. Recurrent slice networks for 3d segmentation on point clouds. *CoRR*, abs/1802.04402, 2018.

[HXKH18] Yen-Chang Hsu, Zheng Xu, Zsolt Kira, and Jiawei Huang. Learning to cluster for proposal-free instance segmentation. *CoRR*, abs/1803.06459, 2018.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[KB15] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015.

[KF17] Shu Kong and Charless C. Fowlkes. Recurrent pixel embedding for instance grouping. *CoRR*, abs/1712.08273, 2017.

[LBSC18] Yangyan Li, Rui Bu, Mingchao Sun, and Baoquan Chen. PointCNN. *CoRR*, abs/1801.07791, 2018.

[Li16] Bo Li. 3d fully convolutional network for vehicle detection in point cloud. *CoRR*, abs/1611.08069, 2016.

[LMB+14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision – ECCV 2014*, 2014.

[LQD+17] Yi Li, Haozhi Qi, Jifeng Dai, Xiangyang Ji, and Yichen Wei. Fully convolutional instance-aware semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[LQQ+18] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. *CoRR*, abs/1803.01534, 2018.

[LS18] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[Lux07] Ulrike Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 2007.

[LWS+15] Xiaodan Liang, Yunchao Wei, Xiaohui Shen, Jianchao Yang, Liang Lin, and Shuicheng Yan. Proposal-free network for instance-level object segmentation. *CoRR*, abs/1509.02636, 2015.

[LYU18] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[MS15] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[NBG+17] Davy Neven, Bert De Brabandere, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Fast scene understanding for autonomous driving. *CoRR*, abs/1708.02550, 2017.

[ND16] Alejandro Newell and Jia Deng. Associative embedding: End-to-end learning for joint detection and grouping. *CoRR*, abs/1611.05424, 2016.

[NSF12] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[PCD15] Pedro O. Pinheiro, Ronan Collobert, and Piotr Dollár. Learning to segment object candidates. In *NIPS*, 2015.

[PLW08] Youngmin Park, Vincent Lepetit, and Woontack Woo. Multiple 3d object tracking for augmented reality. *2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, 2008.

[PSN+18] Christian Payer, Darko Stern, Thomas Neff, Horst Bischof, and Martin Urschler. Instance segmentation and tracking with cosine embeddings and recurrent hourglass networks. *CoRR*, abs/1806.02070, 2018.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research*, 2011.

[QLJ+17] Xiaojuan Qi, Renjie Liao, Jiaya Jia, Sanja Fidler, and Raquel Urtasun. 3d graph neural networks for rgbd semantic segmentation. *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.

[QLW+17] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. *CoRR*, abs/1711.08488, 2017.

[QSMG17a] 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*, 2017.

[QSMG17b] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. https://github.com/charlesq34/pointnet, 2017. [Online; accessed 27-September-2018].

[QYSG17a] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *CoRR*, abs/1706.02413, 2017.

[QYSG17b] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. https://github.com/charlesq34/pointnet2, 2017. [Online; accessed 27-September-2018].

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[RHGS15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc., 2015.

[RMB+08] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Mihai Dolha, and Michael Beetz. Towards 3d point cloud based object maps for household environments. *Robotics and Autonomous Systems*, 2008.

[RUG16] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. *CoRR*, abs/1611.05009, 2016.

[SLD15] Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2015.

[SMAG18] Martin Simon, Stefan Milz, Karl Amende, and Horst-Michael Gross. Complex-yolo: Real-time 3d object detection on point clouds. *CoRR*, abs/1803.06199, 2018.

[SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[TCA+17] Lyne P. Tchapmi, Christopher B. Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese. Segcloud: Semantic segmentation of 3d point clouds. *CoRR*, abs/1710.07563, 2017.

[TPKZ18] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3d. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[UCFB16] Jonas Uhrig, Marius Cordts, Uwe Franke, and Thomas Brox. Pixel-level encoding and depth layering for instance-level semantic labeling. In *German Conference on Pattern Recognition (GCPR)*, 2016.

[Wan18a] Weiyue Wang. Sgpn:similarity group proposal network for 3d point cloud instance segmentation. https://github.com/laughtervv/SGPN, 2018. [Online; accessed 27-September-2018].

[Wan18b] Yue Wang. Dynamic graph cnn for learning on point clouds. https://github.com/WangYueFt/dgcnn, 2018. [Online; accessed 27-September-2018].

[WN18] Weiyue Wang and Ulrich Neumann. Depth-aware CNN for RGB-D segmentation. *CoRR*, abs/1803.06791, 2018.

[WSK+15] Zhirong Wu, S. Song, A. Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[WSL+18] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *CoRR*, abs/1801.07829, 2018.

[WYHN18] Weiyue Wang, Ronald Yu, Qiangui Huang, and Ulrich Neumann. Sgpn: Similarity group proposal network for 3d point cloud instance segmentation. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[YK15] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *CoRR*, abs/1511.07122, 2015.

[YLU18] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[ZG17] Wei Zeng and Theo Gevers. 3dcontextnet: K-d tree guided hierarchical learning of point clouds using local contextual cues. *CoRR*, abs/1711.11379, 2017.

[ZT18] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.