Master of Science Thesis in Electrical Engineering
Department of Electrical Engineering, Linköping University, 2017

Semantic Segmentation of Point Clouds using Deep Learning

Patrik Tosteberg

LINKÖPING
UNIVERSITY

Master of Science Thesis in Electrical Engineering

Semantic Segmentation of Point Clouds using Deep Learning

Patrik Tosteberg

LiTH-ISY-EX--17/5029--SE

Supervisor: **Doktorand Martin Danelljan**
ISY, Linköpings universitet
Doktorand Felix Järemo-Lawin
ISY, Linköpings universitet

Examiner: **Fahad Khan**
ISY, Linköpings universitet

Division of Automatic Control
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping, Sweden

Abstract

In computer vision, it has in recent years become more popular to use point clouds to represent 3D data. To understand what a point cloud contains, methods like semantic segmentation can be used. Semantic segmentation is the problem of segmenting images or point clouds and understanding what the different segments are. An application for semantic segmentation of point clouds are e.g. autonomous driving, where the car needs information about objects in its surrounding.

Our approach to the problem, is to project the point clouds into 2D virtual images using the Katz projection. Then we use pre-trained convolutional neural networks to semantically segment the images. To get the semantically segmented point clouds, we project back the scores from the segmentation into the point cloud. Our approach is evaluated on the semantic3D dataset. We find our method is comparable to state-of-the-art, without any fine-tuning on the Semantic3D dataset.

Contents

Notation

Variables

Notation	
P	Point Cloud
p	Points
C	Camera

1 Introduction

In computer vision, it has become more important to represent data in 3D. In recent years point clouds have become popular to represent 3D data. A point cloud is a set of 3D points $p_i \in R$, which can be created by different kinds of sensors, such as a lidar scanner. A point cloud can also have a RGB value for each point, which gives us a colored point cloud. Today point clouds are commonly used for visualization of 3D objects, 3D maps and for robotics. To distinguish objects in a point cloud, a common method is semantic segmentation.

In computer vision, semantic segmentation is the task of segmenting images or point clouds and to understand what the different segments are. When using semantic segmentation it divides an image or point cloud into semantically meaningful parts and then it semantically label each of the parts into one of the predefined classes. This is very useful in many applications to know what different objects inside a point cloud or image are. In 2D semantic segmentation, it has been shown that convolutional neural network gives good results[4] [20] [21]. In the 3D case it has instead been popular to use a random forest classifier for semantic segmentation of point clouds [19] [8]. Figure 1.1 shows a point cloud and the same point cloud semantically segmented. In this thesis, we want to investigate the use of convolutional neural networks for semantic segmentation of point clouds.

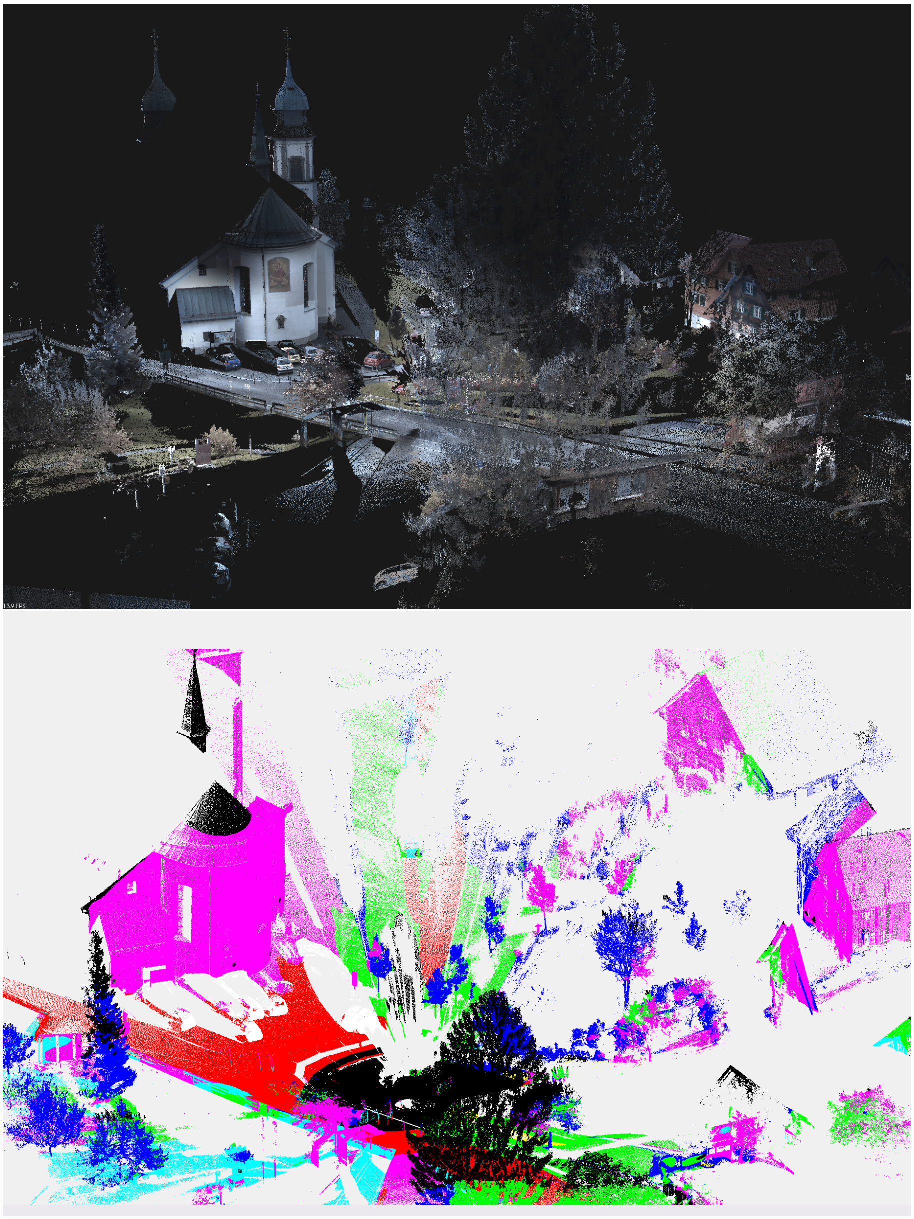

Figure 1.1: *The upper image shows a point cloud with colour, taken with a lidar scanner. The lower image shows the semantic segmentation of the point cloud, run through our pipeline. Red: man-made terrain, green: natural-terrain, blue: high vegetation, purple: buildings, cyan: hard scapes and black: unlabeled/void.*

Semantic segmentation of images or point clouds, is used in different applications today. In robotics, semantic segmentation is for example used to label objects in a robots surroundings. If a robot needs to find a specific object, some kind of object recognition is required. Then semantic labeling is very useful because the robot can then classify and identify the objects in its surrounding. Autonomous driving is also a field where semantic labeling is used. For a vehicle to drive by itself it needs to know what different objects in its surrounding are. One of the most important things for the vehicle to know is where the road is so it can follow it. Another important thing to know is where other vehicles are so it for example can adapt its speed to them or break if necessary. Also in 3D maps semantic labeling is used to visualize objects e.g. buildings, terrain and roads. The semantic labels then gives us a 3D map that is easier to interpret. Another application where semantic segmentation of point clouds is useful, is registration of 3D point clouds. In registration, a rigid transform between two sets of point is calculated to align the two point sets [2].

When performing semantic segmentation on point clouds there will be some more difficulties than in the 2D case. One large difficulty is that there is not as much training data in the 3D case. This is because it is much harder to annotate the datasets in 3D than in 2D. Another challenge is that the point clouds have sparseness between its points, which make it possible to see through objects. This makes it harder to see the structures in the point cloud and to distinguish which structure a point belongs to.

1.1 Problem

The purpose of this thesis is to investigate semantic segmentation of point clouds using pre-trained 2D convolutional neural networks. This is performed by projecting synthetic 2D images from the point clouds and segment them using the convolutional neural networks.

1.2 Motivation

To semantically segment point clouds we choose to project the point clouds into 2D images. This is because in 2D the problem of semantic segmentation has been more explored than in 3D, which means that there are more choices of classifiers to segment the images. Another reason for doing the projection into 2D is that there is abundance of training data in 2D than in 3D. This makes it easier to test on different data and also there are more variety of training data. The choice of using a convolution neural networks (CNN) was done because they have proven to be a good tool for semantic segmentation of images.

1.3 Contributions

In this thesis, we propose a method for semantic segmentation of point clouds. The first step of the method is to project point clouds into virtual 2D RGB images. Then, a pretrained convolution neural networks is used to semantic classify the images. This gives us scores for each class in each pixel in the semantically images. The scores are then projected into the point cloud, which gives us a semantically labeled point cloud. Lastly the labeled point clouds are evaluated using ground truth from the dataset Semantic3D [3].

1.4 Outline

The outline of the report is first a related work chapter with background theory (see chapter 2). Then the proposed method for semantic segmentation of point clouds is described in chapter 3. A brief overview of the implementation is then presented in chapter 4. After this the experiments in the thesis are described and the results is presented in chapter 5. In chapter 6 the discussion and future work are presented. In chapter 7 the conclusion of the thesis is presented.

2 Related Work

This chapter describes related work of the thesis. Section 2.1 is about convolutional neural networks, section 2.2 is about semantic segmentation in 2D and section 2.3 is about semantic segmentation in 3D.

2.1 Convolutional Neural Networks

A CNN [9] [11] is a powerful machine learning method, widely used in computer vision for e.g. classifying images. CNNs are a type of feed forward networks, which means that the information only moves in one direction. A CNN takes input images which are passed through the network and it outputs e.g. a classified image or a pixel-wise classified image. There are different types of layers in a CNN and one of the most important is the convolution layer. This layer consists of multiple learnable filters which can be updated during training. The input of the layer is convoluted with several filters, resulting in a 2D activation. The activation map is then the input to the next layer.

Another important layer is the pooling layer [13]. This layer downsample the input. The most common type of pooling layer is the max pooling layer. The max pooling layers divide the input into small non-overlapping rectangular regions and pick out the maximum value from each region. This layer reduces the spatial size of the image and it also reduces the number of parameters needed to be calculated. This makes it easier to prevent overfitting when training a network. Even though the spatial size is reduced the strong activation remains when a feature has been found. Another common layer is the ReLU layer which uses the activation function $f(x) = max(0, x)$. This layer is used to increase the nonlinear properties of the decision function.

To train the filters in a CNN, a loss-function is used. The loss-function can for example measure the number of wrongly labeled pixels in an image. The loss-function then penalizes all the wrongly predicted pixels with a loss. In the most simple case all wrongly predicted labels are penalized the same. There are different kinds of loss-functions and one of the simplest is the square error seen in the equation below:

$$V(f(\vec{x}), y) = (y - f(\vec{x}))^2 \tag{2.1}$$

y is the true value and $f(\vec{x})$ is the predicted value of an input sample $\vec{x}$. The problem with the square error loss-function is that it tends to penalize outliers more and therefore it takes longer times to converge. Another loss-function more commonly used in deep learning, is the cross entropy loss-function. This function does not penalize outliers too much. For the two-class problem $y = 0, 1$, the cross entropy loss-function is given by:

$$V(f(\vec{x}), y) = -y(ln(f(\vec{x})) - (1 - y)ln(1 - f(\vec{x})) \tag{2.2}$$

During training the loss function is calculated by measuring the error of the output. The network then decides how it will penalize the differences between the predicted labels and the true labels. A common method to train a CNN is back propagation with stochastic gradient decent. This method calculates the gradient of the loss-function and uses this error to recalculate the weights in the filter to minimize the loss-function.[11]

2.2 Semantic Segmentation in 2D

Semantic segmentation [17] in 2D is the problem of finding different objects in images and classify each object into a pre-defined class. In semantic segmentation of images, an image is divided into regions that are classified into one of the pre-defined classes. A common method for semantic segmentation of images in 2D is using convolution neural networks to pixel-wise label images [4] [20] [21]. An example of a semantically segmented image can be seen in figure 2.1.

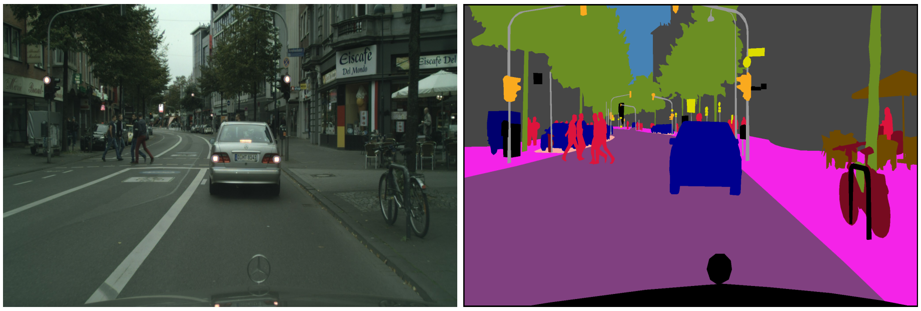

Figure 2.1: *The figure shows an image on the left and on the right, the semantic segmentation of the image.*

2.3 Semantic Segmentation in 3D

In the 3D case of semantic segmentation, the task is to semantically segment point clouds instead of images. There are different approaches to solve the problem and it is popular to use a random forest classifier [19] [8]. A random forest [1] consists of multiple predictions trees. Each tree outputs a predicted class and together all trees vote for the most popular class. In [19] the authors semantically segment point clouds of indoor scenes, by using conditional random fields (CRF). The unary potentials of the CRF is initialized by using the result from a random forest classifier and then the pairwise potentials are learned from training data. Another method [8] use a random forest classifier to semantically segment 3D models of cities. The random forest classifier is trained on light-weight 3D features and is used for an initial labeling of the scene. Then the scene is separated into individual facades by detecting differences in their semantic structures. Finally the authors propose architectural rules that express preferences like alignment and co-occurrence of facades, which helps to improve the results. Another approach is used in [5] where they down-sample the point cloud stepwise to generate a multi-scale neighbourhood. A search structure is then computed for each scale level to fast and easy extract features from neighbourhoods. To semantically segment the point clouds they extract a feature vector and then uses a random forest classifier.

3 Method

In this chapter the proposed method for semantic segmentation of point clouds is described.

3.1 Outline

This section gives a brief overview of the proposed pipeline. The pipeline consist of three parts. The first part is a projection of the point clouds into 2D RGB images, depth images and label images. The second part is using a CNN for semantic segmentation of the projected RGB images. The third part of the pipeline is projecting back the segmented images into the point cloud. An overview of the pipeline can be seen in the figure 3.1.

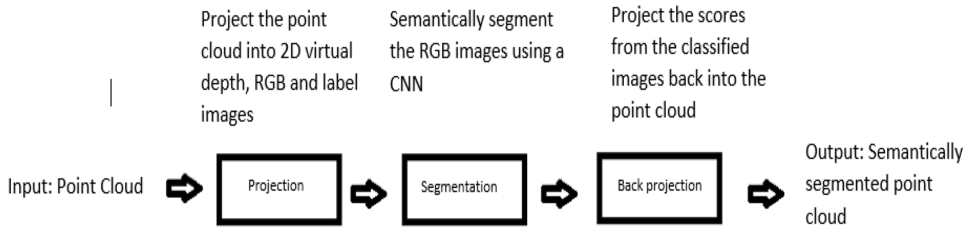
Project the point cloud into 2D virtual depth, RGB and label images
Semantically segment the RGB images using a CNN
Project the scores from the classified images back into the point cloud
Input: Point Cloud
Projection
Segmentation
Back projection
Output: Semantically segmented point cloud

Figure 3.1: *The figure shows an overview of the proposed pipeline for semantic segmentation of point clouds. The first part project the point clouds into virtual 2D images. The second part perform the semantic segmentation of the images. The last part, back projects the semantically segmented images into the point cloud, creating a semantically segmented point cloud.*

3.2 Projection of Point Clouds into 2D

This section describes the projection of the point clouds into virtual 2D images. The projection step outputs an RGB image, a depth image and a label image for each projection.

3.2.1 Defining Camera

The first step in the pipeline was to project the point clouds into virtual 2D images. In our case this step is necessary because we use unorganized point clouds as input and the initial camera is unknown. Because of this we project the point cloud into 2D images. To generate the 2D images we choose to define a camera. Other methods are also possible, such as meshing, but it was too time consuming on the large point clouds. The camera used was defined by an intrinsic and an extrinsic camera matrix. The extrinsic camera matrix is a transformation from world coordinates to camera coordinates. The intrinsic camera matrix instead defines geometric properties of the camera. The pinhole camera is defined by:

$$C = K[R\ T] \tag{3.1}$$

were R is a 3×3 matrix which defines the rotation of the camera. T is a 3×1 matrix which defines the translation of the camera. K is the intrinsic camera matrix which has the following parameters:

$$K = \begin{pmatrix} \alpha_x & \gamma & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.2}$$

where $\alpha_x = f \cdot m_x$ and $\alpha_y = f \cdot m_y$ is the focal length in terms of pixels. These two parameters decide the zoom of the camera. m_x and m_y are scalars that represent the relation between the pixels and the distance and f is the focal length of the camera. γ is the skew coefficient between the x and y-axis. u_0 and v_0 is the principle point of the camera which ideally should be in the center of the image. In our case γ was set to zero and u_0 and v_0 was set to be in the center of the image. This is because we wanted an ideal camera with no skewing factor between the axes.

The camera could be placed anywhere inside the point cloud. In our case the camera was placed in the origin of the point cloud. This is because the point clouds used was taken with a single Lidar scan. The camera would then only project on the areas visible during the scan of the point cloud. In our case the camera was rotated around so it captured most parts of the point cloud. For each rotation, a projection was made. In figure 3.2 projections from two views are shown.

3.2.2 Katz Projection

During the projection, it was important to determine which points were visible from the camera. To solve this, an algorithm called Katz projection [7] was used. The reason for using the Katz projection is that it is a fast method to determine which points in a point cloud are visible. The algorithm uses a hidden point operator that removes all the point that are not visible from the camera. For example if the camera projects a tree inside a point cloud, all points behind it was removed by the Katz projections. The algorithm also removes points that are close to each other to speed up the algorithm. The Katz projection is performed in two steps, inversion and convex hull construction. The method used for the inversion is called spherical flipping. It is performed by defining a sphere with radius R around all the points in the point cloud and a camera C is placed in the origin. The spherical flipping of the points are calculated using following formula:

$$\hat{p_i} = f(p_i) = p_i + 2(R - ||p_i||)\frac{p_i}{||p_i||} \tag{3.3}$$

where R is the radius of the circle, p_i is a point in the point cloud and $||p_i||$ is the norm. This operator will then reflect the points inside the sphere, along a ray from C to p_i, to the image outside the sphere. The points nearest the camera will then be furthest away from the camera after the spherical flipping. The next step of the Katz algorithm is the construction of the convex hull to determine which points are visible. This is because a point p is said to be visible when its transformed point $\hat{p}$ resides in the convex hull.

When it had been determined which points were visible the projection of the 3D points into the pixels was performed. This was done by following formula:

$$\hat{p}_i = Cp_i \tag{3.4}$$

where C is the camera matrix, p_i is the visible points and $\hat{p}_i$ is the projected points. As the points do not always project right onto a pixel, nearest neighbour is used to project the points into the pixels. After this each visible point had its depth from the camera and a weight calculated. This weight determined how much the point contribute in its corresponding pixels. The weights for the points were calculated using splatting [16], which is a method that splat the point into the surrounding pixels. Following Gaussian function was used to calculate the weights:

$$w = \frac{1}{2\pi \cdot 0.25} e^{-0.5 \frac{dist^2}{0.25}} \tag{3.5}$$

where $dist$ is the distance between the pixel and the position of the point.

3.2.3 Mean Shift Clustering

The next problem was to calculate the depth and the RGB values of the pixels. Because it can be more than one point in a pixel, we had to calculated how much each point contributes to the depth and RGB values of a pixel. This was solved by using a mean shift clustering algorithm that calculates a weighted mean of the depth and the RGB, depending on the density of the points. The mean shift clustering algorithm [14] is a non-parametric algorithm that is used for finding the maximum of a density. To get the depth using the mean shift clustering algorithm, following formula is calculated:

$$m(d) = \frac{\sum K(d_i - d)d_i}{\sum K(d_i - d)} \tag{3.6}$$

where $K(d_i - d)$ is a kernel function, d is the current depth estimate and d_i is the depth of the current point being calculated. To calculate the RGB value using the mean shift algorithm, following formula is used:

$$m(rgb) = \frac{\sum K(d_i - d)rgb_i}{\sum K(d_i - d)} \tag{3.7}$$

where rgb is the RGB value for the current point being calculated. In our case is the kernel function defined by:

$$K(d_i - d) = w_k = w \cdot e^{\frac{-0.5dist}{\sigma}} \tag{3.8}$$

where w is the weight for the point calculated in equation 3.5, $dist$ is the difference in depth between a point and current estimate and σ is a variable that defines the width of the kernel.

The last step of the mean shift algorithm is to update the estimate d and this is done by:

$$d \leftarrow m(d) \tag{3.9}$$

The estimate d is updated iteratively until the gradient of the mean shift has converged. When the gradient has converged, the RGB and the depth of the pixel is set to the final estimates $m(rgb)$ and $m(d)$ of the cluster with largest cluster value. Pixels containing no points had the RGB set to black and the depth to zero. An overview for the mean shift clustering and the final calculation of color and depth can be seen in algorithm 1.

3.2.4 Generating Label Images

To evaluate the semantically segmented images, a label image was created for each pose. The Semantic3D dataset (see section 5.1) has labels for each point in the point cloud and we used this dataset for the evaluation. To generate the label images a simple max algorithm was used. Each pixel was labeled to the most frequently occurring label, excluding the label unlabeled, among the points in the pixel. This is because the most frequent occurring label in the point clouds was unlabeled. A pixel was labeled into unlabeled, if unlabeled was the only occurring label among its points. Pixels with no points was labeled to void.

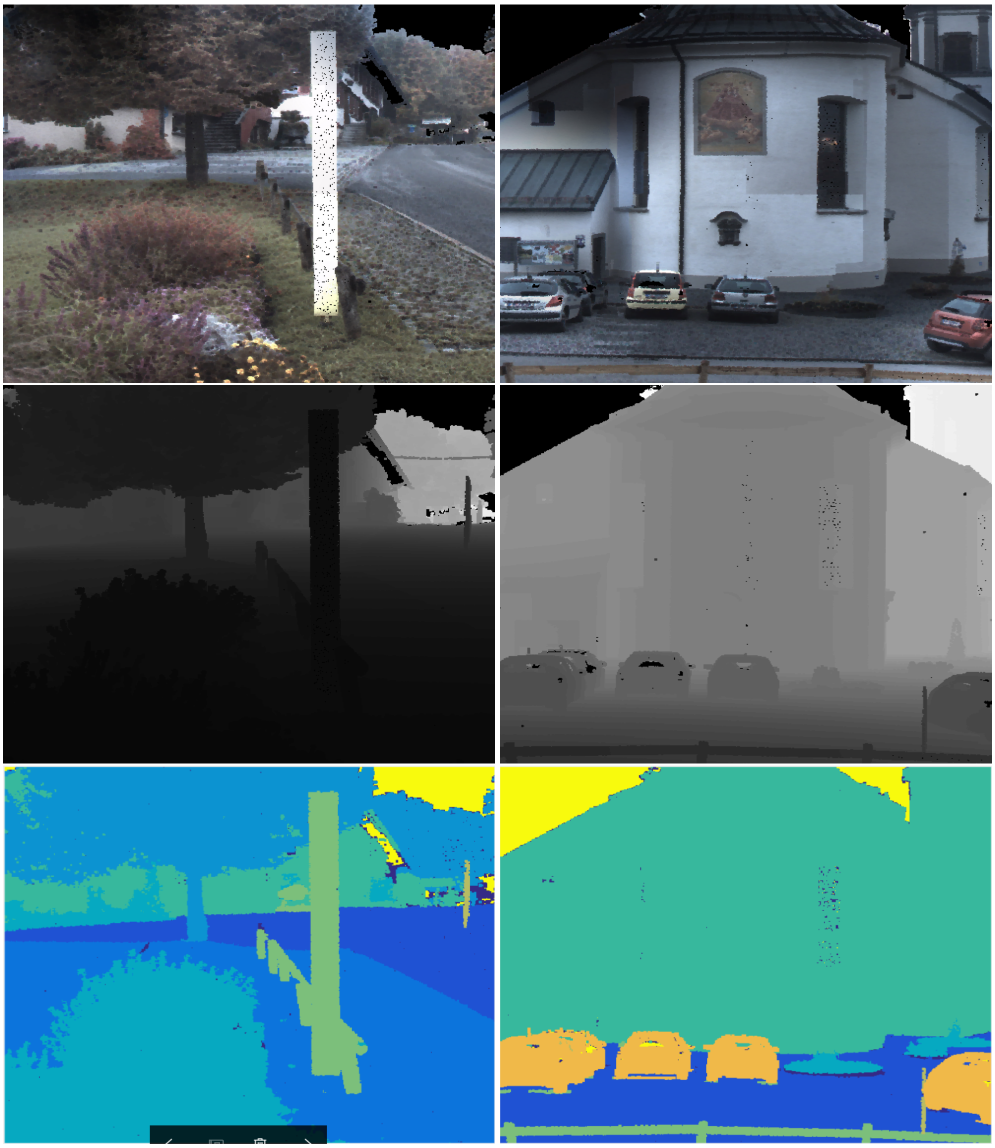

Figure 3.2: *Shows the RGB, the depth image and the label images for two different poses of the bildstein3 point cloud.*

Algorithm 1 : Mean shift clustering

```
Input:
Vector of points { $p_1 ... p_i$ } ▹ each point contains a depth $d$, a $rgb$ and a weight $w$

Output:
Depth value: $d$
RGB value: $rgb$

Initialize:
lim=val                    ▹ set a limit were the gradient is close to zero
$start\_point$             ▹ Vector that store if a point is a start point for the mean shift.
$current\_clustermax$=0

for u=1:i do

If $start\_point(u) == 1$
$md = p_{u,d}$             ▹ start depth for mean shift algorithm for each point

while $diff > lim$ do                                          ▹ Mean Shift

    $md_{old} = md$                    ▹ save old md for calculation off diff
    Calculate $md$ using equation 3.6
    Calculate $mrgb$ using equation 3.7
    $cluster\_val = \sum K(d_i - d)$   ▹ calculate current cluster value
    Update $md$ using equation 3.9
    $diff = md_{old} - md$             ▹ Calculate diff to check lim next iteration

end while

end If

Set $d$ and $rgb$ for the cluster with largest $cluster\_val$
If $cluster\_val > current\_clustermax$

$current\_clustermax = cluster\_val$
$d = md$
$rgb = mrgb$

end If

end for
```

3.3 2D Semantic Segmentation

The next step of the pipeline was to semantically segment the projected RGB images of the point clouds. The semantic segmentation of the images, was performed by a pre-trained CNN. In this thesis two different CNNs were used for the task. This because we wanted to compare how different CNNs affect the result of our method. One of the CNN used was the Laplacian pyramid Reconstruction network [4], which performs pixel wise labeling and is trained on the Cityscapes dataset [12]. The Cityscapes dataset contains sequences of street scenes collected from 50 different cities, with high quality pixel-level annotations. The dataset contains 19 semantic classes, which belongs to 7 different categories of ground, construction, object, nature, sky, human, and vehicle.

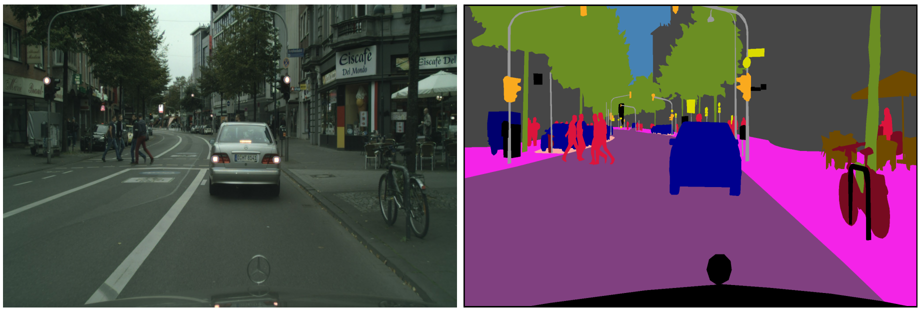

Figure 3.3: *On the left a sample from the Cityscapes dataset is shown. To the right the pixel-wise annotation of the image is shown.*

The Laplacian Pyramid Reconstruction and Refinement (LRR) network defines an architecture of a Laplacian reconstruction pyramid to fuse predictions from high resolution layers with low resolution layers. The LRR-net decomposes the images into disjoint frequency bands, which produces down sampled sub bands. Each sub band is classified and the scores from the sub bands are merged using the LRR architecture. During the reconstruction of the sub bands the CNN uses learned basis-functions. The basis-functions were initialized using PCA and it was trained which basis-functions was most relevant for the task. The CNN finally outputs a score for each pixel when all sub bands has been reconstructed.

The other CNN used was from the paper [21] trained on the ADE20K dataset and performs pixel-wise classification. The ADE20K dataset has 150 classes with a diverse set of scenes. All images in the dataset is fully annotated and many objects are also annotated with their parts.

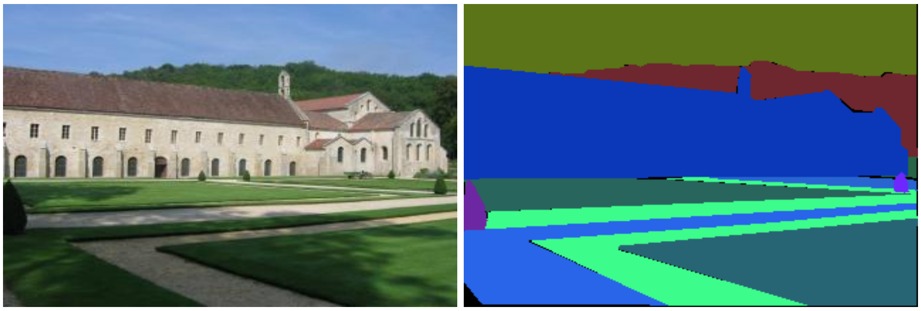

Figure 3.4: *On the left a sample from the ADE20K data set is shown. To the right the images corresponding labels are shown.*

The ADE20K-net are using a design called cascade segmentation module. First are the objects categorized in three large macro classes, stuff (e.g. road, sky, building), foreground (e.g. tree, car, chair) and parts (e.g. car wheel, car door). To recognize the different macro classes, different streams of high-level layers are used. The final segmentation result by fusing all the segmentation from the different streams, which finally outputs scores for each class for each pixel in the image.

The LLR-net and the ADE20K-net were chosen for the semantic segmentation of the images, because these networks had classes suited for Semantic3D and VPS (see section 5.1). Semantic3D and VPS are the two datasets used for evaluation and testing. Both networks were new during the thesis and it had also been proven that the networks had good performance.

3.4 Back projection of scores

The last step of the pipeline is the back projection of the scores from the segmentation into the point cloud. This step outputs two semantically segmented point clouds. One down-sampled to approximately 100000 points, used for experiments, and one full sized point cloud used for evaluation. In the final point clouds the coordinates, the RGB value and the scores for each class were stored for each point. A problem during the projection of the scores into the point cloud, was that a point in the point cloud occurs more than one time in the projected 2D images. To solve this problem, we implemented two different methods, max-pooling of the scores and summation of the scores. An overview for the methods can be seen in algorithm 2 and 3. The max-pooling was chosen because a better score for a class means the network is more confident that it is correct and therefore it chooses the point in the images with the highest score. The summation of the scores was used because it is the most intuitive method to merge the scores from a point.

Algorithm 2 : Max-pooling of scores

```
Input:
Sequence of classified images: { $I_1 ... I_j$}
Point Cloud: $P$
Point index: $i$
Class index: $c$

Output:
Point cloud with scores: $\hat{P}$

Initialization:
$\hat{p_{i,c}} = -99999999$                    ▷ set all scores in point cloud to -∞

for k=1 : j do                                  ▷ loop through all images

    Run Katz algorithm to acquire the points $p$ that the image $I_k$ contains.

    Store the points $p$ in vectors $\{v_1 ... v_x\}$ corresponding to which pixel they
    belongs to.

    for q=1:x do                                ▷ loop through all pixels

        Get the scores $S$ for pixel $q$ in $I_k$

        for c=1:num_classes do                  ▷ loop through classes

        if $S_c > \hat{p_{l,c}}$     ▷ if the scores is larger in the pixel than for the points in $P$
▷ $p_l$ is a point in vector $v_q$ and $\hat{p}_l$ is the corresponding point in the point cloud

            $\hat{p_{l,c}} = S_c$                ▷ max-pooling of scores

        end if

        end for

    end for

end for
```

Algorithm 3 : Sum of scores

```
Input:
Sequence of classified images: { $I_1 ... I_j$}   ▷ each pixel contains as score for each class
Point Cloud: $P$
Point index: $i$
Class index: $c$

Output:
Point cloud with scores: $\hat{P}$

Initialization:
$\hat{p_{i,c}} = 0$                                  ▷ set all scores in point cloud to 0

for k=1 : j do                                        ▷ loop through all images

    Run Katz algorithm to acquire the points $p$ that the image $I_k$ contains.

    Store the points $p$ in vectors $\{v_1 ... v_x\}$ corresponding to which pixel they
    belongs to.

    for q=1:x do                                      ▷ loop through all pixels

        Get the scores $S$ for pixel $q$ in $I_k$

        for c=1:num_classes do                        ▷ loop through classes

▷ $p_l$ is a point in vector $v_q$ and $\hat{p_l}$ is the corresponding point in the point cloud
            $\hat{p_{l,c}} = \hat{p_{l,c}} + S_c$                    ▷ summation of scores

        end for

    end for

end for
```

4 Implementation

This chapter describes the libraries and the implementation of the method.

4.1 Libraries

This section describes important libraries used in this thesis.

4.1.1 PCL

PCL [15] is a library for processing 2D/3D images and point clouds. In this thesis it has been used to process point clouds and doing operation on them.

4.1.2 Matconvnet

Matconvnet[18] is a deep learning library for Matlab that has been used in this thesis. This library has been used for both training networks and semantic segmentation of images in this thesis.

4.1.3 Caffe

Caffe [6] is a deep learning library in C++. This library also has two wrappers, one in python and one in matlab. In this thesis, the wrapper in matlab called matcaffe has been used for semantic segmentation of images.

4.2 Implementation of Semantic Segmentation of Point Clouds

The first part of the pipeline, projection of point clouds into 2D images, was implemented in C++. Also, the PCL library was used to store point clouds and to perform operations on them. First were the point clouds read and stored in PCL point clouds. Then the Katz projection was performed and all visible points were stored in different point-structs with its index, depth value, RGB values, label and weight. All points that were in the same pixel had their point-structs stored in the same vector. Then the mean shift clustering algorithm was performed for each pixel and a depth and RGB images were created. Lastly the label images were created.

The semantic segmentation part used both matconvnet and the matlab wrapper of the caffe library, depending on which network was used. The LRR- net was based on matconvnet. The ADE20K-net was based on the caffe library and therefore the matcaffe was used to segment the images. The scores were saved into bin files so we easily could read them in C++

The back projection of the scores into the point cloud was also implemented in C++ with the help of the PCL library. The first step was to read the point cloud into the PCL point cloud class. Then two new point clouds were created with only the coordinates of the points. One fully sized point cloud and one point cloud down sampled to approximate 100 000 points. Then the Katz algorithm was performed for each camera angle, because we needed to know which point was in which projection. Lastly the scores from the bin-files were read and either the max pooling algorithm or the summation algorithm were used to store the scores into the point clouds.

5 Experiments

This chapter describes the datasets used for the experiments, the performed experiments and the results.

5.1 Datasets

In this thesis two datasets have been used for the experiments, Semantic3D and VPS. Semantic3D [3] is a dataset containing labeled point clouds of urban scenes. All points in the point clouds has a RGB value and a XYZ coordinate. The dataset have 8 different classes and one unlabeled class as is shown in table 5.1. In this thesis the Semantic3D dataset has been used for evaluation of classification for both images and point clouds.

Class	Class number
man-made terrain	1
natural terrain	2
high vegetation	3
low vegetation	4
buildings	5
hard scapes	6
scanning artefacts	7
cars	8
unlabeled	0

Table 5.1: *Shows the classes for Semantic3D dataset.*

The Semantic3D dataset consist of 30 labeled point clouds. We used six of them for evaluation. These six were Bildstein3, Bildstein5, Domfountain1, Domfountain2, Untermaederbrunnen1 and Untermaederbrunnen3. These point clouds were chosen because they are relatively small and well labeled. In figure 5.1, the six point clouds used are shown. Figure 5.2 and 5.3, shows the bildstein3 and domfountain3 point clouds and an image inside of them.

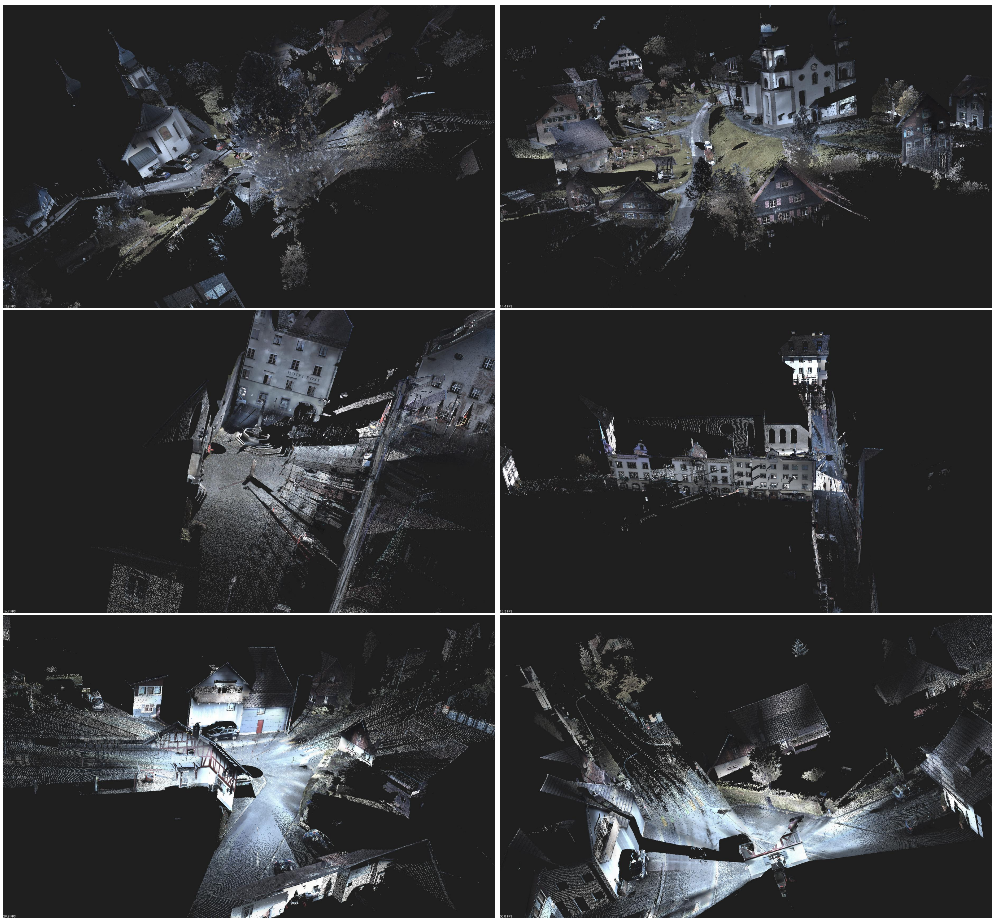

Figure 5.1: *The images shows the six point clouds used for evaluation. Top left: Bildstein3. Top right: Bildstein5. Middle left: Domfountain1. Middle right: Domfountain2. Bottom left: Untermaederbrunnen1. Bottom right: Untermaederbrunnen3.*

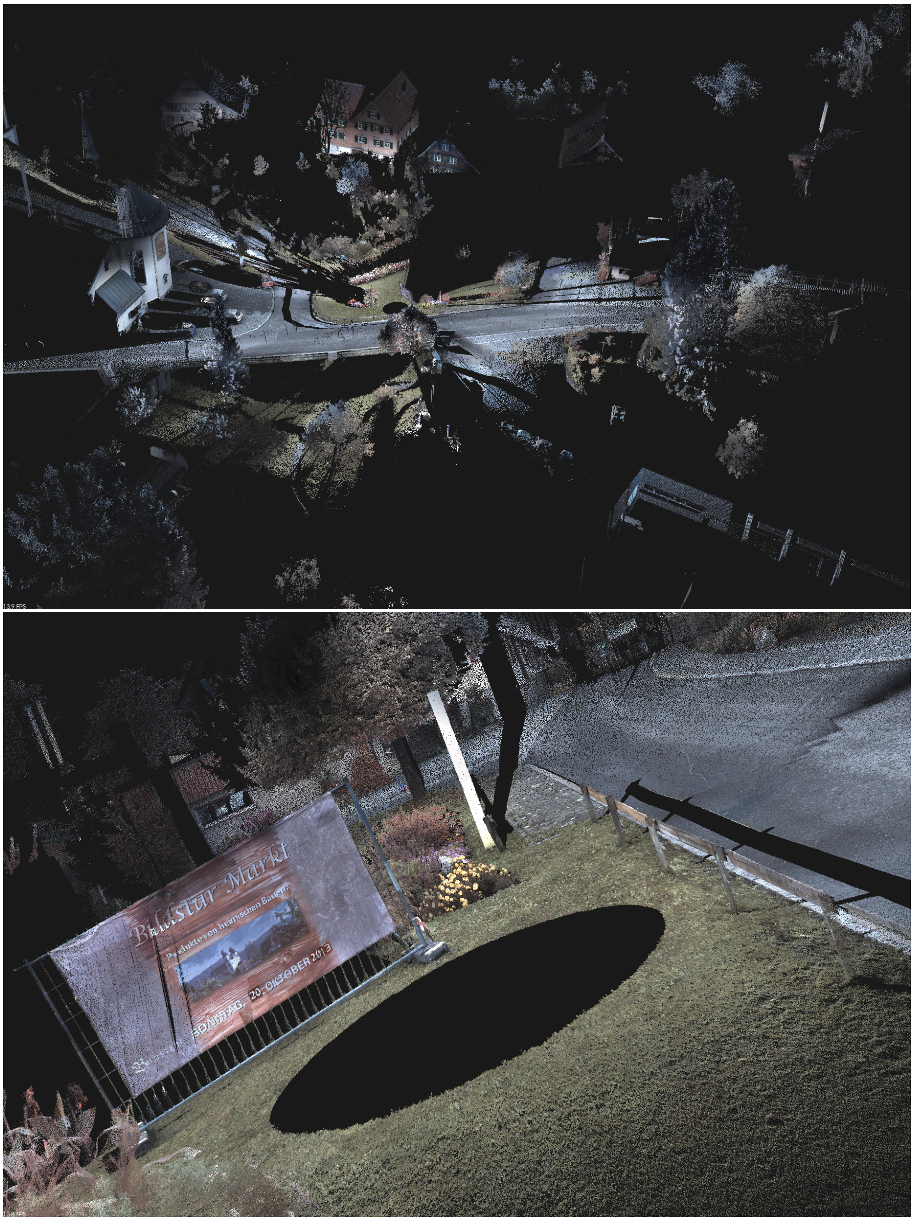

Figure 5.2: *The upper image shows the bildstein3 point cloud from above. The lower image shows a view inside the point cloud.*

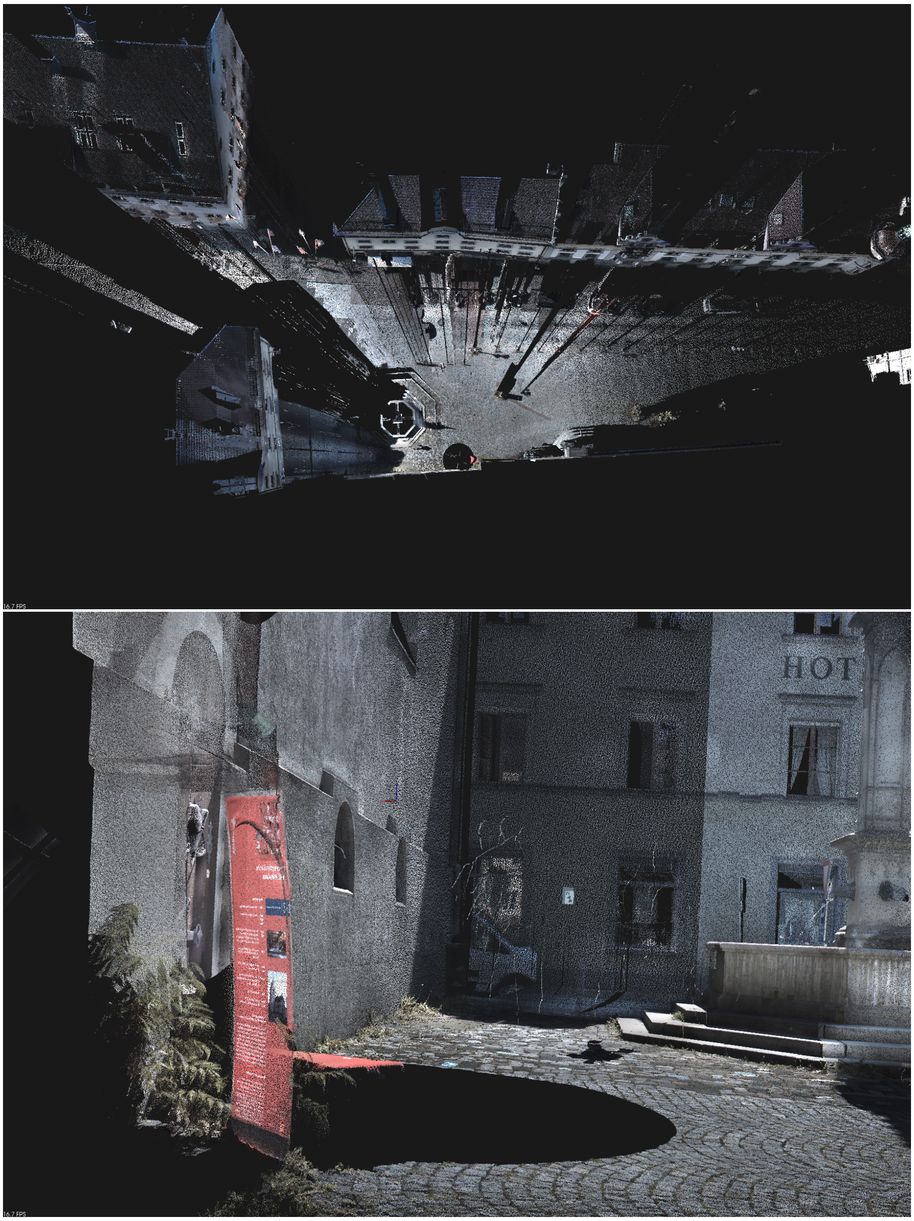

Figure 5.3: *The upper image shows the domfountain1 point cloud from above. The lower image shows a view inside the point cloud.*

The VPS dataset [10] is a dataset made by Linköpings University and contains RGB point clouds acquired using a lidar scanner. There are point clouds from a courtyard and from an indoor scene. This dataset has no labels, hence it is only used for qualitative testing. In this thesis, only the courtyard point clouds are used as we investigate semantic segmentation on outdoor scenes and the CNNs used only has fitting classes for the courtyard.

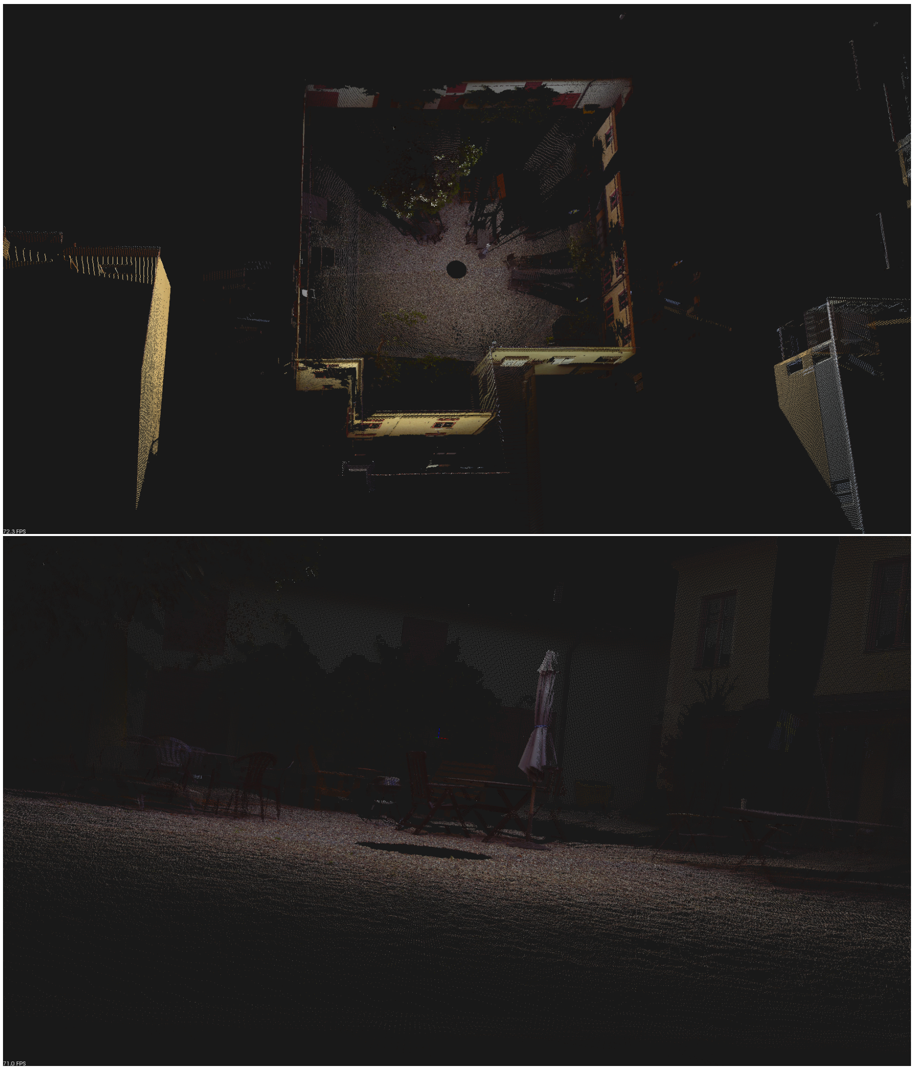

Figure 5.4: *The upper image shows the courtyard from the VPS1 point cloud from above. The lower image shows a view inside the point cloud.*

5.2 Performance Measures

This section describes the experiments and their measures.

5.2.1 Measures for Semantically Segmented Images

To evaluate the performance of the CNNs, we measured the pixel accuracy (see equation 5.1), the mean accuracy (see equation 5.2), the mean intersection over union and the intersection over union for each class (see equation 5.3). To calculate the pixel accuracy following formula was used:

$$PA = \frac{TP}{N} \tag{5.1}$$

where the TP is the true positives, or the correctly labeled pixels. N is the number of all valid pixels in the image.

The mean accuracy over all classes was calculated using the following formula:

$$MPA = \frac{\sum_{i=1}^{k} \frac{TP_i}{FP_i + TP_i}}{k} \tag{5.2}$$

where TP_i is the true positives and FP_i is the sum of the false positives for class i. A false positive is a pixel that were labeled wrongly with a certain class. For example, in our case all pixels that were labeled wrongly with the label car, or all pixels that were wrongly labeled with building.

To calculate the intersection over union for a class, the following formula was used:

$$IoU_i = \frac{TP_i}{FP_i + FN_i + TP_i} \tag{5.3}$$

where TP is the true positives FP is the false positives and FN is the false negative or all the pixels that should have been labeled with a certain class but was not. For example all pixels that was labeled in the ground truth with car, but were labeled with another class. To calculate the mean intersection over union (mIoU) the mean of all IoUs were calculated.

For ground truth the projected label images was used. Because the LRR-net and the ADE20K-net had different classes from Semantic3D, we had to map the classified classes into the 8 classes of Semantic3D. For example, all buildings, walls, roofs were mapped into the buildings class in Semantic3D. Traffic lights, signs, fences and poles were mapped into the hard scapes class. We first decided which class had the highest score and then the mapping was performed. The full mapping table for the ADE20K dataset into Semantic3D can be seen in table A.1 in the appendix. Table A.2 in the appendix shows the mapping from Cityscapes to Semantic3D. To calculate the pixel accuracy, mean accuracy and the mean intersection over union, the confusion matrix for all classified images was calculated. Pixels for which the ground truth was labeled with either void, unlabeled or scanning artefacts was excluded from the calculations.

Because the Semantic3D dataset had 3 different classes for vegetation and the LRR-net only had two, we made two different evaluations of the dataset. One where we merged all the vegetation classes into one class and one where we did not merge them.

5.2.2 Measures for Semantically Segmented Point Clouds

To evaluate the semantically segmented point clouds, we used the same measures as for the images, point accuracy, mean point accuracy, mean intersection over union and intersection over union for each class. For ground truth, the Semantic3D labels were used. Points labeled to unlabeled or scanning artefacts in the ground truth were excluded from the evaluation. Also points without labels from the pipeline were excluded. The confusion matrix was calculated for all the valid points and the point accuracy, mean accuracy and mean intersection union was calculated using the same formulas as for evaluating the projected images.

5.3 Results

This section shows the results of the experiments.

5.3.1 Semantic Segmentation of Images

This section shows semantically segmented images projected from the point clouds, from our pipeline. In the figures 5.5-5.8 we can see that the ADE20K network performs better than the LRR-net. This can be seen by comparing the ground truth images with the mapped semantically segmented images. Especially in figure 5.6 we can see that the LRR-net performs much worse than the ADE20K network. This may be because the LRR-net is trained on street scenes and is therefore not fit to classify images with so much vegetation. The ADE20K network is trained on a more varied data and can therefore perform better. The brown stripes that can be seen in the ground truth of e.g. figure 5.8, are pixels that have been labeled with the class scanning artefacts and are excluded from the evaluation.

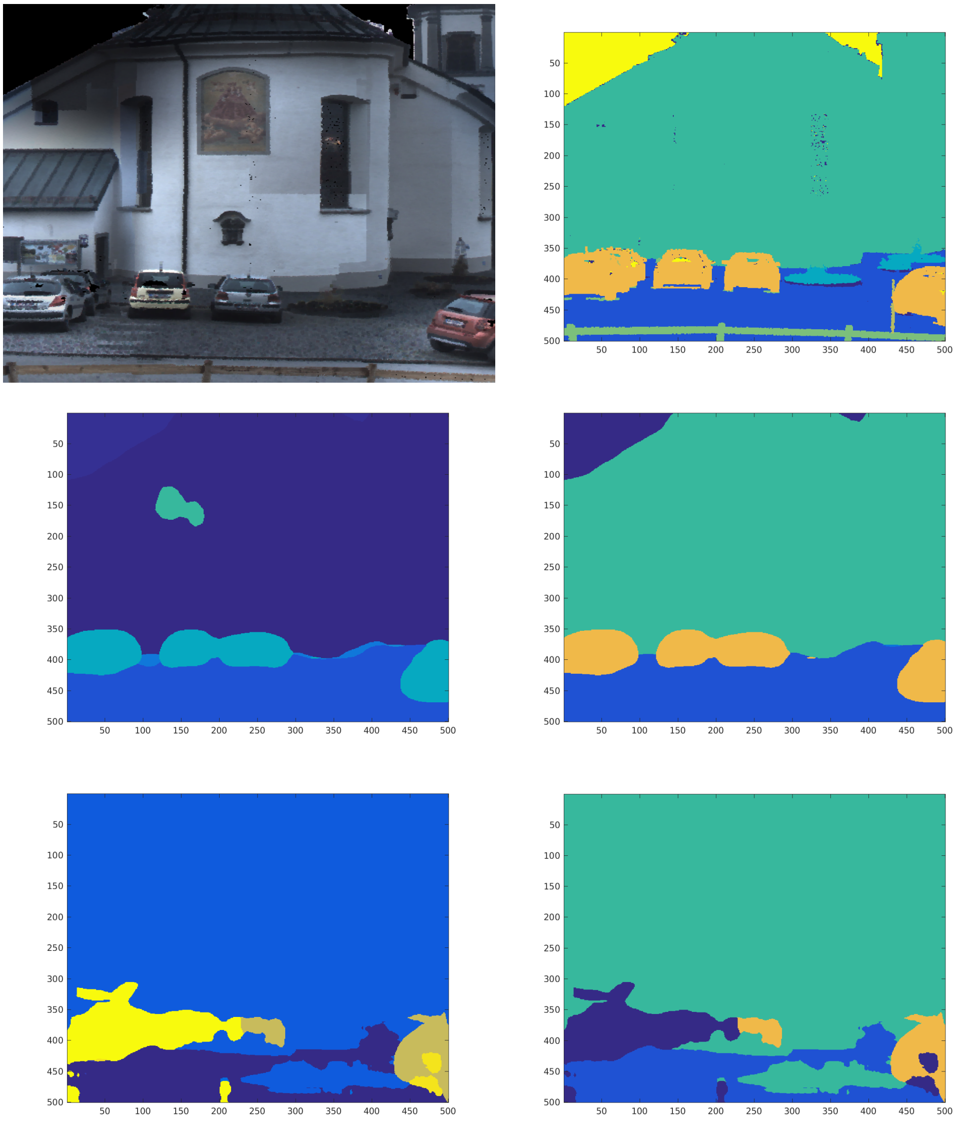

Figure 5.5: *Shows a view for Bildstein3 and the semantic segmentation of the view. Top right: Our projected ground truth for the view. Middle left: The semantically segmented image from the ADE20K-net. Middle right: The semantically segmented image mapped into semantic3D classes for the ADE20K net. Bottom left: Semantic segmentation of the image from LRR-net. Bottom right: The semantically segmented image mapped into semantic3D classes for the LRR-net.*

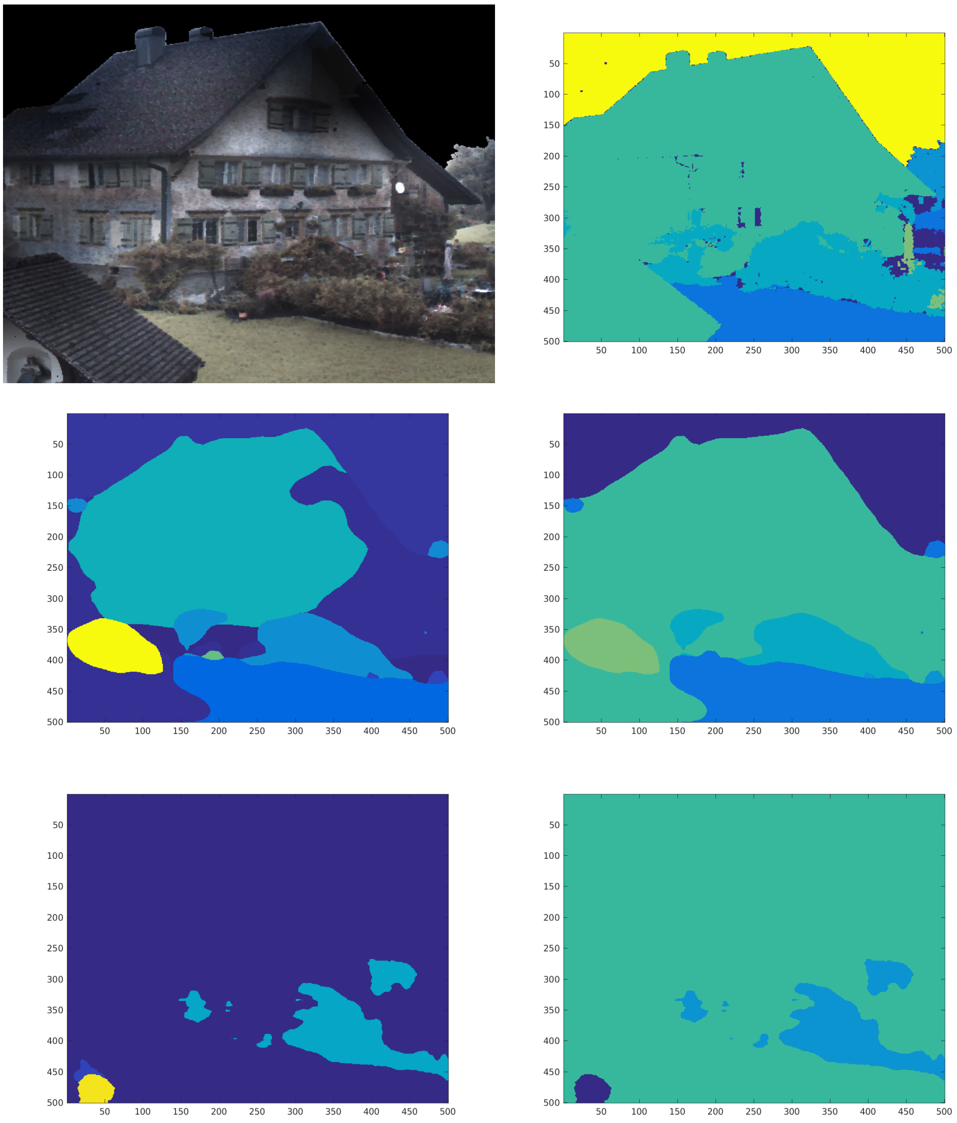

Figure 5.6: *Shows a view for Bildstein5 and the semantic segmentation of the view. Top right: Our projected ground truth for the view. Middle left: The semantically segmented image from the ADE20K-net. Middle right: The semantically segmented image mapped into semantic3D classes for the ADE20K net. Bottom left: Semantic segmentation of the image from LRR-net. Bottom right: The semantically segmented image mapped into semantic3D classes for the LRR-net.*

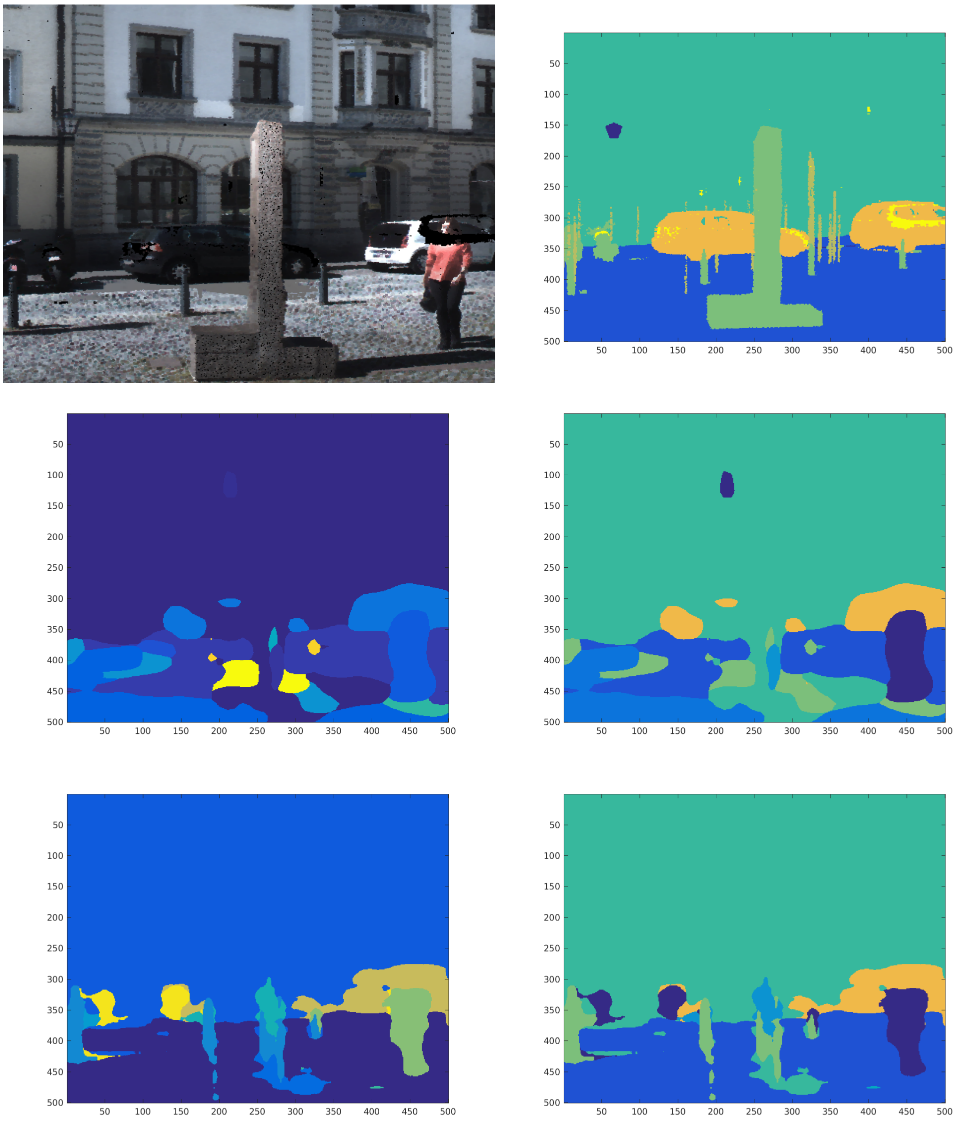

Figure 5.7: *Shows a view for Domfountain1 and the semantic segmentation of the view. Top right: Our projected ground truth for the view. Middle left: The semantically segmented image from the ADE20K-net. Middle right: The semantically segmented image mapped into semantic3D classes for the ADE20K net. Bottom left: Semantic segmentation of the image from LRR-net. Bottom right: The semantically segmented image mapped into semantic3D classes for the LRR-net.*

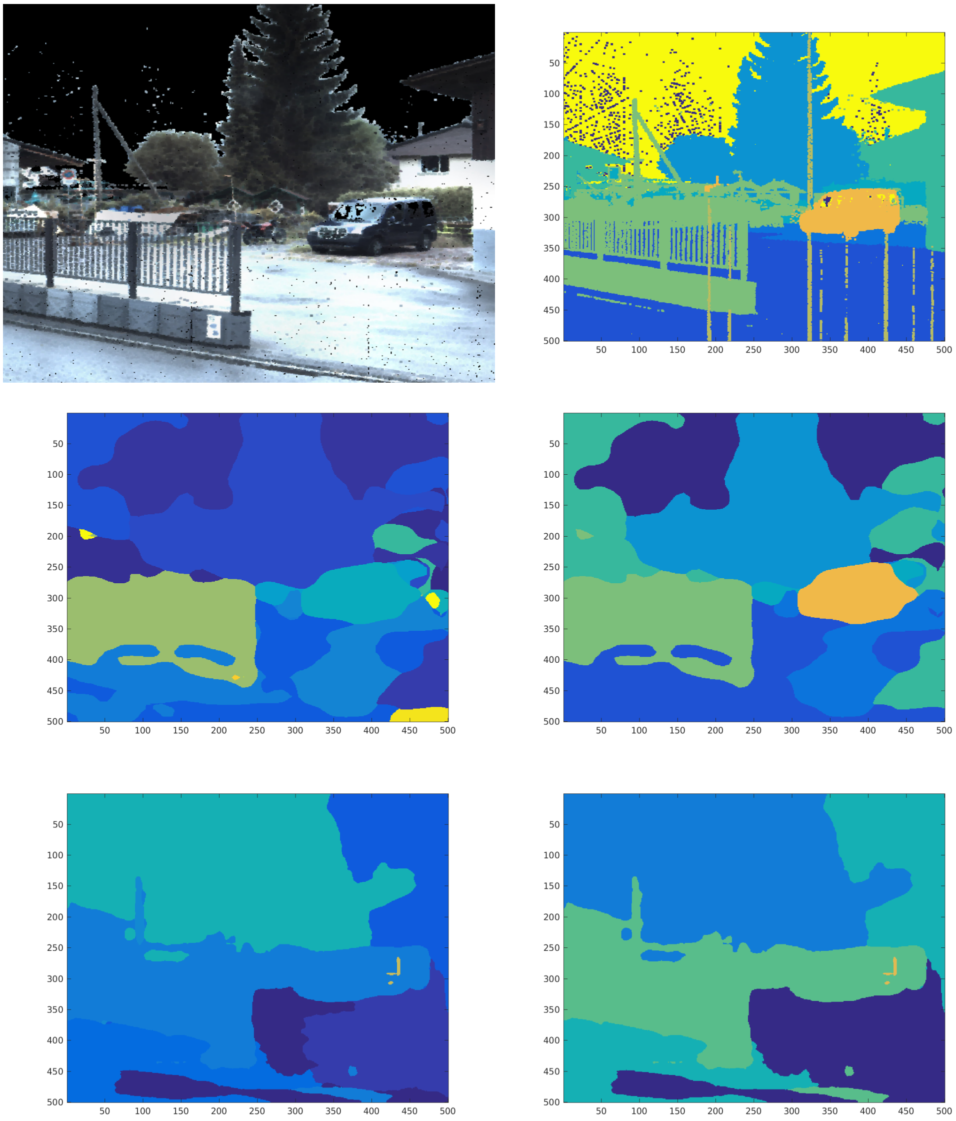

Figure 5.8: *Shows a view for Untermaederbrunnen1 and the semantic segmentation of the view. Top right: Our projected ground truth for the view. Middle left: The semantically segmented image from the ADE20K-net. Middle right: The semantically segmented image mapped into semantic3D classes for the ADE20K net. Bottom left: Semantic segmentation of the image from LRR-net. Bottom right: The semantically segmented image mapped into semantic3D classes for the LRR-net.*

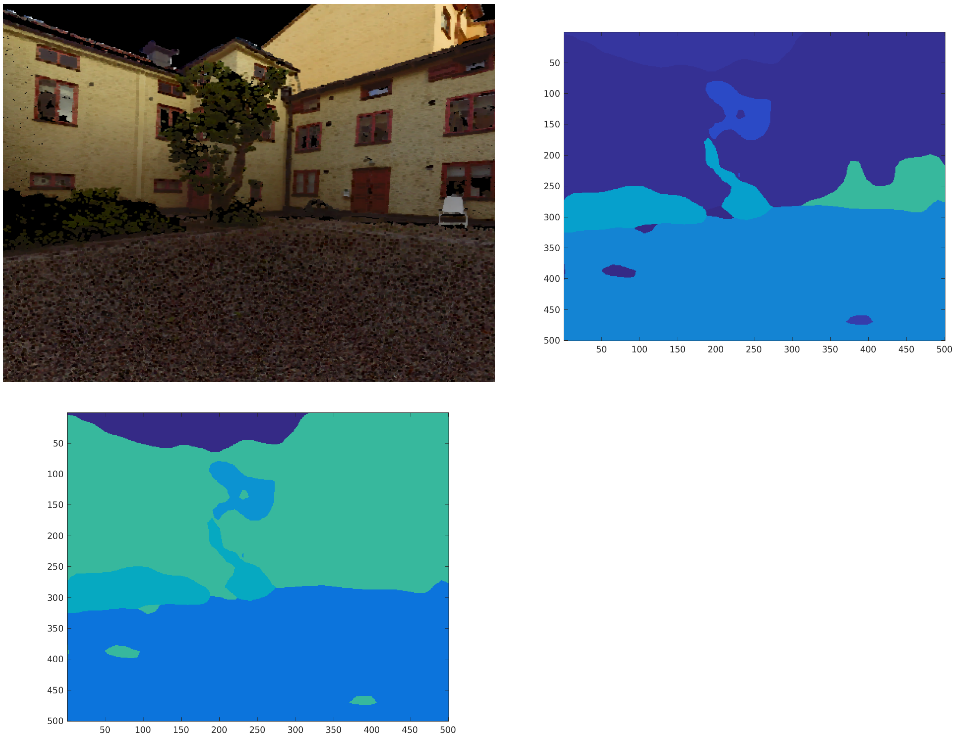

Figure 5.9: *Shows a view from VPS1 and the semantic segmentation for the view. Top right: The view semantically segmented with ADE20K net. Bottom: The mapped semantically segmented view for the ADE20K net.*

5.3.2 Semantic Segmentation of Point Clouds

This section shows semantically segmented point clouds, from our pipeline. As seen in both figure 5.10 and 5.11 the buildings are easy for our pipeline to label correctly for both networks. We also see that the vegetation is not as good classified. This may be because it is hard to distinguish the different objects in the projected images and therefore they are labeled wrongly. We also see that our pipeline labels some of the man-made terrain into buildings. This may be because in some images the ground and the buildings looks alike. The black points in our semantically segmented point clouds are the points set to void. The dark areas in the point clouds, is the points that we did not cover with our projections. Because of this these points did not get a label.

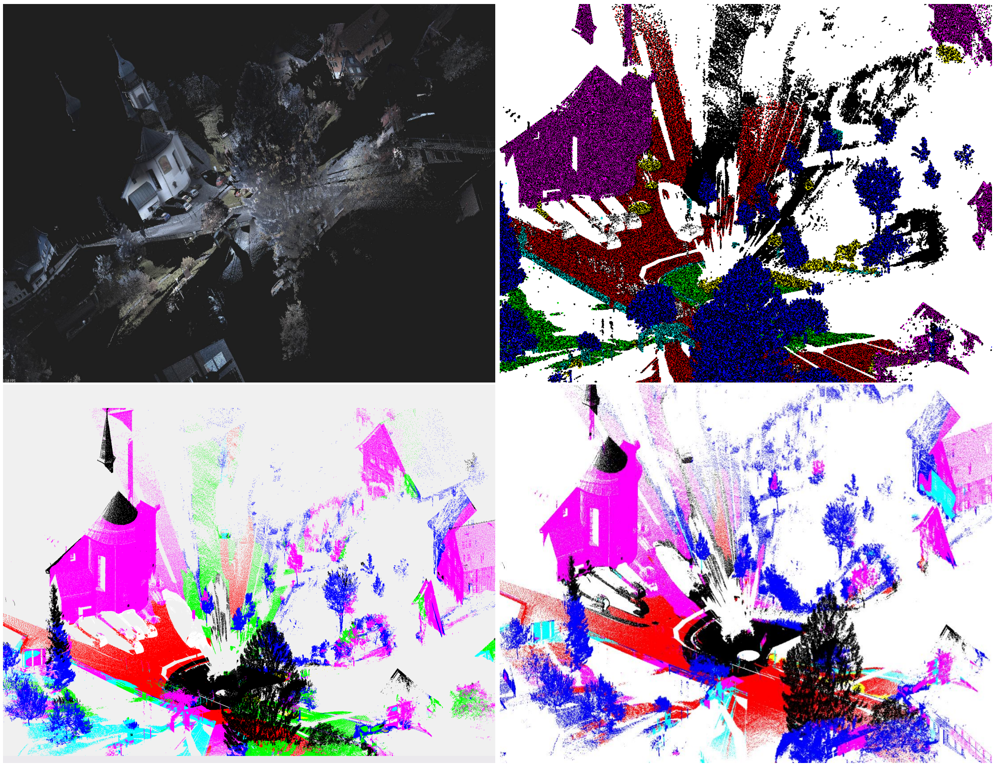

Figure 5.10: *Top left: Bildstein3 point cloud. Top right: Ground truth for Bildstein3. Bottom left: Semantically segmented Bildstein3 using ADE20K net. Bottom right: Semantically segmented Bildstein3 using LRR-net.*

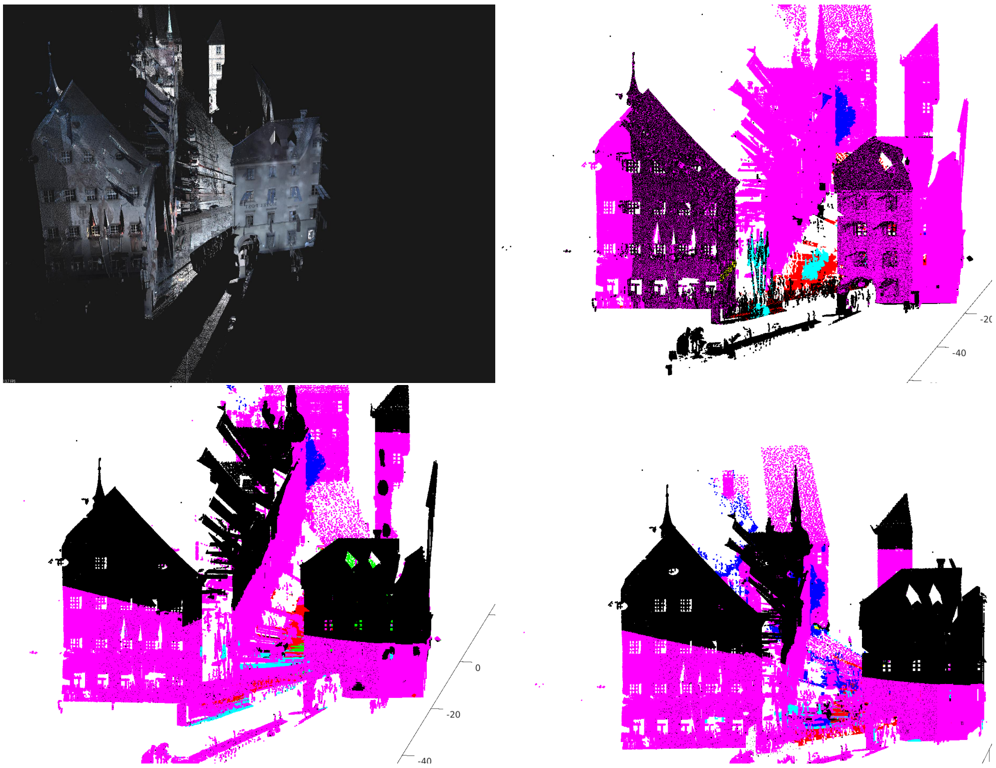

Figure 5.11: *Top left: Domfountain1 point cloud. Top right: Ground truth for Domfountain1. Bottom left: Semantically segmented Domfountain1 using ADE20K net. Bottom right: Semantically segmented Domfountain1 using LRR-net.*

5.3.3 Evaluation of Semantic Segmentation of Images

This section shows the results for the classified images. The measures used are described in section 5.2 and in table 5.2 the pixel accuracy (PA), mean pixel accuracy (mPA) and the mean intersection of union (mIoU) are shown for the ADE20K-net and the LRR-net. In table 5.3 the intersection of union (IoU) for each class is shown.

Network	**PA**	**mPA**	**mIoU**
ADE20K	0.6965	0.5730	0.3980
LRR-net	0.6629	0.4365	0.2801

Table 5.2: *Pixel Accuracy, mean pixel accuracy and mean intersection over union for the semantically segmented images.*

Network	**IoU1**	**IoU2**	**IoU3**	**IoU4**	**IoU5**	**IoU6**	**IoU8**
ADE20K	0.3746	0.3408	0.3790	0.1839	0.7558	0.2433	0.5086
LRR-net	0.4028	0	0.4202	0.0264	0.7254	0.1293	0.2565

Table 5.3: *Intersection over union for each class for the semantically segmented images.*

As seen in table 5.2 the ADE20K-net performed better than the LRR-net. Some reasons for this can be that the ADE20K-net was trained on a more variety of data than the LRR-net that had training data mostly from street scenes of cities. The point clouds from the semantic3D dataset contains both street scenes and areas with more vegetation, which is why the ADE20K-net where more suitable for the task. Another reason for the less performance of the LRR-net is that it only has two classes for vegetation where the semantic3D had 3 classes. For this we map the two vegetation from the LRR-net classes into the two best fitting in vegetation classes in Semantic3D, which lead to one of the classes always getting a 0 percent accuracy as seen in table 5.3. To get a better representation of the measures we merged vegetation classes (class 2-4) into one during the mapping and the PA, mPA and mIou can be seen in table 5.4. In table 5.5 the IoU for merged vegetation and the other evaluated classes can be seen.

Network	**PA**	**mPA**	**mIoU**
ADE20K	0.7409	0.6224	0.4796
LRR-net	0.6907	0.5292	0.3752

Table 5.4: *Pixel Accuracy, mean pixel accuracy and mean intersection over union for the semantically segmented images with merged vegetation classes.*

Network	IoU1	IoU2-4	IoU5	IoU6	IoU8
ADE20K	0.3746	0.5155	0.7558	0.2433	0.5086
LRR-net	0.4028	0.3618	0.7254	0.1293	0.2565

Table 5.5: *Intersection over union for each class for the semantically segmented images with merged vegetation classes.*

Comparing the measures from table 5.2 and 5.4 we can see that all measures got slightly better for both networks when the vegetation was merged. This because some vegetation looks alike in the images and the networks classify them with the wrong vegetation labels. Therefore, when merging the vegetation classes, the accuracy improves as there is only one vegetation class. For the LRR-net the zero accuracy on class 2 also is gone, which improves the accuracy further. In table 5.5 you can also see that the IoU for the merge classes. Comparing the IoU from this merged class with the IoU for the vegetation classes (2,3 and 4) in table 5.3 we see the IoU for the merge class is better. Especially for the ADE20K-net were the IoU of the merge vegetation class was better than for any of the vegetation classes. This indicates that the network had classified many vegetation pixels into the wrong vegetation class.

5.3.4 Evaluation of Semantic Segmentation of Point Clouds

This section shows the result of the semantically segmented point clouds. In table 5.6 the point accuracy (PA) mean point accuracy (mPA) and the mean intersection of union (mIoU) are presented for the point clouds. As seen in the table both networks were evaluated for both projecting methods. Also in table 5.7 the IoU for each class is presented for both projecting methods.

Network	Method	PA	mPA	mIoU
ADE20K	max	0.7114	0.5909	0.4051
LRR-net	max	0.6900	0.4364	0.2859
ADE20K	sum	0.7243	0.5958	0.4167
LRR-net	sum	0.7005	0.4398	0.2923

Table 5.6: *Point Accuracy, mean point accuracy and mean intersection over union for the semantically segmented point clouds.*

Network	Method	IoU1	IoU2	IoU3	IoU4	IoU5	IoU6	IoU8
ADE20K	max	0.3868	0.2875	0.3639	0.2012	0.7680	0.3010	0.5274
LRR-net	max	0.4648	0	0.3992	0.0376	0.7396	0.1094	0.2504
ADE20K	sum	0.3965	0.3029	0.3815	0.1925	0.7813	0.2979	0.5644
LRR-net	sum	0.4749	0	0.4036	0.0293	0.7515	0.1091	0.2776

Table 5.7: *Intersection over union for each class for the semantically segmented point clouds.*

As seen in table 5.6 the method that performed best for both networks was the projection using summation of the scores. A reason for this may be that the max pooling only takes out the maximum score for a point. If this label is not correct the point is labeled wrongly. The summation instead takes the largest summation of a score for the point, which necessarily won't pick the label with the highest individual score.

The problem for the LRR-net with the mapping of 2 vegetation classes into the 3 vegetation classes of semantic3D, also occurs for the point clouds. The same solution as for the semantically segmented images of merging the vegetation classes was used to solve the problem. The PA, mPA and mIoU can be seen for the merged vegetation in table 5.8. Also the IoU for each class can be seen in table 5.9.

Network	**Method**	**PA**	**mPA**	**mIoU**
ADE20K	max	0.7432	0.6490	0.4886
LRR-net	max	0.7097	0.5324	0.3814
ADE20K	sum	0.7557	0.6526	0.5025
LRR-net	sum	0.7217	0.5385	0.3936

Table 5.8: *Point Accuracy, mean point accuracy and mean intersection over union for the semantically segmented point clouds with merged vegetation classes.*

Network	**Method**	**IoU1**	**IoU2-4**	**IoU5**	**IoU6**	**IoU8**
ADE20K	max	0.3868	0.4595	0.7680	0.3010	0.5274
LRR-net	max	0.4648	0.3429	0.7396	0.1094	0.2504
ADE20K	sum	0.3965	0.4723	0.7813	0.2979	0.5644
LRR-net	sum	0.4749	0.3551	0.7515	0.1091	0.2776

Table 5.9: *Intersection over union for each class for the semantically segmented point clouds with merged vegetation classes.*

Comparing table 5.6 and 5.8 we see like for the images that the measures got better with the merging of the vegetation classes. This is because, like for the images there was a mix up between the vegetation classes and also for the LRR-net we do not get the zero accuracy on class 2. In table 5.9 we can see that the IoU for the merge vegetation was better than the vegetation classes in table 5.7.

We also compared our result with four existing results from [3], which are HarrisNet, DeepSegNet, TMLC-MS [5] and TML-PC. An overview for the methods can be seen in table 5.10. HarrisNet is an unpublished method using a 3D convolutional network to semantically segment the point clouds. DeepSegNet is also an unpublished approach which is very similar to our approach. In this method they pick suitable views in the point clouds and generate a RGB image and an image containing geometric features for each view. When the images have been generated, pixel-wise labeling is performed using convolutional networks. Finally they back project the labels into the point cloud. TMLC-MS [5] is a published method that is described in related work, section 2.3. The last method TML-PC is also an unpublished approach. This method uses pure colour information with context aware features and TextonBoost.

In figure 5.12 the intersection of union for each class can be seen for HarrisNet, DeepSegNet, TMLC-MS and TML-PC. In figure 5.13 the point accuracy is presented and in figure 5.14 the mean Intersection of union.

Method	**Description**	**Published**
HarrisNet	Deep 3D Convolutional Network	No
DeepSegNet	Unstructured point cloud labeling using deep seg	No
TMLC-MS	ML with Covariance Features (Multi-Scale)	Yes
TML-PC	Pure color: context aware features and TextonBoost	No

Table 5.10: *Shows the methods that we compare our results with.*

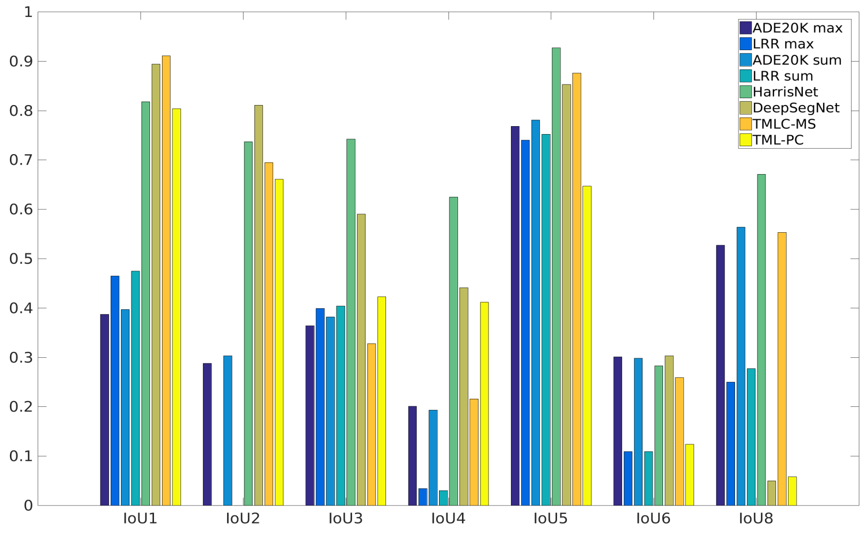

Figure 5.12: *Shows the IoU for each class for each method. Dark blue: Shows the IoU for ADE20K-net with max pooling of the scores. Blue: Shows the IoU for LRR-net with max pooling of the scores. Light blue: Shows the IoU for ADE20K-net with summation of the scores. Turquoise: Shows the IoU for LRR-net with summation of the scores. Green: Shows the IoU for HarrisNet. Brown: Shows the IoU for DeepSegNet. Orange: Shows the IoU for TMLC-MS. Yellow: Shows the IoU for TML-PC.*

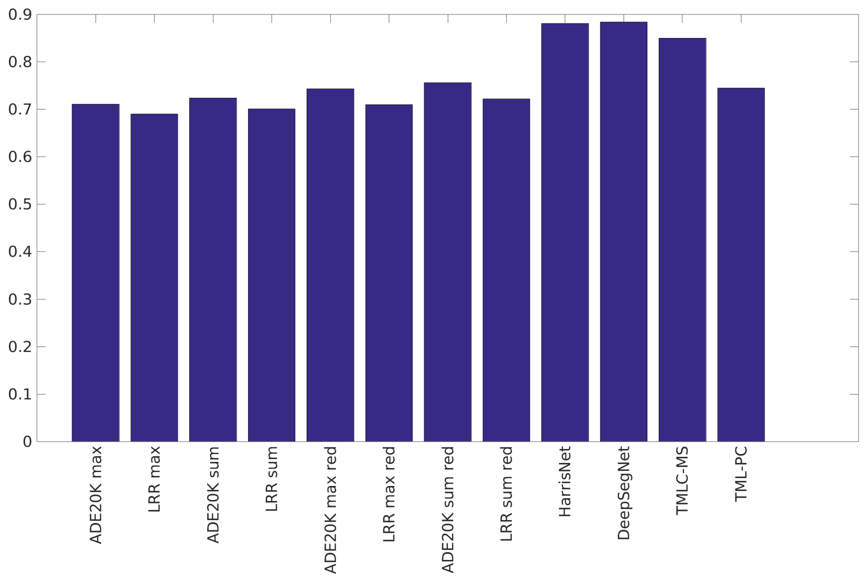

Figure 5.13: *Shows the pixel accuracy for our methods and the methods we compared with.*

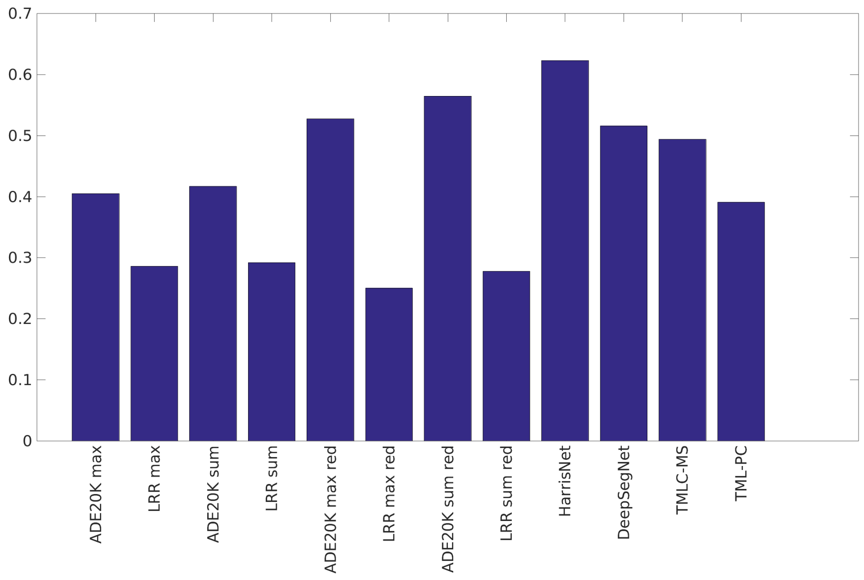

Figure 5.14: *Show the mIoU for our methods and for the methods we compared with.*

As seen in figure 5.12 the IoU for each class for our methods was worse for almost every class. A reason for this is that we did not use a network that was trained on the semantic3D dataset and therefore had to map the classes into the semantic3D classes, which will give us an error. If we instead compare the point accuracy in 5.13 we see that our method had about 10 percent worse accuracy than for the best method but almost the same accuracy as for the worse method. We also see that merging the vegetation gives us more comparable result. The same pattern can be seen in figure 5.14 were the mIoU i shown. Here the merging of vegetation boost the mIoU which gives us results who are second best for the ADE20K-net.

6 Discussion and Future Work

As described in section 3.3 we choose to use two pre-trained networks for the classification. This is because it saved us time to not having to train a CNN and also the focus of the thesis was not to perform the best segmentation. Our main focus was to find a method to project 3D point clouds into 2D images and then project the semantically segmented images back into the point cloud. A better result for the evaluation could be achieved if we instead had trained our own CNN, so it matched the semantic3D data better. The semantic segmentation of the point clouds, could also be better if a depth channel was added into the CNN, as we generate depth images in the projection of the point clouds. To train and integrate the depth images for the semantic segmentation of the images could be future work to improve the results.

In the evaluation of the segmentation many errors were introduced. As discussed in 5.3 the mapping of the classes was necessary to evaluate the segmentation as the labels were for the semantic3D dataset. The mapping introduces errors in the evaluation. The networks could for example predict the class ground for both grass and roads, which is two different classes in the Semantic3D dataset. This will then introduce an error in the mapping. In the evaluation of the images we had to create our own ground truth from the point cloud for each image. As there could be more than one point in a pixel and we set the pixel in the ground truth to the most occurring label, it introduced an error. This could be a reason why the final measures of the point clouds improved a bit comparing with the measures for the images.

From chapter 5 we see that the ADE20K network overall performed better than the LRR network. A reason for this may be that the ADE20K network was trained on a more variety of data. As the LRR-net mostly was trained on street scenes the

LRR-net performance was worse on areas with more vegetation. Another reason for the better performance for the ADE20K network may be that it also had more classes than the LRR-net. This makes it easier for the network to find a fitting class for the objects.

As seen in section 5.3.4 the merging of the vegetation classes improved our results. As discussed this may be because the CNN used confuses the vegetation classes and label the points with wrong vegetation class. We also see that our results are comparable to earlier results, especially when we merge the vegetation classes.

7 Conclusion

In this this thesis, we have proposed a method to semantically segment point clouds using deep learning. The proposed method is to project the point clouds into 2D using the Katz projection and mean shift clustering. Then we semantically segment the projected 2D images using CNNs. Finally, we project back the scores from the semantically segmentated images into the point clouds, using either summation of scores or max pooling of the scores.

We find that our method performs well on the Semantic3D dataset. Our results could be further improved by using CNNs trained on the semantic3D dataset. As our CNNs were either trained on the Cityscapes dataset or the ADE20K dataset we had to map our classes into the classes of Semantic3D for evaluation. Our results could also be improved by integrating the depth images in the semantic segmentation of the image. We also find that our results are comparable to state-of-the-art without any fine-tuning on the Semantic3D dataset.

Appendix

A

Tables

ADE20K Class	ADE20K Class-num	Sem3D Class	Sem3D Class-num
void	0	void	0
wall	1	buildings	5
building	2	buildings	5
sky	3	void	0
floor	4	buildings	5
tree	5	high vegetation	3
ceiling	6	buildings	5
road	7	man-made terrain	1
bed	8	void	0
windowpane	9	buildings	5
grass	10	natural terrain	2
cabinet	11	void	0
sidewalk	12	man-made terrain	1
person	13	void	0
earth	14	natural terrain	2
door	15	buildings	5
table	16	void	0
mountain	17	natural terrain	2
plant	18	low vegetation	4
curtain	19	void	0
chair	20	void	0
car	21	car	8
water	22	natural terrain	2
painting	23	void	0
sofa	24	void	0
shelf	25	void	0
house	26	buildings	5
sea	27	natural terrain	2
mirror	28	void	0
rug	29	void	0
field	30	natural terrain	2
armchair	31	void	0
seat	32	void	0
fence	33	hard scapes	6
desk	34	void	0
rock	35	natural terrain	2
wardrobe	36	void	0
lamp	37	void	0
bathtub	38	void	0
railing	39	hard scapes	6
cushion	40	void	0
base	41	hard scapes	6

box	42	void	0
column	43	hard scapes	6
signboard	44	hard scapes	6
chest of drawers	45	void	0
counter	46	void	0
sand	47	man-made terrain	1
sink	48	void	0
skyscraper	49	buildings	5
fireplace	50	void	0
refrigerator	51	void	0
grandstand	52	hard scapes	6
path	53	man-made terrain	1
stairs	54	hard scapes	6
runway	55	void	0
case	56	void	0
pool table	57	void	0
pillow	58	void	0
screen door	59	buildings	5
stairway	60	hard scapes	6
river	61	natural terrain	2
bridge	62	hard scapes	6
bookcase	63	void	0
blind	64	void	0
coffee table	65	void	0
toilet	66	void	0
flower	67	low vegetation	4
book	68	void	0
hill	69	natural terrain	2
bench	70	void	0
countertop	71	void	0
stove	72	void	0
palm	73	high vegetation	3
kitchen island	74	void	0
computer	75	void	0
swivel chair	76	void	0
boat	77	void	0
bar	78	void	0
arcade machine	79	void	0
hovel	80	buildings	5
bus	81	car	8
towel	82	void	0
light	83	void	0
truck	84	car	8

tower	85	buildings	5
chandelier	86	void	0
awning	87	void	0
streetlight	88	hard scapes	6
booth	89	buildings	5
television	90	void	0
airplane	91	void	0
dirt track	92	man-made terrain	1
apparel	93	void	0
pole	94	hard scapes	6
land	95	natural terrain	2
bannister	96	hard scapes	6
escalator	97	hard scapes	6
ottoman	98	void	0
bottle	99	void	0
buffet	100	hard scapes	6
poster	101	void	0
stage	102	hard scapes	6
van	103	car	8
ship	104	void	0
fountain	105	hard scapes	6
conveyer belt	106	hard scapes	6
canopy	107	buildings	5
washer	108	void	0
plaything	109	void	0
swimming pool	110	hard scapes	6
stool	111	void	0
barrel	112	void	0
basket	113	void	0
waterfall	114	natural terrain	2
tent	115	buildings	5
bag	116	void	0
minibike	117	void	0
cradle	118	void	0
oven	119	void	0
ball	120	void	0
food	121	void	0
step	122	hard scapes	6
tank	123	hard scapes	6
trade name	124	man-made terrain	1
microwave	125	void	0
pot	126	low vegetation	4
animal	127	void	0

bicycle	128	void	0
lake	129	natural terrain	2
dishwasher	130	void	0
screen	131	void	0
blanket	132	void	0
sculpture	133	hard scapes	6
hood	134	void	0
sconce	135	hard scapes	6
vase	136	void	0
traffic light	137	hard scapes	6
tray	138	void	0
ashcan	139	hard scapes	6
fan	140	void	0
pier	141	void	0
crt screen	142	void	0
plate	143	void	0
monitor	144	void	0
bulletin board	145	void	0
shower	146	void	0
radiator	147	hard scapes	6
glass	148	void	0
clock	149	void	0
flag	150	hard scapes	6

Table A.1: *The table shows the mapping from the classes of ADE20K to the classes of Semantic3D*

City Class	City Class-num	Sem3D Class	Sem3D Class-num
void	0	void	0
road	1	man-made terrain	1
sidewalk	2	man-made terrain	1
buildings	3	buildings	5
wall	4	buildings	5
fence	5	hard scapes	6
pole	6	hard scapes	6
traffic light	7	hard scapes	6
traffic sign	8	hard scapes	6
vegetation	9	high vegetation	3
terrain	10	low vegetation	4
sky	11	void	0
person	12	void	0
rider	13	void	0
car	14	car	8
truck	15	car	8
bus	16	car	8
train	17	void	0
motorcycle	18	void	0
bicycle	19	void	0

Table A.2: *The table shows the mapping from the classes of Cityscapes to the classes of Semantic3D*

Bibliography

[1] Leo Breiman. Random forests. *Mach. Learn.*, 45(1), 2001. Cited on page 7.

[2] Martin Danelljan, Giulia Meneghetti, Fahad Shahbaz Khan, and Michael Felsberg. Aligning the Dissimilar : A Probabilistic Method for Feature-Based Point Set Registration. *ICPR16*, 2016. Cited on page 3.

[3] ETH Zurich. Large-Scale Point Cloud Classification Benchmark. URL `http://www.semantic3d.net/`. Cited on pages 4, 23, and 40.

[4] Golnaz Ghiasi and Charless C. Fowlkes. Laplacian reconstruction and refinement for semantic segmentation. *ECCV*, 2016. Cited on pages 1, 6, and 16.

[5] Timo Hackel, Jan D Wegner, and Konrad Schindler. Fast Semantic Segmentation of 3D Point Clouds with Strongly Varying Density. *ISPRS16*, 2016. Cited on pages 7, 40, and 41.

[6] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv:1408.5093*, 2014. Cited on page 21.

[7] Sagi Katz. Direct Visibility of Point Sets. *ACM Trans. Graph.*, 26(3), 2007. Cited on page 11.

[8] Jan Knopp, Hayko Riemenschneider, and Luc Van Gool. 3D All The Way : Semantic Segmentation of Urban Scenes From Start to End in 3D. *CVPR*, 2015. Cited on pages 1 and 7.

[9] Yann LeCun and Yoshua Bengio. The handbook of brain theory and neural networks. chapter Convolutional Networks for Images, Speech, and Time Series, pages 255–258. MIT Press, Cambridge, MA, USA, 1998. Cited on page 5.

[10] Linköping University. Virtual Photo Sets. URL `http://www.hdrv.org/vps/`. Cited on page 27.

[11] Tianyi Liu, Shuangsang Fang, Yuehui Zhao, Peng Wang, and Jun Zhang. Implementation of Training Convolutional. *arXiv:1506.01195*, 2015. Cited on pages 5 and 6.

[12] Marius Cordts. Cityscapes Dataset. URL `https://www.cityscapes-dataset.com/dataset-overview/`. Cited on page 16.

[13] C V May. Fractional Max-Pooling. *arXiv:1412.6071*, 2015. Cited on page 5.

[14] Peter Meer. Mean Shift : A Robust Approach toward Feature Space Analysis. *IEEE*, 24(5), 2002. Cited on page 12.

[15] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). *ICRA*, 2011. Cited on page 21.

[16] Richard Szeliski. Computer Vision : Algorithms and Applications. 2010. URL `http://szeliski.org/Book`. Cited on page 12.

[17] Martin Thoma. A Survey of Semantic Segmentation. *arXiv:1602.06541*, 2016. Cited on page 6.

[18] Andrea Vedaldi and C V May. MatConvNet Convolutional Neural Networks for MATLAB. *arXiv:1412.4564*. Cited on page 21.

[19] Daniel Wolf, Johann Prankl, and Markus Vincze. Fast Semantic Segmentation of 3D Point Clouds using a Dense CRF with Learned Parameters. *ICRA*, 2015. Cited on pages 1 and 7.

[20] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip Torr. Conditional Random Fields as Recurrent Neural Networks. *ICCV*, 2015. Cited on pages 1 and 6.

[21] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic Understanding of Scenes through the ADE20K Dataset. *arXiv:1608.05442*, 2016. Cited on pages 1, 6, and 16.