Semantic Classification in Uncolored 3D Point Clouds using Multiscale Features

Michael Neumann, Dorit Borrmann, and Andreas Nüchter*

Informatics VII – Robotics and Telematics, Julius Maximilian University of Würzburg, Germany
{andreas.nuechter}@uni-wuerzburg.de

Abstract While the semantic segmentation of 2D images is already a well-researched field, the assignment of semantic labels to 3D data is lagging behind. This is partly due to the fact that prelabeled training data is only rarely available since not only the training and application of classification methods but also the manual labeling process are much more time-consuming in 3D. This paper focuses on the more classical approach of first calculating features and subsequently applying a classification algorithm. Existing handcrafted feature definitions are enhanced by using multiple selected reductions of the point cloud as approximations. This serves as input to train a well-studied random forest classifier. A comparison to a recently presented deep learning approach, i.e., the Kernel Point Convolution method, reveals that there are well-justified applications for both modern and classical machine learning methods. To enable the smooth conversion of existing 3D scenes to semantically labeled 3D point clouds the tool *Blender2Helios* is presented. We show that the therewith generated artificial data is a good choice for training real-world classifiers.

1 Introduction

Semantic classification in 3D point clouds is the process of assigning semantic labels to each 3D point in a cloud. Some major differences to the same task for 2D images exist. Besides the considerably high volume of data, it lacks a fixed grid-structure. This makes it challenging to apply common algorithms for 2D images on point clouds. The variable density of the data is another difficulty. It highly depends on the distance to the laser scanner and is influenced by occlusions.

A few years ago a typical approach for classification in point clouds was a two-step procedure. First, hand-crafted local features of each point were calculated. For this, Principal Component Analysis (PCA) was applied on a number of neighbors. Originally, the size of this neighborhood, respectively the number of neighbors was fixed. In 2014 an improvement with varying neighborhood sizes, called *optimal neighborhood size selection* was proposed [1]. Up until 2018 various

* The autors would like to acknowledge Prof. Bernhard Höfle and the 3DGeo Research Group of Heidelberg University for the software HELIOS and Prof. Thomas P. Kersten from Hafencity University of Hamburg for supporting our work.

other improvements were developed and published. For instance, [2,3] proposed the calculation of features on many different scales but with an adapted neighborhood definition. Second, common classification algorithms were applied on the features. While basically any general classifier is applicable, random forests showed high accuracies and became a de-facto standard.

With the availability of datasets with billions of labeled 3D points like *Semantic3D* [4] and capable hardware, deep learning emerged in this field of research. Thus, the manual definition of features became redundant. Some early approaches projected 3D points on 2D planes to create artificial images on which existing methods were applied. Meanwhile, a direct processing of 3D data is done most often. The published solutions range from *superpoint graphs* [5] to *RandLA-Net* [6] and *KPConv* [7]. The latter approach, which is very similar to well-known convolutional neural networks, is used in this paper for comparisons. Publicly available benchmark tests like the Semantic3D challenge [8] provide an overview of the methods that currently achieve the highest accuracies.

The current development towards deep learning for semantic labeling of 3D point cloud data raises the question, if and how much deep learning methods outperform classical approaches. Thus, this paper describes an improved approach using classical hand-crafted features. We propose the calculation of features on selected point cloud reductions as a high-performance approximation. A random forest is afterwards used for the classification. Finally, the results are compared to the output of KPConv, a proponent of the deep learning classifiers. For representative results, all parameters of the learners are tuned on only one dataset, a part of Semantic3D. Their performance is reviewed on a total of four different datasets. This resembles typical applications and differentiates this work from others. The used datasets consist of terrestrial (TLS) and mobile laser scans (MLS) of outdoor scenes.

One set of scans, the *Sim2Real* dataset, allows the examination of classifiers that were only trained with simulated data. For this purpose, we present the open-source software *Blender2Helios* [9] that allows to create semantically labeled training data of existing 3D scenes.

The paper is structured as follows. The next section summarizes random forests as the technical basis for the feature learning approach used in this paper. The following section introduces the tools and datasets used in the evaluation. Section 4 details the training of the classifier proposed in this paper while Section 4.3 introduces the related work used as comparison in the following evaluation in Section 5.

2 Random Forest Classification

Random forests (RFs) are a special form of bagging of decisions trees. We will use the definition of [10] for our work with RFs as it is very common. *Bagging*, short for *bootstrap aggregating*, is used to improve accuracy and stability. Multiple training sets, or bags, are sampled from the whole set of training data. They all have the size of the original training set. By sampling uniformly with

replacement, $1 - \frac{1}{e} \approx 63.2\%$ of each subset are unique examples while the rest are duplicates. In each training iteration one set is used to train a decision tree. In the task of classification each tree then votes for its predicted class.

Compared to simply bagging of decision trees, only a subset of all attributes, typically of size $\lceil \sqrt{d} \rceil$ or $\lfloor \log_2 (d) + 1 \rfloor$ for d attributes, is taken into account for every split decision when training a random forest. This allows the algorithm to create more varying trees.

2.1 Parameters in our RF Implementation

The number of trees in a forest, is a main parameter when constructing a RF. Probst and Boulesteix show in a benchmark study on over 300 datasets that the first 100 trees in a forest achieve the highest performance improvement [11]. In [12] different random forest implementations are tested on five datasets. The average classification accuracy in all their tests is within a one percent range when changing the parameter for the number of trees from 100 to 2,000. Meanwhile, using 100 trees has become a default setting for many random forest implementations that we also stick with.

Another important parameter is the tree depth d_t. It limits the length of a path from the root to all leaf nodes. While a high depth may result in overfitting, a too small depth leads to underfitting. Generally each tree should be able to predict all classes of a data set which gives a constraint of $d_t \geq \lceil \log_2 |C| \rceil - 1$ for a number of $|C|$ classes. Often this parameter is optimized together with the number of trees in a grid search.

2.2 Performance Evaluation Strategy

In our approach we calculate features for optimal local neighborhoods. In the preferred case one neighborhood will contain roughly the same points for many neighboring 3D points representing the same object. This results in nearly the same feature values for all of these points whereas other instances of similar objects will result in diverse values. For instance, the features of two different cars, both in one scan of a parking lot, might be very different due to various factors. They are influenced by occlusions and by the distance and angle between the laser scanner and the car.

For these reasons and because other strategies result in higher computation time, we use the classical *hold-out* strategy where the dataset is split into training, validation, and test data. Between these sets all scans shall be taken at different scenes. Where this is not feasible, at least very differing scanner positions are preferred.

2.3 Handling of Highly Imbalanced Data

The data that is used to train the random forests later is highly imbalanced. There are factors of 100 and 1,000 respectively between occurances of different

classes – even after deletion of very small classes. This is caused by the natural structure of the scenes. While ground and facades are very dominant in urban scenes, motorcycles, bushes, or trash cans are seen less often. Even if many instances of these infrequent objects are scanned, their small footprint results in a discrimination of their share in the 3D point cloud.

Learning a RF directly on this imbalanced data results in a classifier that has a bias towards these predominant classes. We choose to perform *down-sampling*, so each class in the training set is sampled only n_{min} times with n_{min} being the size of the smallest class. We prefer this over *up-sampling* as many data points of the dominant classes are already redundant and up-sampling of small classes is prone to over-fitting on the specific object instances in the data.

3 Tools and Datasets

The presented approach is evaluated using existing and newly created datasets that are introduced in this section. To help with the difficulty of recording and labeling real world data sets, we propose to enhance the training data by use of simulated scenes. Thus, the tools and methods used to generate this kind of data are first described here.

3DTK — The 3D Toolkit [13] is a toolkit for 3D point cloud processing. This paper is implemented as an extension of the open-source part of 3DTK and contributes the *scan2features* tool. Weka (*Waikato Environment for Knowledge Analysis*) is an open source machine learning software. We use its Attribute-Relation File Format for storing attributes and training of the random forests. Helios, the *Heidelberg LiDAR Operations Simulator*, is a software package for interactive real-time simulation and visualization of laser scanning surveys. Terrestrial, mobile and airborne laser scans are supported. Finally, we use Blender 2.81, an open-source 3D graphics software toolset.

Blender2Helios is a Blender 2.8 add-on that was developed for this work. It builds an interface between Blender and Helios. The add-on is released under GNU GPLv3. Code and documentation are available at [9]. Blender2Helios provides the possibility to build various scenes without much effort or even use existing ones. These scenes are then converted to Helios compatible XML files to be able to perform laser scan surveys subsequently. Also, semantic labels can be assigned easily by the collections feature in Blender.

Many benchmark datasets are available for 2D image classification. Often, modern machine learning classifiers achieve accuracies far over 95%. For 3D point clouds, however, less training data is available. Even well-known datasets often consist of only a few million points whereas a typical terrestrial laser scan already contains over ten million points. Most datasets like the *Oakland* [14] and the *IQmulus & TerraMobilita Contest* [15] datasets were created with mobile laser scanners. By design these are sparse compared to terrestrial laser scans. The datasets used in this paper are the following.

Oakland. The class distribution is given in Table 1. As the classifiers are tuned using another – the Semantic3D – dataset, we choose not to use the given val-

Table 1. Class distribution in the Oakland dataset.

	ID					
Subset	**Scatter/Misc**	**Wire**	**Pole**	**Load/Bearing**	**Facade**	**Total**
Training	14,441	2,571	1,086	14,121	4,713	36,932
Validation	8,485	899	1,441	67,419	13,271	91,515
Testing	267,325	3,794	7,933	934,146	111,112	1,324,310

Table 2. Class distribution in the Semantic3D dataset (reduced-8).

		Number of Points	
ID	**Class Name**	**Training**	**Testing**
-	Unlabeled	156,046,453	11,607,777
1	Man-made Terrain	796,491,240	14,317,509
2	Natural Terrain	480,979,227	10,694,746
3	High Vegetation	135,634,808	4,675,871
4	Low Vegetation	99,971,504	3,021,894
5	Buildings	285,746,986	30,911,206
6	Hard scape	83,466,776	2,384,545
7	Artefacts	51,979,924	233,839
8	Cars	15,609,275	851,942
	Total	2,105,926,193	78,699,329

Figure 1. Real parking lot scene in Würzburg.

idation data. We just use the training data to train the classifiers and the much bigger testing portion to test them.

Paris-rue-Madame. The dataset fully called *Paris-rue-Madame database: MINES ParisTech 3D mobile laser scanner dataset from Madame street in Paris* consists of two laser scans containing exactly 10 million points each. 624 objects are annotated and categorized in 26 classes. *Scan 1_2* are used for training while the other one (*scan 1_3*) is kept for testing. As many classes occur not at all or only rarely in both scans, we additionally eliminate very small classes.

Semantic3D. The highest quantity of training instances used in this research is provided by the *Semantic3D.net* [4] dataset. A static terrestrial laser scanner was used to create dense point clouds of outdoor scenes consisting of over four billion points. The scanned environments include churches, streets, railroad tracks, squares, villages, soccer fields, and castles in Central Europe.

Eight semantic classes were assigned by manual labeling. Table 2 pictures the distribution of class labels for the *reduced-8* data.

Sim2Real. We test the performance on our own data as well. In this paper we present one scene of a parking lot near the Department of Computer Science at the University of Würzburg. To determine the applicability of training classifiers on easy to generate, artificial scenes and using them for real world applications, we use two scenes. The first one was created in Blender using freely

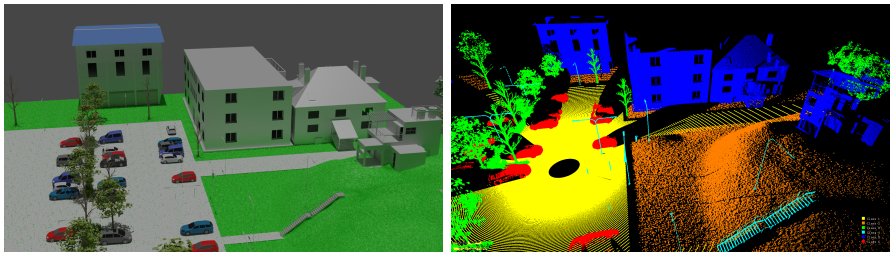

Figure 2. Left: Blender scene created with freely available 3D models similar to Figure 1. Right: Corresponding 3D point cloud generated with a simulated Riegl VZ-400 in Helios. Colors respresent class labels.

Table 3. Semantic classes in our Sim2Real dataset. The dataset is split into training data created with *Helios* and real testing data.

ID	Class Name	Examples	Number of Points	
			Training	Testing
1	Man-made ground	Paved ground, Asphalt	1,120,044	6,381,848
2	Natural ground	Grass, Grassland	128,813	209,347
3	Vegetation	Bushes, Trees	204,011	1,823,701
4	Object	Lamps, Poles, Railings, Stairs	16,900	64,575
5	Facade	Walls, Buildings	280,063	754,867
6	Car	All vehicles	161,480	1,478,190
		Total	1,911,311	10,712,528

available object models. It was then sampled with a simulated laser scanner using Blender2Helios and Helios. Accuracy of the classifier is tested with the manually labeled point cloud of the real parking lot. Figure 1 illustrates the on-site scenery. We use a division into six classes as shown in Table 3.

In Figure 2 one can see the coarsely remodelled scene in Blender and its point cloud representation. For instance, paved/asphaltic ground is modeled by just a flat plane while a Blender *Displacement Modifier* in z direction combined with a *Voronoi texture* is used to generate small spikes that represent grass. As the classifiers show that they have learned the different ground types, no more effort was put into a finer remodeling. One should be able to reproduce our results with his or her own 3D scenes without trying to build an exact copy of the testing data.

4 Semantic Classification

In constrast to [1], we perform feature calculation on multiple differently reduced scans. The process of parameter selection is performed on the Semantic3D dataset. For this, the prelabeled training part is split into two parts. One part, containing twelve of the fifteen scans, is used for training while the remaining three scans (*untermaederbrunnen1*, *untermaederbrunnen3*, and *sg28_4*) are used for validation. The resulting parameters are then used to train and test the

classifiers on all the introduced datasets including the Semantic3D reduced-8 challenge.

We first optimize the feature calculation on the basis of a standard random forest setup with 100 trees and a small tree depth of four. Similar configurations have already been used in literature [1,3] and proven to be sufficient to detect changes generated by different parameters in the feature calculation. Tuning of the RF is done afterwards.

To allow the testing of many configurations, features are calculated on only a reduced number of points. An octree-based reduction with a voxel size of 10 cm is done and features are only calculated for the resulting points. Afterwards, the data is subsampled to bring the classes to a uniform distribution with approximately 63,000 points per class. However, for the calculation of feature values all points are used.

4.1 Finding the Optimal Neighborhood Size

To calculate local features, a number of neighbors must be taken into account. For this set of points the principal components are distinguished using PCA. Typical approaches use a fixed number of k nearest neighbors [16], a sphere with a fixed radius [17], or a cylindrical neighborhood definition [18].

In [1] a state-of-the-art approach which is based on dimensionality features [19] is presented. Instead of defining a fixed neighborhood size, different ways to find an *optimal neighborhood size* k_{Opt} per point are compared. The best results are achieved by choosing a neighborhood size that minimizes the eigenentropy. Compared to the common default of using the nearest $k = 100$ neighbors for feature calculation, this method yields 2% to 14% higher overall accuracy scores.

The eigenentropy E_λ is given by the Shannon entropy of eigenvalues λ_i. Its calculation for the three-dimensional case is:

$$E_\lambda = -\Lambda_1 \ln(\Lambda_1) - \Lambda_2 \ln(\Lambda_2) - \Lambda_3 \ln(\Lambda_3). \tag{1}$$

using normalized eigenvalues to reduce the impact of the local point density:

$$\Lambda_i = \frac{\lambda_i}{\sum_{j=1}^{3} \lambda_j}. \tag{2}$$

When searching for the optimal neighborhood size, one minimizes the eigenentropy. Analyzing Equation (1) reveals that one-dimensional structures are favored, i.e., when one normalized eigenvalue is close to one and the others are close to zero. The maximum entropy of $-\ln(1/3) \approx 1.0986$ is reached by neighborhoods with equally distributed points along all three axes, resulting in $\lambda_1 = \lambda_2 = \lambda_3$ and, accordingly, $\Lambda_1 = \Lambda_2 = \Lambda_3 = 1/3$.

The optimal number of neighbors k_{Opt} is found by repeatedly calculating the eigenentropy of neighborhoods with a varying number of neighbors $k \in \{k_{Min}, k_{Min} + k_\Delta, \ldots, k_{Max}\}$. k_{Min}, k_{Max}, and k_Δ are parameters that need to be set by the user. Common values are $k_{Min} = 10$, $k_{Max} = 100$, and $k_\Delta = 1$.

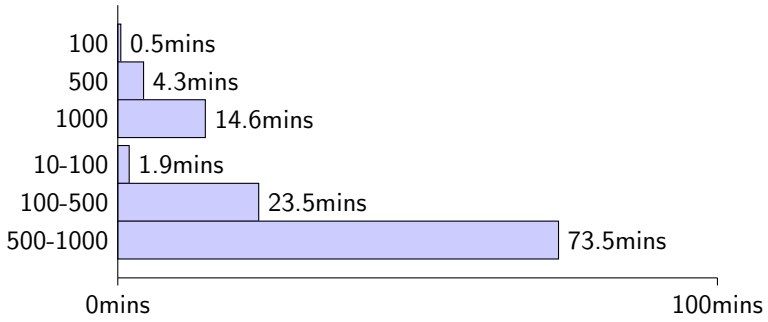

Figure 3. Times for calculating features with different neighborhood sizes using one million 3D points in a typical outdoor scan. The first three measurements use a fixed neighborhood size (100, 500, 1000) whereas for the last three ones our optimal neighborhood definition is used.

Table 4. Impact of varying ranges for finding the optimal neighborhood size. The F_1 scores of our random forest classifier are shown.

	Terrain		**Vegetation**		**Build-**	**Hard**	**Arte-**		**Accu-**
N. size	**Man-m.**	**Nat.**	**High**	**Low**	**ings**	**Scape**	**facts**	**Cars**	**racy**
10-100	**0.916**	0.359	0.749	**0.307**	0.583	0.157	0.227	0.123	64.26%
10-500	0.913	**0.368**	0.749	0.300	0.608	0.153	0.225	0.126	64.49%
100-500	0.822	0.298	**0.789**	0.177	**0.729**	0.139	0.119	0.123	64.58%
500-1000	0.904	0.249	0.783	0.229	0.715	**0.213**	0.355	0.122	**68.36%**
1000-5000	0.880	0.107	0.752	0.289	0.688	0.024	**0.410**	**0.168**	66.46%
10-5000	0.893	0.278	0.749	0.300	0.621	0.144	0.243	0.157	64.16%

This includes the often fixed value $k = 100$ and limits the number of times the eigenvalues have to be calculated per point.

Figure 3 illustrates the time needed to query the neighborhoods of 1 million points. One can see that bigger neighborhoods with 500 or even 1,000 neighbors take longer to find and to perform the PCA on. Especially trying to find optimal neighborhood sizes within big ranges is very time consuming.

Table 4 shows the overall accuracy and class-wise F_1 scores when comparing different optimal neighborhood boundaries. Using between 500 and 1,000 neighbors for a bigger area of influence shows the highest accuracy. However, using that many neighbors for feature calculation is very time consuming as previously presented in Figure 3.

We follow an approximation approach similar to [2]. Instead of calculating features on a large number of neighbors, a smaller number of neighbors is used within a reduced representation of the point cloud. This allows a fast calculation while still taking a wide spatial area into account. Octree-based reductions are used to create less dense point clouds.

$$\boldsymbol{features}(\boldsymbol{p_i}, \boldsymbol{P}) = \begin{bmatrix} \boldsymbol{features}_{\boldsymbol{k}}^{\boldsymbol{3D}}(\boldsymbol{p_i}, \boldsymbol{P}) \\ \boldsymbol{features}_{\boldsymbol{k}}^{\boldsymbol{2D}}(\boldsymbol{p_i}, \boldsymbol{P}) \\ \boldsymbol{features}_{\boldsymbol{r}}^{\boldsymbol{Cyl}}(\boldsymbol{p_i}, \boldsymbol{P}) \end{bmatrix} \qquad \boldsymbol{features}_{\boldsymbol{k}}^{\boldsymbol{2D}}(\boldsymbol{p_i}, \boldsymbol{P}) = \begin{bmatrix} Radius_k^{2D}(\boldsymbol{p_i}, P) \\ LocalDensity_k^{2D}(\boldsymbol{p_i}, P) \\ SumOfEigenvalues_k^{2D}(\boldsymbol{p_i}, P) \\ Ratio_k^{2D}(\boldsymbol{p_i}, P) \end{bmatrix} = \begin{bmatrix} \frac{||\boldsymbol{q_k^{2D}} - \boldsymbol{p_i^{2D}}||}{k} \\ \frac{}{\pi \cdot Radius_k^{3D}(\boldsymbol{p_i}, P)^2} \\ \sum_{i=1}^{2} \lambda_i^{2D} \\ \frac{\lambda_1^{2D}}{\lambda_2^{2D}} \end{bmatrix}$$

$$\boldsymbol{features}_{\boldsymbol{k}}^{\boldsymbol{3D}}(\boldsymbol{p_i}, \boldsymbol{P}) = \begin{bmatrix} Linearity_k^{3D}(\boldsymbol{p_i}, P) \\ Planarity_k^{3D}(\boldsymbol{p_i}, P) \\ Scattering_k^{3D}(\boldsymbol{p_i}, P) \\ Omnivariance_k^{3D}(\boldsymbol{p_i}, P) \\ Anisotropy_k^{3D}(\boldsymbol{p_i}, P) \\ Eigenentropy_k^{3D}(\boldsymbol{p_i}, P) \\ SumOfEigenvalues_k^{3D}(\boldsymbol{p_i}, P) \\ ChangeOfCurvature_k^{3D}(\boldsymbol{p_i}, P) \\ Verticality_k^{3D}(\boldsymbol{p_i}, P) \\ LocalDensity_k^{3D}(\boldsymbol{p_i}, P) \\ Radius_k^{3D}(\boldsymbol{p_i}, P) \\ MaxHeightDifference_k^{3D}(\boldsymbol{p_i}, P) \\ HeightVariance_k^{3D}(\boldsymbol{p_i}, P) \end{bmatrix} = \begin{bmatrix} \frac{\Lambda_1 - \Lambda_2}{\Lambda_1} \\ \frac{\Lambda_2 - \Lambda_3}{\Lambda_1} \\ \frac{\Lambda_3}{\Lambda_1} \\ \sqrt[3]{\Lambda_1 \cdot \Lambda_2 \cdot \Lambda_3} \\ \frac{\Lambda_1 - \Lambda_3}{\Lambda_1} \\ -\sum_{i=1}^{3} \Lambda_i \cdot ln(\Lambda_i) \\ \sum_{i=1}^{3} \lambda_i \\ \Lambda_3 \\ 1 - |(\frac{\boldsymbol{e_3}}{||\boldsymbol{e_3}||})_{vert}| \\ \frac{k}{4/3 \cdot \pi \cdot Radius_k^{3D}(\boldsymbol{p_i}, P)^3} \\ ||\boldsymbol{q_k} - \boldsymbol{p_i}|| \\ \max_{1 \le i \le k}(q_{i,vert}) - \min_{1 \le i \le k}(q_{i,vert}) \\ \frac{1}{k} \sum_{i=1}^{k} (q_{i,vert} - \mu_{vert})^2 \end{bmatrix}$$

$$\boldsymbol{features}_{\boldsymbol{r}}^{\boldsymbol{Cyl}}(\boldsymbol{p_i}, \boldsymbol{P}) = \begin{bmatrix} NrPoints_r^{Cyl}(\boldsymbol{p_i}, P) \\ MaxHeightDifference_r^{Cyl}(\boldsymbol{p_i}, P) \\ HeightVariance_r^{Cyl}(\boldsymbol{p_i}, P) \\ HeightAboveMin_r^{Cyl}(\boldsymbol{p_i}, P) \end{bmatrix} = \begin{bmatrix} k^{Cyl} \\ \max_{1 \le i \le k^{Cyl}}(q_{i,vert}^{Cyl}) - \min_{1 \le i \le k^{Cyl}}(q_{i,vert}^{Cyl}) \\ \frac{1}{k^{Cyl}} \sum_{i=1}^{k^{Cyl}} (q_{i,vert}^{Cyl} - \mu_{vert}^{Cyl})^2 \\ p_{i,vert} - \min_{1 \le i \le k^{Cyl}}(q_{i,vert}^{Cyl}) \end{bmatrix}$$

Figure 4. Features of point $\boldsymbol{p_i}$ in a 3D point cloud P.

4.2 Definition of Multiscale Features

Weinmann et al. also introduce a set of 21 features for every 3D point [1]. The definitions of the special geometric properties that are exploited are based on [20,21]. We slightly change the original feature definition of [1]. While the original features, for instance, include the absolute height of a 3D point, we calculate the height of a point with respect to the lowest point within a cylindrical neighborhood with a fixed radius.

The 21-dimensional feature vector for each point $\boldsymbol{p_i}$ in a point cloud P is given by the equations in Figure 4. λ_i and λ_i^{2D} denote the i-th eigenvalue of the 3D, respectively 2D, covariance matrices of the k neighboring points $\{\boldsymbol{q_1}, \boldsymbol{q_2}, \ldots, \boldsymbol{q_k}\}$ with mean $\boldsymbol{\mu} = \frac{1}{k} \sum_{1 \le j \le k} \boldsymbol{q_j}$ around point $\boldsymbol{p_i}$. The neighbors are ordered by their distance to $\boldsymbol{p_i}$ so that ($||\boldsymbol{q_j} - \boldsymbol{p_i}|| \le ||\boldsymbol{q_l} - \boldsymbol{p_i}||$ for $j \le l$) holds true. Two-dimensional data is created by projecting all the points $\boldsymbol{p_i}$ to two-dimensional points $\boldsymbol{p_i^{2D}}$ on the ground plane by ignoring the vertical axis. The neighbors $\{\boldsymbol{q_1^{2D}}, \boldsymbol{q_2^{2D}}, \ldots, \boldsymbol{q_k^{2D}}\}$ are again ordered by ascending distance to the reference point. We further order the eigenvalues and the corresponding eigenvectors $\boldsymbol{e_i}$ in a way that $\lambda_1 \ge \lambda_2 \ge \lambda_3 \ge 0$ and $\lambda_1^{2D} \ge \lambda_2^{2D} \ge 0$ hold true. Λ_i denotes the normalized equivalents of λ_i again. In all equations the subscript $_{vert}$ means the vertical dimension. For the cylindrical features we further let $\boldsymbol{q_j^{Cyl}}$ with $1 \le j \le k^{Cyl}$ be points within a cylindrical neighborhood with radius r around point $\boldsymbol{p_i}$. $\boldsymbol{\mu^{Cyl}} = \frac{1}{k^{Cyl}} \sum_{1 \le j \le k^{Cyl}} \boldsymbol{q_j^{Cyl}}$ denotes their mean. We also propose the feature calculation with different neighborhood sizes on differently reduced scans.

A prototype of the feature calculation tool *scan2features* is contributed to 3DTK. The four parameters k_{Reds}, k_{Mins}, k_{Maxs}, and k_{Deltas} impact the calculation of feature values. All four parameters hold lists of the same length, here denoted as k_{len}. For each $i \in \{1, \ldots, k_{len}\}$ an octree-reduced scan with

a voxel size of $k_{Reds}[i]$ cm is generated. Only one randomly chosen point is kept per voxel. Within this cloud, denoted as $P_{k_{Reds}[i]}$, the 3D and 2D features ($\boldsymbol{features_k^{3D}(p_i, P)}$ and $\boldsymbol{features_k^{2D}(p_i, P)}$) of an optimal neighborhood between $k_{Mins}[i]$ and $k_{Maxs}[i]$ with a step size of $k_{Deltas}[i]$ are calculated. One defines equivalent scan reductions (cyl_{Reds}) and radii (cyl_{Radii}) of the same length used for the calculation of the cylindrical portion of the feature vector ($\boldsymbol{features_r^{Cyl}(p_i, P)}$). All the feature values calculated in different reductions are concatenated to one feature vector at the end.

The best parameters for the approximate feature calculation are found as follows: A number of different reduction parameters are tested with $10 \leq k_{Opt} \leq 100$. Class-wise F_1 scores and accuracies are printed in Table 5. One can see that different reductions allow different classes to be recognized better. Voxel sizes over 80 cm seem to impact the accuracy negatively.

Using three different reductions shows a reasonable tradeoff between classification accuracy and computation time. Calculating features on point clouds reduced with voxel sizes of 2, 20, and 80 cm shows the best result with 72.42% accuracy. More reductions are imaginable but show only small improvements while leading to higher computation times.

Figure 5 illustrates the varying optimal neighborhood sizes in a whole laser scan for different reduction parameters. While in a detailed scan (2 cm voxel size) especially thin poles and sharp edges on cars show a high number of optimal neighbors, on coarser reduction scales other areas prefer big neighborhoods. Interestingly mainly medium-sized objects like bushes and cars favor a high number of neighbors in a scan reduced with a voxel size of 80 cm. Figure 6 (left and center) shows that the distribution of different neighborhood sizes is similar across all three reductions with a small bias towards big neighborhoods in highly reduced scans.

Table 5. Left: Impact of different and multiple reduction scales for feature calculation. The F_1 scores and accuracies of our random forest classifier are shown. Right: Impact of multiple cylindrical features and more trees in the RF.

Reduction(s), k_{Reds}	**Terrain**		**Vegetation**		**Buildings**	**Hard Scape**	**Artefacts**	**Cars**	**Accuracy**
	Man-m.	**Nat.**	**High**	**Low**					
0	0.946	0.356	0.758	0.335	0.578	0.011	**0.394**	0.132	**66.47%**
1	**0.947**	0.349	0.739	**0.337**	0.547	0.046	0.385	0.129	65.24%
2	0.939	**0.394**	0.744	**0.337**	0.571	0.047	0.363	0.141	65.68%
5	0.932	0.390	0.743	0.330	0.580	0.052	0.265	0.134	65.21%
10	0.916	0.359	0.749	0.307	0.583	**0.157**	0.227	0.123	64.26%
20	0.873	0.374	0.755	0.260	**0.708**	0.139	0.189	0.176	66.41%
40	0.848	0.315	**0.765**	0.187	**0.708**	0.135	0.166	0.164	65.20%
80	0.895	0.277	0.741	0.224	0.679	0.071	0.177	0.191	66.32%
160	0.767	0.048	0.688	0.258	0.618	0.028	0.220	**0.205**	58.31%
2, 10	0.923	0.431	0.764	**0.329**	0.603	0.064	0.350	0.135	65.84%
2, 20	0.933	**0.468**	0.777	0.318	0.627	0.070	0.324	0.130	67.34%
2, 80	**0.938**	0.371	**0.789**	0.325	**0.697**	**0.081**	**0.362**	**0.225**	**70.76%**
2, 10, 80	0.926	0.377	0.805	0.343	0.735	0.152	**0.382**	**0.212**	71.99%
2, 20, 80	**0.931**	**0.429**	**0.812**	0.321	0.740	0.122	0.379	0.196	**72.42%**
2, 8, 32	0.920	0.411	0.774	**0.385**	0.729	**0.158**	0.359	0.186	69.51%
5, 20, 80	0.926	0.402	0.804	0.340	**0.744**	0.125	0.317	0.202	72.09%
10, 20, 40	0.917	0.424	0.791	0.306	0.733	0.154	0.247	0.197	70.46%

#Trees	**Cylindrical Features: cyl_{Reds} (cyl_{Radii}) in cm**
A 100	10 (25)
B 100	2, 20, 80 (25, 150, 500)
C 500	2, 20, 80 (25, 150, 500)

	Weighted Avg. F_1	**Accuracy**
A	0.752	72.42%
B	**0.786**	**76.81%**
C	0.784	76.61%

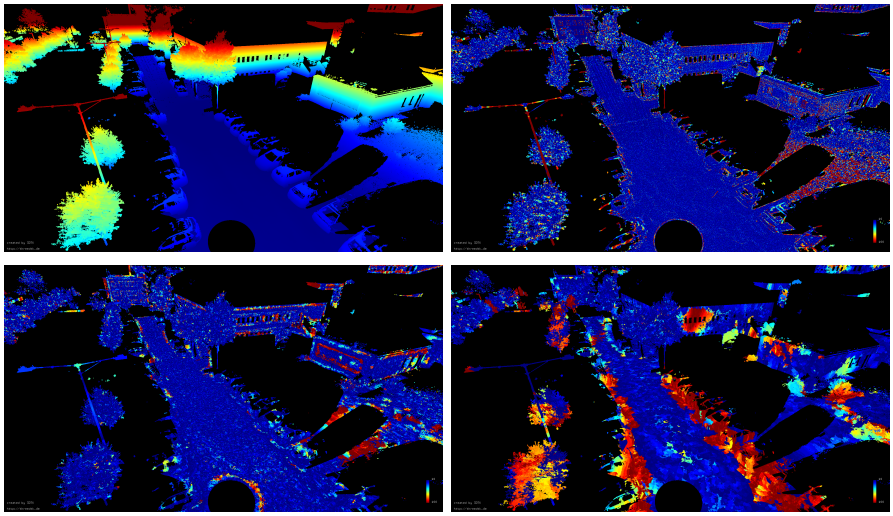

Figure 5. Illustration of optimal neighborhood sizes k_{Opt} on differently reduced scans. The color gradient indicates the found optimal neighborhood size from $k_{Min} = 10$ (blue) to red $k_{Max} = 100$ (red). From top left to bottom right: Reference image colored by point height. Reduction with 2 cm voxel size. Reduction with 20 cm voxel size. Reduction with 80 cm voxel size.

For the previously shown results a standard RF setup with 100 trees and a tree depth of four are used. After determining a good set of features, the classifier itself needs to be tuned. The selected evaluations in Table 5 prove that our expectations are correct and more than 100 trees do not significantly impact the accuracy of the classifiers. The table also shows an example of the impact of calcuting multiple cylindrical features on point cloud reductions. Calculating those features on three different scan reductions (with voxel-sizes of 2, 20, and 80 cm and cylinder radii of 25, 150, and 500 cm) increases the classification accuracy by over 4% compared to only one cylindrical feature set. A more detailed analysis can be found in [22]. Therefore, the only additionally tuned parameter is the tree depth. Figure 6 (right) shows its influence. The overall accuracy increases from 75.96% (depth of 3) to 81.95% (depth of 63). The maximum of 63 is given by the number of 21 attributes we calculate on three different reductions. To avoid overfitting we keep the trees small and pick a tree depth of 20 for the later comparison. It corresponds to 81.59% overall accuracy on our validation data.

4.3 Related Work: Deep Learning for 3D Point Cloud Classification

There are different approaches of using deep learning algorithms for classifying 3D point clouds. Thomas et al. assign these into four different categories [7].

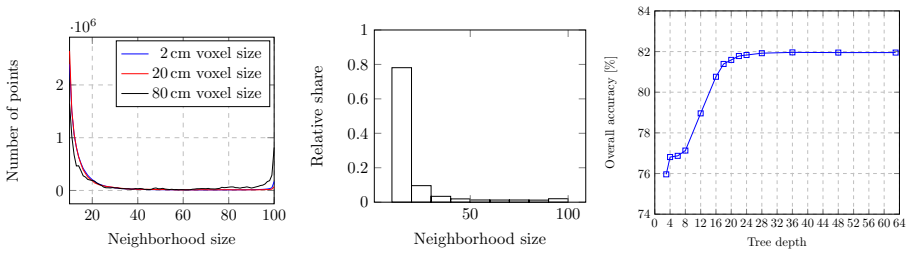

Figure 6. Distribution of optimal neighborhood sizes k_{Opt} on different reduction scales. Input was the outdoor scan already shown in Figure 5. Left: Absolute distribution of k_{Opt} on different reduction scales. Middle: Histogramm showing the relative distribution of k_{Opt} sizes in the scan reduced with voxel size 20 cm. Right: Influence of the tree depth on the random forest's accuracy.

For comparison we chose Kernel Point Convolution (*KPConv*) which belongs to the category of *point convolution networks*. Other well-known representatives are *Pointwise CNN* [23] and *SpiderCNN* [24].

KPConv operates directly on point clouds without any intermediate representation. Due to the lack of a fixed grid structure, the convolution weights of KPConv are located in Euclidean space by so-called *kernel points*. These are applied on the input points close to them.

The authors describe their *KP-FCNN* as the applicable method for our pointwise classification task in 3D point clouds. It implements an encoder-decoder CNN consisting of convolutional blocks. First, higher-order features are calculated step-by-step on reductions of the point cloud. Then, the original detailedness is achieved by nearest upsampling in the decoder part. Skip links are used to transfer information between layers, skipping the network parts in between.

5 Results and Comparison

In this section we show a comparison of the different classifiers applied on the four datasets. The settings of the feature calculation procedure are as previously shown: $k_{Reds} = cyl_{Reds} = (2, 20, 80)$ and $cyl_{Radii} = (25, 150, 500)$ with between 10 and 100 optimal neighbors. Random forests with 100 trees and a tree depth of 20 are trained.

5.1 Classification Results

Oakland. Applied on the Oakland dataset our RF classifier shows an accuracy of 95.27% which is around 3% higher than the results using the original single-scale approach described in [1]. Our approach also minimizes the assumptions made, like a consistent height of the laser scanner. The detailed result of our random forest as well as class-wise F_1 scores and the overall accuracies of the

Table 6. Left: Performance of our RF on the Oakland dataset. Overall accuracy: 95.27%. Right: Comparison of F_1 scores and accuracies of the classifiers on the Oakland dataset.

Our ID	Orig. ID	Class Name	Prec.	Recall	F-Meas.
1	1004	Scatter/Misc	0.953	0.922	0.937
2	1100	Wire	0.101	0.894	0.181
3	1103	Pole	0.376	0.751	0.501
4	1200	Load/Bearing	0.999	0.994	0.996
5	1400	Facade	0.893	0.696	0.782
Weighted Avg.			0.974	0.953	0.961

Classifier	Class F_1 Score 1	2	3	4	5	Over. Accur.
Original RF	0.878	0.165	0.364	0.978	0.749	92.2%
Our RF	0.937	0.181	**0.501**	**0.996**	0.782	95.3%
KPConv	**0.966**	**0.186**	0.429	0.992	**0.813**	**95.7%**

Table 7. Comparison of F_1 scores and accuracies of the classifiers on the Paris-rue-Madame dataset.

Classifier	Class F_1 Score 1	2	3	4	5	6	Overall Accuracy
Original RF	0.960	0.932	0.672	**0.036**	0.206	0.105	90.1%
Our RF	0.983	0.965	0.897	0.000	0.202	0.304	96.4%
KPConv	**0.989**	**0.988**	**0.972**	0.000	**0.733**	**0.767**	**98.6%**

different classifiers are printed in Table 6. KPConv performs slightly better than our RF approach. It has to be noted that this dataset only contains 1,086 points belonging to class *pole*. Due to sub-sampling, our RF classifier is therefore only trained on 5,430 equally distributed points.

Paris-rue-Madame. Due to the small number of pedestrians in the training data, the data is subsampled to 3,656 examples per class for our RF classifier. Compared to the original paper we achieve an over 6% higher accuracy of 96.44% as presented in Table 7. Another 2% are achieved by KPConv.

Figure 7 shows the outputs of our random forest and KPConv after applying them to the depicted test data. Region A in the figure shows an area where the high recall value belonging to the class *cars* of our random forest classifier can be seen. However, also the points belonging to the ground around the cars are assigned the same label. In constrast, KPConv labels cars more conservatively leading to a higher precision score but introducing some errors where *facades* are predicted. The traffic sign in region B is detected with a high accuracy by both methods. The motorcycle in area C is not detected by either classifier. KPConv, however, detects other motorcycles that are not presented in the illustration. The pedestrian in sector D is ignored by both approaches.

Especially the bad matching of pedestrians has to be noted. Precision and recall values are zero for this class as shown in Table 8. We interpret this as an over-fitting on the seen object instances due to the fact that only three pedestrians are part of the training data.

Sim2Real. The simulated laser scan created with Blender, Blender2Helios, and Helios are subsampled to 16,900 examples per class for training our RF. Table 9

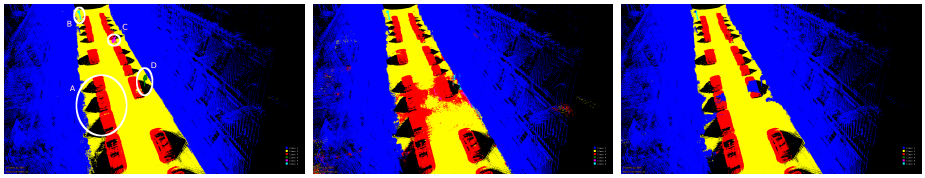

Figure 7. Paris-rue-Madame labeled test data for reference. Colors indicate class affiliations. *Blue*: facade, *yellow*: ground, *red*: cars, *dark green*: pedestrians, *rose*: motorcycles, *light blue*: traffic signs. Left: Ground truth. White circles indicate regions that are later discussed in the text. Middle: Output of our random forest classifier. Right: Output of KPConv.

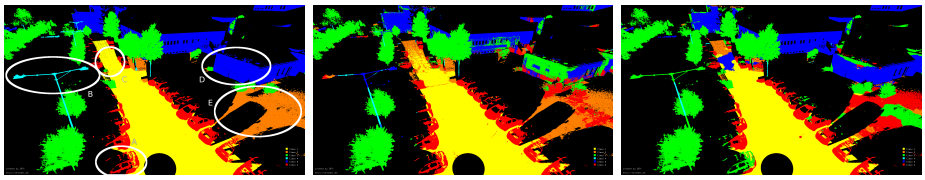

Figure 8. Sim2Real labeled test data for reference. Colors indicate class affiliations. *Yellow*: man-made ground, *orange*: natural ground, *green*: vegetation, *light blue*: object, *dark blue*: facade, *red*: car. Left: Semantic labels in the Sim2Real test data. Annotation was done manualy. Middle: Output of our random forest classifier. Right: Output of KPConv.

shows the detailed performance of our RF classifier. Evaluation is done with the manually labeled real scan from the parking lot in Würzburg. Results of the different classifiers are shown in Table 9. Our RF and KPConv reach the same overall accuracy of 94.21%.

Figure 8 shows an excerpt of the test data. Region *A* in the figure shows a car that is correctly classified by us, but is treated like vegetation by KPConv. The recognition of area *B*, the upper part of a street light, is incorrectly classified by both methods. We have no explanation why KPConv detects a facade in sector *C* which is paved ground. However, in part *D*, KPConv detects the building with a much higher accuracy. Area *E* is probably the most difficult section. The grassland has an irregular incline and is peppered with smaller plants. Big parts of it are correctly classified by the random forest. KPConv, in constrast, heavily labels the points as cars or vegetation.

Semantic3D Challenge. Semantic3D is the dataset containing the most points for training and testing.

To tune the parameters for our RF, an additional split of the supplied training data is used. Now for testing all the prelabeled data is used to create an equally distributed set of features to train a classifier with best performance.

KPConv is already provided with a training procedure for the Semantic3D challenges. As the authors of [7] already put much effort into tuning, this is just

Table 8. Performance of our RF on the Paris-rue-Madame dataset. Overall accuracy: 96.44%.

Our ID	Orig. ID	Class Name	Precision	Recall	F-Measure
1	1	Facade	0.984	0.982	0.983
2	2	Ground	0.994	0.937	0.965
3	4	Cars	0.816	0.995	0.897
4	9	Pedestrians	0.000	0.000	0.000
5	10	Motorcycles	0.291	0.154	0.202
6	14	Traffic signs	0.200	0.635	0.304
Weighted Avg.			0.968	0.964	0.965

Table 9. Left: Performance of our RF classifier on the Sim2Real dataset. Overall accuracy: 94.21%. Right: Comparison of F_1 scores and accuracies of the classifiers on Sim2Real.

ID	Class Name	Prec.	Recall	F-Meas.
1	Manmade Ground	1.000	0.973	0.986
2	Natural Ground	0.784	0.762	0.773
3	Vegetation	0.876	0.961	0.916
4	Lamps/Railings	0.893	0.304	0.454
5	Buildings/Walls	0.959	0.650	0.775
6	Cars	0.827	0.986	0.900
Weighted Avg.		0.947	0.942	0.940

	Class F_1 Score						Over.
Classifier	1	2	3	4	5	6	Accur.
Our RF	0.986	**0.773**	**0.916**	**0.454**	0.775	**0.900**	**94.2%**
KPConv	**0.990**	0.364	0.888	0.373	**0.946**	0.876	**94.2%**

adapted to our needs. The only change is the removal of the color information to allow a fair comparison of the classifiers.

The results of our submissions to the Semantic3D reduced-8 challenge are given in Table 10. For comparison the original KPConv results that use color information are also depicted. F_1 scores and accuracies are shown in Table 11. One can see that KPConv provides better results than our RF on this large dataset.

5.2 Run Times

For the comparison of run times we use a *Lenovo Yoga 720-15IKB* that is equipped with an *Intel Core i5-7300HQ* CPU (4x 2.50 GHz) and 24 GB of RAM. For KPConv an *Asus GeForce RTX 2080 Ti ROG STRIX* with 11 GB of video memory is plugged into a *Razer Core X* external GPU graphics card enclosure. The Thunderbolt connection is a small bottleneck and allows only for utilizing the GPU up to approximately 70-80%. Of course, modern workstations will perform better than our hardware. However, the times shown in Table 12 already provide a coarse reference.

The numbers indicate that calculating handcrafted features and training a random forest is much faster than the neural network approach. It has to be mentioned that the parameters for KPConv are optimized for the huge Semantic3D

Table 10. Results of the Semantic3D reduced-8 submissions: IoU.

	Colors	Intersection over Union for Class								Average
Classifier	Used?	1	2	3	4	5	6	7	8	IoU
Our RF	No	0.886	0.839	0.595	0.371	0.861	0.187	0.203	0.398	0.542
KPConv	No	**0.982**	**0.905**	0.814	0.373	0.943	0.319	0.609	**0.824**	0.721
KPConv	Yes	0.909	0.822	**0.842**	**0.479**	**0.949**	**0.400**	**0.773**	0.797	**0.746**

Table 11. Results of the Semantic3D reduced-8 submissions: F_1 scores and accuracies.

	Colors	Class F_1 Score								Overall
Classifier	Used?	1	2	3	4	5	6	7	8	Accuracy
Our RF	No	0.940	0.912	0.746	0.541	0.926	0.315	0.337	0.569	86.49%
KPConv	No	**0.991**	**0.950**	0.897	0.543	0.971	0.484	0.757	**0.903**	**93.51%**
KPConv	Yes	0.952	0.902	**0.914**	**0.648**	**0.974**	**0.571**	**0.872**	0.887	92.86%

dataset. By changing the learning rate and adapting the stop criterion a speedup of the KPConv learning process seems likely. However, it can be seen that the RF approach adapts better to varying dataset sizes. Application of our classifier on a point cloud with 10 million points takes around 1 hour for feature calculation and 9 minutes to apply the RF.

6 Conclusions and Future Work

The main objective of this research is the comparison of different approaches for semantic classification in 3D point clouds. One can see that classical methods with handcrafted features still have a right to exist. With our optimization of using multiple reduced scans to approximate different neighborhood sizes they show very similar results compared to the deep learning approach on small- to medium-sized datasets. Their short runtime for training makes them an excellent choice for creating a good baseline or for choosing a reasonable set of training data. Due to their low hardware requirements, they can be used on cheap and even mobile hardware.

On the big *Semantic3D* dataset the deep learning approach performs better. We justify this by its capability of learning a more complex model. This comes with the downside of long training times and increased hardware prerequisites.

With the development of *Blender2Helios*, an interface between an easily usable 3D suite and the LiDAR simulation software *Helios* is created. Due to the good compatibility of *Blender*, many existing scenes can now be converted to 3D point clouds. As labeling in *Blender* is much easier than in the point cloud, we hope to promote the creation of highly detailed and labeled 3D point clouds. In this paper we show with the *Sim2Real* dataset that artificial data can be used to train classifiers that are later used in real-world applications. We will dive

Table 12. Training times of our RF and the KPConv classifier.

Number of Points	Step	Time Taken (Approx.) Our RF[1]	KPConv
2 million	Feature calculation	40 seconds (100,000)	7 hours
	Classifier training	40 seconds (100,000)	
10 million	Feature calculation	3 minutes (400,000)	13.5 hours
	Classifier training	4 minutes (400,000)	

[1] The numbers in brackets show the number of points that we are able to choose equally distributed from the whole scan for training of the RF. However, all points have impact on the calculation of feature values.

deeper into this in future work and hope to see some applications of our tool in future publications.

Needless to say, a lot of work remains to be done. Instead of down-sampling the training data, other preprocessing strategies might positively impact the accuracy of our approach. Also many different neighborhood definitions and sets of handcrafted features exist. It is conceivable that other combinations achieve higher accuracies than the ones we have tested.

References

1. M. Weinmann, B. Jutzi, and C. Mallet, "Semantic 3d scene interpretation: A framework combining optimal neighborhood size selection with relevant features," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 2, no. 3, p. 181, 2014.
2. T. Hackel, J. Wegner, and K. Schindler, "Fast semantic segmentation of 3d point clouds with strongly varying density," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2016.
3. H. Thomas, F. Goulette, J.-E. Deschaud, and B. Marcotegui, "Semantic classification of 3d point clouds with multiscale spherical neighborhoods," in *Proc. of the IEEE Intl. Conference on 3D Vision (3DV '18)*, 2018.
4. T. Hackel, N. Savinov, L. Ladicky, J. D. Wegner, K. Schindler, and M. Pollefeys, "Semantic3d. net: A new large-scale point cloud classification benchmark," *arXiv preprint arXiv:1704.03847*, 2017.
5. L. Landrieu and M. Simonovsky, "Large-scale point cloud semantic segmentation with superpoint graphs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4558–4567.
6. Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham, "RandLA-Net: Efficient semantic segmentation of large-scale point clouds," *arXiv preprint:1911.11236*, 2019.
7. H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "KPConv: Flexible and deformable convolution for point clouds," in *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV '19)*, 2019.
8. T. Hackel, N. Savinov, L. Ladicky, J. D. Wegner, K. Schindler, and M. Pollefeys.
9. M. Neumann, "Blender2Helios - github repository," https://github.com/neumicha/Blender2Helios, 2020.

10. L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.
11. P. Probst and A.-L. Boulesteix, "To tune or not to tune the number of trees in random forest," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6673–6690, 2017.
12. M. Amrehn, F. Mualla, E. Angelopoulou, S. Steidl, and A. Maier, "The random forest classifier in Weka: Discussion and new developments for imbalanced data," *arXiv preprint arXiv:1812.08102*, 2018.
13. A. Nüchter, D. Borrmann, and J. Schauer, "3DTK - the 3d toolkit," http://threedtk.de, 2019, accessed: 2020-02-01.
14. D. Munoz, J. A. Bagnell, N. Vandapel, and M. Hebert, "Contextual classification with functional max-margin markov networks," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 975–982.
15. B. Vallet, M. Brédif, A. Serna, B. Marcotegui, and N. Paparoditis, "Terramobilita/iqmulus urban point cloud analysis benchmark," *Computers & Graphics*, vol. 49, pp. 126–133, 2015.
16. J.-F. Lalonde, N. Vandapel, D. F. Huber, and M. Hebert, "Natural terrain classification using three-dimensional ladar data for ground robot mobility," *Journal of field robotics*, vol. 23, no. 10, pp. 839–861, 2006.
17. I. Lee and T. Schenk, "Perceptual organization of 3d surface points," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 34, no. 3/A, pp. 193–198, 2002.
18. S. Filin and N. Pfeifer, "Neighborhood systems for airborne laser data," *Photogrammetric Engineering & Remote Sensing*, vol. 71, no. 6, pp. 743–755, 2005.
19. J. Demantké, C. Mallet, N. David, and B. Vallet, "Dimensionality based scale selection in 3d lidar point clouds," *The Intl. Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2011.
20. A. Toshev, P. Mordohai, and B. Taskar, "Detecting and parsing architecture at city scale from range data," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2010, pp. 398–405.
21. C. Mallet, F. Bretar, M. Roux, U. Soergel, and C. Heipke, "Relevance assessment of full-waveform lidar data for urban area classification," *ISPRS journal of photogrammetry and remote sensing*, vol. 66, no. 6, pp. S71–S84, 2011.
22. M. Neumann, "Semantic classification in uncolored 3D point clouds using multiscale features," Master's thesis, Julius Maximilian University of Würzburg, 2020.
23. Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D ShapeNets: A deep representation for volumetric shapes," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR '15)*, 2015.
24. Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao, "SpiderCNN: Deep learning on point sets with parameterized convolutional filters," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 87–102.