24th European Conference on Artificial Intelligence - ECAI 2020
Santiago de Compostela, Spain

Point Cloud Segmentation with Deep Reinforcement Learning

Marcel Tiator[1] and **Christian Geiger**[1] and **Paul Grimm**[2]

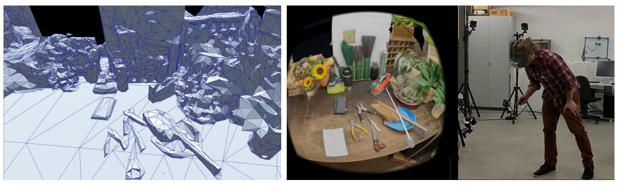

Figure 1: The left figure shows a 3D reconstructed indoor scene as a wire-frame mesh. The right figure shows a user who is exploring this indoor scene in VR. The scene was captured by photos and laser scanner recordings. The 3D reconstruction consists of one mesh and to interact with single objects, the scene has to be segmented. Thus, the scene can only be viewed in VR without any further processing after the reconstruction.

Abstract. The segmentation of point clouds is conducted with the help of deep reinforcement learning (DRL) in this contribution. We want to create interactive virtual reality (VR) environments from point cloud scans as fast as possible. These VR environments are used for secure and immersive trainings of serious real life applications such as the extinguishing of a fire. It is necessary to segment the point cloud scans to create interactions in the VR. Existing geometric and semantic point cloud segmentation approaches are not powerful enough to automatically segment point cloud scenes that consist of diverse unknown objects. Hence, we tackle this problem by considering point cloud segmentation as markov decision process and applying DRL. More specifically, a deep neural network (DNN) sees a point cloud as state, estimates the parameters of a region growing algorithm and earns a reward value. The point cloud scenes originate from virtual mesh scenes that were transformed to point clouds. Thus, a point to segment relationship exists that is used in the reward function. Moreover, the reward function is developed for our case where the true segments do not correspond to the assigned segments. This case results from, but is not limited to, the usage of the region growing algorithm. Several experiments with different point cloud DNN architectures such as PointNet [13] are conducted. We show promising results for the future directions of the segmentation of point clouds with DRL.

1 Introduction

The usage of virtual reality (VR) trainings such as in [16] has a lot of benefits. Serious situations can be trained in secure and immersive VR environments. To create immersive experiences, the 3D content has to be created for the VR environments. Usually, the content consists of objects that are used for the interaction, non active objects and the background environment. Real world environments can be captured by a laser scanner or a camera, reconstructed from a point cloud and used as content. After the acquisition of the content, some objects of interest have to be made interactive. To realise the interaction with objects, certain physical properties have to be assigned to them. If the content is acquired by sensors from a real scene, the objects are not separated and the scene has to be segmented for the assignment of the physical properties and scripting processes. See Figure 1 for an illustration of this problem.

The automation of the segmentation would save a lot of time as the production of an interactive VR scene has a lot of steps such as scanning, segmentation, reconstruction via triangulation, cleaning of the reconstruction, texturing, rigging and scripting. There are several geometric point cloud segmentation algorithms [5] such as region and edge based methods. While the former suffer from the selection of the initial seed points, the latter are susceptible to noise. Model fitting methods such as Random Sample Consensus do not work well in case of complex shapes. When applying a clustering algorithm such as K-Means, an appropriate K has to be selected. The same applies for the P-Linkage algorithm [10], where a scale parameter has to be set by the user. Thus, the point cloud segmentation algorithms only fit to specific use cases or need the input of certain expert parameters.

The usage of a deep neural network (DNN) for the segmentation of point clouds could be promising as there are a lot of supervised approaches with semantic segmentation [9, 30, 12] and object classification [27, 3, 12]. However, a semantic segmentation or object

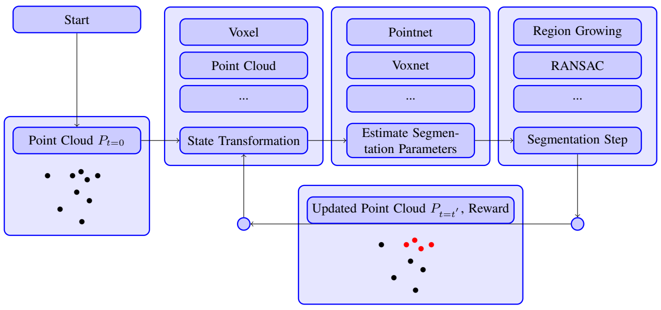

Figure 2: Schema of the proposed DRL point cloud segmentation framework. The point cloud $P_{t=0}$ at time step $t = 0, t \in \mathbb{N}$ is given as state representation. The segmentation parameters are estimated as action by a DNN agent. The segmentation is applied by a geometric segmentation algorithm. After that, the updated point cloud $P_{t>0}$ and the reward are given back to the agent till the point cloud is segmented.

[1] University of Applied Sciences Düsseldorf, Germany, email: {marcel.tiator,geiger}@hs-duesseldorf.de
[2] University of Applied Sciences Fulda, Germany, email: paul.grimm@informatik.hs-fulda.de

classification point cloud DNN cannot recognise objects accurately that are not learned in the training phase. Therefore, we tackle the segmentation problem by deep reinforcement learning (DRL) in this contribution which consists of:

- A theoretical DRL point cloud segmentation framework with a first approach as realisation.
- A framework to produce virtual test data in order to train different models.
- A reward function which rates a true and an assigned segment vector for the case of different segment numbers for the same object.
- Experiments with different point cloud network architectures[3].

The document is structured as follows. In Section 2, the related work is reviewed briefly. The considered point clouds and the segmentation problem are described in Section 3. Our training environment and the reward function are described in Section 4. Subsequently, we describe some experiments with different DNNs in this environment and the results in Section 5. The whole approach is discussed in Section 6 followed by future developments that are described in Section 7.

2 Related Work

2.1 Creation of 3D Environments

3D environments can be created by the reconstruction from 3D point clouds. The point cloud acquisition can be done via photogrammetry, 3D laser scanning, RGB-D scanning, videogrammetry and stereo camera scanning. According to Wang et al. [24], the most accurate procedures are 3D laser scanning and photogrammetry. Currently, we use this techniques to create interactive 3D environments such as in Figure 1. These methods produce large dense point clouds with about 10^7 points which we would like to segment offline. In contrast to our offline approach, Valentin et al. [22] introduced an online semantic segmentation tool, called SemanticPaint, which works with a RGB-D camera. A user can scan and label an environment interactively. Steinlechner et al. [19] developed an interactive system for point cloud segmentation, too. They developed different tools such as a lasso and a brush such that a user can segment the objects of interest. We can imagine to combine interactive solutions such as in [22] or [19] with the proposed segmentation framework to segment point cloud scenes as fast and accurate as possible in case of inaccurate algorithmic segmentations.

2.2 Neural Nets

There are different approaches to input a point cloud into a neural net. View based methods project the 3D data into a 2D representation such that 2D image processing can be applied. For instance, Yavartanoo et al. [27] used a stereographic projection of a 3D mesh model as input of a CNN. Chen et al. [3] developed a model that selects the best perspective projection of a mesh for a 3D classification task. They used the REINFORCE algorithm [25] to train a model for the projection selection. In comparison to our approach, they also used DRL in combination with point clouds.

Volumetric methods transform the 3D point cloud into a voxel grid. The authors of [11, 26] used binary voxel grids to apply 3D CNNs. Additionally, more features can be assigned to a voxel such as the 2D equivalent of adding RGB values to a greyscale image. Liu et al. [9] added RGB values to a voxel. They averaged the colour values of the points that lie within a voxel and assigned this mean colour to it. Moreover, they also used DRL to steer an eye window that samples voxels from a point cloud to conduct a semantic segmentation. In contrast to our work, the task of the DRL agent is to steer an eye window. However, we also use voxels with more than binary features in our experiments. As the memory requirements of voxels grow cubically with their resolution [14], octrees are leveraged with a quadratic growth with respect to the grid resolution [14, 23]. The authors of [14, 23] developed special octree DNN operations such as the convolution to prevent massive calculations with empty voxels. The octree nets can be easily integrated in our system and could speed up our experiments massively or a higher voxel resolution could be chosen.

Qi et al. developed PointNet++ [13] which directly learns from a point cloud and takes different point densities into account. Basically, PointNet++ uses multi layer perceptrons with shared weights to combine local point features and a global point cloud feature for its computation. They used the farthest point sampling to achieve a better coverage of the point set than by using random sampling [13]. However, we used random sampling for simplicity in our first experiments as described in Section 4.3. Zhang et al. introduced the Link Dynamic Graph CNN (LDGCNN) [30] which computes the local point features by considering the nearest neighbourhood. Similar to PointNet [12], the predecessor architecture of PointNet++, a global point feature is calculated in the LDGCNN, too. Both networks, PointNet and LDGCNN, can be used to realise semantic segmentation as well as object classification and are also used in our experiments.

The tasks of these nets are mostly point cloud classification and semantic segmentation. By our knowledge, we are the first who develop a framework for estimating parameters of point cloud segmentation algorithms with DRL.

2.3 Datasets

There are several point cloud datasets with different properties [4, 29, 8, 2, 26, 28]. Some of them include labelled real world scenes where one point cloud contains several objects such as in [4, 6]. To use the ScanNet dataset [4] with our system, the mesh scenes have to be transformed to point clouds. The semantic3D.net dataset [6] can potentially be used without further processing if our system uses another segmentation algorithm as the region growing algorithm that needs the normal information of a point. Additionally, we do not consider to estimate the normals of the points [7] as the estimation results can be very noisy which reduces the performance of the point cloud algorithm. Other datasets contain only point clouds with single objects such as in [26, 18, 21, 28]. To use the datasets with single objects for our training, the objects have to be composed to scenes. Some datasets are generated with different sensors which provide different kind of information [4, 1]. ScanNet [4] was acquired by a RGB-D sensor whereas the Joint-2D-3D-Semantic dataset [1] was captured with a RGB camera. Moreover, some datasets do not provide the colour information such as the ModelNet dataset [26]. As our intend was to develop a first realisation of the DRL segmentation framework with isolating disturbing factors such as noise, we decided to use simple virtual point cloud scenes.

3 Point Cloud Segmentation

A point cloud $P \in \mathbb{R}^{|P| \times D}$ is considered as a matrix of spatial points. There are $|P|$ points in the point cloud and an i-th point

[3] The source code for the training environment can be accessed at `https://github.com/mati3230/smartsegmentation`.

$p_i \in P$ has at least $D \geq 3$ spatial features. The segment of the point p_i is characterised by the value $s_i \in \mathbb{N} \cup \{-1\}$. A point cloud consists of several objects and in contrast to the part segmentation task [28], we want to segment the top level objects in the point cloud P. The objective of the segmentation of a point cloud is to assign each point p_i to a segment $s_i \neq -1$ such that each segment represents an object. Initially, each point p_i is unsegmented what we encoded with $s_i = -1$. The output of the segmentation is a vector $\hat{s} \in (\mathbb{N} \cup \{-1\})^{|P|}$. For the remaining document, we will denote the true segment vector with s and the assigned segments with $\hat{s}$. The assigned segment vector $\hat{s}$ is usually constructed by an algorithm such as the region growing algorithm which is used in this contribution. Ideally, the assigned segment vector $\hat{s}$ should be equal or close to the true segment vector s.

4 Training Environment

Estimating the parameters of a geometric point cloud segmentation algorithm could be formulated as supervised learning problem. Thus, a dataset with appropriate parameters and point cloud states has to be constructed. Currently, it is unknown which geometric segmentation algorithms or states are appropriate for an ideal segmentation such that the construction of a dataset could be suboptimal. Therefore, we consider the segmentation problem as reinforcement learning problem where we can develop appropriate state representations and select appropriate geometric segmentation algorithms without the construction of an explicit dataset.

In reinforcement learning, a markov decision process (MDP) $M = (S, A, T, R, \gamma)$ is considered [20, p. 37]. It contains a set of states S, a set of Actions A, a transition model T, a reward signal R and a discount factor γ. The goal is to optimise a policy so that the accumulated reward is maximised [20, p. 42]. The point cloud segmentation is considered as MDP as follows. Given a point cloud, an agent estimates the parameter of a geometric segmentation algorithm as action and receives feedback through a reward function and it receives the next state as updated point cloud. Practically, the agent is optimised with respect to the discounted reward in the generalised advantage estimation (GAE) of the proximal policy optimisation (PPO) algorithm [17] in this contribution. A schema of the whole process is depicted in Figure 2. Furthermore, the state transition matrix T is not considered as we apply model free learning.

4.1 Action

In our framework, the agent has to segment the point cloud within multiple steps. In one step, the agent has to specify the parameters of the region growing algorithm which is described in the point cloud library [15]. The agent can choose a value from -1 to 1 for every parameter as action which is multiplied to a specific range of the parameters. The region growing parameters are the seed point p_s of a region, the number of neighbours K_s to grow a region and an angle threshold δ as well as a curvature threshold c. The range of a seed point p_s depends on the dimensions of the original scene such as in Table 2. The thresholds are used to specify if points belong to the same region and if they can be used as seed points. The angle threshold concerns the cosine angle between the normal of a seed point and the normal of a candidate point that could belong to a region. If this angle is smaller than δ, the point is assigned as candidate. If the curvature value of a candidate is smaller than c, the candidate belongs to a region.

In sum, an action a is in $[-1, 1]^6$. It is unlikely that an estimated seed point p_s is equal to a point p_i in the point cloud P. Hence, we choose the nearest unsegmented point of the K_q nearest neighbours in the point cloud P to the estimated point as seed point. The episode is finished if every point of the K_q nearest neighbours is already segmented. Additionally, the maximum number of segmentation steps is bounded by a parameter $t_{max} \geq O(s_t), t_{max} \in \mathbb{N}$. The function $O \in \mathbb{N}$ measures the number of objects in a segment vector. Hence, the agent has the possibility to apply at least as much segmentation steps as there are objects in the point cloud scene.

4.2 Scene Generation

To develop a prototypical realisation of the DRL segmentation framework, point clouds which can be segmented are necessary. The point clouds have to be labelled such that a reward value can be calculated after a segmentation. We transform 3D Meshes to point clouds such that a point to object relation exists. The data generation[4] consists of two main steps, namely, data acquisition and point cloud generation. In the first step of the data acquisition, specific categories of objects that are stored in a list will be downloaded from a 3D mesh model provider. The download is managed by a bot and the data is stored in specific folders appropriately. The folder structures, file types and additional data of the downloaded 3D meshes is not standardised. Hence, the second step of the data acquisition is to structure the downloaded files in a data table. In this data table, the files and their paths are listed. For example, a 3D mesh as .fbx file and, if given, the corresponding materials with textures are listed in this data table. After these two steps of downloading and structuring, the point cloud scenes can be generated.

To simulate a simplified environment of a room scan as in Figure 1, the scenes should be indoors and consist of four walls, a floor and a ceiling. Minimum and maximum sizes of the room in meters have to be specified for all three spatial axis by the user. Rooms with random widths, heights and depths between the minimum and maximum sizes are generated for each scene. For each object category which was acquired in the data acquisition process, a minimum and a maximum height in meters is specified to scale an object. Additionally, it is possible to specify heights of subcategories. For instance, the minimum and maximum height can be specified for the category plant. A subcategory of a plant can be a tree with a corresponding minimum and maximum height.

After a room is generated, the next step is to place an object of a specific category into the room. We choose a random category and a random position in the room. Subsequently, the object is uniformly scaled by the specified height of the category. Eventually, the object protrudes out of the wall. Hence, the bounding box of the object is calculated and the object is translated in the negative direction in which it protrudes the most out of the room. After that, the object is assigned with mesh colliders of the Unity 3D engine to prevent object intersections with other objects that does not exist in the realty. A minimum and maximum number of objects that should be placed in the room can be specified such that a random number between this range can be chosen for every room.

Some objects are positioned in the air due to the random positioning. Hence, gravity is applied for three seconds to arrange the objects on the floor. The next step is to transform the mesh scene consisting of a room and objects into a point cloud. For every mesh, points are

[4] The source code for the data generation is available at `https://github.com/mati3230/pc_data_generator`.

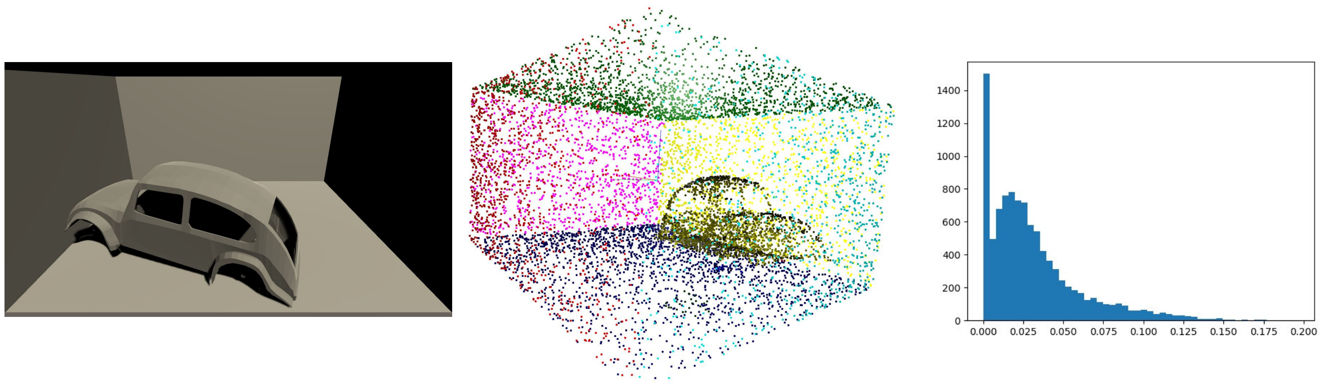

Figure 3: Left: A mesh scene that is created by the data generator. Middle: The corresponding point cloud scene with coloured segments. Right: Histogram of the curvature values of the point cloud scene in the middle.

sampled within the triangles of a mesh with respect to the size of a triangle. An exemplary point cloud scene that originates from a mesh scene is depicted in Figure 3.

4.3 State Representation

The state of the environment is the basis for the action selection of the agent. Currently, there are two different state representations that can be selected, namely a point cloud or a voxel representation. In contrast to the segment feature of P in Section 3, the segment feature is represented as binary feature, i.e. a point is segmented or not what we encoded with $s_i \in \{-1, 1\}$.

As we consider neural nets that process a point cloud with a fixed size of points, we have to sample from the point cloud. The point clouds produced with the data generation in Section 4.2 have more points for the walls than for the remaining objects. Therefore, we selected the points randomly and the user has to specify the proportion of the wall points $\psi \in [0, 1]$ in the sampled point cloud. Additionally to the state as point cloud, P can also be represented as voxel grid. A voxel has an occupied feature and can have optional additional features such as mean normal features and a mean curvature feature. Thus, the whole voxel grid has the dimension of $v_x \times v_y \times v_z \times v_d$ where v_x, v_y and v_z characterising how many voxels are used for the spatial dimensions. The dimension v_d represents the number of additional features. Moreover, all features of a voxel are set to zero if a point p_i that lies within a certain voxel is already segmented.

4.4 Reward Function

The reward function should measure the fraction of the correctly assigned segments $\hat{s}$ subject to the corresponding true segments s of the point cloud P. The pointwise, and thus elementwise, inequality of the segment numbers in $\hat{s}$ and s could be considered. However, if a segmentation algorithm is used that assigns a correct but different segment number $\hat{s}_i = j$ to an object with the true segment number $s_i = k$, a mapping $m(j) = k$ has to be constructed such that the inequality between the segment vectors can be calculated.

Algorithm 28 outputs such a mapping $m(j) = k$ and the number of unsegmented points u. Unsegmented points can occur since a segmentation algorithm not necessarily segments each point. A histogram for the k-th segment with the different assigned segments is created from Line 4 to Line 13 of Algorithm 28. Subsequently, the order of the true segments is determined by a sort function to create the mapping m in Line 14.

Finally, the mapping is created from Line 15 to Line 28. In Line 21, a specific assigned segment is chosen according to a certain criteria. We choose the most frequent assigned segment within an object.

Now it could be iterated over all elements of $\hat{s}$ to get the corresponding true segment with the mapping m. An error e is incremented for every i-th point where the assigned segment $\hat{s}_i$ differs from the true segment s_i. The mapping m is not defined for every j in case of more assigned segments than true segments. In this case, the error e is also incremented for every j that is not defined in m. Lastly, the reward r is calculated by $r = 1 - \frac{e+u}{|P|}$.

The distribution of segments in s can be skewed such that one object dominates the point cloud with most of the points. In this case, it is possible to achieve a high reward when just one segment is assigned to the whole point cloud. This behaviour is punished by subtracting a segment difference d. The segment difference is modelled as $d = |O(s) - O(\hat{s})|$ and the reward function changes to $r = 1 - \frac{e+u}{|P|} - f_d \cdot d$. The factor f_d can gain or dampen the segment difference punishment.

The value range of the reward r is $[-f_d \cdot d_{max}, 1]$ whereas d_{max} is the maximum segment difference. If considering the case $e = 0$ and $u = 0$, it follows that $d = 0$ and $r = 1$. Consider the case of $e + u \approx |P|$ and a maximum segment difference d_{max}. Since the seed point p_s of a region is assigned directly to a segment [15], the error e is less than the number of points $|P|$ and can be 0. The case of $e = 0$ happens if only one point is segmented and the remaining points are unsegmented due to small region thresholds. Hence, r is mainly determined by the value of the segment difference d if considering the case of $e = 0$ and $u = |P| - 1$ than $\lim\limits_{|P| \to \infty} \frac{|P|-1}{|P|} = 1$. The segment difference d depends on the maximum number of steps t_{max}. If $t_{max} = O(s)$, than the maximum value of d is $d_{max} = |O(s) - t_{min}|$. The variable t_{min} is the minimum number of segmentation steps. If $t_{max} > O(s)$, than the following applies:

$$d_{max} = \begin{cases} |t_{max} - O(s)| & \text{if } |t_{max} - O(s)| \geq |O(s) - t_{min}| \\ |O(s) - t_{min}| & \text{else} \end{cases} \tag{1}$$

Finally, the agent gets feedback in terms of the reward for every segmentation time step t in the environment. The feedback is expressed as the change of the reward $dr_t = r_t - r_{t-1}$ with $r_{t=0} = 0$. The first action of the agent is done with time step $t = 1$ and the terminal step of an episode is expressed with $t = T$ whereas T depends on the maximum number of segmentation steps t_{max} and how many

Algorithm 1: Creation of a mapping m which maps an assigned segment $\hat{s}_i = j$ to a true segment $s_i = k$.

```
   Input: $s$, $\hat{s}$
   Output: $m, u$
1  k_hists ← $\emptyset$;
2  $m \leftarrow \emptyset$;
3  $u \leftarrow 0$;
4  for $i \in$ len(s) do
5      $k = s[i]$;
6      $j = \hat{s}[i]$;
7      if $j = -1$ then
8          $u++$;
9          continue;
10     if $k \notin$ k_hists then
11         k_hists.add($k$, histogram);
12     k_hists[$k$].update($j$);
13 end
14 Sort k_hists;
15 for k_histogram $\in$ k_hists do
16     $k$ = k_histogram.key() ;
17     histogram = k_histogram.value() ;
18     while True do
19         if histogram $= \emptyset$ then
20             break;
21         $j \leftarrow$ selection(histogram);
22         if $j \in m$ then
23             remove $j$ from histogram;
24         else
25             $m[j] \leftarrow k$;
26             break;
27     end
28 end
```

points are already segmented. See Section 4.1 for a description of the terminal condition. The complete reward function at a time step t is shown in Equation 2. The subscript t is added to the number of incorrect assigned segments e and the number of unsegmented points u to point out that these quantities are calculated at each time step. The user can decide if the difference punishment is whether be applied at the terminal time step T or when more than the true number of objects $O(s_t)$ are segmented. This user decision is expressed by the $\vee$ sign of Equation 2.

$$r_t = \begin{cases} 0 & \text{if } t = 0 \\ 1 - \frac{e_t + u_t}{|P|} & \text{if } 0 < t < T \quad \vee \quad 0 < t < O(s) \\ 1 - \frac{e_t + u_t}{|P|} - f_d \cdot d_t & \text{if } t = T \quad \vee \quad t \geq O(s) \end{cases} \tag{2}$$

5 Experiments

The point clouds that are considered in this contribution are unordered and the $D = 8$ features are:

- three spatial features for the coordinates of a point,
- three spatial features for the normal of a point,
- one spatial feature for the curvature of a point and
- one feature for the segment of a point.

The normal vector is a direction in $\mathbb{R}^3$ which is perpendicular to the surface of a point. Additionally, the curvature is considered as the rate of change of the tangent vector of a point. The point normals are given from the transformed point cloud scenes. The curvature feature is estimated by using the normal estimation method of Hoppe et al. [7] by calculating the eigenvalues of a 3×3 covariance matrix of a point and its neighbourhood. After that, the curvature feature is calculated as the fraction of the smallest eigenvalue divided by the sum of eigenvalues.

We experimented with four scenes which are generated according to Section 4.2 with about 10^4 points. The details of the point cloud scenes can be seen in Table 2. We set the fraction of points for the wall to $\psi = 0.3$ and the sampling size to 1024.

After a query point is given as action, it is searched for the $K_q = 30$ nearest neighbours. Besides the range of a seed point p_s, the range of the action parameters are specified in Table 1. The range of the neighbourhood parameter K_s and the angle threshold δ were found by applying segmentations manually with the proposed environment. The lower bound of the curvature threshold range was found by assuming that faces lie within one segment. Hence, we assume that a candidate point belongs to a region if it has a lower curvature than 0.025 which is true for the most of the faces in our scenes. The upper bound should just capture the whole range of possible values of the scenes. A histogram of representative curvature values is depicted in Figure 3 on the right. The maximum number of segmentation steps is set to $t_{max} = 15$ and the segment difference factor to $f_d = 0.0125$.

Table 1: Minimum and maximum parameters of an action in the proposed environment.

Parameter	**Min**	**Max**
K_s	10	30
δ	10°	67.5°
c	0.025	0.3

Table 2: Properties of the different point cloud scenes that we used for the experiments. Due to the walls, the scenes have at least six objects plus some additional objects in the room.

Nr.	**Nr. of points**	**Nr. of objects**	**Range $(\mathbf{x}, \mathbf{y}, \mathbf{z})^\mathbf{T}$**
1	9982	9	$\approx (2.7, 3.1, 2.0)^T$
2	9973	8	$\approx (2.7, 3.0, 1.9)^T$
3	9993	9	$\approx (3.4, 3.4, 1.6)^T$
4	9995	9	$\approx (3.3, 3.0, 1.8)^T$

To identify a network for further optimisations, we trained the LDGCNN of Zhang et al. [30], a modified PointNet [12] and a voxel based net as point cloud networks in the proposed environment of Section 4 with the first scene in Table 2. We selected the bubble sort algorithm in Line 14 of Algorithm 28. The true segments were sorted in descending order according to the number of points they occupy. The optimisation is conducted with the PPO algorithm of Schulman et al. [17] with the stable baselines framework[5]. Here, we used the default PPO parameters for the first experiment. All nets output the action parameters that are described in Section 4.1 and a state value estimate which is in $\mathbb{R}$. The network architecture ongoing from a point cloud network is depicted in Figure 4.

The architecture of the voxel based net is described in Table 3. In case of using the point cloud P as input for the LDGCNN and the PointNet, the first three spatial features of P are standardised by a z

[5] The stable baselines framework can be accessed at `github.com/hill-a/stable-baselines` (accessed 13.11.2019).

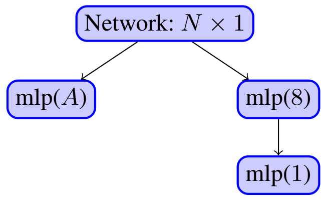

Figure 4: Action and value calculation. We used standard mlp connections whereas the last layers conduct linear activations and the hidden layer (mlp(8)) conducts a ReLu activation. The action estimation is depicted by mlp(A), where A corresponds to the number of segmentation parameters. The state value is estimated by mlp(1).

standardisation to have a zero mean and a standard deviation of one. The layer with the output classes of the LDGCNN is replaced with a fully connected layer with 40 neurons. Furthermore, we used the segmentation PointNet that is proposed in [12].

Table 3: Architecture of the voxel based net. The input point cloud is transformed to a voxel grid with a resolution of $v_x = 60, v_y = 60, v_z = 60$. According to Section 4.3, a voxel has $v_d = 5$ features consisting of an occupied state, a mean normal in $\mathbb{R}^3$ and a mean curvature value. The kernel and stride sizes are equal for all three dimensions. The padding of the Conv and the MaxPool layers are valid. The convolution layers are using the ReLu activation function. The stride and the kernel values are the same for every dimension (e.g. a kernel value of 3 is equal to $(3, 3, 3)$)

Layer	Input	Out Channels	Kernel	Stride
Conv	$60 \times 60 \times 60 \times 5$	16	6	1
MaxPool	$55 \times 55 \times 55 \times 16$	/	3	2
Conv	$27 \times 27 \times 27 \times 16$	32	5	1
MaxPool	$23 \times 23 \times 23 \times 32$	/	3	2
Conv	$11 \times 11 \times 11 \times 32$	64	3	1
MaxPool	$9 \times 9 \times 9 \times 64$	/	3	2
Conv	$4 \times 4 \times 4 \times 64$	64	2	1
MaxPool	$3 \times 3 \times 3 \times 64$	/	3	1
Flatten	$1 \times 1 \times 1 \times 64$	/	/	/

Table 4: Architecture of the transformation net (T-Net). The calculation of the transformation matrix after the last fully connected layer was not changed in comparison to [12].

Layer	Input	Out Channels	Kernel	Stride
Conv	$N \times D \times 1$	64	$1 \times f$	1×1
Conv	$N \times 1 \times 64$	128	1×1	1×1
MaxPool	$N \times 1 \times 128$	/	$N \times 1$	2×2
Flatten	$1 \times 1 \times 128$	/	/	/
FC	128	256	/	/

The models are trained with tensorflow on a machine with 32GB DDR4 RAM, one NVidia GeForce RTX2080 and an Intel Core i7-9800X CPU. The LDGCNN and the voxel based net had a mean update rate of 10 steps per second. Thus, the training of 1.6M training steps takes approx. 2 days on our machine. The LDGCNN conducts an expensive k nearest neighbour operation in the edge convolution [30] step at every frame which slows down the performance. The voxel based net uses expensive 3D convolutions at every step which are slower than the 2D equivalent. The PointNet had a mean update rate of 20 steps per second. According to Figure 6, the PointNet achieves slightly better performance as the other two models. Hence, we decided to conduct the hyperparameter optimisation with the PointNet.

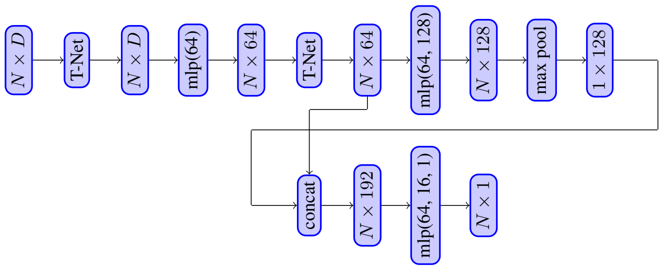

Figure 5: Lightweight PointNet architecture. All mlp connections are realised by convolutions with shared weights. The stride dimensions are always 1×1 besides the first mlp with a stride of $1 \times D$. The padding is always set to valid and the activations are realised by ReLu functions.

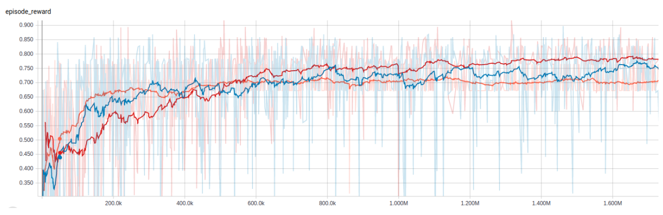

Figure 6: Training progress of the LDGCNN (blue), the voxel based network (orange) and the PointNet (red). The vertical axis represents the achieved reward. The horizontal axis represents the time steps.

Although a high reward of $r = 0.78$ was achieved by the different networks, the segmentation result was insufficient due to the combination of a wall and at least one object in one segment. See Figure 7 on the right for an illustration of this problem. A reason for this property is the order of the true segments after applying the bubble sort in Line 14 of Algorithm 28. Consider the following order of objects: $w_1, o_1, w_2, w_3, w_4, w_5, w_6, o_2, o_3$. All models put a wall $w_i, i > 1$ and the object o_1 together in one segment. Assume the incorrectly segmented elements are the objects o_2, o_3 and the wall w_i that is in one segment with the object o_1. As the object o_1 has the second most points, the reward is still high as the incorrectly segmented wall does not cause a huge error e. To prevent this behaviour, we changed the sorting in Line 14 of Algorithm 28. First, the wall objects are considered and after that the remaining objects are considered in the reward calculation. Moreover, we splitted the reward function into two parts.

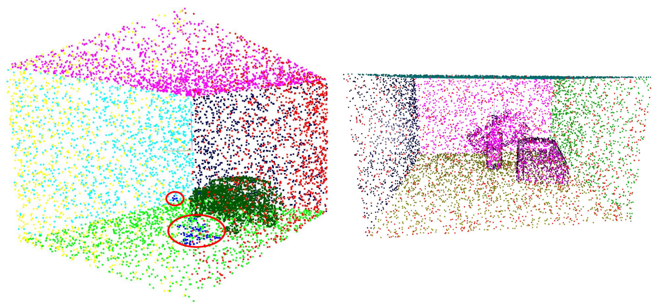

Figure 7: Segmentations achieved by trained PointNets. Left: Two objects with the dark blue colour which are highlighted by the red ellipses were not segmented. The larger unsegmented object is a chair and the smaller one is a cup. The reward is $r = 0.94$ and calculated with Equation 3. Right: The wall in pink is put in one segment with the objects. The reward is $r = 0.78$ and calculated with Equation 2.

One part of the reward function should consider of the walls and the other the objects as follows:

$$r = \theta_w \cdot r_w + (1 - \theta_w) \cdot r_o, \theta_w \in [0, 1]$$
$$r_w = 1 - \frac{e_w + u_w}{|P_w|}, r_o = 1 - \frac{e_o + u_o}{|P_o|} \tag{3}$$

The reward for the segmented walls r_w and the segmented objects r_o is calculated. According to the case in the middle of Equation 2, the rewards r_w and r_o are calculated by summing up the number of erroneous and unsegmented points, divide this sum through the number of points of the wall $|P_w|$ or of the objects $|P_o|$ and subtract this term from 1. Lastly, the object and wall reward are weighted and a linear interpolation with a user defined factor θ_w is applied. The factor θ_w weights the wall reward r_w in context of the whole reward. For the further experiments, we set this wall reward factor to $\theta_w = 0.5$.

With the new reward function in Equation 3, we trained a lightweight PointNet on the first scene of Table 2 and optimised the PPO parameters. The lightweight transformation net (T-Net) component of the used PointNet is described in Table 4. The complete PointNet architecture can be seen in Figure 5. In contrast to the default stable baselines parameters, we achieved nearly a perfect segmentation with a reward of $r = 0.94$ by increasing the buffer to 1024, the batch size to 64 and decreasing the entropy factor to 0.0001. The full list of the optimisation parameters are depicted in Table 5. A segmentation achieved by this net is depicted in Figure 7 on the left.

Table 5: Optimised parameters of the PPO algorithm used in our training experiments.

Parameter	Value
learning rate α	0.00025
clip	0.2
discount factor γ	0.9
GAE factor λ	0.95
entropy factor β	0.0001
buffer	1024
batch size	64

In the last experiment, we trained the PointNet with the parameters of Table 5 on all four scenes. The model often learns to segment the walls in all four scenes and, hence, achieves a reward of $r = 0.5$. In our best run, we achieved a performance of $r = 0.8 \pm 0.1$. Furthermore, we tested the trained model with two scenes that have similar characteristics than the scenes of Table 2 that consist of unseen objects. According to Figure 8, the model has learned to estimate the region growing parameters for the walls and some objects.

6 Conclusion

We presented the concept a DRL framework for point cloud segmentation and realised a first approach. Likewise a semantic segmentation approach, the proposed DRL framework needs segmented point cloud data such that the true segment vector s is available. In this context, a generator to produce virtual data is presented that is able to produce a huge number of different scenes with different objects. The virtual scenes contain the normal information which is not given for point clouds that originate from a scanning process. Thus, an appropriate normal estimation method is necessary for such scanned scenes if the region growing algorithm is applied. One benefit is the

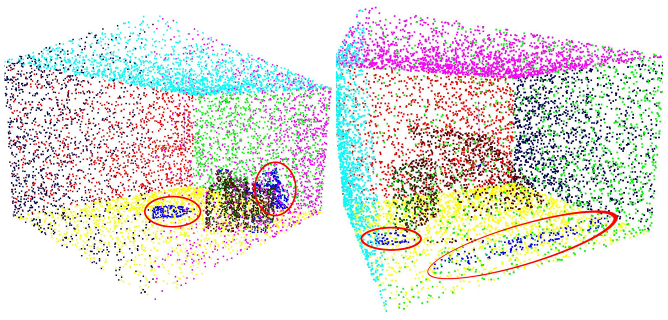
Figure 8: Two unseen point cloud scenes that are segmented by our DRL segmentation with a reward of $r = 0.66$ (left) and $r = 0.80$ (right). The blue points within the red ellipses are not segmented in both scenes.

usage of a variety of 3D meshes which can be downloaded and used freely for research purposes. Another benefit is that the 3D meshes can be placed in the 3D scene arbitrarily. Therefore, a lot of different point cloud scenes can be used for the training.

A reward function which rates a true and an assigned segment vector is defined. It is designed for the case when the segment vectors have different segment numbers for the same object. This is realised by the help of the mapping function m (see Section 4.4). We achieved better segmentation results by considering the wall and the remaining objects in the calculation of the reward function (see Equation 3) separately. Furthermore, we conducted some experiments with different point cloud network architectures and showed promising results. However, the agent is limited as it only sees a sampled version of the point cloud. Despite this limitation, a nearly perfect segmentation with a reward of $r = 0.94$ could be achieved in one scene. Lastly, the results show that an DRL agent is able to learn the segmentation problem in the proposed DRL framework.

In contrast to semantic segmentation approaches, the proposed DRL point cloud segmentation has two advantages if scenes should only be segmented. First, the DRL segmentation is able to segment point cloud scenes with unknown objects. Second, the DRL segmentation is able to segment a whole point cloud obviously an agent sees a sampled set of the point cloud. In most cases of applying semantic segmentation with a neural net, the point cloud has to be sampled for the input layer and the estimated segments are only available for these sampled points. This property complicates the comparison with a DNN semantic segmentation approach.

7 Future Work

Despite the promising results, the DRL framework was only tested on a few simple virtual point cloud scenes. We wish to process complex, real and dense point cloud scans. Therefore, we are currently working on adding the colour and lighting information to the generated data.

Our DRL segmentation approach is as only as good as the segmentation abilities of the region growing algorithm. Therefore, we want to try different geometric segmentation algorithms. Another consideration is to define an action in a hierarchical way as follows: a DNN selects a geometric segmentation method and another DNN estimates the parameters of the method such as in this work.

Acknowledgement

This research has been funded by the Federal Ministry of Education and Research (BMBF) of Germany in the framework of interactive body-centered production technology 4.0 (German: Interaktive körpernahe Produktionstechnik 4.0 - iKPT4.0) (project number 13FH022IX6).

REFERENCES

[1] Iro Armeni, Sasha Sax, Amir R. Zamir, and Silvio Savarese, 'Joint 2D-3D-Semantic Data for Indoor Scene Understanding', *Computing Research Repository (CoRR)*, (feb 2017).

[2] Aseem Behl, Despoina Paschalidou, Simon Donné, and Andreas Geiger, 'PointFlowNet: Learning Representations for Rigid Motion Estimation from Point Clouds', *Computing Research Repository (CoRR)*, (jun 2018).

[3] Songle Chen, Lintao Zheng, Yan Zhang, Zhixin Sun, and Kai Xu, 'VERAM: View-Enhanced Recurrent Attention Model for 3D Shape Classification', *IEEE Transactions on Visualization and Computer Graphics*, **25**(12), 3244–3257, (dec 2019).

[4] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Niebner, 'ScanNet: Richly-Annotated 3D Reconstructions of Indoor Scenes', in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2432–2443. IEEE, (jul 2017).

[5] E. Grilli, F. Menna, and F. Remondino, 'A REVIEW OF POINT CLOUDS SEGMENTATION AND CLASSIFICATION ALGORITHMS', *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, **XLII-2/W3**(2W3), 339–344, (feb 2017).

[6] Timo Hackel, Nikolay Savinov, Lubor Ladicky, Jan D Wegner, Konrad Schindler, and Marc Pollefeys, 'SEMANTIC3D.NET: A NEW LARGE-SCALE POINT CLOUD CLASSIFICATION BENCHMARK', *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, **IV-1/W1**(1W1), 91–98, (may 2017).

[7] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle, 'Surface reconstruction from unorganized points', in *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH '92*, pp. 71–78, Chicago, USA, (1992). ACM.

[8] Jinyong Jeong, Younggun Cho, Young-Sik Shin, Hyunchul Roh, and Ayoung Kim, 'Complex Urban LiDAR Data Set', in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6344–6351, Brisbane, Australia, (may 2018). IEEE.

[9] Fangyu Liu, Shuaipeng Li, Liqiang Zhang, Chenghu Zhou, Rongtian Ye, Yuebin Wang, and Jiwen Lu, '3DCNN-DQN-RNN: A Deep Reinforcement Learning Framework for Semantic Parsing of Large-Scale 3D Point Clouds', in *IEEE International Conference on Computer Vision (ICCV)*, pp. 5679–5688, Venice, Italy, (oct 2017). IEEE.

[10] Xiaohu Lu, Jian Yao, Jinge Tu, Kai Li, Li Li, and Yahui Liu, 'PAIRWISE LINKAGE FOR POINT CLOUD SEGMENTATION', *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, **III-3**(July), 201–208, (jun 2016).

[11] Daniel Maturana and Sebastian Scherer, 'VoxNet: A 3D Convolutional Neural Network for real-time object recognition', in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 922–928, Hamburg, Germany, (sep 2015). IEEE.

[12] Charles R. Qi, Hao Su, Mo Kaichun, and Leonidas J. Guibas, 'PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation', in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2017-Janua, pp. 77–85, Honolulu, Hawaii, (jul 2017). IEEE.

[13] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas, 'PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space', in *Conference on Neural Information Processing Systems (NIPS)*, eds., I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, pp. 5099–5108, Long Beach, USA, (jun 2017). Curran Associates, Inc.

[14] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger, 'OctNet: Learning Deep 3D Representations at High Resolutions', in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6620–6629, Honolulu, Hawaii, (jul 2017). IEEE.

[15] Radu Bogdan Rusu and Steve Cousins, '3D is here: Point Cloud Library (PCL)', in *IEEE International Conference on Robotics and Automation*, pp. 1–4, Shanghai, China, (may 2011). IEEE.

[16] Jonas Schild, Sebastian Misztal, Beniamin Roth, Leonard Flock, Thomas Luiz, Dieter Lerner, Markus Herkersdorf, Konstantin Weaner, Markus Neuberaer, Andreas Franke, Claus Kemp, Johannes Pranqhofer, Sven Seele, Helmut Buhler, and Rainer Herpers, 'Applying Multi-User Virtual Reality to Collaborative Medical Training', in *IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 775–776, Reutlingen, Germany, (mar 2018). IEEE.

[17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov, 'Proximal Policy Optimization Algorithms', *Computing Research Repository (CoRR)*, 1–12, (jul 2017).

[18] Philip Shilane, Patrick Min, Michael Kazhdan, and Thomas Funkhouser, 'The Princeton Shape Benchmark', in *Proceedings Shape Modeling Applications*, pp. 167–388, Genova, Italy, (2004). IEEE.

[19] Harald Steinlechner, Bernhard Rainer, Michael Schwärzler, Georg Haaser, Attila Szabo, Stefan Maierhofer, and Michael Wimmer, 'Adaptive Pointcloud Segmentation for Assisted Interactions', in *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games - I3D '19*, pp. 1–9, Montreal, Quebec, Canada, (2019). ACM.

[20] Richard Sutton and Andrew Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge (Massachusetts), London, 2017.

[21] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Duc Thanh Nguyen, and Sai-Kit Yeung, 'Revisiting Point Cloud Classification: A New Benchmark Dataset and Classification Model on Real-World Data', in *International Conference on Computer Vision (ICCV)*, Seoul, Korea, (aug 2019).

[22] Julien Valentin, Philip Torr, Vibhav Vineet, Ming-Ming Cheng, David Kim, Jamie Shotton, Pushmeet Kohli, Matthias Nießner, Antonio Criminisi, and Shahram Izadi, 'SemanticPaint', *ACM Transactions on Graphics*, **34**(5), 1–17, (nov 2015).

[23] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong, 'O-CNN', *ACM Transactions on Graphics*, **36**(4), 1–11, (jul 2017).

[24] Qian Wang, Yi Tan, and Zhongya Mei, 'Computational Methods of Acquisition and Processing of 3D Point Cloud Data for Construction Applications', *Archives of Computational Methods in Engineering*, (0123456789), (feb 2019).

[25] Ronald J. Williams, 'Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning', *Machine Learning*, **8**(3), 229–256, (1992).

[26] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao, '3D ShapeNets: A deep representation for volumetric shapes', in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1912–1920, Boston, Massachusetts, USA, (2015). IEEE.

[27] Mohsen Yavartanoo, Eu Young Kim, and Kyoung Mu Lee, 'SPNet: Deep 3D Object Classification and Retrieval Using Stereographic Projection', in *Asian Conference on Computer Vision (ACCV)*, pp. 691–706, Perth, Australia, (2018).

[28] Fenggen Yu, Kun Liu, Yan Zhang, Chenyang Zhu, and Kai Xu, 'PartNet: A Recursive Part Decomposition Network for Fine-grained and Hierarchical Shape Segmentation', in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, USA, (mar 2019).

[29] Andy Zeng, Shuran Song, Matthias NieBner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser, '3DMatch: Learning Local Geometric Descriptors from RGB-D Reconstructions', in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 199–208, Honolulu, Hawaii, (jul 2017). IEEE.

[30] Kuangen Zhang, Ming Hao, Jing Wang, Clarence W. de Silva, and Chenglong Fu, 'Linked Dynamic Graph CNN: Learning on Point Cloud via Linking Hierarchical Features', *Computing Research Repository (CoRR)*, 1–8, (apr 2019).