Point Cloud Segmentation: Solving a Perceptual Grouping Task with Deep Reinforcement Learning

Marcel Tiator*, Christian Geiger*, Paul Grimm†

* University of Applied Sciences Düsseldorf
E-Mail: {marcel.tiator, geiger}@hs-duesseldorf.de

† Fulda University of Applied Sciences
paul.grimm@cs.hs-fulda.de

Abstract: We propose a method to segment a real world point cloud as perceptual grouping task (PGT) by a deep reinforcement learning (DRL) agent. A point cloud is divided into groups of points, named superpoints, for the PGT. These superpoints should be grouped to objects by a deep neural network policy that is optimised by a DRL algorithm. During the PGT, a main and a neighbour superpoint are selected by one of the proposed strategies, namely by the superpoint growing or by the smallest superpoint first strategy. Concretely, an agent has to decide if the two selected superpoints should be grouped together and receives reward if determinable during the PGT. We optimised a policy with the proximal policy optimisation (PPO) [SWD+17] and the dueling double deep q-learning algorithm [HMV+18] with both proposed superpoint selection strategies with a scene of the ScanNet data set [DCS+17]. The scene is transformed from a labelled mesh scene to a labelled point cloud. Our intermediate results are optimisable but it can be shown that the agent is able to improve its performance during the training. Additionally, we suggest to use the PPO algorithm with one of the proposed selection strategies for more stability during the training.

Keywords: Point Cloud Segmentation, Deep Reinforcement Learning, Superpoints, Perceptual Grouping Task, Virtual Reality

Figure 1: Textured mesh of an anaesthesia room that was generated by point cloud scans (left) and the visualization of the corresponding wireframe (right). The scene has to be segmented to develop interactive VR elements.

1 Introduction

The real world can be captured in 3D with methods such as photogrammetry and laser scanning. The acquired data of this methods is used to generate a point cloud from which

Veröffentlicht durch die Gesellschaft für Informatik e.V. 2020 in B. Weyers, C. Lürig, D. Zielasko (Hrsg.):
GI VR / AR Workshop 2020, 24. - 25, September 2020, Trier.

http://dx.doi.org/10.18420/vrar2020_28

a 3D mesh can be calculated. The resulting mesh can be used for virtual, augmented or mixed reality applications. Currently, we captured different rooms of a hospital to produce a VR tour for the patients (see Figure 1). All objects of a room will be connected when transforming a room to a 3D mesh without any processing. Hence, the objects have to be extracted from the mesh or the point cloud to realize any interactions with the objects of a room in VR [TGG20]. This extraction process can be done by a segmentation algorithm and can take hours if it is done manually. Our research goal is the segmentation of objects from a point cloud by a certain semantic as, e.g., the level of detail of single objects can be controlled individually during the mesh reconstruction to enhance the creation of interactive VR environments.

On the one hand, the object segmentation can be conducted by a classifier where an object with a certain semantic should be recognized [QSKG17, QYSG17, WLG+17, CZZ+19, ZHW+19, MS15]. On the other hand, the segmentation can be applied by a geometric algorithm such as the region growing or the random sample consensus (RANSAC) algorithm [GMR17]. Obviously the former classification methods are able to work fully automatic as no expert parameters with certain approaches such as [QSKG17] have to be set, they may only are able to recognise the object classes of a specific domain accurately. The latter geometric methods are mainly unbiased and not automatic as certain expert parameters have to be set. Besides that, the segments of a geometric segmentation method do not necessarily represent semantic objects as often homogenous regions are clustered. Hence, the question arises: What is a real-world, cross-domain, automatic, semantic object segmentation solution that does not include the estimation of specific object classes?

In this contribution, a real world, automatic, semantic segmentation is realised as a perceptual grouping task (PGT) that is done by an agent that learns by deep reinforcement learning (DRL). The agent sees two superpoints[1] that originate from the voxel cloud connectivity segmentation (VCCS) algorithm [PASW13] and has to decide if they belong together. The two superpoints are rendered from different perspectives and are fed in into a multiview convolutional neural network (MVCNN) which finally outputs the grouping decision. We implemented two prototypical environments with different superpoint selection strategies to choose the superpoints that can be potentially grouped together (see Section 4). Concretely, the contribution consists of:

- Two PGT environments, namely the superpoint growing and the smallest superpoint first environment, that can be used with any data set where only the segments are labelled. Both environments are experimentally evaluated to decide with which one we should continue our research. Furthermore, the superpoint representation for the training of the agent can be switched arbitrarily which is not possible with previous approaches.

- The experimental optimisation of a MVCNN with the dueling double deep q-learning

[1] By superpoints we refer to a group of points that share common characteristics [PASW13].

(D3QN) and the proximal policy optimisation (PPO) algorithm (see Section 6) to decide if we should continue our research with an off-policy or on-policy method.

- The extension of the reward function of our previous segmentation approach [TGG20] to reward the agent during the segmentation process. The function is described in Section 5.

2 Background & Related Work

We consider point clouds of the form $P \in \mathbb{R}^{|P|\times 6}$ as a matrix of points. There are $|P|$ points in the point cloud and an i-th point $p_i \in P$ has 3 spatial features and 3 colour features. A segment of a point p_i is characterised by the value $s_i \in \mathbb{N} \cup \{0\}$. The objective of the segmentation of a point cloud is to assign each point p_i to a segment $s_i \neq 0$ such that each segment represents an object. Initially, each point p_i is unsegmented what we encoded with $s_i = 0$. The output of the segmentation is a vector $\hat{s} \in (\mathbb{N} \cup \{0\})^{|P|}$. We will denote the true segment vector with s and the assigned segments with $\hat{s}$.

Recently, we proposed a framework for the segmentation of point clouds with DRL [TGG20]. We could show that simple self-generated point clouds can be segmented by an agent that chooses seed points of a region growing algorithm. Thus, the point cloud is segmented semantically and automatically without setting any expert parameters. In contrast to our previous work, we applied the DRL segmentation approach to segment real-world point clouds and extended our reward function to reward the agent during the segmentation process.

2.1 Perceptual Grouping

"Perceptual grouping refers to the process of determining which regions and parts of the visual scene belong together as parts of higher order perceptual units such as objects or patterns." [Bro14]. Richtsfeld et al. [RMP+12] proposed systems to group model based patches such as planes from RGB-D images. They used the RANSAC algorithm, NURBS and B-Splines to represent those patches. A neighbourhood relationship for the patches is constructed such that two Support Vector Machines (SVMs) can decide if the patches belong together. After that, a graph cut algorithm [FH04] which uses probabilities of the SVMs is applied for the global segmentation. Richtsfeld et al. work on the RGB-D domain and used handcrafted features as opposed to us. Furthermore, they trained the SVMs in a supervised setting with annotated samples from humans [RMP+12] where the superpoints that belong together are labelled to train the SVMs. Therefore, a relabelling of the whole data set would be necessary if another superpoint representation (e.g. from planes to circles) is introduced. In contrast to [RMP+12], we only need the segment labels of the point cloud scenes and we are able to switch the superpoint representation arbitrarily.

Xu et al. [XHTS17] proposed a pure geometric segmentation of point clouds by the use of voxels and graph clustering. Geometrical features between the voxels are calculated by

Figure 2: The two leftmost images show a mesh scene of the ScanNet data set: the coloured mesh scene (left) and the corresponding semantically labelled mesh scene (right). The two rightmost images show these scenes as point clouds after their processing: The point cloud scene used for the training as input for the agent (left) and the same point cloud scene where the colour represents the segment label (right).

the perceptual grouping laws of proximity, similarity and continuity. Their workflow can also be applied to superpoints which are calculated by the VCCS algorithm [PASW13] that we used, too. To calculate the final segments, they merged local adjacency graphs. The local adjacency graphs consist of supervoxels that are grouped by a distance metric and are calculated a certain partition method. In contrast to us, Xu et al. do not use an agent which can learn to decide the grouping of supervoxels. Moreover, the usage of handcrafted features [XHTS17, RMP+12] can be biased to certain domains. Therefore, we leverage neural networks where dozens of features are learned during the training process.

2.2 Observation Representation

A certain observation representation as input for a neural network has to be chosen when processing point clouds with DRL. Most of the state of the art point cloud processing approaches with neural networks work with a fixed sized input for the network [QSKG17, QYSG17, WLG+17, CZZ+19, ZHW+19, MS15]. Hence, sampling techniques such as the farthest point sampling [QYSG17] or the voxel based filtering are applied to sample this fixed size. This sampling techniques introduce the problem of choosing the relevant points for the neural network such that a retraining is may necessary when using point clouds with different number of points. Hence, we decided to render a point cloud as images from multiple perspectives by a simple perspective camera model such as Chen et al. [CZZ+19] as the sampling of certain points is not required. Moreover, Chen et al. [CZZ+19] proposed a neural network architecture, the MVCNN, for the processing of the features of different rendered views which we also applied in this work as some important features may occluded in certain views.

3 Ground Truth Data Generation

Labelled ground truth data is necessary to reward an agent with the DRL segmentation approach [TGG20]. Therefore, we choose the ScanNet data set of Dai et al. [DCS+17] for the training of the segmentation agent. The data set consist of real world living room scenes

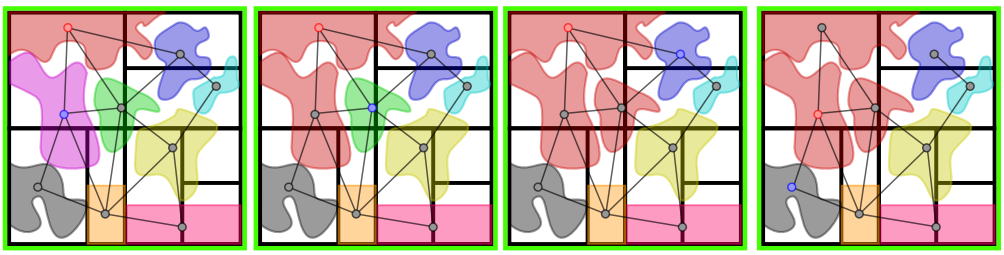

Figure 3: Schematic of a PGT environment with the superpoint growing selection strategy. A green box visualises the point cloud and the state of the PGT. An agent should group the coloured isles such that the black rectangles are fitted optimally. A more elaborate description can be found in Section 4.1.1.

as meshes. We consider two files per scene, a coloured mesh scene (see Figure 2 upper left) and a mesh scene where the colour represents the semantic label of an object (see Figure 2 upper right). To use the data set for our problem, a point cloud P and a true segment vector s have to be created from the meshes.

For our experiments, we focus on the first scene of the data set. We sampled a coloured point cloud P with 44.965 points with the mesh sampling tool of the PCL [RC11]. The sampling size was chosen as trade-off between the training speed and the representation of a real world point cloud scan. The resulting point cloud which can be seen in Figure 2 (second image from the left) is transformed in the origin and rotated such that the bounding box is parallel with the axis of the coordinate system. We sampled the same number of points in the same manner as above from the labelled mesh scene to get a point cloud where the colour of the points express their semantic label. After that, we sort that cloud by its colour values and fed it into a euclidean distance based region growing where the distance threshold is set to 0.4 to split distant objects with the same semantic label. Furthermore, two or more objects with the same semantic can be represented as different segments in our opinion as, e.g., a classification of the segmented objects can be applied afterwards. The resulting 53 regions are used as the segments of the true segment vector s (see rightmost image of Figure 2).

4 Environments

4.1 Perceptual Grouping Environments

The proposed PGT environments share common mechanisms. According to Figure 4, a ground plane segmentation is applied with the RANSAC algorithm. After that, the VCCS algorithm calculates the superpoints with their adjacency graph. Subsequently, a selection of a superpoint with its neighbour according to a superpoint selection strategy is done. An agent gets the point cloud and the point indices of the two superpoints that could be grouped as observation. If there are superpoints left to group in the node "TODO?", the agent will be asked if the superpoints should be grouped. The actions of the agent are to say yes or no. The neighbourhood connection will not be considered for the remaining episode if the

agent says no. If the agent says yes, the main superpoint will be extended with the points and the neighbourhood of the neighbour superpoint. This process is repeated till no more superpoints are available in the "TODO?" node. After that, the next point cloud can be processed in the node "Get P" in the same manner. In sum, we used 163 superpoints for the PGT environments.

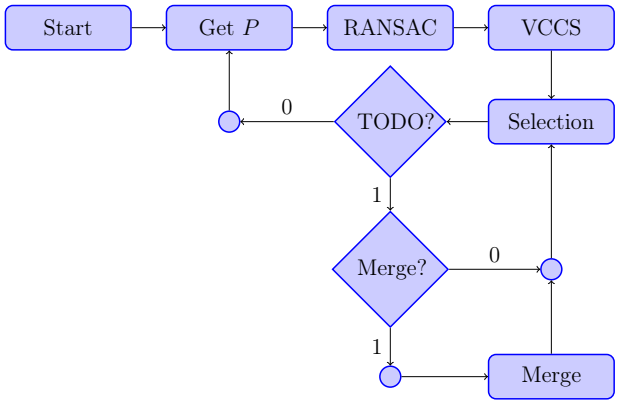

Figure 4: Conceptual flow diagram of a PGT environment.

rel_majs = ← list();
foreach k *where* $c_{kj} > 0$ **do**
  if $m(k)$ *exists* **then** continue;
  if *any* $c_{kl} > c_{kj}$ **then** continue;
  $u_k \leftarrow$ TraverseNeighbourhood(j);
  if $c_{kj} < u_k$ **then** continue;
  rel_majs.append((k, c_{kj}, $|s_k|$));
$k \leftarrow argmax_{|s_k|}$(rel_majs);
$m(k) = j$;

Algorithm 1: Creation of a mapping m which maps a finalised superpoint group j to a true segment k.

4.1.1 Superpoint Growing Environment

The superpoint growing as superpoint selection strategy is similar to the region growing algorithm when segmenting point clouds. A superpoint is chosen as main superpoint and the agent is asked if a neighbour superpoint should be grouped to the main superpoint. After the grouping, the neighbours of the neighbour superpoint will also be considered as neighbours of the main superpoint and put on top of a FIFO queue. The main superpoint region is grown with grouping suggestions till all neighbours are visited. After that, the next main superpoint is chosen and the procedure begins from the beginning.

The superpoint growing principle is depicted in Figure 3 as 2D visualisation for simplicity. The black rectangles should visualise the true segments and a whole green box a point cloud. The coloured isles visualise the superpoints. Generally, the agent should group the coloured superpoints such that the true segments are fitted optimally. The optimal grouping should result in the maximum reward which is described in Section 5. The grouping suggestions are highlighted with coloured dots in the middle of the coloured isles. The main superpoint has a red dot and the neighbour superpoint a blue dot. The neighbourhood adjacency graph is visualised by the black line connections between the superpoints. If a grouping suggestion is confirmed by the agent, then the neighbour superpoint will be also coloured in the colour of the main superpoint.

4.1.2 Smallest Superpoint First Environment

Inspired by the divide and conquer problem solving strategy, the superpoints are sorted by their number of points in every step. Subsequently, the smallest superpoint is chosen as main

superpoint and its smallest neighbour. Hence, the agents starts with the smallest problems in the PGT. After a grouping is confirmed by the agent, the subsegments are grouped such as in Section 4.1.1. After that, the superpoints will be sorted again and the smallest superpoint will be chosen again. If a grouping is declined, the next unseen neighbour of the main superpoint is chosen. If all neighbours of a main superpoint were visited, the superpoint will not be considered for the main selection or as neighbour superpoint any more.

5 Reward Function

In general, the agent should be rewarded for correctly segmented points and may punished otherwise. In our PGT environments, the agent must perform at least as many decisions as the number of superpoints. This results from the case if the agent just confirms every grouping decision. The number of decisions is even higher if the agent confirms and declines some grouping suggestions which is an expected behaviour for complex scenes.

$$r = 1 - \frac{e + u}{|P|} \tag{1}$$

The reward perception of the agent will be very sparse if the reward of Equation 1 is calculated at the end of the episode such as in our previous approach [TGG20]. This would make the PGT hardly learnable. Therefore, we tackle this problem and propose a procedure to calculate the reward of Equation 1 during the segmentation process. The reward function in Equation 1 expresses the normalized fraction of the correctly segmented points whereas the number of incorrectly segmented points are denoted by e and the number of unsegmented points are denoted by u.

In [TGG20], we defined a mapping function m that is constructed to reward the segmentation at the end of an episode which we also need for the PGT approach. The mapping function $m(j) = k$ maps an assigned segment value $j \in \hat{s}$ to a true segment value $k \in s$. This is done by taking the mode j^{max} over the distribution of assigned segment values within a true segment value k. If a mode j^{max} already exists in the mapping m, the next best mode will be selected. If every assigned segment is already mapped in m, all assigned points within the true segment will be considered as erroneous. In contrast, the mapped assigned segment values within a true segment will be considered as correct. Hence, all points can be categorised as erroneous, unsegmented or correct such that a reward can be calculated (see Equation 1).

Our aim is also to construct a mapping function m such as in [TGG20]. During the grouping process of the PGT, all points of a superpoint j will be assigned to one segment value. For the following analysis, we consider a bidirectional adjacency graph such as the graph which is generated by the VCCS algorithm [PASW13]. To find the mode j^{max}, the coverage c_{kj} is defined which represents the number of points of an assigned superpoint j within a true segment k. Additionally, the coverage u_k is defined which represents the maximum number of unsegmented points that could be potentially grouped together within

Table 1: Reward of the best models of the different approaches. The reward for the ground plane segmentation as preprocessing step is not considered.

Reward (Nr. of Objects)	**PPO**	**D3QN**
Superpoint Growing	0.27 (67)	0.22 (22)
Smallest Superpoint First	0.18 (49)	0.13 (13)

a true segment k by following the neighbourhood edges of j within k. If u_k is greater than every c_{kj} for a certain true segment k, then the largest coverage cannot be clearly determined, i.e. most of the points are unsegmented points within a true segment k. However, if a c_{kj} is greater than u_k, then the superpoint j with the largest c_{kj} could potentially be assigned to the true segment k if it not appears in the mapping m. Thus, a superpoint j could be assigned to a true segment k during the segmentation if (a) $c_{kj} \geq c_{kl} \wedge c_{kj} \geq u_k$ for all assigned superpoints $l \neq j$ and (b) $m(j)$ is not defined yet. This properties must be fulfilled, except for the case when a superpoint without any neighbours exists exclusively in a certain true segment. In this case, the mapping relationship for the assigned segment can be created directly.

An assigned superpoint group can intersect multiple true segments. For instance, consider the rightmost green box of Figure 3. The red superpoint group intersects with six true segments which are represented by black rectangles. In this case, we search for the largest true segment k that fulfils the properties (a) and (b). Hence, it is ensured to find largest mode c_{kj} of the distribution of the assigned segment values within a true segment k such that the reward function of Equation 1 can be calculated during the segmentation process. However, we do not consider the constraint $c_{kj} \geq u_k$ of property (a) in [TGG20] such that superpoints without the relative majority over a true segment k could be potentially assigned. This scenario is relevant if two different segment values appear within a true segment whereas the majority of the points is unsegmented or if not all points are assigned by the agent. Both cases can be neglected in this work since the VCCS algorithm covers the whole point cloud with superpoints. Furthermore, note that condition (a) can only be determined with bidirectional adjacency relationship. Algorithm 1 depicts the mapping creation procedure after a superpoint group j is finalised. We consider a superpoint group j as finalised if it will not change any more during the segmentation process. For instance, if the agent of the superpoint growing environment of 4.1.1 declines every grouping suggestion when considering a certain assigned segment.

6 Experiments

We tested the prototypical superpoint growing environment (see Section 4.1.1) and the smallest superpoint first environment (see Section 4.1.2) in a DRL setting. We used a hexa-core i7 processor and a NVIDIA RTX 2080 graphics card.

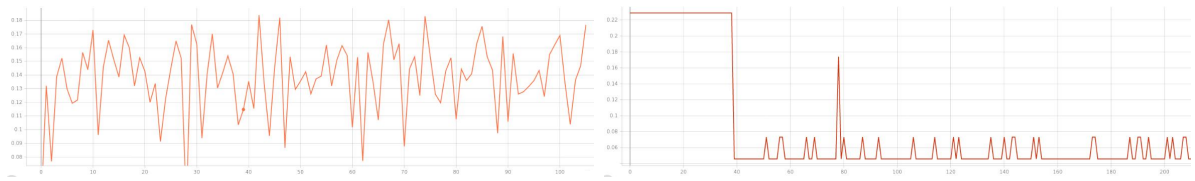

Figure 5: Representative episode rewards of the test episodes with the PPO (left) and the dueling DDQN algorithm (right) during the training.

6.1 Deep Reinforcement Optimisation Algorithms

We applied the D3QN [HMV+18] optimization algorithm with a prioritized replay memory with importance sampling and the PPO algorithm [SWD+17] to evaluate if the sample efficiency of a replay memory of an off-policy network is advantageous. Sampling from the PGT environments can be time consuming as an episode with about 230 steps can last for ≈ 2.76 seconds. The bottlenecks are the rendering of the images which is realised on the GPU with Tensorflow ($\approx$ 10ms) and the action estimation of the neural network ($\approx$ 2ms) which is described in Section 6.1.1. We used the same network architecture for both optimisation algorithms.

6.1.1 Neural Network

A MVCNN [CZZ+19] for the image feature detection with two heads is used. One head outputs the action and the other head output a state value. The MVCNN processes each of the four rendered images (see Section 2.2) by the three CNN units to output a feature vector with 256 units. A CNN unit consist of a convolution and max pooling operation and uses the ReLU activation function. The four resulting feature vectors are processed by a element-wise max operation to choose the best features for further processing. After that, we each apply three fully connected layers with ReLU operations to calculate the action and the state value. The last fully connected layers are activated by a linear activation function. Only the action activation is followed by a softmax layer to confirm or decline a superpoint grouping suggestion. The network has 407.971 weights and the configuration was found on initial experiments with the PPO algorithm.

7 Results

According to Table 1, the models achieved much higher reward in the superpoint growing environment than in the smallest superpoint first environment. Moreover, the training is much more stable with the PPO algorithm (see Figure 5). The corresponding segmentation results are depicted in Figure 6. The model that is trained with the PPO algorithm in the smallest segment first environment produces large superpoint groups and is most far away from the true number of segments which is 53. It grouped more segments but produces a high error as a lot of object boundaries are not respected. The models that are trained in the superpoint growing environments are not perfect but respect the object boundaries

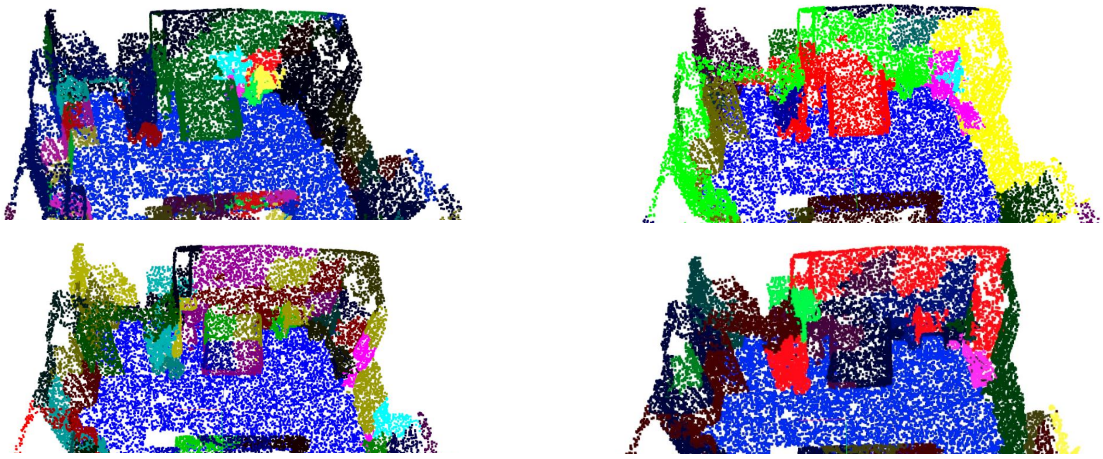

Figure 6: Segmentation results of the best models which correspond to Table 1. The upper images depicts the results with the superpoint growing environment. The results of the smallest superpoint first environment are depicted at the lower images. The images on the left show the results with the PPO algorithm. The images on the right show the results with D3QN algorithm.

better. According to the right plot of Figure 5, the optimization with the D3QN algorithm produces no fruitful results in our experiments as the performance drops drastically when the training begins with the first target network update. After this first update, the model learns very slowly. With the PPO algorithm, we need two by five hours to reach an appropriate improvement of the performance. This is not case after training nine hours with the D3QN algorithm.

8 Conclusion & Future Work

The results of Figure 6 show that the models are far away from a perfect segmentation. However, the learning process with the PPO algorithm in both environments seems to be promising as the training produces high, but noisy, episode rewards. A reason for this behaviour could be the reward function in Section 5. The agent only receives reward if the grouped superpoints are finalised. Hence, some decisions are rewarded very delayed such that it is hard to determine which actions lead to a specific portion of the reward. In this context, we should try other network architectures to reach eventually a more stable learning process. Another consideration to reach more stability would be to lower the learning rate or the clip range of the PPO algorithm, but this produces longer duration of the training. The highest rewards that we have seen during the training with the policies with exploration components such as ϵ-greedy or entropy regularisation are ≈ 0.33. Note, that the maximum performance is limited by the superpoints that are generated by the VCCS algorithm. To reach a fully end to end system, we have to develop a system which generate appropriate superpoints with an adjacency graph automatically. This can be realised by another agent which estimates the parameters of a superpoint algorithm. The long term aim is to train a model on the whole ScanNet data set [DCS+17] and to test it on unknown scenes. Additionally, we have to test

the system with different data sets to test the cross-domain abilities of the approach. Some objects can be composed of other objects and this level of detail is kept fixed with the data set. An appropriate upper bound level has to be chosen with care such that a hierarchical decomposition of the segmented objects can be applied.

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

[Bro14] Joseph L Brooks. *Traditional and new principles of perceptual grouping*, volume 2. Oxford University Press, Oxford, UK, 2014.

[CZZ+19] 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.

[DCS+17] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Niebner. ScanNet: Richly-Annotated 3D Reconstructions of Indoor Scenes. In *Proceedings of the Conference on Computer Vision and Pattern Recognition - CVPR '17*, Honolulu, Hawaii, 2017. IEEE.

[FH04] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient Graph-Based Image Segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[GMR17] E. Grilli, F. Menna, and F. Remondino. A Review of Point Cloud Segmentation and Classification Algorithms. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS*, XLII-2/W3(2W3):339–344, feb 2017.

[HMV+18] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining Improvements in Deep Reinforcement Learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence - AAAI 2018*, New Orleans, Louisiana, USA, 2018. AAAI Press.

[MS15] Daniel Maturana and Sebastian Scherer. VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. In *Proceedings of the International Conference on Intelligent Robots and Systems - IROS '15*, Hamburg, Germany, 2015. IEEE.

[PASW13] Jeremie Papon, Alexey Abramov, Markus Schoeler, and Florentin Worgotter. Voxel Cloud Connectivity Segmentation - Supervoxels for Point Clouds. In *Proceedings of the Conference on Computer Vision and Pattern Recognition - CVPR '13*, Portland, Oregon, USA, 2013. IEEE.

[QSKG17] Charles R. Qi, Hao Su, Mo Kaichun, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proceedings of the Conference on Computer Vision and Pattern Recognition - CVPR '17*, Honolulu, Hawaii, 2017. IEEE.

[QYSG17] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *In Proceedings of the Conference on Neural Information Processing Systems - NIPS '17*, Honolulu, Hawaii, 2017. IEEE.

[RC11] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *Proceedings of the International Conference on Robotics and Automation - ICRA '11*, Shanghai, China, 2011. IEEE.

[RMP+12] Andreas Richtsfeld, Thomas Morwald, Johann Prankl, Michael Zillich, and Markus Vincze. Segmentation of Unknown Objects in Indoor Environments. In *Proceedings of the International Conference on Intelligent Robots and Systems - IROS '12*, Algarve, Portugal, 2012. IEEE.

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

[TGG20] Marcel Tiator, Christian Geiger, and Paul Grimm. Point Cloud Segmentation with Deep Reinforcement Learning. In *Proceedings of the 24th European Conference on Artificial Intelligence - ECAI '20*, Santiago de Compostela, Spain, 2020. IOS Press.

[WLG+17] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-CNN: Octree-based Convolutional Neural Networks for 3D Shape Analysis. *ACM Transactions on Graphics*, 36(4):1–11, jul 2017.

[XHTS17] Y. Xu, L. Hoegner, S. Tuttas, and U. Stilla. Voxel- and Graph-Based Point Cloud Segmentation of 3D Scenes Using Perceptual Grouping Laws. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-1/W1(1W1):43–50, may 2017.

[ZHW+19] 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*, apr 2019.