Semantic 3D segmentation of 3D Gaussian Splats

Assessing existing point cloud segmentation techniques on semantic segmentation of synthetic 3D Gaussian Splats scenes

Karol Jurski
Supervisor(s): Xucong Zhang
EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Karol Jurski
Final project course: CSE3000 Research Project
Thesis committee: Xucong Zhang, Michael Weinmann

Abstract

3D Gaussian Splatting (3DGS) [10] is a promising 3D reconstruction and novel-view synthesis technique. However, the field of semantic 3D segmentation of 3D Gaussian Splats scenes remains largely unexplored. This paper discusses the challenges of performing 3D segmentation directly on 3D Gaussian Splats, introduces a new dataset facilitating evaluation of 3DGS semantic segmentation and proposes use of PointNet++, initially developed for point cloud segmentation, as a 3DGS segmentation model. As the results show, PointNet++ is also capable of performing 3DGS segmentation with performance close to the performance achieved in point cloud segmentation tasks. When taking into account only the positions, 3D Gaussian Splats appear to be more difficult for PointNet++ to process than point clouds sampled from mesh faces, possibly due to their irregularity. However, as shown in the paper, inclusion of size, rotation and opacity of each splat allows PointNet++ to achieve nearly 87% of accuracy, outperforming PointNet++ on point clouds sampled from meshes.

1. Introduction

Recent years have seen an emergence of deep learning methods for 3D geometry understanding tasks, such as 3D segmentation [8]. 3D segmentation is an important task in computer vision with numerous applications in fields such as robotics, autonomous driving [23] and medical analysis [13]. It is a process of assigning fine-grained labels to parts of a 3D object or scene. There exist many 3D representation methods, with meshes, voxels and point clouds traditionally being the most common ones. 3D segmentation methods have been developed for most of them [8]. However, in recent years, new ways of representing 3D scenes have been developed, such as Neural Radiance Fields (NeRF) [14] or 3D Gaussian Splatting [10].

3D Gaussian Splatting (3DGS) [10] is a novel technique of representing and reconstructing 3D scenes from 2D images. Along with an earlier method, NeRF [14], it has revolutionized the field of novel view synthesis. It offers photorealistic quality and fast training and rendering times and is therefore a promising technology, which can find many applications in virtual reality, urban planning and medical imaging [5]. This leads to the need of developing object recognition and segmentation methods for 3D Gaussian Splats.

Current work on 3D segmentation of 3D Gaussian Splats is mostly limited to methods based on performing the actual segmentation in 2D. A notable example is Segment Anything in 3D Gaussians [9], which renders the 3D Gaussians from multiple points of view, performs the segmentation in 2D using the Segment Anything model [12] and then projects the prediction back onto the 3D Gaussians and uses a voting mechanism to determine the final label for each splat. This method is able to achieve decent results, however performing the segmentation in 2D can suffer from problems such as inconsistent per-view segmentation masks, which make the results less reliable [7]. Moreover, it can be hypothesized that performing the segmentation in 2D may lead to losing some of the geometric details not captured in the images that could otherwise be used if the segmentation was done directly on the 3D data.

This highlights the need for creating a method of performing 3D segmentation directly on 3D Gaussian Splats representation, which is the core research question of this paper. Given the fact that 3D Gaussian Splats are in principle a more sophisticated point cloud, it should be possible to utilize some of the existing point cloud segmentation methods and adapt them to work on 3D Gaussian Splats.

As 3D Gaussian Splatting is a new technique, no 3D Gaussian Splats datasets exist yet, therefore developing a method of 3DGS segmentation necessitates preparing a custom 3DGS dataset. This paper describes the considerations related to choosing appropriate kind of dataset and generating 3DGS data based on it.

Furthermore, this paper briefly describes characteristics of some of the existing deep learning architectures capable of 3D point cloud segmentation. Subsequently, it describes how PointNet++ [15], the successor of the pioneering PointNet [4] architecture, can be adapted to work on 3D Gaussians, including incorporating additional features for each point, such as size, rotation and opacity, which were originally not present in the data the point cloud segmentation techniques were developed for.

The key contributions of this paper are as follows:

- presenting a 3DGS dataset preparation pipeline facilitating training and evaluating 3DGS segmentation models,
- showing that PointNet++, a model originally developed for point cloud segmentation, is also capable of performing 3DGS segmentation,
- assessing the influence of using 3DGS and of incorporating the additional features, *i.e.* rotation, scale and opacity on PointNet++ segmentation accuracy.

The next section (Sec. 2) states the problem in a more concrete manner. Section 3 describes the process of data generation and using existing point cloud segmentation deep learning architectures to solve to problem. Section 4 presents the results and compares them to the existing techniques. Section 5 is a discussion about the results, their implications and limitations. Section 6 presents the conclusions of the paper, while outlining recommendations for further research. Finally, Section 7 discusses the aspects related to reproducibility and integrity of this research.

2. Problem statement

This paper discusses and evaluates deep learning solutions for the problem of performing 3D segmentation directly on 3D Gaussian Splats. The input is an unordered set of Gaussian splats, where each splat, as described by Kerbl *et al.* [10], has the following features:

- position, represented as a 3D vector,
- rotation, represented as a quaternion,
- scale, represented as a 3D vector,
- opacity, represented as a real number between 0 and 1,
- direction-dependent color, represented as spherical harmonics.

The output is an assignment of one of c predefined semantic classes to each of the input Gaussian splat.

This paper focuses purely on supervised learning techniques for training the deep learning model, therefore each of the input Gaussian splats must be annotated with a ground truth class label. The process of preparing the data and obtaining the ground truth labels is described in Sec. 3.

3. Method

This section first describes the considerations related to the lack of existing 3DGS datasets and the need of creating a custom one in Sec. 3.1. Subsequently, in Sec. 3.2, it discusses the choice of appropriate existing 3D segmentation deep learning architecture and its adaptation for 3DGS segmentation.

3.1. Dataset

3.1.1 Dataset considerations

One of the biggest challenges of developing a 3D segmentation technique working directly on 3DGS data is the fact that no 3DGS datasets of objects or scenes exists yet, which might be in part due to the relative novelty of the 3DGS method and because object detection on 3DGS has not been well researched yet. Therefore, to be able to develop a 3DGS segmentation method, a custom 3DGS dataset must be created first, based on existing 3D objects or scenes datasets. However, for it to be possible to generate 3DGS representation of a scene from an existing dataset, it needs to fulfill certain criteria. First, it must be possible to obtain comprehensive, multi-camera footage of the scene, such that the scene is covered sufficiently for 3DGS generation to be possible. Furthermore, 3D ground truth should be available to facilitate supervised learning. Moreover, to train and test segmentation, scenes consisting of objects of more than one class are necessary.

A brief review of other papers proposing 3D segmentation methods shows that for the task of semantic segmentation, most often, datasets covering whole scenes are chosen [8], such as S3DIS [2] or SUN RGB-D [19]. However, some of them, such as SUN RGB-D are not suitable, as they do not provide full 3D geometric data, but only selected photographs, which makes it impossible to generate 3DGS data from them. Other datasets, such as Stanford 2D-3D-S Dataset [3] do provide full geometric data, but consist of multiple complicated indoor spaces, making it not trivial to generate 3DGS data from. While being able to use full indoor spaces datasets, such as 2D-3D-S to train and test 3DGS segmentation would undoubtedly be useful and would allow to compare the performance of 3DGS segmentation methods with other techniques such as point cloud segmentation in a straightforward way, generating 3DGS data of scenes like these is complicated enough to deserve to be a research topic on its own and will therefore not be pursued in this paper.

3.1.2 Custom dataset creation

Motivated by the nontriviality of directly generating 3DGS data from one of the existing semantic scene segmentation datasets, this paper chooses a different approach, namely composing custom multi-object scenes based on objects from the ModelNet10 dataset [21]. ModelNet10 is a dataset of nearly 5000 untextured 3D meshes split into 10 different classes, widely used for object classification tasks [16]. Composing custom scenes consisting of multiple objects belonging to multiple categories allows for training and evaluating semantic scene segmentation, while maintaining simplicity and facilitating numerous data augmentation techniques. The detailed process of preparing the data is described in the following paragraphs.

The first step of creating the 3DGS multi-object scenes dataset is obtaining a 3DGS version of each of the ModelNet10 objects. For this, as 3DGS is based on reconstructing 3D scenes from images, comprehensive footage of each object is needed. It is obtained by rendering the object from multiple cameras using Blender [6]. The next step is generating 3DGS data based on the rendered images.

The standard 3DGS generation method, as described in the original paper [10] involves camera calibration and initial sparse point cloud generation using Structure from Motion (SfM) [18]. However, due to simplicity of the single-object scenes, lack of textures and intricate details, use of SfM is impossible in most cases, because not enough key points can be found. Therefore, for 3DGS generation for ModelNet models, use of SfM was avoided altogether by exporting camera transformations from Blender while rendering and by using random point cloud initialization instead of the SfM point cloud, as outlined in the 3DGS paper [10].

As the size of the ModelNet10 models does not necessarily correspond to the size of real-world objects and some of the models are significantly larger or tinier than the rest, the objects' sizes need to be normalized. This is done by measuring the diagonal of the object (understood as the distance between the opposite corners of the 3D axis-aligned bounding box of the object) and scaling it such that the diagonal is equal to 1.

While a 3DGS representation of each ModelNet10 object individually is already sufficient to train a classification model, training a semantic segmentation model requires scenes comprised of points belonging to multiple categories. Therefore the last step of the data preparation process is randomly composing 3DGS scenes from 3DGS representations of the ModelNet10 objects. All the models are randomly split into scenes of 3 to 5 objects (the exact number is chosen randomly for each new scene) placed next to each other, such that each model appears in exactly one scene. Then, for each scene, all its models are placed randomly according to Algorithm 1. The algorithm operates entirely in 2D. The Z coordinate (up-down) of the position of all the objects is set to 0. An example of one of the iterations of the scene composition algorithm can be seen in Fig. 1.

Algorithm 1: Scene composition

```
input  : List $O$ of objects to place
output: Placement (position and rotation) for each
        object in $O$
1  Place($O[0]$, $(0,0)$, RandomRotation());
2  for o $\in O[1:]$ do
3      r $\leftarrow$ RandomRotation();
4      o_polygon $\leftarrow$ Rotate(AABB(o), r);
5      impossible $\leftarrow$
         MinkowskiSum(Union(already placed
         polygons), Scale(o_polygon, $-1$));
6      impossible $\leftarrow$ Dilate(impossible,
         $0.02$);
7      contour $\leftarrow$ Contour(impossible);
8      t $\leftarrow$ RandomPointOn(contour);
9      Place(o, t, r);
10 end
```

The last element needed for training a supervised ML model on the 3DGS data is the ground truth annotation. Thanks to the synthetic scene generation procedure described above, it can be obtained easily. Assuming the generation of 3DGS representations of each ModelNet object is correct and no artifacts, such as floaters occur, it can be assumed that all Gaussian splats in the 3DGS scene representing a single ModelNet object, belong to the class of that object. Then, when composing the multi-object scenes, each Gaussian splat is assigned a class based on the class of the model it comes from.

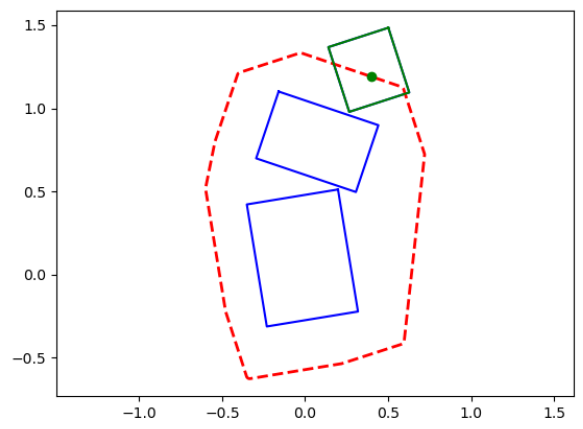


Figure 1. Illustration of an iteration of the scene composition algorithm. Blue polygons are already placed. Red dashed line is the dilated contour of *impossible* (see Algorithm 1). The green polygon (with the green dot at its origin) is being placed at the contour of *impossible* thanks to which it is close to the already placed polygons, while not intersecting with any of them.

3.2. Model

In the search for a deep learning 3DGS segmentation method, it is crucial to realize that 3DGS representation is essentially a point cloud, where each point also has additional attributes, namely rotation, scale, opacity and color. Therefore it makes sense to start the exploration with existing point cloud segmentation models, as long as they are flexible enough to allow for inclusion of the additional attributes. A method worth mentioning is PointNet [4], a Multi Layer Perceptron, which was the first 3D point cloud segmentation method to be proposed. It gave rise to numerous other methods, including convolutional neural networks and transformer networks [8]. Another relevant architecture is PointNet++ (see Fig. 2) [15], a successor of PointNet. Even though it is a relatively simple and not the most recent method, it achieves decent results. A recent paper, PointNeXt [17] has shown that after certain minor modifications to the architecture and improved training methods, PointNet++ is still capable of achieving state of the art results. This is why it has been chosen as a starting point for this paper.

By default, PointNet++ uses position as the only point feature. However, the architecture itself is flexible enough to accept arbitrary number of channels for each point besides the location, which allows for incorporation of all 3DGS features. Specifically:

- position is represented as a 3D vector, as required by the PointNet++ architecture,
- opacity is represented as a real number in the range $[0, 1]$,
- color is skipped, because the training data does not include textures, therefore the color of the 3D Gaussians is

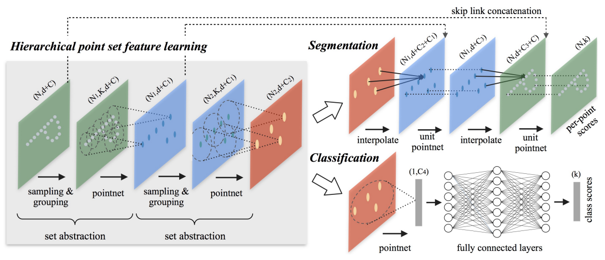


Figure 2. PointNet++ architecture [15]. The input is of size $(N, d+C)$, where N is the number of points/splats, d is the number of spatial dimensions (3 for 3DGS) and C is the number of additional channels on top of the position. The output is of size (N, k), where N is the number of points/splats and k is the number of possible classes.

constant and it would not contribute to the model's decision,

- scale is represented as a 3D vector (x, y, z), where x, y and z mean scaling in the X, Y and Z axes respectively,
- rotation can be represented in multiple ways, including quaternion, rotation matrix, Euler rotation, axis-angle representation, *etc*.; the influence of different rotation representations is assessed in Sec. 4.

PointNet++ requires the input to be of constant size. However, each 3DGS scene has a different number of splats. Therefore uniform sampling is used to get a constant number of 4096 samples for each scene.

4. Experiments

To answer the questions posed in Sec. 1 and to validate the hypothesis that PointNet++ can be used to perform semantic segmentation of 3DGS, a series of experiments is carried out. The results are then analyzed using a quantitative approach, namely by comparing accuracy and mean Intersection over Union (mIoU) of different models. Accuracy and mIoU are the most widely used metrics for assessing performance of semantic segmentation methods [8], [15], [17]. Qualitative approach is then used to verify the correctness of the results for selected scenes.

This section describes the setup used in the experiments in Sec. 4.1. Then, in Sec. 4.2 it introduces the baseline: evaluation of PointNet++ on point clouds sampled from scenes composed according to the algorithm in Sec. 3.1.2. Subsequently, in Sec. 4.3 PointNet++ is evaluated on 3D Gaussian Splats data, the effects of incorporating additional 3DGS features are assessed and the results are compared to the baseline.

4.1. Experimental setup

The models were trained using scenes created according to Sec. 3.1.2. The original train-test data split from ModelNet10 dataset was used, specifically, the training scenes were composed using only the training models and the test scenes were composed using only the test models. To improve the results, data augmentation was used, specifically for each epoch, the scenes were recomposed, *i.e*. the position and rotation of each object in the scene was randomized for each epoch. Additionally, for each epoch, a different (random) subset of samples was used in the training.

All models were trained for 200 epochs using the PyTorch library [1], with the Adam optimizer [11], exponential learning rate decay (Step Decay), weight decay of 0.0001, batch size of 8. The optimal values of initial learning rate (LR) and LR decay rate were chosen using hyperparameter tuning, during which 9 different configurations of LR and LR decay rate were evaluated for each experiment. The hyperparameter tuning was performed on 80% of the training data, after which the model was evaluated on the remaining 20% (validation set). Subsequently, the hyperparameters for which the model achieved the highest accuracy on the validation set were chosen for training the models on the full training dataset in each experiment. The chosen hyperparameter values are reported in each experiment subsection.

4.2. Baseline: PointNet++ on point clouds

Performance of PointNet++ on 3DGS data will be compared with performance of PointNet++ on point clouds sampled uniformly from meshes. Due to the fact that a different dataset is used in this paper, the results from the original PointNet++ paper could not be used. Instead, as a baseline, PointNet++ has been trained on point clouds sampled from meshes from scenes composed according to the description in Sec. 3.1.2. Specifically, for each scene, equal number of samples was taken from each object and samples were taken uniformly from the faces of each object. Each sample is a 3D vector representing its position.

Hyperparameter tuning was performed for the baseline model, the results can be seen in Tab. 1 and Fig. 3. The optimal hyperparameter values were found to be 0.003 for initial LR and 0.8 for LR decay. The model was subsequently trained with these hyperparameter values on the full training dataset three times. Evaluation showed the average accuracy of 84.6% and average mIoU of 71.03%. Qualitative inspection of the segmentation results for selected scenes (see Fig. 4b) confirms that the model correctly learns to segment objects in simple scenes as expected based on [15].

4.3. Experiment: PointNet++ on 3D Gaussian Splats

Finally, the performance of PointNet++ on 3DGS data was evaluated. Several experiments were carried out to measure the accuracy and mIoU for different subsets and different representations of the 3DGS attributes.

Hyperparameter tuning resulted in finding the optimal

	Initial LR		
LR decay	0.001	0.003	0.01
0.9	88.34	88.36	88.97
0.8	88.78	**89.53**	87.91
0.7	87.09	88.77	88.15

Table 1. Hyperparameter tuning results (accuracy, %) for PointNet++ on point clouds

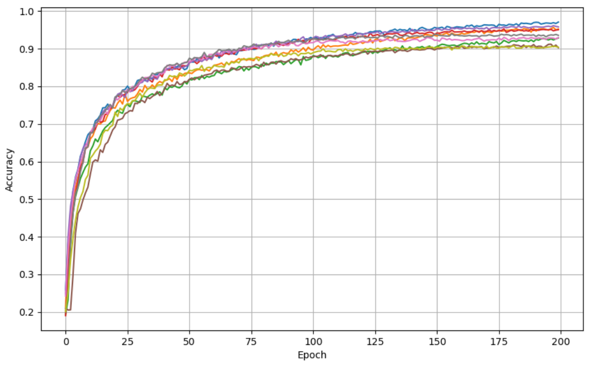
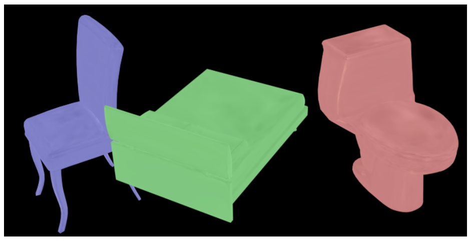
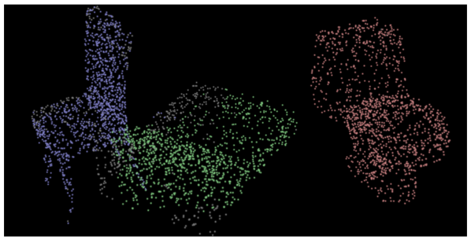


Figure 3. Plot of training accuracy over epochs for all 9 hyperparameter configurations on point cloud (baseline) data. As can be seen, 200 epochs is enough for the models to converge.



(a) Original scene



(b) Point cloud segmentation

Figure 4. Example of a scene and its segmentation performed by PointNet++. Color depicts label: red - toilet, green - bed, blue - chair, gray - other classes. (b) contains only the 4096 points sampled from the faces of the meshes and used by the model.

values of 0.003 and 0.9 for the initial LR and LR decay respectively. The exact results of the hyperparameter tuning can be seen in Tab. 2 and Fig. 5.

PointNet++ trained and evaluated on the position only

	Initial LR		
LR decay	0.001	0.003	0.01
0.9	87.49	**89.39**	71.28
0.8	85.42	88.02	88.63
0.7	82.31	86.22	81.91

Table 2. Hyperparameter tuning results (accuracy, %) for PointNet++ on 3DGS (position only). Results on 3DGS with more attributes show a similar pattern and yield the same optimal values of hyperparameters.

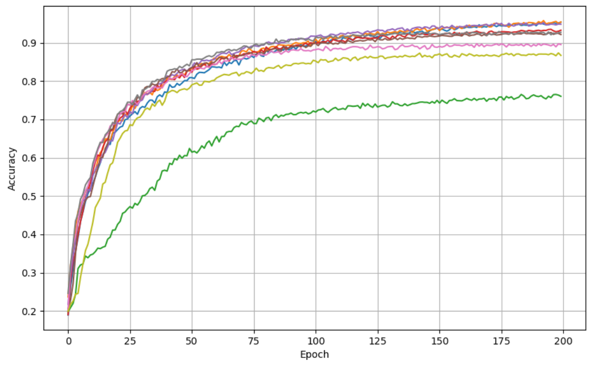


Figure 5. Plot of training accuracy over epochs for all 9 hyperparameter configurations on 3DGS data (position only). As can be seen, 200 epochs is enough for the models to converge. The same holds for 3DGS with more attributes.

(ignoring the rest of the 3DGS attributes) achieved accuracy of 81.19% and mIoU 68.89%, both of which are lower than the baseline. Incorporating the opacity information increased the accuracy to 84.97%, which is slightly higher than the baseline, and the mIoU to 74.97% (3.94%pt higher than the baseline).

Three different methods of including scale and rotation on top of position and scale of each splat were tested. Representing both as a covariance matrix, as described in [10], decreased the accuracy to 81.76% (comparable to the position-only variant, worse than the baseline) and the mIoU to 70.17% (slightly better than position-only, still worse than the baseline). Keeping scale and rotation separate and representing rotation as a matrix resulted in 85.84% accuracy and 75.98% mIoU, better than position+opacity. The best results out of all tested variants were achieved by representing the rotation as a quaternion, which resulted in accuracy of 86.77% and mIoU of 77.40%, both significantly higher than the baseline.

The results can also be seen in Tab. 3. Each of the variants was trained and evaluated three times and the results were averaged.

Qualitative inspection of the segmentation results for selected scenes (see Fig. 6b, Fig. 6c and Fig. 6d) show that

	Accuracy (%)	mIoU (%)
Baseline	84.60	71.03
3DGS: Position	81.19	68.89
3DGS: Position + Opacity	84.97	74.97
3DGS: Position + Opacity + Covariance matrix	81.76	70.17
3DGS: Position + Opacity + Scale + Rotation (matrix)	85.84	75.98
3DGS: Position + Opacity + Scale + Rotation (quaternion)	**86.77**	**77.40**

Table 3. Comparison of PointNet++ accuracy and mIoU on different kinds of data. Each value is an average of the results obtained in three different runs.

the models are generally able to correctly classify each object in the scene, however rarely every splat of the object is correctly classified. Oftentimes certain splats inside an object are assigned a different class; this happens particularly often in parts of objects neighbouring other objects. Comparison of the qualitative results of models trained using different subsets of 3DGS features does not yield any substantial conclusions - each model segments the scenes differently, however this should most likely be attributed to random differences between the models and not inclusion or lack thereof of the additional attributes.

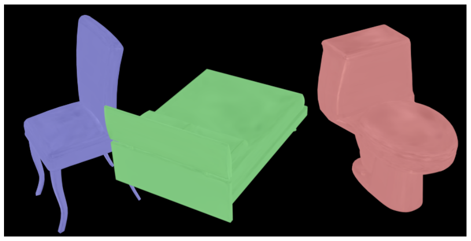
5. Discussion

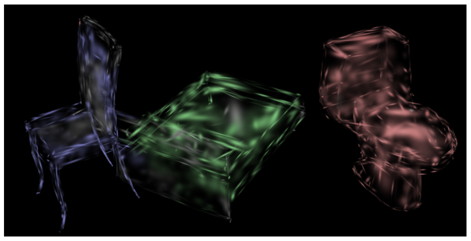
This section reflects on the results in Sec. 5.1 and discusses the limitations of the proposed approach in Sec. 5.2.

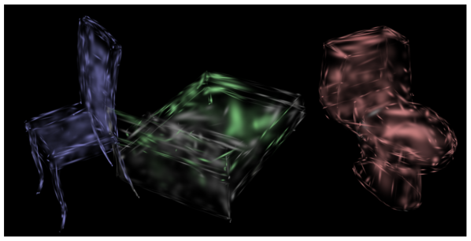
5.1. Reflection

First of all, the results confirm for the first time that, as hypothesized before, 3D Gaussian Splats can be processed directly using deep learning models to perform 3D segmentation. This opens a whole new field of research, which will probably develop as the 3DGS method gets more widespread. Showing that PointNet++ is indeed capable of segmenting 3DGS is the first, but certainly not the last step in developing robust ML models for 3DGS segmentation.

Looking more closely at the obtained results, it is interesting to see that the PointNet++ performance on position-only 3DGS is lower than the baseline PointNet++ performance on position-only point clouds sampled from meshes. This suggests that the 3DGS representation is more difficult to learn from. One possible explanation is that it is less regular than the point clouds sampled from meshes - unlike in the point clouds, the points (splats) are located not only on the outer contour of the object, but also inside the object. Furthermore, in 3DGS the splats are not as evenly spread as in the uniformly sampled point cloud - instead, some parts of the object are represented as multiple small splats (where the complexity is high) and others are represented using few bigger splats (where the complexity is low). On the other hand, the inclusion of opacity infor-


(a) Original scene


(b) 3DGS segmentation (position only)


(c) 3DGS segmentation (position + opacity)

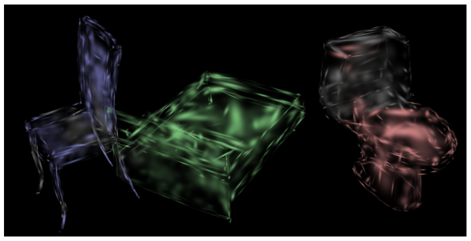

(d) 3DGS segmentation (position + opacity + scale + rotation)

Figure 6. Example of a scene and its segmentation performed by PointNet++. Color depicts label: red - toilet, green - bed, blue - chair, gray - other classes. (b), (c) and (d) contain only the 4096 Gaussian splats used by the model.

mation was enough to significantly increase the accuracy to the level achieved on point clouds. This might suggest, that the biggest problem of position-only 3DGS data was in fact lack of opacity data leading to inability to distinguish between low importance (low opacity) splats and high importance (high opacity) splats.

It should not be surprising that addition of rotation and size information improved the performance compared to the variant without rotation and scale. As mentioned before, some parts of the objects are made of many small splats while others are made of few bigger splats, and making the model aware of the extent of the splats helps it to get a better understanding of the density of different areas of the objects.

It is, however, surprising to see that the accuracy on the irregular 3DGS data, after inclusion of all the available attributes, is higher than the accuracy on the a bit more regular (thanks to being uniformly sampled from a mesh) point clouds. This might suggest that sampling point clouds from meshes actually loses some information about the object (everything that is in between the sampled points is unknown), which is better retained when turning the same mesh into a 3DGS representation. However, as can be noticed by visually inspecting the segmentation results (*e.g.* Fig. 6b), the 4096-element subsets of 3DGS scenes are still not enough to fully cover the surfaces of the objects, and therefore they also lose some geometric information compared to the original meshes, although to a lesser extent than point clouds, thanks to the size and rotation data. The performance increase after adding attributes allowing for better preservation of the original structure of the mesh might therefore suggest that the performance could be further increased by choosing a number of samples bigger than the current 4096. It is, however, unsure whether this increase in the number of samples could be fully utilized by PointNet++, because Qi *et al.* [15] shows that after a certain point, increase in the number of points in the point cloud does not result in increased performance. Evaluating the effect of increase in the number of splats on the segmentation performance of PointNet++, as well as other models, would undoubtedly be an interesting direction of further research.

5.2. Limitations

While the aforementioned results are a significant step towards direct semantic segmentation of 3D Gaussian Splats, they are also limited in certain ways, due to the limited time scope of this research. The limitations, most of which are related to the data used for training and evaluation of the model, are outlined in the following paragraphs.

Synthetic data As the first step, this research focuses solely on simple, synthetic 3D scenes composed of ModelNet objects according to the process described in Sec. 3.1.2. However, this is not a realistic scenario, as in real world, the spaces semantic segmentation is usually applied to, such as real indoor and outdoor scenes, are more complex and contain significantly more objects, which often belong to more than 10 classes. Furthermore, in real world applications, the 3D Gaussian Splats are in most cases not generated based on artificial renders, but on real photographs, which adds a further layer of complexity, due to varying lighting conditions, moving objects and imperfect cameras, which might lead to a lower quality of 3D Gaussian Splats representation, which might in turn pose a greater challenge for a 3DGS segmentation model. Another downside related to using a custom dataset is the difficulty in directly comparing the results to results obtained in other papers.

To overcome this, generating a 3DGS version of existing indoor scene scans datasets would be necessary. However, as already mentioned in Sec. 3.1, it is not a trivial task and could be its own research topic. Generating such a dataset is however necessary to further the research in the field of semantic scene segmentation on 3DGS in the future.

3D ground truth Elaborating on the topic of realistic data, another issue is the ground truth availability. The artificial process of scene composition used in this paper allows for obtaining ground truth labels for each Gaussian splat easily, as described in Sec. 3.1.2. However in real world scenarios, obtaining ground truth so easily is rarely possible, and instead manual annotation is often necessary. It is important to realize though, that manual annotation of 3D Gaussians would be a long and complicated process, especially in complex indoor and outdoor environments. Because of this, in many scenarios, only 2D image annotation is a feasible option. However possessing only the 2D ground truth poses another challenge for the process of segmentation model training. In case of unavailability of 3D ground truth, a different loss calculation method would need to be developed, such as calculating the loss based on the difference between the ground truth annotated image and the 2D render of the same scene rendered using the same camera parameters. This is a research area that could contribute not only to the field of 3DGS segmentation, but of 3D segmentation in general.

Color Color, next to the shape, is an important attribute of real world, as well as artificial 3D objects and could therefore contribute to the decision about assignment of class to object when performing 3D segmentation. It could be therefore hypothesized, that inclusion of color as one of the attributes to the PointNet++ model for 3DGS segmentation could improve the overall performance of the model. This could however not be done in this research because of the choice of the ModelNet dataset, which does not contain textures nor any other color data. Choosing a different dataset with color information and evaluating the effect of including the color on the accuracy of 3D segmentation could be an interesting direction of further research.

Scale Another aspect of 3D objects that could help distinguish between objects of different classes is the size of the object. However, in this paper, the impact of size information on the performance of 3D segmentation could not be examined, because the chosen dataset does not contain reliable scale information and all objects had therefore to be normalized, as explained in Sec. 3.1.2.

ML architecture Moreover, the ML model chosen for 3D segmentation of 3DGS in this paper is PointNet++. While it achieves decent results, it should be noted that in the years following the creation of PointNet++, new deep learning models for point cloud segmentation have been developed, some of them surpassing the performance of PointNet++, such as Dynamic Graph CNN [20] or Point Transformer [22]. It therefore makes sense to research whether use of one of these models could also improve the results of 3DGS segmentation. Moreover, taking into account the differences between point clouds and 3DGS, it is possible that another model could be created, such that it is tailored specifically for processing of 3DGS. This area definitely deserves further research.

Comparison to existing 3DGS segmentation methods While the field of 3DGS segmentation remains largely unexplored yet, there do exist some indirect 3DGS segmentation methods, such as Segment Anything in 3D Gaussians [9]. It would certainly be interesting to compare them to the method used in this paper. However, as Segment Anything in 3D Gaussians [9] relies on visual (not geometric, as PointNet++) information to perform the segmentation, comparing the two methods on a dataset without textures would probably not be a fair comparison. For this, a textured dataset would have to be used.

6. Conclusions

The goal of this research was to answer the questions of whether (and which of) the existing point cloud segmentation methods can be applied for 3DGS segmentation, what is their performance on 3DGS data and how does the inclusion of additional 3DGS features (opacity, rotation and scale) influence the performance. The paper described considerations related to the lack of a 3DGS dataset and the need to create a custom one to facilitate assessing performance of ML models on 3DGS data. Subsequently, after considering various alternatives, PointNet++ was chosen as the point cloud segmentation architecture for the experiments to be carried out on. The experiments allowed to answer the questions stated above, specifically:

- sementic 3DGS segmentation can indeed be performed using existing point cloud segmentation methods,
- PointNet++ is at least one point cloud segmentation method capable of also processing 3DGS, although the similarity between point clouds and 3DGS data allows to hypothesize that other architectures could also work,
- inclusion of the additional 3DGS features indeed positively affects the accuracy and mIoU of PointNet++ on 3DGS data; inclusion of opacity information improves the result compared to the position-only data and additionally including rotation and scale on top of position and opacity further boosts the performance,
- out of all tested variants, a combination of position, opacity, scale and rotation represented as a quaternion allows for achieving the highest performance, namely 86.8% of accuracy and 77.4% of mIoU, which is better than the baseline — PointNet++ on point clouds sampled from meshes (84.6% of accuracy and 71% of mIoU).

While the results are a significant step towards developing robust techniques for 3DGS segmentation, more research is needed, *e.g.*:

- training and evaluating on real-world (non-synthetic) scenes,
- assessing different, possibly more modern point cloud segmentation architectures or developing new methods tailored for 3DGS data,
- using a textured dataset and assessing the influence of incorporating the color data on the performance.

7. Responsible Research

Reproducibility Responsible research necessitates reproducibility. For any research results to be reliable, it must be possible to reproduce them by other scientists to verify their credibility. This is why the code that can be used to generate the dataset and to perform the experiments is published in a GitHub repository [1] such that anyone can use it to reproduce the results by following the detailed steps outlined in the paper.

The 3DGS dataset generation is a long and computationally expensive process. To limit the environmental impact of the research done by our research group, a decision was made to create the dataset of the 3DGS scenes of each ModelNet10 object only once and reuse it by more researchers in our group. The generation was performed by Andrei Simionescu and the dataset was made available at the 4TU website [2] to facilitate easier reproduction of the result, while following the *FAIR* principles.

[1] https://github.com/karol-202/direct-3dgs-segmentation

[2] https://data.4tu.nl/private_datasets/q32Led--j18SvCZ_X4-RLQBsZRy5Ded3Pf6EbdkzXJg

Integrity All data management and experiments were performed with the highest ethical standards in mind. No manipulations (such as leaving out certain data points, data fabrication) were done to the data. The used data is fully synthetic, no individuals were involved in the process of its creation, therefore it should not raise any ethical nor privacy concerns. Both the data [3] and the external code [4] used in this research are licensed in a way making it freely available for research purposes.

References

[1] Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, CK Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Michael Suo, Phil Tillet, Eikan Wang, Xiaodong Wang, William Wen, Shunting Zhang, Xu Zhao, Keren Zhou, Richard Zou, Ajit Mathews, Gregory Chanan, Peng Wu, and Soumith Chintala. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. ACM, 2024. 4

[2] Iro Armeni, Ozan Sener, Amir R. Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2016. 2

[3] I. Armeni, A. Sax, A. R. Zamir, and S. Savarese. Joint 2D-3D-Semantic Data for Indoor Scene Understanding. *ArXiv e-prints*, 2017. 2

[4] R. Qi Charles, Hao Su, Mo Kaichun, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017. 1, 3

[5] Yangsen Chen and Hao Wang. Endogaussians: Single view dynamic gaussian splatting for deformable endoscopic tissues reconstruction, 2024. 1

[6] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. 2

[7] Bin Dou, Tianyu Zhang, Yongjia Ma, Zhaohui Wang, and Zejian Yuan. Cosseggaussians: Compact and swift scene segmenting 3d gaussians with dual feature fusion, 2024. 1

[8] Yong He, Hongshan Yu, Xiaoyan Liu, Zhengeng Yang, Wei Sun, and Ajmal Mian. Deep learning based 3d segmentation: A survey, 2023. 1, 2, 3, 4

[3] https://modelnet.cs.princeton.edu/

[4] https://github.com/yanx27/Pointnet_Pointnet2_pytorch

[9] Xu Hu, Yuxi Wang, Lue Fan, Junsong Fan, Junran Peng, Zhen Lei, Qing Li, and Zhaoxiang Zhang. Segment anything in 3d gaussians, 2024. 1, 8

[10] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. 1, 2, 5

[11] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 4

[12] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3992–4003, 2023. 1

[13] Wenqi Li, Guotai Wang, Lucas Fidon, Sebastien Ourselin, M. Jorge Cardoso, and Tom Vercauteren. *On the Compactness, Efficiency, and Representation of 3D Convolutional Networks: Brain Parcellation as a Pretext Task*, page 348–360. Springer International Publishing, 2017. 1

[14] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 1

[15] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: deep hierarchical feature learning on point sets in a metric space. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 5105–5114, Red Hook, NY, USA, 2017. Curran Associates Inc. 1, 3, 4, 7

[16] Shaohua Qi, Xin Ning, Guowei Yang, Zhang Liping, Peng Long, Weiwei Cai, and Li Weijun. Review of multi-view 3d object recognition methods based on deep learning. *Displays*, 69:102053, 2021. 2

[17] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Abed Al Kader Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. Pointnext: Revisiting pointnet++ with improved training and scaling strategies, 2022. 3, 4

[18] Johannes L. Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[19] Shuran Song, Samuel P. Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 567–576, 2015. 2

[20] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds, 2019. 8

[21] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes, 2015. 2

[22] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip Torr, and Vladlen Koltun. Point transformer, 2021. 8

[23] Andre Ückermann, Robert Haschke, and Helge Ritter. Real-time 3d segmentation for human-robot interaction. In *2013*

IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 2136–2143, 2013. 1