# Point Cloud Annotation Tool

Purna Sowmya Munukutla, Siddhant Jain

January 9, 2019

**Abstract**

We developed a point cloud annotation tool that enables quick labelling of point clouds with minimal human supervision. Our framework converges to the user desired annotation in an iterative fashion with the user giving a seed for each of the target classes and subsequently giving correction strokes for the segmentations predicted by the network. We tested this tool for part segmentation of point clouds in the ShapeNet dataset and found that we were able to achieve target annotations in significantly lesser number of the clicks as opposed to a manual method.

## 1 Introduction

We tackle the problem of annotating a point cloud where the user is interested in assigning a label to each and every point in the point cloud. This is generally a time-consuming, labour intensive process and the aim of our work is to reduce the amount of human supervision needed in this process. The main novelty in our work is to fuse geometric cues along with semantic cues learned from another dataset. As we keep updating the pre-trained network with annotations from successive point clouds, we reduce the work of the user from annotation to verification, making the entire annotation process scalable. In the next few sections we discuss our methodology, followed by some initial results.

## 2 Methodology

Figure 1 gives a quick overview of the system we implemented. The user starts with marking a few points for each of the classes in the point cloud. Ther after we extend each of these regions using geometric features such as nearest neighbours, colors and normals. The choice of feature to be used for this initial region growing is configurable by the user, but through our experiments we found nearest neighbour based region growing to be the fastest and with the least amount of cognitive effort.

### 2.1 Pre-trained network

The pre-trained network to be used in this system can be any segmentation network, pre-trained on an existing dataset in a similar domain. For our experiments, we used PointNet [2] pre-trained on ShapeNet as the base network.

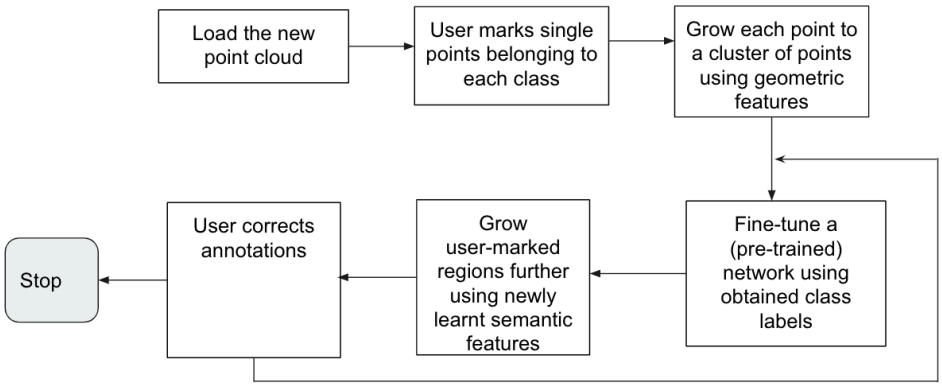


Figure 1: A block diagram for the framework we implemented. User starts with marking one point for each of the corresponding classes. These points are further grown using geometric features (nearest neighbours, normals, colors etc.). A pre-trained network is then fine tuned on these user-marked labels, giving the segmentation output. The user continues the process iteratively till they achieve the desired annotation.

## 2.2 Geometric Region Growing

In our implementation we have various kinds of geometry based region growing algorithms implemented. For point clouds with RGB information, we find color based region growing to be most effective. In the absence of color, normals based region growing is also useful. However, from a user point of view, nearest neighbour based region growing is the easiest to understand and work with and hence for most of our experiments, we stick with a nearest neighbour based region growing method.

## 2.3 Fine Tuning

We fine-tune a base network iteratively based on the given user cues. The initial seed gives a partially labelled point cloud that is used to fine-tune the base network. The segmentation results from the network are shown to the user who can now correct some of these predictions. Subsequently, we finetune further with all the labels (initial seed + corrections) that the user has provided so far. This process keeps continuing till the user reaches a stage where they are happy with the labelled point cloud. At this stage, we fine-tune the network one more time with all the points in the point cloud. Figure-2 ilustrates a sample of the results from this loop of user feedback and finetuning

Based on the number of classes in a point cloud (as configured by the user), we change the last fully connected layer to allow for different number of classes in different annotation use-cases.

## 2.4 Smoothness Loss

Through our initial results we observed that initial segmentation predictions from PointNet are not smooth, i.e we observed that often adjacent points had conflicitng labels. We thus

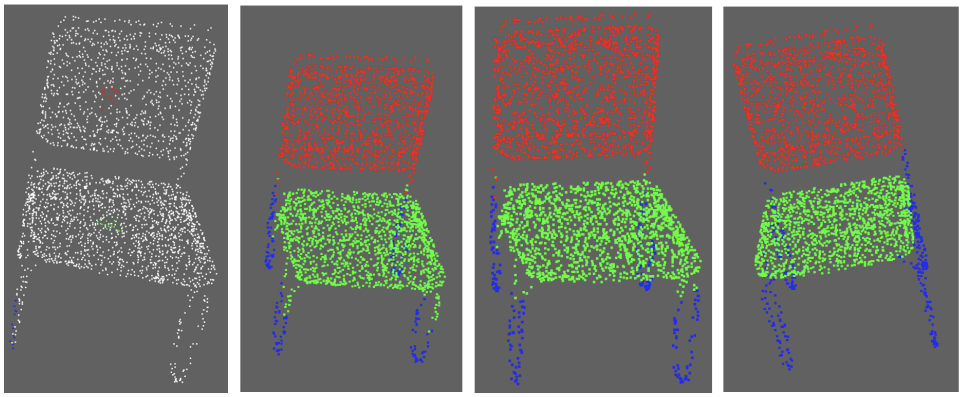

Figure 2: Illustration to show effects of fine tuning in 3 class segmentation of a chair. From left to right i) Initial user seeds ii) Predictions of the network after fine tuning iii) partially corrected point cloud from the user iv) Final prediction after fine tuning with corrections

introduced a *Smoothness Loss* in addition to the regular cross entropy loss. The smoothness loss is formulated as follows:

$$\mathcal{L} = \mathcal{L}_{segment} + \alpha \mathcal{L}_{transform} + \beta \mathcal{L}_{smooth} \tag{1}$$

$$\mathcal{L}_{smooth} = \sum_{i=1}^{points} \sum_{j=i}^{points} KL(p_i, p_j) e^{\frac{-\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}}{\sigma}} \tag{2}$$

$$\mathcal{L}_{smooth} = \sum_{i=1}^{N} \sum_{j=i}^{N} \sum_{k=1}^{K} (p_{ik} \log p_{ik} - p_{ik} \log p_{jk}) e^{\frac{-\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}}{\sigma}} \tag{3}$$

Here, L represents the total loss. $K$ is the number of classes and $N$ is the number of points. $\sigma$ is the variance in the pairwise distance between all points. $x_i$,$y_i$ and $z_i$ represent the 3D coordinates of the point.

Intuitively, we add a loss term if nearby points have divergent logits. Points which are far from each other end up contributing very little to the loss term, regardless of their logits owing to high pairwise distances between them.

Figure 3 shows a qualitative example for the effect of the smoothness loss.

# 3 Experiments

In this section we discuss the initial experiments we ran to validate the effectiveness of our tool. Primarily we were interested in investigating the effectiveness of our tool over a

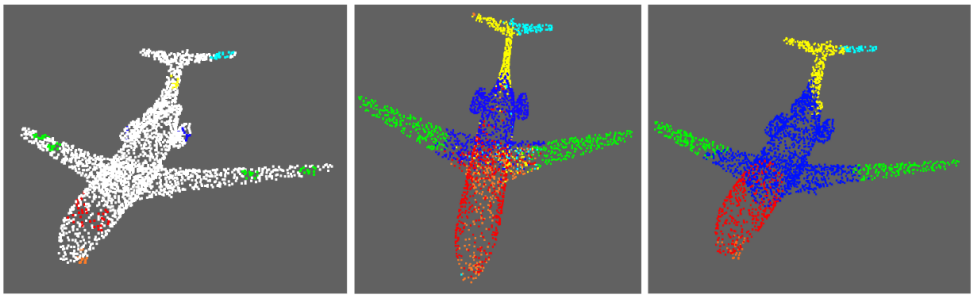

Figure 3: Illustration to show effect of the smoothness loss. From left to right i) Initial user seeds ii) Predictions of the network without smoothness loss iii) Network predictions with smoothness loss

completely manual method. Additionally we also investigated the improvement in annotation efficiency as the total number of annotated point clouds increases.

## 3.1 Improvement over manual methods

We annotated a set of point clouds completely manually and also using our tool, recording the number of clicks it took in each method. Fig 4 show the improvements that came via our methods over a completely manual method:

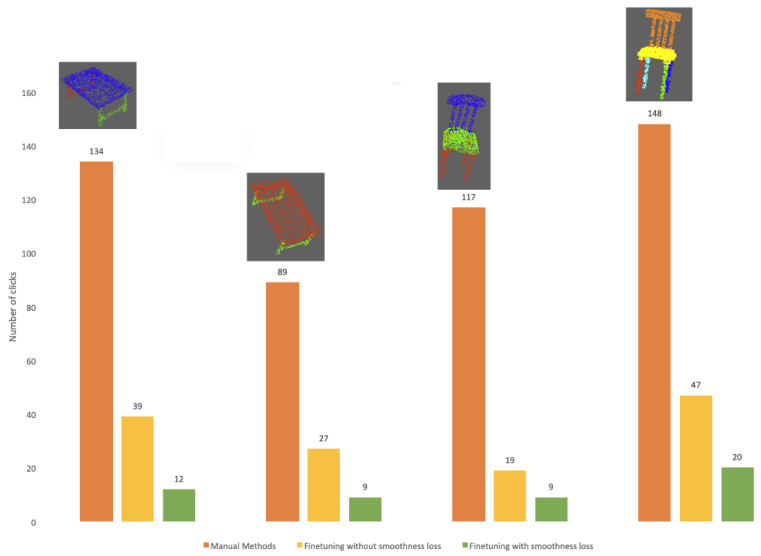


Figure 4: Number of clicks it took to completely annotate a point cloud manually (orange) vs our semi-automatic method without smoothening loss (yellow) vs semi-automatic method with smoothening loss across different point clouds with varying number of classes per point cloud.

### 3.2 Improvement over annotating multiple point clouds of the same type

As the user keeps annotating point clouds, the subsequent fine tuning of the network should make the network more sensitive to the idiosyncrasies of the new dataset. Thus, we should expect lesser user interaction as more and more point clouds are annotated. This hypothesis was validated via an initial experiment that we conducted as illustrated by Figure 5

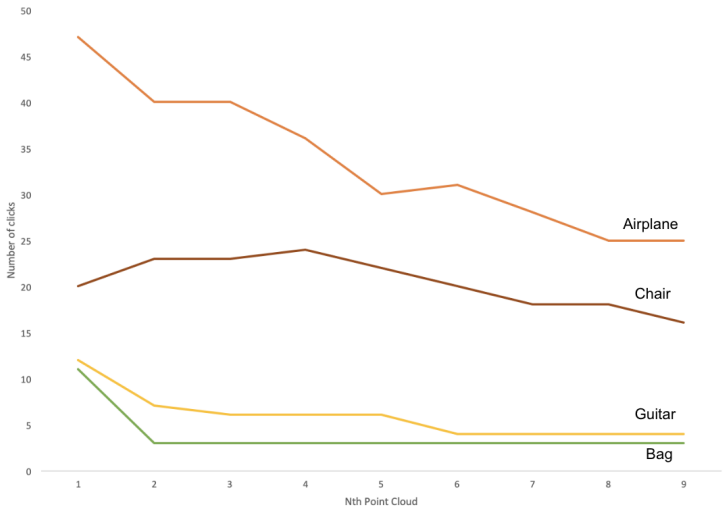


Figure 5: Number of clicks it took annotate subsequent point clouds via the tool. we see a reduction in number of clicks needed as more point clouds of the domain are annotated.

# 4 Future Work

While initial experiments validate the effectiveness of the tool, there is scope to scale up experiments to a larger number of point clouds. In addition we also want to test the tool in its effectiveness for point cloud datasets generated via different sensors, such as LiDARs, Kinect etc.

To this end, we have started work on creating a new validation set for a few categories in ShapeNet. Instead of the synthetic CAD models that are typical of the ShapeNet dataset, we create a new dataset where the models are scanned via an RGB-D camera and the views are reconstructed to get the final point cloud. To test the effectiveness of the tool in creating a new dataset, we will perform a part-segmentation for all the point clouds allowing us to evaluate the tool on various effectiveness metrics such as number of clicks to complete annotation and the reduction in number of clicks with subsequent point clouds.

The dataset thus created will not only help us evaluate the tool but will also be useful to the community at large in testing the generalization capability of various algorithms out there, as they can train on a synthetic dataset like ModelNet and validate the algorithm on a real-world dataset created by us.

As a seed to this dataset, we use the point clouds made available by [1]. Fig 6 gives examples of the some the annotated point clouds in the dataset.

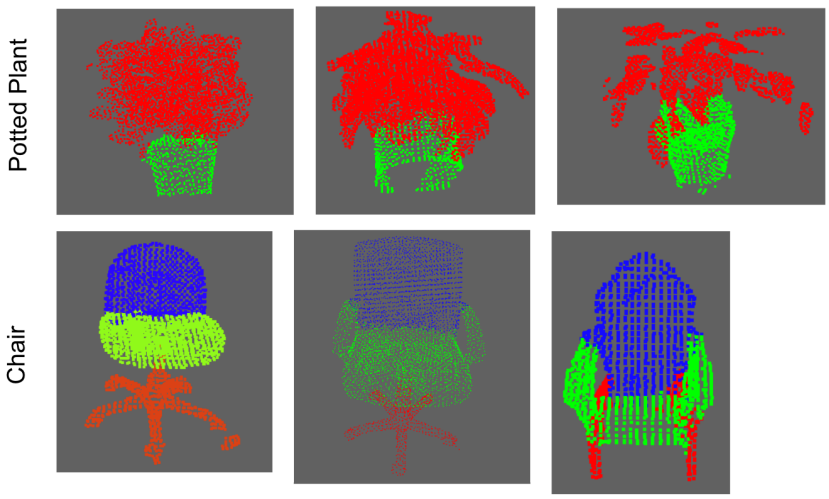


Figure 6: Illustration of some annotated point clouds in the new dataset we are annotating using the tool. These point clouds are noticeable more noisy than the synthetic point clouds in the ShapeNet dataset.

# 5 Conclusions

Fusing geometric features with semantic features learnt via exploiting existing annotated datasets can provide a powerful method to reduce human dependence in the point cloud annotation workflow. Through our tool, we have demonstrated a manifestation of our workflow and tested its effectiveness via simple experiments. In the next stage of this project, we will focus on going through the process of going through a real-world application of the tool by attempting to create a new dataset with class labels for each and every point.

The code for this tool is available here: Code

A video demonstration of the tool in action can be seen here: Demo

# References

[1] Sungjoon Choi et al. "A Large Dataset of Object Scans". In: *arXiv:1602.02481* (2016).

[2] Charles Ruizhongtai Qi et al. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". In: *CoRR* abs/1612.00593 (2016). arXiv: 1612.00593. URL: http://arxiv.org/abs/1612.00593.