Project Progress Report

19-01-2021

Semantic Segmentation of Robotic Arm using RGB-D Data

Onur Berk Töre
otore19@ku.edu.tr

Farzin Negahbani
fnegahbani19@ku.edu.tr

Buğra Can Sefercik
bsefercik@ku.edu.tr

Project Overview

Most robots nowadays are equipped with high-quality depth cameras to understand their surroundings. At first glance, it seems like these sensors combined with current advanced deep learning approaches can solve vision problems. However, many challenges exist on the way to a robust environment perception for robotic arms, especially during interaction with unknown objects. One of crucial and complicated challenge is semantic segmentation or classifying pixels in a digital image into semantically meaningful ones. This problem becomes harder specifically during the interaction when robot arm touches the object of interest. To address this problem, employing a semantic segmentation method that segments the robotic arm in RGB-D frame and extracting the arm's points from the environment with low latency is suggested. This method can also help with the automatic extrinsic calibration of the robotic arm and RGB-D sensors that is a promising future direction of this project. To this end, we aim to gather a custom dataset and adapt a recent semantic segmentation method for our purposes.

1 Progress

In this section, after giving a brief list of achievements, we detail each step by providing samples and results. So far, we have been able to accomplish the items below.

1. Dataset
    - Data gathering plan.
    - Data collection code.
    - Semi-supervised labeling code.
    - Gathering and labeling toy dataset.
2. Baseline Model
    - Setup the baseline code.
    - Adapt a baseline architecture to train with our data.
    - Adapt visualization and testing code.
    - Remove unnecessary parts of the model.

Figure 1: Experiment environment showing the robot, Kinect, table, and one object.

1.1 Dataset

The baseline code is mainly tested with ScanNet [1] dataset. ScanNet is one of the largest datasets available with more than 20 classes for 3D segmentation task. ScanNet data samples are categorized into different scenes, each scene containing multiple RGB-D samples and corresponding segmentation labels, high-quality reconstructed mesh, decimated reconstructed mesh, and aggregated(all images from a scene) labels.

To make our dataset community-friendly, and move fast, we utilize a similar data format. Each scene differs in orientation and position of Kinect and illumination of the environment. Each sample of the scene differs in arm position. For each scene, we first extrinsically calibrate the Kinect with the robot arm to have all the transformations in case we need it. The procedure of collection for a scene starts from getting samples from the environment without the presence of the robot arm. Then with the presence of the robot in the Kinect's field of view, the arm starts executing a task and we collect consecutive frames. Following collection of the data, a data annotator categorizes each point to background or arm point using the priors from the background sample and empirical settings from the user. Furthermore, it stores the labels in the desired format. Figure 2 shows a sample of scene, background, and labeled data.

In addition to our point cloud and RGB data, web build high quality and decimated reconstructed surface meshes that may provide more options and data for

(a) Background sample (b) Data sample (c) Extracted arm point cloud

Figure 2: Data collection and labeling process of the customized dataset.

those who wish to use this dataset in the future. Also, our code base can apply average filter on point cloud data with a desired number of iteration (the more iterations, the smoother the resulted surface). Samples of filtered data is depicted in Fig. 3.

1.2 Baseline Model

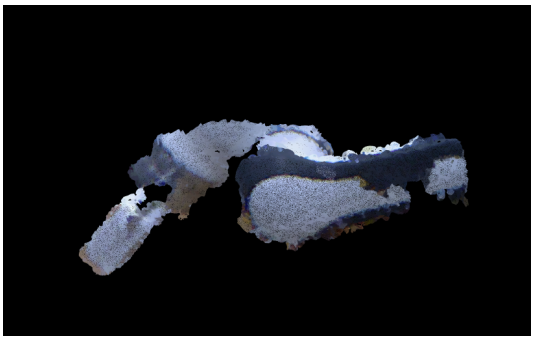
In this project, we decided to start with PointGroup from Li et al.[2] (CVPR2020) as our baseline model. PointGroup come second on the mAP_{50} in ScanNetV2 and 3DSIS datasets.

As the first step, Pointgroup [2] code adapted to work with our own dataset. The first step of adaptation consists of setting the parameters, implementing data loaders, and tweaking the training and testing code of the baseline. Moreover, the PointGroup [2] method trained and tested on the toy dataset providing a rich feedback on the baseline.

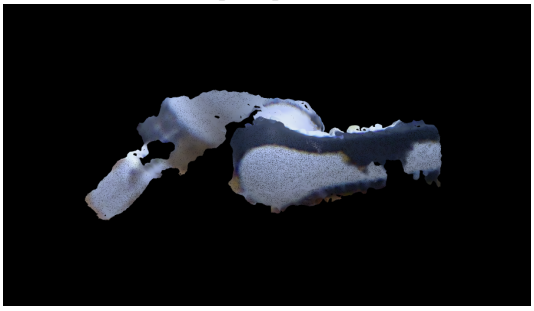
Fig. 6 shows the train and validation loss of the baseline model while training on our own dataset. The toy dataset consists of one scene with 76 training and 22 validation samples.

Class	**mAP**$_{50}$	**mAP**$_{25}$
Arm	0.909	0.977
Background	1.00	1.00
Average	0.955	0.989

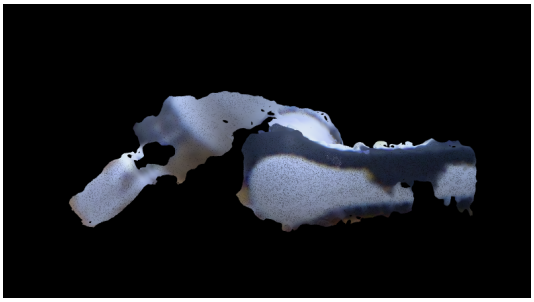
Table 1: **Cropped Pointgroup Results on the validation set.** This table shows the cropped Pointgroup performance on the validation set of our toy dataset. The results reported in terms of average precision and IoU with 25% and 50% intersection over union metrics.



(a) Sampled point cloud

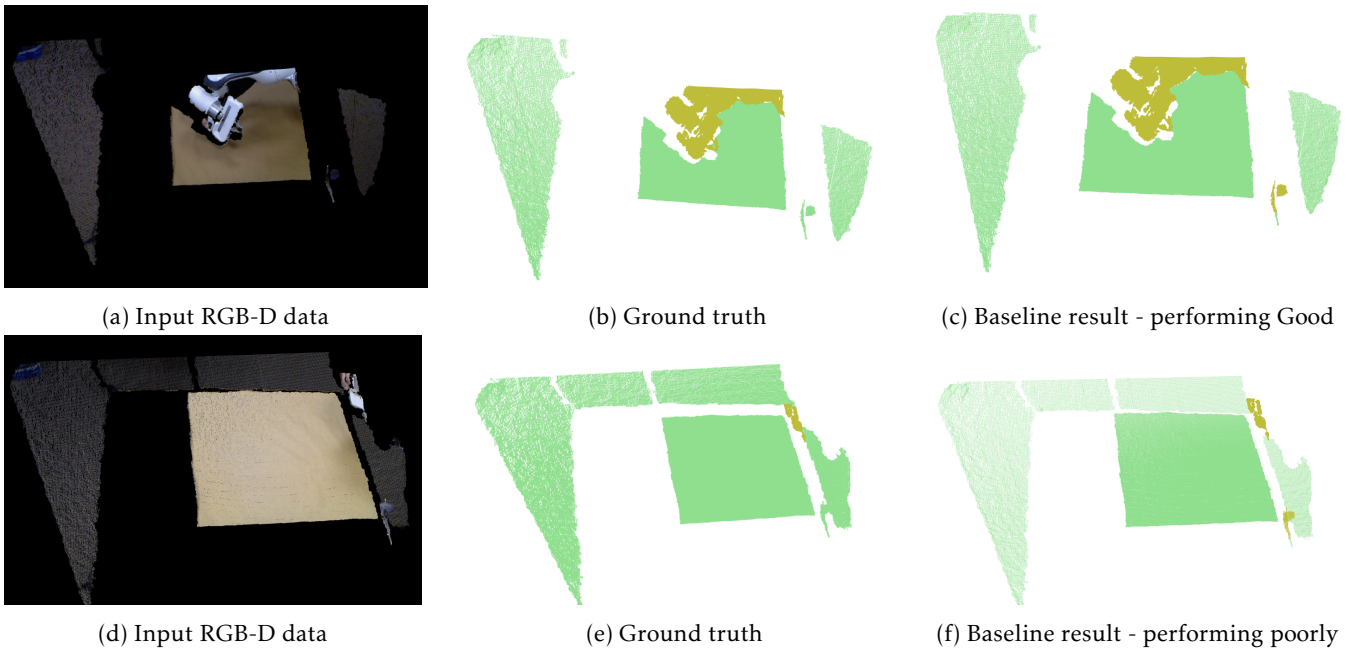


(b) Filtered with 5 iterations



(c) Filtered with 10 iterations

Figure 3: **Filtered surfaces.** Reconstructed surfaces are represented as mesh files in our dataset. To generate point cloud data, after sampling the mesh (sampled from mesh point cloud (a)), an average filtering can be applied to further smooth the surface. Two samples of filtered point cloud with 5 and 10 iterations illustrated in (b) and (c).



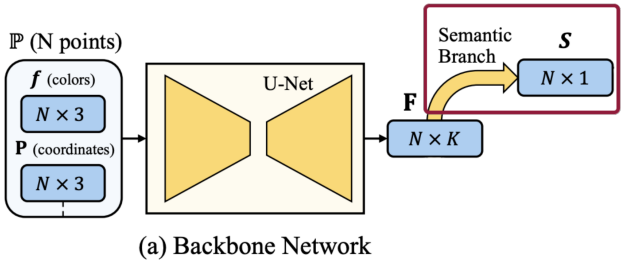
(a) Input RGB-D data (b) Ground truth (c) Baseline result - performing Good

(d) Input RGB-D data (e) Ground truth (f) Baseline result - performing poorly

Figure 4: **Qualitative samples.** Predicted from the validation set by the baseline model trained with our toy dataset. RGB-D input data (a) visualized besides the ground-truth (b) and a well performing prediction (c). In the second row, a sample visualized where the baseline performed low (f), plus, the GT (e) and the sample data (d). Colors in the figures (b), (c), (e), and (f) represent the semantic class of the points. The RGB color of the point omitted for the visualization purposes.

1.3 Increasing Model Throughput

The Pointgroup model has a second part after the semantic segmentation which estimates the clusters of multiple objects to further increase the accuracy of the semantic and instance segmentation. After seeing the qualitative results of the first segmentation part, we decided not to use these part of the model. The resulting cropped model can be seen in Fig.5. The qualitive results on validation set can be seen in the Fig. 4 and the accuracy on the validation set in terms of mean average precision and IoU of our cropped model can be seen in Table.1



(a) Backbone Network

Figure 5: **Customized PointGroup Overview.** Red boxed area shows the semantic branch that we are interested in.

The project aims is to increase inference speed of the model. The inference speed best measured with Frames Per Second (FPS). While 10 FPS may be enough for our real-time measurements, with the overhead of underlying robotics system, we aim to achieve around 25-30 FPS in practice.

Method	Inference Time	FPS	# Params
PointGroup	2.59 (Seconds)	0.38	7714710
Semantic PointGroup	0.06 (Seconds)	16.6	7532162

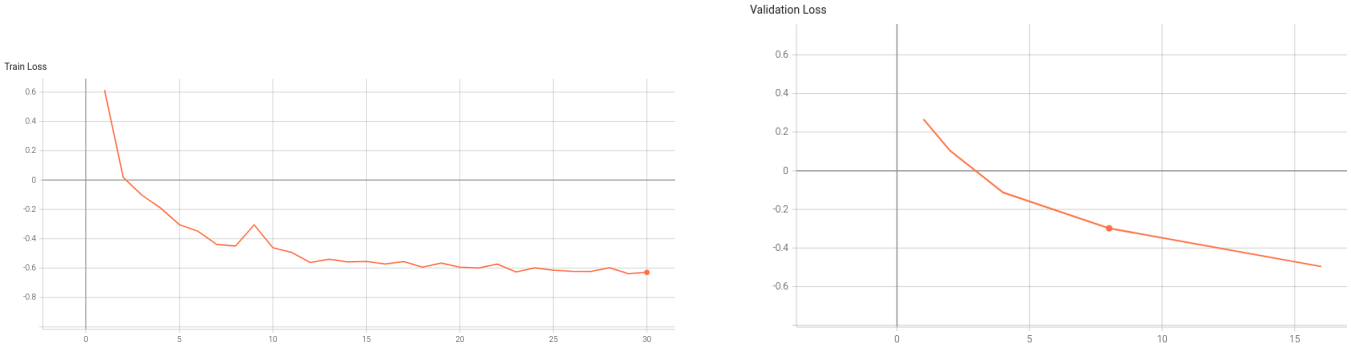
Table 2: **Inference Speed.** PointGroup is the baseline model without any modification. The semantic PointGroup is the version that uses only the semantic prediction part of the PointGroup. Tests are done using Nvidia GTX1080Ti.

The test (Table 2) reveals that while PointGroup model achieves around 0.38 FPS speed on our environment, the cropped Pointgroup achieves around 17 FPS.

Currently, we are investigating effects of new modifications for making the sub-manifold U-Net faster such as using squeezed architectures.

2 Discussion

As low and well performing samples of the baseline method shown in Fig. 4, we investigated the reasons be-



(a) Train Loss (b) Validation Loss

Figure 6: **Train and Val loss.** The baseline model trained for 30 epochs and the validation and train loss visualized to make sure that we are not over-fitting and the model is training.

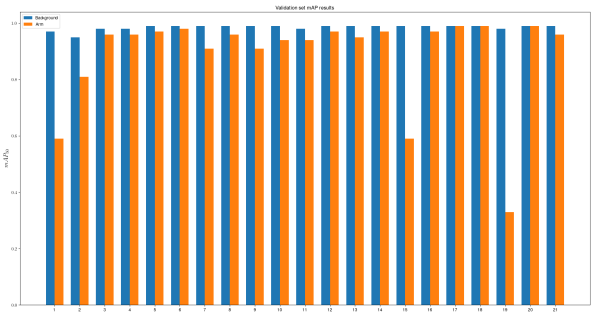


Figure 7: **Prediction Accuracy.** Shows the prediction accuracy on each sample of the validation set. Bars in orange represent the background and blue bars shows arm prediction accuracy.

hind the low performing sample. We believe that our toy dataset is not large enough for model to generalize on the samples where few points belong to the robot arm. Also, because currently the toy dataset is very small, having two low performing samples affected the results by a lot. Fig. 7 shows the accuracy of the model on each sample, we can see that the model performs mainly well.

3 In Progress

After making sure that we have a decent baseline and data collection code, next phases such as gathering final dataset and modifying the architecture are under progress.

4 Code

We organize and develop our code using git service. The latest project code can be accessed via Github.

References

[1] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5828–5839, 2017.

[2] Li Jiang, Hengshuang Zhao, Shaoshuai Shi, Shu Liu, Chi-Wing Fu, and Jiaya Jia. Pointgroup: Dual-set point grouping for 3d instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4867–4876, 2020.