Scene understanding using Segmentation techniques with 3D LiDAR Point Clouds

Sandeep Jalui, Prateek Behera
Electrical and Computer Engineering Department, University of Florida, Gainesville, USA
sjalui@ufl.edu, prateekbehera@ufl.edu

Abstract

The application of Pointclouds has increased a lot with the recent developments in deep learning techniques. Pointclouds can be used in autonomous vision, robotics, analysis of particular sites and many more. Pointclouds can be alternative to 2D image where images capturing and analysis has limitations. Scene segmentation helps us understands the details of every objects in site scenario thus effective analysis can happen. Segmentation tasks helps in clearly partitioning of each object in a pointcloud dataset so than it can be understandable to us. Currently, manual segmentation of pointclouds data happens using 3D CAD tools which is very labour intensive. This process needs to be automated as the pointclouds application can be used in robotics or cars as well. But scene understanding and segmentation of pointclouds can be challenging tasks due to complexity, sparse and dense nature of pointclouds. It is not similar to conventional convolutional neural network. Thus, this project focuses on solving this complex problem by using KPConv algorithm. KPConv uses kernel weight matrices which are multiplied to all the input pointclouds based on the relative position of kernel points. S3DIS dataset is used for the demonstration and testing which has total 13 classes and contains every room objects information such as ceiling, floor, chairs, table and many more. KPConv was able to get max IoU score approx 98% for ceiling. The average IoU score of all the objects is approximately 66%. Cloud compare tool is used for visualization of the output results which are shown in this report.

Keywords: segmentation, lidar, pointclouds, semantic, autonomous, kpconv

1 Introduction

Due to developments in the field of Autonomous Driving, Medical Image analysis and Virtual Reality, the importance of learning from 3D point cloud data from a LiDAR sensor has recently come to the forefront. One of the applications of this type of data is scene understanding using semantic segmentation with 3D data. 3D segmentation enables a more comprehensive form of scene understanding because 3D data like point clouds contain richer information about geometric attributes, and scale of objects in a scene. Moreover, problems like occlusion that can occur in a 2D image, and the lack of distance and depth information can lead to some background noise and hinder the capability to differentiate between similarly colored objects. This can be solved by using 3D data to perform semantic and instance segmentation.

In this project, we use 3D LiDAR point clouds and train a deep learning model to perform semantic scene segmentation. We use the KPConv (Kernel Point Convolution) [3] method to train an FCNN model on scene segmentation data. We use S3DIS which is the Indoor Scene Segmentation dataset. The input data in the form of (x, y, z, r, g, b) point clouds which we pass through a KP-FCNN model for segmentation. KP-FCNN involves an encoder for the initial features and a decoder for the final lower-level features. The output obtained show a 3D scene and contain boundaries to separate different objects and parts in that scene.

2 KPConv

2.1 Why Kernel Point Convolution?

KPconv is a method of point convolution which can take point clouds as input and compute the convolution output without any intermediate representation of the kernel space. Other concurrent works on point cloud representation learning have several shortcomings. In particular projection networks first transform the point clouds

to 2D images or 3D grids and then apply convnets to process them. On the other hand, graph conversion networks aggregate information over edges which ignore the local surface defined by the neighbors which makes it hard to capture local shapes in a simple and efficient.

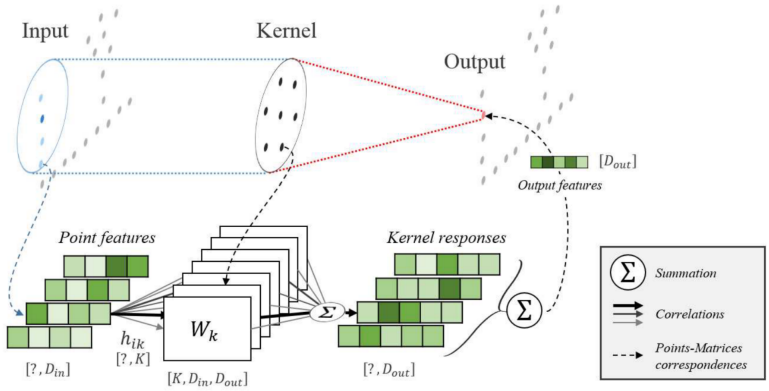

Figure 1: Kernel Point Convolution.

KPConv is a new type of point convolution that learns the kernel function to compute point-wise filters and also introduces a branch that increases the representation of local neighbours. While KPConv uses a 3D sphere for each element of the kernel, 2D convolution on the other hand uses the center position of each grid box. KPConv uses several 3D points in continuous space where the kernel points can be predefined or learned as parameters.

2.2 Architecture

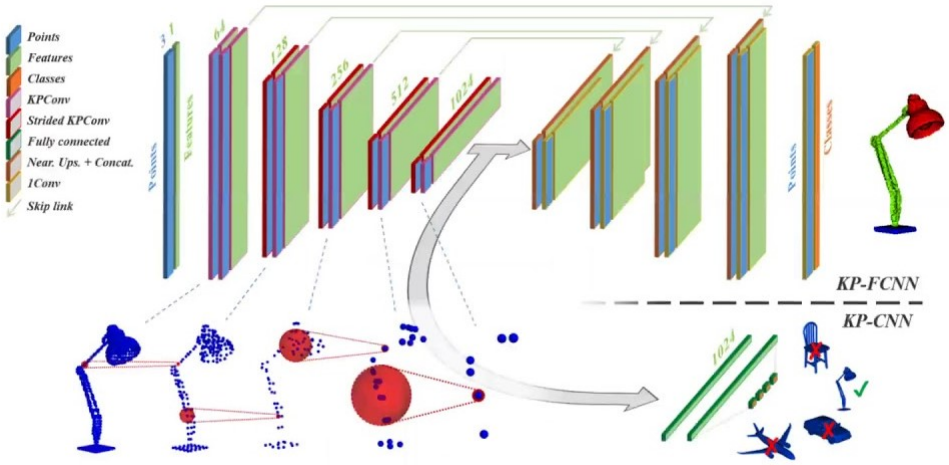

Figure 2: KPConv Network Architecture.

We use KP-FCNN model architecture for segmenting (semantic) S3DIS Indoor Spaces Dataset. KP-FCNN is a fully convolutional neural network for segmentation that involves an encoder for learning initial high-level features and a decoder for the final lower-level features. The final pointwise features are obtained from the output of the decoder which uses nearest-neighbour upsampling. There are skip links that ensure that the features pass between the layers of the encoder to the corresponding layers of the decoder. The features obtained and combined and a 1x1 convolution is performed to get the points and class probabilities.

2.3 How KPConv Segments the Dataset?

The 3D point cloud in the S3DIS dataset cannot be segmented as a whole in one go and have to be segmented in small sub clouds. During model training, the 3D spheres are randomnly picked in the scenes whereas, during

testing, they are picked in regular intervals so that every point is tested multiple times. The color channels are used as features to segment the space. Then, the output predictions/probabilities are averaged for every point similar to a voting system in an ensemble. All the points representing empty spaces have all features as zero.

3 Dataset

S3DIS Dataset (Stanford 3D Dataset) is an Indoor 3D dataset that contains 3D Lidar Point clouds of indoor areas of a large building. The rooms are split into 6 areas. 5 areas are used for training and 1 for evaluation (typically Area_5). The input data is a raw point cloud containing information about the x, y and z coordinates of thousands of points in the point cloud and the R, G and B information of each point that helps identifying and separating the objects, i.e., each point in the raw input point cloud has the following data – (x, y, z, r, g, b). The following classes of data can be found in these areas: Ceiling, floor, wall, beam, column, window, door, table, chair, sofa, bookcase, board and clutter.

4 How to run code

Unzip the project folder. The code for training is saved in training_S3DIS.py file. For testing use the test_model.py. Make sure the project is in a PyTorch and CUDA enabled environment.

Download the dataset from S3DIS[2] and unzip it where the project files were unzipped. To run the code, open the project folder in the terminal window and run the following commands: -

sbatch kpconv_train.sh

sbatch kpconv_test.sh

sbatch kpconv_plot_convergence.sh

After running the code, the train results are generated in the results folder as .ply files. These .ply files are generated after every 50 epochs for training and one file for testing. Cloud compare tool is used to visualize these .ply files. Ensure to convert to scalar field after loading the .ply files in cloud compare and before visualizing.

5 Training and Hyperparameter Tuning

The model was trained on UF Hipergator GPU with 1 GPU node, 1 CPU with 50GB RAM memory. PyTorch framework is used to build the KPConv model. The hyperparameters that were tuned to get optimum results were Input parameters, Model parameters, KPConv parameters and Training parameters. The experiment was run with 393 epochs logging every 50 epochs. The following figure shows the values of the hyperparameters that gave the best results.

```
# Input parameters
# ****************

dataset = S3DIS
dataset_task = cloud_segmentation
num_classes = 13
in_points_dim = 3
in_features_dim = 5
in_radius = 1.200000
input_threads = 10


# Model parameters
# ****************

architecture = simple resnetb resnetb_strided
equivar_mode =
invar_mode =
num_layers = 5
first_features_dim = 128
use_batch_norm = 1
batch_norm_momentum = 0.020000
```

```
# KPConv parameters
# *****************

first_subsampling_dl = 0.030000
num_kernel_points = 15
conv_radius = 2.500000
deform_radius = 5.000000
fixed_kernel_points = center
KP_extent = 1.200000
KP_influence = linear
aggregation_mode = sum
modulated = 0
n_frames = 1
max_in_points = 0

max_val_points = 50000

val_radius = 51.000000
```

```
# Training parameters
# *******************

learning_rate = 0.010000
momentum = 0.980000
lr_decay_epochs = 1:0.984767 2:0.984767 3:0.984767
grad_clip_norm = 100.000000

augment_symmetries = 1 0 0
augment_rotation = vertical
augment_noise = 0.001000
augment_occlusion = none
augment_occlusion_ratio = 0.200000
augment_occlusion_num = 1
augment_scale_anisotropic = 1
augment_scale_min = 0.900000
augment_scale_max = 1.100000
augment_color = 0.800000

weight_decay = 0.001000
segloss_balance = none
class_w =
deform_fitting_mode = point2point
deform_fitting_power = 1.000000
deform_lr_factor = 0.100000
repulse_extent = 1.200000
batch_num = 6
val_batch_num = 10
max_epoch = 350
epoch_steps = 350
validation_size = 50
checkpoint_gap = 50
```

Figure 3: Hyperparameters

6 Results and Analysis

```
results/Log_2022-11-21_06-02-27
   mean   | ceiling   floor     wall      beam      column    window    door      chair     table     bookcase  sofa      board     clutter
----------|-------------------------------------------------------------------------------------------------------------------------------------
   61.9   |   91.7      97.9      79.0      12.4      21.7      51.1      57.0      87.0      74.4      70.5      53.3      55.6      52.6
----------|-------------------------------------------------------------------------------------------------------------------------------------
   66.0   |   93.3      98.2      82.5      0.0       19.6      55.2      68.9      90.8      81.1      74.9      65.5      70.6      57.6
```

Figure 4: Training results - mean IoU.

Fig 4 and fig 5 tabulates and plots the mean IoUs for different classes. It can be seen that a few objects are getting segmented at a very high intersection over union for eg. ceiling, floor, wall, table and chair have mean IoU of more than 80% whereas classes like beam and column has lowest IoU which indicates that the model is finding it difficult to segment these objects. The reasons for this is Beams and columns can often be easily mistaken for other objects, in particular, walls. Other objects like bookcase, sofa etc were segmented to a respectable accuracy.

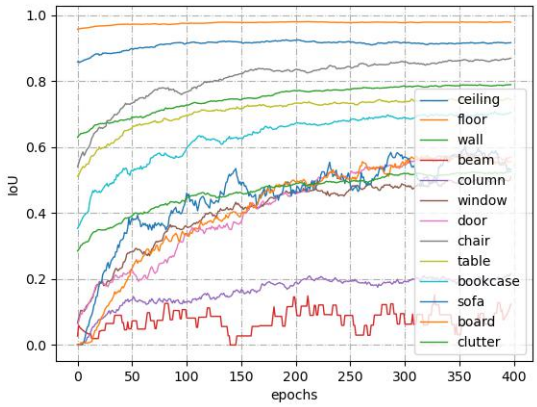

Figure 5: Resultant meanIoUs for different classes

The test results have been tabulated in fig 6. The confusion matrix of subclouds and the final confusion matrix for the full clouds are tabulated below. The test results are similar to the train results which means there is no overfitting. It can be observed from the test results, beam objects are been completely left out of the segmented results. Notice that Beam object has 0.0% IoU score.

```
Confusion on sub clouds
66.12 | 92.50 98.18 82.47  0.00 16.49 54.73 73.88 90.09 80.30 75.68 66.94 69.74 58.52


Reproject Vote #99
Done in 5.7 s

Confusion on full clouds
Done in 3.2 s

--------------------------------------------------------------------------------
66.45 | 93.03 98.16 82.54  0.00 17.08 56.73 73.13 90.58 80.83 75.18 68.08 70.43 58.12
--------------------------------------------------------------------------------
```

Figure 6: Test results - mean IoU.

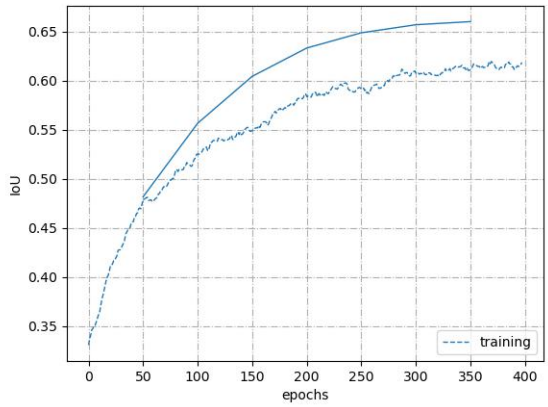

Figure 7: Convergence Plot across epochs

Finally, the convergence plots for training across epochs(0-393) have been plotted in fig 7. It can be observed that as number of epochs increases, the mIoU increases. After 200 epochs, it starts to converge finally converging to 66.45%. Furthermore, Fig 8 shows the visualization of the predicted segmentation results after every 50 epochs and details the corresponding mean IoU. This visualization is for Area 5 in the S3DIS dataset.

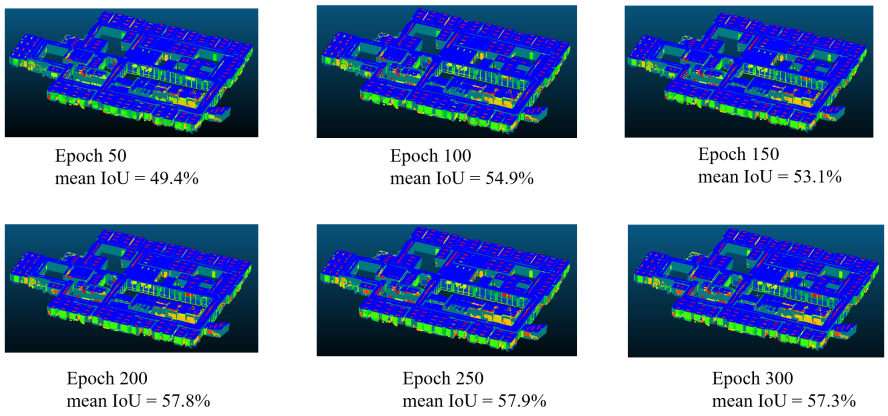

Figure 8: Visualization of segmented results for Area 5 v/s Epochs.

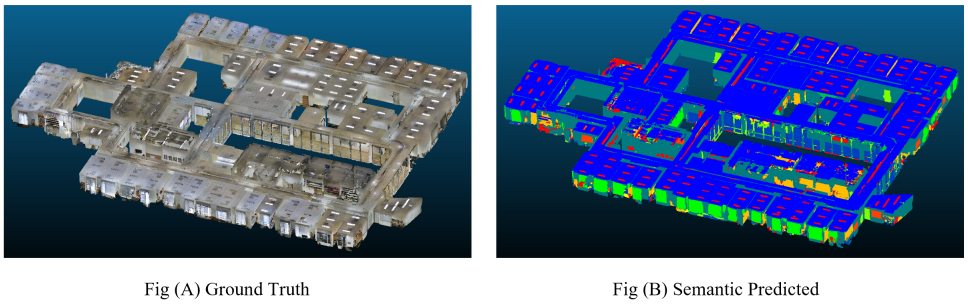

Figure 9: Visualization of segmented results at 350 Epochs v/s Ground Truth - Top View.

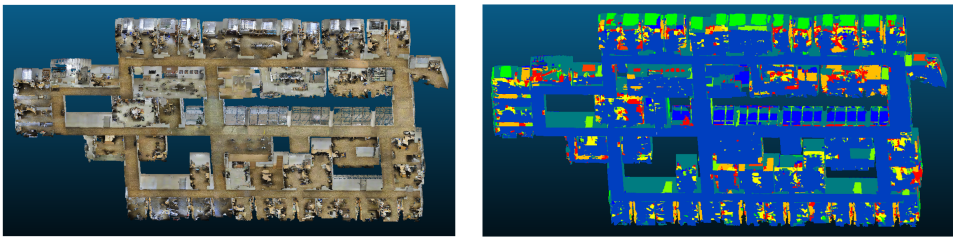

Fig (C) Ground Truth

Fig (D) Semantic Predicted

Figure 10: Visualization of segmented results at 350 Epochs v/s Ground Truth - Bottom View.

Fig 9 and fig 10 show the top view and the bottom view of the predicted segmentation results after 350 epochs. Both the visualization confirms the results that was observed in the IoU. It can be observed that the ceiling(dark blue) and the floor(blue) are segmented accurately. Fig 11 shows the segmented and ground truth of one of the rooms in Area 5 with chairs(green) and tables(yellow). Fig 12 shows segmented and ground truth of objects like bookcase, windows, clutter, sofa and chair.

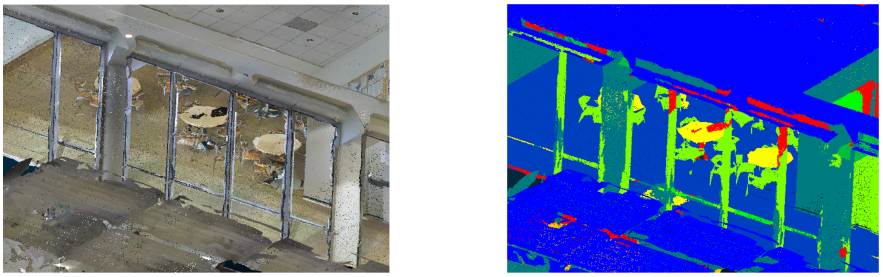

Fig (F) Ground Truth

Fig (E) Semantic Predicted

Figure 11: Visualization of segmented results at 350 Epochs v/s Ground Truth - Individual Room (Cafeteria).

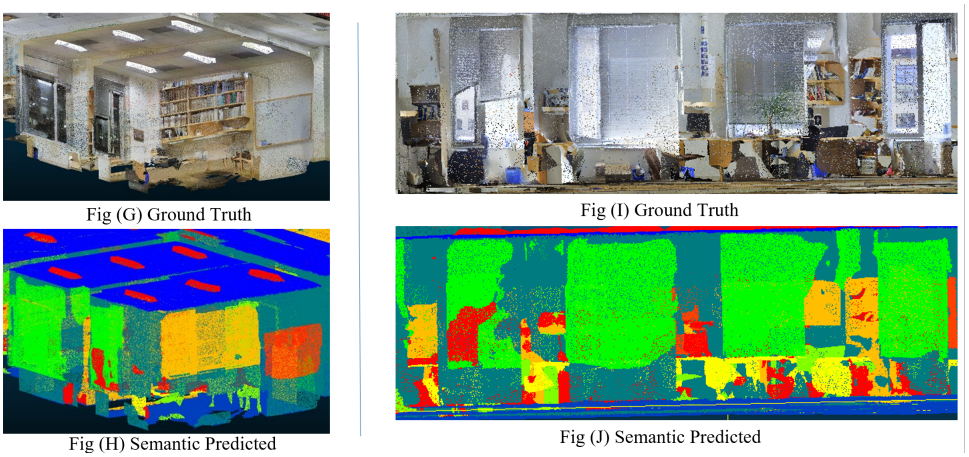

Fig (G) Ground Truth

Fig (H) Semantic Predicted

Fig (I) Ground Truth

Fig (J) Semantic Predicted

Figure 12: Visualization of segmented results at 350 Epochs v/s Ground Truth - Room Interiors.

7 Conclusion and Future Scope

Semantic segmentation of 3D pointclouds was successfully implemented with the KPConv algorithm. The highest mIoU score achieved was 66% with highest IoU score achieved in ceiling, floor, wall, table and chair. Some objects like beam and column were not able to achieve good results and had impact on the overall average score. One further step i.e., Instance segmentation can be done based on the semantic segmentation results. Instance segmentation can help us identify the number of objects in each class. Also, panoptic segmentation models can

be applied to do combined segmentation tasks. More complex algorithms like graphical neural networks approach can be used for more robust and accurate models.

Acknowledgment

I thank Professor Dr. Corey Toler-Franklin for guiding us through the course. I also acknowledge and thanks UF hipergator for high performance GPU access.

References

[1] http://www.semantic-kitti.org/index.html

[2] http://www.open3d.org/docs/latest/python_api/open3d.ml.tf.datasets.S3DIS.html

[3] Thomas, Hugues, et al. "Kpconv: Flexible and deformable convolution for point clouds." Proceedings of the IEEE/CVF international conference on computer vision. 2019.

[4] Sirohi, Kshitij, et al. "Efficientlps: Efficient lidar panoptic segmentation." IEEE Transactions on Robotics (2021).

[5] Torch Point 3D: https://torch-points3d.readthedocs.io/en/latest/

[6] Hu, Qingyong, et al. "Randla-net: Efficient semantic segmentation of large-scale point clouds." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020.

[7] Li, Yangyan, et al. "Pointcnn: Convolution on x-transformed points." Advances in neural information processing systems 31 (2018).

[8] Graham, Benjamin, Martin Engelcke, and Laurens Van Der Maaten. "3d semantic segmentation with submanifold sparse convolutional networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

[9] Vu, Thang, et al. "SoftGroup for 3D Instance Segmentation on Point Clouds." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022.

[10] Grilli, Eleonora, Fabio Menna, and Fabio Remondino. "A review of point clouds segmentation and classification algorithms." The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 42 (2017): 339.