3D SEMANTIC SEGMENTATION FOR SCENE PARSING

Lingyu Zhang and Mingyang Zheng
{lz2494, mz2594}@columbia.edu
Columbia University

ABSTRACT

3D scanners have been largely used in remote sensing and robot interaction for 3D environment reconstruction, modelling and scene parsing. However, the large scale scanned data lacks of structure and semantic interpretation. Without further processing, the information directly gathered from raw point cloud data can be used for maximum distance detection which is far from enough for automatic car driving system or interactive robot sensing. The key procedure for the 3D semantic segmentation task is to assign a label to each single point in the point cloud. Unlike the 2D image which can be represented by pixel to pixel matrix using R, G, B value, 3D point cloud data does not have regular format, unable to utilize CNN for semantic segmentation directly. In this paper, we present a new approach for the 3D semantic segmentation based on feature abstract and deep learning method. Firstly, we estimated the shape outline through SIFT3D keypoint extraction and normal estimation, extracted the local feature of the point cloud and then transform the 3D image into decoupled vectors of each point. Then we build neural network model for point-wise semantic label classification. Finally, we re-projected the vector with predicted label to 3D point cloud for color visualization. Our proposed model has simple architecture, small model size, short training time with decent accuracy.

Index Terms— semantic, classification, 3D point cloud

1. INTRODUCTION

3D scene parsing is a crucial task in computer vision, especially in robot sensing for artificial intelligence and driverless car system. However, point cloud, the basic geometric data structure of 3D images, which characterizes the contours of objects, is an irregular format compared to traditional 2D images. Moreover, due to the input permutation invariance, it is difficult to feed them directly into CNN with a suitable perception field. Basically, most researchers use three different methods to solve this problem. The first one [1] is to project the 3D point cloud data into multi-view 2D images, and apply very deep neural network such as VGG16 to do the pixel-wise classification. Then re-project the 2D classification result to original 3D data. The second one [2] is to transfer point cloud to 3D voxel grids with regular format, then build CNN for this 3D regular format. The third one is to treat the z coordinate as the fourth channel besides R, G, B channels and build 2D CNN. These methods, however, either increase the input data size unnecessarily with huge redundancy, or increase the complexity of neural networks, or cause other issues.

In this paper, we provides a new approach to do semantic segmentation for 3D point cloud. The key idea is to utilize the 3D local descriptor, convert shape outline information of the point cloud data to decoupled vectors and then apply the deep learning neural network to the vector data for point-wise classification. The whole procedure can be roughly divided into four steps including keypoint estimation, 3D feature extraction, neural network design and labelled point cloud reconstruction.

To prepare the classification dataset, before 3D feature extraction, we firstly estimate the keypoint of the raw point cloud data using SIFT3D. For the training set, we use Semantic3D dataset, which belongs to the large scale point cloud classification benchmark. The dataset contains fifteen different scenes, the size of each scene is approximately 1 GB. For the testing dataset, we use both the testing set from Semantic3D dataset and the raw point cloud data acquired from laser scanning.

Then, we estimated the shape outline information through local feature descriptor. In our paper, we use the fast point feature histogram (FPFH). The point feature histograms are calculated by generalizing the mean curvature around the point using a multi-dimensional histogram of values. Our calculation is based on the relationships between the points in the k-neighborhood and their estimated surface normals. For fast point feature histograms, we simplify it by decorrelating the values.

The neural network architecture is trivial compared with other complex CNN or RNN used widely in computer vision. We only use a few full- connected layers to realize point-wise classification. Several complex CNN architectures have been tried as well. Interestingly, we find that CNN increases the complex of the network but decrease the accuracy. A brief analysis of this phenomenon is given in this paper.

With all the predicted labels of the points, we do color coding of the label, for each different label, we assign different color to it. We combine the coordinates information and the label to reconstruct the point cloud and use meshlab to do the data visualization.

Our approach is lightweight with simple architecture, small model size, short training time and decent accuracy, and can be applied to real time scene parsing.

2. RELATED WORKS

The approaches for 3D semantic segmentation differ according to the data format used. Voxnet feeds 3D voxel grid data directly into convolutional neural network[1]. However, this method is constrained by the high computing cost of 3D convolution and data efficiency due to the fact that most voxel in space do not hold meaningful objects. Wang et al. propose a voting scheme utilizing sliding window, sparse convolution and voting to exploit the sparse nature of the space[2]. However, their efficiency decrease when facing large point clouds. Su et al. propose that 3D shapes can be recognized from a collection of their rendered views on 2D images, hence transform 3D point cloud into multi-view images and use method of 2D convolutional network for object recognition[3]. However, representation power of 2D images limits in the accuracy of recognition. Qi et al. proposed a PointNet architecture, where the output of a symmetric function is invariant to the input order, which is efficient to unordered point cloud[4]. However, the complex of network lead to large model size and long training process.

3. METHOD

Overview. The goal of our work is to provides an automated approach to do semantic segmentation for 3D point cloud. In this section, we begin by outlining the properties of point cloud and our design strategies for network architectures. Then, we introduce our model, which utilizes point-wise classification to for semantic segmentation of 3D image.

3.1 Characteristics of Point Cloud

A point cloud is represented as a set of 3D points, where each point is a vector of its (x,y,z) coordinate plus extra feature channels such as color, normal etc[4]. As point cloud are usually generated by laser scanning, points in a point cloud only exist at the contour of objects, so point cloud is an irregular format-- most part of the 3D space does not have points except contour of objects, which is quite different from 2D image. The second characteristic of point cloud is the points in a point cloud are unordered, which means the permutation of points in a point cloud has no influence on the 3D image. This characteristic increases the difficulty for network architecture design, for the input of neural network are required to be ordered array of matrix, of which the permutation usually has influence on the output. Though unordered, points in a point cloud are not standalone. On the contrary, the neighboring points are cross related and form meaningful structure without relying on RGB channels.

3.2 Data Process and Network Architectural Strategies

Due to the uniqueness of point cloud, CNN which has been widely used in 2D computer vision and image processing cannot be directly used for 3D point cloud. Firstly, convolutional filter, which is used to aggregate information from local structures in CNN, must has a regular shape of perception field, making it difficult to deal with irregular points in point cloud. Secondly, as mentioned above, the unordered points requires the network to have input permutation invariance. For example, if there are N points in a point cloud, N! permutations of input should have same output result for the label of each point, which cannot be achieved by most CNN. Thirdly, CNN usually relies on the RGB or gray value to retrieve local structure information. However, in 3D point cloud, besides RGB or gray value, abundant local structure information can also be provided by curvature, normal and relative positional relationship among points, which cannot directly learned by CNN. For these three reasons, a new data processing and network architecture should be used for data of point cloud.

Considering the irregular format of point cloud and unordered points, it is unrealistic to feed the whole 3D image into neural network. As our aim is to point-wise classification, one intuitive way is to feed only one point into the neural network every time and output the label of each point. The input data of each point should contains not only the information of it position but also information of the relative positional and geometric relationship among its neighbors. We call this process 3D feature abstracted dimension transform. After this data processing, The 3D point cloud has been transformed into a group of vectors. Each vector represents one point and keeps the 3D position and geometric feature in it neighborhood. The function of this process is similar to local feature aggregation by convolutional filter and pooling. As the feature of each point no longer need information from other vector, we can directly feed this vector into full connected neural network to get its label. Finally, we reproject the label of each vector into 3D point cloud to get the result of semantic segmentation.

3.3 Model Architecture

Our work can be roughly divided into four main steps: (1) keypoint estimation; (2) dimension transform with 3D feature abstracting; (3) point-wise classification based on neural network; (4) label reprojection for semantic segmentation. In section 3.3.1, we describe the selection for keypoint detector and discuss the performance of different estimation method on our dataset. In section 3.3.2, we discuss different local feature extraction method and the parameters optimization procedure. In section 3.3.3, we describe the neural network structure and the classification process. In section 3.3.4, we show the process to reproject label of each vector into 3D point cloud for visualization.

3.3.1.Keypoint Estimation

Based on the raw point cloud directly acquired from the laser scanning, we need to reject some of the points which are poorly localized along the edge or do not have enough contrast and thus sensitive to noise. Additionally, through the keypoint extraction procedure, we can reduce the size of the whole dataset without decreasing prediction accuracy.

Keypoint descriptor selection. Since our data is acquired from real-world scanned point cloud, we mainly focus on repeatability and transformation invariance. To evaluate the repeatability and transformation invariance, we use the following term to compare different methods. Since the model can be transformed to a scene through rotation, scaling and translation, we use the keypoint correspondence between the model and the scene to evaluate the performance of keypoint extraction method. Figure 1 shows the keypoint matching result between the model and the scene.

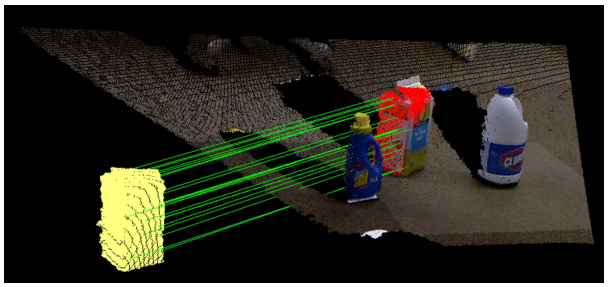

Figure 1 Keypoint matching between model and scene

According to the evaluation survey [5], we use the distance of the keypoint with its nearest neighbor in the keypoints set extracted from the scene to represent the extent of keypoint repeatability. If the distance is small enough, we then say that this keypoint is repeatable. For repeatability, the absolute repeatability

$$r_{abs} = |RK_{hl}|$$

The relative repeatability

$$\mathrm{r} = \frac{|RK_{hl}|}{|K_{hl}|}$$

with RK is the set which contains all the repeatable keypoints extracted from both model and the scene, and K is the set containing all the keypoints extracted from the model and without any occlusion in the scene.

Based on the research result of [5], in comparison with 3D keypoint descriptor including Harris3D, Curvature3D, KLT, the Scale Invariant Feature Transform (SIFT3D) keypoint descriptor achieves best relative repeatability for scale change. And it achieves higher absolute repeatability than Harris3D, Curvature3D and KLT when apply rotation to the model. Additionally, SIFT3D gives us best results when considering the invariance of transformation.

Based on the comparison above, we finally choose SIFT3D as the keypoint descriptor method.

Scale Invariant Feature Transform. SIFT is the local keypoint detection method which is based on scale and orientation descriptor. The key property of SIFT is scale-invariant which means that if the scale of corresponding scene differs from the model, algorithm performance would not be impacted. For the whole pipeline of SIFT algorithm, it can be roughly divided into 5 steps:

- Form the scale space based on convolution kernel.
- Detect extreme point within the scale space based on the comparison with its neighbors.
- Discard noise points.
- Assign orientation coefficients.
- Extract key points.

After this part, we now get the keypoints set of the scanned 3D scene data which can best describe the shape outline information of the original data with relatively small data size.

3.3.2. Dimension Transform with 3D Feature extraction

The key part of our approach is to get the feature map of input data before expensive neural network classification procedure. The goal of this part is to convert detailed coordinates data to vector feature representation without losing much information. In order to get proper representation for the shape outline, we need to choose the proper local feature descriptor to represent the relationship of the points in different regions.

Feature descriptor selection. Since the size of the feature data has the effect on the whole computational efficiency, and we are working with different stationary objects which requires our prediction model to be stable enough, we mainly focus on the compactness, robustness and efficiency in our approach to evaluate the performance of a feature descriptor.

Based on the research result in [6], the FPFH (Fast Point Feature Histograms) feature has the best compactness performance which is evaluated by the average value of AUC_{pr} per float. Additionally, FPFH turns out to be stable with regard to the varying mesh resolution and key point localization error. Also FPFH has the best efficiency when we work with large amount of data points. In general, FPFH is a good choice for our task to evaluate the local feature of our real world scanned scene data points.

Fast Point Feature Histograms. According to point cloud library, the point feature histograms (PFH) are calculated through evaluating the mean curvature around one selected point. This multidimensional histogram of values serves as a representation of the relationship between the points in k-neighborhood. The computation is based on the surface normal of each 3D point. The procedure can be divided into 3 steps:

- Use Kd tree nearest neighbor search to get the nearest neighbors based on surface normals of each point.
- Generate the curvature values for each pair of neighbors.
- Combine the angular values to 125-dimensional histograms.

To accelerate the computation procedure, there is fast point feature histograms (FPFH) in which we add one more weighting scheme step based on the original PFH algorithm. With the result of PFH, we then re-determine the K neighborhoods of the point based on the distance of neighbors with the point. Based on this procedure, the computational complexity is significantly reduced which makes it possible for us to use FPFH in a real-time system. The procedure can be divided into 3 steps:

- Get the result of original PFH algorithm.
- Calculate the distance of neighbors and the selected point and do weighting based on the distance.
- Get the simplified 33-dimensional histogram results of FPFH.

For the different scenes objects, we build the batch processing pipeline to load all the files in ply format into point cloud format. For each point of one object, we extract surface normals and based on the normal estimation, we calculate the 33 dimensional FPFH feature values of which each value is represented by a 4-byte float. For example, two points of a car which have similar FPFH features are visualized as below:

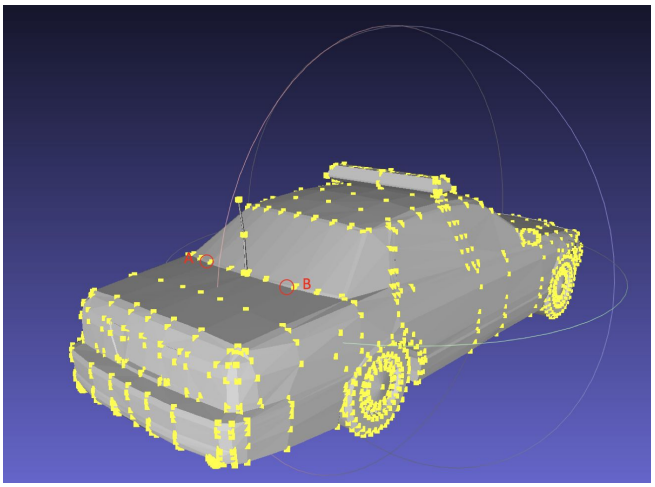

Figure 2(a) Two points with similar FPFH feature

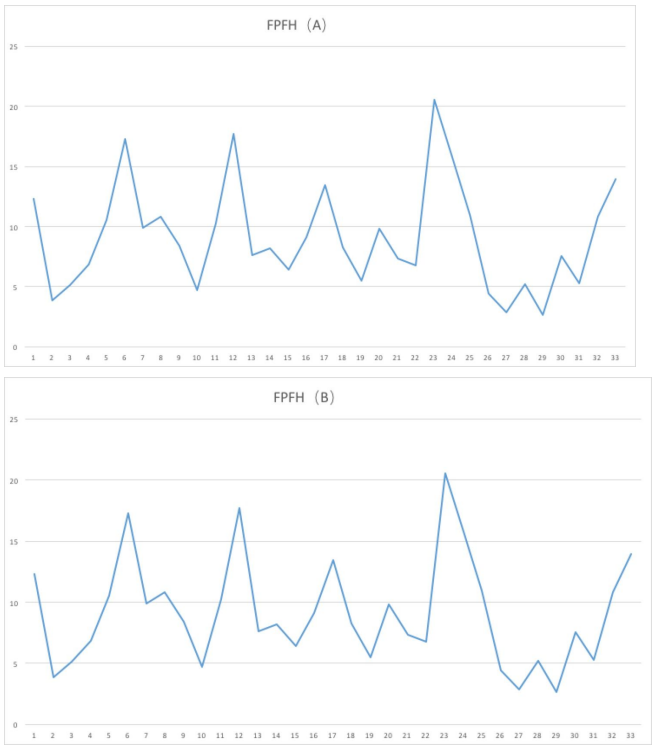

Figure 2(b) Two points with similar FPFH feature

Dynamic scale estimation for FPFH. Since FPFH is a local feature descriptor which requires us to set a searching radius for curvature estimation. For a selected point which belongs to a specific class shape, we need to distinguish it from one similar point in other class shape by recognizing the difference of the neighbors around the selected points.
For example, in the Figure2. showed above, point A and point C belong to different classes (building and car).
If we only focus on the curvature values of a small searching radius of surrounding neighbors which only includes the points on the same plane, the local feature of A or C may be calcu lated to be a series of same curvature values which indicate A and C belong to similar planes and similar shapes. In order to avoid the misclassification, we need not only do calculation of neighbors in the small region, but also include the neighbors which lie farther away to provide sufficient information of the whole shape which the selected point belongs to. With the local feature extracted with different scales, we can fully describe the identity of different points which belong to different class shapes.

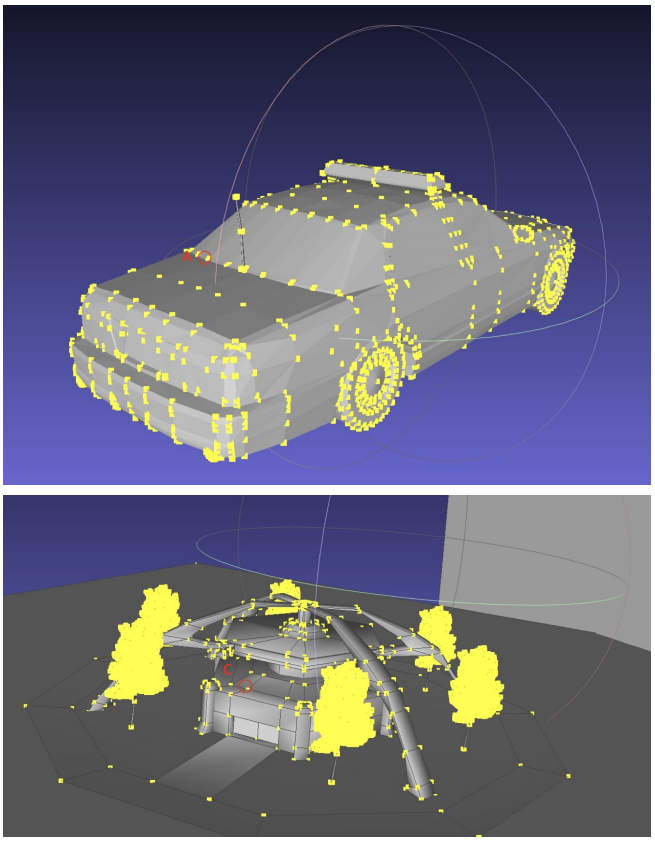

Figure3 Different shape points for same searching radius

In our procedure, in order to acquire proper feature estimation scales for points of a given objects, we firstly calculate the maximum distance of random selected pair of points R of the whole point cloud. To fully describe the detailed information of the surface, we set a fairly small searching radius r= (1/30)R to extract surface normals. To acquire the curvature information in proper scale,, we extract the local feature based on the searching radius R1 = 4r. In this way, the feature of each 3D point is described by one set of histogram values of different scales saved in point cloud files. These files would be used in the next part to be combined into 33-dimensional feature data for each point. And for further optimization, we can combine multiple searching scale for more detailed feature information.

3.3.3. Point-wise Classification based on Neural Network

Neural network architecture seems to be trivial compared to other network in computer vision or image processing. Since the local structure information aggregation, the most nction of convolutional filter, has already been achieved by our first two steps, we use a full connected neural network for classification. The architecture of our proposed neural network is shown in figure 4 below, which outputs kd scores for all the ke candidate classes.

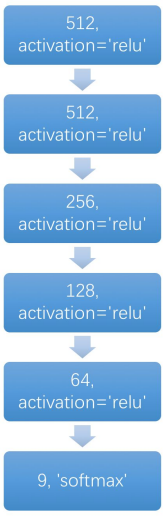

Figure4 Neural network architecture

3.3.4.Label Reprojection for Semantic Segmentation

With each single point already has a predicted class label, we then reform the labeled point cloud data through concatenating the point cloud coordinates data with color coded labels. For label color coding, we use the following coding methods.Then we use meshlab to visualize the predicted point cloud dataset results.

4. EXPERIMENT

4.1. Dataset

Our training data and part of the testing data come from Semantic3D which contains 16 different scenes such as urban scene in Figure 5, each of which is approximately 1GB. The data format in Semantic3D is XYZIRGB which contains x,y,z coordinates, intensity and r,g,b values.

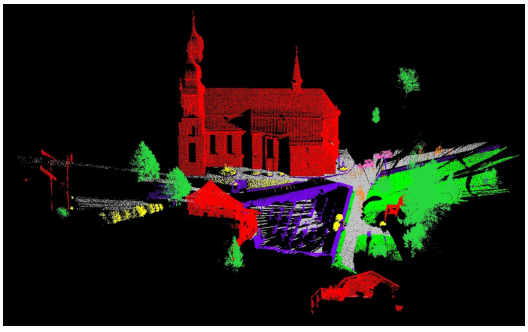

Figure 5 Urban scene in Semantic3D dataset

Our part of the testing data comes from ISPRS WG III/4 which contains urban 3D scene data with x,y,z coordinates, and from laser scanned data which is laz format. For the data of laz format, we use lastools to decompress it and further convert it to ply format which can be load to use processing functions in Point Cloud Library.

The whole training set contains 9 classes and we use different numbers to represent the class labels.

Table 1. Point cloud classes and labels

Class	man-made terrain	natural terrain	high vegetation
Labels	1	2	3
Color	[128, 0, 128]	[0, 255, 255]	[0, 255, 0]
Class	low vegetation	buildings	hardscape
Labels	4	5	6
Color	[0, 128, 0]	[255, 0, 0]	[255,128,0]
Class	scanning artefacts	cars	other
Labels	7	8	0
Color	[255,255,0]	[0, 0, 0]	[153,153,153]

Handling unbalanced multiclass dataset

As the data in the each single scene is unbalanced, with the data size of class "other" is almost 10 times of the class "buildings". We handling the unbalanced dataset through subsampling each class to make sure the number of samples of each class is nearly the same. Then we modify the sequence in-place by shuffling its contents to get random samples.

4.2. Implementation Details

4.2.1 point feature abstraction

For the normal estimation of each single point in the raw point cloud, we use searching radius r= (1/30)R, for FPFH feature estimation, we use R1 = 4r. For further processing, we combine multiple searching radius for local feature estimation which will give us more detailed feature information that properly describes the shape outline of the scene.

4.2.2 Neural Network

The neural network is built using Keras. All the layers are full connected except the fourth and fifth layer with a dropout of 0.1 to avoid overfit. We use ReLu(Rectified Linear Units) as activation function except output layer. The training process takes 20 epochs with batch size 128. We use RMSprop as optimizer with learning rate 0.001.

4.3. Result Evaluation

The training data has 120000 points, sampled from a total of 2 *10e8 points in the training scans. The test set has more than 60000 points. Figure 4 shows that our proposed approach is able to cope rather well with the test data.

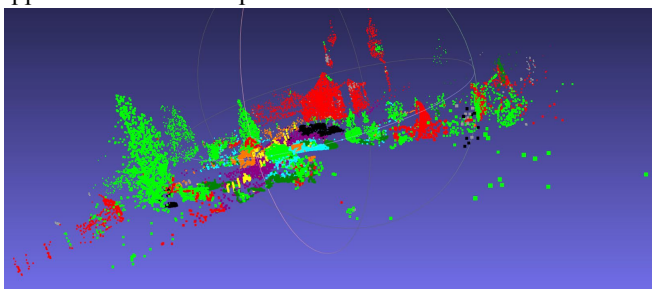

Figure 6 Visualization for test result

Overall accuracy achieved 75.7%. Table 2 compares our model size and that from TML-PC. Our model size is 60 times smaller than TML-PC models with a similar accuracy.

Table 2. Approach Evaluation

Model	overall accuracy	Model Size
our method	75.7%	3.4 MB
TML-PC	74.5%	226 MB

5. DISCUSSION

In this section we will explore the design space of our method. We propose several aspect for our accuracy improvement as our future work.

5.1 Feature Extraction

Up to now our 3D feature extraction is limited to the (x, y, z) coordinate of each point, however, the intensity and GRB value of each point contains more information for semantic segmentation. In future work, we will add these values into the feature vector generation.

Besides, the dimension of vectors may have influence on the overall accuracy. In this paper, we use FPFH which produces 33-dimensional histogram values of different scales for each point. However, FPFH simplifies the histogram feature computation. The FPFH does not fully interconnect all neighbors of point, and is thus missing some value pairs which might contribute to capture the geometry around the query point. The resultant histogram is simplified by decorrelating the values, which is simply creating d separate feature histograms, one for each feature dimension, and concatenate them together. Differently, the PFH models a precisely determined surface around the query point. We can replace FPFH with PFH for better performance.

5.2 Dataset Parameter

One reason that our approach cannot beat the state of art model is that our training set size is quite small. Unfortunately, for the 3D feature abstraction step, there is not GPU support in point cloud library for the specific function, thus it takes long time for data processing before feed them into neural network. Another factor is sample rate. Obviously, with same radius, the more point within that radius, the better description of the relative positional relationship will be. However,the time complexity to calculate FPFH of all the points in a point cloud is O(N^2), which is unacceptable for large point cloud calculation. So this paper does not represent the best performance of our approach. There is still much room for accuracy growth.

6. CONCLUSION

In this paper, we propose a new approach to semantic segmentation for 3D point cloud. Different from the traditional CNN architecture, we do 3D feature abstraction of each point before send them into neural network. And the input of neural network is a vector of each point, instead of the whole 3D image. As 3D feature abstraction has aggregated enough local structure information, achieving the same function of complex CNN architecture, our neural network can reach relative high accuracy with quite trivial structure. Finally, semantic segmentation is realized by combining the coordinates information and the label of each point to reconstruct the point cloud. Our lightweight architecture has advantages of small model size, shorting training time and decent accuracy and can be applied to real time scene parsing such as auto-driving and interactive robot sensing.

7. REFERENCES

[1] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. Guibas. Volumetric and multi-view cnns for object classification on 3d data. In Proc. Computer Vision and Pattern Recognition (CVPR), IEEE, 2016

[2] Z. Wu, R. Shou, Y. Wang, and X. Liu. Interactive shape cosegmentation via label propagation. Computers & Graphics, 38:248–254, 2014

[3] J. Sun, M. Ovsjanikov, and L. Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In Computer graphics forum, volume 28, pages 1383–1392. Wiley Online Library, 2009

[4] C. Qi, H. Su, K. Mo and L. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. CVPR 2017

[5]Filipe, Silvio, and Luís A. Alexandre. "A comparative evaluation of 3D keypoint detectors in a RGB-D object dataset." Computer Vision Theory and Applications (VISAPP), 2014 International Conference on. Vol. 1. IEEE, 2014.

[6]Alexandre, Luís A. "3D descriptors for object and category recognition: a comparative evaluation." Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vilamoura, Portugal. Vol. 1. No. 3. 2012.