

# Road Edge Detection on 3D Point Cloud Data using Encoder-Decoder Convolutional Network

Reza Fuad Rachmadi<sup>†\*</sup>, Keiichi Uchimura<sup>†</sup>, Gou Koutaki<sup>†</sup>, and Kohichi Ogata<sup>†</sup>

<sup>†</sup>Graduate School of Science and Technology (GSST), Kumamoto University, Kumamoto, Japan

<sup>\*</sup>Dept. of Computer Engineering, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia

fuad@navi.cs.kumamoto-u.ac.jp, {uchimura,koutaki,ogata}@cs.kumamoto-u.ac.jp

**Abstract**—The demand of High Definition Maps (HD-Maps) has been increasing, especially for autonomous vehicle application. Usually, HD-Map is created by scanning the road using LiDAR sensor and reconstructing the road on 3D world to capture all aspects of road properties. One of the important properties of a road is its edge or boundary. In this paper, we propose end-to-end 3D Encoder-Decoder Convolutional Network (3D-EDCN) for road edge detection on 3D point cloud data produced by LiDAR sensor. Our 3D-EDCN classifier consists of nine convolutional layers and three deconvolutional layers. For simplification, we use 3D voxel format as input and output of the classifier. Our proposed method was tested using our own 3D point cloud dataset which taken from LiDAR equipment and consisting of 103 3D point cloud data with their respective road edge ground truth. In the training process, we use combinations of Cross-Entropy loss function and Euclidean loss function to help our model converged. As a preliminary result, our proposed 3D-EDCN classifier achieves Mean Square Error (MSE) of  $2.738 \times 10^{-5}$ , precision of 0.37262, and recall of 0.14432.

## I. INTRODUCTION

Intelligent Transportation System (ITS) has been widely adopted for monitoring, managing, and navigating traffic in big cities around the world. One of the most sophisticated ITS technology that will be implemented in the near future is autonomous cars. The implementation of autonomous car has increasing the demand of High Definition Map (HD-Maps) for car navigation. Massow et al. [5] described infrastructure requirements for HD-maps management system that can processed huge amount of probe data from on-board sensor in connected vehicle. Seif and Hu [6] investigate the main key parts of the autonomous car, including infrastructure, autonomous car, and HD-maps. High precision LiDAR sensor is widely used for generating HD maps. There are three steps to generate HD maps using LiDAR sensor; scanning the road using LiDAR sensor, generating a high precision 3D version of the road, and detecting the properties of the road. Unfortunately, the system that can automatically detect the properties of the road from LiDAR data does not yet exist. There are a lot of road properties that needs to be extracted from LiDAR data, but one of the most important properties of a road is the road edge which is very useful for autonomous car navigation systems. The raw data produced from LiDAR sensor are formatted using 3D point cloud data. Figure 1 shows a visualization of a road on 3D point cloud data taken from LiDAR sensor. Convolutional Neural Network (CNN) has been

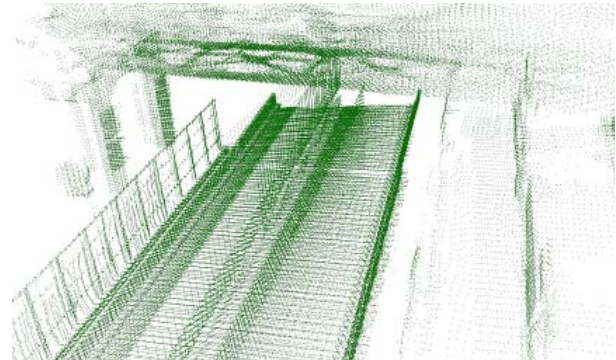


Fig. 1: The visualization of road on 3D point cloud data taken from LiDAR equipment.

proved very good for a lot of image-based application [2], [3], [8], [9], [13], [17] including object detection and image classification on 3D point cloud data [7], [10], [11]. There are only a few research to solve a problem with LiDAR data including that described in [1], [10].

In this paper, we propose a 3D Encoder-Decoder Convolutional Network (3D-EDCN) classifier for road edge detection problem on 3D point cloud produced by LiDAR sensor. To maintain the resolution of road edge prediction, we design the output of our classifier to have the same resolution as the input image. Our contributions can be listed as follows.

- We investigated 3D Encoder-Decode Convolutional Network architecture for road edge detection on 3D point cloud data produced by LiDAR sensor.
- We investigated the use of voxel format for road edge detection problem on 3D point cloud. Although we didn't evaluate the actual accuracy (in cm or mm) but our work shows that the voxel data format can be used as a primary format of the input and output data for the problem.
- We proved that the combination of Cross-Entropy loss function and Euclidean loss function can help our network to converge with the problem.

The rest of the paper is organized as follows. Section 2 discusses the related work that had been carried out by other researchers. Section 3 briefly describes about CNN and Convolutional Autoencoders which is the based architecture of our proposed classifier. We describes the detail of our 3D-EDCN classifier in section 4 and discuss the results of our

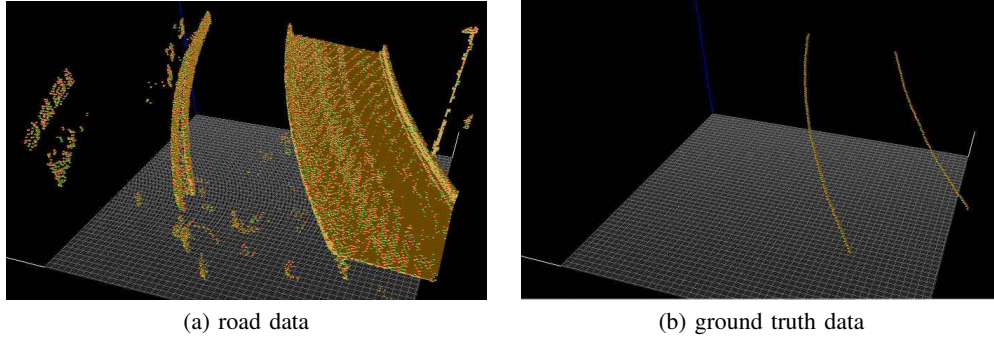


Fig. 2: An example of 3D road edge dataset with their respective ground truth on  $256 \times 256 \times 256$  voxel format.

experiments in section 5. Lastly, we summarize our results in section 6.

## II. RELATED WORK

In the era of modern machine learning, convolutional neural network (CNN) classifier has been widely used for a lot of applications including image and video classification. The CNN classifier becomes very popular after Krizhevsky et al. [17] won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 with significant accuracy improvement comparing with other methods. Recently, a lot of research has used a deep learning method to solve computer vision tasks including 3D vision problems. Some papers that tackle the 3D vision problems using deep learning classifier were described in [1], [7], [10]–[12].

Caltagirone et al. [1] proposed a fully CNN classifier for road edge detection based on KITTI road edge detection dataset [16]. To perform end-to-end prediction, they use hourglass type of CNN architecture with context module in the middle of the network. The context module consists of seven dilated convolution operations followed by convolutional layer with  $1 \times 1$  kernel size. To fix the input and output resolution, they converted the 3D point cloud to stacked 6 top view image with six different statistical values of point cloud. To measure the performance of their method, the KITTI evaluation method was used including MaxF (maximum F1 score), precision, recall, false positive rate, false negative rate, and average precision. As a result, their proposed method achieves all evaluation metrics more than 92% and the execution time is 18 ms. The performance of their proposed method is very high on KITTI road detection dataset, but the dataset only covers urban road, the crowded road, like highway with a lot of level or road with different orientation is not covered.

Girdhar et al. [7] describe a combination of 3D-CNN classifier and 2D-CNN classifier to tackle a 3D layout prediction from a single image input. The voxel format representation was used for the 3D geometry data. They use TL-embedding network, which is a combination of 3D Convolutional Autoencoder (3D-CAE) and 2D-CNN classifier. To connect the 2D-CNN classifier to a 3D-CNN classifier in the training process, they use Euclidean loss function that minimizes the fully-connected layer in the middle of the 3D-CAE and the last

fully-connected layer in the 2D-CNN classifier. They called the network as T-network because the shape of the network was a T-shape like network. At the end of the 3D-CAE classifier, the cross-entropy loss function was used in the training process. During the testing process, the convolution layers in 3D-CAE were not used and the last fully-connected layer of 2D-CNN classifier was attached to deconvolution parts of the 3D-CAE. As a result, the classifier achieves average precision (AP) of 97.6% and outperforms the PCA method.

Maturana et al. [10], [11] use 3D-CNN classifier to solve the 3D object recognition problem [11] and landing zone detection problem on LiDAR data [10]. They proposed the network called VoxNet which is a 3D-CNN classifier with voxel input as a format. The VoxNet was designed to run a real-time scenario and to achieve the goal, the VoxNet must have a shallow network design. The VoxNet consists only four-layer network with two convolutional layers and two fully-connected layers. As a result, the VoxNet for 3D object recognition problem achieves an average accuracy of 83% on ModelNet40 dataset [12], 71% on NYUv2 dataset, and average F1-score of 0.73 on Sydney dataset. The time execution of VoxNet for 3D object recognition problem was around 6 ms using NVIDIA K40 GPU hardware.

In this paper, we investigate a 3D convolutional network to solve road edge detection problem on 3D point cloud data produced by LiDAR sensor. Unlike Caltagirone et al. [1], we processed the data in 3D coordinate and treated the point cloud data as voxel data. Voxel-based CNN classifier is also proposed by Maturana et al. [10] for landing zone detection on LiDAR data, but the base architecture of the CNN is not an hourglass CNN.

## III. CONVOLUTIONAL AUTOENCODERS

### A. Autoencoders

Autoencoders is a neural network architecture that maps input to some latent representation (encode) and then reconstructs the input using the representation (decode) [19]. Let  $\mathbf{x} \in \mathbb{R}^m$  is an input of autoencoders, a single layer encoding operation of the input data can be described as  $\mathbf{h}_\theta(\mathbf{x}) = \sigma(\mathbf{W}^T \mathbf{x} + b)$  with  $\mathbf{h}_\theta \in \mathbb{R}^l$  and  $l < m$ . The latent representation then used to reconstructing the input  $\mathbf{y}$  using  $\mathbf{y} = \sigma(\mathbf{W}_i \mathbf{h}_\theta + b_i)$  and  $\mathbf{W}_i$  is constrained to  $\mathbf{W}_i = \mathbf{W}^T$ . The

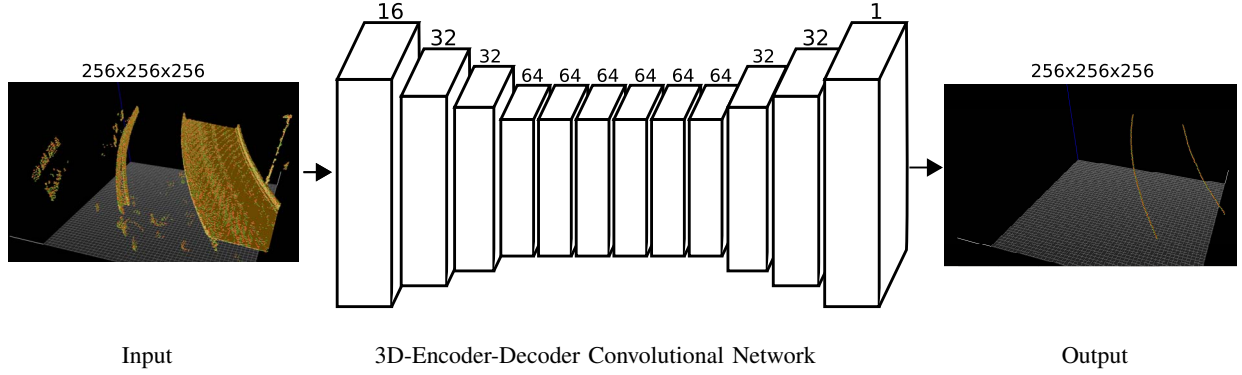


Fig. 3: Our 12 layer of 3D Encoder-Decoder Convolutional Network (3D-EDCN) classifier for road edge detection problem on 3D point cloud data.

encoding and decoding operation usually consists of more than one layer and has the same number of layers. The network parameters are optimized by minimizing a Euclidean loss function over input  $\mathbf{x}$  and output  $\mathbf{y}$ .

#### B. Convolutional Autoencoders

The convolutional autoencoders has a similar architecture as original autoencoders except that the convolution calculation is performed in each layer instead of full matrix-vector multiplication. Convolutional auto-encoders was introduced in [18] as a feature extraction network. Masci et al. [18] use flipped version of encoding weights in all dimension as constraint for decoding weights ( $\mathbf{W}_i = \tilde{\mathbf{W}}$ ). The parameters are optimized by minimizing Euclidean loss function between input  $\mathbf{x}$  and output  $\mathbf{y}$  described as follows

$$E(\theta) = \frac{1}{2n} \sum_{l=1}^n (x_l - y_l)^2 \quad (1)$$

where  $\theta$  is the weight of the network.

#### IV. 3D ENCODED-DECODER CNN

As described in sections 2 and 3, Convolutional autoencoders proved to be very good as features extractions [18], end-to-end 3D object prediction from 2D image [7], and painter classification problem [4]. To solve road edge detection problem on 3D point cloud data, we propose end-to-end encoder-decoder CNN classifier (3D-EDCN). We use 3D voxel format as input and output of the classifier with resolution of  $256^3$ . Figure 2 shows an example of the input and desirable output of road edge detection using 3D voxel format. Our 3D-EDCN classifier consists of 12 layers with 9 convolutional layers and 3 deconvolutional layers. Unlike Girdhar et al. [7] approach, we change the middle part of the network from fully-connected layer to convolutional layer to reduce the memory requirement for the model. The detailed configuration of each layer in our 3D-EDCN classifier is shown on TABLE I. Figure 3 shows the diagram of our 3D-EDCN classifier.

Each layer except the last layer was activated using non-linear ReLU activation function. The ReLU function can be described as follows

$$A^k(x) = \max(0, \mathbf{W}^k \mathbf{x} + b) \quad (2)$$

where  $A^k$  is the output feature map in layer  $k$ ,  $\mathbf{W}^k \mathbf{x} + b$  is an output of layer  $k$  with weights and bias of  $(\mathbf{W}, b)$  and input  $\mathbf{x}$ . The sigmoid activation function was used in the last layer of our 3D-EDCN classifier to ensure that the output of each voxel was ranged from 0 – 1.

In the training process, we use Cross-Entropy and Euclidean loss function alternately to reduce the training loss value with Adam solver [15] as training algorithm. The training configuration of our 3D-EDCN classifier can be viewed in TABLE II. The Cross-Entropy loss function is described as follows

$$E = -\frac{1}{N} \sum_{n=1}^N [p_n \log(\hat{p}_n) + (1 - p_n) \log(1 - \hat{p}_n)] \quad (3)$$

where  $p_n$  is the target label (or occupied voxel) and  $\hat{p}_n$  is the predicted voxel value with 0 – 1 range. Mainly, the Cross-Entropy loss function has been widely used for multi-label classification problem but in our problem, the combination of Cross-Entropy and Euclidean loss function can achieve lower loss value in the training process compared with the same model using only Cross-Entropy loss function in the training process.

#### V. RESULTS

We use Caffe framework [14] for the experiments with GPU NVIDIA GTX 1080 as processing power. The MSE, precision, and recall analysis are used to evaluate the classifier.

##### A. 3D Road Edge Dataset

To evaluate our 3D-EDCN classifier, we use our own road edge dataset consisting of 103 point cloud data with their respective ground truth. The 3D point cloud data were taken from LiDAR sensor attached at the top of the car and we carefully generated the road edge ground truth using the data manually. The data were converted to 3D voxel format based

TABLE I: Layer configuration of our 3D-EDCN classifier.

Layer	Input	Kernel Size	Output
input	-	-	256x256x256@1
conv1	256x256x256@1	1x3x3x3@16s2	128x128x128@16
conv2	128x128x128@16	16x3x3x3@32s2	64x64x64@32
conv3	64x64x64@32	32x3x3x3@32s2	32x32x32@32
mconv4	32x32x32@32	32x3x3x3@64s1	32x32x32@64
mconv5	32x32x32@64	64x3x3x3@64s1	32x32x32@64
mconv6	32x32x32@64	64x3x3x3@64s1	32x32x32@64
mconv7	32x32x32@64	64x3x3x3@64s1	32x32x32@64
mconv8	32x32x32@64	64x3x3x3@64s1	32x32x32@64
mconv9	32x32x32@64	64x3x3x3@64s1	32x32x32@64
deconv10	32x32x32@64	64x4x4x4@32s2	64x64x64@32
deconv11	64x64x64@32	32x4x4x4@32s2	128x128x128@32
deconv12	128x128x128@32	32x4x4x4@1s2	256x256x256@1
output	-	-	256x256x256@1

on the distribution of the point cloud in the 3D-coordinate. The voxel value was set to one when one or more point cloud lay inside the voxel. We use Algorithm 1 to convert the ground truth data to the voxel format. Algorithm 1 works as follows. The continuous 3D point lines of road edge ground truth  $P$  are filtered such that all generated lines are intersect with the bounding box  $BB$ . The filtered data are then converted to voxel coordinate based on the bounding box  $BB$ . To generate line on voxel coordinate, we use Bresenham Line generator with input of the filtered data on voxel coordinate  $V$ . The  $256 \times 256 \times 256$  voxel resolution was used for the data conversion process. We assume that the voxel resolution was enough to support approximately  $12 \text{ cm}^3$  volume representation.

---

**Algorithm 1:** Ground Truth Voxelization

---

**Data:**  $P \rightarrow$  Continuous 3D Line Point;  $BB \rightarrow$  Bounding Box of 3D point cloud

**Result:**  $VOX \rightarrow 256 \times 256 \times 256$  3D Voxel

**begin**

$V \leftarrow \emptyset$ ;  
 $VOX \leftarrow \text{ArrayZero}(256, 256, 256)$ ;  
**for**  $\{P1, P2\} \in \text{TwoPointSegmentLine}(P)$  **do**  
  **if**  $\text{LineIntersectionPointCloudBB}(P1, P2, BB)$  **then**  
     $V \leftarrow V + \{\text{ConvertToVoxelCoord}(x, y)\}$ ;

**for**  $\{P1, P2\} \in \text{TwoPointSegmentLine}(V)$  **do**  
   $\text{BresenhamLine}(P1, P2, VOX)$ ;

**return**  $VOX$

---

### B. Training Process

Three different random training/testing split configuration was used to evaluate our 3D-EDCN classifier with a proportion of 90 data for training and 13 data for testing process. To get

TABLE II: Loss function with their respective training parameters used for the training process of our 3D-EDCN classifier.

Iterations	Loss Function	Learning Rate ( $\alpha$ )
1-2,000	cross-entropy	0.0001 <sup>(*)</sup>
2,000-4,000	cross-entropy	$10^{-8}$
4,000-10,000	cross-entropy	0.0001
10,000-18,000	mean-square	0.0001
18,000-20,000	cross-entropy	0.0001

(\*)Decreasing and at 2000 iteration the learning rate is 0

TABLE III: The results of our experiments.

Data	MSE	Precision	Recall
Split 1	$1.846 \times 10^{-5}$	0.76128	0.24149
Split 2	$3.215 \times 10^{-5}$	0.14355	0.06887
Split 3	$3.152 \times 10^{-5}$	0.21303	0.12262
<b>Average</b>	$2.738 \times 10^{-5}$	0.37262	0.14432

good initialization weights of the network, we trained our 3D-EDCN classifier on ModelNet40 dataset [12] for 100 epochs and finetuned the classifier using the road edge dataset. The same resolution of  $256 \times 256 \times 256$  was used in ModelNet40 dataset training process. The training process using ModelNet40 dataset was running for around one month with 6.5 GB GPU memory occupied for the process. The finetuning process using road edge dataset was running for 20,000 iterations with training configuration as shown in TABLE II. The finetuning process was run for four days with total twelve days for three split training/testing configuration.

### C. Results and Discussion

We evaluated our 3D-EDCN classifier using three different evaluation method, MSE, precision, and recall analysis. Each evaluation method can be described using following equation

$$MSE = \frac{1}{N} \sum_{n=1}^N (y_n - \hat{y}_n)^2 \quad (4)$$

$$Precision = \frac{TP}{TP + FP} \quad (5)$$

$$Recall = \frac{TP}{TP + FN} \quad (6)$$

where  $N$  is the number of output dimension,  $y_n$  is the true value,  $\hat{y}_n$  is the predicted value,  $TP$  is the True Positive cases,  $FP$  is the False Positive cases, and  $FN$  is the False Negative cases.

TABLE III summarizes the results of our experiments. The MSE error of our 3D-EDCN classifier is very low and very good for the problem. The average accuracy of our 3D-EDCN classifier for road edge detection is very high, over 99%. For further micro analysis, we compute the precision and recall of our 3D-EDCN classifier results. We use a threshold of 0.5 to decide whether the voxel was a road edge or not. From TABLE III, the precision and recall average value was quite low. This means that classifier suffers for negative class over-fitting due



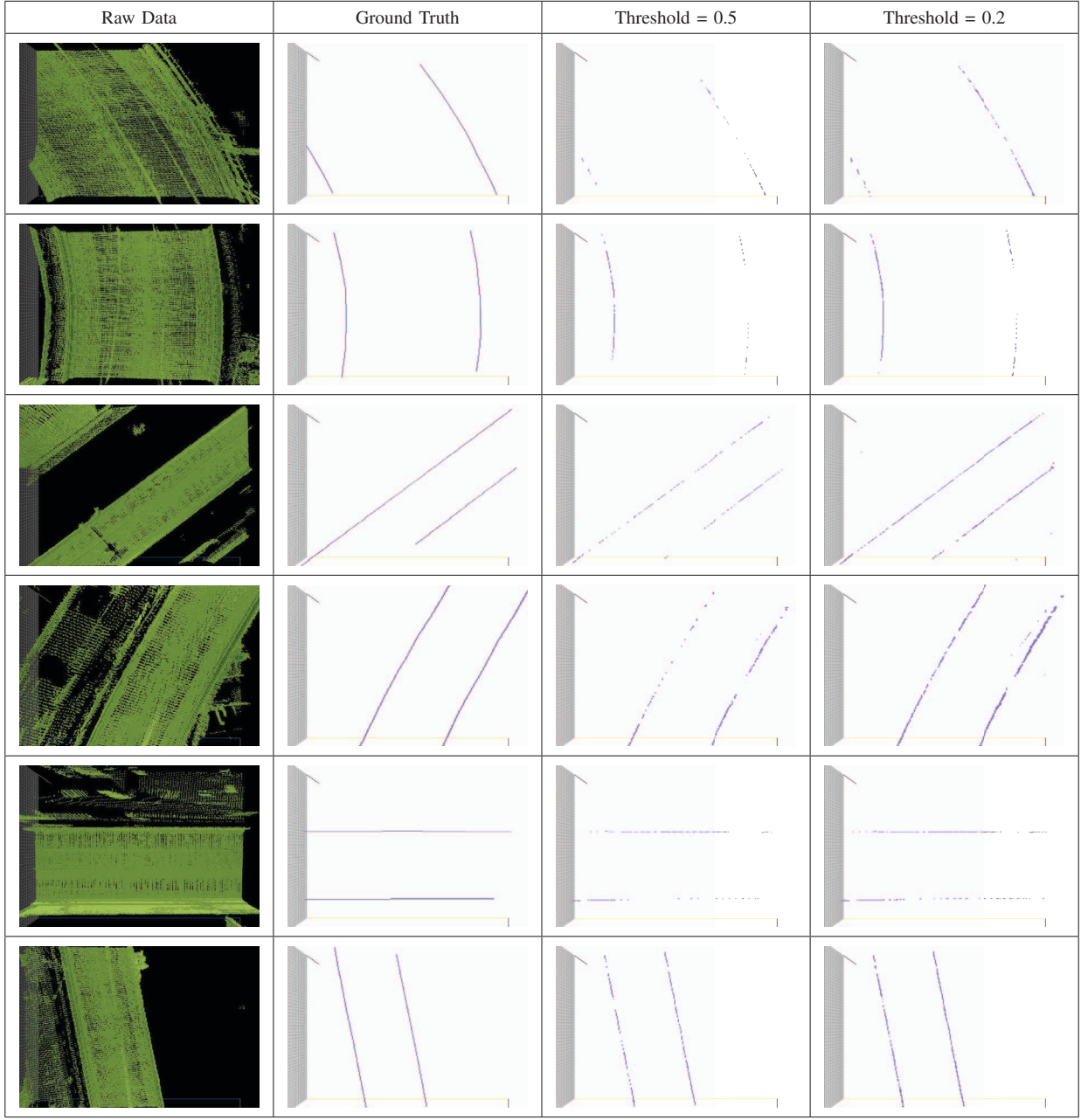


Fig. 4: The result of road edge detection using our 3D-EDCN classifier using two different threshold values. The threshold value was used to decide whether a voxel is a road edge or not.

to the dataset that has more negative class than positive class. The precision and recall show that in the macro level, the 3D-EDCN has very good accuracy and MSE error but in the micro level there are a lot of positive voxels detected as negative voxel (low precision-recall). A large gap of examples between positive and negative class on the label ground truth causes our proposed classifier to negative prediction trend. Generating or collecting more data is required to reduce the negative prediction trend. The data augmentation method is one of the feasible solutions to increase the richness of the data.

Figure 4 shows six example output of our 3D-EDCN classi-

fier along with their respective ground truth. The camera was set to top view for the visualization purpose. Some threshold variation values were used to decide whether the voxel was a road edge or not. As we can see in Figure 4, some results of our 3D-EDCN were quite good but the results is not a continuous line. Figure 5 shows three failed cases of road edge detection using our proposed classifier.

## VI. CONCLUSION

We proposed a 3D Encoder-Decoder Convolutional Network (3D-EDCN) classifier for road edge detection problem

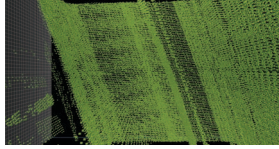


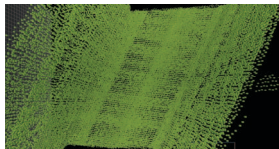

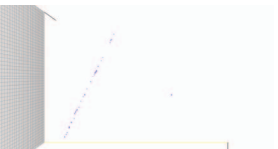
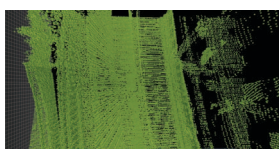


Raw Data	Ground Truth	Threshold = 0.5
		
		
		

Fig. 5: Three failed detection examples of road edge using our proposed classifier.

on 3D point cloud data. Our classifier consists of nine layers of 3D convolutional layer followed by three deconvolutional layers. We use two different loss function with some configuration to train the 3D-EDCN classifier and achieve a very good training loss value. Based on our preliminary results, our 3D-EDCN classifier achieves mean square error of  $2.738 \times 10^{-5}$ , precision of 0.37262, and recall of 0.14432.

For future work, an additional method to reduce the False Negative and False Positive cases is urgently required. A new data augmentation method is also needed to enrich the available dataset. The available data augmentation method cannot be applied to the dataset because the input has a spatial connection with the ground truth. One feasible solution for data augmentation is performing the same data augmentation parameters to the input and the ground truth data.

## REFERENCES

- [1] L. Caltagirone, S. Scheidegger, L. Svensson, M. Wahde. "Fast lidar-based road detection using convolutional neural networks". *arXiv preprint arXiv:1703.03613*, 2017.
- [2] R. F. Rachmadi, Y. Komokata, K. Uchimura, and G. Koutaki. "Road Sign Classification System using Cascade Convolutional Neural Network". *International Journal of Innovative Computing, Informatics, and Control*, Vol. 13, Issue 1, pp. 95-109, 2017.
- [3] R. F. Rachmadi, K. Uchimura and G. Koutaki. "Video classification using compacted dataset based on selected keyframe," 2016 IEEE Region 10 Conference (TENCON), 2016, pp. 873-878.
- [4] O. E. David, and N. S. Netanyahu. "DeepPainter: Painter Classification Using Deep Convolutional Autoencoders". In *International Conference on Artificial Neural Networks (ICANN)*, 2016, pp. 20-28.
- [5] K. Massow, B. Kwella, N. Pfeifer, F. Husler, J. Pontow, I. Radusch, and M. Haueis. "Deriving HD maps for highly automated driving from vehicular probe data". In *IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, 2016, pp. 1745-1752.
- [6] H. G. Seif and X. Hu. "Autonomous Driving in the iCityHD Maps as a Key Challenge of the Automotive Industry". *Engineering*, Vol. 2, Issue 2, pp. 159-162, 2016.
- [7] R. Girdhar, D. F. Fouhey, M. Rodriguez, and A. Gupta. "Learning a predictable and generative vector representation for objects". In *Proceeding of European Conference on Computer Vision (ECCV)*, 2016, pp. 484-499.
- [8] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition". In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770-778.
- [9] B. Leng, S. Guo, X. Zhang, and Z. Xiong. "3D object retrieval with stacked local convolutional autoencoder". *Signal Processing*, Vol. 112, pp. 119-128, 2015.
- [10] D. Maturana and S. Scherer. "3D convolutional neural networks for landing zone detection from LiDAR". In *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 3471-3478.
- [11] D. Maturana and S. Scherer. "VoxNet: a 3D convolutional neural network for real-time object recognition". In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 922-928.
- [12] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. "3D ShapeNets: A Deep Representation for Volumetric Shapes". In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1912-1920.
- [13] K. Simonyan, and A. Zisserman. "Very deep convolutional networks for large-scale image recognition". *arXiv preprint arXiv:1409.1556*, 2014.
- [14] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. "Caffe: Convolutional Architecture for Fast Feature Embedding". In *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 675-678.
- [15] D. P. Kingma, J. Ba. "Adam: A Method for Stochastic Optimization". In *3rd International Conference for Learning Representations (ICLR)*, 2015, pp. 1-15.
- [16] J. Fritsch, T. Kuehnl, and A. Geiger. "A New Performance Measure and Evaluation Benchmark for Road Detection Algorithms". In *International Conference on Intelligent Transportation Systems (ITSC)*, 2013, pp. 1693-1700.
- [17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks". In *Advances in neural information processing systems (NIPS)*, 2012, pp. 1097-1105.
- [18] J. Masci, U. Meier, D. Cirean, and J. Schmidhuber. "Stacked convolutional auto-encoders for hierarchical feature extraction". In *international conference on Artificial neural networks*, 2011, pp. 52-59.
- [19] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. "Greedy Layer-Wise Training of Deep Networks". In *Advances in Neural Information Processing Systems (NIPS)*, 2007, pp. 153-160.