Point Cloud Labeling using 3D Convolutional Neural Network

Jing Huang and Suya You
University of Southern California
Los Angeles, California 90089

***Abstract*—In this paper, we tackle the labeling problem for 3D point clouds. We introduce a 3D point cloud labeling scheme based on 3D Convolutional Neural Network. Our approach minimizes the prior knowledge of the labeling problem and does not require a segmentation step or hand-crafted features as most previous approaches did. Particularly, we present solutions for large data handling during the training and testing process. Experiments performed on the urban point cloud dataset containing 7 categories of objects show the robustness of our approach.**

I. Introduction

Point cloud labeling is an important task in computer vision and object recognition. As a result, each point is assigned with a redefined label, which further serves as a cue for scene analysis and understanding. The classes of interest include most common objects in the urban scenario (Figure 1): large-scale planes (including ground and roof), buildings, trees, cars, poles as well as wires.

Traditionally, hand-crafted features are widely used in existing methods [1], [2], [3]. However, the recent progress of deep learning techniques show that, the simplest feature like pixels can be directly combined with the neural networks and trained together. On the other hand, simply projecting 3D data to 2D representations such as depth images and then applying the 2D techniques can easily lead to loss of important structural information embedded in the 3D representation. Inspired by the success of deep learning on the 2D image problems, we present the voxel-based fully-3D Convolutional Neural Network on the point cloud labeling problem.

In most existing approaches, segmentation is a necessary step before performing tasks such as detection and classification [4]. Our method does not require prior knowledge, such as the segmentation of the ground and/or buildings, the precomputed normals, etc. Everything is based on the voxelized data, which is a straightforward representation. From another point of view, our approach works as an end-to-end segmentation method. Our work shows the power of neural networks to capture the essential features needed for distinguishing different categories by themselves.

Despite the conceptually straightforward idea of representing point clouds as voxels to solve the problem, there are many underlying challenges. First, the dense voxel representation will quickly exceed the memory limit of any computer. Secondly, it will require too much time without proper optimization of the algorithm. Also, the classifier could be easily biased towards some dominating categories (e.g., buildings) without deliberately balancing the training data.

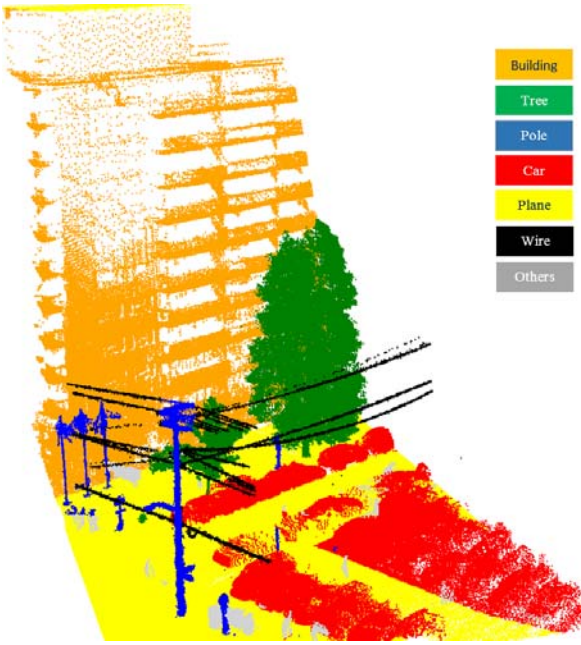


Fig. 1. Objects of interest. The buildings are colored with orange, horizontal planes (e.g., ground and roof) are colored with yellow, trees are colored with green, cars are colored with red, poles are colored with blue, and wires are colored with black. The points in the other categories are colored with light gray.

Our contributions mainly include:

(1) We introduce a framework of 3D Convolutional Neural Network (3D-CNN) and design effective algorithms for labeling complex 3D point could data.

(2) We present solutions for efficiently handling large data during the voxelization, training and testing of the 3D network.

II. Related Work

A. 3D Object Detection and Labeling

Pinheiro et al. applied Recursive Neural Network for scene parsing [5]. Habermann et al. present a 3D object recognition approach based on Multiple Layer Perceptron (MLP) on 2D projected data [4]. However, their method requires data segmentation. Koppula et al. take a large-margin approach to perform the 3D labeling classification based on various features [1]. PCA analysis and the dimensionality feature

based on it have been applied in point-level classification tasks [6], [3]. Still, the parameter selection for the features are highly empirical, and we show in this work that 3D-CNN based on the simplest occupancy voxels could achieve comparable effects without any task-specific features.

B. 3D CNN

3D CNN has been proposed first in the application of video data analysis, because videos can be seen as a temporal 3D extension of the 2D images as have well-defined grid values. Ji et al. proposed 3D-CNN for human action recognition in video data [7]. For 3D point cloud, Maturana and Scherer applied 3D-CNN for landing zone detection from LiDAR point clouds [8]. Prokhorov presented a 3D-CNN for categorization of segmented point clouds [9]. 3D Shape Nets applied 3D CNN to learn the representation of 3D shapes [10]. VoxNet integrated a volumetric Occupancy Grid representation with 3D CNN [11]. All of their approaches require pre-segmented objects before applying the 3D-CNN method. To incorporate the localization problem in the 3D CNN framework, Song and Xiao proposed the deep sliding shapes for 3D object detection in depth images [2] and RGB-D images [12], with the 3D Region Proposal Network (RPN). However, they use the Manhattan world assumption to define the bounding box orientation of indoor objects, which is not feasible for outdoor objects in our case.

III. System Overview

Our labeling system is depicted in Figure 2. The system is composed of an offline training module and an online testing module.

The offline training takes the annotated training data as input. The training data are parsed through a voxelization process that generates occupancy voxel grids centered at a set of keypoints. The keypoints are generated randomly, and the number of them are balanced across different categories. The labels of the voxel grids are decided by the dominating category in the cell around the keypoint. Then, the occupancy voxels and the labels are fed to a 3D Convolutional Neural Network, which is composed of two 3D convolutional layers, two 3D max-pooling layers, a fully connected layer and a logistic regression layer (Section V). The best parameters during training are saved.

The online testing takes a raw point cloud without labels as input. The point cloud is parsed through a dense voxel grid and results in a set of occupancy voxels centered at every grid centers, respectively. The voxels are then used as the input to the trained 3D convolutional network, and every voxel grid would produce exactly one label. The inferred labels are then mapped back to the original point cloud to produce a pointwise labeling result (Section VI).

Note that, due to different requirements of the training and testing modules, the voxelization process are quite different except the parameters such as grid size and voxel number. We will discuss the details of voxelization in Section IV.

IV. Voxelization

We turn the point cloud into 3D voxels through the following process. We first compute the bounding box for the whole point cloud. Then, we describe how the local voxelization is obtained if a center point from the point cloud is chosen. The choice of the center would be different depending on whether we're in the training process or the testing process and will be discussed in the experiment section.

Given the center point (x, y, z), we set up a cubic bounding box of radius R around it, i.e., $[x-R, x+R] \times [y-R, y+R] \times [z-R, z+R]$. Then, we subdivide the cube into a $N \times N \times N$ grid of cells. In our experiment, $R = 6$ and $N = 20$, resulting in a cell of size $0.3 \times 0.3 \times 0.3$ and 8000 cells. We then go through the points that lie within the cubic box and project them with the integer indices. The result of a local voxelization is thus a 8000-dimensional vector.

There are several ways of computing the value of each cell. The simplest one is to compute the occupancy value, i.e., if there's a point inside it, the value becomes 1, otherwise becomes 0. A slightly complicated version is to compute a density value, which could be realized by counting how many points lie within each cell. In our experiment, we find that occupancy value is enough to generate a good result.

By moving the center point around, we can generate a dictionary of different local voxelization results.

Figure 3 shows the original input point cloud and the generated voxels.

The process above is enough for the testing process. However, for training purpose, we need to provide a unique label for each generated voxel grid. We define the label for the entire voxel grid as the label of the cell around its center, i.e., $[x-r, x+r] \times [y-r, y+r] \times [z-r, z+r]$. In our experiment $r = 0.3/2 = 0.15$, so that the size of the cell is identical to the cells constructing the voxel grid. In most cases, there are points of multiple categories lying within a single cell. We apply a voting approach to decide the label of the cell, that is, the category with the most points in the cell will be treated as the representative category of the cell. In the rare case where two or more categories have equal number of points, we just pick a random one.

V. 3D Convolutional Neural Network

After generating the voxels, we feed them to our 3D convolutional neural network. Here are some essential blocks forming the 3D CNN.

A. 3D Convolutional Layer

A 3D convolutional layer could be represented as $C(n, d, f)$, meaning a convolutional layer with input size $n \times n \times n$ and d feature maps with size $f \times f \times f$. Formally, the output at position (x, y, z) on the m-th feature map of 3D convolutional layer l is

$$v_{lm}^{xyz} = b_{lm} + \sum_{q} \sum_{i=0}^{f-1} \sum_{j=0}^{f-1} \sum_{k=0}^{f-1} w_{lmq}^{ijk} v_{(l-1)q}^{(x+i)(y+j)(z+k)}, \quad (1)$$

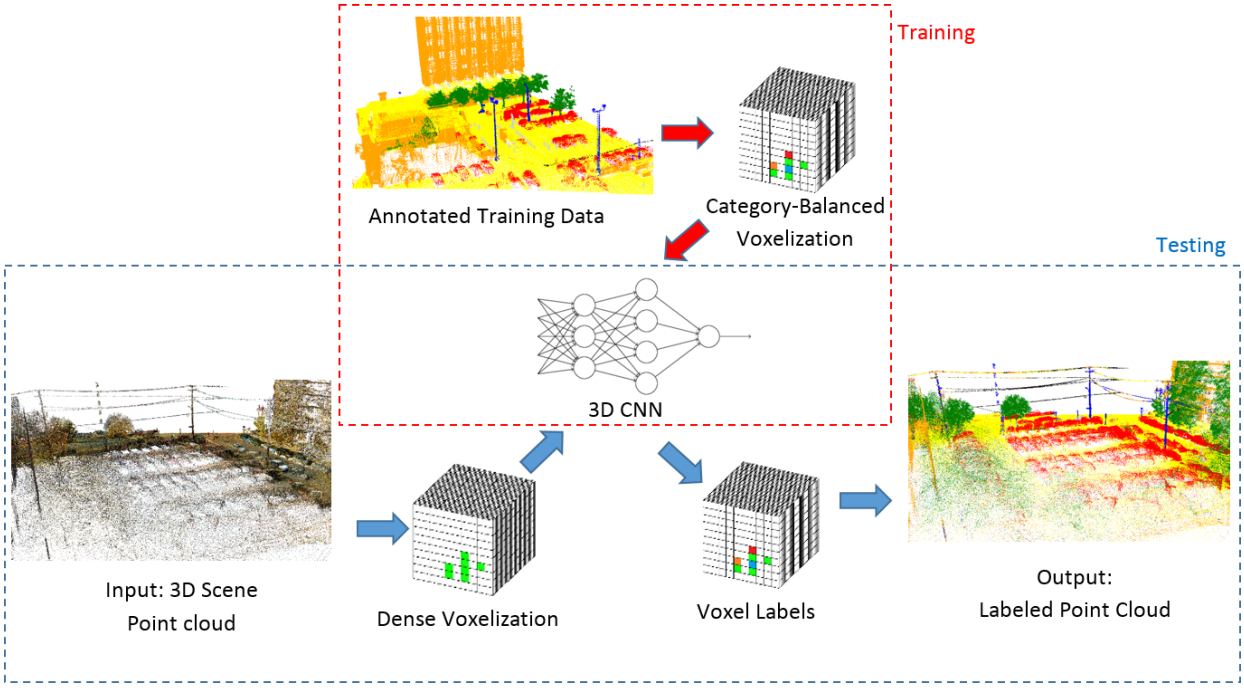
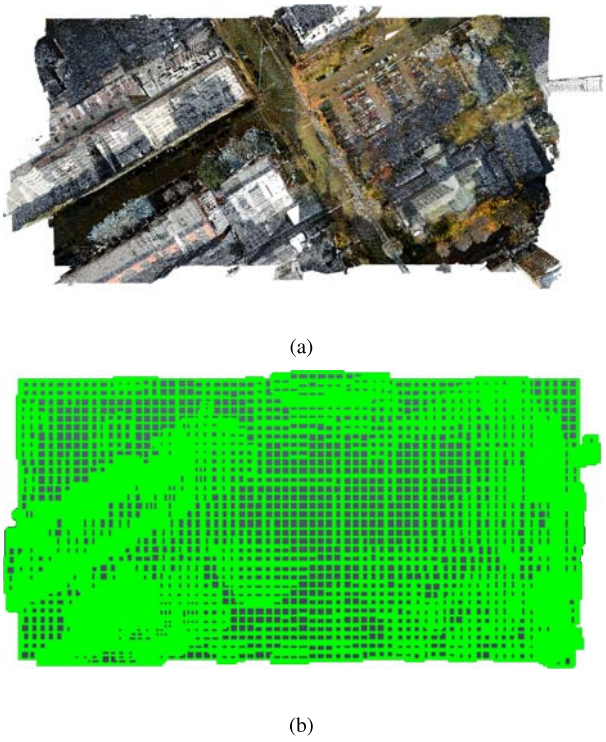


Fig. 2. The labeling system pipeline, including the offline training module and the online testing module.



(a)

(b)

Fig. 3. Illustration for dense voxelization. The input point cloud (a) is parsed through the voxelization process, which generates a dense voxel representation depicted in (b).

where b_{lm} is the bias for the feature map, q goes through the feature maps in the $(l-1)$-th layer, w_{lmq}^{ijk} is the weight at position (i,j,k) of the kernel of the q-th feature map. The weights and the bias will be obtained through the training process.

B. 3D Pooling Layer

A 3D pooling layer can be represented as $P(n,g)$, meaning a pooling layer with input size $n \times n \times n$ and a pooling kernel of $g \times g \times g$. In this approach, we use max pooling. Formally, the output at position (x,y,z) on the m-th feature map of 3D max pooling layer l is

$$v_{lm}^{xyz} = \max_{i,j,k \in \{0,1,\ldots,g-1\}} v_{(l-1)m}^{(gx+i)(gy+j)(gz+k)}. \quad (2)$$

To increase nonlinearity, we use the hyperbolic tangent $(tanh(\cdot))$ activation function after each pooling layer.

C. Network Layout

In terms of the layout of the 2D network, our work is based on the success of LeNet [13], which is composed of 2 convolutional layers, 2 pooling layers and 1 fully-connected layer. We replace the 2D convolutional layers and 2D pooling layers with the 3D convolutional layers and 3D pooling layers, respectively, and obtain our architecture (Figure 4).

The architecture could be represented as $C(n_{c1}, d_{c1}, f_{c1}) - P(n_{p1}, g_{p1}) - C(n_{c2}, d_{c2}, f_{c2}) - P(n_{p2}, g_{p2}) - FC(n_{f1}) - LR(n_{f2})$; $FC(n)$ represents a fully-connected layer with input size n; $LR(n)$ represents a logistic regression layer with input size n. The final output denoting the labeling result is an integer $l \in \{0, 1, 2, ..., L\}$ produced by softmax. In this case

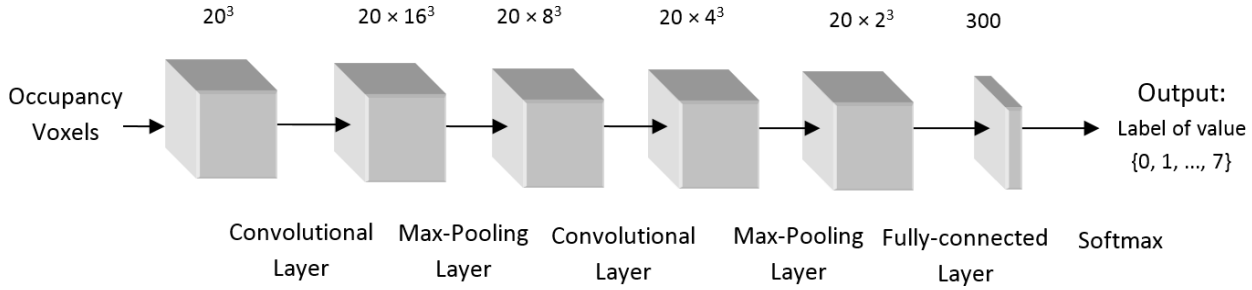


Fig. 4. 3D Convolutional Neural Network. The number on the top denote the number of nodes in each layer. The input is the voxel grid of size 20^3, followed by a convolution layer with 20 feature maps of size $5 \times 5 \times 5$ resulting in 20×16^3 outputs, a max pooling layer with $2 \times 2 \times 2$-sized non-overlapping divisions resulting in 20×8^3 outputs, a second convolutional layer with 20 feature maps of size $5 \times 5 \times 5$ resulting in 20×4^3 outputs, a second max pooling layer with $2 \times 2 \times 2$-sized non-overlapping divisions resulting in 20×2^3 outputs, a fully connected layer with 300 hidden nodes, and the final output is based on a softmax over 8 labels (including 7 categories and an empty label).

$L = 7$, representing the 7 categories specified in Figure 1, while the label 0 denotes the degenerated case where the central region of the voxel grid contains no point.

VI. Label Inference

Given the trained network, we can perform voxel-level classification on the cells. We densely sample the center point of the voxel grids with a distance of 0.3. The coincidence of the center distance and the cell size is intended to generate a unoverlapped compact division of the whole space. Given the labeling result of a local voxel box, we label all points in the cell near the center of the box as the corresponding category.

The compact division would label every point in the data exactly once. However, this would restrict the precision of the labeling to the degree of the cell size. In our experiments, we find the granularity is sufficient to produce quite a good result (see Section VII) with negligible boundary artifacts. In practice, when there're enough computational resources, we can shift the centers around and redo the classification process, and finally, we can employ a voting scheme to decide which label is assigned to each point.

VII. Experiments

A. Dataset and Training Setup

The labeling system is evaluated on a large Lidar point cloud dataset of the urban area of Ottawa. The data come from a fusion of one airborne scanner and four car-mounted scanners. We implement the 3D convolutional network using the Theano library [14] and the network is trained with the Stochastic Gradient Descent method. The size of mini-batch is 30 examples, and the learning rate is 0.1 with a decay rate of 0.95. We manually labeled a few trunks from the entire data, which could generate up to $500k$ features and labels. However, we only selected $50k$ of them as training data and $20k$ of them as validation data under a random and balanced scheme.

B. Balance of Training Samples

During training, we find that a dense sampling of the training data leads to undesired behavior of bias among common categories such as buildings, versus the less common categories such as the wires. Therefore, we implement a balanced random sampling of different categories of the training data. Specifically, we extract the same number of key points from each category, as the centers of the voxelized regions. Experiments show that despite the vastly uneven number of points in category in the real data, the balance of the training data among each categories contributes to a key boost of the performance of the method.

d_{c1}/d_{c2}	10	20	40
20	91.9%	**93.0%**	92.7%

TABLE I
Comparison of different numbers of kernels in the two 3D convolutional layers.

C. Parameter Selection

We perform experiments with a few parameter settings. We fix the kernel size to be 5×5 and evaluate how the numbers of kernels affect the performance. From Table I we can see that the best performance is achieved when $d_{c1} = d_{c2} = 20$, and generally speaking, the parameter here does not have huge impact on the labeling result.

D. Qualitative Result

Figure 6 shows the labeling result of a large area in the city of Ottawa by our approach. Figure 7 shows the closeups. We can see from the result that although we don't have a segmentation step for this task, the generated result shows clearly distinguished labeling result for different objects.

E. Quantitative Result

We evaluate our approach by comparing the results and the ground truth labels. We implement an automatic program to evaluate the methods. The points are sorted according to their coordinates in $O(nlogn)$ time so that the labels can be compared in $O(n)$ time, where n is the number of points.

Figure 5 shows the confusion matrix across the categories. The entry at the i-th row and the j-th column denotes the percentage of points of the j-th truth category that are classified as the i-th category, and the background of the cells is color-coded so that 1 is mapped to black, 0 is mapped to

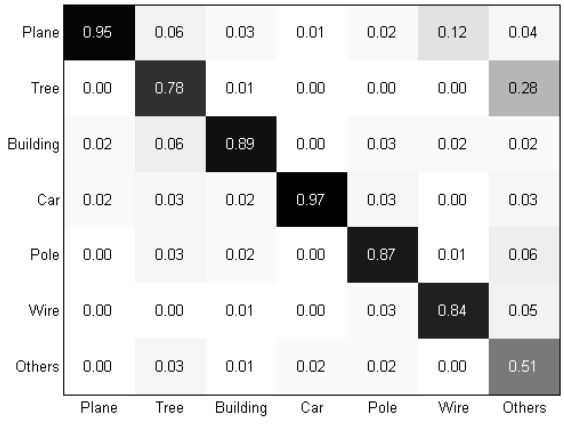


Fig. 5. Confusion matrix for different categories. The entry at the i-th row and the j-th column denotes the percentage of points of the j-th truth category that are classified as the i-th category.

white and anything in between is mapped to the corresponding gray values. From the table we can see that the cars and planes are well-classified with an accuracy higher than 95%, while buildings, poles and wires have an accuracy between 80% and 90%. The accuracy for trees is a little below 80%, mainly due to the confusion with the others category containing many scattered clusters such as humans and bushes. In some area, the parallel wires have a high density and lead to some confusion with the horizontal planes. The overall precision for point labeling of all categories is 93.0%.

Despite the high dimensionality of the voxel representation and 3D CNN, our system is highly efficient in terms of both space and time. The training process takes around 2 hours on a PC with NVIDIA GeForce GTX 980M GPU. For one trunk of data ($100m \times 100m$) and with voxel size of $0.3m \times 0.3m \times 0.3m$, the voxelization process takes less than 5 minutes, and the classification step takes less than 3 minutes.

VIII. Conclusion and Future Work

We propose a 3D point cloud labeling system based on fully 3D Convolutional Neural Network. Our approach does not need prior knowledge for segmentation. In the future, there are a few directions in which we can explore. First, the current labels can be used as input to the network again, which resembles the idea of recursive neural networks. Also, the multi-resolution version of the network might work even better for objects of large scale variance.

References

[1] H. S. Koppula, A. Anand, T. Joachims, and A. Saxena, "Semantic labeling of 3d point clouds for indoor scenes," in *Advances in neural information processing systems*, 2011, pp. 244–252. 1

[2] S. Song and J. Xiao, "Sliding shapes for 3d object detection in depth images," in *Computer Vision–ECCV 2014*. Springer, 2014, pp. 634–651. 1, 2

[3] H. Yokoyama, H. Date, S. Kanai, and H. Takeda, "Detection and classification of pole-like objects from mobile laser scanning data of urban environments," *International Journal of CAD/CAM*, vol. 13, no. 2, 2013. 1, 2

[4] D. Habermann, A. Hata, D. Wolf, and F. S. Osorio, "Artificial neural nets object recognition for 3d point clouds," in *Intelligent Systems (BRACIS), 2013 Brazilian Conference on*. IEEE, 2013, pp. 101–106. 1

[5] P. H. Pinheiro and R. Collobert, "Recurrent convolutional neural networks for scene parsing," *arXiv preprint arXiv:1306.2795*, 2013. 1

[6] J. Demantke, C. Mallet, N. David, and B. Vallet, "Dimensionality based scale selection in 3d lidar point clouds," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 38, no. Part 5, p. W12, 2011. 2

[7] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 1, pp. 221–231, 2013. 2

[8] D. Maturana and S. Scherer, "3d convolutional neural networks for landing zone detection from lidar," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 3471–3478. 2

[9] D. Prokhorov, "A convolutional learning system for object classification in 3-d lidar data," *Neural Networks, IEEE Transactions on*, vol. 21, no. 5, pp. 858–863, 2010. 2

[10] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1912–1920. 2

[11] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 922–928. 2

[12] S. Song and J. Xiao, "Deep sliding shapes for amodal 3d object detection in rgb-d images," *arXiv preprint arXiv:1511.02300*, 2015. 2

[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. 3

[14] J. Bergstra, F. Bastien, O. Breuleux, P. Lamblin, R. Pascanu, O. Delalleau, G. Desjardins, D. Warde-Farley, I. Goodfellow, A. Bergeron *et al.*, "Theano: Deep learning on gpus with python," in *NIPS 2011, BigLearning Workshop, Granada, Spain*, 2011. 4

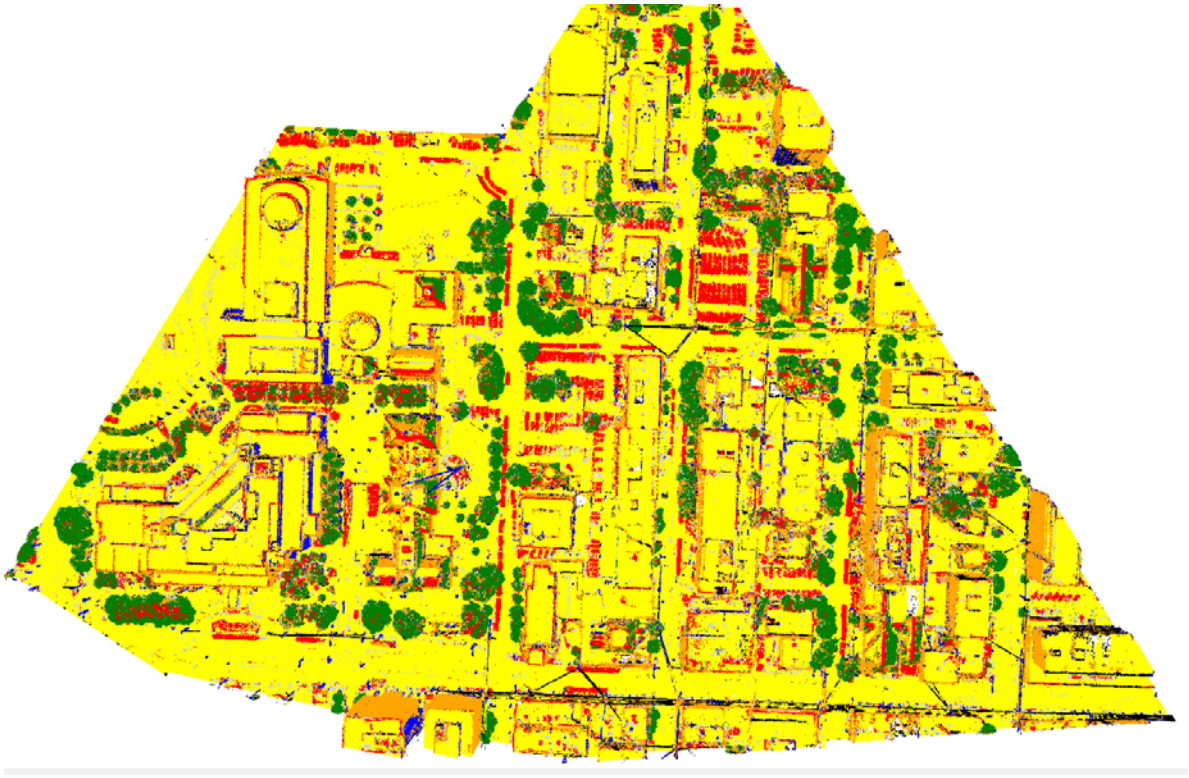
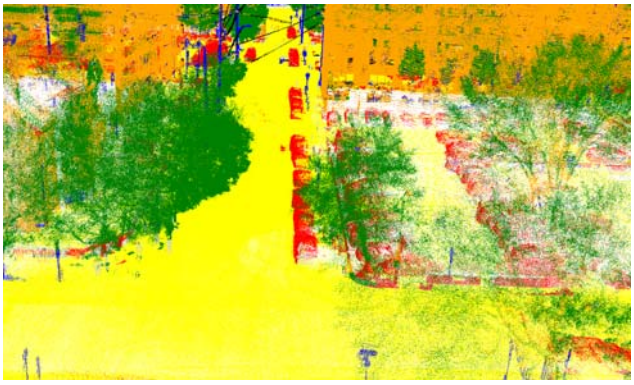
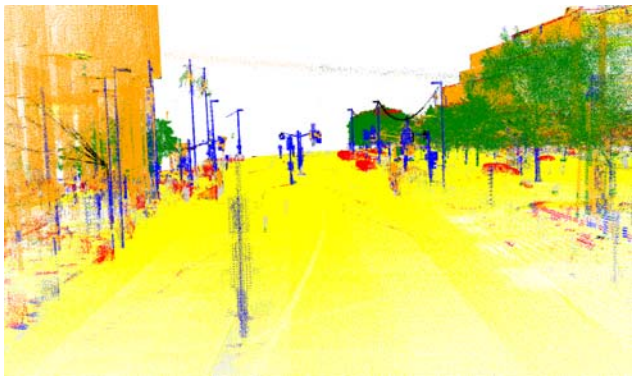


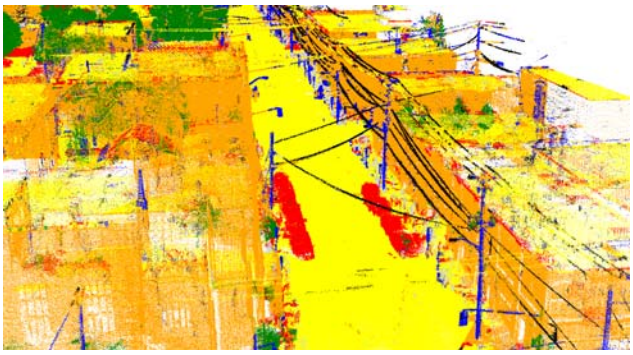
Fig. 6. Labeling result for a large urban area through the 3D-CNN.



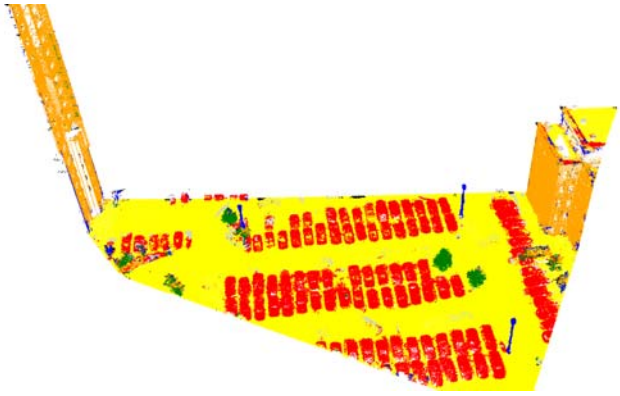
(a)



(b)



(c)



(d)

Fig. 7. Close-ups of the labeling result. (a) The ground planes, buildings, trees and cars. (b) The street view with various poles including light poles, sign poles, utility poles and flag poles. (c) The street view with utility poles and wires. (d) The parking lot scene.