VEHICLE DETECTION IN URBAN POINT CLOUDS WITH ORTHOGONAL-VIEW CONVOLUTIONAL NEURAL NETWORK

Jing Huang and Suya You

University of Southern California

ABSTRACT

In this paper, we aim at detecting vehicles from the point clouds scanned from the urban area. Our detection method consists of a segmentation stage and a classification stage. Prior knowledge for vehicles and urban environment is utilized to help the detection process. Specifically, we incorporate curb detection and removal in the segmentation stage. Moreover, our approach is able to estimate the orientation of the candidates and use it to handle the difficult cases such as the vehicles in the parking lot. In order to distinguish the vehicles from other segments among the 3D point cloud candidates, we develop three architectures of the orthogonal-view CNN, which are based on the orthogonal view projections of the candidates. Detailed evaluations and comparisons are performed on a challenging point cloud dataset of urban area.

Index Terms— Vehicle detection, point cloud, convolutional neural network, orthogonal view

1. INTRODUCTION

Vehicle detection could be seen as a basic task in a variety of applications, including urban modeling at a fine level [1, 2], robot navigation and autonomous driving. Previous works were mostly conducted on the images or videos. For point clouds, the standard approach to vehicle detection is object-based, which involves two steps: candidate generation and verification. For candidate generation, Patterson *et al*. [3] used spin-image to classify the points and then perform clustering, However, their method only deals with cars parking along the roads. Yao *et al*. [4] apply the vehicle-top detection to identify the candidates. Velizhev *et al*. [5] use domain knowledge to generate the hypotheses, while our method does not rely on the street axes as they do. To verify the candidates, one way is to perform matching process based on local/global descriptors with the templates [3, 6]; another way is to compute a few statistics from the candidates [7, 8]; feature voting could also be applied [5]. Unlike their methods, we base the classification on the straightforward visual appearance without hand-crafted descriptors. Golovinskiy *et al*. [1] applied a generic shape-based 3D object detection framework, but the emphasis was on the pole-like objects, and the results for cars were weaker since they did not utilize the properties of the vehicles. Yao [7] compared two methods for vehicle extraction and showed that 3D method has more accurate results than the grid-cell-based method. They used SVM with only five attributes in the classification step, which is capable of dealing with simpler cases, but might fail in the complicated environments.

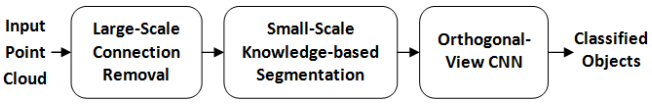


Fig. 1. System overview.

On the other hand, the deep learning technique has achieved a great success in image classification recently. To this end, we apply the deep CNN on the orthogonal-view information from the objects. Our detection and classification system is depicted in Figure 1. The method begins with the removal of large-scale background objects including the terrain, the upper part of the buildings and the curbs. We then exploit the knowledge-based segmentation to generate the candidates. The orientation of the vehicles is a useful yet robust cue, thus plays vital roles in fine segmentation and classification. We further introduce a gap segmentation method for the difficult case of parking lots, which uses the vehicle orientation to limit the number of gap examinations. Finally, we compute the three views of the objects with the orientation and classify them in a trained CNN with orthogonal view projections as input. Note that most 3D extensions of deep neural networks have been toward the temporal domain [9]. There are also a few attempts to incorporate the multi-view information in the deep model [10]. However, they focus on face recognition and reconstruction of multiple views. Therefore, the major contributions of our work are:

(1) We extend the deep learning approach from the 2D image to the 3D domain, by proposing several orthogonal-view-based architectures of CNN for classification of 3D point cloud data. While our focus is on the vehicles, the classification framework is generic and could be applied to any other 3D detection task.

(2) We construct a novel system for segmenting and detecting 3D objects, especially the vehicles, in the urban area, including knowledge-based techniques such as curb removal and gap segmentation for vehicles on the parking lots.

2. LARGE-SCALE CONNECTION REMOVAL

2.1. Ground and Building Removal

Most points in the data are in fact the ground points, which often connect everything together. We remove the ground through the following normal-based algorithm. Given the input point cloud P, we first compute the normal for each point based on the points lying in its neighborhood. We extract all points with the z component of their normals larger than a threshold θ_G. Then, we perform the region growing algorithm within these points with upright normals. The ground point set is defined as the union of all connected components with more than 5000 points, so as to avoid removing the top surface of the vehicles. The upper parts of the buildings that are above the local ground level by 10 meters are also removed.

2.2. Curb Removal

The vehicles are typically seen either on the roads or in the parking lots. Many vehicles on the road are parking along the road side, namely the curbs. Our approach is able to remove the curbs so that the clusters along the curbs are disconnected from each other.

In fact, the problem is similar to the linear segment detection problem. We apply the principal component analysis on the points close to the ground level, and extract those points with linear properties, i.e., one of the eigenvalue is much larger than the other two eigenvalues.

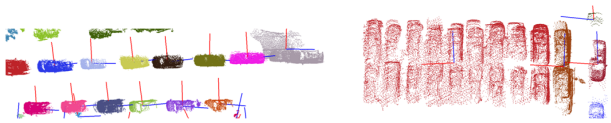
3. KNOWLEDGE-BASED SEGMENTATION

3.1. Adaptive Segmentation

As large-scale background is removed, it's natural to perform the clustering. However, the clusters would be too big with a large margin, while the smaller margin could result in the over-segmentation of the vehicles where the scan is incomplete or sparse. Therefore, we apply the adaptive segmentation method [11], which starts with a large threshold, clusters once, then multiplies the threshold with a decay factor of 0.9 and performs clustering again on the large clusters. This process is repeated until the number of points in each cluster is smaller than a predefined value.

3.2. Orientation Estimation

With the previous steps, while few cars are wrongly removed, the vehicles in the parking lots are usually connected by noises. We notice that the normal distribution would be maximized at three orthogonal directions, one of which is upright, for either a single vehicle or multiple adjacent vehicles in parking lot or on the street. Therefore, we extract the maximum of the horizontal normal distribution histogram to solve the problem. Specifically, we compute the normals for all points in the cluster. Then, we project the ones that are pointing to the near-horizontal directions ($n_z < 0.2$), to the horizontal plane. After that, we quantize the directions and make a histogram of the number of normals lying in each bin (the ones with opposite directions are added up). Finally, the bin with the maximum number of normals is treated as the principal bin, and the direction perpendicular to it as the secondary bin. The accurate principal direction is computed as the average of the normalized projected normals in the principal bin and the adjacent bins. Figure 2 shows the results of orientation estimation.



(a) (b)

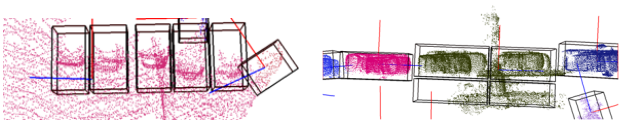
Fig. 2. Results for cluster orientation estimation. (a) Orientation estimation result for candidate clusters along the road. The red line is the principal axis and the blue line is secondary axis. The estimation is pretty close to human perception for the cars. (b) For clusters containing multiple cars (typically in the parking lots), the orientation can be estimated as well.

3.3. Gap Segmentation

Using the orientation information, we can now fit a more accurate bounding box and improve the segmentation result, since the cars typically have rectangular shapes. Furthermore, when viewing from one of the estimated axes, gaps could be observed between the nearby cars. Therefore, we develop the following gap detection method for each cluster: (1) Compute the principal and secondary axes; (2) Along each axis, compute the maximum local height within a quantization unit (interval); (3) Record if the heights of the intervals are larger than a threshold; (4) If several consecutive intervals are higher than the threshold (bounded by intervals lower than the threshold), we consider it as a possible block.

Then, we combine the feasible consecutive intervals. Specifically, if we have M and N feasible consecutive intervals in the principal and the secondary directions, respectively, we can generate $M \times N$ sub-regions from the region. Figure 3 shows some gap segmentation results.

Finally, the separated area is segmented again using a more accurate local orientation, since the orientation of a larger cluster could be slightly different from that of its sub-clusters. We'd also like to remove areas outside the boxes or boxes that are almost empty. This is done through the iterative version of the algorithm, which starts with the process of orientation estimation and gap segmentation, then filters the generated candidates and iterates the process on the remaining sub-clusters until no significant change is made.



(a) (b)

Fig. 3. Results for gap segmentation. In (a), the large cluster is divided into 1×5 sub-regions. Also, the ground that failed to be removed is correctly excluded in the result. In (b), the connected cars along the road is separated.

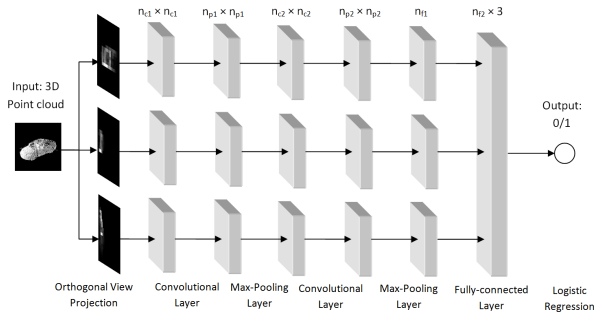


Fig. 4. Orthogonal-View CNN with fusion at the fully-connected hidden layer, which performs the best among the three architectures. All networks have 6 layers.

4. CLASSIFICATION USING OV-CNN

Once the vehicle candidates are segmented in the previous stages, we employ the learning-based classifier to distinguish between the vehicle clusters and the non-vehicle clusters, mainly including facades, bushes, trees, poles and roofs.

4.1. Orthogonal View Projection

For each instance, we generate 3 images ($n \times n$ pixels) from 3 orthogonal views on XY-plane, XH-plane and YH-plane. Note that we restrict each dimension to be no more than l meters. The intensity is proportional to the number of points lying in the corresponding bins. The center of the three views is located at $(\bar{x}, \bar{y}, z_{min} + l)$, where $\bar{x}$ and $\bar{y}$ are the average of the coordinates. One of the major difficulties in the 3D projection is that the direction is unknown. However, since we have estimated principal and secondary orientations of the clusters in Section 3.2, we can directly employ them as the projection directions. A brief test showed that the classifier using the oriented projection performs better than the projection using world coordinates. This is reasonable since we focus on vehicles, which have well-defined orientations.

4.2. Network Architecture

In terms of the layout of a single-view network, our work is based on the success of LeNet [12]. Several structures of CNN that fuse information of orthogonal views are explored.

CNN Combined with Voting. This is the simplest architecture, which counts on votes cast by the classification result of the three identical CNNs on separate views. This acts as a baseline method for evaluating the effect of combination of information from different views. The architecture could be represented as $3 \times (C(n_{c1}, d_{c1}, f_{c1}) - P(n_{p1}) - C(n_{c2}, d_{c2}, f_{c2}) - P(n_{p2}) - FC(n_{f1}) - LR(n_{f2})) - \Sigma_{\geq 2}(3)$. Here $C(n, d, f)$ represents a convolutional layer with input size $n \times n$ and d filters with size $f \times f$; $P(n)$ represents a max-pooling layer with input size $n \times n$; $FC(n)$ represents a fully-connected layer with input size n; $LR(n)$ represents a logistic regression layer with input size n; and $\Sigma(n)$ represents a simple summing operation of the outputs from the logistic regression layers. In other words, the final output is based on the sum of the outputs (0 or 1) from three parallel single-view CNNs, and gets activated only when at least two single-view CNNs output 1. The CNNs are trained using all three views without knowing exactly which direction the view is projected, thus sharing identical weights.

Fused CNN at LR Layer. In this architecture, the outputs of the fully-connected layers do not explicitly decide the class of each view, but rather be concatenated and used as the input for the one single final logistic regression layer. In this way, different views have the chance to affect each other in the very last step. Using the notations above, the architecture could be written as $3 \times (C(n_{c1}, d_{c1}, f_{c1}) - P(n_{p1}) - C(n_{c2}, d_{c2}, f_{c2}) - P(n_{p2}) - FC(n_{f1})) - LR(n_{f2} \times 3)$.

Fused CNN at Fully-Connected Hidden Layer. In this architecture, the outputs of the second max-pooling layers are not used in the decision of each view, but be concatenated and used as the input for one unified fully-connected hidden layer. The architecture could be written as $3 \times (C(n_{c1}, d_{c1}, f_{c1}) - P(n_{p1}) - C(n_{c2}, d_{c2}, f_{c2}) - P(n_{p2})) - FC(n_{f1} \times 3) - LR(n_{f2})$ (Figure 4).

5. EXPERIMENTS

5.1. Experimental Protocol

We evaluate our detection system on a large Lidar point cloud dataset of the urban area of Ottawa [1]. The data was a fusion of one airborne scanner and four car-mounted scanners, which provides higher density along the streets but lower density far from the streets (e.g., parking lots). We implement the networks using the Theano library [13] and train them with the Stochastic Gradient Descent method. The size of mini-batch is 10 examples (30 views), and the learning rate is 0.1 with a decay rate of 0.95. The training and validation dataset containing 436 examples is annotated from segmented clusters outside the 50 blocks of testing dataset. We implement an automatic program to evaluate the methods. We manually annotated 50 trunks of data of $100m \times 100m$ area, containing 728 instances of cars and over 30 million points. If a detected instance has an overlapping percentage of 50% with a ground

ID	Method	TP	FP	Precision	Recall
0	Baseline	58	39	54.2%	58.0%
1	0+Curb	70	43	56.9%	70.0%
2	1+Adaptive	88	89	47.1%	88.0%
3	2+Gap	93	89	48.4%	93.0%

Table 1. Comparison of segmentation result.

d_{c1}/d_{c2}	10	20	30	40
10	80.0%	80.0%	80.0%	77.7%
20	80.8%	**83.8%**	82.3%	80.8%
30	78.5%	79.2%	78.5%	77.7%

Table 2. Evaluation result of different numbers of kernels in the two convolutional layers.

truth instance, we count it as a correct detection.

5.2. Evaluation of Segmentation

Table 1 demonstrates the progress of segmentation methods, including the baseline method using ground removal and clustering and three incremental methods (curb removal, adaptive segmentation and gap detection).

5.3. Parameter Selection

We experimented with a few parameter settings. For clarity, we only list the parameter evaluation for the OV-CNN fused at the FC layer, which is superior in the comparison. The parameter selection process for the other two networks is similar.

Kernels. We fix the kernel size to be 5×5 and evaluate how different numbers of kernels affect the performance. We can see from Table 2 that the best performance is achieved when $d_{c1} = d_{c2} = 20$.

Patch size. To determine the visual patch size, we fix the two pairs of kernels that perform the best in the evaluation above. Table 3 shows the effect of how different patch size affect the performance. We can see that when the input view patch size $n_{c1} = 28$, the best performance is achieved.

5.4. Evaluation of Classification Architectures

To evaluate the performance of the three architectures, we apply the best parameters selected from the previous subsection and compare the results. We also apply SVM on the concatenated representation of the three views as the baseline method. From Table 4 we can see that, compared to SVM, Voting-based CNN does not show better performance, while the two fused architectures give better results. The best OV-CNN is the one that fuses at the fully-connected hidden layer. This is reasonable since the connections among different orthogonal views are more likely to be modeled through the early-fused layers. The best architecture turn out to be $3 \times (C(28, 20, 5) - P(24) - C(12, 20, 5) - P(8)) - FC(16 \times 3) - LR(300)$. Figure 5 shows the final detection results. [1]

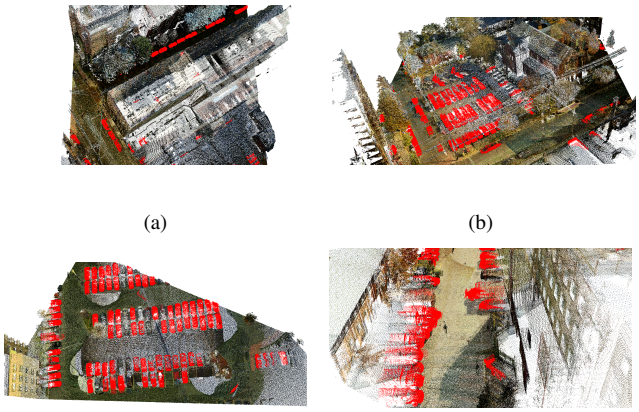
The whole system is highly efficient. For one block of data ($100m \times 100m$), the segmentation step takes less than 5 minutes, and the classification step takes less than 1 minute.

$(d_{c1}/d_{c2})/n_{c1}$	24	28	32	36
(20/20)	80.0%	**83.8%**	80.0%	80.0%
(20/30)	78.5%	82.3%	80.8%	70.8%

Table 3. Evaluation of various view patch sizes. n_{c1} must be divisible by 4 to satisfy the constraints.

Classifier	SVM	Voting	LR-Fused	FC-Fused
Correct%	77.5%	75.9%	80.0%	**83.8%**

Table 4. Comparison of three CNN architectures and SVM.



(a) (b) (c) (d)

Fig. 5. Vehicle detection results. The vehicles are highlighted in red color. (a) shows the example of vehicles parking along the street. (b) and (c) show the example of large parking lots. (d) shows that our method could handle both cars and trucks.

6. CONCLUSION

In this work, we develop a CNN that could deal with 3D point cloud classification using orthogonal views. Combining it with knowledge-based segmentation techniques, we can efficiently perform vehicle detection from the urban point clouds. We evaluate and demonstrate the performance of our method through detailed experiments. Although this work focuses on vehicles, the OV-CNN architecture could easily be applied to other 3D object detection and classification tasks.

[1] We have included a supplementary MP4 file which contains intermediate and final results of our method. This will be available at http://ieeexplore.ieee.org.

7. REFERENCES

[1] Aleksey Golovinskiy, Vladimir G Kim, and Thomas Funkhouser, "Shape-based recognition of 3d point clouds in urban environments," in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 2154–2161. 1, 3

[2] Nico Cornelis, Bastian Leibe, Kurt Cornelis, and Luc Van Gool, "3d urban scene modeling integrating recognition and reconstruction," *International Journal of Computer Vision*, vol. 78, no. 2-3, pp. 121–141, 2008. 1

[3] Alexander Patterson IV, Philippos Mordohai, and Kostas Daniilidis, "Object detection from large-scale 3d datasets using bottom-up and top-down descriptors," in *Computer Vision–ECCV 2008*, pp. 553–566. Springer, 2008. 1

[4] Wei Yao, Stefan Hinz, and Uwe Stilla, "Automatic vehicle extraction from airborne lidar data of urban areas aided by geodesic morphology," *Pattern Recognition Letters*, vol. 31, no. 10, pp. 1100–1108, 2010. 1

[5] Alexander Velizhev, Roman Shapovalov, and Konrad Schindler, "Implicit shape models for object detection in 3d point clouds," *Proc. ISPRS Congr*, pp. 1–6, 2012. 1

[6] Bogdan Matei, Ying Shan, Harpreet S Sawhney, Yi Tan, Rakesh Kumar, Daniel Huber, and Martial Hebert, "Rapid object indexing using locality sensitive hashing and joint 3d-signature space estimation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 7, pp. 1111–1126, 2006. 1

[7] Wei Yao and Uwe Stilla, "Comparison of two methods for vehicle extraction from airborne lidar data toward motion analysis," *Geoscience and Remote Sensing Letters, IEEE*, vol. 8, no. 4, pp. 607–611, 2011. 1

[8] Jixian Zhang, Minyan Duan, Qin Yan, and Xiangguo Lin, "Automatic vehicle extraction from airborne lidar data using an object-based point cloud analysis method," *Remote Sensing*, vol. 6, no. 9, pp. 8405–8423, 2014. 1

[9] Shuiwang Ji, Wei Xu, Ming Yang, and Kai 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. 1

[10] Zhenyao Zhu, Ping Luo, Xiaogang Wang, and Xiaoou Tang, "Multi-view perceptron: a deep model for learning face identity and view representations," in *Advances in Neural Information Processing Systems*, 2014, pp. 217–225. 1

[11] Jing Huang and Suya You, "Segmentation and matching: Towards a robust object detection system," in *Winter Conference on Applications of Computer Vision (WACV)*, 2014. 2

[12] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. 3

[13] James Bergstra, Frédéric Bastien, Olivier Breuleux, Pascal Lamblin, Razvan Pascanu, Olivier Delalleau, Guillaume Desjardins, David Warde-Farley, Ian Goodfellow, Arnaud Bergeron, et al., "Theano: Deep learning on gpus with python," in *NIPS 2011, BigLearning Workshop, Granada, Spain*, 2011. 3