2016 IEEE Intelligent Vehicles Symposium (IV)
Gothenburg, Sweden, June 19-22, 2016

Directional-DBSCAN: Parking-slot Detection using a Clustering Method in Around-View Monitoring System

Soomok Lee, Daejin Hyeon, Gikwang Park, Il-joo Baek, Seong-Woo Kim and Seung-Woo Seo

Abstract— **Parking slot detection algorithms using visual sensors have been required for various automated parking assistant systems. In most previous studies, popular feature detectors, such as the Harris corner or the Hough line detector, have been employed for detecting parking slots. However, these algorithms were originally designed to find distinct features and are inadequate for the short, curvy, faint and distorted parking-lines of long-range surround-view images, especially in around-view monitoring systems. In this paper, we propose a robust parking slot detection algorithm based on the line-segment-level clustering method. The proposed algorithm consists of line-segment detection with the proposed Directional-DBSCAN line-level feature-clustering algorithm and slot detection with slot pattern recognition. In comparison to other feature detectors, we show that the Directional-DBSCAN algorithm robustly extracts lines even when they are short and faint. Moreover, we verify that the parking-slot detection algorithm with pattern recognition can be applicable to diverse slot types and environments with experiments on abundant dataset.**

I. INTRODUCTION

So far, a large number of automated parking assistant systems (PASs) have been proposed to maximize both the safety and convenience of parking. The selection of a particular parking slot among available parking spaces is a primary component in a parking assistant system. Perception methods for available parking space detection have been developed mainly in two ways: The free-space-based approach [11]-[14] and the parking-slot-marking-based approach [2]-[10].

In the free-space-based approach, different types of sensors with ranging capability (e.g. laser scanners and ultrasonic sensors) have typically been employed. This approach guarantees stable measurement, but is highly dependent on the existence and pose of adjacent vehicles. On the other hand, in the marking-based approach, slot markings are the main means of finding parking spaces and provide accurate positions and heading angles. With this information, parking becomes more accurate within a parking space. For these reasons, the slot-marking-based approach has been the focus of recent research rather than the free-space-based approach. Moreover, the marking-based approach has become more influential for PASs since the wider field-of-view (FOV) has become available with multiple visual sensor configurations, such as around-view monitoring (AVM) systems recently.

S. Lee, D. Hyeon, G. Park, S. Kim and S. Seo are with the Department of Electrical Engineering and Computer Science, Seoul National University and I. Baek is with LG Electronics, {soomoklee, daejin.hyeon, bnooo, sinabrlo and sseo}@snu.ac.kr and iljoo.baek@lge.com. This research was supported by LG Electronics (No.0626-20150002), partially by the National Research Foundation of Korea (NRF) grant funded by the Ministry of Science, ICT & Future Planning (MSIP) (No. 2009-0083495) and the Institute of New Media and Communications and Automation and the Systems Research Institute, Seoul National University.

TABLE I. Technology migration in parking-slot detection.

Funct. \ Method	Corner-based	Line-based	
Feature Extraction	Harris/Fast	Detector	Clustering
	[5]-[9]	[2]-[4]	Ours
Pattern Recognition	[6]-[9]	Ours	

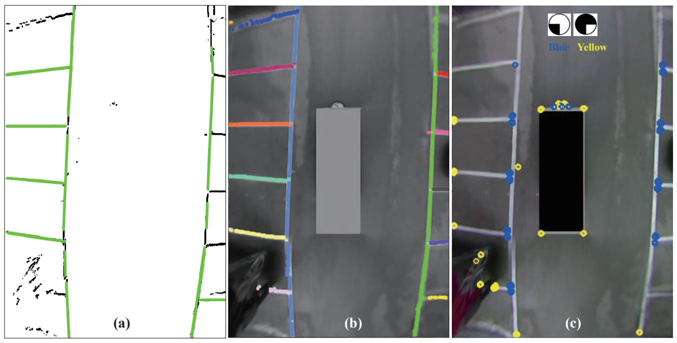


Fig. 1. Difficulties in parking-line detection on distorted, short, and curvy lines: With same marking features (black), (a) Probabilistic HT detector (green)[3] missed many short lines, and long curved lines are snapped compared with (b) proposed D-DBSCAN results. (c) Corner detector [6] is also weak at blurry and distorted angles.

A variety of parking slot detection methods via the slot-marking-based approach have been proposed. With the aspects of feature utilization, the methods can be categorized into two types: Slot-corner-detection-based [5]-[10] and line-detection-based methods [2]-[4]. The corner-detection-based method using FAST corner [5] and Harris corner [6]-[9] is used to find the junction points of parking slots. *Jae-kyu et al.* [6]-[9] utilize corner features to recognize slot patterns and parking spaces. By suggesting a hierarchical classifier considering corner angles, junction types, and parking slot patterns, various kinds of parking slots can be potentially handled. Despite decent performances, these approaches have various limitations. First, corner features cannot always provide precise target positions and heading angles. In addition, since it is difficult for a corner feature to maintain robustness, these algorithms contain excessive constraints in scenarios when 1) multiple frames and slots are required to make a decision, 2) the entering side of slots should be visible, and 3) any distortion on the slot angle caused by the vehicle pitch and roll motion does not exist.

Meanwhile, the parking-line detection-based method provides an accurate direction and has fewer constraints compared with corner-detection-based methods. In addition, it is robust on occlusions caused by an ego-vehicle or objects. In the previous study, line detectors, including the Hough transform (HT) [2][3] or distance transform [10], were utilized to find parking-lines. However, conventional line-detection-based methods mostly target a limited number of slots with a monocular camera having narrow FOV. Additionally, since the AVM image is the synthesized result of four calibrated images, any partially uneven road cause distortions in parking-lines. Consequently, the direct adaptation of these methods in AVM images is fragile for short and distorted lines, as shown in Fig.1-(a). Most importantly, these methods are confined to the simple rectangular slot type in the absence of slot pattern recognition.

In this paper, we present a parking-slot detection algorithm using the parking-line clustering method. The proposed algorithm consists of parking-line segment detection based on directional density-based spatial clustering (Directional-DBSCAN/D-DBSCAN) and parking-slot detection with slot pattern recognition by analyzing line segments. The main contributions of this study can be summarized as follows:

a) We propose a D-DBSCAN clustering algorithm based on DBSCAN [1], but manipulated to be suitable for parking-slot detection applications. The proposed D-DBSCAN not only follows the original attributes (i.e. density reachability and density connection), but it also validates the orientation similarity of adjacent points. As a result, each orthogonal or slanted line segment can be clustered individually. The proposed D-DBSCAN-based line detection has higher capabilities in handling curved, distorted, and short parking-line marks compared with other line detectors.
b) We also propose a slot pattern recognition method based on a decision tree classifier analyzing line segments. Various types of parking slots are identified by the geometric characteristics among line segments following the corresponding pattern. The slot pattern recognition has the following benefits. First, parking-slot detection can be more robust by utilizing the individual characteristics of slot patterns and facilitating outlier elimination involved with non-parking landmarks and objects line edges. Additionally, the vehicle maneuvering strategy can be altered in accordance with the recognized slot pattern.
c) We demonstrate the robustness of the proposed algorithm in diverse types of scenarios. With abundant dataset, including hard weather conditions and extraordinary parking regions, we quantitatively demonstrate the robustness of our algorithm. Diverse parking slots with numerous slot pattern has been included in the demonstration.

The remainder of this paper is organized as follows. Section II presents the parking-line segment detection stage using D-DBSCAN. The step is comprised of line-marking extraction, D-DBSCAN clustering, and line-fitting. The slot pattern recognition and detection stage, which consists of geometric feature extraction, pattern classification, and slot detection, is introduced in Section III. Section IV demonstrates experimental results and an evaluation of the proposed algorithm. Lastly, Section V draws conclusions.

II. Parking-line-segment Detection using Directional-DBSCAN

This section shows how parking-line-segments are extracted through the proposed clustering method. Accurate and robust parking-line detection plays an important role in two aspects. The recognition result is utilized not only to localize an automated vehicle directly using the positions and orientation of the slot, but also to work as input data for slot pattern recognition in the next section. The parking-line-segment detection stage consists of four steps: Pre-processing for the around-view image, line-marking extraction using the cone-hat filter, marking clustering at a line segment level using D-DBSCAN, and line fitting using RANSAC.

A. Pre-process for Setting Around View Monitoring System

AVM is a system that provides a surround-top-view image by concurrently merging four images from front, left, right, and rear fish-eye cameras. By calibrating the intrinsic and extrinsic parameters of an individual camera and optimizing the distortions and mismatches of the overall camera, an appropriate AVM image can be generated [18]. In this paper, we utilize AVM images with pre-defined calibration parameters and do not directly access individual cameras.

B. Parking-line Marking Feature Extraction

In parking-slot recognition using parking-line detection, a robust marking extraction filter is a prerequisite for finding line segments. Line-marking features on roads are extracted by considering their ridge properties. The utilization of a ridge filter is able to cover most of the harsh conditions of shadows, dirt, or cracks on the road. As in [15], we extract lane-markings around each pixel by the following equation:

$$\begin{aligned} L[x] &= I[x] * f[x], \\ (I*f)[n] &= \sum^{img.width} I[n-m]f[m], \end{aligned} \tag{1}$$

where $I[x]$, $f[x]$ and $L[x]$ indicate image intensity, top-hat filter and its response function respectively. The size of filter $f[x]$ is designed to fit into the marking width. We hold local maxima of the filter response $L[x]$ exceeding a certain threshold and regard it as a marking feature. The filter can be built in two-dimensional manner or it can be filtered with respect to x and y axis independently. For all extracted marking features $x_i, \forall i \in \mathcal{N}$, their center position is a utilized dimension for clustering and put to the feature data $\mathbf{D}$, $x_i \in \mathbf{D}$.

C. Line-segment-level Clustering using Directional-DBSCAN

In almost all previous works, line detectors are widely adapted for finding line segments, such as the HT transform, but are fragile for both distorted AVM images and short-length lines, which are frequently occurred. Therefore, we adapt a clustering method rather than utilize the popular line-segment detection algorithm. We propose a novel line-segment clustering algorithm, D-DBSCAN, which basically follows DBSCANs properties, but clusters within the line-segment level. In this section, we explain the additional suggested features of D-DBSCAN compared to DBSCAN.

1) DBSCAN: Among the existing clustering algorithms, we have chosen a DBSCAN algorithm since it has the ability in discovering clusters with arbitrary shapes such as linear, concave, oval, etc. Furthermore, in contrast to some clustering algorithms, it does not require the predetermination of the number of clusters. DBSCAN [1] has been proven its ability of processing very large databases. Before mentioning DBSCAN algorithm, some definitions are explained as:

- Eps: a maximum radius for the boundary region to consider as a same cluster.
- Eps-neighborhood: Eps-neigborhood of a point p is defined as $N_{eps}(q) = \{q \in D | dist(p,q) \leq Eps\}$
- Neigbhorhoods : The set of points which satisfy the eps-neighborhood condition.
- MinPts: MinPts stands for a minimum number of eps-neighbor points to form a core point.
- Core point: A point whose number of eps-neighborhood points is larger than MinPts.
- Border point: A point is an eps-neighbor of the core point but its eps-neighborhood does not exceed MinPts.
- Noise : A point is the initial seed point and its eps-neighborhood points are lesser than MinPts.
- ClusterID (cID) : A class index of a cluster-set.

DBSCAN determines the cluster density by the number of eps-neighborhood points. Points with a density above a specified threshold are constructed as clusters. By examining all eps-neighorhood of core points in an aforementioned manner, density reachable and connectivity of the cluster is analyzed.

The main procedure of DBSCAN is presented in Algorithm 1. For each point in marking feature data **D**, a undesignated point is examined as a core point based on density within *eps* and expands the examination to eps-neighborhoods if the initial point D[i] meets the core point criterion. This procedure is performed by *ExpandCluster* function. If the cluster meets border points and *ExpandCluster* terminates, a new cluster with the consecutive ClusterID is assigned to unclassified points.

Algorithm 1 DBSCAN [1]

```
DBSCAN(D, Eps, MinPts)
1: cID = 1
2: for: i ← 1 to |D|
3:    if D[i].cID = Unclassified
4:      if ExpandCluster(D,D[i],cID,Eps,MinPts)=true
5:          cID = cID + 1
6:      end if
7:    end if
8: end for
```

2) Directional-DBSCAN: The additional main feature of D-DBSCAN is orientation analysis of core point and eps-neighborhood when expanding adjacent points into the same cluster. In D-DBSCAN, a new property, orientation similarity, is additionally included from DBSCAN's original properties (density reachability and density connectivity). Within the original properties in DBSCAN, let a point p be inside of the Eps-neighborhood of a point q and their number $|N_{eps}(q)|$ exceeds $MinPts$ which meets the core point criterion. Then, the additional definition for D-DBSCAN is elaborated as in the following:

Definition 1: (Orientation Similarity) Let O_f be an orientation of an initial core seed and O_{q2p} be an orientation from q to p, then orientations between O_f and O_{q2p} should be similar within a predefined similarity bound.

The main algorithm flow of D-DBSCAN is the same in DBSCAN(Algorithm 1), but the directivity analysis and orientation similarity evaluation (pseudo-code #6-8, 24-25 and 27 in Algorithm 2) are added from the original *ExpandCluster()* function. When an initial core point and its eps-neighborhood point set is found, a derived orientation by *DirectivityAnalysis()* function among the initial seeds is evaluated whether it is unidirectional. When the orientation is considered as unidirectional, D-DBSCAN continues to find a new point which satisfies both density connectivity and orientation similarity until it meets a border point.

Algorithm 2 ExpandCluster for D-DBSCAN

```
ExpandCluster(D, p, cID, Eps, MinPts)
1: seeds ← Neighborhoods(p,Eps)
2: if |seeds| < MinPts
3:    p.cID ← Noise
4:    return false
5: end if
6: if DirectivityAnalysis(p,seeds,direct)≠ unidirection
7:    return false
8: end if
9: p.cID ← cID
10: seeds.delete(p)
11: while: seeds ≠ ∅
12:    q ← seeds.pop()
13:    if q.cID ∉ { Unclassified, Noise}
14:      continue
15:    end if
16:    q.cID ← cID
17:    seeds.delete(q)
18:    S ← Neighborhoods(q,Eps)
19:    if |S| < MinPts
20:      continue
21:    end if
22:    for: x ← 1 to |S|
23:      if x.cID ∉ {Unclassified, Noise}
24:        x.orientation ← getOrientation(q.cID, x)
25:        if |x.orientation − direct | < similarity bound
26:          seeds.append(x)
27:        end if
28:      end if
29:    end for
30: end while
31: return true
```

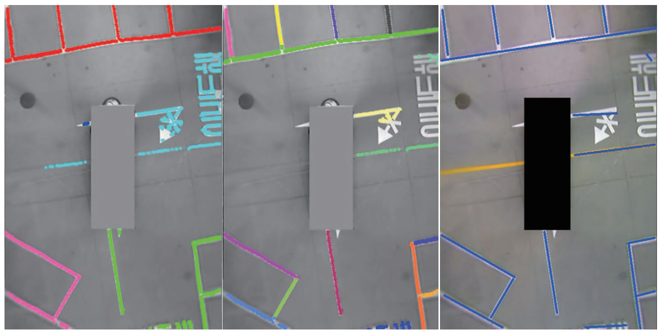


Fig. 2. From left to right: DBSCAN clustering, D-DBSCAN clustering, line fitting results of D-DBSCAN clusters.

3) Directivity Analysis: When an unclassified point which is assigned to the function *ExpandCluster()* and evaluated as a core point(#1-5), *DirectivityAnalysis()* function begins. The function is to analyze whether the seed points have the sole distinct direction. For achieve this goal, we utilize linear-RANSAC to evaluate the direction of the seed point set. If the iteration during RANSAC among a seed set converges at predefined occurrence (e.g., 5) or the inlier ratio meets a criterion (e.g., 0.87), the set is considered as unidirectional and linearly fitted value is assigned to $direct$. Otherwise, it returns omnidirectional decision and *ExpandCluster()* session is terminated and leave the core point unclassified. Otherwise, the seed set is assigned as one distinct D-DBSCAN cluster and stretch the cluster region by examining the eps-neigbhorhood of each seed point and orientation similarity afterwards.

D. Line-fittings of Line-segment-clusters

Although line-segments are individually clustered with D-DBSCAN, these are only a set of feature points. For deriving more meaningful information such as line information, individual line-segment is fitted with the RANSAC based least mean square error fitting. By the Sequential-RANSAC [16], the orientation and intercept of line-segments are collected.

III. Slot Pattern Recognition and Detection

Since the previously extracted line-segment itself dose not contain any parking-slot information, meaningful information such as target slot entrance, orientation and pattern should be recongnized. Normally, parking slots have diverse patterns in order to fit into the limited space in many places. Therefore, a prior knowledge of the target slot pattern is not mandatory but considerably efficient for vehicle maneuvering and facilitating slot detection.

This section presents a slot pattern recognition and detection method using several geometric cues of slot. It is comprised of slot-group partition, classification of slot patterns and slot detection. We assume that all slots linked together have the similar slot patterns by our empirical knowledge.

A. Pre-Processing for Slot Pattern Recognition

Diverse slot-groups having different patterns can be positioned in a single AVM image because the AVM searches every direction. Concerning this matter, we divide each slot-group based on distance and position of each detected line.

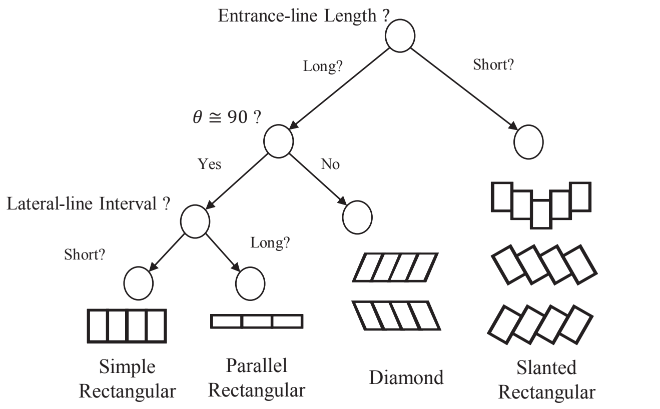


Fig. 3. Decision-Tree Classifier

Before division of each slot-group, an undirected graph $G = (V,E)$ is constructed to evaluate relation among parking-lines. The tip of the lines T and crossing points of lines C are belong to vertices V, $\{T,C\} \subset V$, whereas distance D between vertices belongs to edge E, $\{D\} \subset E$. Using the graph, lines are decomposed into each slot-group node-set on a close distance basis.

When a slot group is partitioned, the slot recognition procedure is performed within each slot group. Within the slot group, lines are categorized into major orientation at most two. Other lines, not belonging to the major orientation, are considered as outliers and discarded. Categorized major lines are discriminated between entrance and lateral lines. In order to discriminate a line state $\mathbf{x} = \{lateral, entrance\}$, the numbers of each line-category N, their average length L, and location of the slot-group P are the subset of observation, $\mathbf{z} = \{N, L, P\}$. With a maximum likelihood function $argmax\ P(\mathbf{z}|\mathbf{x}) = P(\mathbf{x}|\mathbf{z})P(\mathbf{z})$, the categorized lines is assigned to lateral or entrance lines state.

B. Slot Pattern Classifier

For slot pattern recognition, we adapt a decision-tree classifier to elicit specific slot pattern information. It is a commonly used method in data mining. The goal is to create a slot pattern classification model that predicts the classification labels based on several input variables. In the tree structures, leaves represent class labels and branches imply conjunctions of features that lead to those class labels. Fig. 3 describes overall structures of the decision-tree classifier for the proposed model.

The classifier targets the various slot-type among simple rectangular, diamond, parallel and slanted rectangular types, which are designated as the class labels. These notations of the slot pattern types are following the study [9]. On the other hand, the utilized features for classifying specific patterns are comprised of entrance line's length, lateral line's average interval and junction angles. First, entrance line's length discriminates slanted rectangular type. The slanted rectangular type contains multiple number of entrance lines with short lines compared to others containing a single long entrance lines. Regarding the slanted rectangular type, no further categorization is required. After that, lateral lines' average length and junction angle features are utilized to distinguish among simple rectangular, diamond and parallel rectangular type. In order to select the best attributes for classifying training dataset, we adapt Gini-index [17]. As a result shown in fig. 3, each attribute is partitioned into two lists and determines the slot pattern properly. Even though a pruning phase inspects imperfection and prevents the over-fitting, it is not applied due to conciseness of the model.

C. Parking-slot Detection

When the slot pattern recognition step ends, slot pattern, junctions, entrance and lateral lines are attained as by-products. We utilize these information to detect parking-slots. Among them, slot junctions are worked as the most important clue for detecting slots. Junction-set are tracked using the extracted transformation matrix. Based on the transit in matched lines and junctions of consecutive frames, we can derive rotation θ and translation $\mathbf{t}$. Since an AVM image is assumed to represent the plat ground plane with correct calibration, distortions from vehicle's roll or pitch movements are ignored. After deriving the transformation matrix, $\mathbf{x}_{i+1}$ is estimated as follows:

$$\mathbf{x}_{i+1} = \mathbf{R}\mathbf{x}_i + \mathbf{t} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}. \quad (2)$$

By estimating $\mathbf{x}_{i+1}$, the junction positions are preserved even on certain missed junctions due to occlusions or faint lines.

In this tracked junction-set $\mathbf{J}$, every two-junction pair $j_a \in \mathbf{J}, j_b \in \mathbf{J}$ is evaluated whether they compose a slot entrance. There are two conditions to determine a slot entrance. First of all, the orientation between junction pair is considered (eq. 3). The sum of the pair is likely to be around π. If the first criterion is met, the distance between junctions should be within the predefined width range corresponding to the recognized slot pattern. Following equations explain above:

$$|\theta_{j_a} - \theta_{j_b}| < Thres., \ when \ \theta_{j_x} = MIN(\theta_{j_x}, \pi - \theta_{j_x}), \quad (3)$$
$$W_{recog.pattern} - M < |j_a - j_b| < W_{recog.pattern} + M. \quad (4)$$

$W_{recog.pattern}$ represents the entrance width of the recognized slot pattern and M indicates corresponding length margins. When these two conditions are met, the junction-pair is considered as a parking slot.

IV. Experiments and Evaluation

A. Experimental Set-up

The experiments were conducted with an autonomous vehicle platform and the cameras are mounted around the vehicle to form an AVM image. An AVM image sequence is directly obtained in 50 fps with an ECU board for the process. The algorithm is implemented by C++/OpenCV and runs at a frame rate of 31.7 fps on a normal PC(Intel Core i5 Q720 CPU @ 2.2GHz and 8.00GB RAM) with an image size of 480x720. This includes all procedures except for the image acquisition. To achieve effective line-level clustering using D-DBSCAN, applied parameters are as follows: eps = 29.3 px, MinPts = 6 EA, orientation similarity bound = 1.6°, when marking features are extracted per 3 px.

We have acquired dataset containing diverse parking patterns and environments. Simple and slanted rectangular and diamond are included. As a dataset, 2480 frames used for the recognition training set and 1930 frames are tested for parking slot detection evaluation. We adapt two classic measures for performance evaluation:

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}, \quad (5)$$
$$Recall = \frac{TruePositive}{TruePositive + TrueNegative}. \quad (6)$$

B. Evaluation

The contributions of this study are illuminated by the evaluation of two factors: performance comparison between feature detectors and parking-slot detection performance. First, we compare the proposed D-DBSCAN algorithm to probabilistic HT and fast corner detectors. Since the purpose of D-DBSCAN is totally different from original DBSCAN, the direct comparison between them is unachievable. Comparison to a line detector, probabilistic HT, is more appropriate to show the D-DBSCAN's performance instead. For further assessment, we compared a fast corner detector. In this case, the extracted crossing points from D-DBSCAN line-segments are evaluated as corners. Targets consist of 2338 lines and 2018 corners from considerably warped unflat region dataset. Table II shows that the proposed algorithm is robust on detecting short or distorted line-segments. In addition, line-detection-based algorithm has higher performance even in a single frame. Corner detector based algorithms are fragile on the occluded or far-reached faint corners.

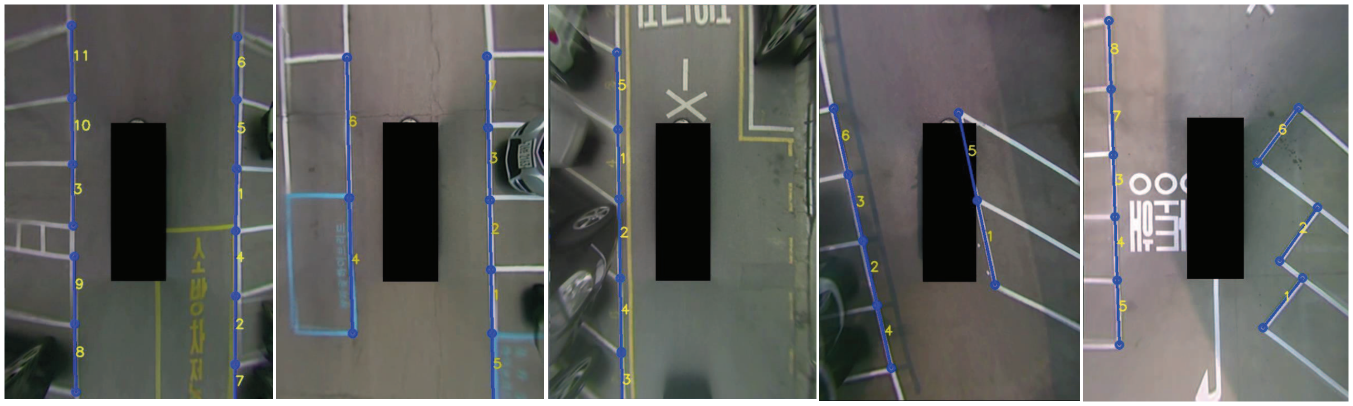
Moreover, we evaluate parking-slot detection performance with varying scenarios such as sunny, sunset and rainy with diverse roads. We regard the ground truth of the slot when two junctions are visible. Fig. 4 and Table III present performance of parking-slot detection results on different slot patterns and environments respectively. Even with intermittent light-reflections in rainy condition and reflective road, the proposed algorithm sustains decent performances.

TABLE II. Performance comparison among different detectors.

	Line-based		Corner-based	
	Prob. HT[3]	Ours	Fast[5]	Ours
Recall	0.872	0.941	0.826	0.933
Precision	0.926	0.912	0.937	0.962

TABLE III. Slot-detection performances on diverse environments.

Scenarios	# of test slots	Recall	Precision
Sunny	5194	0.972	0.979
Sunset	1043	0.911	0.981
Rainy	816	0.904	0.933
Extraordinary Road	1210	0.938	0.977
Total	7790	0.951	0.979


(a) Diverse types of slots(simple rectangular, parallel, diamond and slanted rectangular)

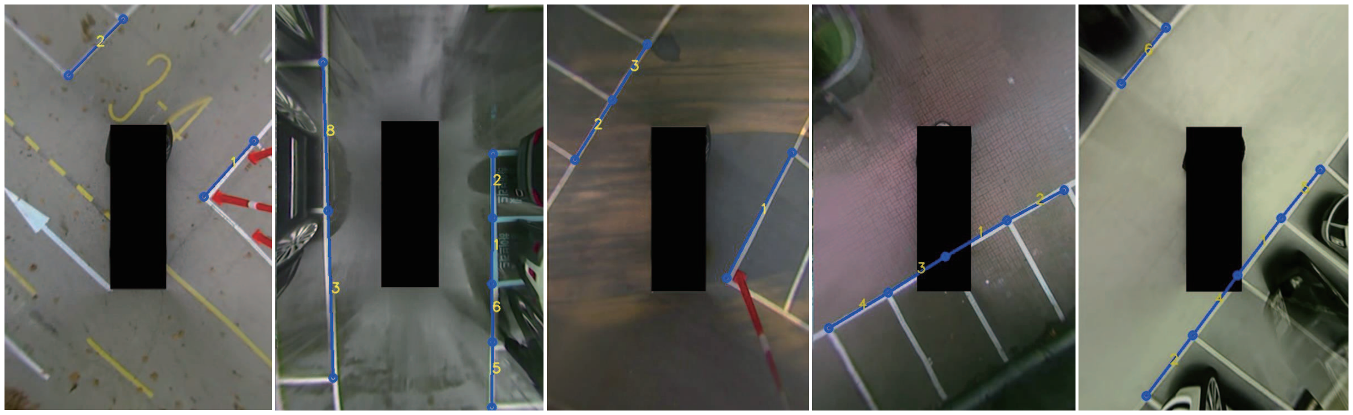

(b) Diverse scenarios of environments (sunny, rainy, sunset, brick road and reflective road)

Fig. 4. Snap-shots of parking-slot detection.

V. Conclusion

This paper proposes a parking-slot detection algorithm based on D-DBSCAN algorithm for line-segment detection. Through quantitative analysis, we evaluate the D-DBSCAN algorithm outperforms on detecting blurry and short line-segments in AVM images. Considering that most of the AVM images contain multiple short and blurry parking-lines on the corner spots frequently, our algorithm shows more robustness on AVM-based parking assistant systems. We plan to broaden our study to AVM-based Parking-map building and vehicle localization accordingly.

References

[1] M.Ester *et al*, " A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," Conf. on Knowledge Discovery and Data Mining, pp. 226-231, Portland, 1996.

[2] Jung, H.G., Kim, D.S., Yoon, P.J., Kim, J., "Parking slot markings recognition for automatic parking assist system", IEEE Intelligent Vehicles Symposium, pp. 106-113, 2006.

[3] Hamada, K., Hu, Z., Fan, M. and Chen, H., "Surround View based Parking Lot Detection and Tracking," in IEEE Intelligent Vehicles Symposium, South Korea, 2015.

[4] C.Wang and M.Yang et al. "Automatic parking based on a bird's eye view vision system," Adv. in Mec. Eng., vol 2014, 2014, pp 1-13.

[5] X. Sevillano et al, "Towards smart traffic management systems: Vacant on-street parking spot detection based on video analytics," 17th International Conference on Information Fusion, Salamanca, July 2014

[6] Suhr, J.K., Jung, H. G., "Fully-automatic recognition of various parking slot markings in Around View Monitor (AVM) image sequences," IEEE Conference on Intelligent Transportation Systems, Alaska, 2012.

[7] Suhr, J. K, Jung, H. G., "Full-automatic recognition of various parking slot markings using a hierarchical tree structure," Optical Engineering 52(3),037203, 2013.

[8] Choi, J., Chang, E., Yoon, D., Ryu, S., et al, "Sensor Fusion-Based Parking Assist System," SAE Technical Paper 2014-01-0327, 2014.

[9] Suhr, J.K. Jung, H.G., "Sensor Fusion-based Vacant Parking Slot Detection and Tracking," Transactions on Vehicular Technology, 15(1), 21-36, Feb. 2014.

[10] Jung, H.G., Lee, Y.H., "Uniform User Interface for Semi-automatic Parking Slot Markings Recognition," Transactions on Vehicular Technology, 59(2), 616-626, Feb. 2010.

[11] P. Degerman *et al*, "Hough transform for parking space estimation using long range ultrasonic sensors," SAE Technical Paper, 2006.

[12] H.G Jung, Y. H. Cho, P. Yoon, J. Kim "Scanning Laser Radar-Based Target Position Designation for Parking Aid System," IEEE Transactions on Intelligent Transportation Systems, Vol. 9, No. 3, 2008

[13] Suhr, J. K., Jung, H. G., Bae, K., and Kim, J, "Automatic free parking space detection by using motion stereo-based 3D reconstruction," Machine Vision and Applications, 21(2), pp. 163-176. 2011.

[14] D. Llorca, S.Alvarez, "Vision-based parking assistance system for leaving perpendicular and angle parking lots." in IEEE Intelligent Vehicles Symposium, Austrailia, 2013

[15] M. Nieto and L. Salgado, "Robust multiple lane road modeling based on perspective analysis," *Image Processing*, ICIP, San Diego, 2008.

[16] S. Lee, S. Kim and S. Seo, "Accurate Ego-Lane Recognition utilizing Multiple Road Characteristics in a Bayesian Network Framework," IEEE Intelligent Vehicles Symposium, South Korea, 2015.

[17] S. Ranka *et al*, "CLOUDS: A decision tree classifier for large datasets," Conf. on Knowledge Discovery and Data Mining, pp. 1-8, 1998.

[18] K. Sung *et al*, "Development of Image Synthesis Algorithm with Multi-Camera," IEEE Conf. on Vehicular Technology Conference, Yokohama, 2012.