Vision-Based Parking-Slot Detection: A DCNN-Based Approach and a Large-Scale Benchmark Dataset

Lin Zhang, *Senior Member, IEEE*, Junhao Huang, Xiyuan Li, and Lu Xiong

***Abstract*— In the automobile industry, recent years have witnessed a growing interest in developing self-parking systems. For such systems, how to accurately and efficiently detect and localize the parking slots defined by regular line segments near the vehicle is a key and still unresolved issue. In fact, kinds of unfavorable factors, such as the diversity of ground materials, changes in illumination conditions, and unpredictable shadows caused by nearby trees, make the vision-based parking-slot detection much harder than it looks. In this paper, we attempt to solve this issue to some extent and our contributions are twofold. First, we propose a novel deep convolutional neural network (DCNN)-based parking-slot detection approach, namely, DeepPS, which takes the surround-view image as the input. There are two key steps in DeepPS, identifying all the marking points on the input image and classifying local image patterns formed by pairs of marking points. We formulate both of them as learning problems, which can be solved naturally by modern DCNN models. Second, to facilitate the study of vision-based parking-slot detection, a large-scale labeled dataset is established. This dataset is the largest in this field, comprising 12 165 surround-view images collected from typical indoor and outdoor parking sites. For each image, the marking points and parking slots are carefully labeled. The efficacy and efficiency of DeepPS have been corroborated on our collected dataset. To make our results fully reproducible, all the relevant source codes and the dataset have been made publicly available at https://cslinzhang.github.io/deepps/.**

***Index Terms*— Self-parking systems, parking-slot detection, deep convolutional neural networks.**

I. Introduction

FOR many drivers, especially the novices, finding and navigating a vehicle into a suitable parking-spot is a great challenge [1]. The main reason is that the driver cannot see around the vehicle or determine the size and shape of the parking space. This has been leading many research institutions and vehicle manufacturers to devote great efforts on the development of self-parking systems. Actually, a self-parking system can be regarded as a special type of unmanned driving system. The workflow of a typical self-parking system is roughly as follows. When approaching the parking area, the vehicle switches to the low-speed unmanned mode and automatically travels along a predetermined track. When working in unmanned mode, the vehicle may need to rely on high-definition maps, GPS signals, or SLAM (Simultaneous Localization and Mapping) [2] technology for self-positioning. During traveling, the vehicle searches for available parking-slots around, or attempts to identify and locate the parking-slot assigned to it by the parking-slot management system. Once an appropriate parking-slot is detected and positioned, the vehicle will switch to the automatic parking mode, plan the parking path, and eventually park the vehicle to the designated parking-slot.

There are many key issues that need to be addressed when building a self-parking system. How to quickly and accurately detect and locate the parking-slots around the vehicle is just one of them.

The remainder of this paper is organized as follows. Section II introduces the related work and our contributions. Section III presents our DCNN-based parking-slot detection approach, DeepPS. Experimental results are presented in Section IV. Finally, Section V concludes the paper.

II. Related Work and Our Contributions

A. Vision-Based Parking-Slot Detection: Algorithms and Datasets

The methods for perceiving available parking spaces during vehicle travel can be categorized into two groups, the free-space-based ones and the vision-based ones. A free-space-based approach designates a target parking position by recognizing an appropriate vacant space between adjacent vehicles. This is the most widely used approach as it can be implemented using various range-finding sensors, such as ultrasonic sensors [3]–[8], laser scanners [9]–[11], short-range radars [12]–[14], structured light [15], depth cameras [16], stereo cameras [17]–[22]. The free-space-based approach has an inherent drawback that it must rely on vehicles that have already been properly parked as a reference. In other words, this kind of approaches cannot work in an open area with no

Manuscript received September 6, 2017; revised February 25, 2018 and June 16, 2018; accepted July 15, 2018. Date of publication July 18, 2018; date of current version August 14, 2018. This work was supported in part by the Natural Science Foundation of China under Grant 61672380, in part by the Fundamental Research Funds for the Central Universities under Grant 2100219068, and in part by the Shanghai Automotive Industry Science and Technology Development Foundation under Grant 1712. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Jing-Ming Guo. *(Corresponding author: Lin Zhang.)*

L. Zhang is with the School of Software Engineering, Tongji University, Shanghai 201804, China, and also with the Institute of Intelligent Vehicle, Tongji University, Shanghai 201804, China (e-mail: cslinzhang@tongji.edu.cn).

J. Huang and X. Li are with the School of Software Engineering, Tongji University, Shanghai 201804, China (e-mail: 1731557@tongji.edu.cn; 1641465@tongji.edu.cn).

L. Xiong is with the Institute of Intelligent Vehicle, Tongji University, Shanghai 201804, China (e-mail: xiong_lu@tongji.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2018.2857407

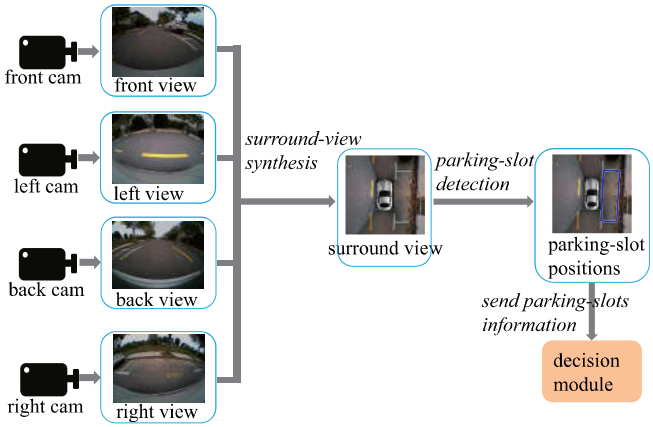


Fig. 1. The high-level structure of a typical vision-based parking-slot detection system. It usually comprises two modules, surround-view synthesis and parking-slot detection from the surround-view. When a parking-slot is detected, its position with respect to the vehicle-centered coordinate system will be passed to the decision module.

vehicles around. In addition, its accuracy highly depends on the positions and poses of adjacent vehicles.

The working principle of a vision-based approach is fundamentally different from that of a method based on free-space. The goal of a vision-based approach is to identify and locate the parking-slots defined by parking line segments painted on the ground. Apparently, the performance of such approaches does not depend on the existence or poses of adjacent vehicles. Moreover, in most cases, parking line segments can provide more accurate parking information than "free-space." Meanwhile, most car manufacturers have started to produce vehicles equipped with wide FOV (field of view) imaging sensors, usually used in an AVM (around view monitoring) system. For these reasons, the vision-based approach has begun to draw a lot of attention recently in the field of parking spot detection, which is also our focus in this paper.

Fig. 1 shows the high-level structure of a typical vision-based parking-slot detection system, which usually comprises two independent modules, surround-view synthesis and parking-slot detection from the surround-view. The surround-view camera system normally consists of four to six wide-angle cameras mounted around the vehicle, each facing a different direction. Based on these wide-angle images, the surround-view synthesis module can generate a 360° surround-view image around the vehicle. In the literature, there are already mature solutions for calibrating wide-angle cameras [23], [24] and constructing a surround-view system [25], [26]. The parking-slot detection module takes the surround-view image as the input, detects the parking-slots, and finally sends their physical positions with respect to the vehicle-centered coordinate system to the decision module for further process. Representative work on vision-based parking-slot detection will be reviewed as follows.

The research in this area begins with Xu *et al.*'s pioneer work [27]. Xu *et al.* [27] claimed that the colors of parking lines are quite uniform and different from the background and thus they trained a neural network to segment parking lines. Then, they estimated two perpendicular lines as the parking-slot contour. The drawback of this simple model is that the type (perpendicular or parallel) of the parking-slot cannot be obtained and it cannot deal with slanted parking-slots either. Jung *et al.* presented a one-touch method that recognizes the line segments of a parking-slot by checking the directional gradient based on a manually provided point inside the target parking-slot. In view of this method can only handle a single type of parking-slot, Jung *et al.* [29] extended it to a two-touch method. This method can recognize various types of parking-slot-markings based on two points with respect to a parking-slot entrance-line provided by the driver. Du and Tan [30] developed a reverse parking system. In order to detect the parking-slot, a ridge detector is applied on the image first and then medial axes of the slot lines are obtained after the steps of noise filtering, connected components labeling, and removal of components with a small number of pixels. However, their system relies on human drivers to identify an empty parking-slot first before initiating the parking process. To sum up, the apparent shortcoming of the methods proposed in [28]–[30] is that they are not fully automated, hereby limiting their usability in practice.

Fully automated approaches are developed along two main streams, the line-based ones and the corner-based ones, according to the primitive visual features they extract. Jung *et al.* [31] assumed that parking-lines consist of lines with a fixed width and recognized them by applying peak-pair detection and clustering in Hough space [32]. Separating parking-line segments were at last recognized by a T-shaped template matching. Based on similar ideas as Jung *et al*'s work [31], Wang *et al.* [25] proposed to detect parking-line segments in Radon space [33] as they considered that Radon transform has a better noise-tolerance ability and is more robust than Hough transform. One potential drawback of the methods in [25] and [31] is their sensitiveness to the parking-line width. After obtaining the edge map of the surround-view image, Hamada *et al.* [34] made use of the probabilistic Hough transform [35] to extract all line segments and then they inferred the valid parking-slots based on some geometric constraints. Suhr and Jung [36] designed a parking-slot detection approach specially for underground and indoor environments. In their approach, the guide line is detected first and then the separating lines are detected. To detect the guide line, they utilized the RANSAC (RANdom SAmpling Consensus) [32] algorithm for robust line fitting from edge pixels and to detect the separating lines, they used the distance transform based chamfer matching [37]. The main limitations of Suhr and Jung's method are twofold: 1) it can only detect perpendicular parking-slots but cannot detect parallel ones; 2) it requires that the guide line of the parking-slots should be visible. Lee and Seo [38] proposed the so-called "cone-hat" filter for line-marking extraction and then the extracted line features were assigned to parking-line segments via entropy-based clustering. After that, the sequential RANSAC algorithm [39] was utilized to fit parking-lines from clusters. At last, parking-slot candidates were generated and then validated by a Bayesian network model [40].

Different from line-based ones, a few other parking-slot detection methods are based on corners and among them Suhr and Jung's work is a representative one [41], [42]. The method proposed in [41] and [42] detects corners via the Harris corner detector [43] first and then generates junctions by combining these corners; finally, parking-slots are inferred from junction pairs. Thus, the success rate of this method highly depends on the robustness of the Harris corner detector.

Quite recently, in [26], we introduced the machine learning theory into the field of parking-slot detection. Specifically, based on boosting decision trees, we trained a marking-point detector. When the marking-points are available, valid parking-slots are inferred through some predefined geometric rules.

As a commonsense, to design and validate parking-slot detection algorithms, a large-scale public benchmark dataset is indispensable. Unfortunately, the vast majority of researchers in this field did not publish the datasets they collected, which undoubtedly hinders the development of this field. To the best of our knowledge, the dataset established in [26] is the only public one in this field, which comprises 8,600 labeled surround-view images.

B. Deep Convolutional Neural Networks

As our proposed parking-slot detection approach, DeepPS (Refer to Sect. III for details), is based on deep convolutional neural networks (DCNN), we give a brief review about the development of DCNN in this section.

As we all know, we are witnessing a rapid, revolutionary change in the computer vision community, mainly caused by DCNN. DCNN is actually a kind of representation learning methods that allow a machine to be fed with raw data and to automatically discover the representations needed for classification or detection [44]. Recently, DCNN has pulled away from competing methods due to the availability of larger labeled datasets, better models and training algorithms, and the availability of GPU computing to enable investigation of larger and deeper models. The development of DCNN traces back to the late 1980s [45] and since 2012, more powerful and popular DCNN architectures have been proposed in the literature, such as AlexNet [46], GoogLeNet [47], VGG [48], and ResNet [49], just to name a few.

DCNN-based approaches have recently been substantially improving upon the state-of-the-art in several areas, such as image classification, object detection, face recognition, image restoration, etc. Among them, the development of the filed of object detection is quite relevant to our work. Applying DCNN in solving the task of object detection begins with Girshick *et al.*'s groundbreaking work, R-CNN [50]. R-CNN is actually a multi-stage detection framework. Given an input image, it first uses an object proposal algorithm to find bounding-boxes that include objects with high probability. Then, a standard DCNN is applied as a feature extractor to each proposed bounding-box and finally a classifier decides the object class inside the box. Following the framework of R-CNN, many researchers proposed their modifications to improve R-CNN's performance and some representative approaches along this direction are Fast-RCNN [51], Faster-RCNN [52], HyperNet [53], etc. R-CNN and all its variants highly depend on object proposals. Hence, the performance of the object proposal algorithm becomes the bottleneck. Quite recently, some researchers have begun to challenge the necessity of an object proposal algorithm in DCNN-based object detection systems and they formulated the object detection as a regression problem to spatially separated bounding-boxes and associated class probabilities. Representatives of such methods include Yolo [54], SSD [55], and YoloV2 [56]. Experimental results from different research groups indicate that this kind of methods can get bounding-boxes as accurate as R-CNN like methods but run much faster. As a result, in our parking-slot detection approach, DeepPS, YoloV2 is adopted for detecting marking-points (Refer to Sect. III-A for details).

C. Our Motivations and Contributions

Through the literature survey, we find that in the field of vision-based parking-slot detection, we need to continue to devote efforts in at least two aspects.

First, the performance of parking-slot detection algorithms still needs to be further improved. In fact, vision-based parking-slot detection is a mission full of challenge. Kinds of adverse factors make it more formidable than it looks. In Fig. 2, eight typical surround-view images containing parking-slots collected by us are shown, from which it can be seen that the visual context of parking-slots varies greatly. The causes that can be attributed to this diversity may include diverse ground materials, various parking-slot types (perpendicular, parallel, or slanted), different parking-line colors, incompleteness of the parking-lines, changes in illumination conditions, shadows caused by nearby trees or buildings, etc. Existing solutions are either based on low-level features (e.g., edges, lines, or corners) or are based on elementary machine learning theories (e.g., AdaBoost used in [26]). Under simple and ideal conditions, they can lead to acceptable performance. However, when the visual scenes become complicated and non-ideal, the performance of these methods is often unsatisfactory due to their inherent limitations. Therefore, how to design a parking-slot detection algorithm that can cope with practical complex conditions and also can achieve a high precision-recall rate is still a challenging task.

Second, large-scale public datasets are quite rare in this field. To design parking-slot detection algorithms and also to objectively compare their performance, a public large-scale benchmark dataset covering a variety of real scenarios is indispensable. Unfortunately, thus far the only publicly available dataset in this field is the one established in [26].

In this work, we attempt to fill the aforementioned research gaps to some extent and our major contributions are summarized as follows.

(1) Upon seeing that DCNN has achieved great successes in various vision-related fields, we attempt to devise a DCNN-based parking-slot detection approach and we name it as DeepPS (short for "deep parking-slot"). Given a surround-view image, DeepPS first detects all the

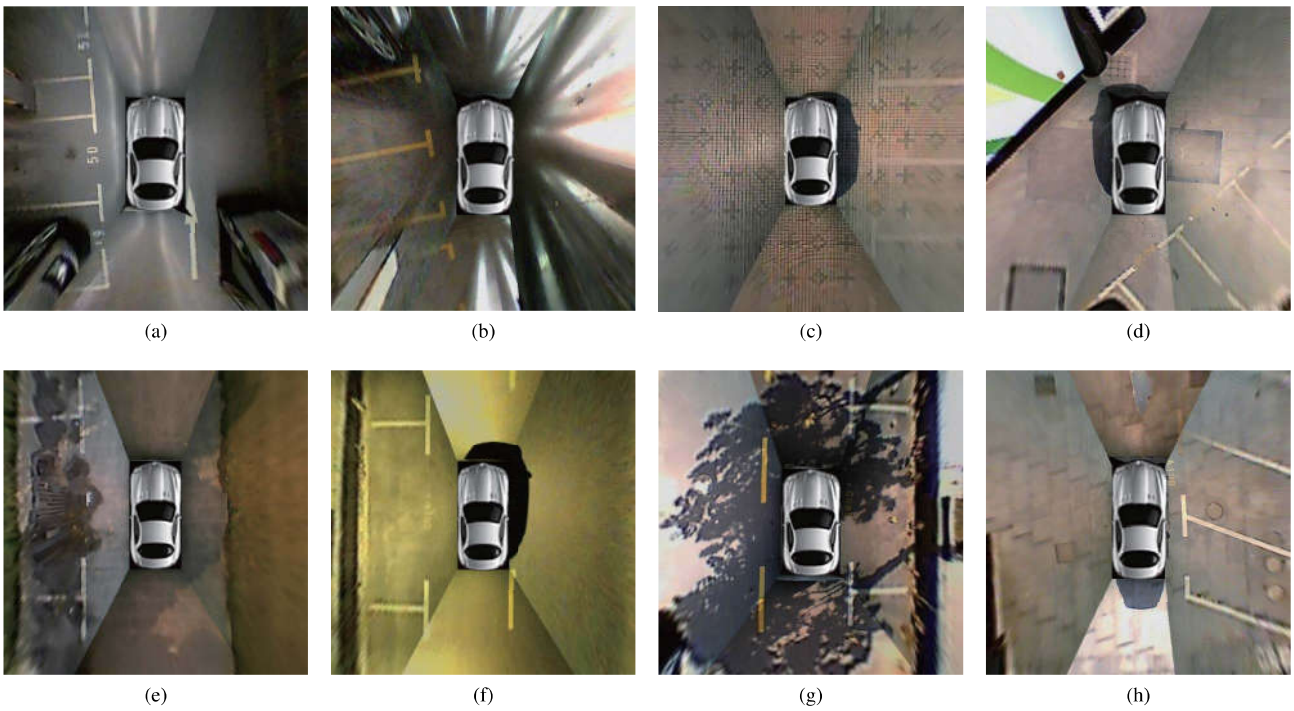


Fig. 2. Eight typical surround-view images are shown here to demonstrate that the visual patterns of parking-slots can vary much in real cases. The parking-slots in (a), (b), (c), and (d) are "perpendicular," the ones in (e), (f), and (g) are "parallel," and the ones in (h) are "slanted." (a) and (b) were taken from indoor parking sites while the others were taken from outdoor parking sites. Parking lines in (b) are yellow while the ones in the other images are white. (e) is taken on a rainy day. (f) is taken under the street light at night. In (g), the strong shadow caused by nearby trees covers the parking lines. Observable damages to parking lines exist in (d) and (e). Ground materials are different from each other in these images.

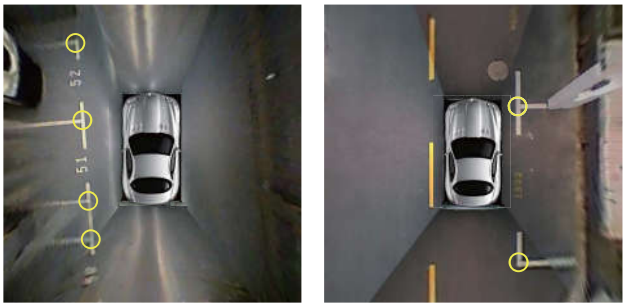


Fig. 3. (a) and (b) are two surround-view images with marking-points marked by yellow circles.

marking-points on it using a pre-trained marking-point detector based on YoloV2 [56]. A marking-point pattern refers to a local image patch centered at a cross-point of two parking-line segments. Examples of marking-points marked by yellow circles are shown in Fig. 3. Given a pair of detected marking-points $\mathbf{p}_1$ and $\mathbf{p}_2$, DeepPS will then decide whether they can form a valid entrance-line and if "yes," DeepPS will also need to decide the type of this parking-slot. This task can be fulfilled via classifying the local image pattern defined by $\mathbf{p}_1$ and $\mathbf{p}_2$ using a standard pre-trained DCNN model. To our knowledge, our work is the first to apply deep learning techniques to parking-slot detection. DeepPS can deal with nearly all the commonly seen types of parking-slots. Its performance has been intensively evaluated on our established dataset. Actually, DeepPS has been equipped on SAIC Roewe E50 electric cars [57].

(2) To facilitate the study of vision-based parking-slot detection, we extend the dataset established in [26] to an even larger one. The new dataset comprises 12,165 surround-view images, covering a wide variety of real cases, and all the images are manually labeled with care. This dataset can serve as a benchmark and be employed for training and validating new parking-slot detection algorithms. Please refer to Sect. IV-A for more details about this dataset.

The differences between the current work and our previous work in [26] are summarized as follows. In [26], we trained a marking-point detector based on ACF (Aggregate Channel Features) + Boosting Decision Trees. When the marking-points are ready, valid parking-slots are inferred through some predefined geometric rules. In this work, we formulate the parking-slot detection problem as two sub-problems (marking-point detection and local image pattern classification), both of which can be efficiently solved by DCNNs. In addition, a larger benchmark dataset is established in this work.

To make the results reported in this paper fully reproducible, the collected dataset and all the relevant source codes are publicly available at https://cslinzhang.github.io/deepps/.

III. DeepPS: A DCNN-Based Approach

In this section, our proposed parking-slot detection approach DeepPS will be presented in detail. DeepPS takes a

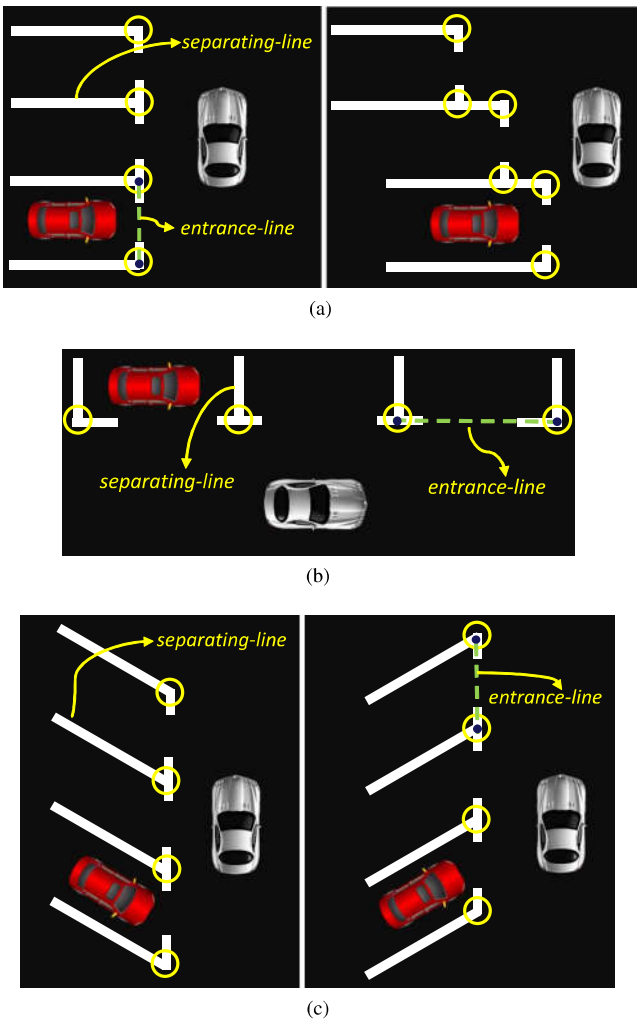


Fig. 4. Ideal parking-slot models that DeepPS can deal with. (a), (b), and (c) show ideal models for perpendicular, parallel, and slanted parking-slots, respectively. Marking-points are marked with yellow circles. Examples of the entrance-lines and the separating-lines are also marked out.

surround-view image as the input and can deal with almost all the commonly seen types of parking-slots composed of "T-shaped" or "L-shaped" marking-points. In Fig. 4, ideal parking-slot models that DeepPS can cope with are shown. Fig. 4(a), 4(b), and 4(c) show ideal models for perpendicular, parallel, and slanted parking-slots, respectively. In Fig. 4, all the marking-points are marked with yellow circles. Besides, examples of the entrance-line (a virtual line linking the two marking-points of a valid parking-slot) and examples of the separating-line (a parking-line separating two adjacent parking-slots) are also marked out.

To detect parking-slots, three major steps are taken by DeepPS, including marking-point detection, local image pattern classification, and parking-slot inference. Details are introduced in the following subsections.

A. Marking-Point Detection

At the testing stage, given a surround-view image, DeepPS first detects from it all the marking-points. For this purpose, we need to train a marking-point detector $\mathcal{D}$ offline beforehand. Through the literature survey, we find that YoloV2 [56] is a state-of-the-art general-purpose object detector based on DCNN. It can achieve quite high detection accuracy and also can run extremely fast. In addition, the YoloV2 framework is straightforward to configure and extend. Hence, our marking-point detector is based on YoloV2.

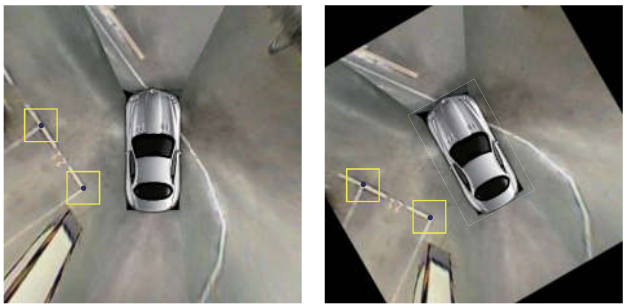


Fig. 5. To make the marking-point detector rotation invariant, when preparing training samples, each original labeled image was rotated to generate a set of its rotated versions. (a) is an original labeled image and (b) is generated by rotating (a) 30 degrees. Marking-points are indicated as purple points and the associated bounding-boxes are shown as yellow squares.

To train the detector, we need to prepare training samples. At the training sample preparation stage, on a given surround-view image, the positions of all its marking-points were manually marked. For each marking-point $\mathbf{p}_i$, a square box of the fixed size $p \times p$ centered on $\mathbf{p}_i$ is regarded as the ground-truth bounding-box of $\mathbf{p}_i$.

It is highly desired that the trained marking-point detector could have the property of rotation invariance. To achieve this goal, we augmented the training set by rotating each original labeled image to generate a number of its rotated versions. Specifically, from each original labeled image $\mathbf{I}$, we could obtain its J rotated versions $\{\mathbf{I}_j\}_{j=0}^{J-1}$, where $\mathbf{I}_j$ is generated by rotating $\mathbf{I}$ with $\frac{360}{J} \cdot j$ degrees. Of course, the labeled coordinates of marking-points were rotated in the same way. Such an idea for data augmentation is illustrated by an example shown in Fig. 5. Fig. 5(a) is an original labeled image while Fig. 5(b) is generated by rotating Fig. 5(a) 30 degrees. Marking-points are indicated as purple points and the associated bounding-boxes are shown as yellow squares.

In implementation, the YoloV2-based marking-point detector $\mathcal{D}$ was fine-tuned from the model trained on VOC dataset, which was provided by the authors of YoloV2. For fine-tuning, the mini-batch size was set to 64; and the learning rate started from 0.0001 and was divided by 10 every 50000 iterations. We used a weight decay of 0.0005 and a momentum of 0.9.

The trained marking-point detector performs quite well at the testing stage. In Sect. IV-C, we will quantitatively evaluate its performance.

B. Local Image Pattern Classification

After applying the marking-point detector $\mathcal{D}$ on the test image, points with confidence scores greater than δ_1 will be considered as marking-points. Suppose that $\mathbf{p}_1$ and $\mathbf{p}_2$

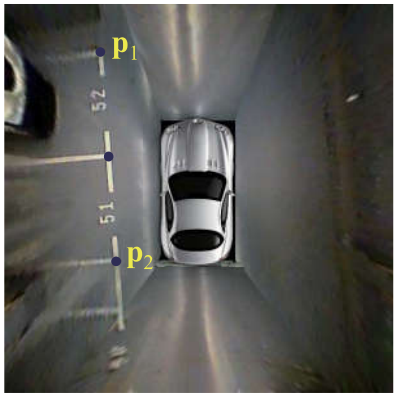


Fig. 6. The distance between $\mathbf{p}_1$ and $\mathbf{p}_2$ satisfies the distance constraint for being an entrance-line of a parallel parking-slot; however, actually, $\mathbf{p}_1$ and $\mathbf{p}_2$ cannot form a valid entrance-line.

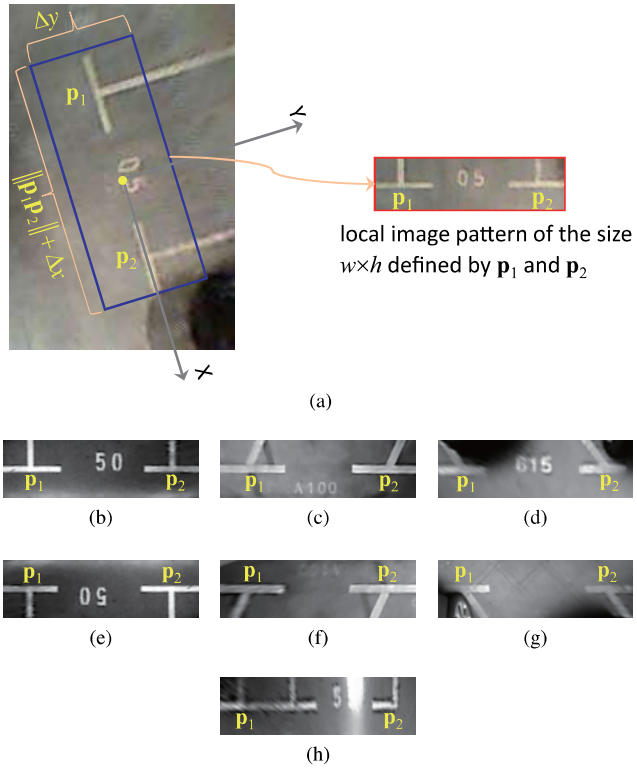


Fig. 7. (a) demonstrates how to extract the local image pattern defined by two marking-points $\mathbf{p}_1$ and $\mathbf{p}_2$. (b) $\sim$ (h) are representative local image patterns belonging to the classes "right-angled anticlockwise," "slanted anticlockwise with an acute parking angle," "slanted anticlockwise with an obtuse parking angle," "right-angled clockwise," "slanted clockwise with an obtuse parking angle," "slanted clockwise with an acute parking angle," and "invalid," respectively.

are two detected marking-points. We need to validate that whether they can form a valid entrance-line. First, if $\overrightarrow{\mathbf{p}_1\mathbf{p}_2}$ can be a valid entrance-line candidate, the distance between $\mathbf{p}_1$ and $\mathbf{p}_2$ should satisfy some constraints. If $\overrightarrow{\mathbf{p}_1\mathbf{p}_2}$ is an entrance-line candidate for a parallel parking-slot, it needs to satisfy $t_1 < \|\mathbf{p}_1\mathbf{p}_2\| < t_2$; if $\overrightarrow{\mathbf{p}_1\mathbf{p}_2}$ is an entrance-line candidate for a perpendicular or a slanted parking-slot, it should satisfy $t_3 < \|\mathbf{p}_1\mathbf{p}_2\| < t_4$. Parameters t_1, t_2, t_3, and t_4 are set based on the priori knowledge about the entrance-line lengths of various types of parking-slots.

Then, we need to further process the marking-point pairs satisfying the distance constraints. First, for a pair of marking-points, although it can satisfy the distance constraints, it is highly possible that they still cannot form a valid entrance-line. For example, in Fig. 6, the distance between $\mathbf{p}_1$ and $\mathbf{p}_2$ satisfies the distance constraint for being an entrance-line of a parallel parking-slot; however, it is obvious that $\overrightarrow{\mathbf{p}_1\mathbf{p}_2}$ is not a valid entrance-line because it passes through another marking-point. In addition, suppose that $\overrightarrow{\mathbf{p}_1\mathbf{p}_2}$ is a valid entrance-line. We need to determine whether the associated parking-slot is on its clockwise side or on its anticlockwise side and to determine whether this parking-slot is right-angled or slanted.[1] All these issues can be solved through classifying the local image pattern defined by $\mathbf{p}_1$ and $\mathbf{p}_2$ into one of the predefined classes.

As illustrated in Fig. 7(a), the local image pattern defined by two marking-points $\mathbf{p}_1$ and $\mathbf{p}_2$[2] on a surround-view image is extracted as follows. At first, a local coordinate system is established, which takes the mid-point of $\mathbf{p}_1$ and $\mathbf{p}_2$ as its origin, and $\overrightarrow{\mathbf{p}_1\mathbf{p}_2}$ as its X-axis. Its Y-axis can be consequently determined. In this coordinate system, we define a rectangular region $\mathbf{R}$, which is symmetric both to the X-axis and the Y-axis. For $\mathbf{R}$, its side length along the X-axis is set as $\|\mathbf{p}_1\mathbf{p}_2\| + \Delta x$ and its side length along the Y-axis is set as Δy. We extract the image region covered by $\mathbf{R}$ from the surround-view image, normalize it to the size $w \times h$, and regard the resultant image patch as the local image pattern defined by $\mathbf{p}_1$ and $\mathbf{p}_2$.

At the training stage, based on label data, we can obtain a set $\mathbb{C}$ comprising all the local image patterns defined by pairs of marking-points. According to the characteristics of the associated parking-slots, we categorize the samples in $\mathbb{C}$ into 7 classes, "right-angled anticlockwise,"[3] "slanted anticlockwise with an acute parking angle,"[4] "slanted anticlockwise with an obtuse parking angle," "right-angled clockwise," "slanted clockwise with an obtuse parking angle," "slanted clockwise with an acute parking angle," and "invalid." Representative exemplars belonging to these classes are shown in Fig. 7(b) $\sim$ (h), respectively. When constructing $\mathbb{C}$, we encountered a practical problem, class imbalance, meaning that a specific class has a very small number of instances relative to other classes. In our case, the root cause of this issue is that in our collected dataset the number of slanted parking-slots is much smaller than that of the right-angled ones. To solve this problem, we adopted SMOTE [58] to over-sample the minority classes.

[1] If a parking-slot is right-angled, its entrance-line and its separating-line yare perpendicular to each other; if it is slanted, its entrance-line and separating-line are not perpendicular.

[2] It is supposed that the distance between these two marking-points satisfies the distance constraints for being a valid entrance-line ycandidate.

[3] It means that the associated parking-slot is right-angled and is on the anticlockwise side of $\overrightarrow{\mathbf{p}_1\mathbf{p}_2}$.

[4] For a slanted parking-slot, the parking-angle is defined as the angle between the entrance-line $\overrightarrow{\mathbf{p}_1\mathbf{p}_2}$ and its separating-line.

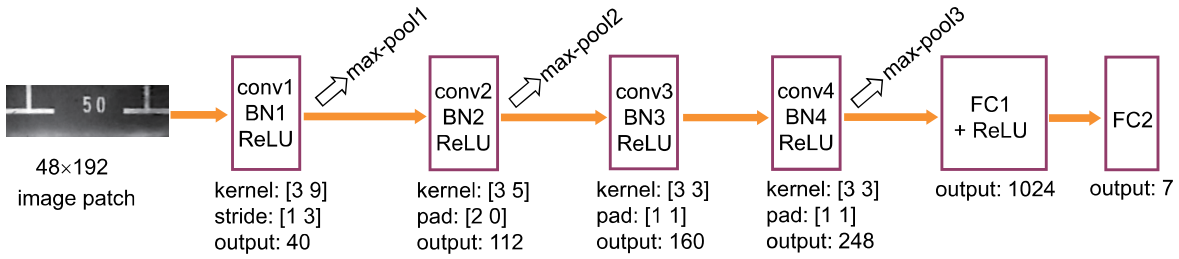


Fig. 8. Our customized DCNN structure for local image pattern classification.

From $\mathbb{C}$, we can train a classification model $\mathcal{M}$ to predict the class label of an unseen local image pattern extracted from a test surround-view image. With respect to the classification model, DCNN is a natural choice. However, based on the following considerations, we do not directly make use of the existing public DCNN architectures (such as AlexNet, GoogLeNet, ResNet, etc.).

1) With the current system configurations, the size of the image samples in $\mathbb{C}$ is 48×192 (more details about parameter settings can be found in Sect. IV-B), much smaller than the required sizes of popular public DCNN architectures.[5] Of course, we can upsample images in $\mathbb{C}$ to make them meet the requirements of public DCNN models; however, obviously, that is not the most efficient way.
2) Nearly all the popular public DCNN structures are designed for the image classification task of ImageNet [59]. The ImageNet classification task has 1000 classes while our classification task has only 7 classes. Moreover, the content complexity of local image patterns in $\mathbb{C}$ is much lower than the general images in ImageNet. For these reasons, we think that a small-scale network can be qualified for our task. Compared with the popular large-scale networks, a small-scale network has more advantages in terms of computing speed, storage space requirement, and so on.

Therefore, taking AlexNet [46] as a template, a customized DCNN architecture is designed to solve our classification task. As shown in Fig. 8, our network takes a grayscale 48×192 image as the input and the output layer has 7 nodes, corresponding to the 7 classes of local image patterns. In Fig. 8, "conv" means it is a convolution layer, "ReLU" means it is a rectified linear unit [60] layer, "max-pool" means it is a max pooling layer, "BN" means it is a batch normalization layer, and "FC" means it is a fully connection layer. For each "conv," "max-pool," and "FC" layer, its parameter settings and the dimension of its output (feature map of this layer) are presented in Table I. "kernel: [kernel_h, kernel_w]" specifies height and width of each filter, "pad: [pad_h, pad_w]" specifies the number of pixels to add to each side of the input, "stride: [stride_h, stride_w]" specifies the intervals at which to apply the filters to the input, and "num_output" specifies the number of filters.

[5]Most modern public DCNN models require that the input image should be of the size $227 \times 227 \times 3$, or $224 \times 224 \times 3$.

TABLE I
PARAMETER SETTINGS AND THE OUTPUT DIMENSION FOR EACH LAYER OF OUR DCNN DESIGNED FOR LOCAL IMAGE PATTERN CLASSIFICATION

layer name	parameter settings	dimension of output
input		$48 \times 192 \times 1$
conv1	kernel: [3 9] stride: [1 3] num_output: 40	$46 \times 62 \times 40$
max-pool1	kernel: [2 3] stride: 2	$23 \times 31 \times 40$
conv2	kernel: [3 5] pad: [2 0] num_output: 112	$25 \times 27 \times 112$
max-pool2	kernel: [3 3] pad: [1 0] stride: 2	$13 \times 13 \times 112$
conv3	kernel: [3 3] pad: [1 1] num_output: 160	$13 \times 13 \times 160$
conv4	kernel: [3 3] pad: [1 1] num_output: 248	$13 \times 13 \times 248$
max-pool3	kernel: [3 3] stride: 2	$6 \times 6 \times 248$
FC1	num_output: 1024	1024
FC2	num_output: 7	7

The customized DCNN model was firstly trained on ImageNet 2012 classification dataset and then was fine-tuned for our specific task. For fine-tuning, the mini-batch size was set to 256; and the learning rate started from 0.002 and was divided by 2 every 5000 iterations. We used a weight decay of 0.0005 and a momentum of 0.9.

C. Parking-Slot Inference

In self-parking systems, a parking-slot is generally considered to be a parallelogram and is usually represented by the coordinates of its four vertices. In most cases, the two non-marking-point vertices are not visible and their coordinates can be obtained only through inference. To this end, we need to assume that the "depth" of the parking-slots is known beforehand as a priori knowledge. As illustrated in Fig. 9, the depth of the perpendicular, parallel, and slanted parking-slots is assumed to be d_1, d_2, and d_3, respectively.

Suppose that $\mathbf{p}_1$ and $\mathbf{p}_2$ are two detected marking-points and that the local image pattern defined by $\mathbf{p}_1$ and $\mathbf{p}_2$ is classified as "right-angled clockwise" or "right-angled anticlockwise." If this is the case, the coordinates of the two non-marking-point vertices $\mathbf{p}_3$ and $\mathbf{p}_4$ can be easily computed. For example,

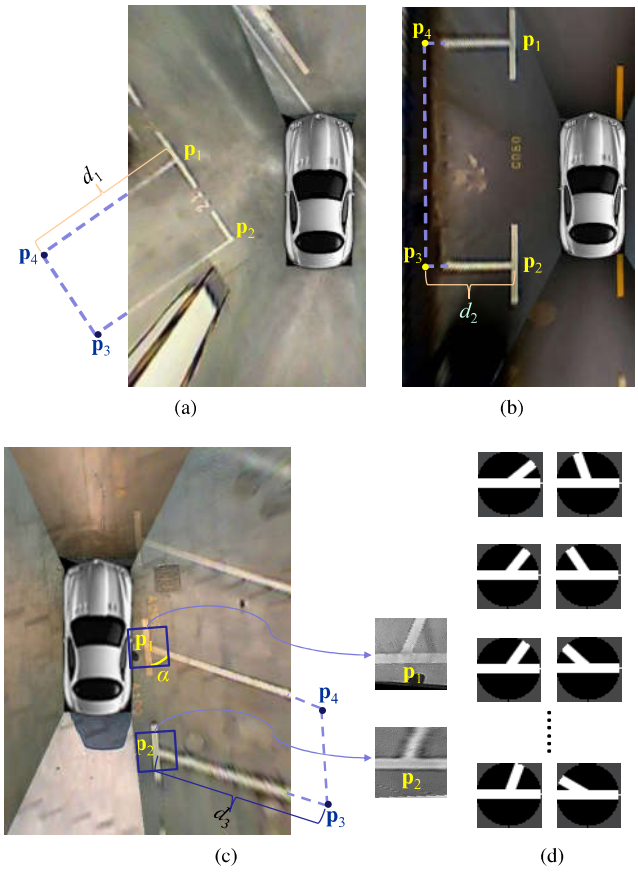


Fig. 9. To infer the positions of non-marking-point vertices, the "depth" of the parking-slot needs to be known as a priori knowledge. As illustrated in (a), (b), and (c), the depth values for the perpendicular, parallel, and slanted parking-slots are set as d_1, d_2, and d_3, respectively. In (c), α is the parking angle that needs to be estimated and to do so, two image patches centered on $\mathbf{p}_1$ and $\mathbf{p}_2$ are extracted for template-matching as shown on the right part of (c). Several examples of ideal "T-shaped" templates corresponding to different parking angles are shown in (d).

in Fig. 9(a), the local image pattern defined by $\mathbf{p}_1$ and $\mathbf{p}_2$ is "right-angled clockwise" and the length $\|\overrightarrow{\mathbf{p}_1\mathbf{p}_2}\|$ indicates that this parking-slot should be a perpendicular one (not a parallel one) and accordingly its "depth" is d_1. As a consequence, its $\mathbf{p}_3$ and $\mathbf{p}_4$ are inferred as,

$$\begin{aligned}\mathbf{p}_3 &= \begin{bmatrix}\cos\frac{\pi}{2} & \sin\frac{\pi}{2}\\ -\sin\frac{\pi}{2} & \cos\frac{\pi}{2}\end{bmatrix}\frac{\overrightarrow{\mathbf{p}_1\mathbf{p}_2}}{\|\overrightarrow{\mathbf{p}_1\mathbf{p}_2}\|}\cdot d_1+\mathbf{p}_2\\ \mathbf{p}_4 &= \begin{bmatrix}\cos\frac{\pi}{2} & \sin\frac{\pi}{2}\\ -\sin\frac{\pi}{2} & \cos\frac{\pi}{2}\end{bmatrix}\frac{\overrightarrow{\mathbf{p}_1\mathbf{p}_2}}{\|\overrightarrow{\mathbf{p}_1\mathbf{p}_2}\|}\cdot d_1+\mathbf{p}_1\end{aligned} \quad (1)$$

When the local image pattern defined by two marking-points $\mathbf{p}_1$ and $\mathbf{p}_2$ is classified as "slanted," the problem will become a little more complicated as we will need to estimate the parking angle as illustrated in Fig. 9(c). In Fig. 9(c), the local image pattern defined by $\mathbf{p}_1$ and $\mathbf{p}_2$ is classified as "slanted anticlockwise with an acute parking angle"; to estimate the positions of the two non-marking-point vertices, the parking angle α needs to be estimated. To solve this issue, a template-matching based strategy is used. A set of ideal "T-shaped" templates $\{\mathbf{T}_{\theta_j}\}_{j=1}^{M}$ as illustrated in Fig. 9(d) are prepared offline, where θ_j is the angle between the two lines of the template j and M is the total number of templates. Each template is of the size $s \times s$ and is zero-mean. At the testing stage, two $s \times s$ image patches $\mathbf{I}_1$ and $\mathbf{I}_2$ centered at $\mathbf{p}_1$ and $\mathbf{p}_2$, respectively, are extracted. $\mathbf{I}_1$ and $\mathbf{I}_2$ are both symmetric to $\overrightarrow{\mathbf{p}_1\mathbf{p}_2}$. Then, the parking-angle α can be naturally estimated as,

$$\alpha = \arg\max_{\theta_j}\left\{\mathbf{I}_1 * \mathbf{T}_{\theta_j} + \mathbf{I}_2 * \mathbf{T}_{\theta_j}\right\}, \quad j = 1, \ldots, M \quad (2)$$

where "*" denotes the correlation operation. Having computed the parking angle, the coordinates of the two non-marking-point vertices can be computed straightforwardly. For example, in Fig. 9(c), the coordinates of $\mathbf{p}_3$ and $\mathbf{p}_4$ can be expressed as,

$$\begin{aligned}\mathbf{p}_3 &= \begin{bmatrix}\cos\alpha & -\sin\alpha\\ \sin\alpha & \cos\alpha\end{bmatrix}\frac{\overrightarrow{\mathbf{p}_1\mathbf{p}_2}}{\|\overrightarrow{\mathbf{p}_1\mathbf{p}_2}\|}\cdot d_3+\mathbf{p}_2\\ \mathbf{p}_4 &= \begin{bmatrix}\cos\alpha & -\sin\alpha\\ \sin\alpha & \cos\alpha\end{bmatrix}\frac{\overrightarrow{\mathbf{p}_1\mathbf{p}_2}}{\|\overrightarrow{\mathbf{p}_1\mathbf{p}_2}\|}\cdot d_3+\mathbf{p}_1\end{aligned} \quad (3)$$

D. Overall Pipeline

In Sect. III-A ~ III-C, we have presented details about our DCNN-based parking-slot detection approach DeepPS. In order to enable the reader to have a clear and overall understanding of our work, the pipeline of DeepPS is summarized in Table II.

IV. Experimental Results

A. Benchmark Dataset

In order to provide a reasonable evaluation platform for parking-slot detection algorithms, we have established and released a large-scale benchmark dataset, which is publicly available at https://cslinzhang.github.io/deepps/.

Surround-view images in this dataset were collected from typical indoor and outdoor parking sites using our self-developed AVM system equipped on a SAIC Roewe E50 electric car [57]. The spatial resolution of each surround-view image is 600×600, corresponding to a $10m \times 10m$ flat physical region, i.e., the length of 1 pixel on the surround-view image corresponds to 1.67cm on the physical ground. It is worth noting that through the offline calibration, we can get the transformation matrix from the surround-view image coordinate system to the vehicle-centered world coordinate system. Consequently, when a parking-slot is detected on the surround-view, its coordinates in the world coordinate system can be automatically determined.

In [26], we released a parking-slot dataset, the only publicly available one in this field, and in this paper it is referred to as Tongji Parking-slot Dataset 1.0 (ps1.0 for short). The newly established dataset is referred to as Tongji Parking-slot Dataset 2.0 (ps2.0 for short). Information of these two datasets regarding their scales and imaging conditions is summarized in Table III. Based on Table III, it can be seen that from the perspectives of the number of samples, the variety of

TABLE II
OVERALL PIPELINE OF DEEPPS

Training phase
Input: A set $\mathbb{S}$, comprising surround-view images with labeled marking-points and parking-slots.
Output: The marking-point detector $\mathcal{D}$ and the local image pattern classification model $\mathcal{M}$.
1. Augment $\mathbb{S}$ as $\mathbb{S}'$, which comprises multiple rotated versions of images in $\mathbb{S}$;
2. For marking-point $\mathbf{p}_i$ in $\mathbb{S}'$, generate a $p \times p$ bounding-box $\mathbf{B}_i$ centered on $\mathbf{p}_i$;
3. Feed $\mathbb{S}'$ and $\{\mathbf{B}_i\}$ to YoloV2, start training, and $\mathcal{D}$ can be obtained;
4. Get the set $\mathbb{C}$, comprising 7 classes of the local image patterns extracted from $\mathbb{S}$;
5. Use SMOTE to over-sample the minority classes of $\mathbb{C}$;
6. Feed $\mathbb{C}$ to the customized DCNN, start training, and finally get $\mathcal{M}$.

Testing phase
Input: A surround-view image $\mathbf{I}$, the marking-point detector $\mathcal{D}$ and the local image pattern classification model $\mathcal{M}$.
Output: $\mathbb{P}$, the set containing detected parking-slots from $\mathbf{I}$.
1. Feed $\mathbf{I}$ to $\mathcal{D}$ to obtain the marking-point set $\{\mathbf{p}_i\}_{i=1}^{L}$;
2. for $i = 1 : L-1$
  for $j = i+1 : L$
    if $\|\mathbf{p}_i\mathbf{p}_j\|$ does not meet the requirements for being a valid entrance-line candidate
      continue;
    end if
    Extract the local image pattern $\mathbf{I}_{\mathbf{p}_i\mathbf{p}_j}$ defined by $\mathbf{p}_i$ and $\mathbf{p}_j$;
    c = classify $\mathbf{I}_{\mathbf{p}_i\mathbf{p}_j}$ by $\mathcal{M}$;
    if c == "invalid"
      continue;
    end if
    if c == "right-angled clockwise" or "right-angled anticlockwise"
      Derive the coordinates of the non-marking-point vertices $\mathbf{p}_{ij}^3$ and $\mathbf{p}_{ij}^4$ similarly as Eq. 1;
    else
      Estimate the parking angle α as Eq. 2;
      Derive the coordinates of the non-marking-point vertices $\mathbf{p}_{ij}^3$ and $\mathbf{p}_{ij}^4$ similarly as Eq. 3;
    end if
    $ps = [\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_{ij}^3, \mathbf{p}_{ij}^4]$;
    if vertices in ps are arranged in an anticlockwise order
      $ps = [\mathbf{p}_j, \mathbf{p}_i, \mathbf{p}_{ij}^4, \mathbf{p}_{ij}^3]$;
    end if
    Add ps to $\mathbb{P}$;
  end for
end for

TABLE III
INFORMATION ABOUT THE DATASETS USED FOR PARKING-SLOT DETECTION

	ps1.0	ps2.0
number of training images	7500	9827
number of testing images	1100	2338
indoor?	√	√
outdoor normal daylight?	√	√
outdoor rainy?	×	√
outdoor strong shadow?	×	√
outdoor street light?	×	√
right-angled parking-slots?	√	√
slanted parking-slots?	×	√

parking-slot types, and the diversity of imaging conditions, ps2.0 is much better than ps1.0 and thus the following experiments were all conducted on ps2.0.

TABLE IV
SAMPLE NUMBERS OF SUBSETS IN PS2.0

subset name	sample number
indoor parking-lot	226
outdoor normal daylight	546
outdoor rainy	244
outdoor shadow	1127
outdoor street light	147
slanted	48

TABLE V
SETTINGS FOR HYPERPARAMETERS OF DEEPPS

parameter	value	parameter	value
p	56	J	72
δ_1	0.715	d_1	195
d_2	83	d_3	195
t_1	160	t_2	279
t_3	86	t_4	139
Δx	44	Δy	48
h	48	w	192
M	82	s	47

In addition, in order to test the algorithm's performance under different imaging conditions, we partition the test set of ps2.0 into six subsets, "indoor parking-lot," "outdoor normal daylight," "outdoor rainy," "outdoor shadow," "outdoor street light," and "slanted." The sample numbers of these subsets are listed in Table IV.

B. Settings for Hyperparameters of DeepPS

In our proposed parking-slot detection approach DeepPS, there are some hyperparameters that need to be determined. It needs to be noted that the open-source YoloV2 framework (used in DeepPS to detect marking-points) requires that the input image should be of the size 416×416. Thus, for convenience, when training and testing DeepPS, all the images were resized to 416×416. All the other hyperparameters were empirically set to be compatible to this resolution and they (except for the ones relevant to DCNN used for local image pattern classification that are presented in Table I) are summarized in Table V. The physical meanings of these parameters can be understood by the context in Sect. III. "p" is the side length of the bounding-box of a marking-point; "J" indicates the number of rotated versions generated from one original image for data augmentation for training the marking-point detector; "δ_1" is the threshold for determining a marking-point; "d_1," "d_2," and "d_3" are the "depth" values of the perpendicular, parallel, and slanted parking-slots, respectively; "(t_1, t_2)" is the length range of entrance-lines for parallel parking-slots; "(t_3, t_4)" is the length range of entrance-lines for perpendicular or slanted parking-slots; the size of the local region determined by a pair of marking-points $\mathbf{p}_1$ and $\mathbf{p}_2$ is $\|\mathbf{p}_1\mathbf{p}_2\| + \Delta x$ by Δy; "$\omega \times h$" is the size of the normalized local image pattern; "M" is the number of templates prepared for estimating the parking angle of a slanted parking-slot and each template is of the size "$s \times s$."

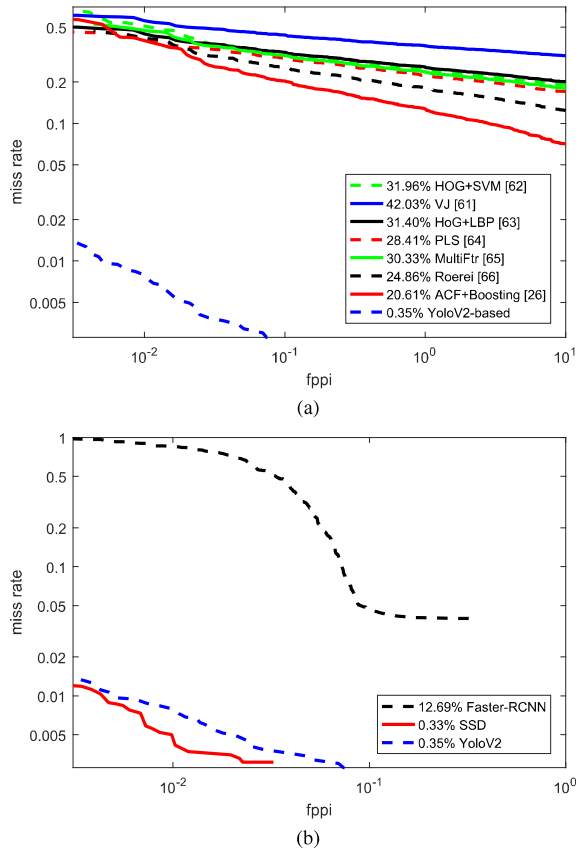


Fig. 10. Marking-point detection results by different methods. (a) shows the results of YoloV2 and classical detectors while (b) shows the results of DCNN-based detectors.

C. Marking-Point Detection

In our parking-slot detection scheme, marking-point detection is a crucial step. To complete this task, we proposed a YoloV2-based approach in DeepPS. In this experiment, its performance will be evaluated and compared with several other classical methods in the field of object detection, VJ [61], HoG+SVM [62], HoG+LBP [63], PLS [64], MultiFtr [65], Roerei [66], and ACF+Boosting [26]. In addition, another two popular DCNN-based detectors, Faster-RCNN [52] and SSD [55], were also evaluated. Evaluations were conducted on the test set of ps2.0.

For a ground-truth marking-point $\mathbf{g}_i$, if there is a detected marking point $\mathbf{d}_i$ satisfying $\|\mathbf{g}_i - \mathbf{d}_i\| < \delta_2$, where δ_2 is a predefined threshold, we deem that $\mathbf{g}_i$ is correctly detected and $\mathbf{d}_i$ is a true positive. In this experiment, δ_2 was set as 10. To compare various detectors, we plot miss rate against false positives per image (FPPI) using log-log plots by varying the threshold on detection confidence. The plots are shown in Fig. 10. As recommended in [67], we use the log-average miss rate (LAMR) to summarize detector performance, computed by averaging miss rate at nine FPPI rates evenly spaced in log-space in the range 10^{-2} to 10^{0}. LAMR achieved by different methods are also shown on Fig. 10.

In addition to the detection rate, the localization error is another important indicator for measuring the performance of a marking-point detector as it will have a direct impact on the success rate of parking. Denote by $\mathbf{g}_i'$ a detected true positive and by $\mathbf{g}_i$ its corresponding ground-truth. The localization error for $\mathbf{g}_i'$ is $e_i = \|\mathbf{g}_i' - \mathbf{g}_i\|$. We use the mean and the standard deviation of $\{e_i\}_{i=1}^{P}$ (where P is the number of detected true positives) as the localization error of the examined marking-point detector. The localization errors of all the marking-point detectors evaluated are summarized in Table VI.

TABLE VI
LOCALIZATION ERRORS OF MARKING-POINT DETECTORS

method	localization error (in pixel)	localization error (in cm)
HoG+SVM [62]	4.03 ± 1.98	6.72 ± 3.30
VJ [61]	4.83 ± 2.24	8.05 ± 3.73
HoG+LBP [63]	4.01 ± 1.97	6.68 ± 3.28
PLS [64]	3.64 ± 1.87	6.07 ± 3.12
MultiFtr [65]	3.92 ± 1.91	6.53 ± 3.18
Roerei [66]	2.96 ± 1.81	4.93 ± 3.02
ACF+Boosting [26]	2.86 ± 1.54	4.77 ± 2.57
Faster-RCNN [52]	3.67 ± 2.32	6.12 ± 3.87
SSD [55]	1.51 ± 1.17	2.52 ± 1.95
YoloV2-based	1.55 ± 1.05	2.58 ± 1.75

TABLE VII
TIME COST FOR DETECTING MARKING-POINTS FROM ONE IMAGE BY DEEP MODELS

method	time cost (ms)
Faster-RCNN	63.7
SSD	27.1
YoloV2	14.3

From the results shown in Fig. 10 and in Table VI, it can be seen that for the task of marking-point detection, DCNN-based detectors (YoloV2, SSD, and Faster-RCNN) can achieve much higher accuracy than classical methods in the field of object detection. Particularly, YoloV2 and SSD perform the best and their performance is comparable. In Table VII, the time costs (achieved on Nvidia TitanX GPU) for detecting marking-points from one image by the three deep models are presented. From Table VII, it can be seen that YoloV2 runs the fastest. Thus, in DeepPS, YoloV2 is finally adopted for marking-point detection.

D. Local Image Pattern Classification

In DeepPS, when the marking-points are detected, we need to classify each local image pattern defined by a pair of marking-points as one of the seven predefined classes. To achieve this goal, a DCNN structure is customized as shown in Fig. 8, which is actually a highly simplified version of AlexNet [46]. In this experiment, we will report the performance of our customized DCNN, in terms of the classification accuracy and the time cost for one prediction operation. As a comparison, we also tested the performance of AlexNet [46], VGG-16 [48], ResNet-50 [49], and SqueezeNet [68]. It needs to be noted that when being tested, all these DCNN models were running on GPU (Nvidia TitanX).

TABLE VIII
PERFORMANCE FOR LOCAL IMAGE PATTERN CLASSIFICATION

DCNN structure	accuracy	time cost for one prediction
customized DCNN	99.83%	1.2ms
AlexNet [46]	99.77%	2.4ms
VGG-16 [48]	99.83%	7.9ms
ResNet-50 [49]	99.71%	13.4ms
SqueezeNet [68]	99.77%	4.1ms

TABLE IX
PARAMETER SETTINGS OF ALEXNET

layer name	parameter settings	dimension of output
input		$227 \times 227 \times 3$
conv1	kernel: [11 11] stride: 4 num_output: 96	$55 \times 55 \times 96$
max-pool1	kernel: [3 3] stride: 2	$27 \times 27 \times 96$
conv2	kernel: [5 5] pad: [2 2] num_output: 256	$27 \times 27 \times 256$
max-pool2	kernel: [3 3] stride: 2	$13 \times 13 \times 256$
conv3	kernel: [3 3] pad: [1 1] num_output: 384	$13 \times 13 \times 384$
conv4	kernel: [3 3] pad: [1 1] num_output: 384	$13 \times 13 \times 384$
conv5	kernel: [3 3] pad: [1 1] num_output: 256	$13 \times 13 \times 256$
max-pool3	kernel: [3 3] stride: 2	$6 \times 6 \times 256$
FC1	num_output: 4096	4096
FC2	num_output: 4096	4096
FC3	num_output: 7	7

The test set was composed of all the local image patterns defined by marking-point pairs, which satisfy the distance constraints for being valid entrance-line candidates, from the test set of ps2.0. The results are presented in Table VIII.

From Table VIII, it can be seen that our customized DCNN can achieve comparable accuracy with the other popular large deep models. However, the customized DCNN consumes much less time to complete one prediction operation, making it more suitable for the time-critical self-parking system. The reason why our customized DCNN is faster than AlexNet is that our customized network is elaborately tailored based on the original AlexNet, in order to better fit our application. Table I and Table IX show the parameter settings for key layers of our customized DCNN and AlexNet, respectively. From them, it can be straightforward to understand why the customized DCNN runs faster than AlexNet. In fact, there are three major differences that can explain why the customized DCNN is faster than AlexNet. First, our customized DCNN is two layers shallower than AlexNet. A convolution layer and a fully-connected layer are removed. Second, compared with AlexNet, roughly corresponding layers in customized DCNN have low-resolution inputs. Third, we reduce the number of kernels used in convolution layers and also the numbers of output units of the fully-connected layers.

TABLE X
PERFORMANCE EVALUATION OF PARKING-SLOT DETECTION METHODS ON THE ENTIRE TEST SET OF PS2.0

method	precision rate	recall rate
Jung *et al.* [31]	98.38%	52.39%
Wang *et al.* [25]	98.27%	56.16%
Hamada *et al.* [34]	98.29%	60.41%
Suhr&Jung [41]	98.38%	70.96%
PSD_L [26]	98.55%	84.64%
DeepPS	**99.54%**	**98.89%**

E. Parking-Slot Detection

In this experiment, the parking-slot detection performance of DeepPS was evaluated. Besides, the performance of several state-of-the-art methods in this field, including Jung *et al.*'s method [31], Wang *et al.*'s method [25], Hamda *et al.*'s method [34], Suhr and Jung's method [41], and our previous work PSD_L [26] was also evaluated for comparison.

The experiments were conducted on the test set of ps2.0 dataset. Precision-recall rates were used as the performance measure, which are defined as,

$$precision = \frac{true\ positives}{true\ positives + false\ positives}$$
$$recall = \frac{true\ positives}{true\ positives + false\ negatives} \quad (4)$$

Each labeled parking-slot is represented as $PS_i = \{\mathbf{p}_1^i, \mathbf{p}_2^i, \mathbf{p}_3^i, \mathbf{p}_4^i\}$, where $\mathbf{p}_1^i$, $\mathbf{p}_2^i$, $\mathbf{p}_3^i$, and $\mathbf{p}_4^i$ are the coordinates of the four vertices. $\mathbf{p}_1^i$ and $\mathbf{p}_2^i$ are the coordinates of the two vertices forming the entrance-line and the four vertices are arranged in a clockwise manner. Suppose that $PS_d = \{\mathbf{p}_1^d, \mathbf{p}_2^d, \mathbf{p}_3^d, \mathbf{p}_4^d\}$ is a detected parking-slot and $PS_l = \{\mathbf{p}_1^l, \mathbf{p}_2^l, \mathbf{p}_3^l, \mathbf{p}_4^l\}$ is a labeled ground-truth. If $\mathbf{p}_j^d$ matches with $\mathbf{p}_j^l$, $j = 1 \sim 4$, PS_d is regarded as a true positive.[6] If PS_d does not match with any ground-truth parking-slot, PS_d is a false positive. If PS_l does not match with any detected parking-slot, PS_l is a false negative.

We adjusted the parameters of all the competing methods to make their precision rates greater than 98% on the entire test set of ps2.0. The results are summarized in Table X. From Table X, it can be observed that when operating at a high precision rate, DeepPS performs much better than the other competitors. DeepPS's recall-rate is 98.89% while PSD_L is the runner-up whose recall-rate is 84.64%.

In order to roughly know the algorithms' performance under different imaging conditions, we also evaluated them on each sub test-set of ps2.0 and the results of two best performing methods DeepPS and PSD_L are given in Table XI.

From the results presented in Tables X and XI, we can draw a number of conclusions.

1) The classical methods [25], [31], [34], [41] cannot achieve satisfied results. The root cause is that they are

[6] If $\left\|\mathbf{p}_j^d - \mathbf{p}_j^l\right\| < \delta_3$, we say $\mathbf{p}_j^d$ matches with $\mathbf{p}_j^l$. We set δ_3 as 12 in our experiments.

TABLE XI
EVALUATION RESULTS OF PSD_L AND DEEPPS ON SUBSETS OF PS2.0

subset	PSD_L (precision, recall)	DeepPS (precision, recall)
indoor parking-lot	(99.34%, 87.46%)	(99.41%, 97.67%)
outdoor normal daylight	(99.44%, 91.65%)	(99.49%, 99.23%)
outdoor rainy	(98.68%, 87.72%)	(100%, 99.42%)
outdoor shadow	(97.52%, 73.67%)	(99.86%, 99.28%)
outdoor street light	(98.92%, 92.00%)	(100%, 100%)
slanted	(93.15%, 83.95%)	(96.25%, 95.06%)

based on primitive visual features (edges, lines, or corners) and are not robust to complicated scenes and various imaging conditions.

2) PSD_L exhibits clear performance advantages over its counterparts depending on low-level visual features. The better performance should be attributed to the machine learning (ACF + Boosting) based marking-point detection scheme used in PSD_L.
3) In all cases, the performance of DeepPS surpass that of PSD_L by a large margin. Compared with PSD_L, the superior performance of DeepPS is mainly for two reasons. First, for detecting marking-points, DeepPS adopts a state-of-the-art DCNN-based object detection framework YoloV2, rather than ACF + Boosting as used in PSD_L. The results shown in Fig. 10 and Table VI clearly demonstrate that the YoloV2-based framework works much better than the ACF+Boosting-based scheme in terms of the detection accuracy and the localization accuracy. Second, to determine the validity and the type of an entrance-line, PSD_L uses a complicated rule-based scheme while DeepPS formulates it as a local image pattern classification problem which can be more concisely and robustly solved by a standard DCNN model.

In Fig. 11, 6 surround-view images are shown with marked parking-slots detected by DeepPS. Actually, they are representatives chosen from six subsets of the ps2.0 test set. The results shown in Fig. 11 once again corroborate the capability of DeepPS in detecting different types of parking-slots under various imaging conditions. More demo videos can be found at https://cslinzhang.github.io/deepps/.

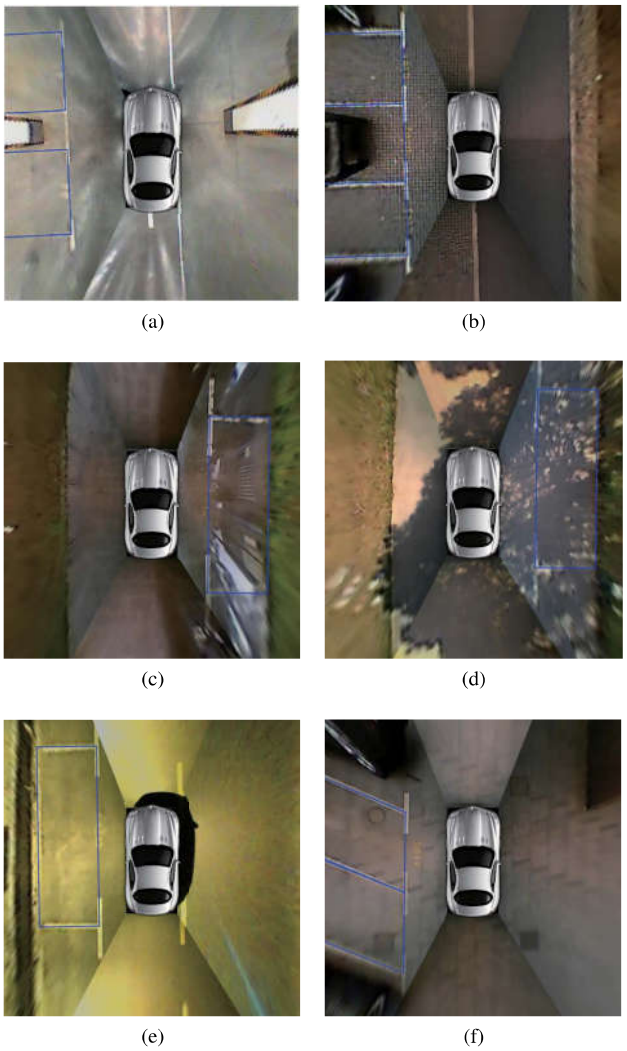


Fig. 11. 6 typical images with marked parking-slots detected by DeepPS. (a) ∼ (f) belong to the subsets "indoor parking-lot," "outdoor normal daylight," "outdoor rainy," "outdoor shadow," "outdoor street light," and "slanted," respectively.

F. Failure Cases of DeepPS

DeepPS currently is not perfect. When the imaging conditions are poor (e.g., there are strong shadows caused by nearby trees), sometimes it will miss a true candidate (false negative). There are two main reasons for missing true candidates. First, the confidence score of a true marking-point is lower than the predefined threshold. Second, the local image pattern classification model may misclassify a valid entrance-line candidate as "invalid."

Compared with false negatives, in a practical self-parking system, false positives actually are more annoying. Occasionally, DeepPS may return a false positive. The root reason is that the DCNN model may (although quite rare) misclassify with a high confidence score the local image pattern defined by two marking-points. Four examples of false positives detected by DeepPS are shown in Fig. 12. In Fig. 12(a), the "right-angled clockwise" entrance-line is misclassified as "right-angled anticlockwise." In Fig. 12(b), the "slanted" entrance-line is misclassified as "right-angled." In Fig. 12(c), the "right-angled anticlockwise" local image pattern defined by $\mathbf{p}_1$ and $\mathbf{p}_2$ is misclassified as "right-angled clockwise." In Fig. 12(d), the "invalid" local image pattern is misclassified as "right-angled clockwise." Actually, misclassifying images with high confidences is a known shortcoming of DCNN [69], and in the future we may adopt strategies (e.g., DeepFool [70]) for creating adversarial training samples to solve this issue.

G. Analysis of Performance Improvement

As compared with PSD_L [26], the novelty of DeepPS lies largely in two directions. First, we replace the ACF+Boosting Decision Tree based marking-point detector used in PSD_L with a YoloV2-based one. Second, for inferring parking-slots

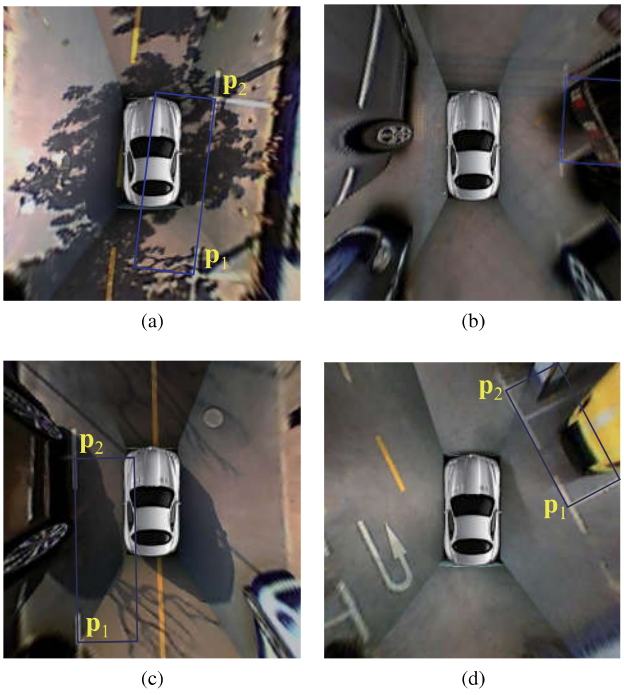


Fig. 12. Four examples of false positives detected by DeepPS. (a) The "right-angled clockwise" local image pattern defined by $\mathbf{p}_1$ and $\mathbf{p}_2$ is misclassified as "right-angled anticlockwise." (b) The "slanted" local image pattern is misclassified as "right-angled." (c) The "right-angled anticlockwise" local image pattern defined by $\mathbf{p}_1$ and $\mathbf{p}_2$ is misclassified as "right-angled clockwise." (d) The "invalid" local image pattern is misclassified as "right-angled clockwise."

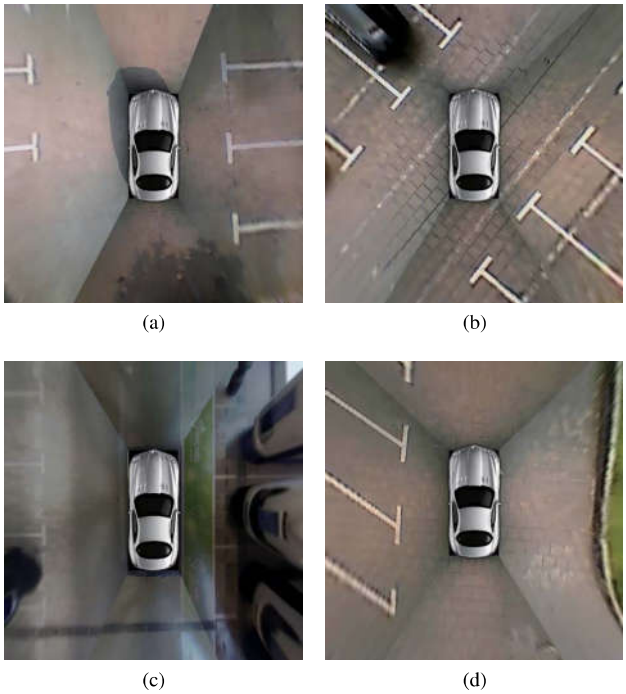


Fig. 13. (a) ∼ (d) are four representative image samples in extraTestSet, which was established to evaluate the generalization capability of DeepPS in practice.

TABLE XII
ANALYSIS OF PERFORMANCE IMPROVEMENT

method	precision rate	recall rate
PSD_L	98.55%	84.64%
PSD_L$_{\texttt{Yolo}}$	98.19%	92.95%
PSD_L$_{\texttt{LIP}}$	99.20%	89.38%
DeepPS	99.54%	98.89%

from marking-points, PSD_L uses line-based template matching while DeepPS resorts to the local image pattern classification. Here we explain and demonstrate the performance improvement afforded by each new aspect of DeepPS. Denote by PSD_L$_{\texttt{Yolo}}$ the model which replaces the PSD_L's marking-point detector with the YoloV2-based one. Denote by PSD_L$_{\texttt{LIP}}$ the model which replaces the PSD_L's marking-point inferring strategy with the local image pattern classification based one. Their performances are reported in Table XII. We also present the performance of PSD_L and DeepPS in Table XII for comparison. From the results presented in Table XII, it can be seen that the appropriate strategies for marking-point detection and for parking-slot inference can both boost the final performance.

H. Running Speed of DeepPS

DeepPS was implemented in C++. Experiments were conducted on a workstation with a 2.4GHZ Intel Xeon E5-2630V3 CPU, an Nvidia Titan X GPU card, and 32GB RAM. The average time for DeepPS to process one image frame is about 23ms, including the time for synthesizing the surround-view.

I. DeepPS's Generalization Capability in Practice

In this experiment, the generalization capability of DeepPS (trained on ps2.0) in practice was validated. Specifically, 1000 more surround-view images were collected from several parking-sites using a Roewe E50 car equipped with our vision-based parking-slot detection system. We denote this image set by extraTestSet. It needs to be noted that parking-sites that appear in extraTestSet do not appear in ps2.0. Four representative image samples selected from extraTestSet are shown in Fig. 13. We labeled images in extraTestSet and then the performance of DeepPS trained on ps2.0 was evaluated on extraTestSet. The achieved precision rate and recall rate are 98.43% and 91.95%, respectively. The results demonstrate that the DeepPS model trained on ps2.0 has a satisfied generalization capability.

V. CONCLUSION AND FUTURE WORK

Vision-based parking-slot detection is a crucial issue encountered when building a self-parking system. It is challenging and still unresolved. To conquer it, in this paper, we proposed a DCNN-based solution, DeepPS. Another contribution is that we collected and labeled a large-scale parking-slot image dataset ps2.0, which comprises 12,165 surround-view images and is the largest one in this field. Extensive experiments conducted on ps2.0 indicate that DeepPS could surpass all its competitors by a large margin.

Specifically, on the entire test set of ps2.0, DeepPS could achieve a recall rate as high as 98.89% while its precision rate is 99.54%. Furthermore, DeepPS can process one frame within 23ms, making it competent for time-critical self-parking systems. Actually, DeepPS has already been deployed on SAIC Roewe E50 electric cars. In the future, we will continue enlarging our dataset to make it a better benchmark in this field.

REFERENCES

[1] M. Wada, K. S. Yoon, and H. Hashimoto, "Development of advanced parking assistance system," *IEEE Trans. Ind. Electron.*, vol. 50, no. 1, pp. 4–17, Feb. 2003.

[2] W. Zhan, H. Shoudong, D. Gamini, *Simultaneous Localization and Mapping: Exactly Sparse Information Filters*. Singapore: World Scientific, 2011.

[3] J. Pohl, M. Sethsson, P. Degerman, and J. Larsson, "A semi-automated parallel parking system for passenger cars," *Proc. Inst. Mech. Eng., D, J. Automobile Eng.*, vol. 220, no. 1, pp. 53–65, 2006.

[4] H. Satonaka, M. Okuda, S. Hayasaka, T. Endo, Y. Tanaka, and T. Yoshida, "Development of parking space detection using an ultrasonic sensor," in *Proc. 13th World Congr. Intell. Transp. Syst. Serv.*, 2006, pp. 1–10.

[5] W.-J. Park, B.-S. Kim, D.-E. Seo, D.-S. Kim, and K.-H. Lee, "Parking space detection using ultrasonic sensor in parking assistance system," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2008, pp. 1039–1044.

[6] S. H. Jeong *et al.*, "Low cost design of parallel parking assist system based on an ultrasonic sensor," *Int. J. Automot. Technol.*, vol. 11, no. 3, pp. 409–416, 2010.

[7] *Ford Fusion*. Accessed: Aug. 2017. [Online]. Available: http://www.ford.com/cars/fusion/2017 /features/smart/

[8] *Toyota Prius*. Accessed: Aug. 2017. [Online]. Available: http://www.toyota.com/prius/prius-features/

[9] H. G. Jung, Y. H. Cho, P. J. Yoon, and J. Kim, "Scanning laser radar-based target position designation for parking aid system," *IEEE Trans. Intell. Transp. Syst.*, vol. 9, no. 3, pp. 406–424, Sep. 2008.

[10] J. Zhou, L. E. Navarro-Serment, and M. Hebert, "Detection of parking spots using 2D range data," in *Proc. IEEE Int. Conf. Intell. Transp. Syst.*, Sep. 2012, pp. 1280–1287.

[11] A. Ibisch *et al.*, "Towards autonomous driving in a parking garage: Vehicle localization and tracking using environment-embedded LIDAR sensors," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2013, pp. 829–834.

[12] M. R. Schmid, S. Ates, J. Dickmann, F. von Hundelshausen, and H.-J. Wuensche, "Parking space detection with hierarchical dynamic occupancy grids," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2011, pp. 254–259.

[13] R. Dubé, M. Hahn, M. Schütz, J. Dickmann, and D. Gingras, "Detection of parked vehicles from a radar based occupancy grid," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2014, pp. 1415–1420.

[14] A. Loeffler, J. Ronczka, and T. Fechner, "Parking lot measurement with 24 GHz short range automotive radar," in *Proc. IEEE Int. Radar Symp.*, Jun. 2015, pp. 137–142.

[15] H. G. Jung, D. S. Kim, and J. Kim, "Light-stripe-projection-based target position designation for intelligent parking-assist system," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 4, pp. 942–953, Dec. 2010.

[16] U. Scheunert, B. Fardi, N. Mattern, G. Wanielik, and N. Keppeler, "Free space determination for parking slots using a 3D PMD sensor," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2007, pp. 154–159.

[17] N. Kaempchen, U. Franke, and R. Ott, "Stereo vision based pose estimation of parking lots using 3D vehicle models," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2002, pp. 459–464.

[18] K. Fintzel, R. Bendahan, C. Vestri, S. Bougnoux, and T. Kakinami, "3D parking assistant system," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2004, pp. 881–886.

[19] C. Vestri, S. Bougnoux, R. Bendahan, K. Fintzel, S. Wybo, F. Abad, and T. Kakinami, "Evaluation of a vision-based parking assistance system," in *Proc. IEEE Int. Conf. Intell. Transp. Syst.*, Sep. 2005, pp. 131–135.

[20] H. G. Jung, D. S. Kim, P. J. Yoon, and J. Kim, "3D vision system for the recognition of free parking site location," *Int. J. Automot. Technol.*, vol. 7, no. 3, pp. 361–367, 2006.

[21] J. K. Suhr, H. G. Jung, K. Bae, and J. Kim, "Automatic free parking space detection by using motion stereo-based 3D reconstruction," *Mach. Vis. Appl.*, vol. 21, no. 2, pp. 163–176, 2010.

[22] C. Unger, E. Wahl, and S. Ilic, "Parking assistance using dense motion-stereo," *Mach. Vis. Appl.*, vol. 25, no. 3, pp. 561–581, Apr. 2014.

[23] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 11, pp. 1330–1334, Nov. 2000.

[24] *OpenCV 3.1*. Accessed: Aug. 2017. [Online]. Available: http://www.opencv.org/opencv-3-1.html

[25] C. Wang, H. Zhang, M. Yang, X. Wang, L. Ye, and C. Guo, "Automatic parking based on a bird's eye view vision system," *Adv. Mech. Eng.*, vol. 6, p. 847406, Mar. 2014.

[26] L. Li, L. Zhang, X. Li, X. Liu, Y. Shen, and L. Xiong, "Vision-based parking-slot detection: A benchmark and a learning-based approach," in *Proc. IEEE Int. Conf. Multimedia Expo*, Jul. 2017, pp. 649–654.

[27] J. Xu, G. Chen, and M. Xie, "Vision-guided automatic parking for smart car," in *Proc. IEEE Intell. Vehicles Symp.*, Oct. 2000, pp. 725–730.

[28] H. G. Jung, D. S. Kim, P. J. Yoon, and J. Kim, "Structure analysis based parking slot marking recognition for semi-automatic parking system," in *Proc. Joint IAPR Int. Workshops Stat. Techn. Pattern*, 2006, pp. 384–393.

[29] H. G. Jung, Y. H. Lee, and J. Kim, "Uniform user interface for semiautomatic parking slot marking recognition," *IEEE Trans. Veh. Technol.*, vol. 59, no. 2, pp. 616–626, Feb. 2010.

[30] X. Du and K. K. Tan, "Autonomous reverse parking system based on robust path generation and improved sliding mode control," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 3, pp. 1225–1237, Jun. 2015.

[31] H. G. Jung, D. S. Kim, P. J. Yoon, and J. Kim, "Parking slot markings recognition for automatic parking assist system," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2006, pp. 106–113.

[32] M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis, and Machine Vision*. Atlanta, GA, USA: Thomson-Engineering, 2008.

[33] S. R. Deans, *The Radon Transform and Some of Its Applications*. New York, NY, USA: Dover, 1983.

[34] K. Hamada, Z. Hu, M. Fan, and H. Chen, "Surround view based parking lot detection and tracking," in *Proc. IEEE Intell. Vehicles Symp.*, Jun./Jul. 2015, pp. 1106–1111.

[35] J. Matas, C. Galambos, and J. Kittler, "Robust detection of lines using the progressive probabilistic hough transform," *Comput. Vis. Image Understand.*, vol. 78, no. 1, pp. 119–137, 2000.

[36] J. K. Suhr and H. G. Jung, "Automatic parking space detection and tracking for underground and indoor environments," *IEEE Trans. Ind. Electron.*, vol. 63, no. 9, pp. 5687–5698, Sep. 2016.

[37] G. Borgefors, "Hierarchical chamfer matching: A parametric edge matching algorithm," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-10, no. 6, pp. 849–856, Nov. 1988.

[38] S. Lee and S.-W. Seo, "Available parking slot recognition based on slot context analysis," *IET Intell. Transp. Syst.*, vol. 10, no. 9, pp. 594–604, Nov. 2016.

[39] M. Nieto, L. Salgado, F. Jaureguizar, and J. Arrospide, "Robust multiple lane road modeling based on perspective analysis," in *Proc. IEEE Int. Conf. Image Process.*, Oct. 2008, pp. 2396–2399.

[40] D. Barber, *Bayesian Reasoning and Machine Learning*. Cambridge, U.K.: Cambridge Univ. Press, 2013.

[41] J. K. Suhr and H. G. Jung, "Full-automatic recognition of various parking slot markings using a hierarchical tree structure," *Opt. Eng.*, vol. 52, no. 3, p. 037203, 2013.

[42] J. K. Suhr and H. G. Jung, "Sensor fusion-based vacant parking slot detection and tracking," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 1, pp. 21–36, Feb. 2014.

[43] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. 4th Alvey Vis. Conf.*, 1988, pp. 147–151.

[44] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.

[45] Y. LeCun *et al.*, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, 1989.

[46] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.* 2012, pp. 1097–1105.

[47] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1–9.

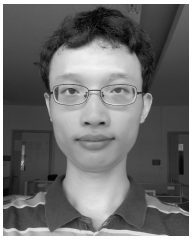
[48] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–14.

[49] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.

[50] R. Girshick, J. Donahue, T. Darrel, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587.

[51] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 1440–1448.
[52] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.
[53] T. Kong, A. Yao, Y. Chen, and F. Sun, "HyperNet: Towards accurate region proposal generation and joint object detection," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 845–853.
[54] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 779–788.
[55] W. Liu *et al.*, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 21–37.
[56] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 7263–7271.
[57] *Roewe E50 review*. Accessed: Aug. 2017. [Online]. Available: http://www.autocar.co.uk/car-review/roewe/e50
[58] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, no. 1, pp. 321–357, 2002.
[59] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F. Li, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.
[60] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 807–814.
[61] P. Viola and M. J. Jones, "Robust real-time face detection," *Int. J. Comput. Vis.*, vol. 57, no. 2, pp. 137–154, 2004.
[62] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2005, pp. 886–893.
[63] X. Wang, X. Han, and S. Yan, "An HOG-LBP human detector with partial occlusion handling," in *Proc. IEEE Int. Conf. Comput. Vis.*, Sep./Oct. 2009, pp. 32–39.
[64] W. Schwartz, A. Kembhavi, D. Harwood, and L. S. Davis, "Human detection using partial least squares analysis," in *Proc. IEEE Int. Conf. Comput. Vis.*, Sep./Oct. 2009, pp. 24–31.
[65] C. Wojek and B. Schiele, "A performance evaluation of single and multi-feature people detection," in *Proc. 30th DAGM Symp. Pattern Recognit.*, 2008, pp. 82–91.
[66] R. Benenson, M. Mathias, T. Tuytelaars, and L. Van Gool, "Seeking the strongest rigid detector," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 3666–3673.
[67] P. Dollár, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 4, pp. 743–761, Apr. 2012.
[68] F. Iandola, S. Han, M. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. (2016). "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size." [Online]. Available: https://arxiv.org/abs/1602.07360
[69] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 427–436.
[70] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 2574–2582.

Lin Zhang (M'11–SM'15) received the B.Sc. and M.Sc. degrees from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2003 and 2006, respectively, and the Ph.D. degree from the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, in 2011. In 2011, he joined the Department of Computing, The Hong Kong Polytechnic University, as a Research Assistant. In 2011, he joined the School of Software Engineering, Tongji University, Shanghai, where he is currently an Associate Professor. His current research interests include environment perception of intelligent vehicle, pattern recognition, computer vision, and perceptual image/video quality assessment.



Junhao Huang received the B.S. degree from the School of Software Engineering, Tongji University, Shanghai, China, in 2017, where he is currently pursuing the master's degree. His research interests are environment perception for self-driving vehicle, pattern recognition, and machine learning.

Xiyuan Li received the B.S. degree from the Department of Electronic Engineering, Tongji University, Shanghai, China, in 2016, where she is currently pursuing the master's degree with the School of Software Engineering. Her research interests include object detection, scene understanding, and machine learning.

Lu Xiong received the B.S., M.S., and Ph.D. degrees from the School of Automotive Studies, Tongji University, Shanghai, China, in 1999, 2002, and 2005, respectively. In 2005, he joined the School of Automotive Studies, Tongji University, as a Faculty Member, where he is currently a Full Professor. His research interests include electric vehicle design and self-driving vehicle design.