

Received August 7, 2020, accepted September 9, 2020, date of publication September 18, 2020, date of current version September 30, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3024668

Context-Based Parking Slot Detection With a Realistic Dataset

HOSEOK DO^{1,2} AND JIN YOUNG CHOI¹, (Member, IEEE)

¹Department of Electrical and Computer Engineering, Automation and Systems Research Institute, Seoul National University, Seoul 08826, South Korea

²CTO Division, LG Electronics, Seoul 06772, South Korea

Corresponding author: Jin Young Choi (jychoi@snu.ac.kr)

This work was supported by the Institute for Information and communications Technology Promotion (IITP) Grant funded by the Ministry of Science and ICT, Outdoor Surveillance Robots, under Grant 2017-0-00306.

ABSTRACT The autonomous parking of vehicles requires the ability to accurately locate an available parking slot in the vicinity of a vehicle. Since parking slots have a variety of shapes and colors, may be occluded by obstacles, or look different due to surroundings such as lighting, accurately locating them can be a challenging task. In this paper, we propose a context-based parking slot detection method inspired by the process of a human driver finding a parking slot. Our method consists of two deep network modules: a *parking context recognizer* and *parking slot detector*. The *parking context recognizer* identifies the parking environment (type, angle, and availability of a parking slot), whereas the *parking slot detector* locates the exact position of a parking slot by multiple type-based fine-tuned detectors with rotated anchor boxes and a rotated non-maximal suppression. In addition, we release a realistic parking slot dataset, which comprises 22817 images of parking slots having various attributes and external conditions. We also propose a new evaluation metric for parking slot detection, reflecting whether a vehicle can be parked within the detected parking slot. Through comparison and ablation study in experiments, we demonstrate that our method outperformed the previous deep-learning-based methods, along with having a short operation time. The source codes and the dataset are available at <https://github.com/dohoseok/context-based-parking-slot-detect/>.

INDEX TERMS Parking slot detection, context-based detector, rotated object detector, parking slot dataset.

I. INTRODUCTION

An autonomous parking system is essential for autonomous vehicles. Such a system must first detect the parking slot and then control the vehicle to park it in the designated slot. Previous methods for vision-based parking slot detection focused on finding the accurate location of the parking slot directly from a surrounding image. This is because most parking-assist systems installed in many mass-produced vehicles are operated after a person manually drives the car to a space where parking is available. In the era of self-driving cars, the vehicle should be able to search for available parking spaces on its own, and then parking slot detection should work to locate the exact position of a parking slot. During this process, false positives in parking slot detection can cause accidents or parking violations and thus should be avoided.

Since previous algorithms for vision-based parking slot detection used handcrafted features such as corners and lines [1], [2], they could not achieve satisfactory accuracy.

The associate editor coordinating the review of this manuscript and approving it for publication was Jiachen Yang.

In recent years, deep learning has been applied to parking slot detection, which has significantly improved performance. However, the deep learning approach still detects specific shapes such as parking slot junctions and infers the parking slots via post-processing of the detected shapes [3]–[5]. Therefore, the deep learning methods are also vulnerable in the absence of specific shapes of the parking slots as in the handcrafted feature-based methods. In the real world, the appearance of parking slots can vary dramatically, and some parking lines may be occluded or even non-existent. Conversely, non-parking lines similar to a parking slot marker can cause false positives.

This paper proposes a two-stage parking slot detection method based on context information of the entire image. The proposed method is inspired by the fact that when a person parks, he first recognizes a parking context on a space where parking is available (called *parking context recognizer*) and then locates the exact position of the parking slot (called *parking slot detector*). After the *parking context recognizer* filters out the space without parking slots, the *parking slot detector* operates only near the parking space. The *parking*

context recognizer decides the availability of the parking slot and also estimates parking information such as the parking type and orientation of the parking slot. The *parking slot detector* locates the exact position of the parking slot by utilizing the parking information obtained by the *parking context recognizer*.

The proposed two-stage approach reduces the amount of computation and also false positive errors because the parking slot detector utilizes useful information from the parking context recognizer and does not operate when there are no parking spaces in the vicinity of the car. Specifically, depending on the angle of orientation, similar-shaped parking slots appear differently in the image. Thus, the angle information provided by the parking context recognizer is useful for accuracy improvement by rotating the anchor box of the detector to have a similar appearance to the actual parking slot.

To cover various types of parking slots in diverse environments, we introduce a new, realistic dataset for detecting and classifying parking slots. Our dataset includes images of parking slots that have various attributes and external conditions and also contains harsh data samples where part of the parking slot is occluded by various obstacles. In addition, data samples of a *not-parking-space* class with a similar appearance to the parking slot are contained in the dataset to mitigate the false positive problem in parking slot detection. Examples of *not-parking-space* class data samples are shown in Figure 1. Images have been acquired using three different vehicles with fish-eye cameras on the left and right. In addition to the labeling on the parking slot location, the parking availability and parking slot type are also labeled, which is useful for developing autonomous parking systems.



FIGURE 1. Examples of the not-parking-space class data sample.

We propose a performance evaluation metric that considers when a parking slot detection method is applied to an autonomous parking system. It is assumed that the vehicle is parked according to the detected parking slot information, and the parking score is calculated based on how well the vehicle can be parked in that case.

The contributions of this paper are summarized as follows:

- We propose a two-stage context-based parking slot detection method which is efficient and robust to surrounding hindrance factors.
- A rotated object detector is developed to detect the rotated parking slots by using a rotated anchor box based on the orientation information.

- We construct a realistic parking slot dataset with a variety of images and useful labeling information, and a new evaluation metric that fits the actual parking problem.

II. RELATED WORK

Vision-based parking slot detection uses the image of the parking slot on the ground to recognize its location. The space around the vehicle can be recognized using fish-eye cameras such as around view monitoring (AVM) cameras mounted on the vehicle. Distortion of the fish-eye camera is corrected when the image is transformed to a bird's-eye-view image. A parking slot consists of four painted parking lines; however, sometimes walls or curbs replace some parking lines. The part where the vehicle enters in the parking slot is called the entrance line, and the boundary between neighboring parking slots is called the separating line.

The line-based method detects the parking slot by finding two separating lines which are spaced apart. The corner-based method first detects the corner where the entrance line and the separating line intersect and then detects the parking slot by combining the coordinates of the corners. However, both the line-based method and the corner-based method could not achieve satisfactory performance due to the limitations of the handcrafted features.

As deep learning has developed, convolutional neural networks (CNNs) have been applied to various computer vision tasks. Recently, CNN-based methods have been proposed for the parking slot detection problem [3]–[6]. The above methods use CNNs to find junctions, which are the most common feature of parking slots. Then, the parking slot is inferred by heuristically combining the recognized results or with post-processing. CNN-based methods show a great performance improvement in comparison with the handcrafted feature-based methods. However, there is also a limitation in CNN methods based on a specific part of the slot such as a junction, simply because there are some parking slots without such junctions. In addition, some objects, such as traffic signs rendered on the ground, appear similar to junctions. Also, the estimated angle for the separating line may not be accurate when only the junction marker at the entrance of the parking slot is used to estimate the angle.

Researchers currently have access to ps1.0 [6] and its upgraded version, ps2.0 [3], which are public datasets for the parking slot detection problem. These datasets replaced smaller private datasets and contributed to various studies on parking slot detection. However, in experiments reported in published papers, the performance measured while utilizing the ps2.0 dataset had already reached a level of over 99%. Therefore, it was felt that further studies of parking slot detection problems required a more difficult research dataset. When evaluating the performance of parking slot detection problem approaches in previous papers, the pixel error of the vertices of the parking slot and the angular error of the line were used to determine whether the inferred parking slot was true or not. Compared to intersections over union (IoU), which is the general evaluation metric of an object detection

problem, it is more appropriate to use the above errors as evaluation metrics of the parking slot detection problem. This is because, unlike other general objects, a parking slot has important information at the edge of the object. However, a simple error value can not depict the accuracy of the performance when parking a real vehicle in an inferred parking slot.

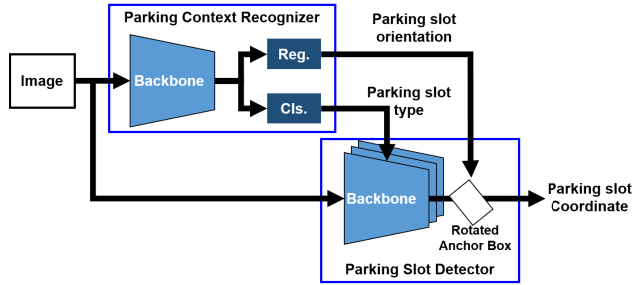


FIGURE 2. The overall scheme of the proposed method.

III. PROPOSED METHOD

A. OVERVIEW

The overall scheme of our method is depicted in Figure 2. The coordinates of the parking slot are estimated using a two-stage deep learning model consisting of a *parking context recognizer* (PCR) and a *parking slot detector* (PSD). In the PCR, the type and the orientation, which are rough information of the parking slot, are estimated. The coordinates of the parking slot are estimated at the PSD using the context information. From an intuitive perspective, our method is similar to a human operator's process of driving and parking a vehicle. The driver roughly searches for a place to park his car and then estimates the exact location of the parking slot. In our method, images captured by the fish-eye cameras on the left and right of the vehicle are converted into bird's-eye-view images and used as input to the PCR and PSD.

In the PCR, the input image is classified into one of four classes: *parallel*, *perpendicular*, *diagonal*, or *not-parking-space*. Also, the parking slot orientation is estimated to be an angular value between -90 and 90 degrees based on the separating lines of the parking slots. In most cases, since the adjacent parking slots will have the same orientation and type, the PCR outputs one parking slot type and one angular value per input image containing multiple slots. The PSD estimates the coordinates of the four vertices of a parking slot. The anchor box of the detector is rotated by the angle estimated by the PCR. In the PSD, there are three detectors, and only one of them works for detection according to the parking slot type inferred by the PCR. One detector is first trained with all the types of data, which is then copied into three detectors. Finally, each of the three detectors is fine-tuned with a particular type of data. When the image is classified as *not-parking-space*, the PSD is not activated since parking is not available in that space.

B. PARKING CONTEXT RECOGNIZER

When applied to an actual parking system, the PCR operates at all times to recognize whether a parking slot exists near the vehicle or not. In actual situations, when parking is not available, it is not necessary to estimate an accurate location of the parking slot. Hence, only when the PCR recognizes that the surrounding space is an available parking type does the PCR provide the type and slot angle to the PSD to estimate the exact location of the parking slot. In this way, the PCR works as a sort of filter and can reduce the amount of calculation in the parking system. Therefore, MobileNetV2 [7], which is a deep learning model suitable for embedded systems and fast, is sufficient as the backbone for our implementation.

The PCR receives 64×192 RGB images as input. The backbone is followed by two sibling branches; one is for type classification, and the other is for orientation regression. The classification branch consists of two fully-connected layers: a 128 hidden unit layer with rectified linear unit (ReLU) activation and a 4 output unit layer with softmax activation. Each of the 4 output units in the classification branch indicates one of the parking slot types: *parallel*, *perpendicular*, *diagonal*, or *not-parking-space*.

The regression branch consists of two fully-connected layers: a 128 hidden unit layer with ReLU activation and one output unit layer with sigmoid activation. The output unit yields the angle of the parking slot from which the ground truth is calculated based on the separating lines of the parking slot. This is because the entrance line of the parking slot may not exist, and a person usually parks the vehicle based on the separating line. Letting the angle of the line perpendicular to the vehicle's direction of travel be 0 degrees, the angle for a line rotated in a clockwise direction is set to a positive angle, whereas the angle for a line rotated in a counterclockwise direction is set to a negative angle. Hence, the angle has a value in a range between -90 and $+90$ degrees, which is normalized to a range between 0 and 1.

The training loss for the PCR is given by a multi-task loss as

$$L = -\frac{\lambda}{4} \sum_{c=1}^4 p_c \log \hat{p}_c + (1 - \lambda)(\theta - \hat{\theta})^2,$$

where \hat{p}_c is the c -th output score in the classification branch, $\hat{\theta}$ is the output value of the regression branch, whereas p_c and θ are the ground truths of the parking slot type and orientation, respectively. λ is the parameter for balancing the multi-task loss.

C. PARKING SLOT DETECTOR

The PSD is constructed based on the structure of YOLOv3 [8] and estimates the coordinates of the four vertices of the quadrangle tightly containing the parking slot. Common object detectors, including YOLOv3, estimate width, height, and center coordinates because they aim to find a horizontal bounding box enclosing an object. Parking slots are quadrangles but are sometimes not rectangular and may be rotated or some areas may be truncated. Our detector

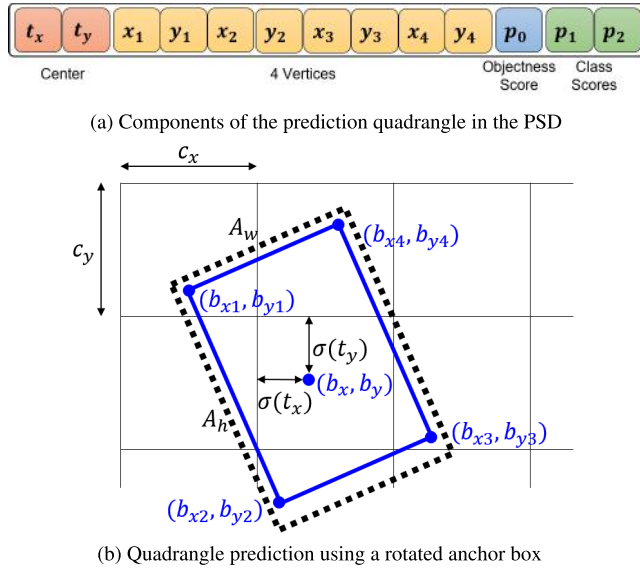


FIGURE 3. Components of the prediction box and the quadrangle using them. In (b), the black dotted line implies the rotated anchor box, and the blue solid line implies the parking slot.

estimates the coordinates of the four vertices of the parking slot to accurately depict the location of the parking slot even in the above case. The components depicting a quadrangle predicted by the detector are designed as shown in Figure 3a.

In object detectors, anchor boxes are used to select the most accurate coordinates among the various sizes of candidates for an object. In the common object detectors, horizontal bounding boxes are estimated using horizontal anchor boxes. In contrast, in this paper, the anchor box is rotated to detect the rotated parking slot more accurately. The rotation angle of an anchor box is estimated by the PCR and denoted by $\hat{\theta}$. The rotated anchor box is expressed by (b_x, b_y) and (b_{xi}, b_{yi}) , which denote the coordinates of the center and the i -th vertices of the parking slot, respectively. The coordinates are calculated as

$$\begin{aligned} \begin{bmatrix} b_x \\ b_y \end{bmatrix} &= \begin{bmatrix} \sigma(t_x) \\ \sigma(t_y) \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix}, \\ \begin{bmatrix} b_{xi} \\ b_{yi} \end{bmatrix} &= \begin{bmatrix} b_x \\ b_y \end{bmatrix} + \begin{bmatrix} \cos \hat{\theta} & -\sin \hat{\theta} \\ \sin \hat{\theta} & \cos \hat{\theta} \end{bmatrix} \begin{bmatrix} A_w \tanh x_i \\ A_h \tanh y_i \end{bmatrix} \\ (i &= 1, 2, 3, 4), \end{aligned}$$

where t_x, t_y are the feature data for estimating the center coordinates of the parking slot; x_i, y_i are the feature data for estimating the coordinates of the vertices of the parking slot; c_x, c_y are the base coordinates of the grid; and A_w, A_h are the width and height of the anchor box, respectively. These notations are illustrated in Figure 3b. By rotating the anchor box, the appearances of the various parking slots become similar, as shown in Figure 4.

Fine-tuned detectors are used for each of the three types of parking slots, and the detectors are not operated when the parking type is classified as *not-parking-space*. Each detector has unique weights for each parking slot type and is operated only when the corresponding type is activated by the PCR.

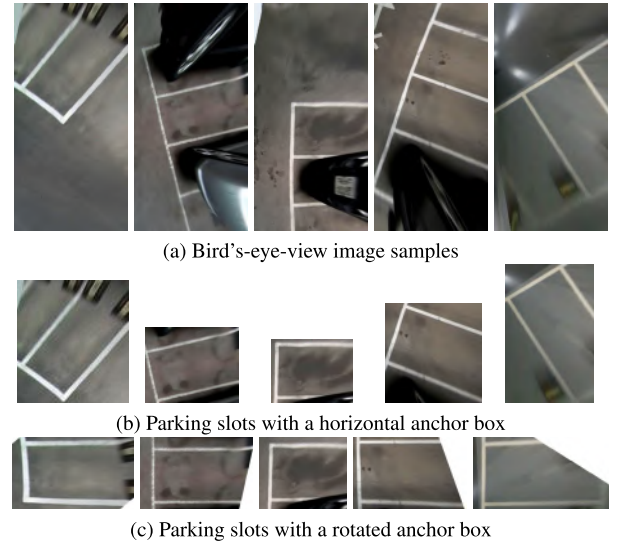


FIGURE 4. Example images of rotated parking slots having a similar appearance due to the rotated anchor.

For the inference, a rotated non-maximum suppression (rNMS) method is designed to be suitable for the proposed PSD. First, we remove the parking slot proposals that have lower objectness scores than the threshold Th_{obj} . We then calculate the IoU between the remaining proposals and remove the overlapped proposals. However, calculating the IoU between rotated quadrangle pairs is complicated because the intersection is not a simple rectangle and has a variety of polygonal shapes [9]. To cope with this difficulty, we rotate all proposals in the opposite direction by the angle (ϕ in Algorithm 1) of the proposal with the highest objectness score (b^* in Algorithm 1) in the image. Then, we generate the rectangle enclosing each rotated proposal and apply the NMS to the generated rectangles with Th_{nms} as the IoU threshold. Finally, the resulting rectangle by the NMS is rotated back to its original coordinates to get the final result. By using this method, the general NMS algorithm already ported to the embedded system can be used without performance degradation. If there are N proposals with objectness scores greater than Th_{obj} in the image and the coordinates of the i -th vertex of the n -th proposal are $b_i^n = (b_{xi}^n, b_{yi}^n)$, the process of obtaining the detection result B_{nms} by using the rotated NMS is shown in Algorithm 1.

IV. DATASET

A. LABEL INFORMATION AND ACQUISITION METHOD

The dataset includes label information for each image and label information for each parking slot in the image. Each image has a parking slot type and an angular value of the parking slot orientation as its label information. Each parking slot has coordinates of four vertices and availability as its label information. The coordinates of each parking slot are within the image resolution range. The parking slot is labeled as *non-available* when the vehicle can not be parked in the

Algorithm 1 Rotated Non-Maximum Suppression**Input:** Detection Proposal Set $B = \{b^1, b^2, \dots, b^N\}$ **Output:** Result Detection Set B_{nms}

```

1: function rNMS( $B$ )
2:    $b^* = \operatorname{argmax}(B)$ 
3:    $\phi = \arctan(\frac{b_{y3}^* + b_{y4}^* - b_{y1}^* - b_{y2}^*}{b_{x3}^* + b_{x4}^* - b_{x1}^* - b_{x2}^*})$ 
4:    $R = \{r^1, r^2, \dots, r^N\}$  where  $r^n = \operatorname{Rotate}(b^n, -\phi)$ 
5:    $R_{nms} = \operatorname{NMS}(R)$ 
6:   For  $R_{nms} = \{m^1, m^2, \dots, m^K\}$ 
7:      $s^k = \operatorname{Rotate}(m^k, \phi)$ 
8:   return  $B_{nms} = \{s^1, s^2, \dots, s^K\}$ 
9: end function

```

parking slot due to obstacles or a sign of prohibited parking is present in the parking slot.

The image acquired by the fish-eye camera mounted on the side mirror of a vehicle is transformed into a bird's-eye-view image and used as a data sample. The images from the right camera are used as the basis for the dataset, and the images from the left camera are used after rotating them 180 degrees to fit the right image. The spatial resolution of each side image is 768×256 pixels, corresponding to a $14.4 \text{ m} \times 4.8 \text{ m}$ flat physical region. The length of one pixel on the bird's-eye-view image corresponds to 1.875 cm on the physical ground. Using a single image has several advantages in contrast to using a synthesized image of multiple AVM camera images. In the case of using multiple camera images, the detection algorithm can only work after the vehicle's main chipset has synthesized the images from the multiple cameras, and this synchronization imposes a computational burden that causes a time delay. In contrast, if a single camera image is used, the detection algorithm immediately works in each individual camera module. In addition, the synthesized image may contain image distortion, for example, the parking line may appear disconnected when the ground is not flat [10]. In a single camera image, this kind of image distortion is not present.

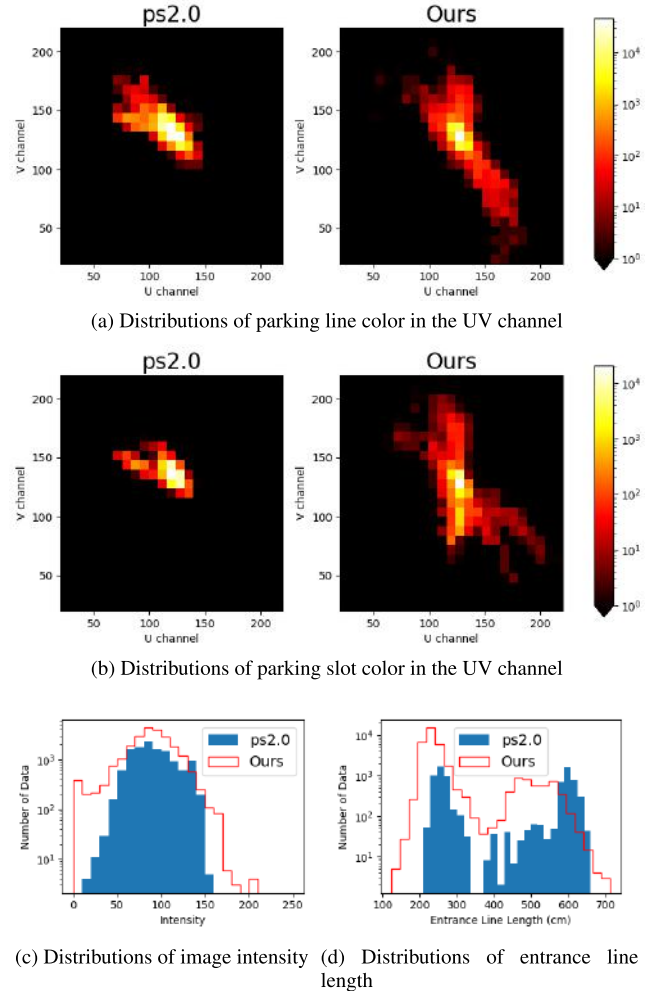
We used three types of vehicles to acquire images for our dataset: two sedans and one sport utility vehicle (SUV). The cameras on each vehicle had different extrinsic parameters, intrinsic parameters, and image resolutions. By using various types of vehicles to acquire the images, the dataset could be constructed without being dependent on a specific camera module or a vehicle. Therefore, our dataset is more useful for developing algorithms for various types of vehicles.

B. NOVELTY OF THE DATASET

Our dataset consists of 22817 images, of which 18299 are for training and 4518 for testing. The total number of images in our dataset is 1.87 times that of ps2.0, the number of scenes is 3.4 times, and the number of parking slots is 5.3 times as shown in Table 1. When acquiring data samples, a series of frames were taken, and some of them were manually labeled and used as data samples. A scene refers to a series of frames

TABLE 1. Numerical comparison between ps2.0 and our dataset.



















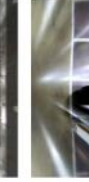












| | ps2.0 | Ours |
|---|-------|-------|
| training images | 9827 | 18299 |
| testing images | 2338 | 4518 |
| training scenes | 166 | 571 |
| available parking slots | 7923 | 35829 |
| non-available parking slots | 3726 | 26600 |
| images with parallel parking slots | 4643 | 6002 |
| images with perpendicular parking slots | 3359 | 12846 |
| images with diagonal parking slots | 218 | 1436 |

**FIGURE 5.** Comparison of image diversity between ps2.0 and our dataset.

taken over a period of time, and the dataset consists of multiple scenes in order to depict various parking slots in different environments. When we split the dataset into a training set and a testing set, it is not split randomly by image units but is split randomly by scene units so that images of the same scene should not be included in both the training and testing sets. This split intends to prevent over-estimating the performance on scenes included in both training and testing sets. Actually, when the training and testing sets were randomly split by image units, the performance was too high even with a small number of training iterations.

As shown in Figure 5, our dataset samples are more diverse than the ps2.0 dataset samples. Figure 5a shows the

TABLE 2. Parking slot attributes, external conditions, and sample images.

| Parking Slot Layout | | | | | Parking Line Style | | | | Parking Line Color | | | |
|--|--|--|--|--|--|--|--|--|--|--|--|--|
| Closed | *Open | Partially Open | *Single Separating | *With Mark | Solid | *Double | *Striped | Damaged | White | Yellow | *Pink | *Blue |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| Parking Slot Color | | | | | Non-available Slots | | | Ground Condition | | | | |
| Gray | *Green | *Red | *Blue | Orange | *Pink | Parked | *Standing Sign | *Ground Sign | Asphalt | Reflective | Brick | Grass |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| Ground Obstacle | | | | | Standing Obstacle | | | Environment | | | | |
| Shadow | *Old Line | Puddle | *Fallen Leaves | *Soil | *Paint Crack | Sewer Drain | Vehicle | Pillar | Human | Day | Night | Indoor |
|  |  |  |  |  |  |  |  |  |  |  |  |  |

* mark indicates a sample case which the ps2.0 [3] dataset does not contain.

distributions of the color of the parking line for ps2.0 and our dataset, and Figure 5b shows the distributions of the colors inside the parking slot. Our dataset contains data samples of various colors, but most of the parking slot colors in the ps2.0 dataset are achromatic, with U and V values close to 128. Figure 5c shows the image intensity distributions for ps2.0 and our dataset, and the intensity is calculated based on the median of all pixels in the image. Our dataset contains images with more varied brightness, from very dark to very bright, compared to those in the ps2.0 dataset. Figure 5d shows the distributions of the lengths of the parking slot entrance lines for ps2.0 and our dataset. Our dataset contains parking slots with more varied sizes compared to the ps2.0 dataset.

Table 2 shows various categories of the parking slots and sample images of each category. The categories are clustered according to multiple criteria based on the parking slot attributes and the external conditions. The slot attributes consist of parking slot layout, line style, line color, slot color, and availability. The external conditions consist of ground conditions, obstacles, and environments. As shown in Table 2, our dataset contains diverse data samples that are not included in the ps2.0 dataset. In an autonomous parking system, it is necessary to decide whether it is possible to park in a detected parking slot. To cover this situation, our dataset contains data samples of non-available parking slots which appear to be

parking slots but are not available for parking because they are already occupied or parking is prohibited.

C. EVALUATION METRIC

IoU, which is the most general evaluation metric for object detection, is not suitable for the parking slot detection problem. Even when the predicted parking slots have the same IoU values, how well the vehicle can be parked in a parking slot can be quite different as shown in Figure 6. To mitigate this problem, we propose a new evaluation metric, referred to as a *parking score*, that reflects whether a vehicle can be parked within the parking slot.



FIGURE 6. Examples of the predicted parking slot, its ground truth, and the vehicle which is parked at the center of the predicted parking slot. The black dotted line implies the ground truth, and the blue square implies the predicted parking slot. All four examples have the same IoU value of 0.8.

The *parking score* S for a parking slot is calculated by multiplying the two scores: area score S_{Area} (on how accurately the area of the parking slot is estimated) and location score S_{Loc} (on how accurately the location is estimated). The two scores are estimated using the coordinates of the ground truth parking slot G and the predicted parking slot P . The area score

is calculated by comparing the areas of G and P , that is,

$$S_{\text{Area}} = \frac{\min\{\text{Area}(G), \text{Area}(P)\}}{\max\{\text{Area}(G), \text{Area}(P)\}}. \quad (1)$$

The closer the two areas are to being the same size, the closer the score is to the value 1. The location score is a value to measure how much of the area of P is inside the area of G , which is calculated as

$$S_{\text{Loc}} = \sqrt{\text{Area}(P')/\text{Area}(P)}. \quad (2)$$

where P' is a scale-downed location of P so that $P' \subset G$. If P is completely contained inside G , S_{Loc} has the maximum value of 1. If parking score S exceeds a threshold, the prediction is judged as TRUE; otherwise, it is FALSE. Precision, recall, and average precision for the entire testing set are then calculated based on the TRUE and FALSE judgments. The parking score calculation process is summarized in Algorithm 2.

Algorithm 2 Parking Score Calculation

Input: Ground Truth Parking Slot G , Predicted Parking Slot P

Output: Parking Score S

```

1: function CalcParkingScore( $G, P$ )
2:    $c = \text{Centroid of } P$ 
3:   if  $c \in G$  then
4:      $S_{\text{Area}} = \frac{\min\{\text{Area}(G), \text{Area}(P)\}}{\max\{\text{Area}(G), \text{Area}(P)\}}$ 
5:     Scale-down  $P$  to  $P'$  until  $P' \subset G$ 
6:      $S_{\text{Loc}} = \sqrt{\text{Area}(P')/\text{Area}(P)}$ 
7:      $S = S_{\text{Area}} \times S_{\text{Loc}}$ 
8:   else
9:      $S = 0$ 
10:  end if
11:  return  $S$ 
12: end function
  
```

This scoring process is highly related to the actual parking problem. Assuming that the vehicle will be parked in the center of P , the location score represents the relative size of the vehicle that can be parked inside G based on the coordinates of P . For example, if the width of the parking slot is 2.5 m and the width of the vehicle is 2 m, a vehicle can be parked completely inside the parking slot only when S_{Loc} is greater than 0.8.

If the predicted parking slot is larger than the real parking slot, an accident can occur while parking. Meanwhile, if the predicted parking slot is smaller than the real parking slot, there is a disadvantage of wasting time since there could be no attempt to park in an acceptable slot. Therefore, a parking slot predicted to be larger than the ground truth, which can cause an accident, should be scored lower than one predicted to be smaller. To reflect this case, P is scale-downed to P' to be $P' \subset G$ in (2), which leads to a location score less than 1 when the parking slot size is overestimated unlike the case in which the parking slot size is estimated smaller than the ground truth

(having a location score of 1). In both of the above cases, the area score is the same with a value less than 1 and so gives just a penalty since the size of the predicted parking slot is not accurate.

For every parking slot predicted from the testing set images, the parking score is calculated by the proposed metric, and if the score exceeds Th_{score} , the prediction is determined TRUE. There may be multiple parking slots in the image, but the parking slots do not overlap each other, so each predicted parking slot corresponds to only one real parking space. An automatic parking system needs to search for an available parking slot in order to park the vehicle. Therefore, we calculate the scores only for the parking slots which are labeled as *available* and judge whether they are TRUE or FALSE. The performance of the algorithms are then compared by precision, recall, and average precision.

The size of vehicles varies widely from minicompacts to large SUVs, and the size of parking slots can vary from country to country. When developing an autonomous parking system for a vehicle, the vehicle size and the target parking slot size can be provided. The size ratio between the vehicle and the parking slot can be used as Th_{score} to measure the performance of the parking slot detection algorithm for the vehicle. That is, the algorithm may be further developed using a threshold value suitable for the vehicle and the environment.

V. EXPERIMENTS

A. IMPLEMENTATION DETAILS

The proposed method was quantitatively evaluated using our dataset and the ps2.0 dataset. Since the characteristics of the two datasets are different, the implementation was slightly different for each dataset. We augmented the training set by rotating each original image to generate a number of its rotated versions. For our dataset, 22 images per one original image were obtained by flipping the original image vertically and rotating the original image from -5 to 5 in 1 degree units. For the ps2.0 dataset, 24 images per one original image were obtained by rotating the original image from 0 to 345 in 15 degree units.

For the PCR, the size of the input image is 192×64 for our dataset and 128×128 for the ps2.0 dataset. The weights of the PCR were optimized by the adam optimizer whose learning rate, β_1 , β_2 , and ϵ were set to 0.001, 0.9, 0.999, and 10^{-8} , respectively. The PCR was trained for 50 epochs, and the batch size was set to 64.

The size of the input image for the PSD was 256×768 for our dataset and 416×416 for the ps2.0 dataset. We implemented our detector based on the YOLOv3 [8] architecture using darknet-53 as the backbone. The weights of the PSD were optimized by a momentum optimizer whose momentum was set to 0.9. The PSD was first trained for 10 epochs with the entire dataset and then fine-tuned for 10 epochs for each type of detector. Each detector was trained with a learning rate of 10^{-4} for an initial 6 epochs, then 3×10^{-5} for 2 epochs, and 10^{-5} for the last 12 epochs. The hyper-parameters for training and inference were $\lambda = 0.1$, $Th_{\text{obj}} = 0.3$, and $Th_{\text{nms}} = 0.1$.

All the experiments were conducted using Tensorflow on a PC with an Intel Core i7-7700 CPU @ 3.60 GHz, one NVIDIA GeForce GTX 1080 card, and 16 GB RAM. Each backbone of the PCR and the PSD were pre-trained on ImageNet [11].

B. PARKING CONTEXT RECOGNIZER

In this experiment, we evaluated the performance of the PCR using various networks as the backbone. The evaluation was conducted on our dataset. As shown in Table 3, MobileNetV2 [7] showed reasonable performance with the fastest processing time and the smallest model size. The classification accuracy and orientation error of the MobileNetV2-based model were 98.38% and 1.38 degrees, respectively. The MobileNetV2-based models had slightly lower classification accuracy than the other backbone-based models but had a similar overall performance. In our method, since the PCR module is always running, a backbone with a small amount of computational power is suitable. In addition, the size of the MobileNetV2-based model was the smallest, so it was relatively faster than the other models when ported to the embedded systems with limited memory buses. Hence, we adopted the MobileNetV2-based model with the fastest operating time of 4.52 ms and the smallest model size of 29.3 MB.

TABLE 3. Parking slot classification and angle regression performance on our dataset.

| Method | Type Accuracy | Orientation Error (degree) | Time (ms) | Model Size (MB) |
|--------------|---------------|----------------------------|-----------|-----------------|
| MobileNetV2 | 98.38% | 1.38 ± 4.10 | 4.52 | 29.3 |
| VGG-16 | 98.56% | 1.74 ± 4.62 | 4.95 | 169.9 |
| DenseNet-121 | 98.74% | 1.31 ± 3.72 | 11.47 | 82.9 |
| Resnet-50 | 98.54% | 1.44 ± 4.16 | 9.03 | 275.5 |

TABLE 4. Performance evaluation of parking slot detection on our dataset.

| Method | precision(%) | recall(%) | mAP(%) | Time(ms) |
|-------------|--------------|-----------|--------|----------------|
| VPS-Net [5] | 74.16 | 75.14 | 64.99 | 58.59 |
| Ours | 87.75 | 88.52 | 82.17 | 42.79 |
| | | | | (4.52 + 38.27) |

C. PARKING SLOT DETECTION

Table 4 shows the parking slot detection performances of the proposed method and VPS-Net [5] on our dataset. The detection result of VPS-Net was obtained from the publicly available code released by its authors. The parking score evaluation threshold Th_{score} was set to 0.8. We compared the performance of our method with VPS-Net [5] since the other deep-learning-based methods suggested in [3] and [4] could not classify available parking slots and non-available parking slots. As shown in Table 4, our method outperformed VPS-Net. The mean average precision of our method was 82.17%, in contrast, that of VPS-Net was 64.99%. The operation time of our method was 42.79 ms, which was faster than the 58.59 ms of VPS-Net. Since the operating time of the PCR was 4.52 ms, only a small amount of computational power was used when there were no parking slots. Figure 7 shows the parking slot detection results of the proposed

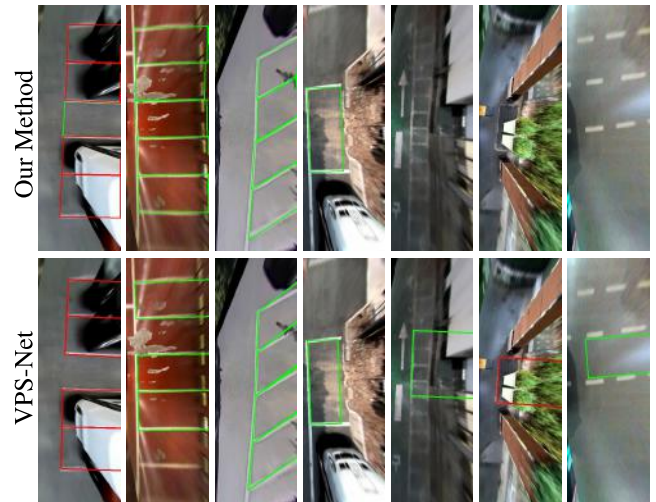


FIGURE 7. Examples of parking slot detection. The green box indicates an available parking slot, and the red box indicates a non-available parking slot.

method and VPS-Net in various situations. It shows that the proposed method robustly detected parking slots of various types and filtered out the not-parking-slots that looked similar to parking slots. In contrast, VPS-Net, which is a junction-based method, had low detection performance for parking slots without junctions and yielded false positives for not-parking-slots similar to parking slots.

TABLE 5. Performance evaluation of parking slot detection on the ps2.0 dataset.

| Method | precision(%) | recall(%) |
|-------------|--------------|-----------|
| DeepPS [3] | 97.26 | 96.63 |
| VPS-Net [5] | 98.34 | 98.25 |
| Ours | 98.70 | 97.88 |

Table 5 shows the parking slot detection performances on the ps2.0 dataset. In the previous papers [3], the handcrafted feature-based methods were not compared since they are inferior to the deep-learning-based methods. A parking slot detection was considered TRUE if the two junctions and orientations were estimated within 20 pixels and 2 degrees of error from the ground truth parking slot, respectively. The tolerance for rotation error of 10 degrees used in the previous paper was reduced to 2 degrees because it was too large to be used in a real parking problem. The performance of our method was 98.70% precision and 97.88% recall, which was comparable to VPS-Net with 98.34% precision and 98.25% recall and better than DeepPS with 97.26% precision and 96.63% recall.

As seen in the results of the above two experiments, although our method showed comparable performance to the previous methods for ps2.0, it outperformed the recent deep learning method (VPS-Net) by a large margin on our dataset containing many realistic parking slot images.

D. ABLATION STUDY

We conducted several ablation experiments on our dataset to show the validity of our method. Four ablation factors were

TABLE 6. Ablation study of our parking slot detection method.

| Method | precision(%) | recall(%) | mAP(%) |
|------------------------|--------------|-----------|--------|
| w/o rotated anchor box | 85.40 | 86.87 | 80.33 |
| w/o rNMS | 88.81 | 84.72 | 79.03 |
| w/o type classifier | 86.74 | 88.07 | 80.70 |
| Full | 87.75 | 88.52 | 82.17 |

selected from our method and the effects of the factors were experimentally examined as shown in Table 6: (1) rotated anchor box, (2) rNMS, and (3) type classifier.

When the rotated anchor box was not used, we estimated the four vertices of the parking slot using the horizontal anchor box. When the rNMS was not used, we applied the general NMS to the horizontal bounding boxes enclosing the rotated proposals with the same IoU threshold Th_{nms} . When the type classifier was not used, a single detector worked for all images without pre-classifying the parking slot type and without using the fine-tuned detectors by type.

As shown in Table 6, the rotated anchor box improved the precision from 85.40% to 87.75% and the recall from 86.87% to 88.52%. This means that the rotated anchor box reduced both the false positives and false negatives despite the orientation error of the PCR. The rNMS improved the recall from 84.72% to 88.52% and the average precision from 79.03% to 82.17%. By using the rNMS, the correctly predicted parking slots were less filtered. The type classifier improved the average precision from 80.70% to 82.17% since the prediction accuracy was enhanced while fine-tuning. Also, the type classifier improved the precision from 86.74% to 87.75% and the recall slightly since it filtered out the *not-parking-space* class. Through this ablation study, it can be seen that the context information used in our method contributed to the performance improvement of the PSD as intended.

VI. CONCLUSION

In this paper, we have proposed a context-based parking slot detection method inspired by the process of a human driver finding a parking slot. In addition, we have released a realistic parking slot dataset, which comprises 22817 images of parking slots having various attributes and external conditions. We have also presented a new evaluation metric for parking slot detection, which is fit for an actual parking problem. As validated by comparison and ablation study in experiments, our method outperformed the previous deep-learning-based method, along with a short operation time. Regarding the advantages of our work, the context information (type, orientation of the parking slot) recognized by our method can improve the performance and efficiency of the detector. By using the type information, the parking-slot-like spaces that are actually not parking slots are filtered out, and by using the orientation information, the anchor boxes are rotated to increase the detection performance of the rotated parking slots. The released dataset will promote future work in the autonomous vehicle research community, and the new evaluation metric will be able to contribute to research applicable to actual environments.

REFERENCES

- [1] J. K. Suhr and H. G. Jung, "Fully-automatic recognition of various parking slot markings in around view monitor (AVM) image sequences," in *Proc. 15th Int. IEEE Conf. Intell. Transp. Syst.*, Sep. 2012, pp. 1294–1299.
- [2] 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.
- [3] L. Zhang, J. Huang, X. Li, and L. Xiong, "Vision-based parking-slot detection: A DCNN-based approach and a large-scale benchmark dataset," *IEEE Trans. Image Process.*, vol. 27, no. 11, pp. 5350–5364, Nov. 2018.
- [4] J. Huang, L. Zhang, Y. Shen, H. Zhang, S. Zhao, and Y. Yang, "DMPR-PS: A novel approach for parking-slot detection using directional marking-point regression," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2019, pp. 212–217.
- [5] W. Li, L. Cao, L. Yan, C. Li, X. Feng, and P. Zhao, "Vacant parking slot detection in the around view image based on deep learning," *Sensors*, vol. 20, no. 7, p. 2138, Apr. 2020.
- [6] 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 (ICME)*, Jul. 2017, pp. 649–654.
- [7] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*. [Online]. Available: <http://arxiv.org/abs/1704.04861>
- [8] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*. [Online]. Available: <http://arxiv.org/abs/1804.02767>
- [9] J. Ma, W. Shao, H. Ye, L. Wang, H. Wang, Y. Zheng, and X. Xue, "Arbitrary-oriented scene text detection via rotation proposals," *IEEE Trans. Multimedia*, vol. 20, no. 11, pp. 3111–3122, Nov. 2018.
- [10] S. Lim, S. Lee, J.-G. Kim, and D. Lee, "Adaptive on-line calibration for around-view monitoring system using between-camera homography estimation," *J. Appl. Remote Sens.*, vol. 12, no. 1, 2018, Art. no. 015014.
- [11] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.



HOSEOK DO received the B.S. degree in electrical engineering and the M.S. degree in computer science from Seoul National University, Seoul, South Korea, in 2007 and 2009, respectively, where he is currently pursuing the Ph.D. degree in electrical and computer engineering.

Since 2009, he has been with LG Electronics, Seoul, where he is also a Senior Research Engineer. His current research interests include object detection, computer vision, machine learning, deep learning, and their applications.



JIN YOUNG CHOI (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in control and instrumentation engineering from Seoul National University, Seoul, South Korea, in 1982, 1984, and 1993, respectively.

From 1984 to 1989, he was with the Electronics and Telecommunications Research Institute (ETRI), Daejeon, South Korea, where he was involved in the Project of Switching Systems. From 1992 to 1994, he was with the Basic Research Department, ETRI, where he was a Senior Member of Technical Staff involved in the neural information processing system. From 1998 to 1999, he was a Visiting Professor with the University of California at Riverside, Riverside, CA, USA. Since 1994, he has been with Seoul National University, where he is currently a Professor with the School of Electrical Engineering. He is also with the Automation and Systems Research Institute, Engineering Research Center for Advanced Control and Instrumentation, and the Automatic Control Research Center, Seoul National University. His current research interests include adaptive and learning systems, visual surveillance, motion pattern analysis, object detection, object tracking, and pattern recognition.

• • •