2024 IEEE Intelligent Vehicles Symposium (IV)
June 2-5, 2024. Jeju Island, Korea

SAM-PS: Zero-shot Parking-slot Detection based on Large Visual Model

Heng Zhai[1,2,3,*], Jilin Mei[1,†], Liang Chen[1], Fangzhou Zhao[1], Xijun Zhao[5,6], Yu Hu[1,4,†]

2024 IEEE Intelligent Vehicle Symposium (IV) | 979-8-3503-4881-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/IV55156.2024.10588736

Abstract— **Large visual models have recently demonstrated their promising performance on zero-shot transfer. However, so far, none of the existing methods explicitly possess the ability to perform zero-shot transfer on parking-slot detection, which results in current deep-learning based methods relying on training datasets, and methods based on traditional computer vision exhibiting poor robustness. In this paper, we propose a large visual model-based parking-slot detection method, which utilizes a large visual model (segment anything) to segment an around-view image and infer parking-slots by analyzing the relationship of marking-points in masks. In addition, we classify real-world parking-slots into two categories, line-based and area-based. The proposed method employs a two-stage approach which has a manually designed post-processing step without training. Multiple experiments have been carried out on public benchmarks, and our method demonstrates the capability for zero-shot transfer. The code will be released at https://github.com/Zhai0123/SAM-PS.**

I. Introduction

The automatic parking system is a significant part of autonomous vehicles, where parking-slot detection stands out as a crucial component [1]. In contrast to traditional manual parking, automatic parking system can offer a more dependable trajectory for parking and control, thereby reducing the risk of damage due to errors in manual operation. In implementing the automatic parking system, one of core difficulties is how to detect and locate parking-slots accurately with sensors. Methods can be broadly categorized into two types: traditional computer vision-based methods and deep learning-based methods [2].

In recent years, a growing trend of vehicles are being equipped with around-view monitor systems, intended to enhance the observation of surrounding environments [3]. Most of the early vision detection methods were predominantly based on traditional computer vision, involving the extraction of parking line features and the addition of manual [4]–[6]. However, parking-slot and lane markings detected through traditional computer vision-based methods are susceptible to light changes and fading, resulting in poor robustness.

Deep learning have made significant achievements in computer vision. The exceptional performance of deep learning has established it as the preferred perceptual method in autonomous driving. Parking-slot detection methods utilizing deep convolutional neural networks have demonstrated high accuracy [7]–[9]. Currently, deep learning-based methods are categorized into two types. One is a two-stage process, which first detects the marking-point through a CNN, then uses manually designed geometric constraints for post-processing to predict the parking-slot. The other is an end-to-end approach [10], which uses a model to integrate feature extraction and inference of parking-slots into a one-stage model that can directly output the predicted parking-slots. However, deep learning-based methods often exhibit poor generalization across different scenes.

Adapting the “Segment Anything Model” (SAM) [11] presents a potential solution to mitigate these issues in the field of parking-slot detection. SAM is a robust image segmentation model trained on a vast dataset, enabling it to exhibit noteworthy zero-shot generalization ability. However, despite SAM has strong zero-shot capabilities in image segmentation, its raw output can not be directly applied to parking-slot detection.

Therefore, we propose a parking-slot detection method utilizing SAM. This approach employs SAM to segment the around-view image and post-process masks to identify the parking-slots. Owing to its zero-shot transfer capabilities, the proposed model does not require training and demonstrates robust generalization across diverse scenarios. The method divides the task into two stages, as depicted in Fig. 1. In the first stage, features are extracted from the around-view image. The second stage involves filtering out irrelevant masks from the multi-mask output and extracting parking-slot features, followed by matching different corner types based on predefined rules, akin to traditional computer vision methods [5]. In short, the highlights of our work are summarized in the following points:

- To author’s knowledge, this work represents the first application of the large visual model SAM in parking-slot detection, incorporating a filter strategy to eliminate extraneous information and enhance performance.
- We introduce a novel two-stage parking-slot detection method without training.
- Our proposed method, SAM-PS, has undergone testing on public parking-slot datasets PS2.0 [7] and PSDD [8], demonstrating a zero-shot transfer ability that significantly surpasses existing deep learning-based baselines.

‡This work was supported by National Natural Science Foundation of China under Grant No.62203424, No.62176250 and No.U23B2034, and the Collective Intelligence & Collaboration Laboratory (Open Fund Project No.QXZ23013301)

[1]Research Center for Intelligent Computing Systems, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100190, China.
[2]School of Information Science and Technology, ShanghaiTech University, Shanghai, 201210, China.
[3]Shanghai Innovation Center for Processor Technologies(SHIC).
[4]School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing, 100190, China.
[5]China North Artificial Intelligence & Innovation Research Institute.
[6]Collective Intelligence & Collaboration Laboratory (CIC).
[*]Work done as an intern at Institute of Computing Technology, Chinese Academy of Sciences.
[†]Correspondence: Jilin Mei, Yu Hu, {meijilin, huyu}@ict.ac.cn

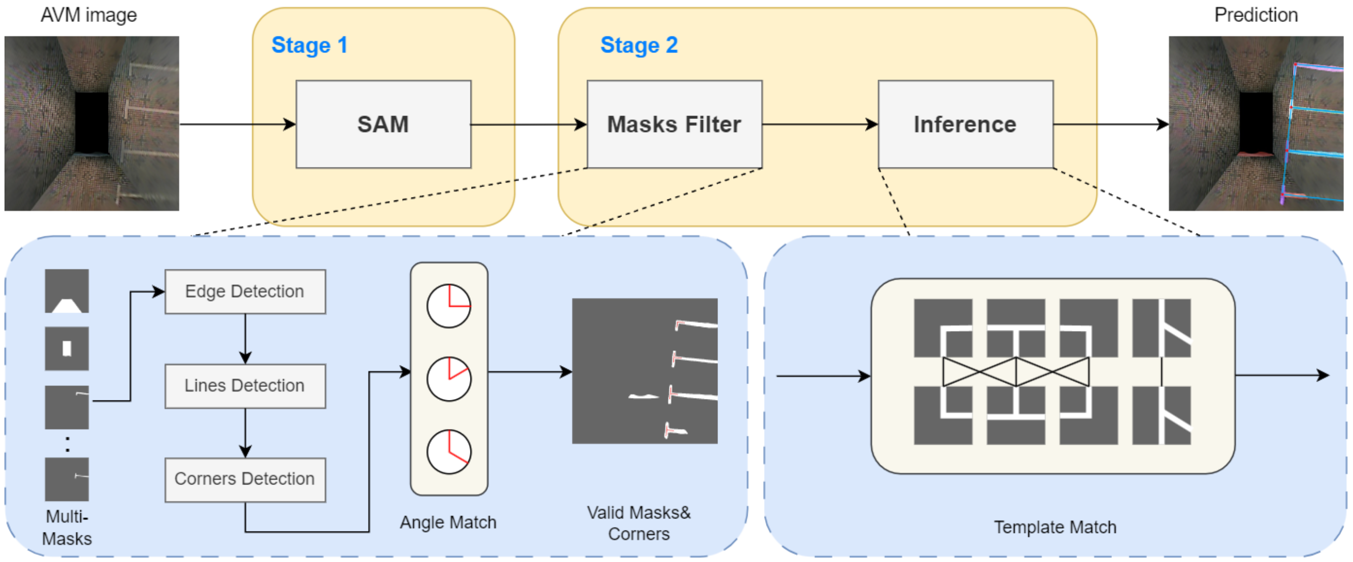

Fig. 1. The architecture of the presented SAM-PS. This model includes two stages. The first stage utilizes the SAM model as a backbone network to generate multiple masks. In the second stage, the multi-masks filter will remove masks without parking-slot information and extract the feature of marking-points, and then locate the parking-slot using template matching techniques. Finally the accurate positions of parking-slot in the around-view images are detected.

II. Related Work

A. Traditional computer vision-based detection

Parking-slot detection methods utilizing traditional computer vision techniques have undergone extensive research for decades [4]. Parking-slot detection approaches of traditional computer vision can be classified as line-based and corner-based approaches. Within line-based approaches, parking lines are initially detected in around-view image with various edge detection algorithms, followed by the prediction of line parameters using line detection algorithms to fit lines [12], then geometric constraints that have been manually formulated are utilized to infer the parking-slot. Similarly, corner-based approaches begin by identifying corners in the around-view image through a Harris corner detector [5], followed by parking-slot localization using template matching techniques [13]. While these traditional methods often yield effective results, they exhibit sensitivity to environmental changes and are less suited for complicated and changeable real-world environments.

B. Deep learning-based detection

In recent years, there has been a surge in proposing deep learning-based methods to detect parking-slot [14]. DeepPS [7] firstly represents the benchmark in deep learning-based approach. This method detects marking-points in around-view images by employing a CNN, followed by the use of another CNN to pair these marking-points, and it can be classified as a two-stage method. DMPR-PS [15] also belongs to a two-stage approach. In its first stage, DMPR-PS utilizes a CNN architecture for calculating the shape, position and orientation of marking-points, while in second stage, it applies manually designed templates to remove and match pairs of marking-points. Suhr et al. [16] were the first to propose a one-stage and end-to-end parking-slot detection approach. The GCN-based method [10], another end-to-end approach, employed a graph convolutional network (GCN) to analyze adjacent information and ascertain parking-slot locations directly. Bui et al. [17] proposed an end-to-end method based on transformer. These models, trained and tested on the open-source dataset PS2.0 [7], exhibit diminished detection performance when applied to the PSDD dataset [8]. These deep learning-based methods depend on the training set and tend to overfit. Our proposed model, originating from advancements in deep learning, primarily leverages the zero-shot transfer capabilities of SAM, enabling superior performance without reliance on specific dataset distributions.

C. Segment Anything Model

Adapting the “Segment Anything Model” (SAM) [11] represents a promising solution to these challenges in the field of parking-slot detection. SAM is a robust image segmentation model trained on a large-scale SA-1B dataset. However, despite its satisfying performance, SAM encounters a major limitation: its lack of real-time processing capability. This limitation constrains its broad applicability in scenarios demanding immediate segmentation results. Additionally, recent advancements like Fast-SAM [19], employing trainable YOLO [20] in prompt section, significantly enhance SAM’s inference speed. EfficientSAM [21] introduces SAM-leveraged masked image pre-training (SAMI), generating optimal pre-trained lightweight ViT backbones for varied segmentation tasks. Nevertheless, none of these models are directly applicable for parking-slot detection. Therefore, our approach is possible the first application of SAM in parking-slot detection, utilizing its zero-shot transfer ability to address limitations in feature extraction and robustness.

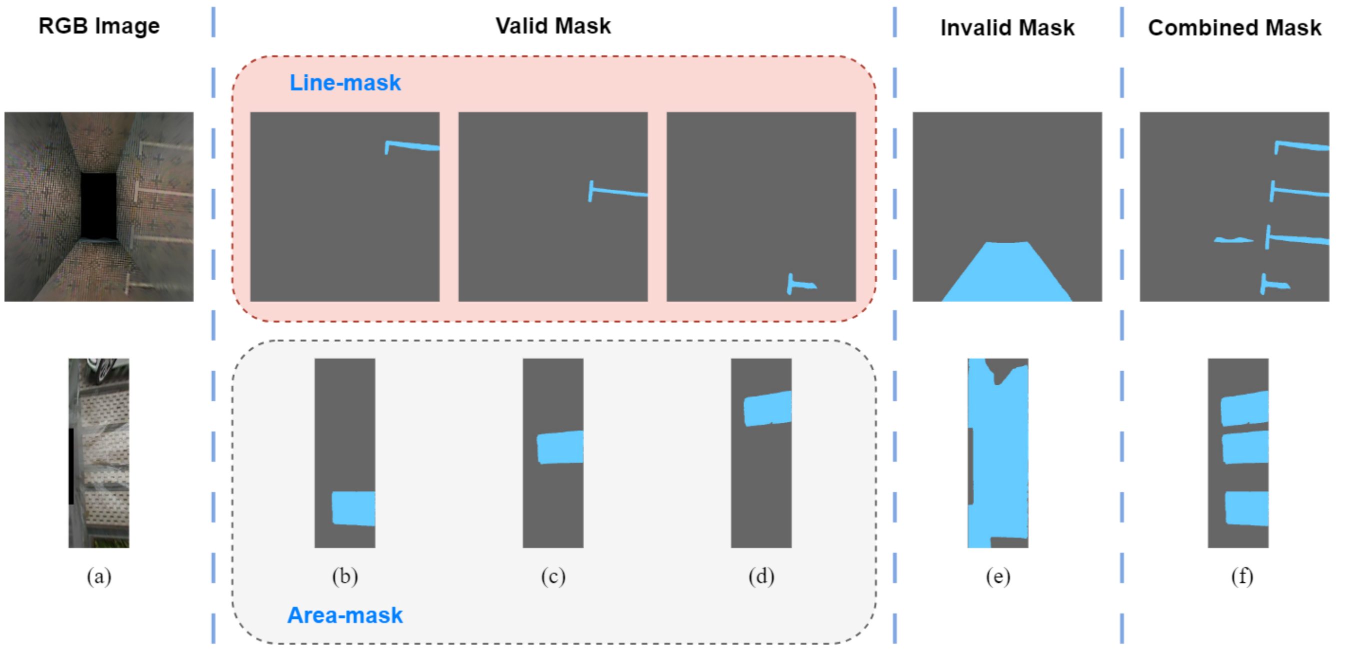

Fig. 2. Example masks generated by SAM, mainly in two types of valid masks, line-mask and area-mask. The first row represents masks from an image from PS2.0 [7], [18] with line information. The second row represents masks of an image from brick type in PSDD [8], with area information, without line information. (a) The raw birds'-eyes view images as input of SAM. (b)-(d) Valid masks generated by SAM, including parking line-masks and parking area-masks. (e) Invalid masks generated by SAM, without useful parking-slot information. (f) Valid mask combination.

III. PROPOSED METHOD

In this section, the details of the proposed parking-slot detection method SAM-PS will be presented. The framework of SAM-PS, as shown in Fig. 1.

A. Segment Anything Model(SAM)

SAM employs the MAE pre-trained ViT [22] for processing high-resolution inputs. The image encoder processes inputs at a resolution of 1024×1024.The mask decoder effectively maps image embeddings and outputs tokens into masks. The transformer decoder block and dynamic mask prediction head are modified according to the form of large visual model SAM. For parking-slot detection, manual prompts are not required for the mask decoder. Instead, an evenly spaced grid is selected within the image, utilizing all its points as prompts for segmenting the entire image. Furthermore, as this is an instance segmentation task without semantic information, each pixel may correspond to multiple instances and belong to different categories. Consequently, SAM generates a multitude of masks as the output of this stage.

B. Multi-mask Filter

Parking-slots in real-world can be classified into two types: line-based and area-based, as illustrated in Fig. 2. The multiple masks output by SAM include several invalid masks. Consequently, we introduce a multi-mask filter to reduce this redundant information. Before referring to the filter's specific algorithm, it is essential to define the following terms:

- Masks: Masks output by SAM, with invalid masks.
- MinArea: The minimum area contains valid parking-slot, less than this value are *Noise*.
- MaxArea: The maximum area contains valid parking-slot, larger than the value are *Oversize*.
- ClassID(cID): Class information of each mask.
- FilteredMasks: Filtered masks with information.
- MidArea: A threshold for briefly distinguishing between line-mask and area-mask.

Algorithm 1 Multi-masks Filter

```
Input: Masks, MinArea, MaxArea
Output: FilteredMasks
 1: FilteredMasks = []
 2: for $i \leftarrow 1$ to $|Masks|$ do
 3:   if Masks[$i$].cID = Unclassified then
 4:     if Masks[$i$].Area $<$ MinArea then
 5:       Masks[$i$].cID $\leftarrow$ Noise
 6:     else if Masks[$i$].Area $>$ MaxArea then
 7:       Masks[$i$].cID $\leftarrow$ Oversize
 8:     else if EvaluateMasks(Masks[$i$]) = true then
 9:       FilteredMasks.append(Masks[$i$])
10:       Masks[$i$].cID $\leftarrow$ Valid
11:     else
12:       Masks[$i$].cID $\leftarrow$ Invalid
13:     end if
14:   end if
15: end for
16: return FilteredMasks
```

The main process of the multi-mask filter is shown in

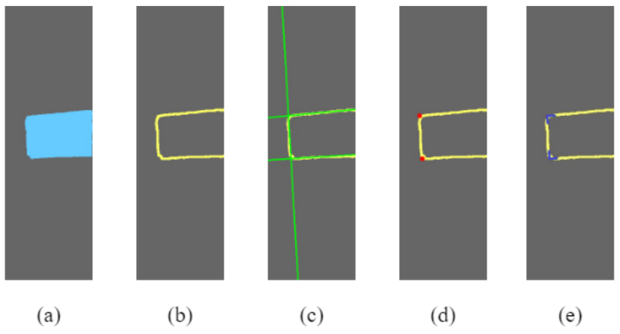
Fig. 3. Example about *FilteredMasks()* function. (a) Mask output by SAM. (b) Result after edge detection. (c) Lines detected from edge image after merging and extending. (d) Corners with parking-slot information.

Algorithm 1. As the shape and area of parking-slots must adhere to specific standards in real-world, the filter initially eliminates masks with areas that are too small or too large. Masks with overly small areas are typically noise, while those with excessively large areas often cover too many regions, lacking sufficient parking-slot information. Subsequently, the filter employs *FilteredMasks()* function to eliminate a substantial number of invalid masks(pseudo-code #10 in Algorithm 1). This function evaluates the validity of masks with appropriate areas. A mask is classified as valid if it has an appropriate area and contains parking-slot information; otherwise, it is deemed invalid.

The *FilteredMasks()* function constitutes the core component of the filter, with its pseudo-code outlined in Algorithm 2. The outputs of each step are illustrated in Fig. 3. For preliminary filtered masks in Algorithm 1, the visual closure operation is used to reduce the noise and fill the void space, followed by the application of a Sobel Filter to delineate the mask's edge. Subsequently, line detection identifies potential lines in the edge image. (pseudo-code #1-3 in Algorithm 2). After obtaining and merging lines, the intersection points of these lines are recorded as corner points. The *getCorner()* function outputs not only the degree of the corner but also its symmetry and orientation(pseudo-code #9 in Algorithm 2). Symmetry is defined as whether the corner is symmetrical relative to a line on which one of its sides lies. These corners are compared with those of standard templates for parking-slots in real-world scenarios(pseudo-code #11-20 in Algorithm 2), as illustrated in Fig. 4, the template can be rotated to any angle for matching. Within area masks, if a corner's angle near 90°, the function categorizes it as a L-shape. Subsequently, the orientation and type of each corner are determined. Masks whose corners meet these criteria are considered as valid candidate marking-points for parking-slot detection.

Finally, to assimilate all valid parking-slot information, the filter aggregates all corners, categorizing them as candidate marking-points characterized by position, type, and orientation. In addition, pairs of L-shaped corners of area masks in close proximity and with the same orientation are merged to form a T-shaped marking-point.

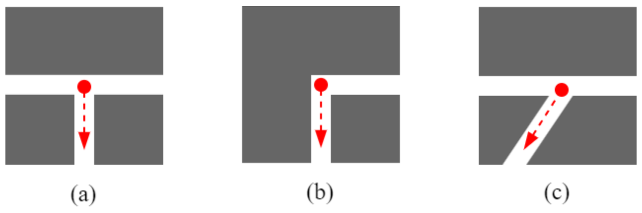
Fig. 4. Corner templates, with an arrow and its starting point indicate the orientation and position of each corner. (a) T-shape. (b) L-shape. (c) Y-shape.

Algorithm 2 EvaluateMasks Function

```
Input: Mask, Template, MidArea
Output: true or false
1:  if Mask.Area < MidArea then
2:      Mask.cID ← Line
3:  else
4:      Mask.cID ← Area
5:  end if
6:  Edges ← Sobel(Closure(Mask)
7:  lines ← LineDetection(Edges)
8:  corners ← getCorner(lines)
9:  for $i$ ← 1 to |corners| do
10:     $t$ ← corners[$i$].Degree
11:     if abs[$t-60$] < 5 or abs[$t-120$] < 5 then
12:         Mask.corners.append(corners[$i$])
13:         Mask.cornersID ← Y-shape
14:         return true
15:     else if abs[$t-90$] < 5 then
16:         Mask.corners.append(corners[$i$])
17:         return true
18:         if Mask.cID = Area then
19:             Mask.cornersID ← L-shape
20:         else if corners[$i$].Symmetry = true then
21:             Mask.cornersID ← T-shape
22:         else
23:             Mask.cornersID ← L-shape
24:         end if
25:     end if
26: end for
27: return false
```

C. Parking-slot Inference

After filtering out invalid corners and obtaining orientation and type of marking-points, parking-slots can be inferred from marking-points in the combination of remaining valid masks. Following the method proposed in DMPR-PS [15], to exclude invalid cases, two ranges of entrance-line distances are established as a priori knowledge. Subsequently, our approach employs the distance constraint to eliminate adjacent marking-points that do not meet the appropriate distance criteria.

Then valid pairs of marking-points are evaluated to determine whether they correspond to one of the predefined types of entrance-line, as illustrated in Fig. 5. In this figure,

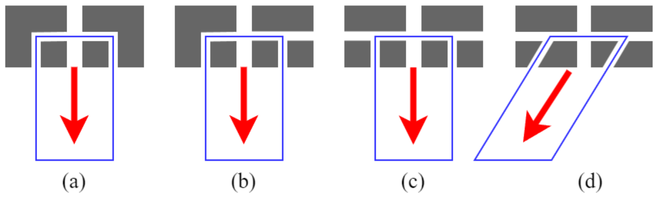

Fig. 5. Four types of slots categorized based on the properties of the marking-point pairs. (a) LL-slot, (b) LT-slot, (c) TT-slot, (d) YY-slot. A blue quadrangle indicate the parking-slot, and a red arrow indicate the orientation.

a blue quadrangle indicate the parking-slot, and a red arrow indicate the orientation. This figure present a brief example of parking-slot types. For example, an LL-slot consists of two adjacent marking-points of L-shape. For each pair of marking-points which determines an entrance-line, it is essential to categorize each parking-slot into one of the proposed four cases. This classification entails comparing the shape and orientation of each pair. When both marking-points in a pair conform to one of these four distinct cases, they are deemed to constitute a valid entrance-line and determine an ordered pair of marking-points. Consequently, the parking-slot corresponding to this entrance-line can be identified. Compared with method of DMPR-PS, our approach includes the addition of a YY-slot case to detect oblique parking-slots, as shown in (d) of Fig. 5.

IV. Experiment

A. Dataset

Our proposed parking-slot detection method is evaluated on public dataset PS2.0 [7] and PSDD [8]. PS2.0 consists of 9,827 images as training set and 2,338 images as test set. Following the setting of DMPR-PS [15], our test set consists of 2,290 images. Images within the PS2.0 are sourced from typical outdoor and indoor environments under diverse environmental conditions, and the resolution of each image is 600×600 pixels. The open-source PSDD consists of 17,696 calibrated BEV images, sourced from eight different scenes. To constitute the test set, a total of 1375 images were uniformly selected from five scenes—brick, grass, open, rectangle, and stereo. The brick, grass, and stereo types, absent in PS2.0, are categorized as novel classes, while open and rectangle types are designated as base classes. Due to images of the open-source PSDD are not complete around-view images. We resize their resolution to 600×600 pixels via padding when testing.

B. Experimental Setting

Due to the scarcity of open-source code, we choose DMPR-PS [15], VPS [23] and GCN [10] as baselines to evaluate the zero-shot transfer ability of our method. We divide experiments into two parts. The first part is conducted on base classes of our test set from PSDD, with parking-slot types including open and rectangle. The second part runs on novel classed, including brick, grass and stereo. To test the generalization performance of DMPR-PS, VPS and GCN, we make use of these models with pre-trained parameters on PS2.0.

Subsequently, the proposed method was evaluated against the following parking-slot detection methods on both PS2.0 [7] and PSDD [8]: two traditional vision-based methods :Wang et al.'s method [12], Hamda et al.'s method [4], and six deep learning-based methods: DeepPS [7], DMPR-PS [15], VPS [23], GCN [10] and Bui et al.'s method [9]. Our experiments were only compared using two traditional methods because the code and dataset from these works have not been open-source, making experimental comparison infeasible. This paper uses the precision-recall rate and $F1$ score as the evaluation metric.

$$precision = \frac{TP}{TP + FP} \tag{1}$$

$$recall = \frac{TP}{TP + FN} \tag{2}$$

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{3}$$

Ground-truth parking-slot is represented as $PS_{gt} = \{p_1^{gt}, p_2^{gt}, t^{gt}, \theta^{gt}\}$, where p_1^{gt}, p_2^{gt} are the corners of the entrance line, t^{gt} and θ^{gt} are the type and angle of parking-slot. Suppose $PS_d = \{p_1^d, p_2^d, t^d, \theta^d\}$ is a detected parking-slot. Following DeepPS [7], DMPR-PS [15] and GCN [10], we defined the conditions:

$$\|(p_1^{gt} - p_1^d, p_2^{gt} - p_2^d)\|_2 < 10 \tag{4}$$

$$t^{gt} = t^d \tag{5}$$

$$|\theta^{gt} - \theta^d| < 30^\circ \tag{6}$$

If above conditions are satisfied, we consider that PS_{gt} is correctly detected and PS_d is a *TP*. Otherwise, PS_d is a *FP*. If PS_{gt} does not match with any PS_d, PS_{gt} is a *FN*.

C. Implementation Details

This paper uses Pytorch for SAM, the average frame rate reaches 1.2 fps for 600*600 resolution images on RTX 3090. The parameters of SAM [11] are set to 16, 0.8, 0.6, 0.9, and 1.0 for points_per_side, pred_iou_thresh, stability_score_thresh, and stability_score_offset. The SAM model uses "segment everything" mode to automatically generate multiple masks without providing prompt. In addition, some parameters in multi-mask filter, such as MinArea, MaxArea and MidArea, are set to 50, 500 and 250.

D. Results and Discussions

a) Zero-shot Transfer: To evaluate the zero-shot generalization capabilities of various models, we designed two experimental sets on the PSDD dataset. The first set of experiments focuses on parking-slot types common to both PS2.0 and PSDD, while the second set targets parking-slots in scenes exclusive to PSDD, including brick, grass, and stereo. To test the generalization performance of DMPR-PS, VPS and GCN, these models were utilized with parameters pre-trained on PS2.0. TABLE I presents the detection results

TABLE I
ZERO-SHOT TRANSFER PERFORMANCE ON BASE CLASSES AND NOVEL CLASSES OF PSDD

Method	**No Training***	**Year**	**Base**†			**Novel**‡		
			Precision	**Recall**	**F_1 score**	**Precision**	**Recall**	**F_1 score**
DMPR-PS [15]	✓	2018	71.32%	31.14%	43.35%	67.80%	10.31%	17.90%
VPS [23]	✓	2020	76.64%	34.54%	47.61%	72.41%	7.25%	13.18%
GCN [10]	✓	2021	78.44%	48.60%	60.02%	64.17%	6.87%	12.41%
SAM-PS(Ours)	✓	2024	**70.81%**	**56.37%**	**62.77%**	**66.23%**	**40.27%**	**50.09%**

* Methods with weights of PS2.0 or without training.
† Common types of PSDD, including open and rectangle.
‡ Exclusive types of PSDD, including brick, grass and stereo.

TABLE II
PERFORMANCE ON PS2.0 AND PSDD

Method	**No Training***	**Year**	**PS2.0**			**PSDD**		
			Precision	**Recall**	**F_1 score**	**Precision**	**Recall**	**F_1 score**
SAM-PS (Ours)	✓	2024	**94.29%**	**81.35%**	**87.34%**	**68.17%**	**54.41%**	**60.38%**
Wang et al. [12]	✓	2014	98.29%	58.33%	73.21%	60.13%	10.24%	17.50%
Hamda et al. [4]	✓	2015	98.45%	61.37%	75.61%	40.96%	12.11%	18.69%
DeepPS [7]	×	2018	98.99%	99.13%	99.06%	80.23%	78.57%	79.39%
DMPR-PS [15]	×	2019	99.42%	99.37%	99.39%	88.45%	81.24%	84.69%
VPS [23]	×	2020	99.63%	99.10%	99.36%	-	-	-
GCN [10]	×	2021	99.56%	99.42%	99.49%	-	-	-
Bui et al. [9]	×	2023	99.63%	99.63%	99.63%	-	-	-

* Methods without training.

of VPS, DMPR-PS, GCN and SAM-PS(Ours). Primarily, the results indicate that our method outperforms these open-source deep learning methods in terms of F_1 score. This outcome suggests that in diverse scenes, deep learning-based methods often exhibit a notable decline in performance, as demonstrated in Fig. 6. The results for base classes indicate that deep learning-based methods retain a certain level of detection capability, attributed to the similarity of parking-slot between base classed and PS2.0. However, in novel classes, these deep learning-based methods demonstrate negligible detection capability. The results indicate that our proposed method(SAM-PS) attains state-of-the-art performance in zero-shot generalization ability, and can be used for zero-shot parking-slot detection. Consequently, it can be concluded that SAM-PS, as designed, effectively possesses zero-shot transfer capability for parking-slot detection, with the clarity of the around-view image being the primary performance determinant, rather than the type of parking-slot.

b) Parking-slot Detection: Our proposed method significantly outperforms two traditional vision-based parking-slot detection methods. Among methods without training on parking-slot dataset TABLE II demonstrates that our large visual model-based approach achieves state-of-the-art performance in F_1 score, when compared to traditional vision-based parking-slot detection methods. Our method can extract a broader range of discriminative features compared to traditional vision-based methods, offering enhanced robustness. DeepPS, DMPR-PS, VPS, GCN and Bui's method are all deep learning-based methods, with performance that is dependent on the parking-slot dataset. While deep learning methods can achieve nearly 100% F_1 score after being trained, our proposed method offers a distinct advantage: it does not necessitate the construction of a dataset, thus eliminating training costs and mitigating the risk of creating an unbalanced dataset.

V. CONCLUSION

We propose a method for parking-slot detection based on the large visual model SAM in this paper. Our method utilizes a single RGB image with around-view perspective as input, filters multi-masks, extracts the corner feature from each mask, and infers parking-slot through post-processing. A core contribution of our approach is the first time to use large visual model SAM for parking-slot detection. Leveraging SAM's zero-shot transfer capabilities, our method can detect parking slots without requiring additional training. Experimental results on PS2.0 and PSDD datasets demonstrate that our approach possesses zero-shot capabilities for parking-slot detection, comparable to advanced deep learning-based methods. In the future, we plan to optimize our model for enhanced real-time performance without compromising accuracy and expand our approach to more complex and various real-word scenarios, such as unstructured scene.

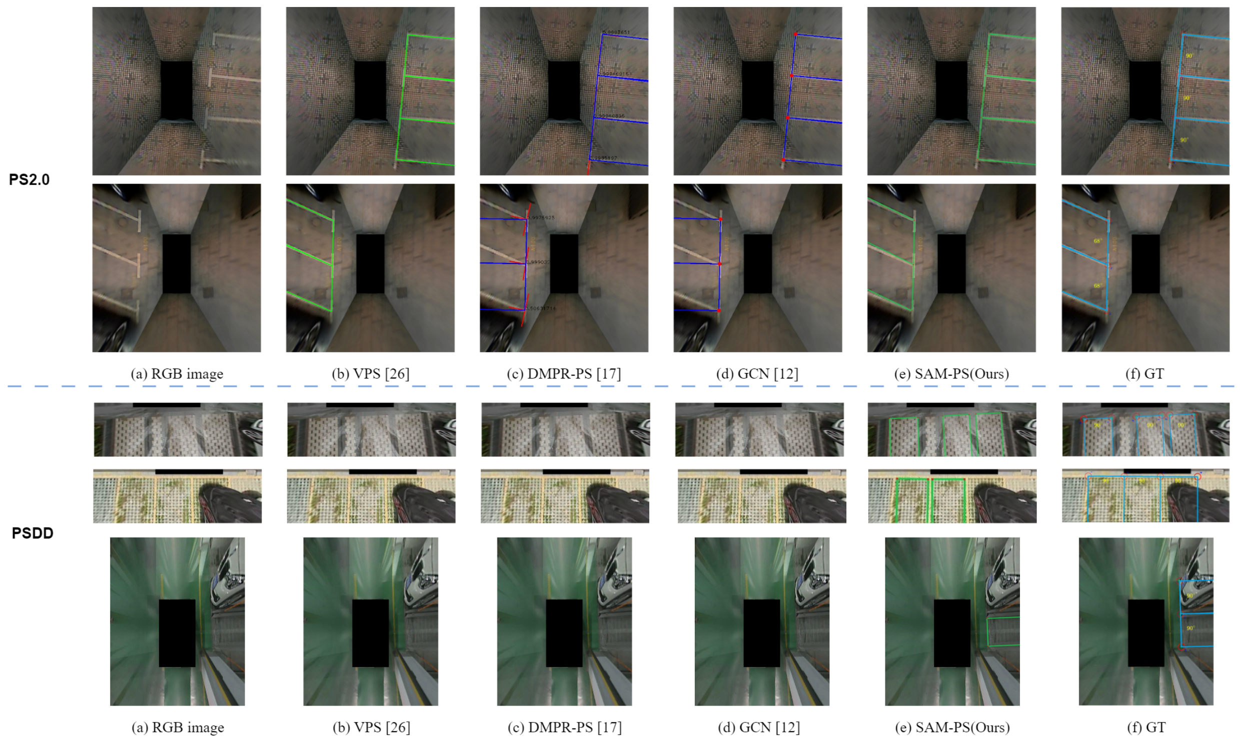

Fig. 6. Qualitative comparison for parking-slot detection of images on PS2.0 [7] and PSDD [8] dataset. The first two rows are qualitative comparisons of PS2.0 dataset. The remaining rows are PSDD dataset results, and the scene from top to bottom in order is brick, grass and stereo, according to different types of parking-slot in PSDD. (a) Input RGB image. (b)-(e) Results of VPS [23], DMPR-PS [15], GCN [10] and SAM-PS(Ours). (f) Ground Truth. The results shows that VPS [23], DMPR-PS [15] and GCN [10] are incapable of detecting in novel classes of PSDD.

REFERENCES

[1] M. Heimberger, J. Horgan, C. Hughes, J. B. McDonald, and S. K. Yogamani, "Computer vision in automated parking systems: Design, implementation and challenges," *Image Vis. Comput.*, vol. 68, pp. 88–101, 2017.

[2] K. Kumar, V. Singh, L. Raja, and S. N. Bhagirath, "A review of parking slot types and their detection techniques for smart cities," *Smart Cities*, 2023.

[3] G. S. Wong, M. G. K. Ong, C. Tee, and A. Q. M. Sabri, "Review of vision-based deep learning parking slot detection on surround view images," *Sensors (Basel, Switzerland)*, vol. 23, 2023.

[4] K. Hamada, Z. Hu, M. Fan, and H. Chen, "Surround view based parking lot detection and tracking," *2015 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1106–1111, 2015.

[5] J. K. Suhr and H. G. Jung, "Full-automatic recognition of various parking slot markings using a hierarchical tree structure," *Optical Engineering*, vol. 52, 2013.

[6] S. Lee, D. Hyeon, G. Park, I. joo Baek, S.-W. Kim, and S.-W. Seo, "Directional-dbscan: Parking-slot detection using a clustering method in around-view monitoring system," *2016 IEEE Intelligent Vehicles Symposium (IV)*, pp. 349–354, 2016.

[7] 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 Transactions on Image Processing*, vol. 27, pp. 5350–5364, 2018.

[8] Z. Wu, W. Sun, M. Wang, X. Wang, L. Ding, and F. Wang, "Psdet: Efficient and universal parking slot detection," *2020 IEEE Intelligent Vehicles Symposium (IV)*, pp. 290–297, 2020.

[9] Q. H. Bui and J. K. Suhr, "One-stage parking slot detection using component linkage and progressive assembly," *IEEE Intelligent Transportation Systems Magazine*, vol. 15, pp. 33–48, 2023.

[10] C. Min, J. Xu, L. Xiao, D. Zhao, Y. Nie, and B. Dai, "Attentional graph neural network for parking-slot detection," *IEEE Robotics and Automation Letters*, vol. 6, pp. 3445–3450, 2021.

[11] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. B. Girshick, "Segment anything," *ArXiv*, vol. abs/2304.02643, 2023.

[12] C. Wang, H. Zhang, M. Yang, X. Wang, L. Ye, and C. Guo, "Automatic parking based on a bird's eye view vision system," *Advances in Mechanical Engineering*, vol. 6, 2014.

[13] J. K. Suhr and H. G. Jung, "Sensor fusion-based vacant parking slot detection and tracking," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, pp. 21–36, 2014.

[14] H. Do and J. Y. Choi, "Context-based parking slot detection with a realistic dataset," *IEEE Access*, vol. 8, pp. 171 551–171 559, 2020.

[15] 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," *2019 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 212–217, 2019.

[16] J. K. Suhr and H. G. Jung, "End-to-end trainable one-stage parking slot detection integrating global and local information," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, pp. 4570–4582, 2020.

[17] Q. H. Bui and J. K. Suhr, "Transformer-based parking slot detection using fixed anchor points," *IEEE Access*, vol. 11, pp. 104 417–104 427, 2023.

[18] S. Zhou, D. Yin, and Y. Lu, "Passim: Parking slot recognition using attentional semantic segmentation and instance matching," *2022 IEEE 5th International Conference on Big Data and Artificial Intelligence (BDAI)*, pp. 169–175, 2022.

[19] X. Zhao, W.-Y. Ding, Y. An, Y. Du, T. Yu, M. Li, M. Tang, and J. Wang, "Fast segment anything," *ArXiv*, vol. abs/2306.12156, 2023.

[20] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Yolo by ultralytics. https://github.com/ultralytics/ultralytics, 2023.

[21] Y. Xiong, B. Varadarajan, L. Wu, X. Xiang, F. Xiao, C. Zhu, X. Dai, D. Wang, F. Sun, F. Iandola, R. Krishnamoorthi, and V. Chandra, "Efficientsam: Leveraged masked image pretraining for efficient segment anything," *ArXiv*, vol. abs/2312.00863, 2023.

[22] K. He, X. Chen, S. Xie, Y. Li, P. Doll'ar, and R. B. Girshick, "Masked autoencoders are scalable vision learners," *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15 979–15 988, 2021.

[23] 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, p. 2138, 2020.