Embedded Parking Slot Detection and Classification for Smart City Scenarios

Toma Lungoci
Department of Computer Science
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
lungocitoma@gmail.com

Mihai Negru
Department of Computer Science
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
Mihai.Negru@cs.utcluj.ro

***Abstract*—As cities are becoming more crowded due to the population moving to urban areas, smart parking and management are essential elements of the transportation system. There are efforts in developing vision-based automated parking systems by detecting the ground markings accurately in real-time from multiple surround-view cameras. We are not interested in automated parking, but in collecting relevant parking occupancy statistics from the city streets. In this paper, we propose a novel method based on a single front-view camera and a convolutional neural network (CNN) for detecting the parking slots and classifying them based on occupancy. This information collected by the camera is sent to a central processing cloud and can serve as a basis for many Smart City parking management applications, by providing useful dynamic occupancy statistics.**

I. Introduction

Smart cities promise to provide innovative and efficient solutions for optimizing the infrastructure, leveraging the power of advanced technology. The aim is to make the most use of the linked data available, maximizing the use of limited resources. Artificial Intelligence and the Internet of Things can change the way we live and interact in urban surroundings, improving the quality of life. The focus of our project is to use modern computer vision algorithms for smart city applications. A study by INRIX [1] shows that UK drivers spend an average of 44 hours a year searching for parking, and many of them want real-time parking availability statistics. We aim to solve this problem by developing a system which monitors traffic patterns to help with traffic congestion, find open parking slots and direct drivers to them, dynamically manage the pricing to distribute the traffic in the city.

The main component of our system is a neural network that detects the parking slots and classifies them based on occupancy. The model must not only be accurate, but also efficient such that is can be deployed on an embedded target and achieve real-time performance. The camera mounted on the windshield of the vehicle sends the detection results to the data cloud which does the processing and displays the analytics.

The parking slot detection models present in the literature are intended for automated or assisted parking. We not only have a different purpose, that is data collection, but also a cheaper and more scalable single front-view camera setup. This means that we had to construct our own labeled dataset with images collected from our cameras. The neural network was trained on our dataset and deployed on our test cameras, ensuring that the system works end-to-end.

II. Related Works and our contributions

The vision-based parking slot detection task is a specific application of object detection. Consequently, state-of-the-art methods for this task closely follow the advancements in object detection models. This section will present the most relevant methods that influenced our work.

A. Parking Slot Detection

Previous work for vision-based parking slot deteciton was based on classical image processing and pattern recognition techniques, more specifically line-based methods. Jung et al. [2] considered that the ground marking should have constant width and applied peak-pair detection and clustering in the Hough space. Based on a similar approach, Wang et al. [3] proposed parking line-segments detection in the Radon space, due to the fact that the Radon transform is more tolerant to noise. However, this low level visual feature detection proves not to be robust to environmental changes and does not generalize.

Machine learning methods using convolutional neural networks (CNN) are the standard for parking slot detection. Zhang et al. [4] was the first to propose a CNN-based detection method, with a YOLO v2 [5] network for detecting the delimiting corners of the parking slots. A local image patch containing valid ground markings is sent

Acknowledgement: This work was funded under the IPCEI Microelectronics and Communication Technologies (IPCEI ME/CT) as part of the European microelectronics value chain for highly automated driving and green mobility (EuroDrives) project.

979-8-3315-3997-9/24/$31.00 ©2024 IEEE

to a subsequent classfier network for determining the type of the parking slot. The authors were also responsible for establishing a large-scale labeled dataset publicly available (https://cslinzhang.github.io/deepps/), called ps2.0, to facilitate further research efforts. Their dataset is used now as a baseline for novel parking slot detection systems.

Improving on the YOLO-based architecture, Yu et al. [6] designed a more efficient network for the purpose of running on lower-powered devices in real-time. They introduced the hourglass module [7] for accurate keypoint-prediction as a backbone for their fully convolutional network. In addition, the authors designed an efficient convolutional module that prunes redundant channels for better performance. We extend on their work, since our goal is to achieve high processing speed on an embedded target, while maintaining high accuracy. However, our system will not be based on top-view images constructed from multiple surround-view cameras, but only one single front-view camera mounted on the vehicle's dashboard.

B. Keypoint-based Object Detectors

Even though transformer-based architectures achieve the best detection performance, they are still not suitable for deployment on lower-powered devices for real-time performance. Architectures like CenterNet [8] and CornerNet [9] model the objects as points and use a robust keypoint prediction network as a backbone component. This presents a lot of advantages, especially considering our purpose of embedded detection: simple in terms of architecture and operations, faster processing speed, end-to-end differentiable and most importantly reduced processing time outside the neural network inference. The postprocessing algorithm required for keypoint-based methods is negligible in comparison with the reduction of redundant bounding boxes in regressor detectors (non-maximum suppression).

C. Our Motivations and Contributions

Our network is inspired by the fully convolutional architecture proposed by Yu et al. [6]. We are using a stacked hourglass backbone essential for precise keypoint estimation, while a sine and cosine value necessary for constructing the individual parking slots are regressed from image features. In addition, a specialized head for the classficiation task is introduced. We treat the occupancy classification problem as a binary segmentation and train the network jointly. The classification head provides us a binary mask used for determining whether the parking slot is occupied. Moreover, our method is unique and more efficient, since we are using a single front-view camera setup. We require a inverse perspective mapping (IPM) for transforming the perspective image into a top-view image which is fed into the neural network.

III. Methodology

We use a reduced resolution, grayscale top-view image as input for our network. The inverse perspective mapping algorithm used for obtaining it is key for our approach. The output heatmaps provided by the CNN are used for the final step, which is the slot extraction. The result will be a list of all the bounding parallelograms and a binary value for the classification. Figure 1 shows the three separate modules of our system.

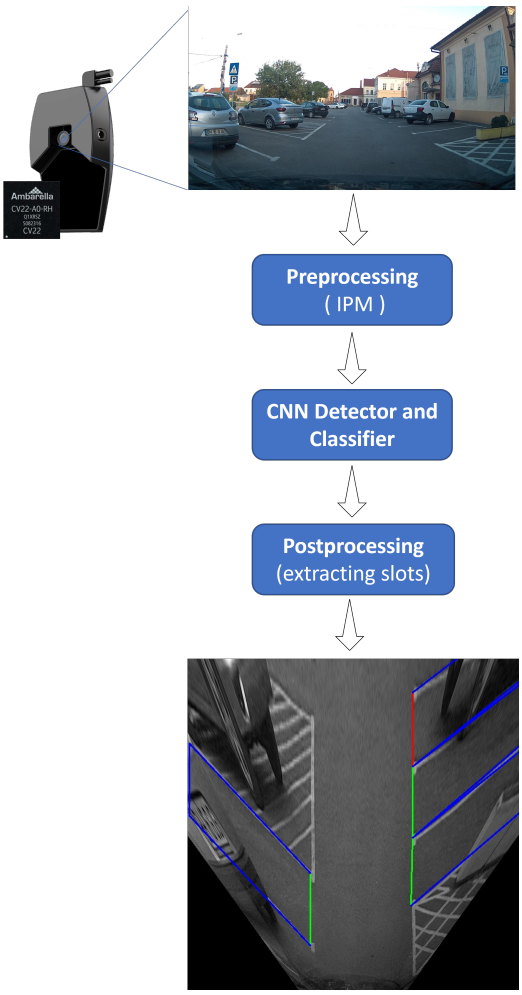

Fig. 1. Processing pipeline of the system.

A. Inverse Perspective Mapping

The inverse perspective mapping is a geometrical transform that removes the perspective effect from the acquired image, remapping it into a 2D space in which the information content is homogeneously distributed among all pixels [10]. The ground markings on the road are more visible in the bird's-eye view image produced by the IPM, making it suitable for tasks like parking slot detection and lane detection among others. We need to find the corresponding mapping between the points in the top-view image and the ones in the perspective image. The perspective transformation is illustrated in equation (1), with the following parameters:

- nu and nv: The new coordinates in the perspective image.
- f_{nu}: Focal length in the u direction.
- f_{nv}: Focal length in the v direction.
- $ppnu$: Principal point offset in the u direction.

$$\begin{bmatrix} nu \\ nv \\ 1 \end{bmatrix} = \begin{bmatrix} f_{\text{nu}} & 0 & ppnu \\ 0 & f_{\text{nv}} & ppnv \\ 0 & 0 & 1 \end{bmatrix} R_{\text{cal}} \begin{bmatrix} \frac{1}{f_{\text{top-view}}} & 0 & -\frac{ppu_{\text{top-view}}}{f_{\text{top-view}}} \\ 0 & \frac{1}{f_{\text{top-view}}} & -\frac{ppv_{\text{top-view}}}{f_{\text{top-view}}} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{1}$$

- $ppnv$: Principal point offset in the v direction.
- R_{cal}: Rotation matrix from the calibration component.
- $f_{\text{top-view}}$: Focal length in the top-view (will be equal with the camera height).
- $ppu_{\text{top-view}}$: Principal point offset in the top-view, u direction.
- $ppv_{\text{top-view}}$: Principal point offset in the top-view, v direction.
- u: u coordinate in top-view.
- v: v coordinate in top-view.

During the transformation, we reduce the resolution of the image from 1920x848 to 224x224 pixels, thus making the operation fast and allow for an efficient neural network, with fewer parameters. The complete preprocessing step runs in 8 ms on the CPU present on the SoC. Figure 2 shows our images compared to the dataset used in the literature. It can be seen that our data presents scenarios of higher degree of difficulty.

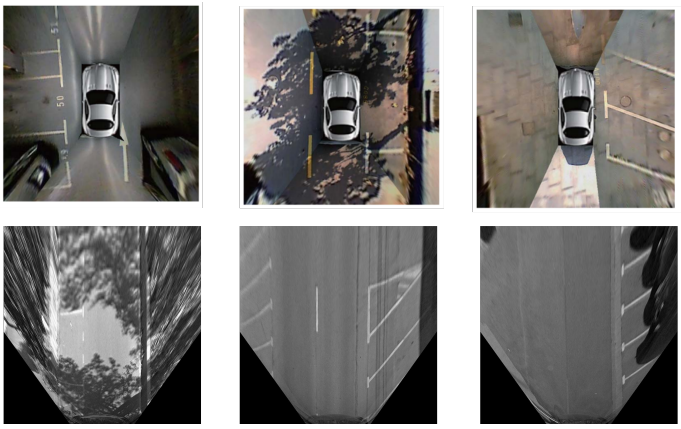

Fig. 2. Images from our dataset (bottom row) compared to the ones used in literature (top row) [4].

B. Network Structure

Inspired by the architecture proposed by Yu [6], we are using a stacked hourglass backbone for precise keypoint prediction. Trying to achieve the best accuracy while maintaining reduced memory usage and inference time, our model consists of two hourglasses. As presented in the original paper [7], intermediate supervision enables significant accuracy improvements. We followed the strategy proposed in CornerNet [9], where the input and the output of the first hourglass are merged together using element-wise addition and provided as input for the second hourglass module.

We experimented with different convolutional modules: residual [11], inception [12], select and prune [6], selective kernel [13]. The selective kernel (SKModule), see figure 3, proved to achieve the best performance for our specific task. We attribute it to the capability of adjusting the receptive field sizes based on the input, since similar ground markings can appear at different scales. Even though the select and prune module is similar, the pruning strategy hurts performance on our specific dataset.

TABLE I
PARKING SLOT CNN CHARACTERISTICS

Parking Slot CNN	
Layers for one hourglass	9
Channel dimensions	[32, 44, 64, 92, 128]
Hourglass modules	2
Trainable parameters	7.4 million
Module	SKModule

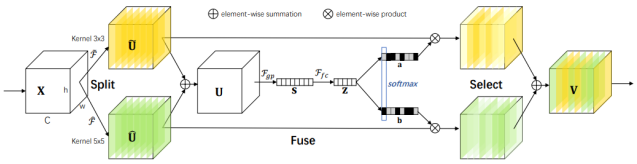

Fig. 3. Selective Kernel Convolution [13].

C. Slots Inference

Our network has two separate heads: one for detection and one for occupancy classification. Firstly, the detection head provides the keypoint heatmap. The marking points, that is the T-shapes delimiting the corners of a parking slot, are modeled as 7x7 2D gaussian kernels. Peaks in the keypoint heatmap represent the ground markings we are trying to detect. For each marking point position, we are regressing a sine and cosine value that construct the complete parking slots. We use the coordinates of a detected *P* point and the cosine and sine values to calculate its backside correspondent *P'*, using the equations (2), where *SlotLength* is a predefined value. In figure 4, the network output probabilities are plotted on top of the input image.

$$\begin{aligned} X_{P'} &= X_P + \textit{SlotLength} \cdot \cos\alpha \\ Y_{P'} &= Y_P + \textit{SlotLength} \cdot \sin\alpha \end{aligned} \tag{2}$$

Secondly, the classification head's purpose is to segment the road area based on occupancy. After detecting the individual slots, the binary mask is used for inferring the occupancy. In the area bounded by each parking slot, computed in the detection phase, a mean pixel value from the classification heatmap is calculated and a logistic regression decision is

$$L_k = -\frac{1}{N}\sum_{xy}\begin{cases}(1-\hat{Y}_{xy})^{\alpha}\log(\hat{Y}_{xy}) & \text{if } Y_{xy}=1\\ (1-Y_{xy})^{\beta}(\hat{Y}_{xy})^{\alpha}\log(1-\hat{Y}_{xy}) & \text{otherwise}\end{cases} \tag{3}$$

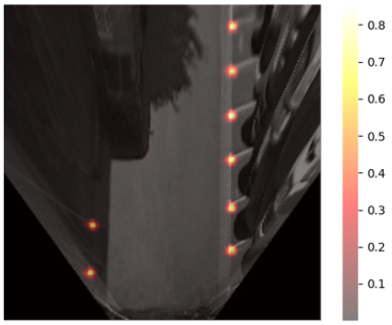

Fig. 4. Keypoint heatmap prediction.

made for the binary occupancy output. Figure 5 presents the results of parking spot detection and classification heatmap. In addition to the parking marking points and lines, a binary label mask was constructed, where the occupied parking slots have been marked with ones and the rest with zeros.

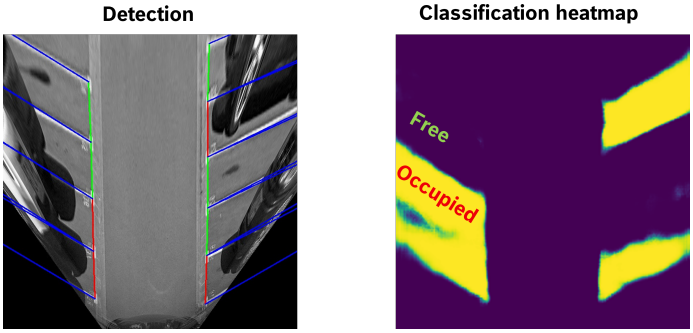

Fig. 5. Parking slot detection and classification based on the output occupancy heatmap.

IV. Experimental Results

A. Dataset

We constructed our own dataset with images collected by our test camera. The current version of our dataset contains 2873 labeled images. A straightforward augmentation is the horizontal flip, resulting in a total of: 4350 training images, 577 test images and 819 validation images. We also consider extending the dataset by introducing slight errors in the IPM preprocessing, to make our system robust to perturbations in the camera calibration process. We are still in the process of collecting and labeling more images.

B. Network Training

The final loss function is a weighted sum of the losses for the individual components. First, for the keypoint prediction, we are using a Focal Loss:

Where $Y \in [0,1]^{224\times224}$ is the heatmap with the ground truth keypoints and $\alpha = 2$ and $\beta = 4$, following Law [9]. For the regressed sine and cosine, we use a Mean Absolute Error between the actual and predicted values:

$$L_{\text{sine}} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{4}$$

For the classification heatmap, we are applying a per-pixel Binary Cross Entropy (BCE) loss. The final loss is a weighted combination of the separate loss components:

$$L_{\text{total}} = \lambda_k L_{\text{keypoints}} + \lambda_s L_{\text{sine}} + \lambda_c L_{\text{cosine}} + \lambda_{cls} L_{\text{classification}} \tag{5}$$

We used the Adam optimization algorithm [14], and manually adjust the learning rate and the loss weighing parameters during the training process, to ensure that all features are learned properly. The most important component is definitely the keypoint prediction, so it was the main focus at the beginning of the optimization process. We did not gain performance by pretraining the network on the public ps2.0 dataset and doing transfer learning on ours.

C. Parking Slot Detection and Occupancy Classification

We conducted experiments on our test set, using the precision, recall and accuracy as performance metrics. Training on the public ps2.0 dataset achieves state-of-the-art performance, but there is a drop when using our more difficult dataset, containing many images with degraded and irregular ground markings, plus a larger variety of slanted slots. The results are summarized in table IV. The results exceeded our initial expectations and we consider that they are good enough for our purpose of collecting parking availability information, especially since information can be aggregated from multiple vehicles and multiple rides.

TABLE II
Confusion Matrix Parking Slot Detection

		Predicted	
		Valid Slot	No Slot
Actual	Valid Slot	2454 (True Positive)	434 (False Negative)
	No Slot	184 (False Positive)	- (True Negative)

D. Failure Cases and Generalization Capability

The system must be able to adapt to scenarios with degraded and irregular parking markings, in various environmental conditions, as this may be the case in most countries. Again, the most important feature of the system, directly indicative

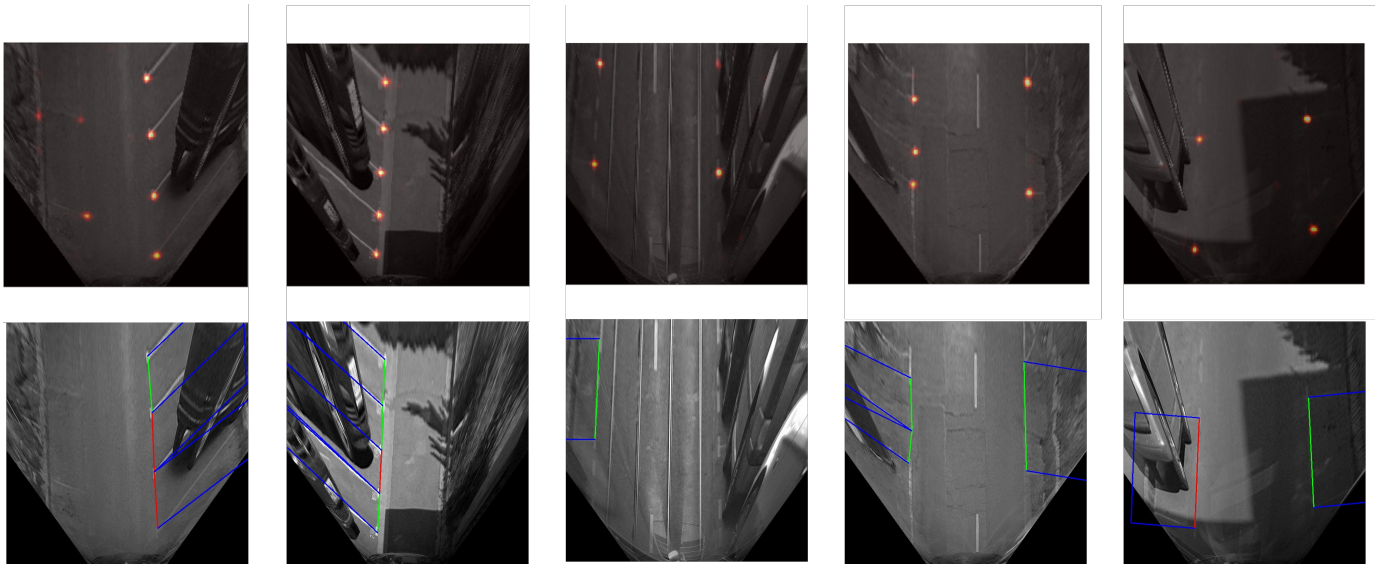

Fig. 6. Network predictions on difficult cases: keypoint heatmap (top row), parking slot detection and classification (bottom row).

TABLE III
CONFUSION MATRIX PARKING SLOT OCCUPANCY

		Predicted	
		Free Slot	Occupied Slot
Actual	Free Slot	863 (True Positive)	10 (False Negative)
	Occupied Slot	200 (False Positive)	385 (True Negative)

TABLE IV
PERFORMANCE METRICS FOR PARKING SLOT DETECTION AND OCCUPANCY CLASSIFICATION

	Parking slot detection	**Occupancy classification**
Precision	0.93	0.81
Recall	0.85	0.98
Accuracy	-	0.85

of the overall performance is the keypoint prediction. Errors in the keypoint predictor result in erroneous parking slots. Fortunately, the keypoint-based detection provides heatmaps interpretable to humans and can be used as means for explainability. The predicted heatmaps serve as visual clues for figuring out what influenced the unwanted behaviour. Figure 6 illustrates some of the difficult cases encountered: vehicles parked on two slots, occluded parking slots, irregular and degraded markings.

Also, for our solution to be scalable, it must adapt to different calibration parameters with slight errors. We are trying to mitigate this problem by introducing more data augmentation, running the IPM several times on the input image, with slight variations of the pitch angle and the camera height. Considering the small dataset the network was trained on, the system has promising generalization power.

E. Runtime

We used the Momenta AutoRing A4 Retrofit video camera for our project. The main reason for choosing this camera is the Ambarella CV22 Soc. It has a CVFlow computer vision processing unit and a 64-bit ARM Quad Core Cortex-A53 CPU. The IPM runs in 8ms on the CPU, while the CNN inference takes 60ms on the AI accelerator. The total runtime is within bounds, since in our current implementation the video camera sends data to the processing cloud once every second. In addition, pruning techniques like the one proposed by Yu [6] can improve the runtime if needed.

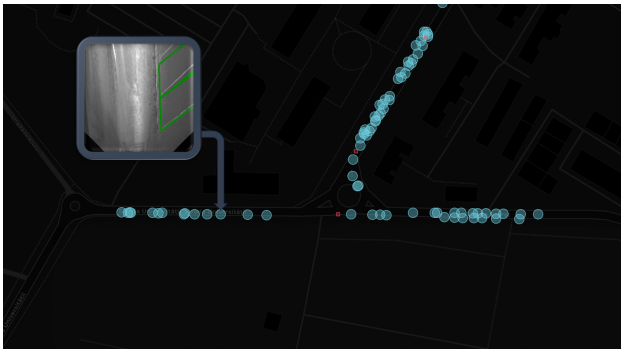

Fig. 7. Camera detections visualized in the browser dashboard.

V. CONCLUSION AND FURTHER WORK

We developed a novel method of parking slot detection using only a single front-view camera. On top of this, we added a occupancy classification module, enabling real-time parking occupancy applications. The neural network was deployed on our target hardware. The detection results are correctly sent to the data processing cloud, such that our system works end-to-end without user intervention. We tested the system on one of our test cameras, visualizing the results on a dashboard in the browser 7. There are various means to further improve the performance of our network in the near future, like for example extending the currently very small dataset.

In order to achieve accurate dynamic occupancy statistics, mapping and tracking techniques must be employed on top of our detection system. A complete parking management application should provide occupancy rates for the streets in the city. Such information can be used to distribute drivers and reduce traffic, optimize the parking areas, dynamically adjust the parking pricing among many others. Nevertheless, good parking management is key for a Smart City. Our goal is to extend the system with these features, resulting in an dashboard application containing live parking occupancy statistics, similar with figure 8, useful for both drivers and governments.

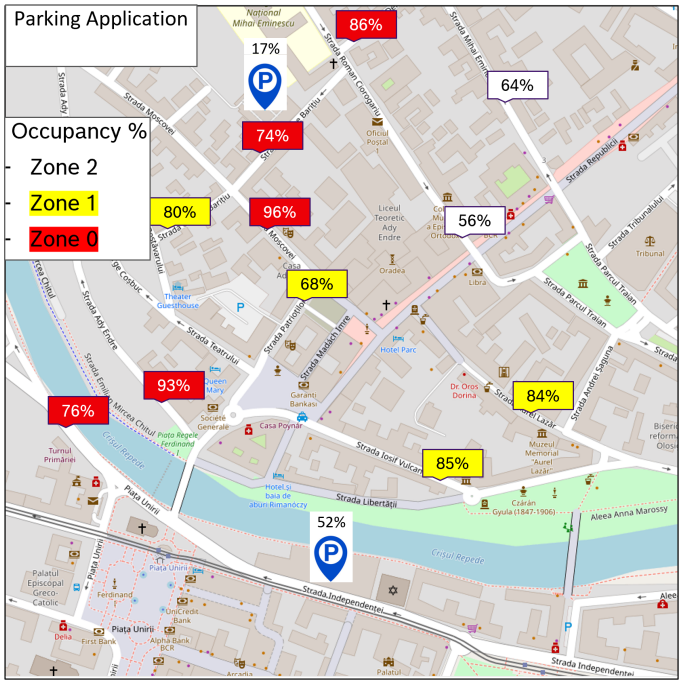

Fig. 8. Example of dashboard application with occupancy statistics.

REFERENCES

[1] G. Cookson, "Smart Parking – A Silver Bullet for Parking Pain," https://inrix.com/blog/parkingsurvey/. [Online]. Available: https://inrix.com/blog/parkingsurvey/

[2] H. G. Jung, D. Kim, P. Yoon, and J. Kim, "Parking slot markings recognition for automatic parking assist system," 01 2006, pp. 106 – 113.

[3] 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, pp. 847 406–847 406, 02 2015.

[4] 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, no. 11, pp. 5350–5364, 2018.

[5] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," 2016. [Online]. Available: https://arxiv.org/abs/1612.08242

[6] Z. Yu, Z. Gao, H. Chen, and Y. Huang, "Spfcn: Select and prune the fully convolutional networks for real-time parking slot detection," 2020.

[7] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," 2016.

[8] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," 2019.

[9] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," 2019.

[10] M. Bertozz, A. Broggi, and A. Fascioli, "Stereo inverse perspective mapping: theory and applications," *Image and Vision Computing*, vol. 16, no. 8, pp. 585–590, 1998. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0262885697000930

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2014.

[13] X. Li, W. Wang, X. Hu, and J. Yang, "Selective kernel networks," 2019.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017. [Online]. Available: https://arxiv.org/abs/1412.6980