PARKING MANAGEMENT SYSTEM BASED ON KEY POINTS DETECTION

Kristián MIČKO, Peter PAPCUN
Department of Cybernetics and Artificial Intelligence, Faculty of Electrical Engineering and Informatics,
Technical University of Košice, Slovak Republic, Tel.: +421 55 602 5115, +421 55 602 2508
E-mails: kristian.micko@tuke.sk, peter.papcun@tuke.sk

ABSTRACT

In urban areas, efficient parking management is crucial for reducing traffic congestion and environmental impact. This research introduces a new view for making the parking management system that leverages the capabilities of the NVidia Jetson Nano Single Board Computer (SBC) and OpenCV for real-time detection and classification of parking slot occupancy. Unlike traditional systems that rely on intrusive sensors, our proposed solution employs non-intrusive Oriented Fast and Rotated Brief (ORB) key point detection techniques using video feeds. The system architecture integrates video stream processing, ORB via OpenCV, cloud-based data storage, and a Flask server for user notifications. The methodology prioritizes traditional computer vision methods optimized for the Jetson Nano's CUDA cores, offering a computationally efficient alternative to deep learning approaches. Python's versatility and MongoDB's document-based storage are employed for backend development. Our system's performance, evaluated using open datasets, demonstrates high accuracy, precision, recall, and F1 scores, underlining its effectiveness in real-world urban parking scenarios. This study not only presents a robust solution for parking management but also opens avenues for similar applications in traffic measurement and urban planning.

Keywords: *Internet of Things, Computer Vision, Key Points Detection, Edge Computing, Web Technologies*

1. INTRODUCTION

With the surge in urbanization and the growth of the automotive industry, parking management has become a focal point in urban planning and traffic flow optimization. Efficient parking systems not only lead to better utilization of space but also reduce traffic congestion and environmental pollution caused by vehicles searching for parking slots [1]. Efficient parking systems could be implemented by two approaches such as intrusive or non-intrusive sensors [2, 3]. The sensors collect data necessary to monitor the parking slots in real-time conditions.

The advantage of intrusive sensors is the cheap production cost. On the other hand, they have to be installed under every parking slot if it needs to be verified vehicle motion or presence. The increased computational capacities of single-board computers (SBC) enable us to open new potentials in sensing systems. Surveillance camera systems connected to as well as SBC can use computer vision (CV) methods to automatically extract useful information about events such as the intrusion of the area or traffic sign recognition [4, 5]. CV-based methods leverage sophisticated algorithms that can process real-time video feeds to accurately detect and map free parking slots.

The integration of CV in parking management systems can significantly improve the user experience by providing real-time parking slot status. The parking management systems CV-based methods deployed on the SBC would have better conditions to maintain the hardware equipment. CV-based methods require only the camera system installation that is a non-intrusive sensor. The non-intrusive sensors do not need to use road excavation works.

This paper is structured as follows. Section 2 analyses "State-of-the-Art" technologies, emphasizing solutions that employ intrusive sensors in parking management systems as well as camera-based systems operating on the edge-fog-cloud computing architectures. Section 3 delves into the design of our proposed solution. Section 4 describes the classification methods to determining the free parking slots. The following Section 5 describes the technical side of our solution in this research paper. Section 5 delineate the hardware and software technologies. Section 6 evaluates the success of our solution in the sample open datasets. Section 7 concludes the proposed solution and its experimental results on the sample open datasets.

2. RELATED WORK

Study [6] proposed using the system of piezoelectric sensors installed above the pavement surface. The surface area is the parking slot. The sensors are proposed to transform the surface strain into electricity with increased power. This power is efficient for sending signals as well as supplying the control unit and parking slot lights. Furthermore, the vehicle's visiting frequency impacts to degradation of piezoelectric crystals in the sensor. The degradation affects the reliability of detecting vehicles by this sensor.

Another approach for the management of parking spaces is the usage of a distributed wireless camera system, enabling the capture of images from the parking area and analyzing them to determine the occupancy status. This study [7] proposed a design system based on Raspberry Pi Zero camera modules with optimized methods and algorithms for the detection occupancy status. This system also provides information about the vehicle's orientation to improve accuracy. The implementation was tested in various conditions and achieved more than 90% accuracy. The advantage of this method is making the potential cost-balanced and effective solution.

The study [8] describes the solution based on the Raspberry Pi SBC-based parking management system for smart campus conditions. It could help users find available parking spots with real-time navigation through GPS coordinates shown in the smartphone app. The hardware prototype in this study consists of the Raspberry Pi 4 B+, Pi cam-

era, GPS sensor, and ultrasonic sensors. Ultrasonic sensors collect and send data to the Raspberry Pi 4 B+. The Raspberry Pi 4 B+ has a program that processes data to the cloud layer. In the cloud layer, it is implemented the Blynk IoT server via Wi-Fi. The system prototype is tested in practice and can effectively monitor the occupancy status of outdoor parking spaces in real-time conditions.

3. ARCHITECTURE DESIGN

Our parking management solution consists of a connection to the camera, SBC NVidia Jetson Nano, with Ubuntu OS. SBC NVidia Jetson Nano will use the OpenCV library and Python implementation of the ORB key point descriptor [9]. We use the sample open data from a time-lapse video sequence to test our solution. The proposed architecture is based on new research results, and a new approach was composed to the architecture published in the paper [10].

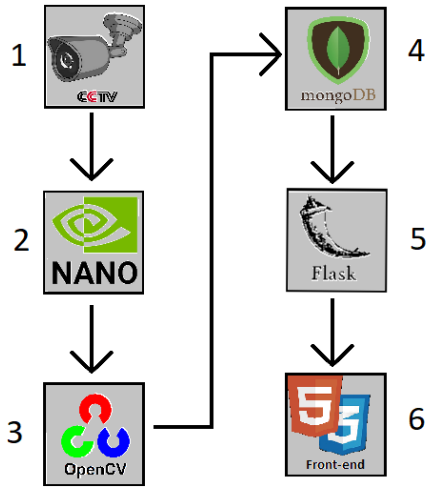

Fig. 1 Schema of proposed workflow architecture

Classification of free parking slots is realized by NVidia Jetson Nano. The classified situation data could be sent to the cloud. Fig. 1 describes the workflow of collecting data and next processing step by step:

1. Camera records and sends the video stream to the NVidia Jetson Nano.
2. Nvidia Jetson Nano converts camera records to the numpy matrix suitable for OpenCV library.
3. NVidia Jetson Nano uses OpenCV library to detect key points and classify the parking place occupancy.
4. The Classified situation is sent to cloud No-SQL database MongoDB.
5. Flask server is configured as a change database listener. Flask server sends notifications to client applications.
6. Client applications show acquired information from parking slots.

The illustrated workflow allows changing the camera and SBC CV-based software solution into the independent, smart sensor [4]. This smart sensor can be used not only for car lot occupancy but it can be used as a navigation or localization system for vehicles to find the way to park cars in free slots [11].

The described architecture in Fig. 1 could be implemented with many technologies that are mentioned in Section 5.

4. METHODOLOGY

There were mentioned CV methods for making the parking management system in the previous sections. These methods are deployed on the edge computing layer and must be optimized to run in SBCs. There was considered use of the traditional CV methods instead of the DL approach. Traditional CV methods use the basic matrix operations, which are perfectly optimized for running CUDA cores in the NVidia Jetson Nano. The steps are described below in Fig. 2 and Fig. 3.

In the first step, manually or automatically, somebody prepares the binary mask where the ones (for example, with white color) represent the parking lot and zeros represent the background. It is also necessary to include the white lanes as the background because the next algorithm outputs could be biased.

In the following step, we use the Canny's edge detector on the input frame from the camera. The binary image with the detected edges is compared to the parking lot binary mask with the bit operator 'and', which calculates whether the white pixel will be kept in the next step.

After the bit operation, the output image will use the orb key point detector to identify key points. The key points will be drawn as circles with a specified diameter on another empty image (black image) with kept sizes (width and height) as the camera's resolution.

The drawn key points as circles create a new binary mask that will be analyzed with the following step. The binary mask of the parking lot places makes borders of each rectangle. The rectangle's borders will be used to calculate the pixel's mean values on the generated binary mask with key points.

Each rectangle has its own pixel's mean value. The drawn circles use the value of 255 for pixels that render the circle. Other pixels have the value of 0. So, if we find zero circles inside the rectangle, the pixel's mean will equal 0. On the other hand, if we find at least one circle inside the rectangle, the pixel's mean will equal more than 0. The increased circles inside the rectangle increase the pixel's value.

We only have to calibrate the threshold value to correct classification, identifying enough key points inside the rectangle by the pixel's mean value to classify the parking slot status.

The methodology's success is described in Section 4.

5. IMPLEMENTATION

This section describes the technical approaches for implementing the parking management system. As was men-

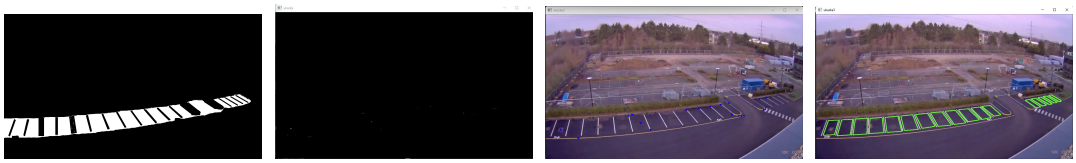

Fig. 2 The first image on the left is the parking lot's binary mask. The second image on the left is the output of Canny edge detector. The third image on the right represents the drawn keypoints from orb detector which was computed from Canny edge detector shown in the previous image. The last image on the right shows the parking lot area shown in green rectangle. Green rectangle represents the free parking slot

tioned above, the solution is made by the combination of CV methods and edge computing. CV methods require matrix operations [12]. If we want to convert the CV methods as smart sensor firmware, we have to use programming languages and libraries optimized for using matrix operations under milliseconds per frame.

It is necessary to select suitable embedded hardware that is compact and has efficient computational power on low energy consumption. For this purpose, microcontrollers (MCU) are too weak, and not all SBCs are suitable, such as Raspberry Pi of the first and second generation. To simplify the solution mentioned in the research [10, 13], we decided to use a more powerful SBC NVidia Jetson Nano instead of using a laptop, desktop, or server computer as a fog computing layer and Raspberry Pi SBC as an edge layer.

In this study, we decided to use the Python programming language, which is easy to deploy across many operating systems (OS) like Windows, Linux, Android, or iOS distributions, and hardware architectures such as ARM, Intel, or AMD. Python programming language enables us to use optimized matrix operations because the libraries are developed in C/C++ languages, and Python's syntax is simple with a declarative programming approach. Python also could use other programming paradigms and approaches, such as object-oriented programming (OOP) or functional programming principles [14].

Many CV and deep learning (DL) applications require the use of the computational power of not only central processing units (CPUs) but also graphical processing units (GPUs). GPUs produced by the company NVidia developed special instructions for GPUs in CUDA cores. These CUDA cores support basic matrix operations, and they are embedded in SBCs such as the NVidia Jetson series. It is necessary to use special software development kits (SDKs) for implementing programs with CUDA cores powered computation. The libraries of the python support CUDA SDKs that optimize the performance of the CV and DL applications [15].

The data scientists prefer using Python for DL, CV, and other machine learning (ML) projects. The reason was mentioned in this paragraph 5. Another advantage of Python is its suitability for making web applications from the back-end side using popular frameworks such as Django or Flask. We chose the Flask framework for its simplicity and possibility with a few lines of code to build REST API services that are parsable for many front-end web or mobile applications [16].

The proposed system was implemented and tested for a car parking management system use case scenario. However, users could apply the same CV techniques not only for parking management systems but also for automatic monitoring of traffic systems based on mast-mounted highway closed-circuit television (CCTV) camera systems. The back-end applications require a storage system to keep data generated through the app's live cycle. The storage system for many applications is based on a database management system.

The database management systems are divided into table-based approaches using a structured query language (SQL) for create-update-read-delete (CRUD) actions, document-based approaches using, for example, key-value structures for querying CRUD actions, and graph-based databases use the graph theory algorithm for querying stored data on strong relationships between records and complex queries. The graph-based databases for defining edges and nodes require some rules based on the domain's ontology mapping or other complex structures. Social net-

Fig. 3 The first image on the left shows the output of Canny edge detector when the parking lots contains the vehicles. The image in the middle shows the vehicle´s keiponts made from orb detector. The last image on the right shows the classified rectangles. The green one shows free parking slot, and the red one shows the places with vehicles

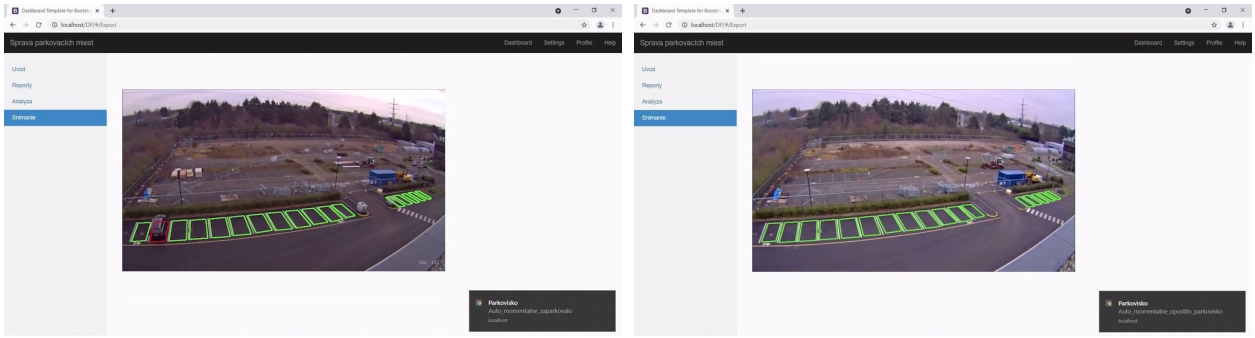

Fig. 4 The example of implemented front-end web application based on the Bootstrap framework with notifications on right bottom corner. The notifications show the changes in parking lots

works use graph-based databases for mapping relationships between each user [17].

The table-based databases use relational schemas to structure the relations between each table. If we want to use CRUD operations, all data attributes and relations must be defined before using the database instance. Another disadvantage is scaling the database. The table-based databases could be easily scaled only by vertical directions (migrate database records on the more powerful hardware) because the relational restrictions and standard database normal forms complicate scaling databases across many computational nodes.

We can find the use case when the applications have to write and read simple structured data fast from the database, and vertical scaling is almost impossible because the application has to run nonstop. In this situation, document-based databases are more suitable than table-based databases. Document-based databases do not require any attribute definition before using the database instance to compare with table-based databases, and document-based databases enable them to scale capacity horizontally (to increase the storage is efficient, only add the new hardware to the computational cluster). The document-based database engine can easily replicate and query data across all physical hardware connected into one computational cluster, similar to data stored in one hardware [18].

Therefore, we prefer to use a document-based database for our architecture. The database enables modifying the purpose application with minor changes in back-end and front-end layers according to the use case scenario (instead of a parking management system, the CV methods are used for a traffic surveillance system). Data stored in JSON format directly in the database are suitable for direct output response in REST API service endpoints without postprocessing from the database's loaded data in the back-end application layer. The famous document-based database used in this work is MongoDB [19].

Notifications are one of the important features in the front-end application of our parking management system. They can be implemented in two ways: client-side and server-side.

Client-side notification uses the principle of repeated requests to the server and compares the data stored in cache storage with the server's actual data. HTML5 pages use cookies, local, or session storage as cache storage.

The server-side notifications are based on sending Server-Sent Events (SSE) technologies. SSE is an approach implemented on the HTTP protocol layer, so this technology supports all web and mobile applications supporting the HTML5 paradigm. If the internal logic of the back-end application identifies the data changes, the application automatically sends SSE for all connected clients to the server. The clients register the event listener to the server's endpoint and passively observe the changes on the server. MongoDB connector to Python includes the database change observer. This observer is suitable for implementing SSE endpoint for clients (front-end) applications [20].

The front-end application shown in Fig. 4 was implemented using HTML5 technologies, and the layout design was based on the Bootstrap framework. Simple vanilla JavaScript with JQuery framework consumes REST API services from back-end, including SSE. The improvement of this application compared with [10, 13] is the feature to observe every parking slot independently. In these works [10, 13], we consider focusing on observing only one parking slot.

6. PERFORMANCE EVALUATION

With the CV methods used in this back-end application, we tried to evaluate classification metrics on the open dataset from the perspectives of CCTVs and aerial drones.

Fig. 5 The image on the left shows the parking lots from the Aerial's perspective [21]. The image on the right shows the parking lots from the CCTV's perspective [22].

Confusion matrixes evaluate the classification metrics. There were used two video samples with the mentioned perspectives. These samples were time-lapse videos, which were divided into independent frames saved as JPEG pictures.

In Table 1, we evaluate the parking lots across 147 images sampled from time-lapse video. In one image, we count 17 parking slots. Every parking slot is evaluated individually in every 147 images. The taken images are from the classical CCTV's camera perspective. The final results are 98.31% Accuracy, 69.5% Precision, 85.54% Recall and 75.9 F1 Score.

Table 1 The Confusion Matrix of CCTV's perspective shown in Fig. 5

		Predicted	
		Positive	Negative
Actual	Positive	66	13
	Negative	29	2380

Table 2 describes the possibility of using the same approach as in the CCTV perspective. The final results are 83.11% Accuracy, 81.34% Precision, 99.73% Recall and 89.61 F1 Score. The limit of this approach in the aerial images is the resolution detail. When the aerial image is made from a satellite, the resolution is approximately 50 cm per pixel. It means that data from 50 square cm of the ground made the summarised one color from RGB or other channels. If we want to get more details, it is necessary to use drones in the area of interest. Our approach requires stable masks, so the drones must fly on the preprogrammed trajectory.

Table 2 The Confusion Matrix of Aerial's perspective shown in Fig. 5

		Predicted	
		Positive	Negative
Actual	Positive	225000	600
	Negative	51600	31800

We used the same approach in an open annotated dataset [23] as in the CCTV's perspective [22]. Fig. 6 shows the dataset mentioned in [23], which is robust for all possible weather conditions. Metrics such as Accuracy, Precision, Recall, and F1 Score in the dataset [23] have similar percentage levels to the CCTV's perspective dataset [22] shown in Table 1.

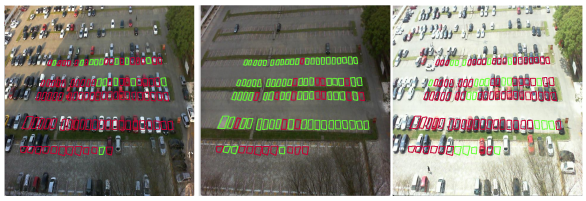

Fig. 6 The images show us the classification performing on the open testing dataset mentioned in study [23].

The results are quite good for the conditions avoiding the use of DL or ML approaches. The final summarisation of the experiments is written in the following section.

7. CONCLUSION AND SUMMARY

There are many approaches that can check the free parking lots as a part of automatic parking system management. The automatic parking system management can be built by intrusive sensors or the camera surveillance system.

In this paper, we focused on the techniques using the camera system and CV methods. The main idea was the implementation of the CV methods on the edge computing layer. The DL models require powerful computation capacity, and therefore, the standard solution is the connection edge layer (devices) to the fog layer, where complex DL models are deployed for classifications.

We decided to use the traditional CV methods for the extraction of key features. These traditional CV methods do not require high computational power, so they are more suitable to deploy on the edge layer. When we proposed the exact CV methods to classify the parking lot free spaces, we expected significantly less accuracy than DL methods. The evaluated results show us that the traditional CV methods used and described in Section 4 are comparable to DL models in classifying simple events such as checking parking slots. These methods also show us the possibility of making virtual sensors by connecting entities like cameras, SBC, and suitable CV methods into one independent smart sensor. Our methods have disadvantages, such as the sensitivity of the clear view. For example, when someone builds between the parking lot and the camera's view of a new object, such as a lamp, the ORB key point extractor would incorrectly classify the lamps as cars on the parking lot. Another disadvantage is that making a binary mask with the area of interest (parking lot borders) is time-consuming.

The parking lot occupancy methods and algorithms used in this study have some limitations regarding accurate classification. They are developed for advanced digital cameras. These cameras use special firmware that reduces the hazes and noises resulting from the shutter action in various weather conditions such as intense sunlight [24], fog [25], snow [26], or rain [27]. The image sensors and their firmware use varying techniques on the hardware [28] and software [29,30] levels to remove haze and noise caused by outdoor conditions. These techniques have been optimized for cheap production [24,28]. Consequently, cameras of the new generation have a built-in mechanism for high-quality image preprocessing in any weather conditions. Our developed methods and algorithms rely on this mechanism. The frames obtained from older cameras without any digital preprocessing mechanism make the methods and algorithms implemented in this study weather-sensitive. Specifically, they affect the classification of parking lot occupancy.

In this paper, there was described the whole back-end and front-end applications. There was a focus on modularity and simplicity in using the technologies that enable the repurposing of the same CV methods for other management systems, such as traffic measurement, etc. The applications are also easily deployable because the front-end applications use Content Delivery Networks (CDN)s, and Python programming language has good community support for automatic library management systems. The suitable docker file enables the application's configuration on

any server with minimal knowledge of using cloud technologies.

ACKNOWLEDGEMENT

This paper was supported by EDEN: EDge-Enabled intelligeNt systems, project number VEGA 1/0480/22. (100%).

REFERENCES

[1] L. BAO, Q. WANG, and Y. JIANG, "Review of Digital twin for intelligent transportation system," in *2021 International Conference on Information Control, Electrical Engineering and Rail Transit (ICEERT)*, 2021, pp. 309–315.

[2] S. M. A. SHEIK., N. JOSHI, B. GEORGE, and L. VANAJAKSHI, "Application of random forest algorithm to classify vehicles detected by a multiple inductive loop system," in *2012 15th International IEEE Conference on Intelligent Transportation Systems*, 2012, pp. 491–495.

[3] L. MAINETTI, L. PALANO, L. PATRONO, M. L. STEFANIZZI, and R. VERGALLO, "Integration of RFID and WSN technologies in a Smart Parking System," in *2014 22nd International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 2014, pp. 104–110.

[4] K. MICKO, P. PAPCUN, and I. ZOLOTOVA, "Review of IoT sensor systems used for monitoring the road infrastructure," *Sensors*, vol. 23, no. 9, p. 4469, 2023.

[5] S. JUNG, U. LEE, J. JUNG, and D. H. SHIM, "Real-time Traffic Sign Recognition system with deep convolutional neural network," in *2016 13th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, 2016, pp. 31–34.

[6] A. JAIN and K. SHAMBAVI, "Energy Efficient Smart Parking System Using Peizoelectric Material And Sensors."

[7] S. VITEK and P. MELNICUK, "A Distributed Wireless Camera System for the Management of Parking Spaces," *Sensors*, vol. 18, no. 1, 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/1/69

[8] W. A. JABBAR, C. W. WEI, N. A. A. M. AZMI, and N. A. HAIRONNAZLI, "An IoT Raspberry Pi-based parking management system for smart campus," *INTERNET OF THINGS*, vol. 14, JUN 2021.

[9] M. AWALUDIN and V. YASIN, "Application Of Oriented Fast And Rotated Brief (Orb) And Bruteforce Hamming In Library Opencv For Classification Of Plants," *JISAMAR (Journal of Information System, Applied, Management, Accounting and Research)*, vol. 4, no. 3, pp. 51–59, 2020.

[10] K. MICKO, F. BABIC, P. PAPCUN, and I. ZOLOTOVA, "Temporary parking via computer vision and deep learning," in *2022 IEEE 20th Jubilee World Symposium on Applied Machine Intelligence and Informatics (SAMI)*. IEEE, 2022, pp. 000 455–000 460.

[11] Z. SHANGGUAN, L. WANG, J. ZHANG, and W. DONG, "Vision-Based Object Recognition and Precise Localization for Space Body Control," *International Journal of Aerospace Engineering*, vol. 2019, p. 7050915, Mar 2019. [Online]. Available: https://doi.org/10.1155/2019/7050915

[12] R. SZELISKI, *Computer vision: algorithms and applications*. Springer Nature, 2022.

[13] K. MICKO and F. BABIC, "Evidencia dočasného parkovania pomocou metód počítačového videnia a umelej inteligencie," *Electrical Engineering and Informatics XII*, pp. 61–64, 2021.

[14] L. P. COELHO, "Mahotas: Open source software for scriptable computer vision," *arXiv preprint arXiv:1211.4907*, 2012.

[15] B. TUOMANEN, *Hands-On GPU Programming with Python and CUDA: Explore high-performance parallel computing with CUDA*. Packt Publishing Ltd, 2018.

[16] N. IDRIS, C. F. M. FOOZY, and P. SHAMALA, "A generic review of web technology: Django and flask," *International Journal of Advanced Science Computing and Engineering*, vol. 2, no. 1, pp. 34–40, 2020.

[17] F. GESSERT, W. WINGERATH, S. FRIEDRICH, and N. RITTER, "NoSQL database systems: a survey and decision guidance," *Computer Science-Research and Development*, vol. 32, pp. 353–365, 2017.

[18] G. D. SAMARAWEERA and J. M. CHANG, "Security and Privacy Implications on Database Systems in Big Data Era: A Survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 1, pp. 239–258, 2021.

[19] A. BOICEA, F. RADULESCU, and L. I. AGAPIN, "MongoDB vs Oracle – Database Comparison," in *2012 Third International Conference on Emerging Intelligent Data and Web Technologies*, 2012, pp. 330–335.

[20] B. JOSHI and B. JOSHI, "Event Handling," *Beginning jQuery 2 for ASP. NET Developers: Using jQuery 2 with ASP. NET Web Forms and ASP. NET MVC*, pp. 103–133, 2013.

[21] B. TOM, "Busy Parking Lot - Aerial Time-Lapse," accessed: 26/4/2017. [Online]. Available: https://www.youtube.com/watch?v=yojapmOkIfg

[22] K. NOOR, "how to yolov5 counting vehicles in parking area yolov5 vehicles detection computer vision," accessed: 11/2/2023. [Online]. Available: https://www.youtube.com/watch?v=9Wf7_2Bv5No

[23] M. GREGOR, R. PIRNIK, and D. NEMEC, "Transfer Learning for Classification of Parking Spots using Residual Networks," *Transportation Research Procedia*, vol. 40, pp. 1327–1334, 2019, tRANSCOM 2019

13th International Scientific Conference on Sustainable, Modern and Safe Transport. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352146519303527

[24] A. IGNATOV, R. TIMOFTE, and et al., "PIRM Challenge on Perceptual Image Enhancement on Smartphones: Report," in *Computer Vision – ECCV 2018 Workshops*, vol. 5, no. 15, Munich, Germany, September 2018, pp. 315–333.

[25] B. DAS, J. P. EBENEZER, and S. MUKHOPADHYAY, "A comparative study of single image fog removal methods," *The Visual Computer*, vol. 38, no. 1, pp. 179–195, Jan 2022.

[26] B. CHENG, J. LI, Y. CHEN, and T. ZENG, "Snow Mask Guided Adaptive Residual Network for Image Snow Removal," *Computer Vision and Image Understanding*, vol. 236, p. 103819, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1077314223001996

[27] X. FU, J. HUANG, X. DING, Y. LIAO, and J. PAISLEY, "Clearing the Skies: A Deep Network Architecture for Single-Image Rain Removal," *IEEE Transactions on Image Processing*, vol. 26, no. 6, pp. 2944–2956, 2017.

[28] M.-C. LU, "Advancement of Chip Stacking Architectures and Interconnect Technologies for Image Sensors," *Journal of Electronic Packaging*, vol. 144, no. 2, p. 020801, 09 2021.

[29] A. S. PARIHAR and A. JAVA, "Densely connected convolutional transformer for single image dehazing," *Journal of Visual Communication and Image Representation*, vol. 90, p. 103722, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1047320322002425

[30] G. SAHU, A. SEAL, D. BHATTACHARJEE, M. NASIPURI, P. BRIDA, and O. KREJCAR, "Trends and Prospects of Techniques for Haze Removal From Degraded Images: A Survey," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 6, no. 4, pp. 762–782, 2022.

Received December 1, 2023, accepted January 8, 2024

BIOGRAPHIES

Kristián Mičko graduated (MSc) from the Department of Cybernetics and Artificial Intelligence, Faculty of Electrical Engineering and Informatics, the Technical University in Košice. Since 2021, he has been pursuing a Ph.D. at the same university. His research is related to the Internet of Things (IoT), Computer Vision, and IoT architectures.

Peter Papcun graduated (MSc) from the Department of Cybernetics and Artificial Intelligence, Faculty of Electrical Engineering and Informatics, the Technical University in Košice. Since 2015, he has been pursuing a Ph.D. at the same university. His research is focused on the Industry 4.0, Cyber-Physical Systems (CPS) and Industrial Internet of Things (IIoT). Since 2021, he has started an associate professor career in the same Department.