Monitoring of parking space occupancy via UAVs

O. Kainz, K. Briškár, M. Michalko, F. Jakab, I. Nováková, P. Feciľak
Department of Computers and Informatics, *Technical University of Kosice*, Kosice, Slovakia
ondrej.kainz@tuke.sk, kristian.briskar@tuke.sk, miroslav.michalko@tuke.sk, frantisek.jakab@tuke.sk,
ivana.novakova@tuke.sk, peter.fecilak@tuke.sk

***Abstract*—The research aims to analyze, design, and implement a prototype system for recognizing the current occupancy of the parking lot using visual input from an unmanned aircraft. The experimental solution should also be able to detect, identify and recognize the vehicle registration number and store it in a database. The character recognition process itself is done separately from the vehicle park occupancy analysis. The recognized license plates are stored in a database, which keeps a record of occupancy at a given time. The technique for defining the area of interest was deployed with a subsequent analysis of the object in the area. The overall success of the applied method exceeded 95 %, which indicates the reliability of the implemented method.**

***Index Terms*—computer vision, digits, character recognition, OpenCV, parking, Python, vehicle detection**

I. Introduction

Sight belongs as the main information resource for humans and animals as well. Although shape, colors, light, and many other recognitions are seeming to be easy to categorize, find similarities, and recognize by their nature, we learned them as we got older.

These processes become natural for us as we use them every day and every minute of our life. Computer vision is nowadays used in vast fields of use. From statistics through security to development. Computer vision is a fast-growing field not only in the development of artificial intelligence but already applied on a day-to-day basis.

Characters and number recognition are essential for humans, as they allow us to pass and maintain data. Although this data does not have the same meaning for computers as they have for humans, the goal of this work is to help human beings. When combined with object recognition there is the prospect to help recognize free parking spots and ease parking in locations with limited capacity and high visit rates. Solution may be used even for the electric vehicle [1] charging spots.

II. Analysis of Detection and Recognition Processes

A. Edge detection

The basics of every image processing program in computer vision are based on the recognition of different colors, objects, lights, or changes between given images. With this knowledge, we need to determine the fundamentals of the approach to solving a problem we are dealing with. Detecting edges is one of the most reliable approaches to finding borders characterizing areas, which represent some specific objects and what is background in frames where we run edge detection. Exist more options on how to detect edges.

1) Zero crossing edge detection: The theorem of edge detection based on crossing zero is to find the point where the intensity of pixels in the image given for analysis changes rapidly. This approach at first appeared as a filter for frequency diagram analysis, but since can be considered after using correspondent kernel for image transformation, change of the color can be represented as frequency by calculating 1st and 2nd derivation [2] of given the image.

2) Perwitt operator edge detection: In images we standardly detect two-dimensional(2D) coordinates system. Generally, Prewitt operator [4] uses a system of 3x3 matrix in this 2-dimensional field.

Since this method is based on relatively simple mathematical calculations, seems logical, that fast processing speed and lower computation requirements will be the main advantage. Yet there is a disadvantage in form of noise sensitivity which caused higher requirements on input image quality.

3) Sobel edge detection: The main difference between the Perwit operator and Sobel [5] operator is the usage of 2nd power in calculations for mask application square root of $(Gx^2 + Gy^2)$ where Gx represents horizontal representation of coordinates and Gy vertical then. G represents gradient.

The main advantage in comparison with the Perwit operand is less noise sensitivity, thus the output can be relatively cleaner in comparison with some cases of not-so-good quality input data.

B. Canny edge detection process

The process developed by John F. Canny in 1986 for edge detection became widely used mainly in Python programing language and C++ since was rooted in the OpenCV library for computer vision. The process itself contains 5 main steps [6]:

- noise reduction,
- gradient calculation,
- non-maximum suppression,
- double threshold,
- edge tracking by Hysteresis.

C. Detection and recognition of objects

Since we start recognizing and detecting objects in frames from the bottom, it would be appropriate to look at it in a more complex way. We will look at more approaches for detecting objects such as the utilization of trained neural network.

979-8-3503-2033-6/22/$31.00 ©2022 IEEE

D. Detection of objects via deep learning

To understand the term deep learning we must suppose a gain of certain ability to computers. Deep learning as part of machine learning is a process when computer machines achieve entry-level artificial intelligence. Algorithms can recognize patterns after the training process were made. This means providing a dataset with marked objects we want to recognize. The algorithm then trains their ability to recognize given patterns through similarities in samples. Such algorithms [8] include:

- Region-based convolutional neural network (RCNN)
- Faster-RCNN
- Single shot detector(SSD)
- You only look once (YOLO)
- Support Vector Machines (SVM)

As a part of this research, we will take a closer look at YOLO. YOLO algorithm is assembled on the principle of using only one forward propagation on a neural network to detect the object. As a result, most of the time quicker results in comparison with other CNN algorithms but with the cost of higher error occurrence. Despite that, there is still a good error/speed ratio with results, and is widely used for his still sufficient quality of results and speed. The input in YOLO is divided into an SxS grid. The appearance of the object center in one of the grid cell gives responsibility for recognition to these cells. As Du [7] explains, each grid cell predicts B bounding boxes, confidence scores for those boxes, and C class probabilities of the grid. These predictions are encoded as an $SxSx(Bx5 + C)$ tensor. Nowadays exist an updated version of YOLO, in particular, YOLOV3 is in use.

E. Detection and recognition of numbers

It became a common practice to recognize numbers or letters in the image, extract and save them in digital form for the next processing. This is quite a useful method. Nowadays is a common practice to interact with the system using this feature.

1) Optical character recognition: Optical character recognition (OCR) aims to extract machine-encoded characters from digital images by applying electronic conversion. Pytesseract library seems the best option, as the solution will be using the Python programming language, [9] for OCR not only to be used on characters like letters but also numbers. The whole process is based on a few following steps [10]:

- Scanning.
- Segmentation.
- Pre-processing:
  - Resize.
  - Convert color.
- Feature extraction/localization:
  - Highlightning characters.
  - Suppressing background.
- Recognition.

The process itself was optimized for common use in various sectors as the private sector and public sector. Pytesseract library was created as an OCR tool for Python. The main function is to mimic the ability of humans to read text from any given image.

F. Recognition of digits with Convolutional Neural Network

We will take a closer look at the use of CNN to recognize a digit. Many approaches to CNN are applied using TensorFlow which is a neural network library in Python. The method we will describe [11] uses a different combination of hidden layers for the process of analysis.

After gray scaling, 5 layers are applied. Here are the responsibilities of each layers:

1) Feature extraction from data input.
2) Convolution operations accomplishing.
3) Reduce the number of constraints and output information.
4) Dense layer (connect layers to determination layer to reduce over-fitting).
5) Determination of digits number (0 to 9).

G. Digit recognition based on template matching

Using template matching methodology for the identification of numbers brings some advantages like the utilization same format in a license of vehicle registration. This brings two ways how to look at the problem. The first can be a template of a rectangle object with a blue strip on the left side. The second can be a template of font, which is normalized for all identification pates. Using a font template seems to be a better fit for the situation with template matching considering a bigger variety of countries have the same fonts but a similar sequence of format. This brings to mind the possible procedure of detection. The sequence of steps for template matching were presented by Roy et al. [12]:

1) Input of image.
2) Plate recognition (only plate not digits).
3) Skew correction.
4) License plate extraction.
5) Segmentation.
6) Digit classification – recognition.

Plate recognition can be done based on the color scheme of plates since all have the same format. For skew correction is possible to apply a rotation matrix to determine the same viewpoint as a template has. This brings us to segmentation which is used for the determination of the position of digits and separate areas for digit classification. Roy et al. [12] used during experiment 180 colored images however with different quality. This could affect the result of 88% of successful recognition.

H. Comparsion of digit recognition algorithms

Now we can have a look at the comparison of those algorithms. If we take a 7-segment approach, though has been accurate, it is not possible to apply such an approach to different fonts henceforth we cannot choose this approach in our problem since there is a necessity for the ability to detect other than 7-segment numbers. The ability to detect digits with

CNN seems to be very accurate with a success rate of 97%, however, the amount of data used to create such an accuracy indicates the necessity of a large dataset and the time needed to learn a model of such ability. Template matching does not require of large dataset and can perform on live video as Roy et al. [12] describe it as a fundamental technique to license plate recognition(LPR) but with the success of recognizing attacking 88% leaves the lowest percentage of possible algorithms able to use on our problem. This leaves us with OCR. Method of optical character recognition shows a high level of successfully recognized digits (99%) and does not require building and training model and such a large dataset is not necessary either. OCR can also be applied to video-stream and therefore can be run simultaneously with other processes. This is the reason we choose to apply this method furthermore in this research as the best fit for finding the solution to our problem.

III. Materials and Methods

In this section, we focus our attention to design and implement a user-friendly and universal solution based on the recognition of the object. This process also with digit recognition helps us to maintain an overview of the selected parking lot. The main goal is to design a system running on a server able to process data from input and not a system running directly on the UAV itself.

A. System restrictions and requirements

We aim for a system able to process video input with full high-definition resolution (FHD) and with 30 frames per second. For testing purposes a video was captured via DJI Mavic mini personal drone. This particular model involves a sensor size of 12 megapixels and with 1/2.3" CMOS sensor. The default format of the output is mp4. Server handling process of recognition and evaluation run on Intel Xeon E5-2630 with 2.20GHz and 6 cores together with 16GB of RAM. Maximum disk space represents 400GB. As we can see, this much computing power grants us much more freedom in way of solution design. No significant computing power restrictions had to be made since there is no need to run the whole software solution on the UAV itself. In addition, video as a source of analysis also must keep up with some conditions. To begin with, light is according to assumption most significant factor. Video input with the poor light condition will cause more errors. Meaning effort for best visibility is a crucial criterion for correct recognition and classification. This restricts the usage of our solutions only during daylight or in well-lightened places without a deficit of light. A possible angle for maximizing of input quality can be seen in Fig. 1.

IV. Implementation of Experimental Solution

We created the proposed solution of software implementation as the web application with optimization primarily for desktop users, solutions were inspired by the work of authors [13], [14]. The web solution is undoubtedly a more friendly approach to a wider range of users. We can list the pros of this approach:

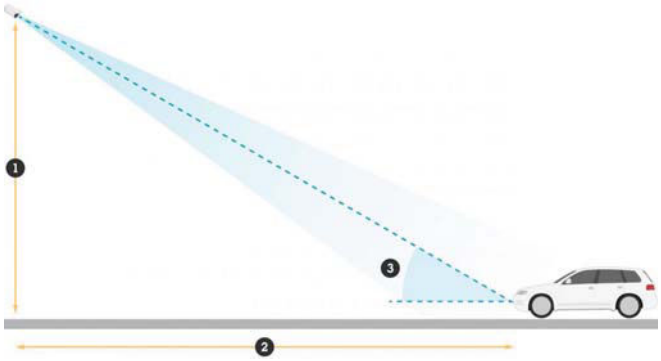

Fig. 1: Capturing the license plate of an incoming vehicle.

- No need for installation of software on the user side.
- No necessity to own machine with higher hardware specification in cases of more demanding software solutions.
- Can be opened on all platforms that support web browsers.
- Accessible to the user on every device with connectivity.

On the other hand, we had to suppose a stable internet connection. This seems to be a disadvantage. Without a connection to the internet, the software solution can not perform any action.

1) Detection of the license plate position: For the implementation of the CNN model for detection of the position of the license plate, we had to create a data file, .csv in our case. With all this data representing a dataset with which help we will get the information we demand. To fill this file with real or in other words, meaningful data, we had to manually specify an area of interest. A free online dataset was found with XML files containing data about the position of the license plate.

A. Labeling

To label additional images we use the open-source graphical image annotation tool LabelImg. Output from this process is an XML file with the set structure.

B. Data transformation

To get from XML files data in order to save as .csv we created Python script. Its whole functionality can summarize as follows: At first, all XML files are loaded. Then we define the structure of the file we will output all data, after which follows the loop function. The purpose of this function is to pull out data about box coordinates from all XML files and save it to our output file.

C. Training of model

First, we need to load data from the file. The next step will include preprocessing during which we save chosen information to arrays. Data in the array are already resized to fit the training process and also normalized. After preprocessing, the training and saving model can be reached.

For training, we use the Keras API of Tensorflow. More specifically, the function InceptionResnetV2 was utilized. It

is a combination of two functions for model training as the name suggest – function Inception and function Resnet. This function returns the Keras image classification model pre-trained on ImageNet. The model was trained in two iterations. Each iteration was set for 100 epochs with batch size 10. We managed to limit validation losses as well as time in the final 10 epochs in comparison with the first 10.

D. Predictions with model

As the first step, we load the model and then parse some data to it, in this case, the image. It is necessary to remember the size of images passed to the model during the training phase. Not because of model would work only on those images but because for correct functionality, we must pass the same size input as was learned and expected to handle. Therefore, after loading, the second step is preparation or so-called pre-processing, after which we call the model itself to make predictions on the input we provided.

E. Reading of symbols

Once we get predictions of the location of a licence plate we can proceed to the process of extraction of character in the labeled area. To do so, we apply OCR.

To make this engine functional, a path to an executable file must be defined to access the functionality of the engine. This allows us to provide a callback on the engine in proposed script. Nevertheless, for correct functionality, the only image of the labeled area must be provided if we do not wish to finish the program with empty output or with an unexpected error.

As database for this project, we choose Firebase real-time database – Firestore. This feature keeps us updated on data samples from testing functionality for this project.

F. Parking lot capacity monitoring

In this section, we describe the process behind monitoring the capacity of the parking lot. The technology used to accomplish such a goal was used in a much simpler way than the procedure behind license plate recognition and reading. After first attempts to use a similar process also for this case, there were some issues:

- To recognize the object of a vehicle we would need a massive dataset which would consist of frames of vehicles from a so-called bird's eye view.
- Frames of different light conditions would be necessary.
- Frames would have to be from the same high above the surface.

Considering all this information we gained, the search for alternative approaches to these issues continued. In the final version, we decide to promote much faster and hardware more low-cost solution. The area of interest was marked, and pre-processing of the image followed. In the final solution, we monitor the density of pixels in areas defined by us which allows us to perform fast and reliable parking lot capacity monitoring.

G. Labeling areas of interest

To gain the ability to count parking slots, a definition of those areas must be made. The solutions utilizes the Python Pickle module which provides a serialization function. With this, we can define our areas and store them for video analysis. The whole process consists of the following steps:

1) Define size for the areas which are subsequently stored in the array.
2) Catch input from mouse being clicked.
3) Append area predefined size to point where was clicked by the mouse.
4) Remove position saved in an array.

H. Counting vehicles

Once prior functionalities are defined, we can describe the functionality of the process counting process.

From the pickle file, we access the saved positions of areas of interest, prepare positions, and call the function which will load the video and prepare it accordingly. Preparation means loading and perform the next steps:

1) grayscaling,
2) blurring,
3) thresholding,
4) dilatation,
5) median(Gaussian) blur activation,
6) checking slots.

These steps have to be in this order to secure accurate functionality. Grayscale is necessary since we need to have all frames in the same color palette. When we can get only two colors (black and white) we can categorize them, or a better expression would be to set value for these colors. Border can be set to define which values represent one and which is the second color.

Blurring helps during the color recognition process since it does what it says. This process prevents unwanted behavior, especially borders, where few close areas have a high difference in the value of color and could split one object.

The next step is thresholding, which is an expression for the assignment of the pixel value with the threshold value provided. In thresholding, each pixel value is compared to a defined threshold value. If the pixel value is smaller than a threshold, it is set to 0, otherwise, it is set to the maximum value, 255.

When we have the value specified, we apply these changes. Prior to the final output, we apply the dilate functionality of OpenCV. Which, simplify said, is the filter which can connect close areas of high similarity to make bigger contours and unite them to one area. After all this, we apply Gaussian Blur to smooth the image and in the finale, the human eye will see it more as clusters of pixels than objects or contours of objects as we can observe in Fig. 2.

This enables to determine in the final output, how much white we will consider as the occupied slot. During the testing, this value was set to 620 pixels. If there is too much white pixels, the slot will be considered occupied and the counter of free slots will be updated.

Fig. 2: Output after Gaussian Blur.

TABLE I: Success rate of occupancy recognition.

Video no.	No. of slots	No. of detected vehicles	Occupied slots	Free slots	Success rate
#1	21	3	3	18	100%
#2	19	6	7	13	85.7%
#3	21	7	8	13	100%
#4	46	45	46	0	97.8%

Areas of interest for pixel counting will be represented by our definition of rectangles as represented in Fig. 3.

Fig. 3: Labeled areas of interest.

V. Evaluation

The main testing phase of parking lot capacity was made on videos from areas of the Technical University of Kosice. Two parking slots were chosen with more angles of capturing. After that, we defined areas of interest and run recognition. Overall success rate exceeded 96% as we can see represented below in Tab. I

These results prove the proper way of the selected approach. In one case we had occupied one slot more than were vehicles, but this was caused by cross parking of one vehicle. However, the detection of occupancy of those slots (even if it was only one vehicle but blocks two slots) proves a high success rate in the detection of occupied parking slots.

In the case of license plate detection and symbol recognition, we proceed series of sets of tests. The overall number of images used for testing of success rate was 17 and we evaluated 4 categories.

Firstly, we evaluated predictions of the successful localization of license plates. For success, in this case, we consider every area labeled which contains at least 50% of the license plate from the image. The next evaluation was more difficult because we consider only labels with the whole license plate inside the labeled frame. When we manage to get all those parameters, then the initiation of the OCR algorithm came into place. Those cases where OCR was able to extract some strings could be considered valid output. The final and most complex solution with a result of 59% means cases where from 17 initial inputs we got correctly read the string from the license plate.

In the testing process of license plate reading, we use various images where in a few of them license plate was of high quality and take a significant portion of the whole frame. In others, the license plate was smaller and therefore quality after extracting the license plate even in successful attempts OCR could not read character due to drop-off quality in the labeled frame.

VI. Conclusion

The research presented in this paper was focused on the detection of free or occupied parking lots using UAVs. Subsequently, from the input video, the system locates the license plate and allows the detection of its alphanumerical characters. Prior to the design and implementation, the analysis of the detection and recognition processes was carried out. This knowledge was put to use in both the design and implementation phases. The OCR technique is used for the recognition of the digits. For the detection of the parking lot's state, the region of interest is used. This is based on the presumption that the UAVs will fly to a fixed position.

Problems we were dealing with have high potential in the field of monitoring parking lots in private but also public areas. Testing proved the experimental solution to be fully functional. The average efficiency of parking lot monitoring exceeds 96%. However, the efficiency of the license plate prediction, in terms of its localization, was 69,25%.

In future research, it is expected that the UAVs will be used to capture the vehicle and parking lot while providing multiple regions of interest. Moreover, detection of the automobile and various other types of vehicles may be improved, e. g. using neural networks. Moreover transfer of data [16] may be optimized using novel IoT (Internet of Things) protocols.

Acknowledgment

This publication was realized with support of the Operational Programme Integrated Infrastructure in frame of the project: Intelligent systems for UAV real-time operation and data processing, code ITMS2014+: 313011V422 and co-financed by the Europezn Regional Development Fund.

References

[1] Pál D.; Beňa Ľ.; Oliinyk M. The Impact of Electric Vehicles On Smart Grid. Acta Electrotechnica et Informatica. Vol. 21. No. 1. 2021. pp. 35–42.

[2] Fisher, R.; Perkins, S.; Walker, A.; Wolfart, E.. Edge detectors. 2000.

[3] Clark, J.J. Authenticating edges produced by zero-crossing algorithms. IEEE Transactions on Pattern Analysis and Machine Intelligence. vol. 11. no. 1. 1989. pp. 43–57.
[4] Yang, L.; Wu, X.; Zhao, D.; Li, H.; Zhai, J. An improved Prewitt algorithm for edge detection based on noised image. In: 2011 4th International congress on image and signal processing. vol. 3. 2011. pp. 1197–1200.
[5] Israni, S.; Jain, S. Edge detection of license plate using Sobel operator. In: International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT). 2016, pp. 3561–3563.
[6] Sahir, S. Canny edge detection step by step in python-computer vision. Towards Data Science. 2019.
[7] Du, J. Understanding of Object Detection Based on CNN Family and YOLO. In: Journal of Physics¿ Conference series. IOP Publishing, 2018.
[8] CHandan G., Jain A., Harsh J. and MOHANA. Real Time Object Detection and Tracking Using Deep Learning and OpenCV. In: International Conference on Inventive Research in Computing Applications (ICIRCA). 2018. pp. 1305-1308.
[9] Hoffstaetter S.; Bochi J.; Lee M.; Kistner L.; Mitchell R.; Cecchini E.; Hagen J.; Morawiec D.; Bedada E.; Akyüz U. Python Tesseract. 2021.
[10] Mithe, E.; Indalkar, S.; Divekar, N. Optical character recognition. International journal of recent technology and engineering (IJRTE). vol. 2. no. 1. 2013. pp. 72–75.
[11] Siddique, F.; Sakib, S.; Siddque, Md.A.B. Recognition of Handwritten Digit using Convolutional Neural Network in Python with Tensorflow and Comparison of Performance for Various Hidden Layers. In: 2019 5th International Conference on Advances in Electrical Engineering (ICAEE). 2019. pp. 541–546.
[12] Roy, A.Ch.; HOSSEN, Muhammad Kamal; NAG, Debashis. License plate detection and character recognition system for commercial vehicles based on morphological approach and template matching. In: 2016 3rd International Conference on Electrical Engineering and Information Communication Technology (ICEEICT). 2016. pp. 1–6.
[13] Cymbalak, D.; Jakab, F.; Vapenik, R.; Michalko, M.; Turna, J.; Bilsky, L.; Kovacova, J. Enhanced Interoperability with New Collaborative and Streaming Functions Implemented in Slovak National Telepresence Infrastructure (NTI). In 2018 16th International Conference on Emerging eLearning Technologies and Applications. pp. 105-110. 2018. IEEE.
[14] Jakab, F.; Michalko, M.; Turna, J.; Kovacova, J.; Cymbalak, D. The experience from implementation of National telepresence infrastructure in Slovakia to support research, development and technology transfer. In 2016 International Conference on Emerging eLearning Technologies and Applications. 2016. pp. 127-132.
[15] Kainz, O.; Mak, A.; Cymbalak, D.; Kardoš, S.; Fecilak, P.; Jakab, F. Low-cost assistive device for hand gesture recognition using sEMG (Erratum). In First International Workshop on Pattern Recognition. vol. 10011. 2016. pp. 265-271.
[16] Petija R.; Jakab F.; Feciľak F.; Michalko M. Optimization of Data Transfer In The Internet Of Things Environment. Acta Electrotechnica et Informatica. vol. 21. no. 2. 2021. 13–16.