Design laboratory 2023/2024

Intelligent parking based on monitoring analysis(2023)

Jakub Lasak
Faculty of Computer Science, Electronics and Telecommunications
AGH, Krakow
klasak@student.agh.edu.pl

***Abstract*—The project assumed the creation of an application to operate a smart parking lot based on video monitoring. In addition to the software part, the project was also to have a physical layer that was to be used to present the effects of the operation. The program part was completed completely, but the physical part was not created due to the low resolution of the cameras used, which will be described later in the article. Another major problem is optimizing the CPU resources used due to their high consumption by cameras and attached libraries, especially OCR. There were many more problems of minor importance, which I will try to describe. Each of the problems listed above and below has been solved, mainly thanks to scientific studies, library documentation and other resources available on the Internet.**

***Keywords*— Parking share, IoT system, Automated parking, Image processing, OCR, Harcascade**

I. INTRODUCTION

The project basically consists of 3 main parts, the GUI, the algorithm for detecting and recognizing license plates, as well as the code handling the collected data and storing it in files so that it can be recreated the next time the program is run. A GUI, or Graphical User Interface, is a visual way for users to interact with electronic devices or software applications. It employs graphical elements such as icons, buttons, and windows, providing an intuitive and user-friendly experience. GUIs aim to simplify complex tasks by enabling users to perform actions through visual representations, reducing the reliance on text-based commands. This approach enhances accessibility, making it easier for individuals to navigate and operate various systems, from operating systems to software applications. In this project, the GUI is used to operate the entire intelligent parking lot, from its level the user has access to all application functionalities. I created the algorithm for detecting license plates, and generally speaking, inscriptions, based on the OpenCV library, which in Python is called CV2. The library is built on the foundation of various algorithms and functions that cover a broad spectrum of computer vision tasks. These tasks include image and video processing, feature detection, object recognition, machine learning, and more. OpenCV provides a vast array of tools to manipulate and analyze visual data efficiently. Key components of OpenCV include image processing functions, video analysis tools, machine learning algorithms, and computer vision utilities. It incorporates advanced techniques like Haar cascades for object detection, feature matching with and SURF, and support for deep learning frameworks such as TensorFlow and PyTorch. In this case, the ability to detect license plates is responsible for a prepared, trained machine learning model that I obtained from the repository of one of the GitHub user. Haar Cascade in OpenCV Python refers to a machine learning-based object detection technique used for identifying objects in images or video streams. This approach is particularly effective for detecting faces, eyes, and other objects with distinct features. The Haar Cascade algorithm relies on Haar-like features, which are patterns resembling edges, lines, and rectangles. These features are used to create a cascade of classifiers, with each stage refining the detection process. The cascade is trained on positive and negative samples, learning to distinguish between the target object's features and background noise. To recognize already detected license plates, I used the OpenCV library again, and I also used the EasyOCR library to directly recognize characters and symbols in the extracted image. EasyOCR is an open-source optical character recognition (OCR) library designed to simplify the extraction of text from images. The library utilizes deep learning techniques, particularly convolutional neural networks (CNNs), to achieve accurate and efficient OCR results across diverse languages. Another key problem during the project implementation was the effective use of hardware resources, only after installing CUDA and using the GPU for calculations the project began to work in real time, the handling of cameras and the GUI was a heavy burden on the processor. Compute Unified Device Architecture is a parallel computing platform and programming model developed by NVIDIA. It is designed to harness the computational power of NVIDIA GPUs for general-purpose computing tasks beyond graphics rendering. CUDA allows to accelerate applications by offloading parallelizable workloads to the GPU. One of the key advantages of CUDA is its ability to significantly accelerate computations compared to traditional CPU-based approaches. This is particularly beneficial for applications that can be parallelized, as the massively parallel architecture of GPUs allows them to process large volumes of data concurrently. The simplest and

Jakub Lasak,
Faculty of Computer Science, Electronics and Telecommunications
Electronics
klasak@student.agh.edu.pl

least time-consuming element of the 3-stage project creation was the creation and operation of the database. I used xlsx format files for this purpose and I used the pandas library to handle them.

II. BACKGROUND STUDY

A. Image

An image is defined as a two-dimensional function,F(x,y), where x and y are spatial coordinates, and the amplitude of F at any pair of coordinates (x,y) is called the intensity of that image at that point. When x,y, and amplitude values of F are finite, we call it a digital image.
In other words, an image can be defined by a two-dimensional array specifically arranged in rows and columns.
Digital Image is composed of a finite number of elements, each of which elements have a particular value at a particular location. These elements are referred to as picture elements, image elements, and pixels. A Pixel is most widely used to denote the elements of a Digital Image.

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & f(0,2) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & f(1,2) & \cdots & f(1,N-1) \\ \vdots & \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & f(M-1,2) & \cdots & f(M-1,N-1) \end{bmatrix}$$

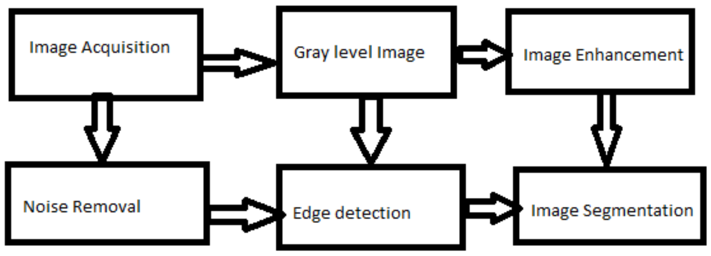
B. Image processing

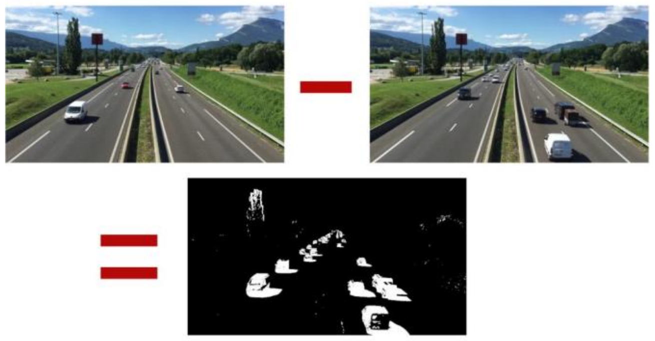
Image processing is the process of transforming an image into a digital form and performing certain operations to get some useful information from it. The image processing system usually treats all images as 2D signals when applying certain predetermined signal processing methods.
There are five main types of image processing:

- Visualization - Find objects that are not visible in the image
- Recognition - Distinguish or detect objects in the image
- Sharpening and restoration - Create an enhanced image from the original image
- Pattern recognition - Measure the various patterns around the objects in the image
- Retrieval - Browse and search images from a large database of digital images that are similar to the original image

Especially pattern recognition will be used in particular in this project to recognize license plates or detect objects appearing in the camera view. Camera image management and processing will be handled by a specially designed algorithm written in Python. It will do this with the help of two specialized libraries: OpenCV and EasyOCR.
When it is integrated with various libraries, such as NumPy which is a highly optimized library for numerical operations, then the number of weapons increases in your Arsenal i.e. whatever operations one can do in NumPy can be combined with OpenCV. The OpenCV library, like any other library of its kind, uses fundamental steps to process an image:

- Image Acquisition
- Image Enhancement
- Image Restoration
- Color Image processing
- Wavelets and Multiresolution Processing
- Compression
- Morphological Processing
- Segmentation
- Representation and Description
- Recognition





C. Parallel computations

Parallel computing, particularly leveraging the power of GPUs, has become increasingly prevalent in scientific and computational tasks. Python, being a versatile and widely used programming language, has seen the integration of various libraries and frameworks to facilitate parallel computation. One such library is imutils, and when combined with CUDA and GPU acceleration, it opens up new avenues for high-performance computing. ‘Imutils’ is a Python library primarily focused on image processing and video stream handling. It simplifies common tasks, making it an attractive choice for developers working on computer vision and image-related applications. When coupled with GPU acceleration through CUDA, the performance gains can be substantial. Numerous scientific articles highlight the benefits of parallel computing using Python and GPU acceleration. They emphasize the efficiency and speed achieved when harnessing the parallel processing capabilities of GPUs for computationally intensive tasks. These tasks include image processing, deep learning, simulations, and more. Research papers often delve into the technical aspects of integrating CUDA with Python libraries, showcasing how the parallel execution of code on GPUs significantly enhances the overall performance. These studies frequently provide code snippets and examples illustrating the implementation of parallel algorithms in Python, emphasizing the seamless integration of CUDA for GPU computing.

III. PROBLEM FORMULATION

A. Creating GUI:

Creating a GUI in Python is relatively simple, but writing the code manually is quite time-consuming and requires knowledge of a lot of documentation. Many solutions, especially at the beginning of the journey with libraries from the Qt family, need to be checked and tested, which again lengthens the implementation process.

B. Monitoring system:

The resulting model should be equipped with cameras in the transition zone where vehicles will enter and leave the parking lot, in the traffic zone to monitor their driving, and in parking zones to control the parking lot occupancy and identify vehicles and assign them to occupied parking spaces.

C. Algorithms:

The algorithms used in the project should recognize vehicles based on their license plates, parking spaces and traffic zones. Identify occupied places and send information to the database. And also, direct traffic if necessary.

D. Data storage:

Data collected in the hardware layer should be processed and sent to an external file for storage and statistics.

E. Data management:

The implementation of the project requires that it be managed by a single-board computer and a microcontroller that directly controls electronic devices included in the model, such as servomechanisms, sound sensors, lighting, and other sensors included in the project.

IV. SOLVING PROBLEMS

A. Creating GUI:

The GUI was created using a dedicated tool called 'Qt designer'. This is a program that allows you to manually create an interface using ready-made blocks. The file created and saved in this way should be imported into the program using the PyQt5 library; access to individual blocks is possible from the code level.

B. Monitoring system:

The program uses 4 cameras to observe 4 key points of the parking lot, entrance, exit, traffic zone and parking zone. The import of data received from cameras is performed by the cv2 library and the VideoCapture() and read() functions.

C. Algorithms:

After importing the image into the program, it is converted into a gray image, then a special function available after importing it, harcascade, recognizes license plates, its arguments are the image we want to process, the scale value, and the number of neighboring pixels to be combined and processed, it returns a data matrix which contains information where the inscription was detected. Then a frame is superimposed on the camera image, which shows the user where the desired object was detected.

When you press the 'Capture photo' button, a function is called that saves a snapshot of the delivered image to memory and processes it. The first step is to convert the image to shades of gray and then filter it to remove noise. The next point of this function is to convert the image into black and white and detect the edges of objects in the image. This is done using the Canny algorithm. The Canny algorithm first removes noise using a Gaussian filter, then the gradient and u are calculated to detect strong changes in intensity and direction. The algorithm divides the edges into strong and weak, then creates straight lines along these edges. The image is binary where the edges are represented by the color white, i.e. logical 1. Then, such an image is processed to detect contours and then the function is designed to detect rectangular shapes and save the location of the points where the edges meet. The algorithm creates a mask with the dimensions of an image taken from a camera, draws the designated contours and cuts out the area marked by them, then expands, and zooms in to make the task easier for OCR algorithms. Then the previously obtained image is converted to RGB and passed to global variables.

The obtained image is passed to the readtext() function included in the EasyOCR library, which recognizes the characters contained in it and returns a matrix from which the desired string of characters can be extracted.

To speed up calculations and enable the program to run in real time, I used CUDA software from Nvidi, which allowed me to obtain additional computing power, the resources of which are intensively used by the OpenCV and EasyOCR libraries.

D. Data storage:

Car lists are stored in ‘xlsx’ files, each time the program is started they are entered into a global variable and processed by simple functions whose functionality can be learned by reading the code. After each change in the database, it is saved.

V. EVALUATION

The application's operation is not error-free. It is not always possible to recognize the registration at the first attempt, it depends on the position of the display in relation to the camera, the light intensity and the quality of the presented registration image. If appropriate conditions are maintained, i.e. during the day, the adjusted brightness of the phone's screen must be as low as possible due to the poor quality of the cameras used, as well as the appropriate positioning of the phone in relation to the camera - a distance of about 20-30 centimeters with a 6-inch display, the effectiveness of the operation is approximately 87%. In unsuitable working conditions, the efficiency drops to 30-40% and in extremely unfavorable conditions even to 0%. These values were determined by tests in various combinations of the conditions described above. The effectiveness of operation is satisfactory considering the quality of the equipment and test conditions.

VI. CONCLUSION

In conclusion, the project aimed at developing a smart parking lot application based on video monitoring has achieved significant progress in its software component. The creation of a user-friendly Graphical User Interface (GUI) allows for seamless interaction with the intelligent parking system. The algorithm for license plate detection, leveraging the OpenCV library, has been successfully implemented, showcasing the capability to recognize and extract valuable information from camera images.

Despite the successful completion of the software part, challenges were encountered in the physical layer of the project. The use of low-resolution cameras hindered the creation of the intended physical layer, impacting the overall effectiveness of the system. Additionally, optimizing CPU resources proved to be a significant concern, especially with the high consumption by cameras and the Optical Character Recognition (OCR) libraries, such as EasyOCR.

Efforts were made to address these challenges, and solutions were found through scientific studies, library documentation, and online resources. The integration of CUDA, a parallel computing platform by NVIDIA, and GPU acceleration significantly improved real-time processing capabilities and alleviated the strain on the processor.

The project showcased the importance of pattern recognition, image processing, and parallel computing in the implementation of an automated parking system. The collaboration of libraries like OpenCV, EasyOCR, and imutils, along with the utilization of CUDA, provided a robust foundation for efficient and accurate processing of visual data.

While the project faced some limitations, such as recognition efficiency being influenced by lighting conditions and camera quality, it laid the groundwork for further improvements and enhancements. The implementation of a database for data storage and management, using xlsx files and the pandas library, demonstrated a streamlined approach to handling collected information.

In summary, the project's software component successfully demonstrated the potential of intelligent parking systems, but further refinement and optimization are required to overcome challenges related to hardware limitations and environmental conditions. The journey undertaken in this project contributes valuable insights into the realm of IoT-based parking solutions, paving the way for future advancements in smart parking technology.

VII. CHANGES

The physical layer could not be implemented, but the theoretical assumptions were met. The library for recognizing extracted images has also changed, EasyOCR was used instead of TensorFlow. The proposal did not assume the use of Harcascade, which turned out to be necessary. The project was completed in a shorter time than expected, but with greater workload and without the implementation of the physical layer.

References

Basic format for books:

[1]. Bartosz Ziółko, Mariusz Ziółko: Przetwarzanie mowy. AGH 2011.

[2]. Tomasz Zieliński: Cyfrowe przetwarzanie sygnałów. WKŁ 2005.

[3]. Richard G. Lyons: Wprowadzenie do cyfrowego przetwarzania sygnałów. Wydawnictwa Komunikacji i Łączności, WKŁ 1999, 2000.

[4]. Dag Stranneby: Cyfrowe przetwarzanie sygnałów. BTC 2004.

[5]. Włodzimierz Kwiatkowski: Wstęp do cyfrowego przetwarzania sygnałów. Warszawa 2003.

[6]. Marian Pasko, Janusz Walczak: Teoria sygnałów. Wydawnictwo Politechniki Śląskiej, Gliwice 1999.

[7]. Jacek Izydorczyk, Grzegorz Płonka, Grzegorz Tyma: Teoria Sygnałów. Helion 1999.

Basic format for reports:

[8]. A Survey of Intelligent Car Parking System / Faheem, S.A. Mahmud, G.M. Khan, M. Rahman, H. Zafar // Center for Intelligent Systems and Networks Research, University of Engineering and Technology, Peshawar, Pakistan, Department of Computer Systems Engineering, University of Engineering and Technology, Peshawar, Pakistan

[9]. Playback attack detection for text-dependent speaker verification over telephone channels / Jakub GAŁKA, Marcin Grzywacz, Rafał SAMBORSKI // Speech Communication ; ISSN 0167-6393. — 2015 vol. 67, s. 143–153. — Bibliogr. s. 152–153, Abstr.. — tekst: http://www.sciencedirect.com/science/article/pii/S0167639314000880/pdfft?md5=a6b9637edc5d22ba1e17bb9dbfb17d16&pid=1-s2.0-S0167639314000880-main.pdf

[10]. Voice authentication embedded solution for secured access control / Jakub GAŁKA, Mariusz MĄSIOR, Michał Salasa // IEEE Transactions on Consumer Electronics ; ISSN 0098-3063. — 2014 vol. 60 iss. 4, s. 653–661. — Bibliogr. s. 660–661, Abstr.. — tekst: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7027339

[11]. System for multimodal data acquisition for human action recognition / Filip MALAWSKI, Jakub GAŁKA // Multimedia Tools and Applications ; ISSN 1380-7501. — 2018 vol. 77 iss. 18, s. 23825–23850. — Bibliogr. s. 23848–23850, Abstr.. — Publikacja dostępna online od: 2018-02-02. — tekst: https://link-1springer-1com-1nyztlj1v00fd.wbg2.bg.agh.edu.pl/content/pdf/10.1007%2Fs11042-018-5696-z.pdf

[12]. Composition of wavelet and Fourier transforms / Mariusz ZIÓŁKO, Marcin WITKOWSKI, Jakub GAŁKA // Matematyka Stosowana : pismo Polskiego Towarzystwa Matematycznego = Mathematica Applicanda ; ISSN 1730-2668. — 2018 vol. 46 no. 1, s. 159–168. — Bibliogr. s. 166–167, Abstr., Streszcz.. — J. Gałka - pierwsza afiliacja: University of Warsaw. — tekst: https://wydawnictwa.ptm.org.pl/index.php/matematyka-stosowana/article/view/6376/5896

[13]. Wavelet speech feature extraction using mean best basis algorithm / Jakub GAŁKA, Mariusz ZIÓŁKO // W: Advances in nonlinear speech processing : international conference on Nonlinear speech processing, NOLISP 2009 : Vic, Spain, June 25–27, 2009 : revised selected papers / eds. Jordi Solé-Casals, Vladimir Zaiats. — Berlin ; Heidelberg : Springer-Verlag, 2010. — (Lecture Notes in Artificial Intelligence ; ISSN 0302-9743 ; LNAI 5933). — ISBN: 978-3-642-11508-0 ; ISBN10: 3-642-11508-X. — S. 128–135.

[14]. Spectral features of the clarinet sound revealed by the set of STFT-based parameters / Tomasz J. WILCZYŃSKI, Len Gelman, Piotr KLECZKOWSKI // W: 18th WCNDT [Dokument elektroniczny] : 18th World Conference on Non-Destructive Testing : 16–20 April 2012, Durban, South Africa

[15]. Proposed deep learning pipeline for an automatic covid-19 detection for medical images / Nedjoua houda kholladi // Eloued University, Algeria