DOI: https://doi.org/10.25140/2411-5363-2025-3(41)-40-45
UDC 621.8

Marek Sukop[1], Rastislav Jurko[2]

[1]Professor, Professor of the Department of Production Systems and Robotics
Technical University of Košice, (Košice, Slovakia)
E-mail: marek.sukop@tuke.sk. **ORCID:** https://orcid.org/0000-0001-7987-3557
ResearcherID: AAH-5495-2019. **Scopus Author ID:** 36615762200

[2]Engineering student of the Department of Production Systems and Robotics
Technical University of Košice (Košice, Slovakia)
E-mail: rastislav.jurko@student.tuke.sk

IMPLEMENTING COMPUTER VISION FOR A MOBILE ROBOT

This article focuses on the creation of a sample application that demonstrates the use of computer vision in mobile robotics. This was accomplished using a Raspberry Pi and four types of tags. The goal was to prove that it's possible to create a relatively simple and low-cost image processing solution that's applicable for use in mobile robotics. The article describes the algorithms used as well as the implementation process on the hardware. In conclusion, the article outlines the results achieved and provides recommendations for further research.

Keywords: *mobile robot; image processing; cascade; Raspberry Pi.*
Fig.: 5. References: 7.

Relevance of the research. Nowadays, at every step we encounter the requirement of controlling the movement of mobile robots based on the recognition of its surroundings. There is a requirement that the robot can recognize where it is in space at any time and also how the obstacles in its surroundings are arranged. At the same time, there is a requirement for the lowest possible price of the hardware part of the robot. That is why it is necessary to optimize and implement relatively complex algorithms into relatively cheap hardware platforms. A critical requirement is then the real-time response after image processing, which needs to be addressed.

Problem statement. Real-time image processing on low-cost platforms is a challenge. Since the image is mostly in a two-dimensional field (monochromatic), image processing algorithms are forced to "run" through this field several times, which is computationally demanding. Therefore, it is necessary to optimize and test simplified image processing algorithms and thereby increase the speed of image processing despite reduced reliability.

Analysis of recent research and publications. Artificial intelligence is the ability of technological devices to perform tasks like human abilities, such as reasoning, creativity, planning, and learning. Artificial intelligence systems can operate autonomously and adapt their behaviour based on previous steps. Thanks to artificial intelligence, technical systems can distinguish the surrounding environment and solve identified problems to achieve a specific goal. To do this, a computer obtains data from various sensors, such as cameras, which are then processed and interpreted. Image and video processing plays a significant role in the field of artificial intelligence [2]. The main goal of the work was to prepare computer vision and self-object detection to a functional phase suitable for demonstration, so that the subsequent implementation of the entire solution into a mobile robot remains just a detail.

Uninvestigated parts of a common problem. In addition to research goals, the motivation for creating an image processing application on the Raspberry Pi was also for educational purposes. During the educational process, it is appropriate to show students how some image processing algorithms work. It is also appropriate to show them a platform other than a PC and the performance of algorithms running on a given platform. Also, the implementation of algorithms on different platforms may differ. When testing the proposed application, students can also edit and change the photos used in the cascade and thus test and verify the stability of the recognition algorithms.

Research objective. The aim of the article was to run an optimized image recognition algorithm (cascade classifier) for the recognition of four different images. Of course, the result is not optimal, since the algorithm needs more samples. However, according to the computational complexity, it was not, and therefore the proposed algorithm was implemented with such a few trained images that it was not too convenient to delay the evaluation.

The statement of basic materials. The Raspberry Pi is a platform that, thanks to its flexibility and low cost, has become popular not only among teachers, students, and tech enthusiasts but also among developers and even industrial companies [3]. Users can create their own applications and programs using various programming languages, such as Python, Java, C++, and others. Of course, there are also many open-source projects available online that enable and help users more easily create their own projects and programs. This platform is ideal for development projects and IoT applications, such as smart homes, temperature and humidity measurement, process automation, creating your own gaming consoles, or, as in the case of this thesis, reading road signs using computer vision and evaluating the results. Overall, it can be said that the Raspberry Pi is a great tool for development and experimentation, giving users the ability to create their own projects and solve various problems using technology [1].

The first boot of a Raspberry Pi is completely different from other computers or mobile phones. To function, the Raspberry Pi uses an operating system called Raspbian, which first needs to be installed on an SD card that is then inserted into the hardware. Therefore, in addition to the Raspberry Pi microcomputer itself, we will also need a helper computer with a Windows or macOS operating system, which is only used for the initial installation, and an SD card. If a user is familiar with the Linux OS, they will also enjoy Raspbian, as the Raspbian OS is based on this platform. It is available in two versions: Lite and Pixel. The Lite version contains only the basic components of the operating system and is controlled via a terminal window, i.e., a text command line. This version is suitable as a small Linux server or for use without a monitor, for example, to control logic in home automation and the like. The Pixel version includes a graphical user interface (GUI), the tkinter library, and many pre-installed applications [4].

To solve the control and, more importantly, to utilize computer vision, a few key steps must be completed:

Installation of the libraries (OpenCV), connecting the display and camera to the board,

Creating your own database to detect only the objects you need.

The OpenCV library was invented by Gary Bradsky in 1999. It is based on optimized C/C++ code and supports Java and Python. OpenCV has more than 2,500 optimized algorithms, including an extensive collection of computer vision and machine learning algorithms. With it, it is easier to perform tasks that may seem complex at first glance, such as face identification and recognition, object identification, classifying human actions in videos, tracking moving objects, extracting 3D models of objects, and much more [5].

To create our own cascades, we used the Cascade Trainer GUI application [6] for this thesis. When creating a custom cascade with this application, it's crucial to prepare your materials in advance. The first necessary step is to create a file containing two folders. The folder names must be n and p. The content of these folders is essential for the detection to work correctly. In the n folder, we place negative images–images we don't want to be detected. In our case, these are houses, cars, trees, and roads (Fig. 1).

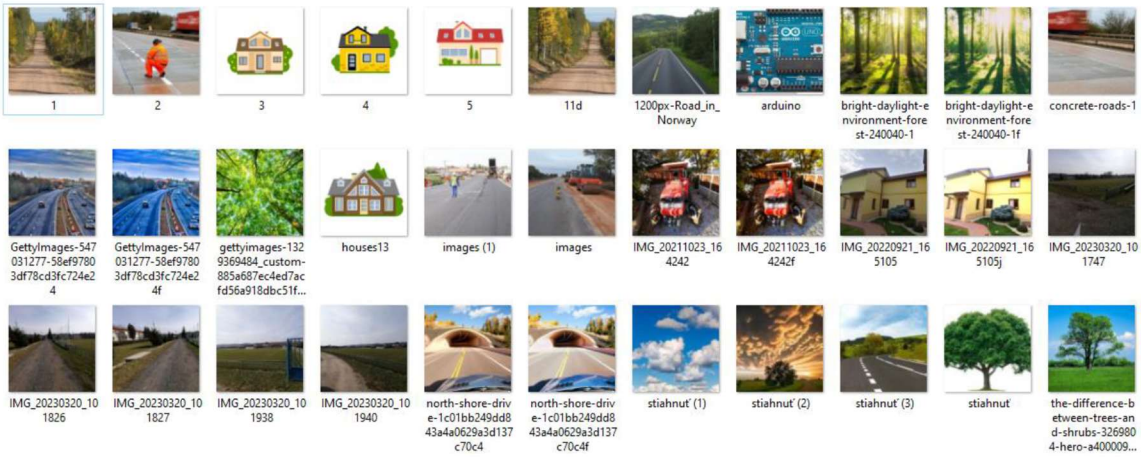

Fig. 1. Image file for negative detection [1]

The images placed in the folder named p are illustrative examples of STOP signs from various angles, with different brightness values, and in some cases, we also used filters and rotation (Fig. 2). The rule is that the more comparison images the file contains, the more accurate, faster, more efficient, and more reliable the detection will be. The typical number of images needed for the software to detect an object is in the hundreds to thousands. As you can see in the following image, we used 32 images in our case. For simple detection, this number is sufficient, but it is not 100 % successful. Also, when working with a camera, external light always plays a crucial role, as it drastically affects computer vision.

Fig. 2. Image file for possitive detection [1]

A crucial detail when saving images for detection is their format. They must all have the same resolution and aspect ratio. If this is not followed, the application will display an unidentifiable error. In our case, we chose an aspect ratio of 1:1 and a resolution of 800x800 for all images.

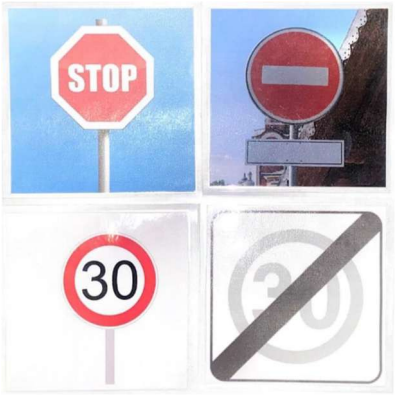

Fig. 3. Tags ready for detection [1]

The entire procedure explained in this chapter was repeated four times, as the goal of this thesis is to detect and create custom cascades for multiple road signs. In our case, these were the road signs for “STOP”, “Yield”, “Speed limit 30 km/h”, and “End of speed limit” (Fig. 3).

The program, written in Python for this specific application, has 162 lines. The programming was done continuously, and we progressively fine-tuned the details until we had a functional program code. Flowchart diagram of the main part of the written code is on the Fig. 4.

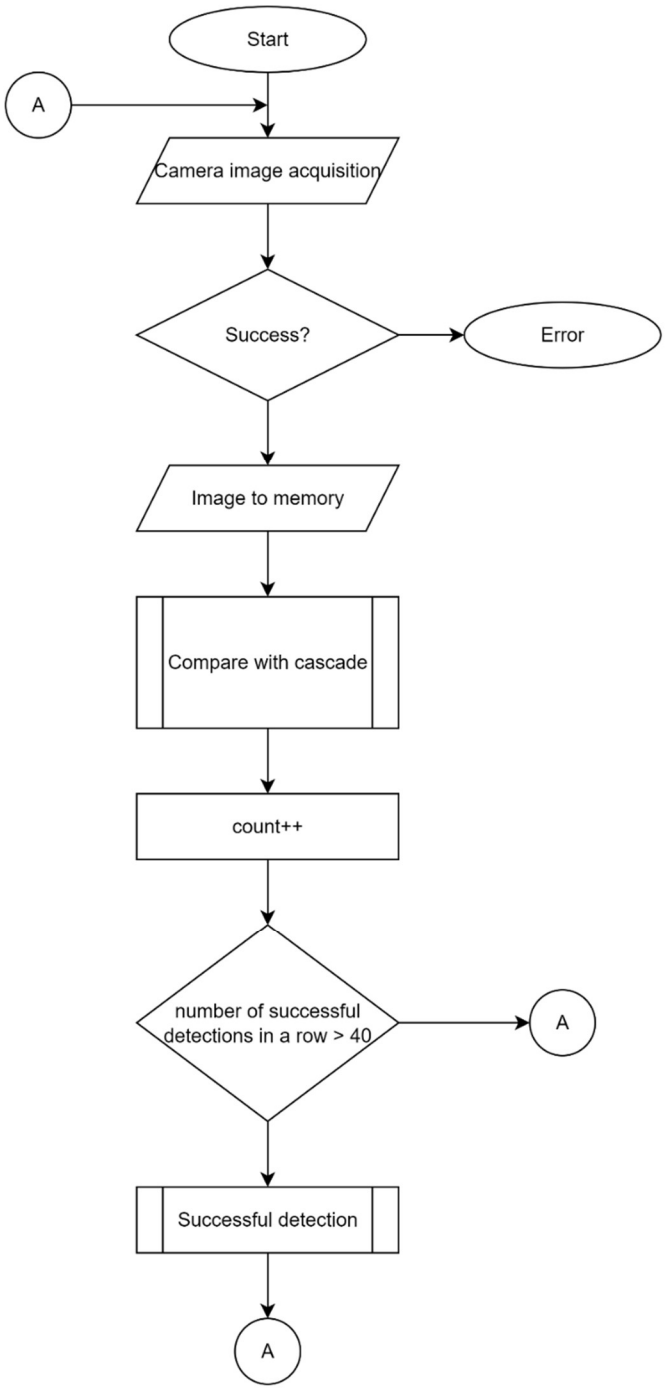

Fig. 4. Flowchart diagram of the main part of the code

In Fig. 5, all four detected tags are shown. The images also show the subsequent reaction, meaning that if a speed limit sign is detected, the robot will brake (the rear brake lights can be seen) etc.

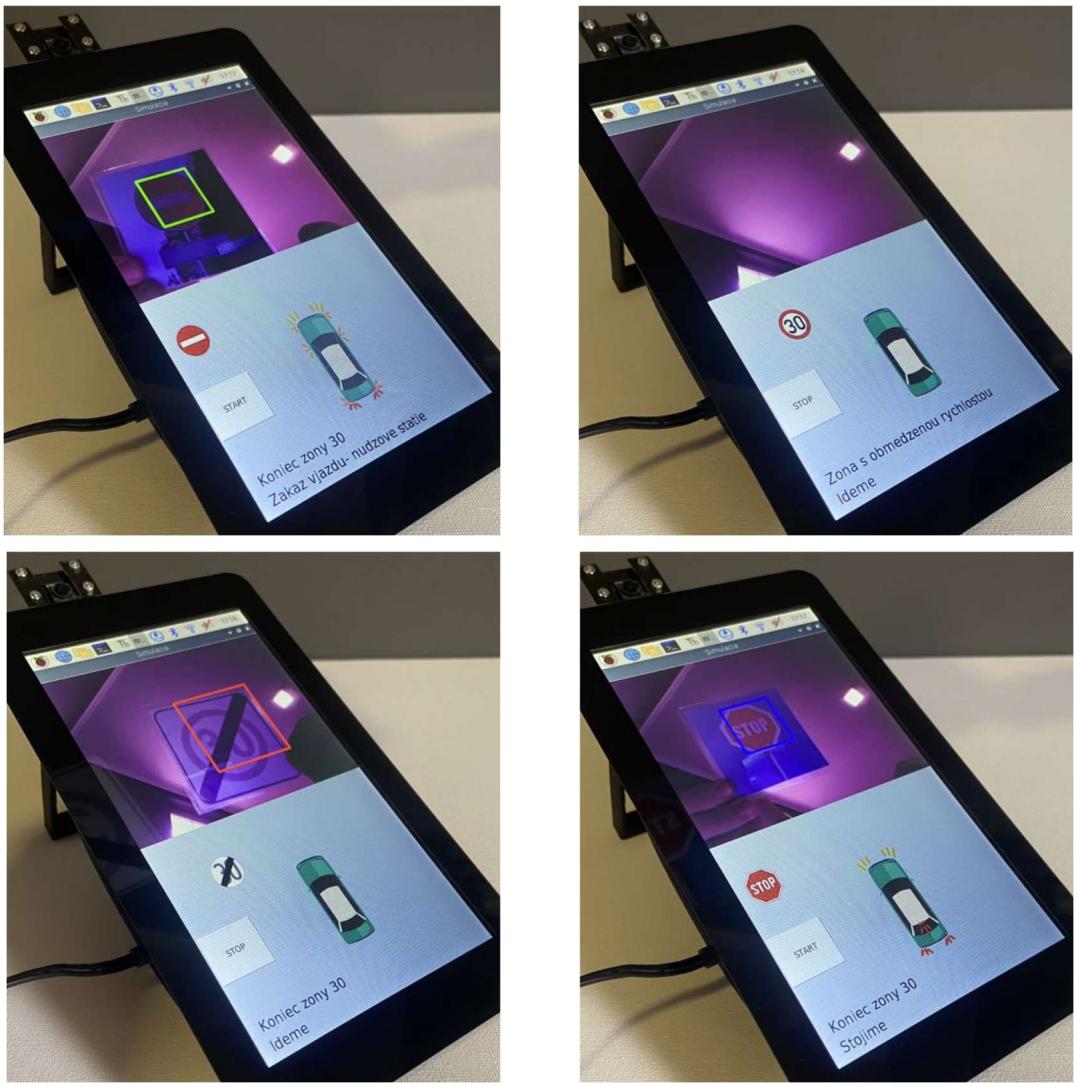

Fig. 5. Tags detection and application reaction

Conclusions. The main goal of the article, which was to detect custom objects, was successfully achieved, and its functionality was proven with the application that was created. The next objective, which was beyond the scope of this thesis but would be a future requirement, is to install this program into a service robot, create control using UART communication, receive and send signals via GPIO pins, and build a functional service robot model.

When creating a custom database, various problems can arise, such as a lack of sample images, incorrectly set resolution, and an unsuitable aspect ratio. To create a proper cascade database, you need to follow certain rules. Within the cascade creation application, it's necessary to set the constants appropriately so that the creation process isn't overly time-consuming and inefficient. This thesis shows the exact procedure that proved to be the best alternative after long processes of testing and fine-tuning.

The practical part also includes an introduction to the application, its appearance after switching it on, and its functioning, so that the reader can form their own opinion about what is behind the creation of such a functional result.

Acknowledgements

This article was created thanks to the VEGA project support: 1/0294/24 Research and development of multi-robotic system with distributed intelligence in the cloud.

References

1. Jurko, R: (2023), Implementácia počítačového videnia do riadenia mobilného robota. Technická Univerzita v Košiciach, Strojnícka fakulta, Diploma work, 2023.

2. EuroparlTV. (2020, August 27). Artificial Intelligence: Definition and Use. European Parliament [cit.2023-01-04]. Dostupné na internete: https://www.europarl.europa.eu/news/sk/headlines/society/20200827STO85804/ umela-inteligencia-definicia-a-vyuzitie?at_campaign=20234-Digital&at_medium=Google_Ads&at_platform=Search&at_creation=DSA&at_goal=TR_G&at_audience=&at_topic=Artificial_Intelligence&gclid=CjwKCAjwl6OiBhA2EiwAuUwWZVi_CbL5gJipMIZ0sk5FYe-7qRc86VevFwpg07T7j-nZfwICONLPOhoCm2wQAvD_BwE.

3. Raspberry Pi. (2023). *Raspberry Pi Software* [Computer software]. Retrieved January 10, 2023, from https://www.raspberrypi.com/software.

4. Gendzo. (2018): Inštalácia a prvé spustenie Raspberry Pi. [online], [cit.2023-02-06]. Dostupné na internete: https://www.gendzo.sk/navody/instalacia-a-prve-spustenie-raspberry-pi/ [28. apríla 2023].

5. MyGreatLearning. (2022). OpenCV tutorial in Python. [online]. Dostupné na internete: https://www.mygreatlearning.com/blog/opencv-tutorial-in-python/.

6. AHMADI, A.: Cascade Trainer GUI. [online]. Dostupné na internete: https://amin-ahmadi.com/cascade-trainer-gui/

7. Hajduk, M., Sukop, M., & Haun, M. (2019). *Cognitive multi-agent systems: Structures, strategies and applications to mobile robotics and robosoccer.* Springer Nature. https://doi.org/10.1007/978-3-319-93685-7.

Отримано 29.08.2025

УДК 621.8

Марек Сукоп[1], Растислав Юрко[2]

[1]професор, професор кафедри виробничих систем і робототехніки
Технічний університет Кошице (Кошице, Словакія)
E-mail: marek.sukop@tuke.sk. **ORCID:** https://orcid.org/0000-0001-7987-3557
ResearcherID: AAH-5495-2019. **Scopus Author ID:** 36615762200

[2]студент інженерного факультету кафедри виробничих систем і робототехніки
Кошицький технічний університет (Кошице, Словакія)
E-mail: rastislav.jurko@student.tuke.sk

ПЕРЕВІРКА ГЕОМЕТРИЧНИХ ХАРАКТЕРИСТИК РОБОТА ЗА ДОПОМОГОЮ ЛАЗЕРНОГО ІНТЕРФЕРОМЕТРА

У статті розглядається розробка прикладу докладання, демонструє практичне застосування комп'ютерного зору у сфері мобільної робототехніки. В основі проєкту лежить доступна та економічна апаратна платформа, що складається з Raspberry Pi, зовнішньої камери та сенсорного дисплея. Основна функція програми – розпізнавання та обробка чотирьох різних типів дорожніх знаків. Основною метою було довести можливість створення щодо простого та недорогого рішення для обробки зображень, що безпосередньо застосовується до проєктів мобільної робототехніки. Важливою мотивацією даної роботи стало створення педагогічного стенда, призначеного для тестування та вивчення подібних алгоритмів. Цей стенд є базовим освітнім інструментом, надаючи студентам практичну платформу для розуміння фундаментальних принципів обробки зображень та базових алгоритмів штучного інтелекту. Отриманий додаток і методологія, що лежить в його основі, будуть використовуватися для того, щоб допомогти студентам освоїти ці складні теми в практичному, реальному контексті. У статті докладно описані алгоритми та покроковий процес їх реалізації на апаратному рівні. У статті коротко описаний алгоритм розпізнавання дорожніх знаків, наочно проілюстрований блок-схемою для кращого розуміння. Програмне забезпечення розроблене на Python для платформ Raspberry Pi з використанням потужних бібліотек, таких як Numpy і OpenCV, які необхідні для виконання точних обчислень і широкого спектра завдань обробки зображень. Крім того, у статті описується процес створення бази даних зображень та їх підготовки, які використовуються як позитивні або негативні збіги для алгоритму розпізнавання. У разі позитивних збігів додаток навчався розпізнавати конкретні знаки, включаючи знак «Стоп», знак «В'їзд заборонений», знак «Максимальна швидкість 30 км/год» та знак «Кінець обмеження швидкості». У заключних розділах статті обговорюються досягнуті результати, підтверджується успішність проєкту в досягненні початкових цілей та пропонуються цінні рекомендації для майбутніх досліджень, спрямованих на розширення поточної роботи. Зокрема, пропонуються заходи щодо підвищення продуктивності алгоритму та розширення різноманітності розпізнаваних системою об'єктів.

Ключові слова: *мобільний робот; обробка зображень; каскад; Raspberry Pi.*
Рис.: 5. Бібл.: 7.

Sukop M., Jurko R. (2025) Implementing computer vision for a mobile robot. *Technical sciences and technologies,* (3(41)), 40-45. DOI: https://doi.org/10.25140/2411-5363-2025-3(41)-40-45.