DOI: https://doi.org/10.36910/6775-2524-0560-2024-57-03
УДК 004.415.3
Pekh Petro, PhD
https://orcid.org/0000-0002-6327-3319
Hrygorychenko Vladyslav, master of science
Lutsk National Technical University, Lutsk, Ukraine

DEVELOPMENT OF THE SYSTEM FOR RECOGNITION OF CAR LICENSE PLATE USING ARTIFICIAL INTELLIGENCE

Pekh P., Hrygorychenko V. Development of the system for recognition of car license plate using artificial intelligence. The article proposes a technology for creating a license plate recognition system using artificial intelligence. The license plate recognition program is developed in the PyCharm integrated development environment (IDE), which supports Python and provides access to libraries for machine learning and image processing. Libraries for working with images and text, such as OpenCV, imutils, pytesseract, and others, are installed through the PyCharm terminal. For the processing and recognition of license plates, a set of images containing various car license plates was prepared. These images are made from different angles, with different lighting and quality, which is caused by the need to ensure the appropriate response of the program to different input data to ensure its correct operation. After loading the image using the OpenCV library, it is converted to grayscale, since contour detection and image processing are much easier in grayscale. The Canny algorithm was used to detect contours. Contours in the image are used for further identification of the license plate. Determination of horizontal and vertical contours was performed using the Sobel operator. We use Tesseract OCR to recognize license plate text. After the license plate has been recognized, we compare it with those in the database. After setting up the program, it is necessary to test it with different images to make sure the accuracy of license plate recognition. This allows you to detect possible problems in the recognition process, such as low contrast of the image, contamination of license plates or unstable lighting, tilting or distortion of license plates in the image, which can affect the accuracy of recognition.
Keywords: License plate recognition, OpenCV, Tesseract OCR, Easy OCR.

Пех П.А., Григориченко В.Ю. Розробка системи розпізнавання номерних знаків засобами штучного інтелекту. У статті запропоновано технологію створення системи розпізнавання номерних знаків засобами штучного інтелекту. Програма розпізнавання номерних знаків розроблена в інтегрованому середовищі розробки (IDE) PyCharm, яке підтримує Python та забезпечує доступ до бібліотек для машинного навчання та обробки зображень. Бібліотеки для роботи з зображеннями та текстом, такі як OpenCV, imutils, pytesseract, та інші, встановлюються через термінал PyCharm. Для обробки та розпізнавання номерних знаків був підготовлений набір зображень, які власне містять різні автомобільні номерні знаки. Ці зображення виконані з різних ракурсів, з різною освітленістю та якістю, що викликане необхідністю забезпечення відповідного реагування програми на різні вхідні дані для забезпечення її коректної роботи. Після завантаження зображення за допомогою бібліотеки OpenCV воно перетворюється у відтінки сірого, оскільки детекція контурів та обробка зображення значно простіша в градаціях сірого. Для виявлення контурів застосовувався алгоритм Canny. Контури на зображенні використовуються для подальшого виявлення номерного знака. Визначення горизонтальних та вертикальних контурів виконувалися за допомогою оператора Собеля. Для розпізнавання тексту номерного знака застосовуємо Tesseract OCR. Після того, як номерний знак був розпізнаний, порівнюємо його з наявними у базі даних. Після завершення налаштування програми необхідно провести її тестування з різними зображеннями, щоб переконатись у точності розпізнавання номерних знаків. Це дозволяє виявити можливі проблеми у процесі розпізнавання, такі як низька контрастність зображення, забруднення номерних знаків чи нестабільне освітлення, нахил або спотворення номерних знаків на зображенні, що може впливати на точність розпізнавання.
Ключові слова: Розпізнавання номерних знаків, OpenCV, Tesseract OCR, Easy OCR

The problem statement. Automated license plate recognition (ALPR) systems have become widely used in the areas of road safety, speed control, parking and other types of traffic monitoring. In the conditions of the growing number of cars on the roads and the need to improve the efficiency of law enforcement agencies, the issue of developing and improving such systems becomes urgent. In particular, the application of machine learning methods allows to significantly improve the accuracy of number recognition in difficult conditions, such as poor lighting, fast movement or different angles of the camera [1,2,3,5].

In addition to the technical benefits, ALPR systems have a significant social impact, helping to reduce traffic violations, improve pedestrian and driver safety, and improve the overall transportation infrastructure. Thanks to the integration with other digital systems, such as vehicle registration databases or access control systems, they become an integral part of modern smart cities [4,6].

The research purpose formulation. The purpose of the work is the development and research of automated car license plate recognition systems based on machine learning methods, with an emphasis on increasing its efficiency and accuracy in real operating conditions.

The latest research and publications analysis. Automated license plate recognition systems occupy an important place in the field of intelligent transport systems. They are used in many industries, such as

traffic control, parking lot security, parking violation monitoring, toll road and border automation, and law enforcement. These systems make it possible to significantly reduce the time and resources needed to perform these tasks, thanks to a high level of automation. That is why many scientists pay attention to the development and research of these systems. [3,5,7]

The main research material presentation, Let's consider the development of the license plate recognition algorithm The virtual environment is an important component for the development and testing of modern software systems, in particular for license plate recognition and computer vision tasks. In the organization of such an environment, a special role is played by the selection of tools and libraries, as well as software settings to achieve maximum results when implementing algorithms. Here are some key aspects of setting up a virtual environment for license plate recognition tasks.

Choice of operating system and software. The first thing to do when organizing a virtual license plate recognition environment is to determine the operating system and set of tools that will be used. The most popular operating systems for such tasks are Linux and Windows, as they support most of the necessary libraries and tools.

In most cases, the virtual development environment is configured using software tools such as Python and OpenCV.

Python is a versatile tool for developing and testing machine learning systems thanks to numerous libraries for working with computer vision and image processing.

OpenCV is a library for working with images and videos, which includes a wide set of functions for image processing, such as filtering, contour detection, object selection, text recognition, etc.

Tesseract OCR is an optical text recognition that allows you to extract text from images (for example, license plates from cars).

NumPy and *Matplotlib* are libraries for scientific computing and data visualization that are widely used in image processing tasks.

Imutils is a utility library designed to make working with OpenCV easier. It contains functions for quickly performing common operations such as resizing, rotating, automatically changing the orientation of images or removing objects in a certain area of the frame. This library makes the code more compact and easy to read.

EasyOCR is a modern library for text recognition based on deep learning. EasyOCR has built-in neural network models that allow you to recognize text even on complex images with low quality, complex backgrounds or text written in different languages. Due to its simplicity and power, this library is an excellent choice for systems requiring high recognition accuracy.

To develop the program, we use the PyCharm integrated development environment (IDE). This environment supports Python, provides auto-completion tools, built-in debuggers, and makes it easy to work with libraries for machine learning and image processing. To install PyCharm, we download it from the official website using the PyCharm Download link. After installation, open PyCharm and create a new project, selecting the required version of Python.

Used libraries for working with images and text, such as OpenCV, imutils, pytesseract, and others, can be installed through the PyCharm terminal. To do this, you need to execute the following commands.

To install the OpenCV library (Fig. 1) we run the command

```
pip install opencv-python.
```

For additional utilities that make working with OpenCV easier (imutils) we run the command

```
pip install imutils.
```

To configure optical text recognition (Tesseract OCR) via the pytesseract library, we use this command

```
pip install pytesseract.
```

OpenCV is the main library for image processing, which includes functions for filtering, contour detection, object selection, etc.;

imutils – auxiliary library for convenient scaling and rotation of images;

pytesseract – interface to Tesseract OCR for text recognition on images;

numpy is a library for working with arrays, often used to process image pixels;

winsound - a library for playing sounds in the Windows environment, used to inform about the found license plate.

To use Tesseract OCR, you must also download and install it from the official repository on GitHub, and then specify the path to the executable file in the code:

`pytesseract.pytesseract.tesseract_cmd = r"C:\Program Files\Tesseract-OCR\tesseract.exe".`

For license plate processing and recognition, it is necessary to prepare a set of images containing car license plates. Images can be from different angles, with different lighting and quality, which requires appropriate settings for optimal operation of contour detection and text recognition algorithms.

It is recommended to save images in a separate CarPictures/ folder to simplify file access (Fig. 1).

Figure 1 – Folder CarPictures

To compare recognized license plates with registered ones, you can create a text file that will contain a list of license plates. This file is stored in a separate folder, for example Database/Database.txt.

The first step is to load the image using the OpenCV library (Fig. 2). For this, we use the function cv2.imread()

`image = cv2.imread('CarPictures/10.jpg').`

Images can be reduced for ease of processing:

`image = imutils.resize(image, width=500).`

For more convenient processing, the image is converted to shades of gray, since contour detection and image processing are much simpler in grayscale (Fig. 3):

`gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY).`

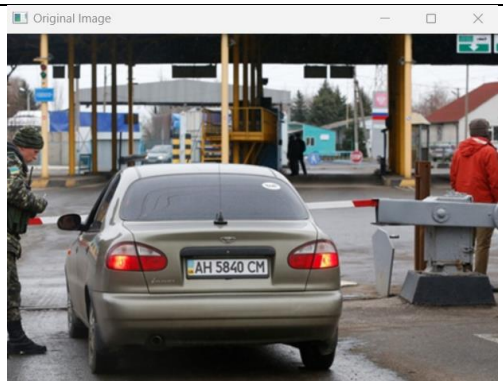

Figure 2 – Original Image

Figure 3 – Gray Scale Image

Contour detection using Kenny's method. To detect contours, we use the Canny edge detection algorithm. This method uses threshold values to extract strong gradients in the image corresponding to contours, (Fig. 4):

`canny_edges = cv2.Canny(gray, 170, 200.`

Contours in the image are used for further identification of the license plate.

Determination of horizontal and vertical contours using the Sobel operator.

The Sobel operator is used to select horizontal and vertical image gradients. It helps detect lines and contours (Fig. 5):

```
sobel_x = cv2.Sobel(gray, cv2.CV_64F, 1, 0, ksize=3);
sobel_y = cv2.Sobel(gray, cv2.CV_64F, 0, 1, ksize=3);
sobel_combined = cv2.magnitude(sobel_x, sobel_y).
```

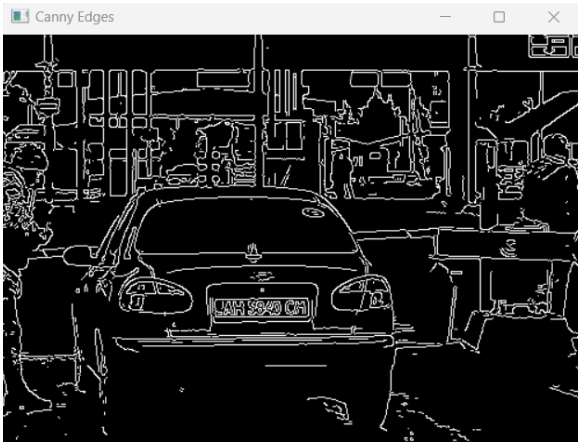

Figure 4 – Canny Edges

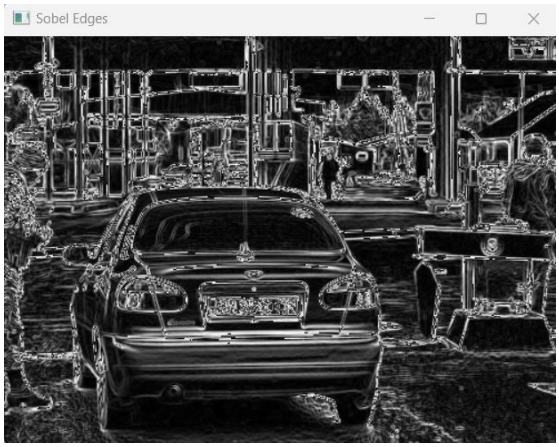

Figure 5 – Sobel Edges

Searching for a probable license plate. Using the cv2.findContours() method, contours on the image are detected (Fig. 6):

```
cnts, _ = cv2.findContours(canny_edges.copy(), cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE).
```

In order to find a probable license plate, the shape of the contours is checked. A license plate usually has a rectangular shape, so we are looking for a contour with four sides (Fig. 7):

```
approx = cv2.approxPolyDP(i, 0.02 * perimeter, True);
if len(approx) == 4.
```

If the contour found meets the requirements, the license plate image is cropped and saved:

```
cv2.imwrite('plate.png', crp_img).
```

Text recognition using OCR. To recognize text from a license plate, we use Tesseract OCR. We use the pytesseract.image_to_string() function, which extracts text from a cropped image of a license plate:

```
text = pytesseract.image_to_string(crop_img_loc, lang='eng').
```

Checking the number in the database. After the license plate has been recognized, we compare it with the data in the database. To do this, we use the function that searches for a license plate in a text file:

```
def check_if_string_in_file(file_name, string_to_search);
with open(file_name, 'r') as file;
for line in file;
if string_to_search in line;
return True;
return False.
```

If the number plate is found in the database, the system emits a signal (for example, via winsound.Beep):

```
winsound.Beep(frequency, duration).
```

After the system setup is complete, it is necessary to test with different images to ensure the accuracy of license plate recognition. This will reveal possible problems such as low contrast, dirty license plates or unstable lighting. Depending on the test results, it may be necessary to adjust the contour detection parameters or OCR settings to improve the performance of the system in different conditions. It is also important to consider factors such as the tilt or distortion of license plates in the image, as this can affect the accuracy of the recognition.

This step ensures the integration of all system components, allowing automatic detection of license plates in images and checking them against the database.

Here's a step-by-step description of how to build an EasyOCR-based system that performs text-to-image recognition using OpenCV and Matplotlib. Our code already includes the main steps, and we will add a scientific description of each step.

Before starting work, we need to install all the necessary libraries for working with images, such as OpenCV, EasyOCR, NumPy and Matplotlib (Fig. 8).

Figure 6 – Final Image with Detected Plate

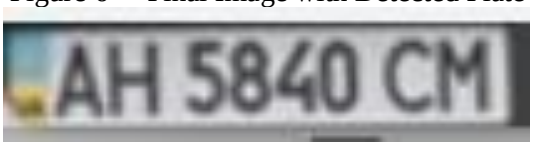

Figure 7 – Final Image

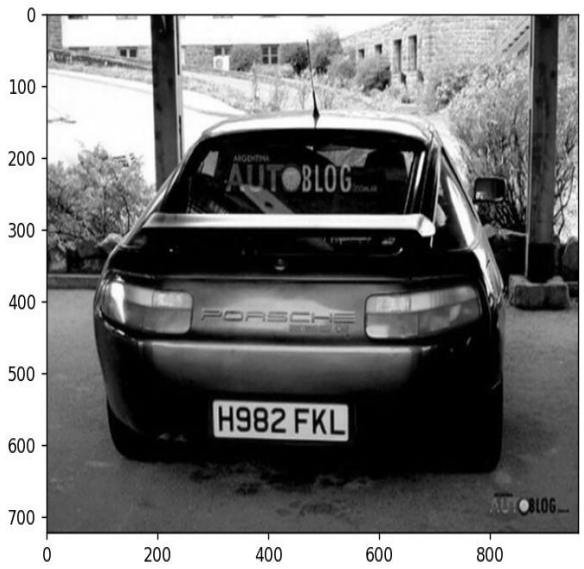

Figure 8 – Conversion to grayscale

```
pip install opencv-python easyocr matplotlib numpy imutils.
```

The image is loaded using the OpenCV library:

```
img = cv2.imread(r'F:\mag\image4.jpg').
```

To reduce the complexity of the image, before processing, we apply the conversion to grayscale:

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY).
```

Applying a filter to reduce noise: To improve the quality of the image, we apply a two-sided filter.

```
bfilter = cv2.bilateralFilter(gray, 11, 17, 17).
```

Edge detection using Canny: We use an edge detector to detect objects in the image.

```
edged = cv2.Canny(bfilter, 30, 200).
```

Search for contours: Using cv2.findContours, we search for contours in the image.

```
keypoints = cv2.findContours(edged.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE).
```

Definition of rectangular contours: Define contours that resemble the shape of a license plate (four corners).

for contour in contours;

```
approx = cv2.approxPolyDP(contour, 10, True);
if len(approx) == 4;
location = approx;
break.
```

Highlighting the license plate: Create a mask to highlight the license plate on the image (Fig. 9.).

```
mask = np.zeros(gray.shape, np.uint8);
new_image = cv2.drawContours(mask, [location], 0.255, -1);
new_image = cv2.bitwise_and(img, img, mask=mask).
```

Image cropping for recognition: Crop the area of the license plate for further text recognition (Fig. 10).

```
(x, y) = np.where(mask == 255);
cropped_image = gray[x1:x2+1, y1:y2+1].
```

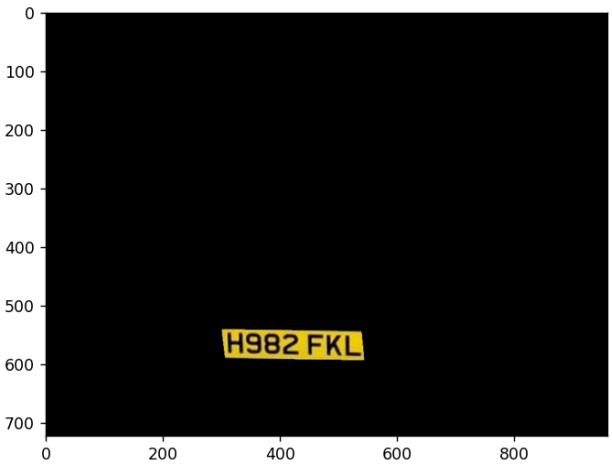

Figure 9 - Selection of license plate

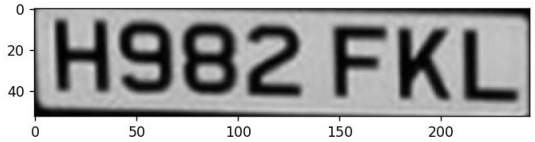

Figure 10 - Cropping of a license plate

Text recognition using EasyOCR. We use the EasyOCR library to recognize text from a cropped image:

```
reader = easyocr.Reader(['en']);
result = reader.readtext(cropped_image).
```

Result display. We display the result on the image by drawing a rectangle and adding the recognized text:

```
font = cv2.FONT_HERSHEY_SIMPLEX;
res = cv2.putText(img, text=text, org=(approx[0][0][0], approx[1][0][1] + 60), fontFace=font, fontScale=1, color= (0, 255, 0), thickness=2).
```

These steps enable license plate recognition from images using EasyOCR and OpenCV for preprocessing.

Conclusion and prospects for further research. The developed model for recognizing car license plates takes into account the specifics of image processing in real-world environments, such as changing lighting, sign pollution, and a variety of shooting angles.

The review of modern approaches to license plate recognition revealed the main trends in the use of machine learning methods, in particular convolutional neural networks (CNN), which demonstrate high efficiency in computer vision tasks.

References

1. Face, H. 2020. Trocr. https://huggingface.co/docs/ transformers/en/model_doc/trocr (дата звернення 14.09.2024).
2. Buzzelli, M. and Segantin, L. « Revisiting the compcars dataset for hierarchical car classification: New annotations, experiments, and results» 2021.
3. Cherti, M., Beaumont, R., Wightman, R., Wortsman, M., Ilharco, G., Gordon, C., Schuhmann, C., Schmidt, L., and Jitsev, J. «Reproducible scaling laws for contrastive language-image learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition» 2023.
4. Dosovitskiy, A. «An image is worth 16x16 words: Transformers for image recognition at scale» 2020.
5. Henry, C., Ahn, S. Y., and Lee, S.-W. «Multinational license plate recognition using generalized character sequence detection» 2020.
6. Hu, M., Bai, L., Fan, J., Zhao, S., and Chen, E. «Vehicle color recognition based on smooth modulation neural network with multi-scale feature fusion. Frontiers of Computer Science» 2020.
7. Kemertas, M., Pishdad, L., Derpanis, K. G., and Fazly, A. «Rankmi: A mutual information maximizing ranking loss. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition» 2020.