Report

Advanced Programming

Kaliaskaruly Daneker IT-2201, Orynbasarova Akerke IT-2207,

Marat Aizere IT-2207

1. Introduction

1.1. Problem

Number Plate Recognition (NPR), known by different titles, utilizes optical character recognition on images to decipher vehicle registration plates. Our aim is to develop a web application incorporating NPR through Python. The main goal is to extract data from license plates, facilitating the generation of location-based vehicle information. This technology serves purposes like identifying violations such as speeding, red light running, improper parking, crime prevention, and enhancing transportation management.

In our project, we endeavor to craft a dynamic web application harnessing the power of Python to seamlessly integrate NPR functionalities. The overarching objective revolves around the extraction of pertinent data from license plates, thereby facilitating the generation of comprehensive, location-based vehicle information.

This technology transcends mere novelty, finding invaluable utility across a spectrum of applications. Firstly, it serves as a linchpin in the identification and documentation of traffic violations such as speeding, red light running, and improper parking. By automating the process of plate recognition, it not only expedites law enforcement efforts but also bolsters the efficacy of traffic management systems.

1.2. Literature review

a) The proposed system focuses on optimizing and improving the efficiency of ANPR. Initially the system was trained with the YOLO (You Only Look Once) object detection algorithm. The system employs the ImageAI framework, which proves effective for both detection and recognition.

The system is trained using NVIDIA Jetson Nano for car and number plate recognition, with software built using Python. It's designed to assist in number plate recognition, utilizing cameras connected to a central system. The process involves vehicle detection, number plate localization, character segmentation, and Optical Character Recognition (OCR).

For training, datasets are utilized, such as the Stanford car dataset for vehicle detection and a collection of number plate images for plate localization. The system undergoes rigorous training and validation processes, optimizing neural network parameters for improved performance.The results show significant enhancements in ANPR efficiency compared to existing systems, especially in dynamic environments. The proposed system successfully detects vehicles, localizes number plates, and accurately recognizes characters,

leading to better traffic management and enforcement.
a-review-paper-on-automatic-number-plate-IJERTV8IS12039820200107-64305-1nwjir3-libre.pdf (d1wqtxts1xzle7.cloudfront.net)

b) The project works by first preprocessing the input image, converting it to grayscale, and then adapting it into a binary image using Otsu's method. This step helps in localizing the number plate. Then, in the character segmentation phase, the characters on the number plate are identified and segmented into individual images. Next, in the character recognition phase, supervised machine learning techniques are used to recognize the segmented characters. The recognized characters are then stored in a database. Training and evaluation of the model, which could involve algorithms such as: Support Vector Machines (SVM), Random Forest, Convolutional Neural Networks (CNN) for deep learning-based approaches. Finally, a web application is developed to visualize the dataset created from the recognized characters using JavaScript libraries like D3.js and DC.js, allowing for interactive data analysis and visualization.
IEEE Xplore Full-Text PDF:

1.3. Current work

Our team has created and developed two applications of ANPR. The first version is a web-based application leveraging Python libraries such as OpenCV and EasyOCR. It extracts text from images containing license plates, facilitating the generation of location-based vehicle information. The second version, a desktop application, showcases the culmination of our efforts, employing similar technologies but with enhanced functionality and performance. Key features include real-time detection of license plates in video streams, leveraging image processing techniques and optical character recognition algorithms. The applications serve diverse purposes, including traffic regulation, law enforcement, crime prevention, and transportation management. The development process involved iterative phases, including requirement analysis, design, implementation, testing, and deployment. Python served as the primary programming language, chosen for its versatility and extensive libraries for image processing and OCR tasks. The Tkinter GUI toolkit was employed to create an intuitive user interface for seamless interaction. The report details the methodologies employed, challenges encountered, and potential avenues for future improvement.

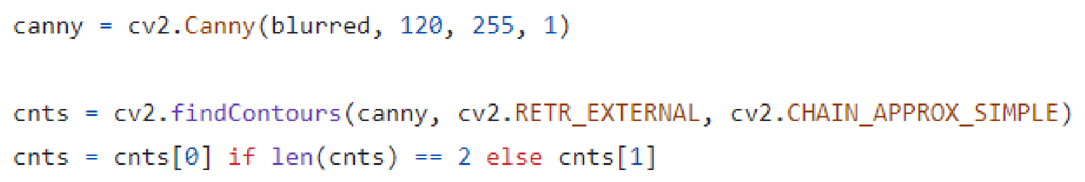
2. Data and Methods

2.1. Web App

Our team relied on the Python programming language as the foundation of our application. Python's flexibility and extensive libraries made it ideal for implementing various tasks, including image processing algorithms, developing the user interface, and integrating OCR functionalities.

To handle image and video processing tasks, we turned to OpenCV, a powerful library known for its capabilities in computer vision. OpenCV allows us to capture frames efficiently, perform operations such as grayscale conversion and contour detection, and recognize objects in images and video streams. These features were crucial for identifying and extracting license plate regions accurately.

Canny Edge Detection was used for identifying edges in the image, which is crucial for detecting object boundaries or shapes. Contours detection algorithm identifies contours in the canny edges image, which aids in detecting shapes like car plates.



```
canny = cv2.Canny(blurred, 120, 255, 1)

cnts = cv2.findContours(canny, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
cnts = cnts[0] if len(cnts) == 2 else cnts[1]
```

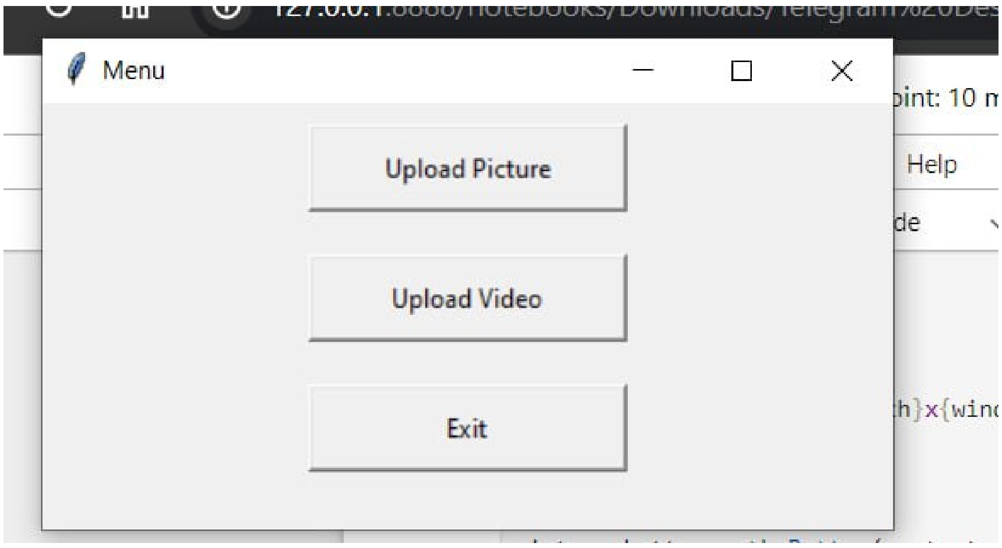
The Tesseract OCR engine is one of the models we used that has been pretrained, and it is widely used for text recognition tasks. Tesseract is an open-source OCR engine developed by Google and is known for its accuracy in extracting text from images. The training on various datasets has enabled it to recognize alphanumeric characters, which makes it suitable for our application's needs.

Furthermore, we made use of EasyOCR, which uses pre-trained models to recognize text in images. EasyOCR is built on top of the PyTorch framework and integrates multiple OCR models trained on diverse datasets, including languages and fonts. This enables EasyOCR to provide robust and accurate text extraction capabilities across different scenarios.

Our methodology includes two key methods: 'Extract Texts from Directory' and 'Detect Car Plates'. The first iteration iterates through image files in a specified directory, using EasyOCR to extract text from each image and filtering out text strings based on their length. The latter processes video files to detect and extract license plate information, utilizing OpenCV for frame manipulation, image processing techniques, and Pytesseract for OCR. FastAPI is imported and initialized to create the web application, providing the backend infrastructure for handling HTTP requests and responses. In addition to FastAPI on the backend, Angular was utilized for the frontend development.

2.2. Desktop App

The desktop application utilizes Tkinter, a standard GUI toolkit in Python. For image processing we have employed OpenCV. Tesseract is an open source tool for optical character recognition, which was developed by Google. Pytesseract is a very useful Python library that serves as a bridge to the Tesseract engine. Also, this version uses pretrained models for text recognition Pytesseract and EasyOCR. So, as a result users will be able to see the outcomes of these powerful tools. In the picture below we can see the main menu of the app. It has 3 buttons for uploading pictures or video and to exit.



The work flow can be described as follows:

1. Upload image or video. When the user clicks on the button “Upload image” or “Upload Video”, the program opens dialog, which allows the user to select image or video. Subsequently, when the image is selected, the process of ANPR starts. When video is uploaded, the application systematically processes the video frame by frame.
2. ANPR processing. This stage involves multiple subprocesses like contour detection, region cropping etc. The first thing that happens when an image is uploaded is that it gets converted to grayscale in order to make processing easier later on. Next, a Gaussian blur is used to reduce noise, which improves the quality of the processes that follow.

   For image:

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
blurred = cv2.GaussianBlur(gray, (5, 5), 0)
```

For video:

```
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
blurred = cv2.GaussianBlur(gray, (3, 3), 0)
canny = cv2.Canny(blurred, 120, 255, 1)
```

After that the process of contour detection only for video starts.

```
cnts = cv2.findContours(canny, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
cnts = cnts[0] if len(cnts) == 2 else cnts[1]
```

Subsequently, Pytesseract and EasyOCR is utilized to extract text from input.

For image:

```
reader = easyocr.Reader(['en'])
result_easyocr = reader.readtext(blurred)

result_pytesseract = pytesseract.image_to_string(blurred, lang='eng')
```

For video:

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

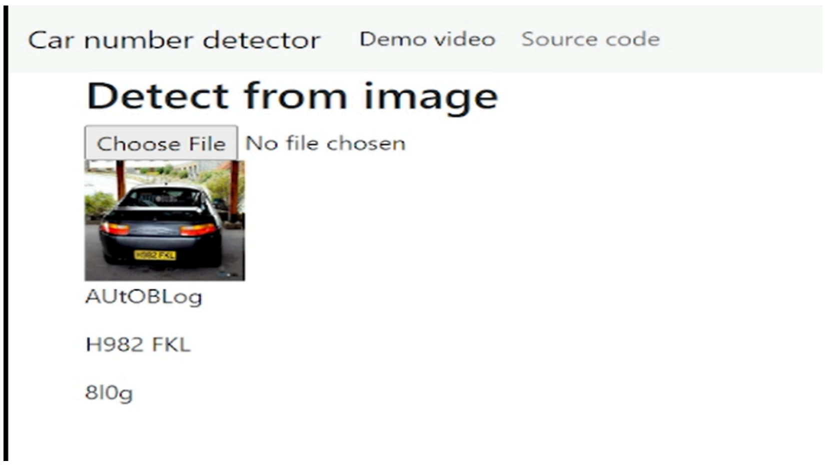
3. Displaying results. For this part, we are creating Tkinter windows to display the detected text.

```
tk.Label(result_window, text="Text Detected (EasyOCR):", font=("Arial", 12, "bold")).pack()
for detection in result_easyocr:
    tk.Label(result_window, text=detection[1]).pack()

tk.Label(result_window, text="Text Detected (Pytesseract):", font=("Arial", 12, "bold")).pack()
tk.Label(result_window, text=result_pytesseract).pack()
```

3. Results

Web App outcomes:


Car number detector   Demo video   Source code

Detect from image

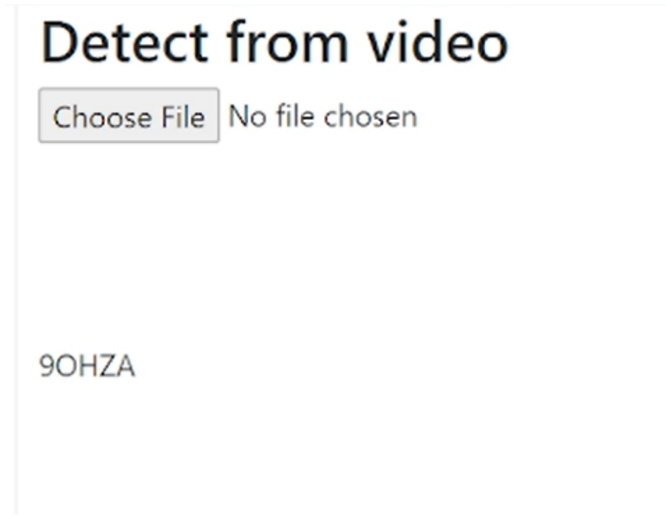
Choose File No file chosen

AUtOBLog

H982 FKL

8l0g

Our application instantly recognizes the car plate number from the image and reproduces it as text.

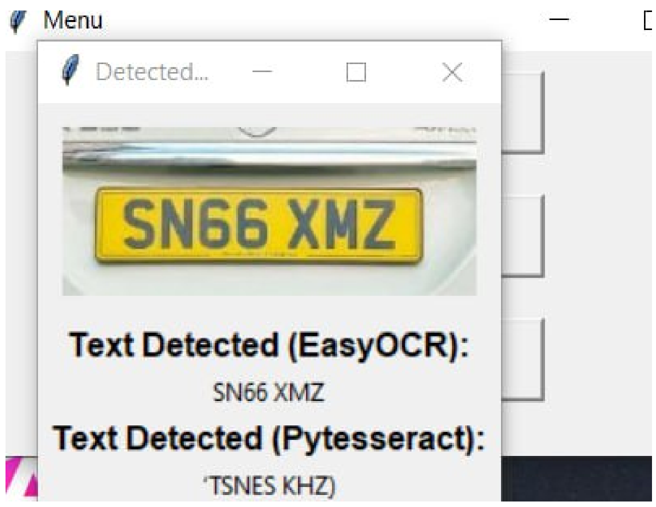


9OHZA

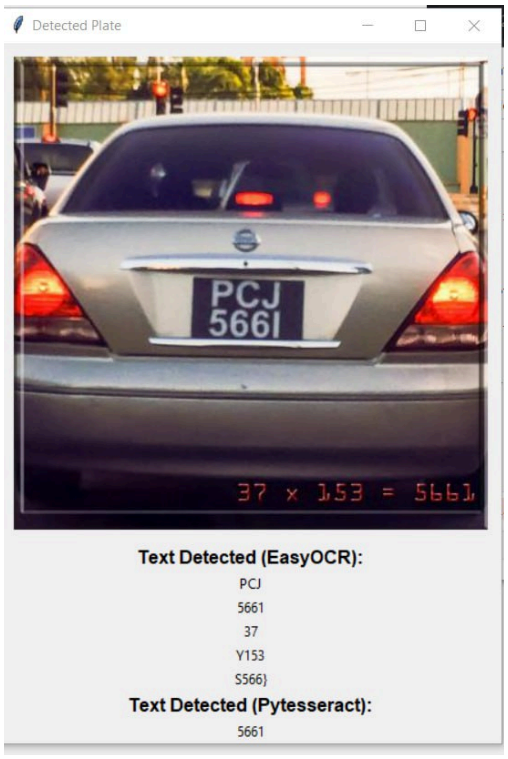
Although detecting car plate numbers from video may take longer, particularly for videos with high quality, the application still has the ability to identify the numbers. The data processing and analysis process may be more time-consuming because of the large amount of data to process and analyze in each frame of the video.

Desktop App outcomes:

Findings from processing previously cropped images are more accurate than findings from studying pictures of the complete car. The precision of optical character recognition (OCR) algorithms is improved by cropped photographs, which provide a focused view of the number plate area. On the other hand, examining whole car images adds complexity and makes it harder to accurately identify features like lighting fluctuations and unnecessary parts. Thus, using photos that have been cropped guarantees more consistent results.

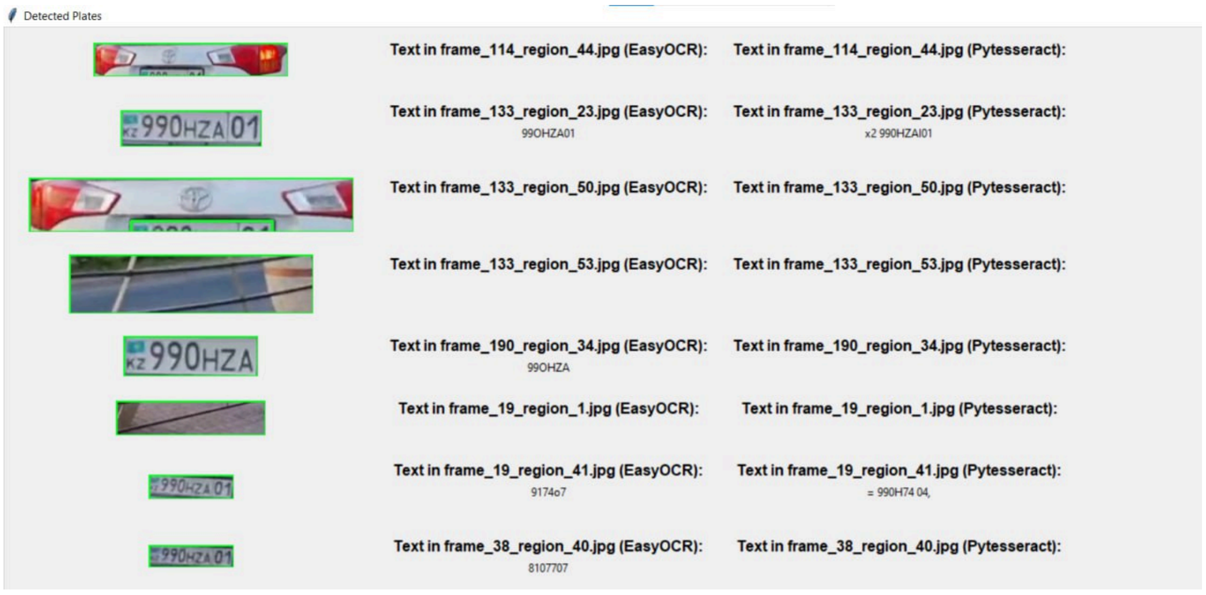


Text Detected (EasyOCR):
SN66 XMZ
Text Detected (Pytesseract):
'TSNES KHZ)



```
Text Detected (EasyOCR):
PCJ
5661
37
Y153
S566}
Text Detected (Pytesseract):
5661
```

Each frame of the incoming video is processed separately by the application in order to recognize license plates from video. To do this, each frame in the video sequence must be examined and probable number plate information extracted. So, we get information from every frame.



```
Text in frame_114_region_44.jpg (EasyOCR):      Text in frame_114_region_44.jpg (Pytesseract):

Text in frame_133_region_23.jpg (EasyOCR):      Text in frame_133_region_23.jpg (Pytesseract):
              99OHZA01                                        x2 990HZAI01

Text in frame_133_region_50.jpg (EasyOCR):      Text in frame_133_region_50.jpg (Pytesseract):

Text in frame_133_region_53.jpg (EasyOCR):      Text in frame_133_region_53.jpg (Pytesseract):

Text in frame_190_region_34.jpg (EasyOCR):      Text in frame_190_region_34.jpg (Pytesseract):
               99OHZA

Text in frame_19_region_1.jpg (EasyOCR):        Text in frame_19_region_1.jpg (Pytesseract):

Text in frame_19_region_41.jpg (EasyOCR):       Text in frame_19_region_41.jpg (Pytesseract):
               9174o7                                          = 990H74 04,

Text in frame_38_region_40.jpg (EasyOCR):       Text in frame_38_region_40.jpg (Pytesseract):
               8107707
```

4. Discussion

Promising outcomes from the development and deployment of the Automatic Number Plate Recognition (ANPR) system demonstrate the technology's promise in a range of applications, including access control, law enforcement, and traffic monitoring. In conclusion, our system shows significant improvements in the efficiency of Automatic Number Plate Recognition, particularly in dynamic environments, as demonstrated by our comprehensive literature review. Our system is better than traditional methods in accuracy and reliability thanks to the use of preprocessing techniques, supervised machine learning algorithms. Our application provides solutions for both web and desktop users, meeting their diverse needs. The web application enables easy access and swift results, while the desktop version provides more accurate results and it has 2 solutions using different pretrained models. Our project sets the stage for future research by exploring the integration of advanced machine learning techniques to further enhance ANPR efficiency. Furthermore, the accessibility and adoption of our application can be enhanced by improving the user interface design and usability, which can lead to continued innovation and development in this field.

5. Github Links:

The machine learning part for desktop file is in *pts_easyocr_desktop.ipynb*, for web application is located in file called *ai.py*

https://github.com/aizeremarat/adv_prog_final.git

https://github.com/rnbsrva/adv_prog_final

https://github.com/youngAndMad/adv_prog_final.git

6. Video Link: https://www.youtube.com/watch?v=QTaQb_E58HM