TILAK RAJ CHADHA INSTITUTE OF MANAGEMENT AND TECHNOLOGY, YAMUNA NAGAR

Affiliated To Kurukshetra University, Kurukshetra

(APPROVED BY AICTE, NEW DELHI)

Project Thesis on Topic:

"Vehicle Number Plate Detection and Recognition"

Submitted by:

Dhruv Garg

(BCA -final year)

Submitted to:

Ms. Gunjan

(Assistant professor)

INDEX

Abstract

The need for vehicle number plate detection and recognition systems has grown as a result of the fast growth in vehicle ownership. This project is mainly about developing a Python based Vehicle Number Plate Detection and Recognition system. In this system, computer vision techniques are used to detect and recognize number plates of vehicles from images and video streams. It therefore seeks to offer an efficient solution for vehicle identification and monitoring by merging OpenCV for image processing with Tesseract OCR for text recognition.

1. Introduction

1.1. Background

Vehicle number plate recognition (VNPR) serves as an important technology in different applications such as traffic monitoring, parking management and law enforcement. Automatic license plate recognition systems are more efficient and accurate compared to manual methods. The system identifies number plates from vehicle pictures, and then extracts the alphanumeric characters which can be further processed.

1.2. Problem Statement

Manual identification of vehicle number plates is tedious and can easily lead to mistakes. An automated system is required which can accurately identify and recognize number plates from images or video feeds that have varying plate sizes, fonts, and lighting conditions.

1.3. Objective

The primary objectives of this project are:

1. **Detection:** Identify vehicle number plates within images.
2. **Recognition:** Extract and recognize alphanumeric characters from detected number plates.
3. **Integration:** Combine detection and recognition to provide a complete vehicle identification system.

2. Literature Review

2.1. Vehicle Number Plate Recognition Technologies

Vehicle number plate recognition systems generally involve two main stages: detection and recognition. Detection involves locating the number plate within an image, while recognition involves extracting and decoding the plate's text.

2.2. Existing Solutions

- **OpenALPR:** A commercial library that provides advanced license plate recognition features. It supports various plate formats and languages.
- **Plate Recognizer:** Offers API-based plate recognition services with high accuracy.
- **YOLO (You Only Look Once):** An object detection framework that can be adapted for license plate detection.

2.3. Techniques in Number Plate Recognition

- **Image Preprocessing:** Techniques such as grayscale conversion, thresholding, and edge detection to enhance image quality and improve detection accuracy.
- **Character Segmentation:** Dividing the number plate image into individual characters for recognition.
- **Optical Character Recognition (OCR):** Tools like Tesseract OCR are used to convert segmented images of characters into machine-readable text.

3. Problem Statement

A lot of cars on the road worldwide necessitate an efficient automatic vehicle number plate identification apparatus. Current manual methods for identifying number plates are time consuming, prone to errors and inefficient. The other drawback is that many automated systems cannot handle different plate designs, various environmental conditions as well as being able to process in real-time. These limitations cause inefficiencies in traffic management, parking enforcement and security operations leading to reduced accuracy rates and higher costs.

The task here is to develop Automated Vehicle Number Plate Detection and Recognition system which will accurately detect and read number plates from images or video feeds regardless of the number plate design or environmental factors. The system should provide real-time processing, ensuring high reliability

and efficiency. Addressing these issues will enhance vehicle identification, improve traffic/ parking management while at the same time boost security measures thus providing a cost-effective solution for various uses.

4. Objective

The project aims to achieve the following objectives:

1. **Detection Accuracy:** Develop a robust detection algorithm to identify number plates in diverse environments.
2. **Text Recognition:** Use OCR to accurately extract and decode characters from detected plates.
3. **User Interface:** Create a user-friendly interface for uploading images and displaying recognition results.

5. Methodology

5.1. Tools and Technologies

- **Programming Language:** Python
- **Libraries:** OpenCV, Tesseract OCR, NumPy, Matplotlib
- **Development Environment:** Anaconda, Jupyter Notebook or any Python IDE

5.2. System Design

1. **Image Acquisition:**
    - Capture images from a camera or load images from files.
2. **Image Preprocessing:**
    - Convert images to grayscale.
    - Apply image thresholding to enhance plate features.
    - Use morphological operations to clean up the image.
3. **Plate Detection:**
    - Detect contours in the image to locate potential number plates.
    - Use the aspect ratio and size of detected contours to filter out non-plate regions.
    - Extract the region of interest (ROI) containing the number plate.
4. **Character Segmentation:**
    - Extract individual characters from the detected plate region.
    - Apply image processing techniques to separate characters.
5. **Text Recognition:**
    - Use Tesseract OCR to recognize and decode text from segmented characters.

6. **Integration and Testing:**
    - Combine detection and recognition components into a cohesive system.
    - Test the system with a variety of images to ensure accuracy and robustness.

5.3. Implementation Steps

1. **Setup Environment:**
    - Install Python and required libraries.
    - Configure Tesseract OCR.
2. **Develop Detection Module:**
    - Implement image preprocessing and plate detection using OpenCV.
3. **Implement Recognition Module:**
    - Integrate Tesseract OCR for text recognition.
4. **Create User Interface:**
    - Develop a simple GUI using Tkinter or a web-based interface for ease of use.
5. **Testing and Evaluation:**
    - Evaluate the system on different datasets to assess detection and recognition performance.
    - Fine-tune algorithms based on test results.

6. System Requirements

6.1. Hardware Requirements

- **Processor:** Intel Core i5 or equivalent
- **RAM:** 8 GB
- **Storage:** 100 GB free disk space
- **Camera:** Optional, for live video feed

6.2. Software Requirements

- **Operating System:** Windows 10 / Linux (Ubuntu 20.04)
- **Python Version:** 3.7 or higher
- **Libraries:** OpenCV, Tesseract OCR, NumPy, Matplotlib
- **IDE:** Jupyter Notebook, PyCharm, or Visual Studio Code

6.3. Network Requirements

- **Internet Access:** Required for downloading libraries and tools.

7. References

- **OpenCV Documentation:** OpenCV. (n.d.). OpenCV Documentation. Retrieved from https://docs.opencv.org

- **Tesseract OCR Documentation:** Tesseract OCR. (n.d.). Tesseract OCR Documentation. Retrieved from https://github.com/tesseract-ocr/tesseract

- **Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 779-788).**

- **Harris, C., & Stephens, M. (1988). A Combined Corner and Edge Detector. In Proceedings of the Fourth Alvey Vision Conference (pp. 147-151).**

- **Pytesseract. (n.d.). Pytesseract Documentation. Retrieved from** https://pytesseract.readthedocs.io/en/latest/

- **Gonzalez, R. C., & Woods, R. E. (2002). Digital Image Processing. Prentice Hall.**

- **Nair, V., & Hinton, G. E. (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. In Proceedings of the 27th International Conference on Machine Learning (ICML) (pp. 807-814).**

- **Zhou, X., & Wang, X. (2017). YOLO9000: Better, Faster, Stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 6517-6525).**

- **Matplotlib Developers. (n.d.). Matplotlib Documentation. Retrieved from https://matplotlib.org/stable/contents.html**