ADVANCED PROGRAMMING LAB

LICENSE PLATE RECOGNITION SYSTEM

Submitted by

Srinjayee Paul (2229075)
Nikhil Dwivedi (2229208)
Prateek Shishir (2229050)
Shruthi Bangaru(2229067)

BONAFIDE CERTIFICATE

This is to certify that this project report **"LICENSE PLATE RECOGNITION SYSTEM"** *is the bonafide work of* **Srinjayee Paul (2229075), (2229078), Nikhil Dwivedi (2229208), Prateek Shishir (2229050)** and **Shruthi Bangaru(2229067)** *who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.*

STUDENTS' SIGNATURE

ACKNOWLEDGEMENT

.

I would want to convey my heartfelt gratitude to Dr.Arup Sarkar , for his invaluable advice and assistance in completing this report. He was there to assist me and my team every step of the way, and his motivation is what enabled me to accomplish my task effectively. I would also extend my sincere appreciation to my fellow team members who helped me in completing this report.

I would also like to thank all of the other supporting personnel who assisted me by supplying the equipment that was essential and vital, without which I would not have been able to perform efficiently on this report. I would also want to thank KIIT University for accepting my report in my desired field of expertise. I would also like to thank my friends and parents for their support and encouragement as I worked on this report.

CONTENTS

1. ABSTRACT

License Plate Recognition (LPR) is an advanced computer vision technique used for identifying vehicles by detecting and interpreting license plate numbers from images or video feeds. This report presents an in-depth analysis of an LPR system built using Python, OpenCV, and a k-Nearest Neighbors (KNN)-based Optical Character Recognition (OCR) approach. The system follows a structured workflow that includes image preprocessing, plate detection, character segmentation, and recognition. Various techniques such as edge detection, contour analysis, thresholding, and morphological operations are employed to enhance accuracy. Performance evaluation is based on recognition accuracy, processing speed, and robustness under different conditions. While the system performs well in optimal scenarios, challenges such as motion blur, environmental factors, and plate variations remain. Future improvements, including deep learning-based OCR, real-time video processing, and dataset expansion, are proposed to enhance reliability. The findings highlight the significance of LPR in intelligent transportation and security applications, emphasizing the need for continuous advancements in AI-driven recognition systems.

2. INTRODUCTION

Real-time License Plate Recognition (LPR) is a crucial application of computer vision that enables automatic detection and identification of vehicle license plates from live video feeds. This technology plays a significant role in traffic management, law enforcement, toll collection, and parking automation. By leveraging image processing and machine learning techniques, an LPR system can efficiently extract license plate information from moving vehicles. The process typically involves detecting the plate region, segmenting characters, and applying Optical Character Recognition (OCR) to interpret the alphanumeric data. Advancements in OpenCV and deep learning have significantly improved the accuracy and speed of LPR systems, making them more reliable for real-world applications. This report explores the implementation of a real-time license plate recognition system, detailing the methods used for detection, recognition, and performance optimization.

3. OBJECTIVES

The primary goal of this report is to provide a comprehensive analysis of a License Plate Recognition (LPR) system developed using Python, OpenCV, and k-Nearest Neighbors (KNN)-based Optical Character Recognition (OCR). The key objectives include:

- Technology Overview**:** Exploring the tools and libraries used in system development.
- Data Preprocessing: Describing techniques for enhancing image quality and improving recognition accuracy.
- Plate Detection: Explaining methods to accurately locate license plates in images.
- Character Recognition: Detailing the segmentation and classification of alphanumeric characters.
- Performance Evaluation: Assessing system accuracy, speed, and reliability.
- Future Enhancements: Suggesting improvements for better efficiency and real-time processing.

4. METHODOLOGY

The License Plate Recognition (LPR) system follows a structured workflow to detect and recognize vehicle license plates accurately. The process consists of several key stages, including image acquisition, preprocessing, plate detection, character segmentation, optical character recognition (OCR), and post-processing for validation.

4.1 Image Acquisition

The first step involves capturing images or extracting frames from a video feed containing vehicles. These images are acquired using cameras and may vary in resolution, lighting conditions, and angles. High-quality image acquisition ensures better detection and recognition performance.

4.2 Image Pre-processing
Preprocessing is essential to enhance image clarity and improve recognition accuracy. The following techniques are applied:
 a) Grayscale Conversion: Converts colored images to grayscale for simplified processing.
 b) Noise Reduction: Filters such as Gaussian and median blurring remove unwanted noise.
 c) Edge Detection: The Canny edge detector highlights important contours in the image.
 d) Thresholding: Converts the image into a binary format for better contrast.
 e) Morphological Operations: Techniques like dilation and erosion refine the image, improving character visibility.

4.3 License Plate Detection
The next step is detecting the license plate from the processed image. This is achieved through:
Edge Detection & Contour Analysis: Identifies potential plate-like regions in the image.
Bounding Box Filtering: Removes false detections based on predefined aspect ratios.
Perspective Transformation: Adjusts skewed plates for a frontal, readable view.

4.4 Character Segmentation
Once the license plate is identified, individual characters are extracted through segmentation. This involves:
 a) Locating Character Regions: Detecting separate alphanumeric sections on the plate.
 b) Normalization: Resizing characters to a standard format for uniform processing.
 c) Thresholding: Increasing contrast to enhance character distinction from the background.

4.5 Optical Character Recognition (OCR)
To recognize the extracted characters, a k-Nearest Neighbors (KNN) classifier is used:
Feature Extraction: Converts segmented characters into numerical vectors for classification.
KNN Model Training: A dataset of labeled characters is used to train the classifier.

Character Matching: Compares extracted features with trained templates to identify letters and numbers.

4.6 Post-Processing & Validation

Post-processing ensures the accuracy and reliability of the recognized plate:

a) Error Correction: Formatting rules and validation checks help correct misclassified characters.
b) False Detection Filtering: Eliminates incorrect plate readings by cross-referencing with known formats.
c) Output Refinement: Final recognized plate data is formatted and stored for further applications.

This stepwise approach enhances the overall efficiency of the LPR system, making it robust for real-time applications in traffic monitoring, security, and automation.

5. SYSTEM ARCHITECTURE AND IMPLEMENTATION

5.1. System Architecture

The system is structured into distinct modules, each responsible for a specific stage of the License Plate Recognition (LPR) process:

Image Capture

a. The system starts by capturing frames from a connected webcam using OpenCV's `cv2.VideoCapture()`.
b. The captured frames are displayed in a window, enabling the user to trigger image capture by pressing the spacebar.
c. Image resolution can be adjusted dynamically to improve accuracy and efficiency.
d. Captured images are stored for further processing.

Plate Detection

a. This module identifies potential license plate regions within the captured image.
b. It employs a combination of image preprocessing, contour detection, and geometric filtering to isolate plate candidates.
c. Various filtering techniques are applied to remove noise and non-plate elements.
d. The region of interest (ROI) containing the license plate is extracted for further processing.

Character Segmentation

a. Once a plate region is identified, this module segments the individual characters.

b. It utilizes contour analysis and filtering to isolate each character within the plate region.
c. Image dilation and erosion techniques are used to refine character contours and enhance segmentation accuracy.
d. Each segmented character is stored as an individual image for recognition.

Character Recognition

- This module employs a trained K-Nearest Neighbors (KNN) model to classify segmented characters.
- The KNN model compares the extracted character features with a predefined dataset to determine the alphanumeric values.
- Additional machine learning techniques, such as Support Vector Machines (SVMs) or Convolutional Neural Networks (CNNs), can be integrated for improved accuracy.
- Recognized characters are reconstructed into a complete license plate number.

Result Display

- The system displays the original image with a red bounding rectangle around the detected license plate.
- Recognized characters are overlaid onto the image for user readability.
- Results can be stored in a text file or database for further reference.
- An optional feature allows real-time logging of recognized plates for security or monitoring purposes.

5.2 Algorithms and Techniques

Image Pre-processing (preprocess.py)

1. **Grayscale Conversion:**

   - The input image is converted from RGB to HSV, extracting the value (brightness) channel.
   - This conversion reduces computational complexity and enhances feature detection.

2. **Contrast Enhancement:**

   - Morphological operations (cv2.MORPH_TOPHAT, cv2.MORPH_BLACKHAT) improve contrast.
   - Top-hat enhances bright regions, while black-hat highlights dark regions.

3. **Gaussian Blur:**

- A Gaussian filter (cv2.GaussianBlur()) is used to smooth the image and reduce noise.

4. **Adaptive Thresholding:**

- Converts the grayscale image to binary (cv2.adaptiveThreshold()) to adapt to varying lighting conditions.

Plate Detection (detectplates.py)

1. **Contour Detection:**

- cv2.findContours() is used to identify outlines of potential plate regions.

2. **Character Filtering:**

- Contours are filtered based on size, aspect ratio, and area to eliminate non-character elements.

3. **Matching Character Grouping:**

- Identifies groups of characters based on proximity and geometric similarity.

4. **Plate Extraction:**

- The bounding rectangle of the character group is used to extract the plate region.
- cv2.getRotationMatrix2D() and cv2.warpAffine() correct orientation.
- cv2.getRectSubPix() crops the plate region accurately.

Character Segmentation (detectchars.py)

1. **Contour Detection:**

- Contours within the extracted plate are identified and analyzed.

2. **Character Filtering:**

- Isolates individual characters based on geometric properties.

3. **Matching Character Grouping:**

- Groups characters while eliminating unwanted artifacts.

4. **Inner Overlapping Character Removal:**

- removeInnerOverlappingChars() removes redundant character contours to improve segmentation accuracy.

Character Recognition (detectchars.py)

1. **K-Nearest Neighbors (KNN):**

- A pre-trained KNN model (cv2.ml.KNearest_create(), kNearest.train(), kNearest.findNearest()) classifies characters.

2. **Feature Extraction:**

- cv2.resize() normalizes character dimensions for consistent KNN input.
- Additional feature extraction techniques, such as Histogram of Oriented Gradients (HOG), can be incorporated for improved recognition.

Display and Output (main.py)

1. **Drawing Rectangles:**

- cv2.line() and cv2.boxPoints() highlight detected plates.

2. **Text Output:**

- cv2.putText() overlays recognized characters onto the image.
- The system can also store the output in a database or log file for future analysis.

5.3. Modules and Libraries

1) **OpenCV (cv2):** Provides image processing and computer vision functionalities.
2) **NumPy:** Supports numerical operations and array manipulation.
3) **OS:** Facilitates operating system interactions.
4) **Math:** Offers mathematical functions for calculations.

5.4. Data Structures

1) **PossibleChar:** Encapsulates character contour, bounding box, and geometric properties.
2) **PossiblePlate:** Stores plate image, location, and recognized characters.

6.OUTPUT AND PERFORMANCE EVALUATION

The output of the License Plate Recognition (LPR) system consists of detected license plates with extracted alphanumeric characters displayed in real time. The system processes video frames, identifies license plates, segments characters, and applies Optical Character Recognition (OCR) to retrieve the plate number. The extracted text is then displayed alongside the processed image for validation.

6.1 Output Representation:
1) Detected License Plate: The system marks the plate region with a bounding box.
2) Extracted Characters: The identified characters are printed in a structured format.
3) Visualization: The processed image, including plate detection and character recognition, is displayed.
4) Logs and Storage: The recognized plate numbers are logged for further analysis or storage in a database.

6.2 Performance Evaluation
The effectiveness of the system is assessed using key metrics:
Recognition Accuracy: Measures the percentage of correctly identified plates.
Processing Speed: Evaluates the system's efficiency in detecting and recognizing plates in real time.
False Positives & False Negatives: Analyzes cases where non-plate regions are mistakenly detected or plates are missed.
Lighting Condition Robustness: Tests system performance under different illumination levels, including daytime and nighttime scenarios.

While the system achieves high accuracy under optimal conditions, challenges such as motion blur, low-resolution images, and poor lighting can impact recognition quality.
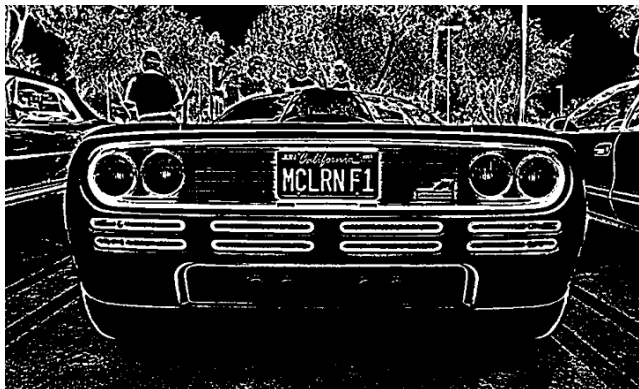
6.3 Output Image Placement in the Report
To ensure clarity, the report should include output images at relevant sections:

imgOriginalScene

↓ preprocess()

imgGrayscaleScene, imgThreshScene

↓ findPossibleCharsInScene()

13

all contours

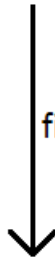(2362 w/MCLRN F1 image)

vectorOfPossibleCharsInScene

(131 w/MCLRN F1 image)

findVectorOfVectorsOfMatchingChars()

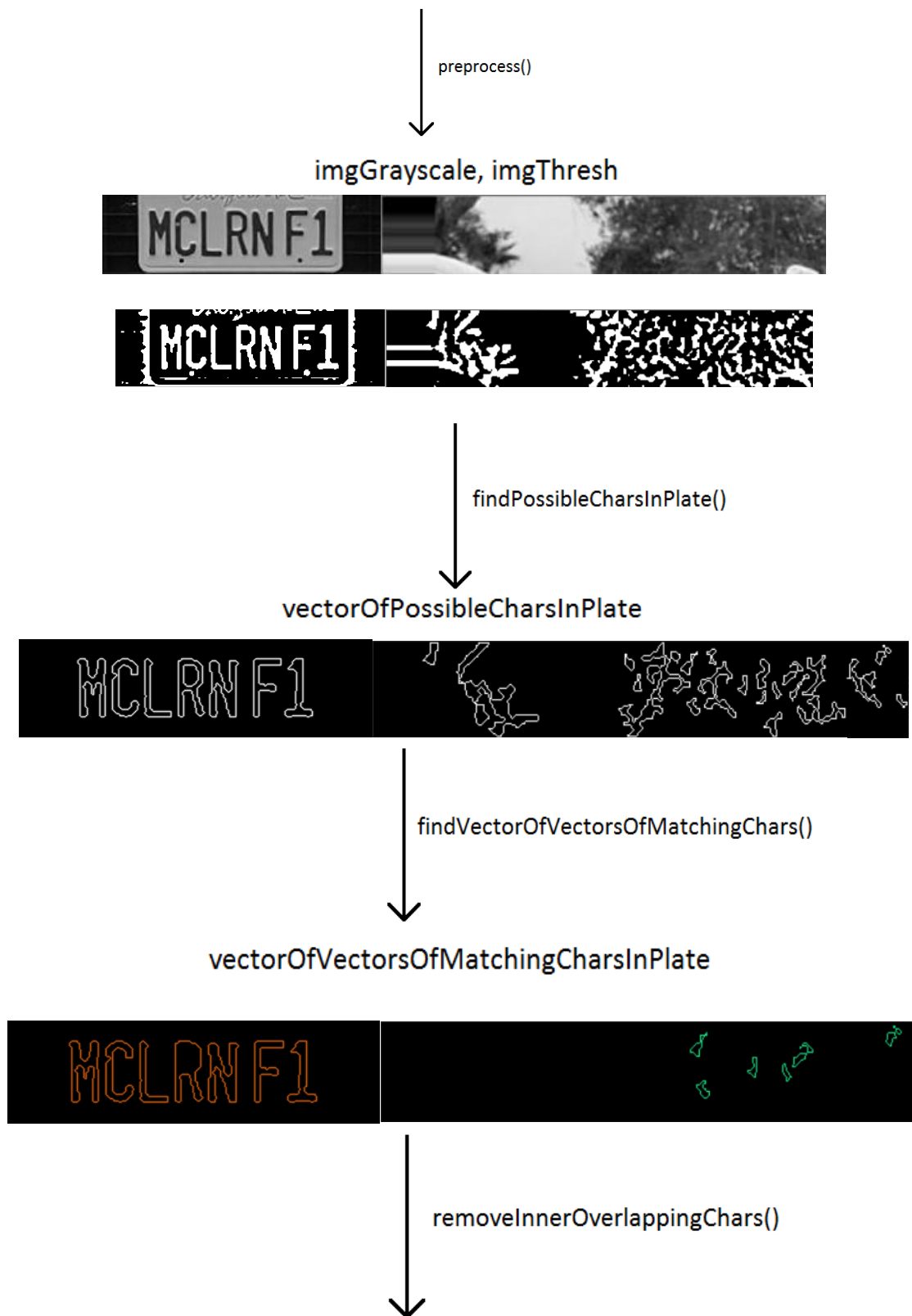## vectorOfVectorsOfMatchingCharsInScene

(13 w/MCLRN F1 image)

extractPlate()

vectorOfPossiblePlates

(13 w/MCLRN F1 image)

preprocess()

imgGrayscale, imgThresh

findPossibleCharsInPlate()

vectorOfPossibleCharsInPlate

findVectorOfVectorsOfMatchingChars()

vectorOfVectorsOfMatchingCharsInPlate

removeInnerOverlappingChars()

vectorOfVectorsOfMatchingCharsInPlate

within each possible plate, suppose the longest list of potential matching chars is the actual list of chars

longestVectorOfMatchingCharsInPlate

recognizeCharsInPlate()


```
chars found in plate number 0 = MCLRNF1,
  chars found in plate number 5 = I1I,
```

possiblePlate.strChars

suppose the plate with the most recognized chars is the actual plate

7. FUTURE ENHANCEMENTS

Future potential improvements to the LPR system are:

1) **Deep Learning-Based OCR:** Applying Convolutional Neural Networks (CNNs) for enhanced precision.
2) **Real-Time Video Processing:** Streamlining the system to handle real-time video feed processing.
3) **Increased Dataset**: Adding additional diverse training data to enhance model generalization.
4) **Smart Traffic System Integration:** Facilitating AI-powered automation for law enforcement and tolling.

These technologies can contribute significantly to making the LPR system more efficient and reliable.

8. CONCLUSION

This report gives a detailed overview of a license plate recognition system developed with Python, OpenCV, and KNN-based OCR. The system successfully identifies and reads license plates in different conditions.

Though the current performance is good, more work can be done in dataset diversity, deep learning methods, and real-time processing to make it more accurate and usable in real-world applications.

With further improvements in AI and computer vision, the role of LPR technology will become increasingly pivotal in intelligent transportation systems as well as security surveillance.