License Plate Recognition System

Hoza Violeta Maria
Technical University of Cluj-Napoca
Email: Hoza.Il.Violeta@student.utcluj.ro

***Abstract*—This project implements a basic license plate recognition system using OpenCV and C++. The goal is to detect license plates in images and read characters using simple computer vision techniques. The system works in three main steps: first, it finds license plates by looking for rectangular shapes in the image using edge detection and contour analysis; then, it extracts and processes the detected plate region to enhance contrast and readability; and, finally, it recognizes the characters using Tesseract OCR.**

I. Introduction

License Plate Recognition (LPR) systems have become a fundamental component in intelligent transportation systems, with applications in traffic monitoring, law enforcement, automated toll collection, and access control in parking facilities. According to recent market research, the global market size for automatic license plate recognition is calculated at USD 3.79 billion in 2024 and is predicted to reach around USD 9.27 billion by 2034, highlighting the increasing importance of this technology [1].

LPR systems utilize computer vision and optical character recognition (OCR) techniques to extract license plate information from images or video streams. The main challenge in LPR systems is to achieve high accuracy in various real-world conditions.

For this project, my objective is to develop a basic yet functional license plate recognition system focusing on three core components:

- **Plate Detection**: Implementing a reliable method to identify and localize a license plate in an input image/frame
- **Character Segmentation**: Creating a pipeline to isolate individual characters
- **Character Recognition**: Developing a practical Optical Character Recognition (OCR) solution to recognize the extracted characters.

As a student implementing for the first time an LPR system, I will take a step-by-step approach to ensure that each component works correctly before moving forward. First, I will set up my development environment by installing Tesseract OCR and then collect a small set of test images showing license plates. For plate detection, I will start by processing images through grayscale conversion, Gaussian blur, and Canny edge detection to identify potential plate locations and then use contour analysis to filter for rectangular shapes with appropriate aspect ratios. Once plates are reliably detected, I will focus on character segmentation by applying Otsu's thresholding to create binary images and morphological operations to clean up noise before isolating individual characters. For the recognition phase, I will integrate Tesseract OCR, configuring it specifically for license plate reading by setting the appropriate page segmentation mode and character whitelist. Throughout the process, I will test each component independently using my collected images, beginning with ideal cases before tackling more challenging scenarios like poor lighting or angled plates.

II. Bibliographic research

License plate recognition has evolved significantly over the past few decades, transitioning from classical computer vision techniques to modern deep learning-based systems.

The typical pipeline involves the following phases:

- **Image acquisition**: Capturing vehicle images using cameras, which can be static (toll gates) or mobile (police vehicles). The image quality at this stage significantly impacts subsequent processing.
- **Preprocessing**: Enhancing image quality through:
  - Grayscale conversion to simplify processing
  - Noise reduction using Gaussian blur
  - Contrast enhancement for better edge detection
- **Plate localization**: Identifying potential plate regions using:
  - Edge detection (Sobel, Canny, Prewitt, Roberts): Canny edge detection, introduced by Canny [2], is a multi-stage algorithm and is often preferred due to its effectiveness in identifying and highlighting edges within digital images (it detects sharp transitions in intensity).
  - Color segmentation (for colored plates)
  - Geometric analysis (aspect ratio, rectangularity)
- **Character segmentation**: Isolating individual characters through:
  - Binarization (Otsu's thresholding)
  - Connected component analysis
  - Vertical/horizontal projections
- **OCR**: Recognizing characters using either:
  - Traditional methods (template matching)
  - Feature-based classifiers: SVM (Support Vector Machine) with HOG (Histogram of Oriented Gradients) features
  - Machine learning approaches (neural networks)
- **Post-processing**: Validating and correcting results using:
  - Dictionary checks
  - Syntax rules (plate format knowledge)
  - Temporal consistency in video streams

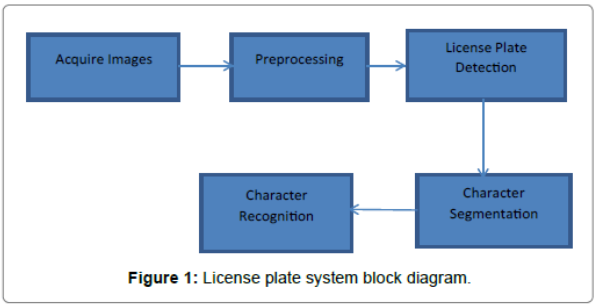

Figure 1: License plate system block diagram.

A. Historical Development

Early LPR systems relied heavily on traditional image processing techniques:

- **Edge-Based Methods**: Edge detection techniques have been pivotal in license plate localization. Kamat and Ganesan proposed a two-stage approach using Sobel edge detection followed by morphological operations to identify potential plate regions with high edge density, achieving 85.7% accuracy in varied conditions [3]. The Sobel operator calculates the gradient of image intensity at each pixel, providing both edge magnitude and direction information that helps distinguish plate boundaries from other vehicle parts. Hough transform techniques have also been widely applied to detect the rectangular shape of license plates after edge detection. Draghici et al. [4] combined Canny edge detection with probabilistic Hough transforms to locate license plates by identifying parallel line segments with appropriate spacing, demonstrating effectiveness even with perspective distortion.
- **Texture and Color-Based Approaches**: Texture analysis provides another avenue for plate localization. Parisi et al. [5] utilized Gabor filters to capture the distinctive texture patterns of license plates - the regular arrangement of characters creates a unique texture signature compared to vehicle body and background. Their approach achieved 91.2% detection accuracy by analyzing the spatial frequency characteristics of image regions. For plates with distinctive colors, HSV (Hue, Saturation, Value) color space segmentation has proven effective.
- **Character Segmentation Techniques**: Character segmentation in traditional systems relies heavily on binarization followed by connected component analysis. Anagnostopoulos et al. [6] implemented an adaptive thresholding approach that determines optimal binarization parameters locally rather than globally, addressing problems with uneven illumination across the plate surface. Vertical projection profiles have been particularly useful for separating characters on standard license plates. This technique projects the binary image onto the horizontal axis, creating a histogram where valleys indicate spaces between characters. Zheng et al. [7] enhanced this approach by incorporating character width constraints to handle touching or broken characters, improving segmentation accuracy by 7.3% compared to basic projection methods.
- **Traditional Recognition Methods**: Template matching, despite its simplicity, has been effectively deployed for license plate character recognition. Martinsky [8] implemented a normalized cross-correlation template matching system that achieved 89% character recognition accuracy for frontal images under good lighting conditions. The approach created templates for each alphanumeric character and compared them against segmented characters, selecting the highest correlation match.

These methods, while effective in controlled environments, often struggle with real-world variations in lighting, perspective, and plate formats. Environmental factors such as poor lighting (night, shadows) [9], weather conditions (rain, snow), vehicle speed variations, and distance ranges between camera and vehicle significantly impact recognition rates. Rotated or skewed plates present another significant challenge, with most traditional methods requiring plates to be approximately horizontal for reliable character segmentation. In addition, technical challenges arise from different plate formats, non-standard fonts, and physical obstructions that may partially obscure license plates. These variables make it difficult to maintain consistent performance without resorting to expensive specialized hardware or computational resources.

B. Modern Approaches

Recent advancements in deep learning have significantly improved LPR accuracy. Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are widely used for plate detection and character recognition. Models such as YOLO (You Only Look Once) and SSD (Single Shot MultiBox Detector) provide high-speed and accurate plate localization, while OCR engines like CRNN combine CNNs for feature extraction and RNNs for sequence prediction. Although these achieve a 95-99% accuracy [10], they require substantial computational resources and large labeled datasets.

C. Tesseract OCR

Tesseract is an open-source OCR engine originally developed by Hewlett-Packard in the 1980s and currently maintained by Google since 2006 [11]. It was initially based on feature extraction and matching techniques, but since version 4.0, Tesseract has incorporated Long Short-Term Memory (LSTM) networks, significantly improving its accuracy and robustness. While not specifically designed for license plate recognition, Tesseract can be configured for this task by:

- Specifying appropriate page segmentation modes (e.g., treating the license plate as a single line of text)
- Defining a character whitelist limited to alphanumeric characters found on license plates
- Training or fine-tuning the model on license plate fonts if necessary.

The engine supports multiple languages and character sets, making it suitable for international license plate formats.

III. METHOD

The license plate recognition system implemented in this project follows a pipeline based on classical computer vision techniques combined with modern OCR.

A. Image Preprocessing

The preprocessing stage prepares the input image for plate detection by enhancing relevant features:

1) **Grayscale Conversion**: The color image is converted to grayscale to simplify processing while maintaining structural information.
2) **Noise Reduction**: Applying a bilateral filter with parameters (11, 17, 17) that effectively reduces noise while preserving edges. Unlike a regular Gaussian blur, the bilateral filter considers both spatial distance and color intensity differences, making it particularly effective for preserving plate boundaries.
3) **Edge Detection**: Canny edge detection identifies strong gradients corresponding to plate boundaries.
4) **Morphological Processing**: Applying a dilation operation with a simple structuring element to connect nearby edges, which helps form complete contours around license plates.

As evident in the implementation, each preprocessing step includes visual debugging outputs when a flag is enabled, allowing for easy verification of the pipeline's effectiveness.

B. License Plate Detection

After preprocessing, the system detects license plates through the following steps:

1) **Contour Extraction**: All external contours are detected using OpenCV's findContours function
2) **Contour Filtering**: Contours are sorted by area and filtered based on:
   - **Aspect ratio**: $2.0 < \frac{width}{height} < 6.0$
   - **Minimum area**: $area > image\_area \times 0.002$ (eliminates very small regions)
   - **Maximum area**: $area < image\_area \times 0.1$ (eliminates very large regions)
   - **Minimum width**: The plate should be wide enough to be readable: $width > image\_width \times 0.08$ (ensures sufficient detail for OCR)
   - **Shape measures**: Using extent (area/rect area) and solidity (area/convex hull area) to ensure the contour resembles a rectangular plate

This helps distinguish plates from irregular shapes with a similar aspect ratio.

C. Plate Region Extraction

Verified plates are extracted and rectified:

1) **Orientation Correction**: Using OpenCV's minAreaRect function to find the best-fitting rotated rectangle, enabling the system to handle tilted license plates.
2) **Perspective Transformation**: Applying a rotation correction based on the angle of the rotated rectangle to normalize the plate orientation.
3) **Region Extraction**: Using getRectSubPix to extract the plate region with proper size, including a 10% padding to ensure the entire plate is captured.
4) **Border Removal**: Applying an additional crop to remove potential borders around the plate text (implemented as a percentage-based ROI extraction).

This implementation allows for accurate extraction even when the plate is captured at non-frontal angles, which is crucial for real-world applications.

D. Plate Preprocessing for OCR

Before applying OCR, the extracted plate undergoes specific preprocessing to optimize character recognition:

1) **Grayscale Conversion**
2) **Contrast Enhancement**: Applying histogram equalization followed by CLAHE (Contrast Limited Adaptive Histogram Equalization) with a clip limit of 2.0 and grid size of 8×8 to enhance character visibility.
3) **Noise Reduction**: Using Gaussian blur with a 3×3 kernel to reduce high-frequency noise that might interfere with character recognition.
4) **Binarization**: Adaptive thresholding is used to convert the image to binary. The inverse binary image (white text on black background) is used as it often gives better OCR results.
5) **Morphological Cleanup**: A closing operation with a 2×2 rectangular kernel removes small holes in the characters.

These preprocessing steps are particularly important for license plate recognition, as they compensate for varying lighting conditions, dirt, and other real-world factors that might affect character visibility.

E. OCR and Post-processing

The final step applies Tesseract OCR with specialized configuration:

1) **OCR Configuration**: Tesseract is configured with specific parameters for license plate recognition:
   - Page segmentation mode: to treat the plate as a single text line
   - Character whitelist: to restrict recognition to relevant characters (capital letters and digits)
2) **Multi-variant Processing**: Three versions of the plate image are processed:
   - The processed binary image
   - An upscaled version (2x scaling) for better detail resolution
   - An inverted version to handle both dark-on-light and light-on-dark plates
3) **Romanian Plate Format Validation**: The recognized text is validated against Romanian license plate formats using regular expressions

4) **Confidence Scoring**: A confidence score combines OCR confidence with format validation:

$$score = OCR\_confidence + 0.5 \times isValidFormat \quad (1)$$

5) **Result Selection**: The version with the highest confidence score is selected as the final result.

F. Implementation Details

The system was implemented in C++ using OpenCV 4.5 and Tesseract 4.1.1 with the following characteristics:

- **OCR Optimization**: Multiple preprocessing approaches are tried in parallel with a scoring system to select the best result, improving robustness across varying plate conditions.
- **Romanian Plate Specialization**: The system is specifically tuned for Romanian license plates with county code validation:

$$countyCode \in \{"B", "AB", "AR", "AG", ..."VN"\} \quad (2)$$

- **Visual debugging** at each processing step when enabled.
- **Result Persistence**: Detected plates with recognized text can be saved along with their XML annotations for later use in training or evaluation datasets.

G. System Overview

The complete LPR system integrates multiple components in a modular, sequential pipeline written in C++:

- The program begins by allowing the user to select an image from disk through a graphical file dialog.
- Once loaded, the image is preprocessed using grayscale conversion, bilateral filtering, and Canny edge detection.
- Candidate license plate regions are located by analyzing contours and filtering them based on geometric heuristics.
- For each detected region, a perspective-corrected subimage is extracted and passed through a contrast-enhancing preprocessing stage.
- The resulting cleaned plate image is passed to Tesseract OCR for recognition.
- The output text is validated against known Romanian license plate formats and the best candidate is selected.
- Results are optionally displayed with annotations and saved to disk (including XML format with bounding box and recognized text).

IV. Tests and Results

The implemented license plate recognition system was evaluated on a subset of real-world images of Romanian vehicles. The goal of this section is to assess the detection and recognition accuracy of the system in various conditions and to understand its limitations.

A. Dataset

For evaluation, I used a subset of the publicly available license plate dataset by Robert Lucian [12], which contains real-life vehicle images captured under different lighting conditions, angles, distances, and levels of occlusion. From this dataset, I selected 46 images, focusing on diversity in environment and plate visibility.

The images include:

- Frontal and slightly angled views
- Clean plates
- Daylight and evening conditions with varying illumination

B. Processing Pipeline — Visual Example

Figure 1 shows a full step-by-step visualization for a successful case: the vehicle with plate `B 144 MOS`. The system correctly localized the license plate and performed OCR with a high-confidence result.

Fig. 1. Pipeline part 1: original, grayscale, filtered, edges, plate detected, final OCR result

Fig. 2. Pipeline part 2: plate close-up, contrast enhancement, blurred, binary

C. Evaluation

The license plate detection system was evaluated using some standard metrics:

- **Precision**: Measures the proportion of detected license plates that are actual plates.

$$Precision = \frac{CorrectlyDetectedPlates}{TotalPredictedPlates} \times 100 \quad (3)$$

- **Recall**: Measures the proportion of actual license plates that were correctly detected.

$$Recall = \frac{CorrectlyDetectedPlates}{TotalGroundTruthPlates} \times 100\% \quad (4)$$

- **Recognition Accuracy** — whether the OCR output matches the correct license plate text.

$$OCRAccuracy = \frac{CorrectOCRResults}{DetectedPlates} \times 100\% \quad (5)$$

The system was evaluated using an IoU (Intersection over Union) threshold of 0.5, which is a standard threshold in object detection tasks. This means that a predicted bounding box is considered a correct detection if it overlaps with a ground truth box by at least 50%.

D. Detection Results Analysis

The evaluation metrics for the license plate detection component revealed:

Metric	Value
Total nr of plates	59
Total predicted plates	46
Plates Correctly Detected	46
Correct OCR Results	42
Precision	100%
Recall	77.96%
OCR Accuracy	91.30%

TABLE I
PERFORMANCE OF THE LPR SYSTEM ON A SUBSET OF THE DATASET PROVIDED BY ROBERT LUCIAN

These results reveal important characteristics of the detection system:

- **Perfect Precision (100%)**: Every region the system identified as a license plate was indeed a license plate. This indicates that the filtering criteria based on aspect ratio, area constraints, and shape measures were highly effective at eliminating false positives. The system makes very conservative predictions, only declaring a region as a license plate when it has high confidence.
- **Moderate Recall (77.96%)**: The system detected approximately three-quarters of all license plates in the dataset. The missed detections (13 plates) likely occurred due to:
  - Plates with low contrast that didn't produce strong edges
  - Partially occluded plates
  - Plates in poor lighting conditions
  - Plates at a distance that fell below the minimum size threshold

The detection performance metrics demonstrate a clear trade-off between precision and recall. The system was tuned to prioritize precision over recall, which is often preferable in license plate recognition applications where false positives (incorrectly identifying a non-plate region as a plate) can be more problematic than false negatives (missing some plates).

E. Performance Insights

The system performed better across the test set when:

- Plates were captured at near-frontal angles (within approximately ±15 degrees)
- License plates had good contrast between characters and background
- Fully visible and not occluded by objects or dirt

Most recognition failures were due to:

- Very low lighting conditions where edge detection failed to identify plate boundaries
- Motion blur and significant image noise, which affected character segmentation
- Plates being captured at steep angles or partially cropped
- Strong reflections or overexposure from headlights
- Partial occlusions of plate characters by dirt, shadows, or other objects

Despite these challenges, the preprocessing pipeline and multiple OCR variants (e.g., binary, inverted, upscaled) greatly improved robustness. The additional logic for validating Romanian plate formats also helped reject false positives from OCR errors.

V. CONCLUSION

In this project, I designed and implemented a classical license plate recognition (LPR) system using OpenCV and Tesseract OCR. The system can:

- Detect license plates in images using edge detection and contour analysis
- Preprocess and isolate plate regions
- Apply OCR and post-processing to extract and validate alphanumeric text

Tested on 46 images of Romanian license plates, the system achieved a precision of 100% and recall of 77.96%, demonstrating excellent reliability in avoiding false positives while still identifying over three-quarters of all license plates in challenging real-world scenarios. These results show that classical methods, when well-designed, can achieve strong performance on constrained datasets without requiring deep learning.

Future Work

To further improve accuracy and robustness, I could:

- Integrate a deep learning-based detector like YOLOv5 for more robust plate localization in challenging scenarios
- Train a custom Tesseract model on Romanian license plates for better OCR accuracy
- Add real-time video stream support with plate tracking and temporal consistency
- Introduce multi-language support for international use.

REFERENCES

[1] Precedence research - Automatic Number Plate Recognition System Market Size, Share and Trends 2024 to 2034

[2] Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *8*(6), 679-698.

[3] V. Kamat and S. Ganesan, "An efficient implementation of the Hough transform for detecting vehicle license plates using DSP's," *in Proceedings of Real-Time Technology and Applications, 1995, pp. 58-59.*

[4] Sorin Draghici, "A neural network based artificial vision system for licence plate recognition," *in International Journal of Neural Systems, vol. 8, no. 1, pp. 113-126, 1997.*

[5] R. Parisi, E. D. Di Claudio, G. Lucarelli, and G. Orlandi, "Car plate recognition by neural networks and image processing," *in IEEE International Symposium on Circuits and Systems, 1998*

[6] C. N. E. Anagnostopoulos, I. E. Anagnostopoulos, V. Loumos, and E. Kayafas, "A license plate-recognition algorithm for intelligent transportation system applications," *in IEEE Transactions on Intelligent Transportation Systems, vol. 7, no. 3, pp. 377-392, 2006.*

[7] D. Zheng, Y. Zhao, and J. Wang, "An efficient method of license plate location," *in Pattern Recognition Letters, vol. 26, no. 15, pp. 2431-2438, 2005*.
[8] O. Martinsky, "Algorithmic and mathematical principles of automatic number plate recognition systems," *B.Sc. Thesis, BRNO University of Technology, 2007*.
[9] Yuren Du, Wen Shi, and Caiyun Liu, "Research on an Efficient Method of License Plate Location", *2012 International Conference on Applied Physics and Industrial Engineering*
[10] License Plate Recognition Techniques: Comparative Study - Munaisyah Abdullah, Salah Mohammed Al-Nawah, Husna Osman, Jasrina Jaffar
[11] R. Smith, "An Overview of the Tesseract OCR Engine" (2007).
[12] R. Lucian, "Romanian (European Union) Dataset of License Plates", GitHub repository, https://github.com/RobertLucian/license-plate-dataset