License Plate Detection and Recognition System with Only Image Processing

Ali Han Batmazoğlu[1]

1. Department of Computer Engineering, Eskisehir Technical University
alihanbatmazoglu@ogr.eskisehir.edu.tr

Abstract

A license plate detection and recognition system is commonly used for control and security purposes. In this system, our main goal is to detect license plates using only image processing techniques, without relying on artificial intelligence models, even though such tasks are typically easier with AI. The system is designed for rectangular-shaped plates. An LPRS consists of two main stages. The first stage involves detecting the license plate region. In this stage, several operations are applied to vehicle images, including grayscale conversion, Canny filtering, dilation, skeletonization, and contour detection. The second stage involves cropping the plate region and recognizing the characters on the plate. Here, we also attempted to solve the problem using only image processing techniques without any AI models. In this part, the cropped original image is first converted to grayscale, then histogram equalization and thresholding are applied, and finally, characters are detected by matching them with a predefined set of alphabet and number templates. For implementation, Python programming language and the OpenCV library were used.

Keywords: License Plate Recognition, Image Processing, Filtering, Contours, Optical Character Recognition.

1. Introduction

With the increasing number of vehicles today and the growing need for advanced traffic control, Automatic License Plate Recognition Systems (LPRS) have become a critical component in intelligent transportation, security, and automation. LPRS systems are widely used in various areas such as smart parking management, toll collection systems, border security, traffic violation detection, and vehicle tracking. The primary objective of these systems is to automatically detect and identify the alphanumeric characters on vehicle license plates.

Traditional LPRS approaches in the literature generally consist of three main stages: (1) license plate detection, (2) character segmentation, and (3) character recognition. High accuracy rates can be achieved using models such as Artificial Neural Networks (ANN), Support Vector Machines (SVM), and more recently, deep learning architectures like Convolutional Neural Networks (CNN). However, these methods often require large datasets and high computational resources. Given the limited size of our dataset, we aimed to demonstrate that this problem can be effectively solved using only image processing techniques.

In this study, we propose a template matching-based LPRS system that can detect and recognize characters on license plates using a step-by-step structured image processing pipeline. The developed system combines classical image processing techniques such as grayscale conversion, Canny edge detection, contour extraction, adaptive thresholding, and morphological operations to accurately isolate the plate region even under varying lighting conditions and image qualities.

The most significant contribution of this study is the character recognition mechanism based on template matching, which does not require any training. Character segmentation is performed using connected component analysis, and each segmented character is compared with predefined templates in BMP format. This method provides a simple yet effective solution, as it is fast and does not require model training.

The system was developed using Python, with the help of OpenCV and NumPy libraries. Each image processing step is visually presented to the user step-by-step, allowing transparent monitoring of the system. The results demonstrate that license plate recognition can be successfully achieved without deep learning, by applying suitable thresholding and filtering parameters.

1.1 Literature Review

Automatic License Plate Recognition Systems (ALPRS) have become widely used in recent years in the fields of security and traffic management, with the success of these systems being directly dependent on the image processing and character recognition methods employed. In the literature, LPRS solutions typically consist of three core stages: license plate detection, character segmentation, and character recognition. This three-stage structure is common across both traditional image processing-based systems and AI-based systems.

In a study published by IEEE, Zheng et al. (2016) [1] proposed a Line Density Filter (LDF) for plate detection, which combined an enhanced Sobel operator with adaptive thresholding to achieve high accuracy. Additionally, a cascaded SVM classifier (CLPC) was used for color-based classification, identifying different plate colors to improve overall accuracy. This study demonstrated the potential of classical image processing techniques by striking an effective balance between speed and accuracy for real-time systems.

Similarly, in the system developed by Göde & Doğan (2023) [2], a low-cost and practical approach was implemented on a Raspberry Pi. The method involved grayscale conversion, bilateral filtering, Canny edge detection, and contour extraction, followed by adaptive thresholding and character recognition using Tesseract OCR. A notable contribution of this system is its support for language-independent character recognition and its ability to achieve high accuracy (91.82%) without requiring any training data.

AI-powered LPRS systems typically utilize neural network-based character recognition models. For example, in the study by Zhang et al. (2019) [3], license plate characters were recognized using machine learning algorithms, achieving 100% accuracy on a dataset of 172 plate images. In another study by Lee et al. (2018) [4], video-based license plate recognition was performed using a CNN architecture, with a reported success rate of 97%. However, a common drawback of these methods is their reliance on large training datasets and high computational resources.

In this context, the system developed in this study aims to eliminate the need for training and demonstrate that this problem can be solved solely with image processing techniques. The image processing pipeline includes commonly used methods from the literature, such as bilateral filtering, Canny edge detection, contour extraction, and thresholding. The character recognition stage stands out as an alternative solution that does not require training, relying instead on template matching, making it a lightweight and accessible approach to the LPRS problem.

2. Image Preprocessing

The first stage of image processing is the preprocessing phase. Images captured through cameras and various devices cannot be directly analyzed, as they often contain noise, poor lighting, and other unfavorable conditions. Therefore, various filters are applied to enhance the source image and eliminate these issues. The preprocessing stage includes techniques such as color space conversions, scaling, histogram operations, thresholding, filtering, and morphological transformations.

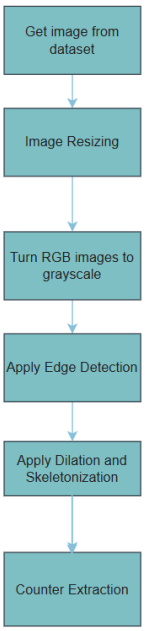

Figure 1. The basic steps are applied in image preprocessing

2.1 Contrast Equalization

Histogram equalization: Histogram equalization is one of the fundamental image enhancement techniques applied to make visual information more distinguishable in images with a low dynamic range. This method increases the contrast of an image by expanding or redistributing the intensity values more evenly across the

histogram. One of the most commonly used global contrast enhancement methods is Histogram Equalization.

The technique works by analyzing the histogram of intensity levels in the image and transforming it into a more uniform distribution. As a result, details that were previously indistinct become more visible. This method is particularly effective for dark or low-contrast images where intensity values are clustered within a narrow range.

Advantages:

- Enhances details in low-contrast images.
- Simple to apply and computationally efficient.
- Useful for applications such as satellite imagery.

Disadvantages:

- Since it is applied globally, it may cause over-brightening or darkening in certain regions of the image.
- In noisy images, it may amplify the noise and introduce artifacts.
- May result in information loss in areas where fine details are crucial.

Figure 2-3. Implamantion of histogram equalization and adaptive histogram equalization

Contrast Limited Adaptive Histogram Equalization (CLAHE): Contrast Limited Adaptive Histogram Equalization (CLAHE) was developed to overcome the disadvantages of global histogram equalization. Instead of applying contrast enhancement to the entire image at once, CLAHE operates on small, localized regions called tiles. By doing so, it preserves local details and achieves balanced contrast enhancement across different parts of the image. This localized approach prevents over-amplification of noise and excessive brightness or darkness in specific areas, resulting in a more natural and visually pleasing output.

CLAHE divides the image into small regions (tiles) and applies histogram equalization to each region independently. However, to prevent over-enhancement of contrast and noise amplification, a contrast limit (clip limit) is set during this process. This limit restricts the maximum amplification of pixel intensities. After enhancing each region, the neighboring tiles are then smoothly blended to avoid artificial boundaries and ensure a seamless transition across the entire image.

Advantages:

- Enhances local contrast, making small details more visible.
- A contrast limit can be set to prevent excessive noise amplification.
- Highly effective in applications where fine details are critical, such as medical imaging, face recognition, and license plate recognition.
- Minimizes the information loss commonly caused by global histogram equalization.

Disadvantages:

- Parameter selection (e.g., tile size and clip limit) directly affects the result and must be carefully optimized.
- Processing time is longer compared to global histogram equalization.
- If not properly tuned, artificial transitions or borders may appear in the image.

2.2 Low-Pass Filters (Blurring)

In license plate recognition systems, blurring (smoothing) operations are crucial for reducing noise in the image and making the license plate more distinguishable. Low-pass filtering suppresses sudden intensity changes and noise, improving the effectiveness of subsequent steps such as character segmentation and

edge detection. In this study, mean, median, Gaussian, and bilateral filtering methods were used to enhance the clarity of the plate region.

Median Filtering: The median filter is particularly effective for removing random noise such as "salt-and-pepper noise" commonly found in license plate images. Each pixel is replaced with the median value of the surrounding pixels within a defined window.

License plate images captured by traffic cameras or in outdoor environments may suffer from sudden light flares or sensor-induced noise. Median filtering helps eliminate such noise while keeping the characters sharp and intact. The median filter can be represented as follows:

$$f(x,y) = \min_{(s,t)\in S_{xy}} \{g(s,t)\}$$

Figure 4.Formula of median filtering

Figure 5.Implemantion of median filtering

Mean Filtering: The mean filter recalculates the center pixel by taking the arithmetic average of the pixel values within a defined neighborhood. It is a linear filtering method and is commonly used to remove low-level noise.This filter helps make the overall appearance of the license plate image smoother and more uniform. If there are random distortions on the characters, they can be reduced using this filter. However, because it has weak edge-preserving capabilities, it should not be applied too aggressively.

$$\hat{f}(x,y) = \frac{1}{mn} \sum_{(s,t)\in S_{xy}} g(s,t)$$

Figure 6.Formula of mean filtering

Figure 7.Implemantion of mean filtering

Gaussian Filtering: The Gaussian filter is a two-dimensional (2D) blurring filter used to smooth images and reduce noise. It applies a weighted average to each pixel based on the Gaussian function, giving more importance to pixels closer to the center. The smoothing effect is controlled by the parameter σ (sigma); as sigma increases, the level of blurring also increases.

The Gaussian filter is defined as:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Figure 8.Formula of gaussian filtering

Figure 9.Implemantion of gaussian filtering

Bilateral Filtering: Bilateral filtering is a nonlinear filter that reduces noise in an image while preserving edges. Unlike traditional smoothing methods, it considers both spatial proximity and pixel intensity differences. This dual consideration allows the filter to smooth flat regions while maintaining sharp edges, preventing edge blurring.

It is particularly effective for objects with well-defined boundaries, such as license plates. By combining spatial and intensity information, bilateral filtering enhances the image without distorting important structural features, making it ideal for preprocessing in recognition tasks.

Figure 10.Implemantion of bilateral filtering

2.3 High-Pass Filters (Edge Detection)

High-pass filters are used to reveal abrupt changes in an image — namely, the edges. Edge detection is one of the most critical stages in license plate recognition systems, as these filters enable the identification of the character boundaries on the plate.

In this study, Sobel, Canny, and Laplacian edge detection methods were applied. The goal is to identify the license plate frame within the image.

Sobel Filtering: The Sobel filter is a gradient-based method used to detect edges in both the horizontal (x) and vertical (y) directions of an image. It responds strongly to sudden changes in pixel intensity.

Sobel filtering is particularly effective for detecting the directional edges of characters on license plates — especially horizontal lines — making it a valuable tool for segmenting character regions.

Figure 11.Implemantion of sobel filtering

Laplacian Filtering: The Laplacian filter is a second-derivative-based edge detection method used to highlight and sharpen edges in an image. It detects areas with rapid intensity changes, emphasizing contour lines and object boundaries.

Unlike gradient-based filters like Sobel, which detect edges in specific directions, the Laplacian responds equally to changes in all directions, making it a direction-independent operator.

A commonly used kernel for the Laplacian filter is:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Figure 12.Formula of laplacian filtering

This kernel simultaneously evaluates changes in both the x and y directions.

Figure 13.Implemantion of laplacian filtering

Canny Edge Detection: The Canny edge detector is a multi-stage algorithm that includes steps such as noise reduction, gradient calculation, edge suppression, and double thresholding. It is considered one of the most stable and accurate edge detection methods.

Thanks to the Canny filter, the boundaries of the license plate frame and characters can be clearly separated. It is especially effective when the background is complex, as it helps isolate the character edges. The method is applied in five key steps:

- **Smoothing:** The image is smoothed using a Gaussian filter to reduce noise.
- **Gradient Calculation:** The gradient direction and magnitude are computed to detect edge strength and orientation.
- **Non-Maximum Suppression:** Non-maximum pixels in the gradient direction are suppressed to thin the edges.
- **Double Thresholding:** Weak and strong edges are identified by applying two threshold values.
- **Edge Tracking by Hysteresis:** Weak edges connected to strong ones are retained, while others are discarded, producing the final edge map.

Figure 14.Implemantion of canny edge detection

2.4 Thresholding Operations

Thresholding is a commonly used method for converting grayscale images into binary (black-and-white) images. This process enables the clear separation of objects from the background by distinguishing pixels based on whether their intensity values are above or below a certain threshold.

Thresholding methods are generally categorized into two types: **static (fixed)** and **dynamic (adaptive)**:

- **Static Thresholding:** A fixed threshold value is manually set by the user and applied uniformly across the image.
- **Dynamic Thresholding (Adaptive or Automatic):** The threshold value is automatically calculated by the algorithm based on the content of the image, allowing for more accurate results in images with varying lighting conditions.
-

1.1.1. Static Thresholding (Global Thresholding)

Static thresholding uses a **single fixed threshold value (T)** for the entire image. Every pixel is compared to this value: if the pixel value is higher than the threshold, it's set to a maximum value (e.g., 255); otherwise, it's set to 0. This method is simple, fast, and effective for images with **uniform lighting**. However, it performs poorly when the image has varying brightness or shadows.

Binary: Binary thresholding assigns each pixel either a maximum value (e.g., 255) if it is greater than a threshold T, or 0 otherwise. It is commonly used to separate the foreground from the background in an image.

$$dst(x,y) = \begin{cases} maxValue & \text{if } src(x,y) > T(x,y) \\ 0 & \text{otherwise} \end{cases}$$

Figure 15.Formula of binary thresholding

Binary Inv: Binary inverse thresholding is the opposite of binary thresholding; pixels greater than the threshold are set to 0, while those less than or equal to the threshold are set to the maximum value.

$$dst(x,y) = \begin{cases} 0 & \text{if } src(x,y) > T(x,y) \\ maxValue & \text{otherwise} \end{cases}$$

Figure 16.Formula of binary inverse thresholding

Trunc: Truncation thresholding sets all pixel values greater than the threshold T to T, while leaving other pixel values unchanged. It is used to flatten out brighter regions without affecting darker areas.

$$\texttt{dst}(x,y) = \begin{cases} \texttt{threshold} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$$

Figure 17.Formula of truncation thresholding

ToZero: ToZero thresholding keeps pixel values that are greater than the threshold T, and sets all others to 0. It is useful for retaining only the bright regions in an image.

$$\texttt{dst}(x,y) = \begin{cases} \texttt{src}(x,y) & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$$

Figure 18.Formula of tozero thresholding

ToZero Inv: ToZero inverse thresholding is the inverse of ToZero; it keeps pixel values less than or equal to the threshold and sets the rest to 0. It is useful for isolating dark regions.

$$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$$

Figure 19.Formula of tozero inverse thresholding

Otsu: The Otsu thresholding method separates image pixels into two distinct classes based on their intensity distribution: foreground (object) and background. In this method, the optimal threshold value is automatically determined by maximizing the **between-class variance**, ensuring the best possible separation.

Otsu's method is frequently used in images with variable lighting conditions due to its robustness and consistent performance.

The method works by calculating the pixel proportions and corresponding mean values for each gray level, aiming to minimize the **within-class variance** or, equivalently, to maximize the **between-class variance**.

The key formulas are:

$$\text{Class}(t) = \sum_{i=0}^{t} p(i), \qquad \omega_1 \sum_{i=+1} p(i)$$

Class means:

$$\mu_0(t) = \frac{1}{\omega_0} \sum_{i=0} p(i), \quad \frac{1}{\omega_{1(t}} i \cdot p(i)$$

$$\mu_1(t) = \frac{1}{\omega_0 t} \sum_{i=t+} p(i), \frac{1\text{-}1}{\omega_{1(t}} i \cdot p(i)$$

Between-class variance:

$$\sigma_b^2(t) = \omega_0(t) \cdot \omega_1(t) \cdot [\mu_0(t) - \mu_1(t)]^2$$

Figure 20.Formulas of otsu

1.1.2. **Dynamic Thresholding (Adaptive Thresholding)**

Dynamic thresholding (also called adaptive thresholding) computes a different threshold value for each pixel based on its local neighborhood. It is especially useful for images with non-uniform lighting, shadows, or gradients. The threshold is usually determined by the mean or weighted average (Gaussian) of the surrounding pixels. This method is slower but much more robust in real-world scenarios.

Adaptive Mean: Adaptive mean thresholding calculates a threshold for each pixel based on the mean of its neighboring pixels within a defined window. It is effective for images with varying illumination.

$$f(x,y) = \begin{cases} \text{maxVal}, & \text{if } g(x,y) > \text{mn1} \sum_{(s,t) \in S_{xy}} g(s,t) - C \\ 0, & \text{otherwise} \end{cases}$$

Figure 21.Formula of dynamic thresholding

Adaptive Gaussian: Adaptive Gaussian thresholding determines a pixel's threshold using a weighted sum (Gaussian kernel) of its neighbors. It offers smoother and

more localized results in non-uniform lighting conditions.

$$f(x,y) = \begin{cases} \text{maxVal}, & \text{if } g(x,y) > \sum_{(s,t)\in S_{xy}} G(s,t)\cdot g(s,t) - C \\ 0, & \text{otherwise} \end{cases}$$

Figure 22.Formula of adaptive gaussian

Figure 23.Implemantion of thresholdings

2.5 Morphology operations

Morphology is defined as the science of shapes. In image processing, morphological operations are techniques developed based on set theory and applied according to the geometric structures of objects within an image. These operations are used to distinguish objects from the background, extract desired structures, reduce noise, and perform segmentation. Morphological filtering can be applied at the beginning or end of an image processing pipeline. These methods can be used on both **binary** and **grayscale** images.

The following operations describe the main purposes of morphological processing:

- **Dilation:** Expands the boundaries of objects, filling small holes and connecting nearby elements.
- **Erosion:** Shrinks object boundaries, removing small noise and isolating objects.
- **Opening:** Erosion followed by dilation. It removes small objects or noise while preserving the shape and size of larger structures.
- **Closing:** Dilation followed by erosion. It fills small gaps and holes within objects without significantly altering their shape.
- **Morphological Gradient:** Highlights object edges by calculating the difference between dilation and erosion.

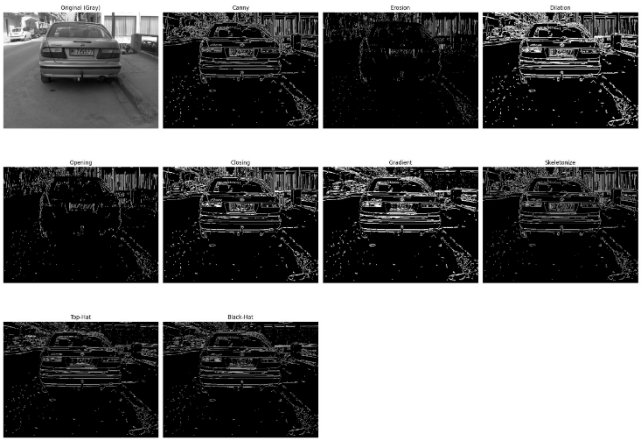

Figure 24 .Implemantions of morphology operations in canny edge detection

3. Chracter Recognition

Character recognition is one of the final stages in the image processing pipeline and enables the conversion of textual content from visual data into digital form. At this stage, Optical Character Recognition (OCR) engines are commonly used to make text from various sources—such as PDF documents, books, printed materials, and images—readable and editable by computers [55]. Popular OCR solutions include Pytesseract, Microsoft Office Document Imaging, OmniPage, and FineReader.
However, in this study, instead of conventional OCR engines, a simpler, more controllable, and training-free method—template matching—was chosen. In this approach, each character is compared against a set of predefined template images of fixed size (e.g., templates prepared in .bmp format).
The region containing the character is first detected using connected component analysis, then resized to match the template size. The cv2.matchTemplate() function is used to compute a similarity score between the target character region and each template. The character corresponding to the highest match score is selected as the recognized character.

This method:

- Does not require any machine learning or deep learning models,
- Is suitable for working with customizable character sets,

- Can be used efficiently in embedded systems with small datasets and low computational power.

This allows the characters on the license plate to be quickly and effectively identified, resulting in textual output from the system. This approach is particularly effective in license plate systems based on the Latin alphabet, yielding highly successful results.

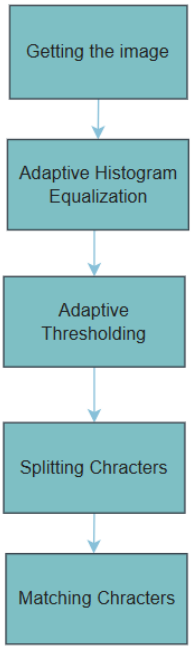

Figure 25. The basic steps are applied in chracter recognition

4. Experimental Study

In this study, vehicle license plates were detected and their characters recognized using only image processing techniques. A total of 69 different vehicle images were used, derived from 50 distinct images provided for the study. Using these visuals, a License Plate Recognition System (LPRS) was developed.

The developed LPRS consists of three main stages:

1. **License Plate Region Detection:**In this stage, the input image was processed sequentially using grayscale conversion, Canny edge detection, and contour extraction. The resulting contours were evaluated based on geometric ratios and angle constraints to select the most likely license plate candidates from among the detected regions.
2. **Cropping of Plate Regions:**
The detected plate regions were isolated from the original image using masking and rectangular cropping. This ensured that only the license plate area was passed on to subsequent processing steps.
3. **Character Recognition:**
Instead of commonly used OCR solutions, this study employed a template matching method, which requires no training data and is more efficient in terms of system resources. Character regions within the plate were identified using connected component analysis**,** each character was resized to a fixed dimension, and then compared against predefined character templates using the cv2.matchTemplate() function. The template with the highest similarity score was considered the recognized character.
Thanks to this method, successful character recognition results were achieved even in systems with low resolution and limited data.

Step 1: Grayscale conversion and Resize image

The first step in the license plate recognition process is converting the image to grayscale. Colored images consist of red, green, and blue (RGB) components; however, in tasks such as character recognition, this color information is often unnecessary. Instead, working with a grayscale image—containing only intensity information—not only reduces computational load but also provides more consistent results in subsequent steps.

Grayscale conversion is performed by calculating the brightness (intensity) value for each pixel in the color image. This transformation creates a more suitable foundation for operations such as contour extraction, thresholding, and character segmentation.

After converting to grayscale, the image is resized. This ensures that all vehicle images are processed at a consistent fixed resolution within the system. In cases of low-resolution or small license plate images, resizing helps make the characters more visible and improves the

success of character segmentation via connected component analysis. Additionally, since the characters will later be compared with templates using the template matching method, standardizing their size to match the templates is critically important.

With these two fundamental preprocessing steps, the system operates with reduced noise and provides high-quality input for the character recognition stage.

Figure 26. Implemantion of Grayscaling

Step 2: Canny edge detection

After the image was converted to grayscale, various filtering methods—such as median blur, bilateral filtering, histogram equalization, and CLAHE—were tested for enhancement prior to edge detection. However, in the experiments, none of these filters produced results as successful as the original grayscale image. Therefore, the **Canny edge detection** was applied directly on the grayscale image, which resulted in sharper and more accurate edge detection.

Figure 27. Implemantion of cannyEdge

Step 3: Morphological Operations (Dilation and Skeletonize)

After the edge detection step, morphological operations were applied to enhance the continuity of the license plate boundary. Initially, a **horizontal dilation** was performed to connect the unmerged contours. A kernel with a height of 1 pixel and a width of 2 pixels was used specifically to promote horizontal growth, and after experimenting with different kernel structures, this configuration yielded the most optimal results.

Figure 28. Implemantion of dilation

Subsequently, in order to **eliminate unwanted vertical connections** and preserve the essential contour structure, the **skeletonize** method was employed. Although **erosion** was also tested as an alternative, it resulted in undesired disconnections in critical regions. Therefore, in this study, a combination of horizontal dilation followed by skeletonization was applied as the most effective morphological approach to refine the edge structure.

Figure 29. Implemantion of skeletonization

Step 4: Contour extraction

After the completion of the morphological operations, all detected contours were visualized on the color image to facilitate the interpretation of their shapes and locations.

Each contour was evaluated based on specific geometric criteria, including **area size**, **aspect ratio**, and **angle of orientation** of the bounding rectangle.

Through this filtering process, candidate regions likely to contain license plates were identified and enclosed within **bounding boxes**. This approach enabled effective selection of potential plate regions from among all detected contours.

Figure 29. Implemantion of contour extraction

Step 5: ROI (Region of Interest) Cropped and Resized

Candidate license plate regions (ROIs), obtained through contour analysis, were cropped directly from the grayscale image. Each ROI was then evaluated in terms of its size, and regions below a predefined threshold were **resized by a factor of 2.5**. This resizing step was crucial to improve character readability in low-resolution plate images. Without this enlargement, characters in very small ROIs often lacked sufficient visual detail, leading to poor recognition performance during the **template matching** process.

Figure 30. Implemantion of ROI

Step 6: Contrast Enhancement

To enhance the local contrast within the Region of Interest (ROI) and to ensure better separation of characters from the background, the **CLAHE (Contrast Limited Adaptive Histogram Equalization)** technique was applied. This method significantly improves character visibility, particularly in regions with uneven lighting, thereby contributing to more accurate results in the subsequent thresholding and recognition stages.

Figure 31. Implemantion of contrast enhancment

Step 7: Thresholding

Following the application of CLAHE, **adaptive thresholding** was applied to distinguish characters from the background. This process produced a binarized image that is structurally suitable for **connected component analysis**, enabling more accurate and stable segmentation of individual characters.

Figure 32. Implemantion of Thresholding

Step 8: Component Analysis

Connected component analysis was applied to the binarized image to identify distinct regions that may represent individual characters. This analysis treats each cluster of foreground pixels as a separate component, allowing the extraction of positional and dimensional information for potential character candidates.

Figure 33. Implemantion of Component Analysis

Step 9: Template Matching

After the character candidates were identified, each component was resized to a fixed dimension and compared with a set of predefined character templates. In this stage, the **template matching** technique was applied to each candidate using the cv2.matchTemplate() function. Among the calculated similarity scores, the template with the highest value was selected as the

recognized character. This approach enabled character recognition to be performed **without the need for any training data**, relying solely on direct visual comparison.

The template images used for matching are shown in **Figure 34**. These templates were prepared in .bmp format, with each representing a single character at a standardized size.

Figure 34. Templates

Step 10: License Plate Recognition

Following successful character recognition, the detected license plate and its corresponding text were visualized on the original color image. For this purpose, the plate region was highlighted with a **bounding rectangle**, and the recognized character string was overlaid above the detected area. This visualization allowed the system output to be presented clearly to the user while also enabling visual verification of the recognition accuracy.

Figure 35. License Plate Recognition example

5. Conclusions

In this study, a license plate recognition system was developed using only classical image processing techniques. The system detects license plate regions through grayscale conversion, edge detection, morphological operations, and contour analysis. Character recognition is performed via template matching without the need for any training data.

Experimental evaluations were conducted using standard performance metrics. In the license plate detection phase, the system achieved 60 true positives (TP), 234 false positives (FP), and 22 false negatives (FN), resulting in:

Precision: 0.204 Recall: 0.732 F1-score: 0.318

In an alternative approach where only candidates containing text were considered as valid plates, the results improved significantly, yielding 43 TP, 1 FP, and 22 FN:

Precision: 0.977 Recall: 0.662 F1-score: 0.789

In the character recognition stage, the system achieved 3 TP, 40 FP, and 10 FN:

Precision: 0.0698 Recall: 0.2308 F1-score: 0.1072

These results indicate that the proposed system offers a lightweight and training-free solution suitable for low-resource environments. The filtering strategy based on the presence of textual content in candidates significantly enhanced detection performance. However, limitations of the template matching method in character recognition were also observed. Future work will focus on comparing this module with deep learning-based approaches to improve recognition, accuracy and robustness.

.

6. References

1. Zheng, L., Zhao, Y., Zheng, Z., & Shen, M. (2016). A robust and efficient approach to license plate detection. *IEEE Transactions on Image Processing*.
2. Göde, A., & Doğan, A. (2023). License Plate Recognition System Based on Artificial Intelligence with Different Approach. *El-Cezerî Journal of Science and Engineering*, 10(1), 121-136.
3. Koval, O., et al. (2012). A New License Plate Recognition System Based on Probabilistic Neural Networks. *Procedia Engineering*, 39, 986-993.
4. Saini, R., & Saini, M. (2020). A Novel Method for Indian Vehicle Registration Number Plate Detection and Recognition. *Procedia Computer Science*, 167, 2401-2410