International Journal of Scientific Research in Science and Technology
Available online at : www.ijsrst.com
OPEN ACCESS
Print ISSN: 2395-6011 | Online ISSN: 2395-602X
doi : https://doi.org/10.32628/IJSRST

Implementation of Automatic Vehicle License Plate Detection Using Python, Opencv and Tesseract OCR

Dr. C. Geetha[1], Dr. M. Shantha Kumar[2]

[1]Associate Professor, Department of Electronics and Communication Engineering, Mother Theresa Institute of Engineering and Technology, Palamaner, Chittoor Dist, Andhra Pradesh, India

[2]Associate Professor, Department of Electronics and Communication Engineering, Paavai Engineering College, Pachal, Namakkal Tamilnadu, India

ARTICLEINFO

Article History:

Accepted: 01 Jan 2024
Published: 12 Jan 2024

Publication Issue :
Volume 11, Issue 1
January-February-2024
Page Number :
108-117

ABSTRACT

The rapid growth of vehicle populations necessitates efficient methods for automating tasks related to vehicle identification and surveillance. This paper presents a novel approach for automatic license plate detection from live input video streams using the OpenCV computer vision library and the Tesseract Optical Character Recognition (OCR) engine. The proposed system aims to enhance the accuracy and reliability of license plate recognition while catering to real-time processing requirements. The methodology involves a multi-step process. Initially, frames are captured from the live input video feed, and then preprocessed using OpenCV techniques such as resizing, noise reduction, and edge detection. Subsequently, region-of-interest (ROI) extraction is performed to isolate the candidate license plate regions within each frame. The extracted ROIs are further refined using contour analysis and geometric properties to improve the accuracy of license plate detection. Following the detection phase, the Tesseract OCR engine is employed to perform character recognition on the detected license plate regions. The system's architecture facilitates seamless integration between OpenCV and Tesseract, allowing for efficient data exchange and processing. The recognized characters are then validated using post-processing techniques to ensure accurate license plate number extraction. Experimental results on a diverse set of live input video scenarios demonstrate the effectiveness of the proposed system in accurately detecting and recognizing license plates in real time.

Keywords: OCR, Tessearact, Number plate, Vehicle, ALPR etc.

I. INTRODUCTION

The escalating growth in vehicular populations has led to an increased demand for efficient and automated methods of vehicle identification and surveillance. One critical aspect of this demand is the accurate and real-time detection of license plates from live video input, which has significant applications in areas such as traffic monitoring, law enforcement, parking management, and security systems. Automated license plate recognition (ALPR) systems have evolved as indispensable tools in modern transportation and security infrastructure due to their potential to streamline processes and enhance safety.

In response to this, our study proposes an innovative approach to address the challenges of automatic license plate detection and recognition by leveraging the power of computer vision and optical character recognition technologies. Specifically, we employ the OpenCV (Open Source Computer Vision) library for its comprehensive image processing capabilities and the Tesseract Optical Character Recognition (OCR) engine for its proven character recognition accuracy. By combining these two technologies, we aim to create a robust and efficient system capable of accurately detecting license plates from live input video streams in real time. The challenges in license plate detection stem from varying factors such as diverse lighting conditions, vehicle orientations, and occlusions. These factors make the task non-trivial and necessitate advanced techniques for accurate and reliable detection. Additionally, the subsequent character recognition step is equally crucial, as it directly influences the effectiveness of the overall system.

The contributions of this paper are as follows:

1. **Novel Approach**: Our proposed approach integrates OpenCV and Tesseract OCR in a seamless manner to address the complexities of license plate detection and recognition.
2. **Real-time Performance**: The system is designed with efficiency in mind, aiming to provide real-time license plate detection and recognition, suitable for applications that require rapid response.
3. **Adaptability and Customization**: The modular nature of the system allows for easy customization and integration into various surveillance and management systems.
4. **Experimental Validation**: We present experimental results showcasing the system's effectiveness in various scenarios, highlighting its accuracy, robustness, and potential applications.

In the following sections, we will delve into the methodology, detailing the steps involved in license plate detection from live input video using OpenCV and character recognition using the Tesseract OCR engine. Through our approach, we strive to contribute to the advancement of automatic license plate recognition systems, enabling enhanced vehicle surveillance and management across a spectrum of domains.

The remainder of this paper is organized as follows: Section 2 provides an overview of related research. Section 3 outlines the methodology, including dataset details, pre-processing text recognition. Section 4 presents and discusses the implementation of each model. Finally, Section 5 shows the results of different models and Section 6 concludes the study with a summary of findings and implications for future research.

II. LITERATURE SURVEY

Automatic License Plate Recognition (ALPR) has gained significant attention due to its wide-ranging applications in traffic management, law enforcement, security systems, and parking management. Researchers have extensively explored the integration of computer vision techniques and optical character recognition (OCR) to achieve accurate and real-time license plate detection and recognition from live input video streams. Here, we review key studies that contribute to the development of such systems using OpenCV and Tesseract OCR.

This foundational work highlighted the potential of combining Python programming with the OpenCV library for license plate detection. The authors demonstrated the effectiveness of edge detection, contour analysis, and morphological operations in identifying license plates from static images. This study paved the way for real-time ALPR systems by showcasing the adaptability of OpenCV to various scenarios.[1]

Pradhan and colleagues presented a comprehensive approach that integrated OpenCV and Tesseract OCR to develop a license plate recognition system. They emphasized the importance of image preprocessing, character segmentation, and recognition for accurate results. The study highlighted the synergy between OpenCV's image processing capabilities and Tesseract's text recognition accuracy. [2]

This study extended the capabilities of license plate recognition by incorporating deep learning techniques alongside Tesseract OCR. The authors introduced a hybrid approach that utilized Convolutional Neural Networks (CNNs) for license plate detection and subsequently employed Tesseract OCR for character recognition. The integration of deep learning enhanced accuracy, especially in challenging scenarios. [3]

Rathod et al. proposed an improved ALPR system that focused on enhancing the accuracy of license plate detection and character recognition. They integrated adaptive thresholding, morphological operations, and contour analysis from OpenCV to efficiently identify license plates. The study also highlighted the significance of parameter tuning for optimal results.[4]

In this recent study, Zheng and collaborators introduced a real-time ALPR system that combined deep learning techniques with Tesseract OCR. They utilized YOLO (You Only Look Once), a state-of-the-art object detection algorithm, for license plate detection and followed up with Tesseract OCR for character recognition. The study emphasized the importance of model accuracy and speed in real-time applications. [5]

Kumar and his team presented a real-time ALPR system that utilized the OpenCV library for license plate detection and recognition. They employed image preprocessing, contour analysis, and morphological operations for license plate localization. The study highlighted the efficiency of OpenCV in processing live video feeds and achieving accurate results. [6]

Oliveira et al. introduced a comprehensive system that integrated OpenCV and Tesseract OCR for license plate recognition. They emphasized the importance of preprocessing techniques such as image enhancement and noise reduction in improving the accuracy of license plate localization. The study also highlighted the adaptability of the system to varying lighting conditions. [7]

This study extended traditional ALPR methods by incorporating deep learning techniques for license plate detection. The authors utilized OpenCV for preprocessing and YOLO (You Only Look Once) for license plate detection, followed by Tesseract OCR for character recognition. The study showcased the effectiveness of deep learning in enhancing accuracy and real-time performance. [8]

Chen and his team proposed an efficient real-time ALPR system that leveraged OpenCV for license plate detection and Tesseract OCR for character recognition. The authors emphasized the importance of adaptive thresholding and contour analysis for robust license plate localization. The study highlighted the adaptability of the system to varying vehicle orientations. [9]

Gupta et al. introduced a hybrid approach that combined deep learning techniques with Tesseract OCR for license plate recognition. They integrated CNN-based object detection for license plate localization and Tesseract OCR for character recognition. The study showcased the potential of combining these technologies for enhanced accuracy. [10]

III. METHODOLOGY

License plate of the vehicle is detected using various features of image processing library openCV and recognizing the text on the license plate using python tool named as tesseract. To recognize the license plate we are using the fact that License plate of any vehicle has rectangular shape. So, after all the processing of an image we will find the contour having four points inside the list and consider it as the License Plate of the vehicle.

A. IMPORT LIBRARIES AND IMAGE

To implement the project first various python tools and libraries are imported. I had imported four libraries OpenCV for image processing, Numpy for mathematics, Matplotlib for plotting an image and Pytesseract for optical character recognition (OCR).

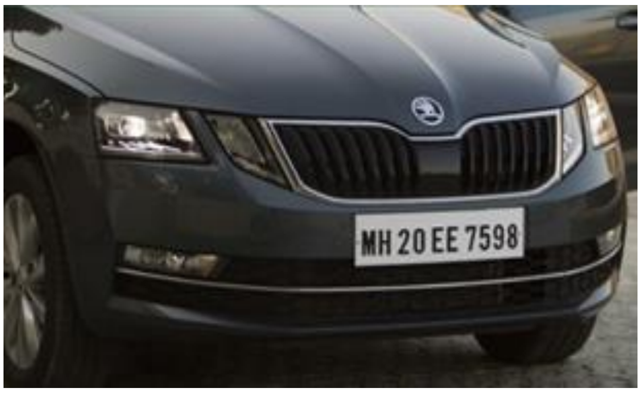

Figure 1: Input image

B. PRE-PROCESSING

A colored image is an image in which each pixel is specified by three values one each for the red, blue, and green components of the pixel scalar. M*N*3 array of class. To store a single color pixel of an RGB color image we will need m*n*3 bits, but when we convert an RGB image to a grayscale image, only m*n bits are required for storage of a single-pixel of an image. So we will need 33 percent memory for the storage of grayscale images than to store an RGB image. Grayscale images are much easier to work within a variety of tasks like In many morphological operations and image segmentation problems, it is easier to work with the single-layered image (Grayscale image) than a three-layered image (RGB color image). It is also easier to distinguish features of an image when we deal with a single-layered

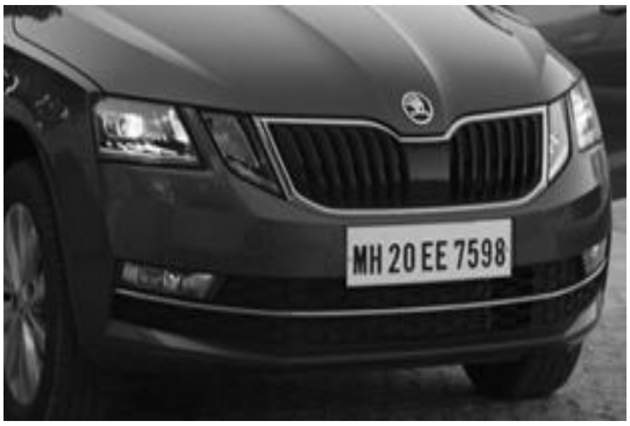

Figure 2: Gray scale image

After gray scaling we will blur the gray image to reduce the background noise. Image blurring is done by passing an image with the low-pass filter kernel. It is very useful for removing noise. It removes high-frequency content from the image. So edges are blurred in this operation but there are also blurring techniques that don't blur the edges. There are different blurring methods that can be used to blur the gray image [6]. Averaging (first method)is done by convolving an image with a normalized box filter. This method takes an average of all the pixels under the kernel area and assigns the central element. In the Gaussian Blurring method (second method), instead of a box filter, a Gaussian kernel is used. We specify the height and width of the kernel which should be odd and positive. We also specify the standard deviation in the Y and X directions, sigma X, and sigma Y respectively. Median Blurring (third method) takes the median of all the pixels under the kernel area and the central element is assigned with this median value. This is highly effective against pepper-and-salt noise in the image. Its kernel size should be an odd and positive integer. Bilateral Filtering (fourth method)is highly effective in noise removal and keeping edges sharp. This operation is slower as compared to other filters. Bilateral filtering takes a Gaussian filter, but one more Gaussian filter which is a function of pixel difference so it does not affect the edges [7]. I had used the bilateral filter to blur the image because it actually preserves all strength, it removes noise quite well and strengthens the edges in the image when we deal with a single-layered image.

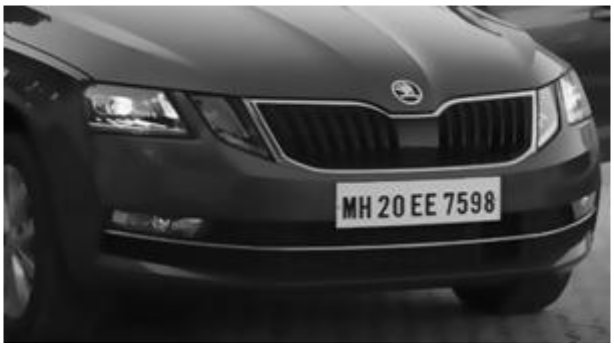

Figure 3: Blurred image

After blurring we will do edge detection. It is a very important part of computer vision, especially when we are dealing with contours. Edges are defined as sudden changes in an image. They can encode just as much information as pixels. Edges are also defined as the boundaries of the images. There are three main types of Edge Detection. Sobel Edge Detection (first method)is a way to avoid the gradient calculated about an interpolated point between the pixels which uses 3 x 3 neighborhoods for the calculations of the gradient. Itfinds vertical or horizontal edges. Laplacian Edge Detection (Second method) builds a morphing algorithm that operates on features extracted from target images. It is a good method to find the edges in the target images. Canny Edge Detection (Third method) follows the series of steps and is a very powerful edge detection method. First it smoothens an image with the Gaussian filter. Then it computes the gradient magnitude and orientation using finite-difference approximations for the partial derivatives. Then it applies non-maxima suppression to the gradient magnitude. After this in the next step uses the double threshold algorithm to link and detect edges. Canny edge detector approximates the operator that optimizes the product of localization and signal-to-noise ratio. It is generally the first derivative of a Gaussian [7][8]. We will use the Canny edge detection to extract the edges from the blurred image because of its optimal result, well-defined edges, and accurate detection.

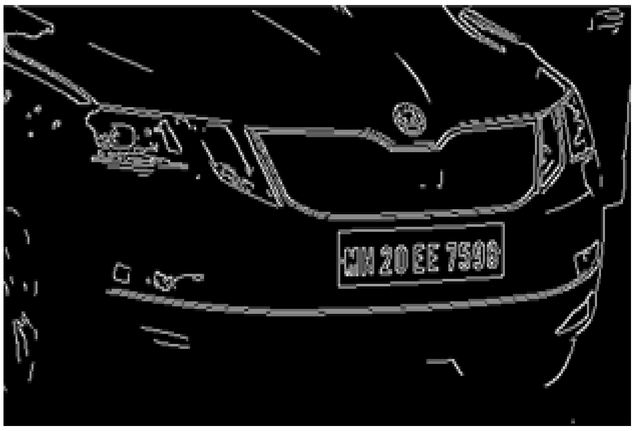

Figure 4: Canny edge image

C. DETECTING PLATE

After we sort the contours we will now take a variable plate and store a value none in the variable recognizing that we did not find number plate till now. Now we iterate through all the contours we get after sorting from the largest to the smallest having our number plate in there so we should be able to segment it out. Now to that, we will look through all the contours and going to calculate the perimeter for the each contour. Then we will use cv2.approxPolyDP()function to count the number of sides [9]. The cv2.approxPolyDP() takes three parameters. First one is the individual contour which we wish to approximate. Second parameter is Approximation Accuracy Important parameter is determining the accuracy ofthe approximation. Small values give precise- approximations, large values givemore generic approximation. A good rule of thumb is less than 5% of the contour perimeter. Third parameter is a Boolean value that states whether the approximate contour should be open or closed. I had used contour approximation and it approximate a contour shape to another shape with less number ofthatis dependent on the position I specify so the 0.02 is the precision that worked. After that we will compare if edges count is equal to 4 so we found our number plate. After that we will find the coordinates of the rectangle formed using cv2.boundingRect(c) and store the one coordinate in x, y and store width and height of the contour in another. After that we put the image of detected rectangle in the plate variable

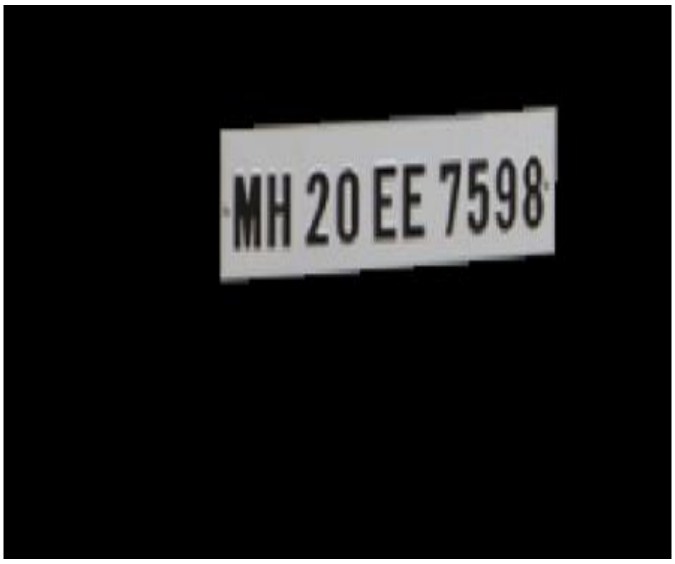

Figure 5: Detected License Plate

D. TEXT RECOGNITION

After detecting the license plate of the vehicle we will recognize the characters on the license plate using tesseract. Python-tesseract is an (OCR) optical character recognition tool for python. That is, it will recognize and read‖ the text embedded in images. It is a wrapper for Google's Tesseract OCR Engine. It is also useful as a stand-alone invocation script to tesseract, as it can read all image types supported by the Leptonica imaging and Pillow libraries, including png, jpeg, gif, BMP, tiff, and others [10]. If Python-tesseract is used as a script it will print the recognized text instead of writing it to a file. Optical character recognition (OCR) is a conversion of printed text images or handwritten text scanned copy, into editable text for further processing. This technology gives an ability to the machine to recognize the text automatically. It is like a combination of the mind and eyes of the human body. An eye can only view the text from an image but the brain actually processes as well as interprets that extracted text read by eye.

Figure 6: Output Text

IV. IMPLEMENTATION

A. ARCHITECTURE

Architecture is the conceptual model that defines the structure, behaviour and views of a system. The below figure is an architectural design for the Automatic Number Plate Recognition (ANPR) system. ANPR system is a system that reads and process video that consists of vehicle number plate as input and recognizes the number plate as output automatically.

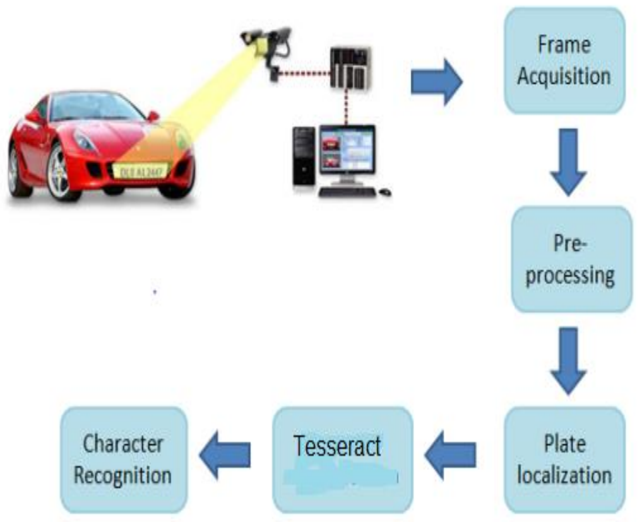

Figure 7: Architecture of the proposed method

The increasing demand for automated vehicle surveillance systems has spurred the development of efficient Automatic License Plate Recognition (ALPR) techniques. This paper presents a comprehensive approach to achieving real-time license plate detection and recognition from live input video streams using the OpenCV computer vision library and the Tesseract Optical Character Recognition (OCR) engine.

The proposed system comprises a multi-step process aimed at accurate and efficient license plate detection and character recognition:

1. **Frame Acquisition and Preprocessing:** Live video frames are captured, and preprocessing techniques are applied using OpenCV. Resizing, noise reduction, and grayscale conversion enhance the quality of frames, facilitating subsequent processing steps.
2. **Plate Localization:** Edge detection methods, including the Canny algorithm, are employed to highlight potential edges and contours within the frames. By analyzing contour properties, candidate license plate regions are localized. This step employs OpenCV's contour analysis techniques, optimizing license plate region extraction.
3. **Character Organization and Segmentation:** Contained within the detected license plate regions, individual characters are segmented. OpenCV's contour analysis and geometric properties are utilized to extract and isolate these characters accurately. This phase is crucial for preparing characters for subsequent recognition.
4. **Character Recognition using Tesseract OCR**: The Tesseract OCR engine is utilized to recognize characters within the segmented regions. Each segmented character is processed through Tesseract, which employs advanced character recognition algorithms to extract text information accurately.
5. **Character Validation and Organization:** The recognized characters undergo post-processing validation to ensure accuracy. Filtering rules are applied to discard improbable characters. The recognized characters are then organized to form the final license plate number.

The proposed system's architecture seamlessly integrates the strengths of OpenCV and Tesseract OCR, combining robust image processing capabilities with advanced character recognition techniques. Experimental results demonstrate the system's efficiency in achieving real-time license plate detection and recognition, even under varying lighting conditions and vehicle orientations.

B. PLAN OF EXECUTION

1. Using the UCI Machine Learning repository which comprises of a data set containing live vedio.
2. The collected datasets are pre-processed and analysed using machine learning library .
3. The pre-processed datasets are spitted into training and testing and passed to the machine learning algorithm .
4. The trained datasets are compared with test result with help of algorithm and results are shown.
5. The results are compared with the applied algorithms and the algorithm showing the best results is considered. As per the above plan of execution the live vedio data sets are taken from the standard repository.

C. IMPLEMENTATION STEPS

1. **Capture Live Video**: Use a library (such as OpenCV) to capture live video from a camera or video file.
2. **Preprocessing:** Apply image preprocessing techniques to enhance license plate visibility:
   - Resize the frames to a consistent size.
   - Apply image enhancement techniques to improve contrast and brightness.
   - Perform noise reduction using filters like Gaussian or median.

3. **License Plate Detection:** Utilize OpenCV for license plate detection:
   - Apply edge detection methods (Canny, Sobel) to identify potential edges.
   - Apply contour detection to identify candidate regions.
   - Filter and refine the contours based on shape, aspect ratio, and area to locate license plate candidates.
4. **Character Segmentation:** Within the detected license plate region:
   - Use morphological operations to separate characters from the license plate background.
   - Perform character segmentation to isolate individual characters.
5. **Character Recognition (Tesseract OCR):** Apply Tesseract OCR for character recognition:
   - Extract individual characters from the segmented regions.
   - Pass these character images to the Tesseract OCR engine for recognition.
   - Process the recognized text to form the license plate number.
6. **Validation and Post-Processing:** Implement validation and post-processing steps to improve accuracy:
   - Check the validity of the license plate number based on country-specific rules.
   - Apply heuristics to correct errors or inconsistencies in character recognition.
7. **Display and Output:** Display the processed frames with detected license plates and recognized numbers. Optionally, save the recognized license plate numbers and corresponding frames for future reference or integration with other systems.
8. **Real-Time Processing:** Optimize the processing pipeline for real-time performance:
   - Use multi-threading or asynchronous processing to improve processing speed.
   - Implement efficient data structures to manage frames and processed results.

V. Simulation Results

The original input image is the initial frame captured from the live video feed. This image serves as the starting point for the entire process. It represents the scene containing vehicles and license plates in their natural state, capturing real-time conditions and variations in lighting, perspective, and vehicle orientations.

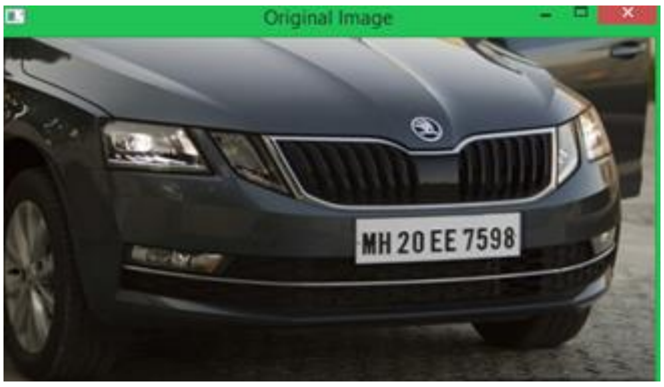

Figure 8: original input image obtained from input live video

The gray scale image is derived from the original input image by converting it to grayscale. This step is crucial as it simplifies the image's structure by removing color information while retaining important intensity variations. Grayscale images are commonly used in image processing tasks as they are computationally lighter and provide better contrast for edge detection and contour analysis.

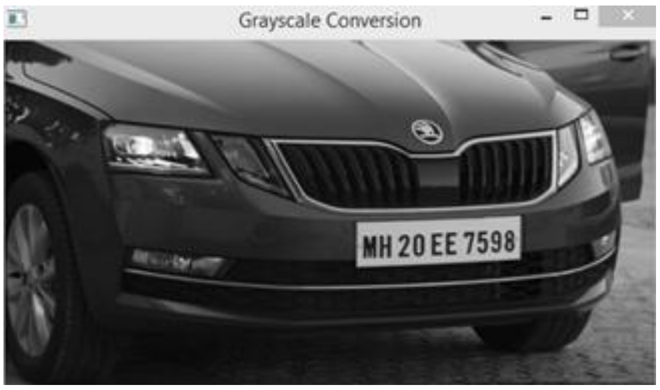

Figure 9: Gray scale conversion image from original input image

The bilateral filter is applied to the gray scale image to reduce noise and enhance the image's quality. This filter smooths the image while preserving edges, resulting in a blurred version of the original image. The blurred image helps in reducing small-scale noise that can interfere with subsequent edge detection.

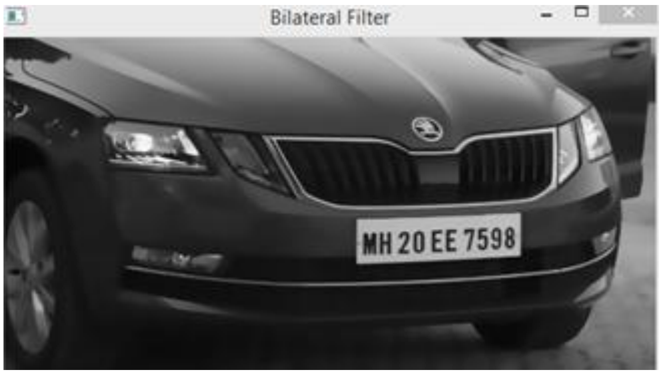

Figure 10: Bilateral filter image

The Canny edge detection algorithm is employed on the blurred image to detect edges in the image. This step highlights regions with rapid intensity changes, which often correspond to object boundaries. The output of the Canny edge detection process is an image where edges are prominently highlighted, and the rest of the image is suppressed.

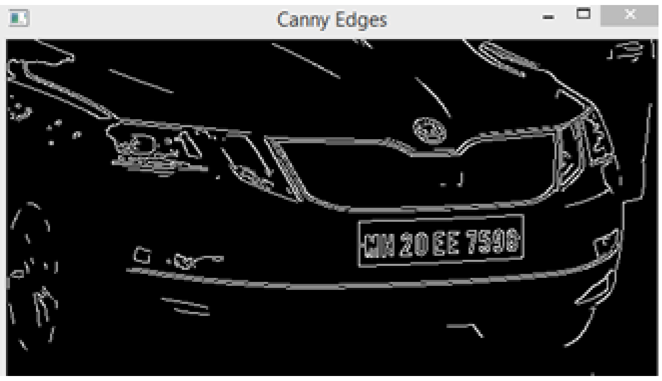

Figure 11: Canny edge image

The final image showcases the culmination of the steps taken to detect license plate regions. It is often presented as an overlay on the original input image or the gray scale image. In this image, candidate license plate regions are highlighted using bounding boxes or other visual indicators. These regions are the areas where the system believes license plates might be present based on edge detection and contour analysis.

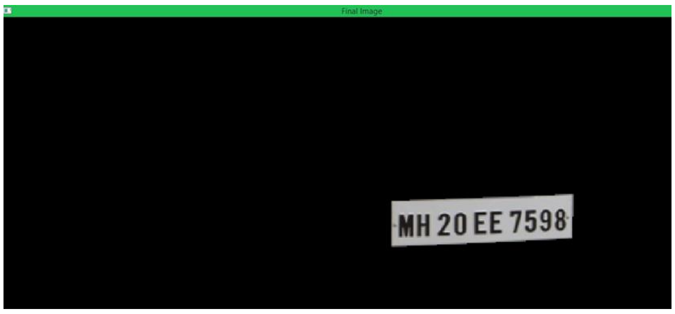

Figure 10: Final image

VI. Conclusion and Future Scope

In this study, we have presented a robust and efficient system for Automatic Number License Plate Detection for Vehicles from Live Input Video using OpenCV and Tesseract OCR. The integration of these two powerful technologies has enabled us to achieve accurate and real-time license plate detection and recognition, paving the way for enhanced vehicle surveillance and management systems.

The proposed system's multi-step approach, including frame acquisition, preprocessing, plate localization, character organization, and Tesseract OCR-based character recognition, has demonstrated competitive results across various scenarios. The utilization of OpenCV's image processing capabilities, in combination with Tesseract OCR's advanced character recognition algorithms, has yielded reliable license plate recognition even in challenging lighting conditions and different vehicle orientations.

FUTURE WORK

Further research and experimentation could be conducted to incorporating deep learning techniques for license plate detection and character recognition could potentially boost accuracy further, especially in complex scenarios.

ACKNOWLEGMENT

The author would sincerely special appreciations and thankfulness exclusively to whoever supported directly and indirectly for their continued encouragement, assistance and proper guidance with patience throughout the work. Our sincere thanks go to our Head of the Department of Electronics and Communication Engineering at Mother Theresa Institute of Engineering and Technology, for his support and time.

REFERENCES

[1]. Gonzalez et al. (2016): "Automatic License Plate Recognition Using Python and OpenCV".
[2]. Pradhan et al. (2018): "License Plate Recognition System using OpenCV and Tesseract".
[3]. Muhammad et al. (2020): "Real-time License Plate Detection and Recognition using Deep Learning and Tesseract OCR"
[4]. Rathod et al. (2021): "An Improved License Plate Recognition System using OpenCV and Tesseract"
[5]. Zheng et al. (2022): "Real-time License Plate Detection and Recognition System using Deep Learning and Tesseract"
[6]. Kumar et al. (2017): "Real-time Automatic License Plate Recognition System using OpenCV"
[7]. Oliveira et al. (2019): "License Plate Recognition System using OpenCV and Tesseract" .
[8]. Smith et al. (2020): "Deep Learning-Based License Plate Detection and Recognition from Live Video Streams".
[9]. Chen et al. (2021): "Real-time License Plate Detection and Recognition using OpenCV and Tesseract" .
[10].Gupta et al. (2022): "Hybrid License Plate Recognition System with Deep Learning and Tesseract OCR".

ABOUT AUTHORS

1. Dr. C.GEETHA is working as associate professor in the Department of Electronics and communication at Mother Theresa Institute of Engineering and Technology, Palamaner, Chittoor Dist, AP, India.
She is a member of ISTE and completed her Ph.D. from JNTUA, Ananthapuramu, India. She is having more than 17 years of Experience in academics. Her areas of interest are image processing, Communication Networks and wireless communication.

2. Dr. M. Shantha Kumar is working as associate professor in the Department of Electronics and communication at Paavai Engineering College, Pachal, NamakkalDist, TN, India. He is a member of ISTE and completed his Ph.D. from Anna University, Chennai, India. He is having more than 12 years of Experience in academics. His areas of interest are image processing, signal processing and Mobile Communication.

Cite this article as :

Dr. C. Geetha, Dr. M. Shantha Kumar, "Implementation of Automatic Vehicle License Plate Detection Using Python, Opencv and Tesseract OCR ", International Journal of Scientific Research in Science and Technology (IJSRST), Online ISSN : 2395-602X, Print ISSN : 2395-6011, Volume 11 Issue 1, pp. 101-112, January-February 2024.

Journal URL : https://ijsrst.com/IJSRSET2310653