MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

Report

On

Automatic Real-Time License Plate Recognition System

Submitted by

Kaushik Naidu 200929104

Anadya Dang 200929196

Aryaman Dev 200929064

Siddharth Sharma 200929126

Machine Learning (PE-VII)
MTE

Department of Mechatronics
MANIPAL INSTITUTE OF TECHNOLOGY, MANIPAL

November 2023

Table of Contents

Sr. No.	Topic	Pg. No.
1.	Chapter 1	3
2.	Chapter 2	4
3.	Chapter 3	5
4.	Chapter 4	7
5.	Chapter 5	12
6.	Chapter 6	15
7.	Chapter 7	17
8.	Chapter 8	19
9.	Chapter 9	20

Chapter 1

Introduction:

In an era characterized by raid advancement, the development of efficient monitoring and surveillance systems has become paramount due to the large increase in the number of cars on the road. One critical aspect of monitoring is real-time license plate detection (LPR) and the subsequent classification of vehicles into different types. The problem at hand involves developing a comprehensive system for real-time license plate recognition and accurate classification into the type of vehicles. This system aims to address the challenges associated with traffic surveillance, security, and road congestion. To accomplish the task, we must overcome several key challenges as stated below. Variability in license plates in terms of fonts, colours, sizes, and styles, both nationally and internationally. Real-time processing for LPR and vehicle classification requires high computational efficiency and robust algorithms capable of handling the dynamic nature of traffic on roads. The solution should not be hampered by adverse environmental conditions including low light, poor weather, and varying camera angles. Ensuring data security and privacy with respect to the data collected during recognition is a fundamental concern. Compliance with data protection regulations and safeguarding sensitive information are vital. Finally, the accuracy of vehicle classification is a crucial parameter for generating reliable traffic statistics and optimizing urban planning efforts. Misclassification can lead to the generation of inaccurate data, impacting the decision-making process.

The proposed system has far-reaching implications and applications. Some of which are cited below. Identifying vehicles and recognizing license plates can aid law enforcement agencies in tracking stolen vehicles, identifying offenders, and monitoring traffic violations. Real-time data on vehicle counts, types and movements can be invaluable for optimizing traffic flow, minimizing congestion, and reducing travel time. The data collected from vehicle classification can assist in urban planning strategies, influencing decisions related to road infrastructure, public transportation, and parking facilities. Lastly, the system applied with security and surveillance contexts, monitoring the access points, and facilitating investigations by identifying vehicles can provide an optimal solution to a problem.

Chapter 2

Objectives:

These systems play a pivotal role in facilitating law enforcement, boost security, and enhance traffic management and the classification of vehicles by types provides crucial data for urban planning, transport management, and security purposes. In this study, we provide a framework for real-time license plate recognition (LPR) and vehicle type classification (VTC) using computer vision and learning approaches. Designing and implementing an LPR capable of accurately recognizing license plates in real-time, accommodating variations in plate design and appearance can help in developing a robust solution. Developing an algorithm for accurately classifying vehicles into types. Enhance data privacy and security by implanting measures to ensure privacy and security of the collected license plate data, complying with data protection regulations and ethical considerations.

Chapter 3

Literature Review:

In recent years, the field of Automatic License Plate Recognition (ALPR) has witnessed a shift towards leveraging Deep Learning (DL) techniques. This literature review with reference to the base research paper of the project provides an overview of recent advancements in ALPR, with a specific emphasis on DL approaches.

LP Detection:

The LP detection stage has seen notable contributions employing Convolutional Neural Networks (CNNs) for object detection. Montazzolli and Jung (20) utilized a cascaded CNN for detecting car frontal-views and LPs, demonstrating high recall and precision. Hsu et al. (21) customized CNNs exclusively for LP detection, showing improved performance. Rafique et al. (22) explored Support Vector Machines (SVM) and Region-based CNN (RCNN) for real-time LP detection, favoring RCNNs. Li and Chen (5) introduced a CNN trained on characters cropped from general text, achieving superior recall and precision. Bulan et al. (3) integrated a Sparse Network of Winnows (SNoW) and CNN for LP detection, significantly enhancing baseline methods.

Character Segmentation:

DL-based ALPR systems often combine character segmentation and recognition. Montazzolli and Jung (20) proposed a CNN for both segmenting and recognizing characters within a cropped LP, achieving a high accuracy rate. Bulan et al. (3) achieved remarkable accuracy in LP recognition by jointly performing character segmentation and recognition using Hidden Markov Models (HMMs).

Character Recognition:

Character recognition is addressed through various DL approaches. Menotti et al. (23) utilized random CNNs for feature extraction, outperforming traditional methods. Li and Chen (5) framed character recognition as a sequence labeling problem, employing a Recurrent Neural Network (RNN) with Connectionist Temporal Classification (CTC) for recognizing entire LPs without character-level segmentation. Svoboda et al. (24) achieved high-quality LP deblurring reconstructions using a text deblurring CNN, contributing to character recognition indirectly.

Miscellaneous:

Miscellaneous DL approaches in ALPR include Masood et al. (7), who presented an end-to-end ALPR system utilizing a sequence of deep CNNs, and Li et al. (6), proposing a unified CNN for simultaneous LP location and recognition in a single forward pass.

Final Remarks:

The literature underscores that many studies focus on specific stages of the ALPR pipeline, and some evaluations lack representation of real-world scenarios. Moreover, the real-time applicability of these approaches remains a challenge. To address this, the utilization of YOLO object detection CNNs in each stage is recommended to create a robust and efficient end-to-end ALPR system. Additionally, data augmentation is emphasized for character recognition, recognizing its significance as a bottleneck in some ALPR systems.

Chapter 4

Methodology:

Our proposed system utilizes cutting-edge Convolutional Neural Networks (CNNs) for initial image processing, Region of Interest (ROI) extraction, and Recurrent Neural Networks (RNNs) for sequence recognition. Simultaneously, the system extracts feature from detected vehicles to classify them into types such as cars, trucks, motorcycles, and buses. The general methodology to accomplish this task is stated as follows.

- Data collection and preprocessing image data
    - Collection of a diverse set of images containing vehicles.
    - Annotating dataset with bounding boxes around license plates.
    - Preprocessing images by resizing, normalizing, and augmenting to enhance the quality and diversity of the dataset.

- License Plate detection using CNN
    - Train CNN for objection detection using Yolo v7 or Faster R-CNN
    - Apply the trained detector around the input images to obtain bounding boxes.

- ROI Extraction
    - Extract ROIs from input images using the bounding boxes obtained in the previous step.

- License Plate recognition using CNN
    - Preprocess the extracted ROIs which may include resizing and enhancing the contrast for better recognition.
    - Train a CNN for character segmentation and recognition on the license plate ROIs.

- Character Segmentation
    - Use the trained CNN to segment characters or digits obtained within the license plate ROI.
    - This step involves additional pre-processing steps such as thresholding and morphological operations.

- Sequence Recognition using RNN
    - Train an RNN like LSTM, or BiLSTM to recognize the sequence of characters obtained from the segmented license plate.
    - The RNN takes segmented characters as input and generates a sequence of characters which represents the recognized license plate.

- Post-Processing
    - Apply post-processing steps to improve recognition accuracy.

- This includes spell-checking and filtering out the improbable license plate sequences based on patterns or country-specific rules.

- Evaluation and fine-tuning
  - Evaluating the overall system performance using metrics like precision, accuracy, recall and F1-score.
  - Fine tune by adjusting epoch, batch size, activation function, optimizers, dropout layers and various other hyperparameters.

- Deployment
  - Deploy the trained system in real-time or batch processing scenarios to detect, extract, and recognize license plate images.

- Validation and testing
  - Conduct extensive testing and validation in real-world scenarios, accounting for variations in lighting, weather, and different license plates.

• **1. Grayscale Conversion:** •	- converting a colour image into a black and white single-channel image - It simplifies the image by removing colour information, making it easier to process - The grayscale image retains important structural details for subsequent analysis
2. Bilateral Filtering for Noise Reduction:	- smoothing an image while preserving important edges - reduces noise without blurring significant features - Consists of two components: spatial domain (distance between pixels) intensity domain (difference in pixel values). - High intensity differences are preserved as edges, while low-intensity differences are smoothed - Helps in removing salt-and-pepper noise and other high-frequency noise
3. Canny Edge Detection:	- used to identify edges in an image. - finds rapid changes in intensity, which often correspond to object boundaries
4. Contour Detection:	- identifying the boundaries of objects within an image. - helps in recognizing and extracting the shape and structure of objects. - Common algorithm for contour detection: Canny edge detector - The output is a set of points that represent the contours of objects in the image
5. Polygon Approximation:	- simplifies the representation of those contours by reducing the number of vertices which makes subsequent processing more efficient - The result is a simplified polygon that closely matches the shape of the original contour
6. Masking:	- using a binary image (mask) to selectively manipulate certain parts of an image while preserving others - A binary mask is created where certain regions are set to either 1 (retain) or 0 (discard)

	- used for segmenting and isolating specific regions of interest in an image. - Masking is applied by element-wise multiplication or addition of the pixel values in the mask and the original image
7. ROI Cropping:	- selecting a specific region or area of interest from an image for further processing - Helps focus on the relevant part of an image, especially in scenarios where only a portion contains valuable information. - Coordinates of the region to be cropped are defined, and the selected area is extracted. - Useful for improving processing efficiency and accuracy by reducing the amount of unnecessary information
8. Text Recognition using EasyOCR:	- EasyOCR is an open-source OCR library that provides a simple interface for performing text recognition on images

Machine Vision Project Integration Overview:

1. Image Acquisition

   - gather dataset

2. Grayscale Conversion

   - colour images to grayscale

   - for simplicity and better processing efficiency.

3. Bilateral Filtering

   - for noise reduction while preserving important edges.

4. Contour Detection

   - to identify object boundaries.

5. Polygon Approximation

   - Simplify contours to reduce the number of vertices.

6. ROI Cropping

   - to focus on specific areas containing valuable information.

7. Masking

   - to isolate and manipulate specific regions or objects within the image.

8. Text Recognition using EasyOCR(optical character recognition)

   - Utilize EasyOCR for text recognition on selected regions or the entire image.

9. post-processing

   - Analyse the results of contour detection, polygon approximation, and OCR.

   - Handle errors, format text, and extract relevant features.

10. Object Analysis

- Combine information from contour detection, polygon approximation, and OCR for comprehensive object analysis.

11. Decision Making

- Use the extracted information and analysis results to make decisions within the machine vision project.

The main model used is Haar Cascade Clssifier:

A Haar Cascade Classifier is a machine learning object detection method that uses a set of Haar-like features and a cascade of classifiers to detect objects in images or video. This technique was introduced by Viola and Jones in their paper titled "Rapid Object Detection using a Boosted Cascade of Simple Features."

Here's a more in-depth explanation of the Haar Cascade Classifier:

Haar-like Features:

Haar-like features are simple rectangular filters that are used to represent different characteristics of an object. These features are similar to the convolutional filters used in image processing. They can capture information about edges, corners, and other important patterns.

Integral Image:

To efficiently calculate Haar-like features over an image, an integral image is used. The integral image representation allows for the rapid computation of the sum of pixel values within any rectangular region of the image.

Adaboost Training:

The training process involves the AdaBoost algorithm, which is a boosting algorithm that combines multiple weak classifiers to create a strong classifier. In the context of Haar Cascade, each weak classifier corresponds to a Haar-like feature. The algorithm assigns weights to misclassified samples, giving more importance to the ones that are harder to classify correctly.

Cascade of Classifiers:

The AdaBoost algorithm produces a series of weak classifiers. These classifiers are organized into a cascade, where each stage of the cascade represents a strong classifier. The cascade is designed such that it quickly rejects regions of the image that are unlikely to contain the object of interest.

Sliding Window:

During the detection phase, a sliding window is used to scan the image at different scales and positions. At each window position, the Haar-like features are computed, and the cascade of classifiers is applied. If a window passes all stages of the cascade, it is considered a positive detection.

False Positive Rejection:

The cascade structure is crucial for efficient object detection. At each stage, if a region is classified as negative, the process stops, and the region is rejected. This helps in quickly discarding regions that are unlikely to contain the object, reducing the computation time.

Training with Positive and Negative Samples:

Training a Haar Cascade Classifier requires positive and negative samples. Positive samples are images containing the object of interest, and negative samples are images without the object. The classifier is trained to distinguish between these two classes.

XML File:

The trained Haar Cascade Classifier is often stored in an XML file. This file contains information about the structure of the cascade, including the type and parameters of each weak classifier.

Chapter 5

Experiment and Implementation:

CODE

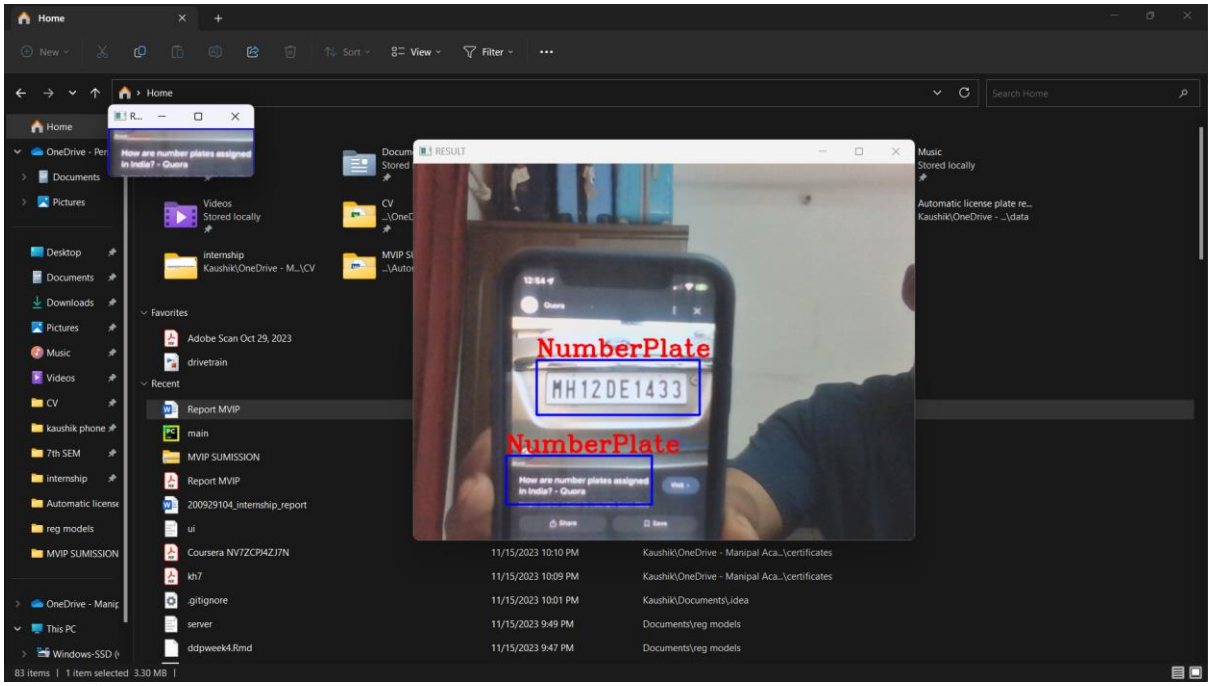
PART 1

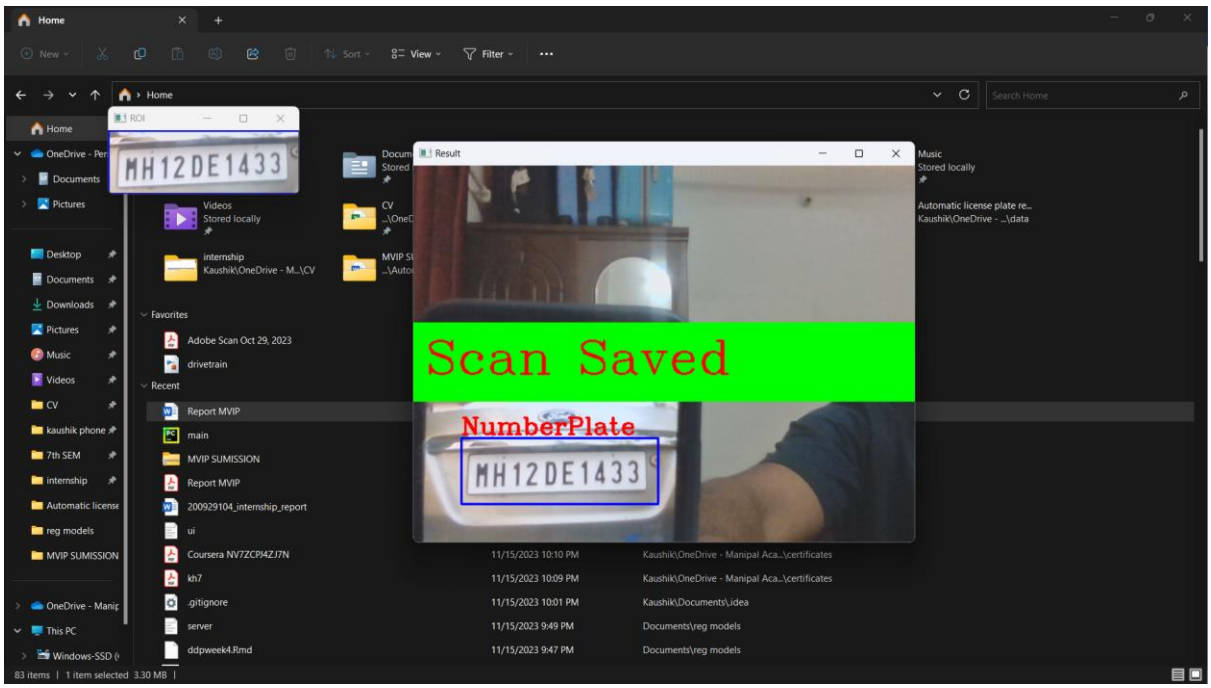
```
import cv2

platecascade =
cv2.CascadeClassifier("haarcascade_russian_plate_number.xml")
minArea = 500
cap = cv2.VideoCapture(0)
count = 0
while True:
    success, img=cap.read()
    imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    numberplates = platecascade.detectMultiScale(imgGray,1.1,4)
    for(x,y,w,h) in numberplates:
        area = w*h
        if area >minArea:
            cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)
            cv2.putText(img,"NumberPlate",(x,y-
5),cv2.FONT_HERSHEY_COMPLEX,1,(0,0,255),2)
            imgRoi = img[y:y + h, x:x + w]
            cv2.imshow("ROI", imgRoi)
    cv2.imshow("RESULT",img)
    if cv2.waitKey(1) &0xFF ==ord('s'):
        cv2.imwrite("C:\Desktop\MLALPR\imageplates\image"+str(count)+
".jpg",imgRoi)
        cv2.rectangle(img, (0, 200), (640, 300), (0, 255, 0), cv2.FILLED)
        cv2.putText(img, "Scan Saved", (15, 265), cv2.FONT_HERSHEY_COMPLEX,
2, (0, 0, 255), 2)
        cv2.imshow("Result", img)
        cv2.waitKey(500)
        count += 1

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```





Part 2

```
import cv2
from matplotlib import pyplot as plt
import numpy as np
import imutils
import easyocr
img = cv2.imread('image1.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.imshow(cv2.cvtColor(gray, cv2.COLOR_BGR2RGB))
bfilter = cv2.bilateralFilter(gray, 11, 17, 17) #Noise reduction
edged = cv2.Canny(bfilter, 30, 200) #Edge detection
plt.imshow(cv2.cvtColor(edged, cv2.COLOR_BGR2RGB))
keypoints = cv2.findContours(edged.copy(), cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
```

```
contours = imutils.grab_contours(keypoints)
contours = sorted(contours, key=cv2.contourArea, reverse=True)[:10]
location = None
for contour in contours:
    approx = cv2.approxPolyDP(contour, 10, True)
    if len(approx) == 4:
        location = approx
        break
location
mask = np.zeros(gray.shape, np.uint8)
new_image = cv2.drawContours(mask, [location], 0,255, -1)
new_image = cv2.bitwise_and(img, img, mask=mask)
plt.imshow(cv2.cvtColor(new_image, cv2.COLOR_BGR2RGB))
(x,y) = np.where(mask==255)
(x1, y1) = (np.min(x), np.min(y))
(x2, y2) = (np.max(x), np.max(y))
cropped_image = gray[x1:x2+1, y1:y2+1]
plt.imshow(cv2.cvtColor(cropped_image, cv2.COLOR_BGR2RGB))
reader = easyocr.Reader(['en'])
result = reader.readtext(cropped_image)
result
text = result[0][-2]
font = cv2.FONT_HERSHEY_SIMPLEX
res = cv2.putText(img, text=text, org=(approx[0][0][0],
approx[1][0][1]+60), fontFace=font, fontScale=1, color=(0,255,0),
thickness=2, lineType=cv2.LINE_AA)
res = cv2.rectangle(img, tuple(approx[0][0]), tuple(approx[2][0]),
(0,255,0),3)
plt.imshow(cv2.cvtColor(res, cv2.COLOR_BGR2RGB))
for detection in result:
    text = detection[-2]
    confidence = detection[-1]
    print(f"Text: {text}, Confidence: {confidence}")
```

Chapter 6

Challenges of License Plate Detection

1. Plate Variation in Size, Position, and Color:

- License plates can vary significantly in size, position, and colour across different vehicles.

- Differentiating between standard and non-standard plate sizes is essential for accurate recognition.

- Account for variations in plate positioning, such as centred, skewed, or at an angle.

- Handling variations in plate colours due to different plate materials and environmental conditions.

2. The number of License Plates in the Image:

- An image may contain multiple vehicles, each with its own license plate.

- Developing mechanisms to identify and isolate individual plates in a scene with multiple vehicles.

3. Plate Occlusion and Plate Character Font:

- Plates may be partially occluded by other objects, such as frames, stickers, or other vehicles.

- Recognizing characters on partially visible plates and handling occlusions for accurate identification.

- Variation in character fonts on plates and adapting recognition models accordingly.

4. Presence of Noise and Unwanted Components:

- Images may contain noise, artefacts, or unwanted components that can interfere with recognition.

- Implementing noise reduction techniques to improve the signal-to-noise ratio in the image.

5. Physical Obstructions and Damaged Plates:

- Plates may be damaged, have missing characters, or be partially obscured due to wear and tear.

- Developing robust recognition algorithms that can handle damaged plates and still extract relevant information.

6. Camera Mounting Variation and Elevation Angle:

- Cameras can be mounted at different positions on vehicles, affecting the perspective of the plates.

- Handling variations in camera mounting positions and adjusting recognition models accordingly.

- Accounting for elevation angles to accurately recognize plates in images taken from different heights.

7. Camera Resolution, Camera Focus Length, and Camera Shutter Speed (Motion Blur):

- Varying camera resolutions can impact the clarity of the captured image.

- Adaptation to different camera focus lengths to handle variations in image sharpness.

- Handling motion blur caused by fast-moving vehicles, especially in scenarios with low camera shutter speeds.

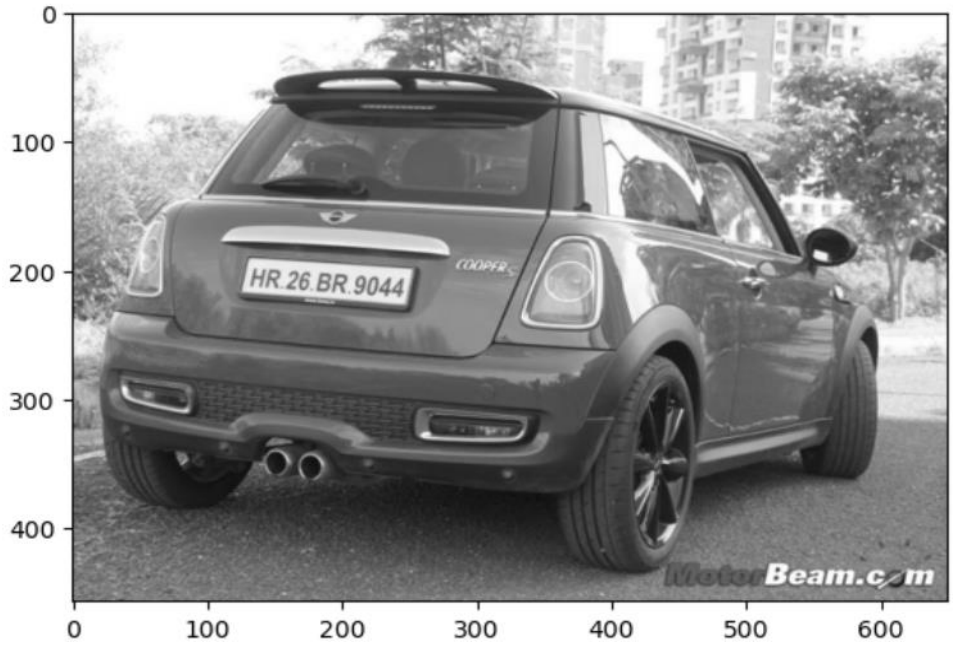
Addressing these challenges requires a combination of advanced image processing techniques, robust machine learning models, and careful consideration of the environmental and hardware factors affecting the captured images. It's important to thoroughly test and validate the system under diverse conditions to ensure its effectiveness in real-world scenarios.

Chapter 7

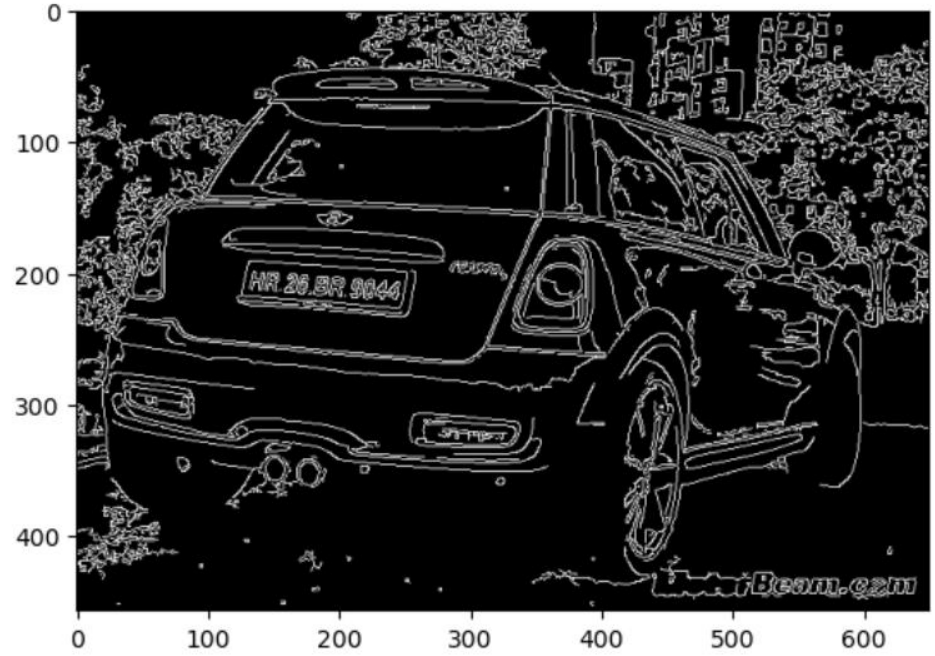
Results

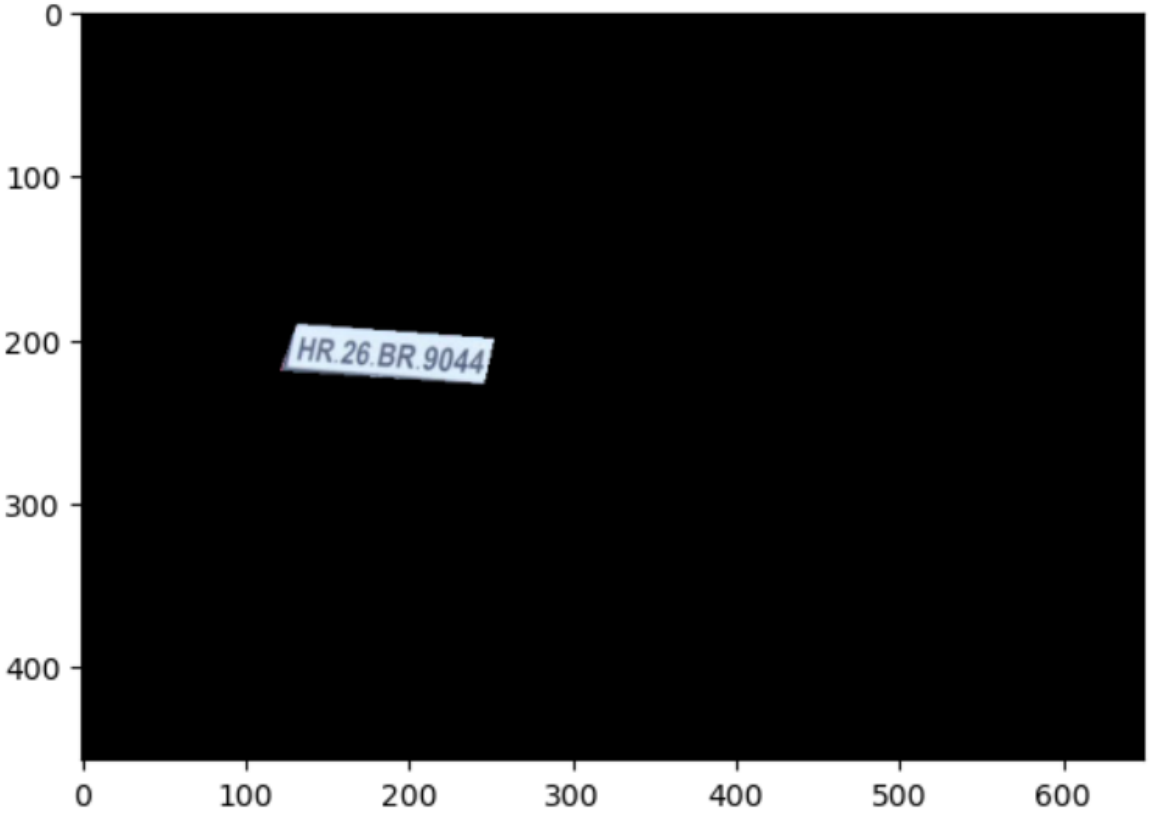
Clear Image

Input



Processing




HR.26.BR.9044
0
100
200
300
400
0
100
200
300
400
500
600


HR.26 BR.9044
HR.26 BR.9044
MotorBeam.com
0
100
200
300
400
0
100
200
300
400
500
600

Chapter 8

Conclusions and Future Works:

In conclusion, the Automatic License Plate Recognition (ALPR) system presents a cutting-edge solution that harnesses the power of machine vision and image processing to revolutionize vehicle identification and tracking. Throughout our exploration, we have delved into key components and methodologies that constitute the ALPR pipeline. From image acquisition and preprocessing, including techniques such as grayscale conversion, bilateral filtering, and Canny edge detection, to the utilization of state-of-the-art models like Single Shot Multibox Detector (SSD), we have witnessed a synergy of technologies driving accurate and efficient license plate detection.

The implementation of ALPR extends beyond mere license plate identification; it serves as a cornerstone in enhancing traffic management, law enforcement, and security systems. The robustness of the system, demonstrated through its capability to handle diverse environmental conditions and varying license plate formats, underscores its adaptability and real-world applicability.

Moreover, our foray into training and fine-tuning the ALPR model underscores the flexibility of these systems, allowing customization to specific use cases and environments. We have navigated through the intricacies of model training, evaluation, and deployment, empowering users to tailor the ALPR system to their unique requirements.

The demonstrated real-time detection capabilities, coupled with features like region-of-interest (ROI) cropping and text recognition, highlight the versatility and potential of ALPR across a spectrum of applications, from smart city initiatives to intelligent transportation systems.

As we embrace the future of automated systems, ALPR stands as a testament to the fusion of machine learning, computer vision, and artificial intelligence, ushering in a new era of efficiency, accuracy, and security in license plate recognition technology.

Chapter 9

References:

- Park, Se-Ho, Saet-Byeol Yu, Jeong-Ah Kim, and Hyoseok Yoon. 2022. "An All-in-One Vehicle Type and License Plate Recognition System Using YOLOv4" *Sensors* 22, no. 3: 921. https://doi.org/10.3390/s22030921
- Li Yao et al 2019 J. Phys.: Conf. Ser. 1237 022155

Details of Group Member

S.No.	Name	Registration No.	Contribution
1	**Kaushik Naidu***	**200929104**	**Research and Model Dev**
2	**Anadya Dang**	**200929196**	**Research and documentation**
3	**Aryaman Dev**	**200929064**	**Research and Model Dev**
4	**Siddharth Sharma**	**200929126**	**Research and Model Dev**
5	**Tejas Dhingra**	**200929294**	**Research and documentation**