**Chapter IV**

**RESULT AND DISCCUSION**

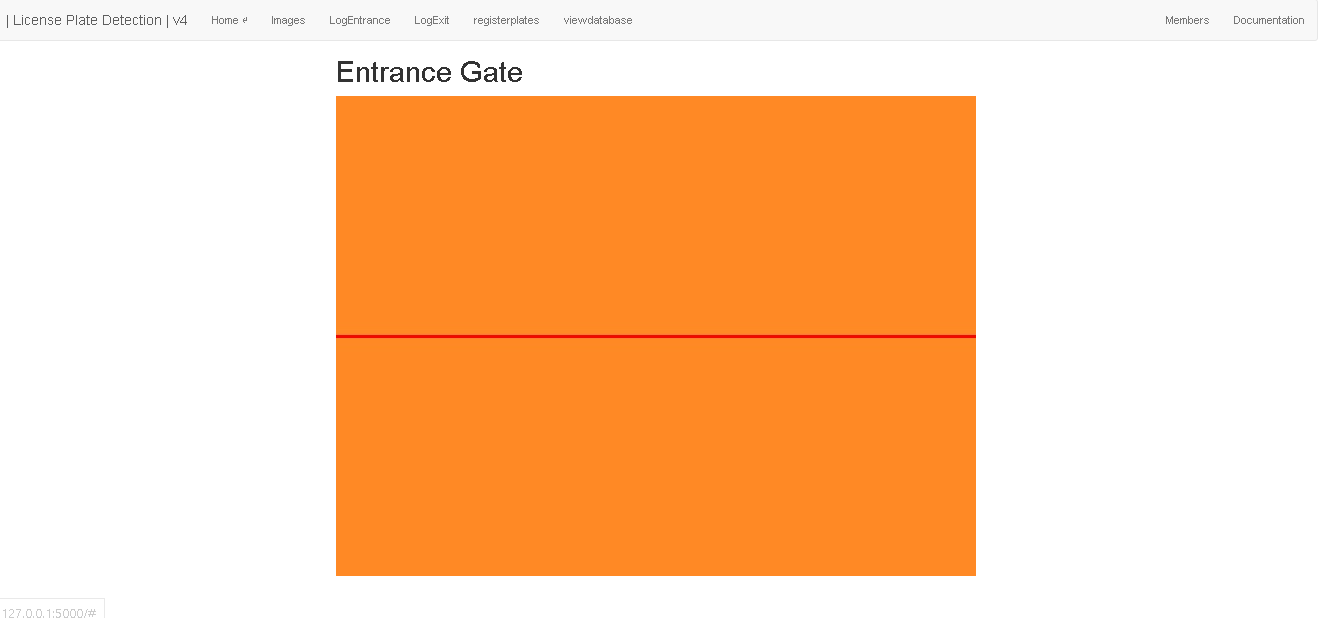
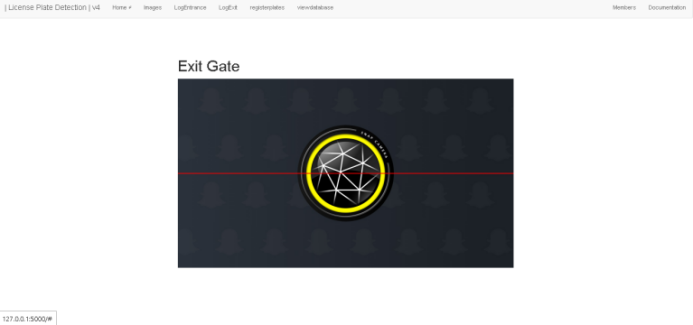
1.Develop an image capturing system for gate surveillance that enhances the security of the premises.

**Home Detection and recognition page**

**Figure 13**

Home Page Proponents Successfully Developed the Computer Vision Based Plate Number Recognition and Tracking with Deblurring Algorithm for Security of the University Premises. Using Python flask as Back-end with the functionalists of Video feed ,Image Capturing, Object detection,Object Tracking, Optical Character Recognition ,Speed detection and Image Enhancing/ Deblurring , this page have Dual Camera Inputs for Entrance and exit Camera.

## Entrance and exit View camera (separate view)



## Figure 14-15.

## Entrance And Exit Page

On this page of The web-application it shows that the system has a separate Camera inputs for CCTV,IP Camera and also a web-cam, this also have the functionalities of the Home page.

**Captured and enhanced Images**

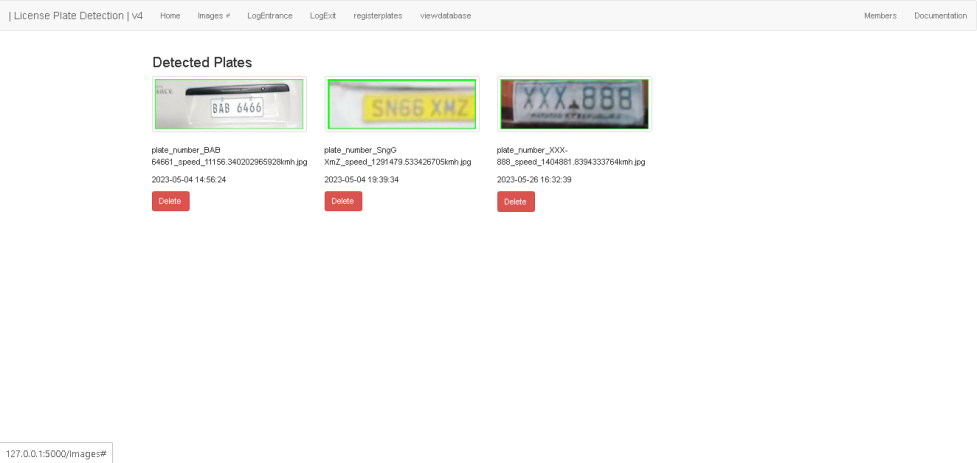
Figure 16.

Figure 16 shows that the Web application have a Image Processing Fuctionalities to return the image that was capture during the detection process, it also shows the name of the image with its plate number, Speed , and the time and date. To indicate its UID Unique identifier



**Deblurring Enhancing Image page**

In Figure above shows that Deblurring page were user can upload the blurred image to enhance it.

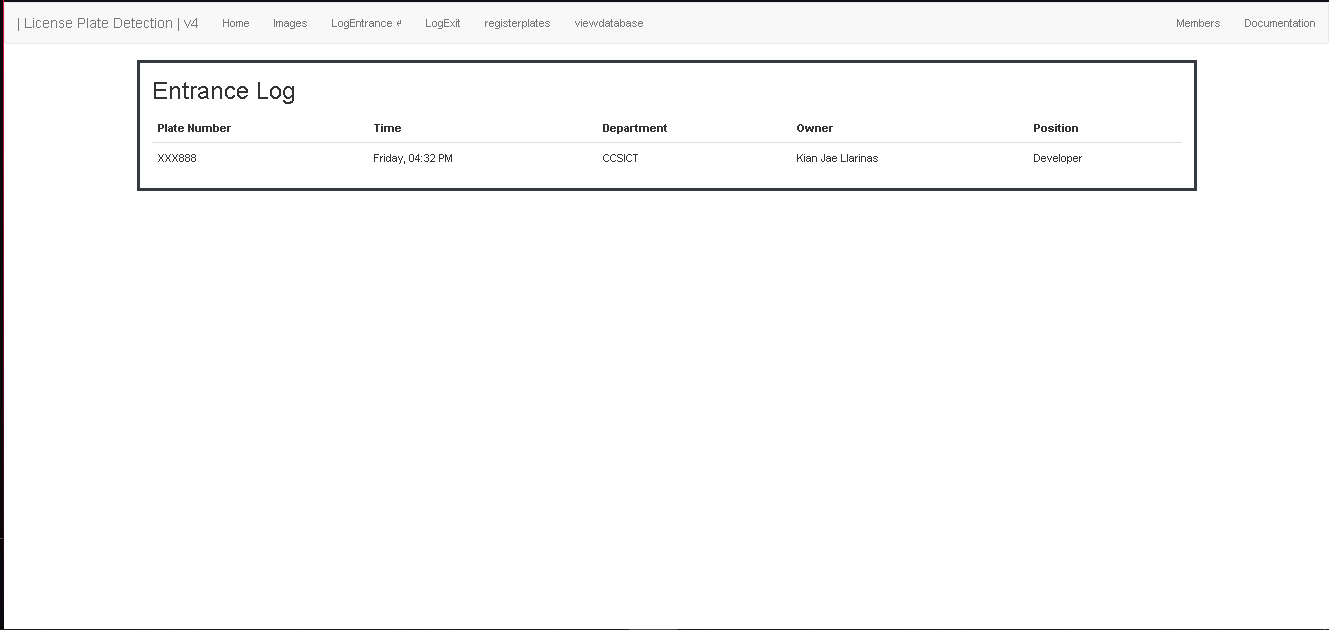
**Entrance & Exit Log**

Figure 17.

Entrance log page

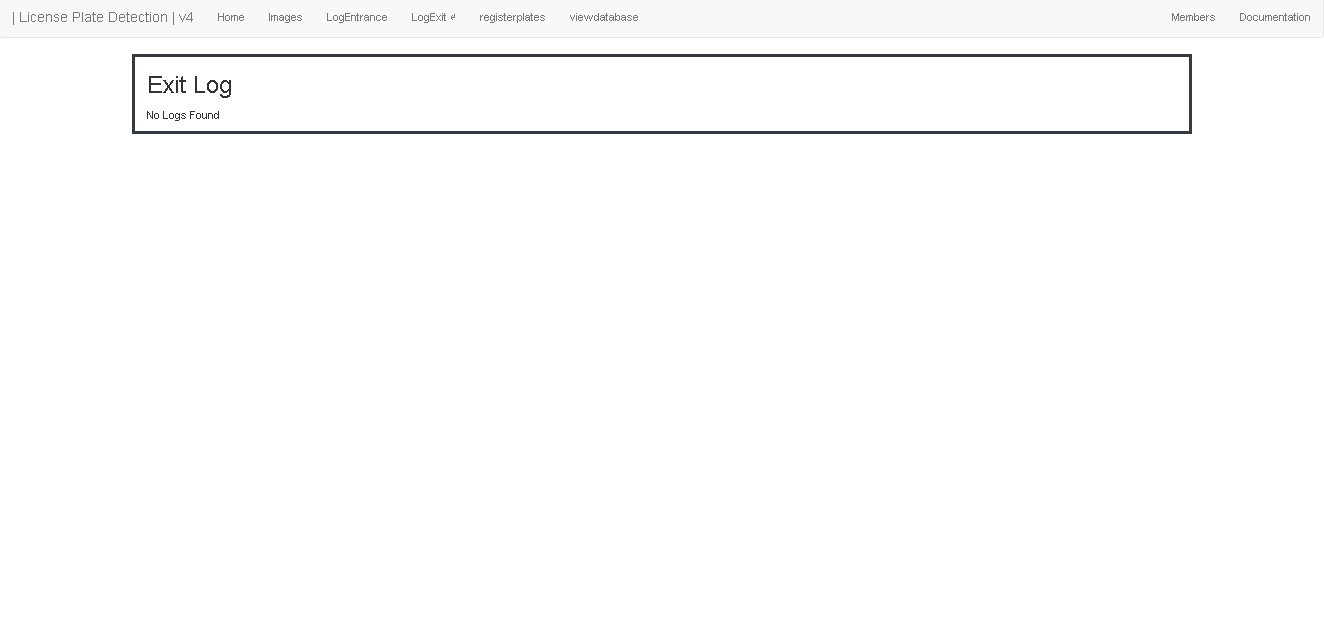


Figure 18

Exit log page

Figure 17,18 Shows that the system has a Fuction that returns the Time, date and platenumber’s information, that the system captured on the gates

**Register and registerd Plate Number**

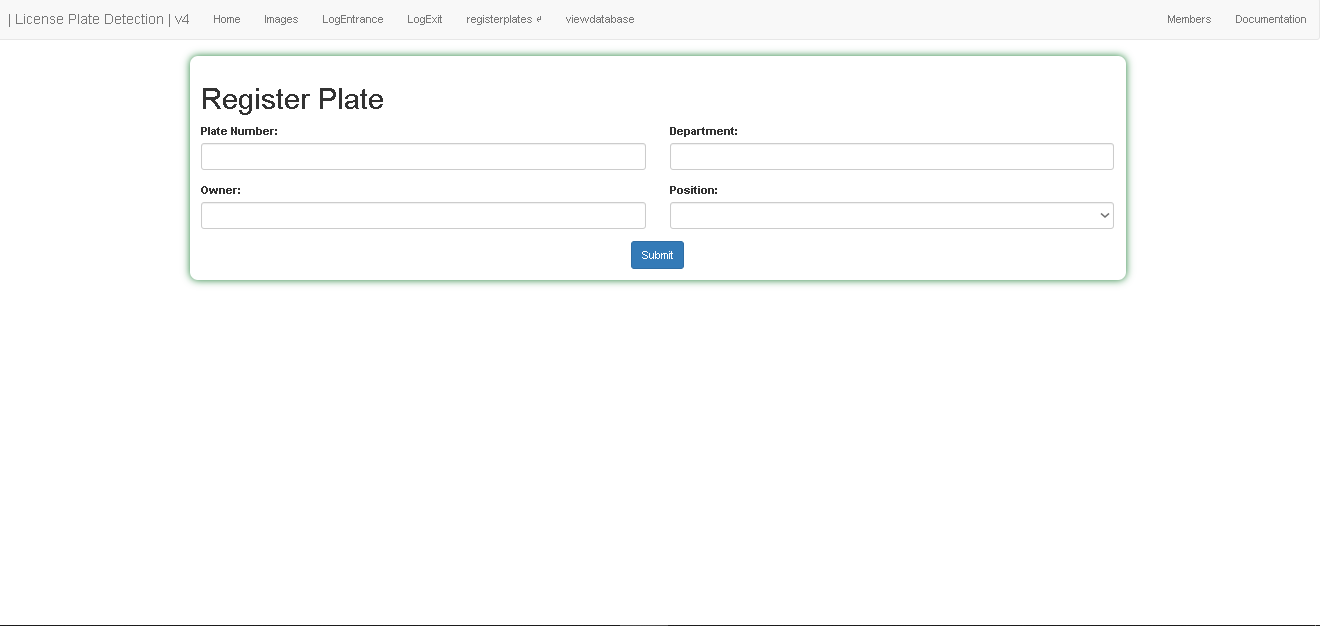


Figure 19.

Register Plate Page

Figure 19 The system has Registration form to register new plate numbers so that the system can Identify Entering, and exiting the gate of isu. By registering a Plate Number, Owner, Department, and University Position (President,Vicepresident,EO,ect..) This will be possible to get recognize by the system

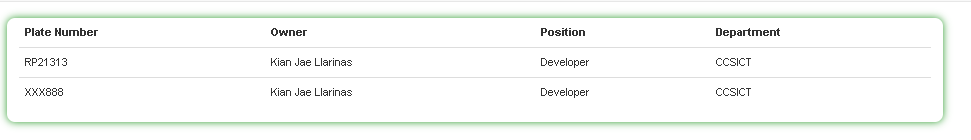


Figure 20.

Registered Plate Page

Figure 20. Shows all the information of the Registered plate on the database. That will be recognize by the system.

2,Analyze the real time information regarding vehicle recognition of the plate numbers.

**Data sets**

The Data sets proponents used to Implement the Developed system is from a personal data set and a secondary data set from Kaggle.

Haar cascade is a machine learning-based object detection algorithm used in computer vision.

To Create a Haar cascade data set proponents need to make a Positive(Gray Scaled) and Negative(Gray Scaled) Image to train the data, were Positive image must be the Model / Object it self and the Negative Image is the Back ground image that the system don’t want to detect like the Road way, Headlights, ect..

**Positive Image (Gray Scaled)**  Objects you want to detect

**Negative Image (Gray Scaled),** Objects you don’t want to detect

IMG_256

**Plate Number Detection and and tracking.**

Proponents use Machine learning-base to Detect and Track the Object.





If the vehicle enters the field of view of the camera, with the data ser applied, It will detect that the plate number is present on the camera.

Steps to Create a Object Detection and tracking

1. Load the trained cascade classifier or the Data set
2. Load the image or video frame
3. Convert the image to grayscale
4. Perform object detection
5. Draw bounding boxes around the detected objects
6. Display the result.

**Speed detection.**

The Speed detection process. The Speed detection method Proponent used as you can see in the figure below, The camera setup has a line inside the camera feed, that represent the Speed detection of the Developed System.





After The vehicle Entered the FOV( Field of view) of the camera, The Bounding box represent as the key to start the Speed detection.

**Speed Detection Process.**

1. Declares the variables for the Red Line inside the frame, this will act as the Calculator of the speed detection
2. Receive the frames during the Object detection process
3. While the Bounding box is present on the frame get the current time frame.
4. If the bounding box touches the Red line it will calculate the speed.

**Speed Detection Algorithm.**

Step 1: Capture the current time frame and the previous time frame.

Step 2: Calculate the time elapsed between the current and previous time frames.

Step 3: Measure the distance traveled between the current and previous positions.

Step 4: Calculate the speed by dividing the distance traveled by the time elapsed.

Step 5: Convert the speed to the desired unit (e.g., kilometers per hour) based on the pixel-to-meter conversion factor.

Step 6: Update the necessary information for the next iteration. Store the current time frame as the previous time frame and the current position as the previous position.

Step 7: Return the calculated speed.

**Optical Character recognition Algorithm.**

In the developed system Proponents use EasyOCR library to do the task, EasyOCR is a Python library that provides a simple and straightforward way to perform Optical Character Recognition (OCR) on images. OCR is the technology used to recognize and extract text from images or scanned documents.

After the Plate Number is detected the system will now read the Characters inside the plate number



XYZ 000

Steps to Perform and use OCR to the system.

1. Load the OCR model
2. Perform OCR on an image by using build in the Function readtext()
3. Process the OCR results.
4. Remove non-alphanumeric characters.
5. Upper case all the letters
6. Show The out put

**Deblurring Algorithm:**

After the Plate Number is detected the system will now perform a Blind Deconvolution Algorithm. To enhance the Visual of the result Image.

Deblurring Images Using the Lucy-Richardson Algorithm.

The Lucy-Richardson algorithm is an iterative method used to remove blurriness from images.

1. Initialization:

* Start with an initial estimate of the original signal.
* Define a point spread function (PSF) that represents the blurring effects of the convolution.

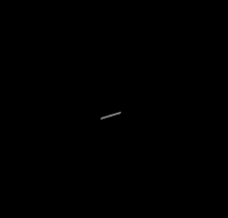
1. Iterative Steps:

* Multiply the current estimate of the original signal with the PSF.
* Normalize the result by dividing it by the convolution of the estimate with the PSF.
* Multiply the obtained result by the degraded signal.
* Normalize the result by dividing it by the convolution of the estimate with the PSF.

1. Output:

* The final estimate of the original signal is obtained after the iterations.

**Computation :**



f(x,y) \* PSF h(x,y) = g(x,y)

(Scene) (Point Spread Function) (Image Blurred)

f(x,y) \* h(x,y) = g(x,y)

let f' Prime be the recovered scene.

f'(x,y) \* h(x,y) = g(x,y)

F'(u,v) H(u,v) = G(u,v)

F'(u,v) = G(u,v) / H(u,v) = f'(u,v)

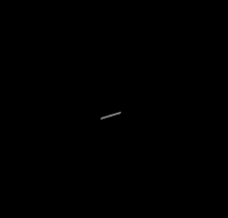


Image g(x,y) / PSF h(x,y) = f’(x,y)

(Blurred) (Point Spread Function) (Recovered)

F'(u,v) = G(u,v) / H(x,y) ➡ IFT ( inverse Fully Transform) ➡ f'(x,y)

Step 1 recover F'(u,v) in Fourier Domain

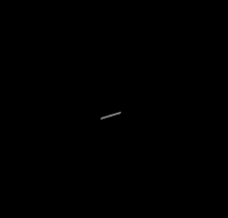


Image g(x,y) deconvolve PSF h(x,y) = f’(x,y)

(Blurred) (Point Spread Function) (Recovered)

Image g(x,y)blurred deconvolve PSF h(x,y) = Rcoverd f'(x,y)

**Firebase in Python:**

After the system Recognize and read the plate number’s UID number (unique identifier) the system will use Firebase to see the information of the plate number.

Firebase's real-time database is a No-SQL cloud-hosted database that allows you to store and sync data in real-time across connected clients. It enables you to build real-time collaborative applications like chat apps, live collaboration tools, and multiplayer games. Perfect for Plate Number recognition in real time information processing.

3.Evaluate the system Functionalities using Easy-OCR and Open-CV base in the following:

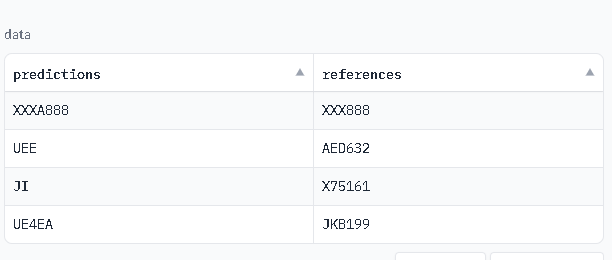
**Easy-OCR :**

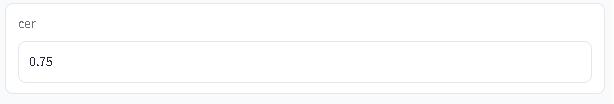
Getting the Accuracy Of Easy-OCR Using Character Error Rate (CER)

**Errors to consider:**

1. Substitution error: Misspelled characters/words
2. Deletion error: Lost or missing characters/words
3. Insertion error: Incorrect inclusion of character/words

|  |  |  |
| --- | --- | --- |
| **IMAGE** | **REFERENCE** | **OUTPUT** |
|  | XXX888 | XXXA888 |
| IMG_256 | AED632 | UEE |
|  | X75161 | JI |
|  | JKB199 | UE4EA |
|  | CAT9951 | CAT9951 |





**Formula:**

CER = (Substitution error + Insertions + Deletions) / Length of Characters in plate number.

**Functionality of Easy-OCR.**

The main functionality of EasyOCR in the system

1. Text Recognition: EasyOCR can recognize and extract text from various sources such as images, PDF files, and scanned documents.
2. Batch Processing: EasyOCR supports batch processing, allowing you to process multiple images or documents in a single run.

**Reability**

The reliability of EasyOCR depends on factors such as image quality, text complexity, and the implementation used. High-resolution, well-lit images yield better results, while blurry or low-contrast images may affect accuracy. Common fonts and languages are recognized more accurately, while unusual fonts, handwriting, or complex scripts pose challenges. The library's accuracy can be influenced by its version, training data, and preprocessing techniques. Keeping the library updated and following best practices can enhance its reliability .

**OpenCV :**

Getting the Accuracy Of OpenCV Using Evaluation Metrics.

**Data gathered**

|  |  |  |  |
| --- | --- | --- | --- |
| Image ID | Ground Truth | Detected Plate | Accuracy |
| 1 | ABC123 | ABC123 | Correct |
| 2 | XYZ789 | XYZ789 | Correct |
| 3 | DEF456 | DEF456 | Correct |
| 4 | GHI789 | GHI789 | Correct |
| 5 | JKL012 | JKL012 | Incorrect |
| 6 | MNO345 | --- | Missed |
| 7 | PQR678 | --- | Missed |
| 8 | STU901 | STU901 | Incorrect |
| 9 | VWX234 | VWX234 | Incorrect |
| 10 | YZA567 | --- | Missed |
| … | … | … | … |
| 60 | RST890 | RST891 | Incorrect |

**Result :**

|  |  |
| --- | --- |
| **Evaluation Metrics** | **Value** |
| True Positives (TP) | 49 |
| False Positives (FP) | 5 |
| False Negatives (FN) | 6 |
| Total Images | 60 |
| Accuracy | 83.05% |
| Precision | 90.74% |
| Recall | 89.66% |

**Formula:**

Accuracy = (49 / (49 + 5 + 6)) \* 100

Precision = (49 / (49 + 5)) \* 100

Recall = (49 / (49 + 6)) \* 100

**Functionality:**

1. Image Preprocessing: OpenCV offers numerous functions for image preprocessing, such as resizing, blurring, thresholding, and edge detection. These operations help enhance the quality of the input image and prepare it for further processing.
2. Plate Localization: OpenCV provides techniques for locating license plates within an image. This can be achieved using methods like edge detection, contour analysis, and template matching. The goal is to identify the region of the image that contains the license plate.
3. Output Generation: OpenCV can be used to display or store the recognized license plate number along with any additional information extracted from the image, such as the location or timestamp

**Reability :**

1. Image quality: The accuracy of license plate recognition heavily relies on the quality of the input image. Factors like lighting conditions, image resolution, blurriness, and noise can affect the performance of plate recognition algorithms. It's important to have clear and well-illuminated images for better results.
2. Plate variation: License plates can vary in terms of size, font type, color, and background. OpenCV provides several techniques, such as edge detection, contour analysis, and character segmentation, to handle these variations. However, extreme variations or unconventional license plates might pose challenges to recognition accuracy.
3. Algorithm selection: OpenCV offers a variety of algorithms and techniques for license plate recognition, such as template matching, feature-based methods, and machine learning approaches. The choice of algorithm depends on the specific requirements and constraints of the application. Some algorithms may perform better under certain conditions than others.
4. Training and customization: OpenCV provides a platform to develop custom recognition models using machine learning techniques. Training the model on a diverse dataset that covers various license plate types and conditions can improve recognition accuracy. Customization allows you to optimize the model for specific regions, plate formats, or other unique characteristics.
5. Real-time processing: Real-time license plate recognition requires efficient algorithms and optimization techniques to handle video streams effectively. OpenCV provides tools for video processing, and with proper implementation, real-time plate recognition can be achieved.

**CHAPTER V**

**RECOMMENDATIONS AND CONCLUSIONS**

**Conclusions**

In this chapter, we discussed the results and discussed the functionalities of the developed image capturing system for gate surveillance. The system aimed to enhance the security of the premises by implementing computer vision-based plate number recognition and tracking with a deblurring algorithm. The system was developed using Python Flask as the backend, incorporating various functionalities such as video feed, image capturing, object detection, object tracking, optical character recognition (OCR), speed detection, and image enhancing/deblurring.

The developed system provided a user-friendly web application with separate camera inputs for entrance and exit views. It allowed users to capture and enhance images, detect and track plate numbers, log entrance and exit information, register plate numbers, and access the registered plate information stored in the database.

To analyze real-time information regarding vehicle recognition of the plate numbers, the system utilized machine learning-based object detection algorithms and Haar cascade data sets. Positive and negative images were used to train the data, and the detection and tracking process were performed on the captured frames. Additionally, the system incorporated speed detection by calculating the speed of the vehicle using a designated line in the camera feed.

The system employed EasyOCR library for optical character recognition, which enabled the recognition and extraction of text from images, including the plate numbers. The accuracy of EasyOCR was evaluated using character error rate (CER), considering substitution, deletion, and insertion errors.

OpenCV was utilized for image preprocessing, plate localization, and output generation. The accuracy of OpenCV was evaluated using evaluation metrics such as true positives, false positives, false negatives, accuracy, precision, and recall.

Overall, the developed system demonstrated promising results in enhancing gate surveillance and improving security. The functionalities of EasyOCR and OpenCV proved effective in recognizing and extracting plate numbers from images, although their accuracy depended on various factors such as image quality, text complexity, and plate variations. The reliability of the system relied on well-lit and clear images, appropriate algorithm selection, training, customization, and real-time processing optimization.

**Recommendations**

1. Utilize a variety of sensors, including RI sensors, inductive loop detectors (ILD), and other appropriate sensors, to enhance monitoring and data collection capabilities.
2. Deploy effective speed reduction measures, considering options such as speed bumps, traffic calming devices, signage, and other engineering solutions to encourage drivers to adhere to safe speed limits.
3. Implement advanced camera systems capable of capturing high-resolution footage, such as UHD/4K (3840 x 2160) resolution, for improved surveillance and identification of incidents on the road.