Smart Housing: Enhancing Security Through License Plate Recognition

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology

in

Computer Science Engineering Core

by

TANMAY AGRAWAL (21BCE2825)

AKSHAT KHANNA (21BCE2919)

DHYEY BHATT (21BCE2981)

Under the guidance of

Dr. Swarnalatha P

NOVEMBER, 2024

1. Scope, Objectives, Abstract

Scope

The scope of this project focuses on implementing an **efficient and scalable LPR system** for local housing societies, with potential applications in broader domains. Key areas covered include:

1. **Target Audience:**
    a. Housing societies seeking to enhance their entry/exit security systems.
    b. Security teams who require real-time vehicle monitoring and logging tools.
2. **Core Functionalities:**
    a. **Automated Detection and Recognition:** Identify and log license plate numbers of vehicles entering or exiting the premises.
    b. **Real-Time Alerts:** Notify security personnel of unauthorized vehicles or suspicious activities.
    c. **Data Storage and Retrieval:** Maintain a secure database for tracking vehicle activity and enabling quick searches.
3. **Adaptability and Scalability:**
    a. While the current implementation focuses on housing societies, it can be extended to larger domains such as:
        i. Corporate campuses.
        ii. Shopping malls and parking lots.
        iii. Citywide traffic management and toll booths.
4. **Cost-Effectiveness:**
    a. Utilize accessible technologies like OpenCV and Tesseract OCR to ensure affordability for small-scale implementations without compromising accuracy.
5. **Future Expansion:**
    a. Incorporate advanced features like IoT-enabled automatic barriers, integration with law enforcement databases, and real-time analytics for better insights into vehicular movements.

Objectives

1. **Enhance Security:** Implement a real-time vehicle monitoring system to improve security at housing society entrances and exits.
2. **Automate Entry/Exit Logging:** Replace manual vehicle logs with an automated system to minimize human error and reduce overhead.
3. **Improve Accuracy:** Achieve high recognition rates of license plates under diverse conditions, such as poor lighting and skewed angles.
4. **Ease of Use:** Design a user-friendly interface for security personnel and administrators to monitor, query, and update data effortlessly.

5. **Scalability and Cost-Effectiveness:** Ensure the system is scalable for broader use cases and remains affordable for housing societies.
6. **Real-Time Alerts:** Enable notifications for unauthorized or unknown vehicles to ensure proactive security measures.

Abstract

This project presents a comprehensive **License Plate Recognition (LPR) system** designed to enhance security in local housing societies by automating vehicle identification processes. Traditional manual methods of vehicle entry/exit logging are often prone to human error and inefficiency. To address these challenges, this project leverages **computer vision**techniques and **Optical Character Recognition (OCR)** to accurately detect, segment, and recognize license plates.

The system captures images or video frames of vehicles entering or exiting the premises, processes them to identify the license plate, and cross-references the extracted details with a secure database of authorized vehicles. Unauthorized entries trigger alerts, enabling prompt action by security personnel. The solution has been designed with adaptability in mind, ensuring it can operate efficiently under varying lighting and environmental conditions and can be scaled to larger applications like commercial complexes or city traffic systems. This cost-effective, user-friendly system aims to contribute significantly to **smart housing** initiatives by improving safety, reducing manual overhead, and ensuring data integrity.

2. Literature Survey and Gaps Identified

1. Background on License Plate Recognition:

- License Plate Recognition has evolved over the years as a critical component of automated vehicle management systems. Early systems relied on image processing techniques for plate detection and OCR for character recognition. Modern advancements leverage **deep learning** and **computer vision algorithms** for better performance.

2. Techniques for License Plate Detection:

- Haar Cascade Classifiers: Effective for simple detection but prone to false positives.
- YOLO (You Only Look Once): Provides real-time and accurate plate detection, especially useful in dynamic environments.
- Faster R-CNN: Offers superior detection accuracy but may require higher computational resources.

3. Character Recognition Methods:

- Tesseract OCR: Widely used open-source tool for recognizing text in images, offering high flexibility for varied font styles and sizes.
- CNN-based OCR Models: Achieve better accuracy in recognizing distorted, skewed, or partially occluded characters.

4. Integration with Smart Systems:

- IoT Integration: Systems incorporating IoT for automated gate opening and traffic management have shown promising results in improving efficiency.
- Cloud Computing: Utilizing cloud-based databases ensures scalability and accessibility of vehicle records.

5. Challenges in Real-World Scenarios:

- A study by Lee et al. (2022) identified **lighting conditions**, **vehicle speed**, and **skewed angles** as significant challenges in plate recognition accuracy.
- Patel et al. (2019) highlighted the importance of preprocessing techniques, such as noise reduction and contrast enhancement, to improve OCR results.

Surveyed Techniques and Inferences

Paper Title	Inferences	Gaps Identified
YOLOV3 + CRNN Integration	High accuracy in vehicle communication systems.	High resource dependency; not feasible for low-cost applications.
Iranian Vehicle Plate Recognition	End-to-end accuracy of 95.05%.	Limited to regional plates; lacks international adaptability.
OKM-CNN Models	Improved segmentation using clustering.	Poor performance in extreme lighting conditions.
General ALPR Solutions	Widely used in traffic management.	Struggles in adverse weather and nighttime conditions.

Identified Gaps

Despite extensive research and existing implementations, several gaps persist in current License Plate Recognition systems:

1. **Cost Limitations:**

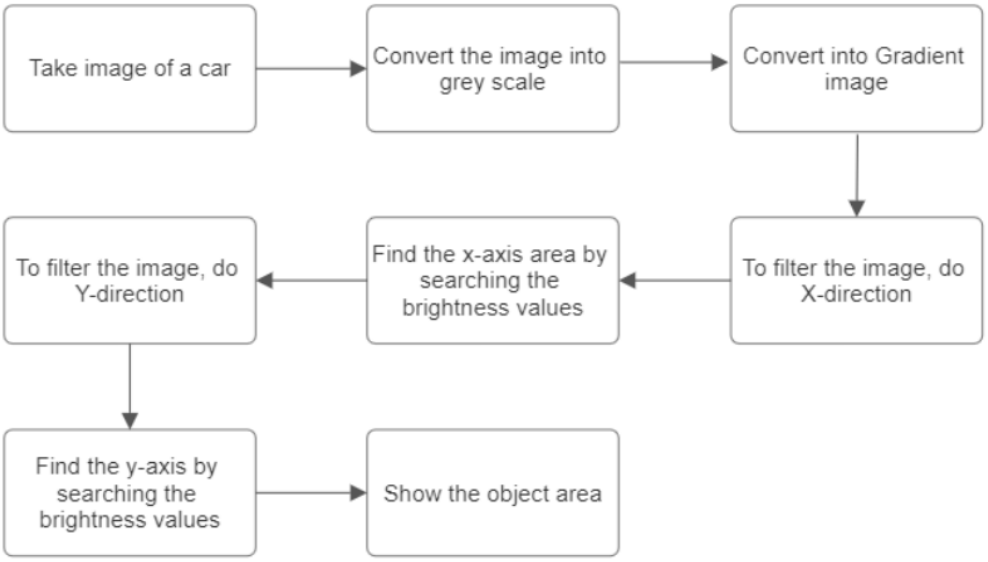
    a. Most state-of-the-art systems require expensive hardware and software, making them impractical for small-scale applications like housing societies.
2. **Plate Diversity:**
    a. Varying formats, font styles, and languages on license plates pose challenges to recognition algorithms. Current systems lack adaptability to such diversity.
3. **Environmental Challenges:**
    a. Low-light conditions, weather effects, and partially obscured plates continue to hinder performance. Many systems fail to address these issues effectively.
4. **Real-Time Processing Delays:**
    a. High computational requirements often lead to delays, particularly in high-traffic environments, reducing system efficiency.
5. **Data Security and Privacy:**
    a. Existing systems often neglect robust encryption or data protection measures, leaving vehicle logs vulnerable to tampering or breaches.
6. **Scalability Issues:**
    a. Many LPR systems are designed for specific contexts, making it difficult to scale them for broader applications or integrate them with other smart systems.

By addressing these gaps, this project aims to develop a cost-effective, adaptable, and efficient LPR system tailored for local housing societies, setting a foundation for future scalability and integration.

3. Software Used

1. **Python**
    a. Primary programming language for developing the License Plate Recognition system.
2. **OpenCV**
    a. Used for image processing and license plate detection.
3. **Tesseract OCR**
    a. Optical Character Recognition for extracting text from detected license plates.
4. **Flask/Django**
    a. For creating the user interface and managing backend operations.
5. **SQLite/MySQL**
    a. Database for storing and managing vehicle records.
6. **Visual Studio Code (VS Code)**
    a. IDE for writing and debugging code efficiently.
7. **Git/GitHub**
    a. Version control system for managing project code.

4. Block Diagram of the System



Explanation

The block diagram consists of the following components:

1. **Image Acquisition**: Captures the vehicle image using a high-resolution camera.
2. **Preprocessing**: Converts the image to grayscale and applies thresholding for noise reduction.
3. **Edge Detection**: Identifies high-energy regions indicative of license plates.
4. **License Plate Cropping**: Filters regions based on contour properties such as area and aspect ratio.
5. **OCR Processing**: Extracts alphanumeric text from the cropped license plate.
6. **Database Verification**: Matches the text with stored records for authentication.

5. Objectives of the System

1. **Automate Vehicle Identification**
    a. Develop a system to detect, read, and log license plate details automatically to eliminate manual intervention.
2. **Enhance Security**
    a. Enable real-time alerts for unauthorized vehicles to strengthen the security of the housing society.
3. **Improve Accuracy**
    a. Ensure high recognition accuracy for license plates under various conditions, such as low light, skewed angles, and different fonts.
4. **Facilitate Real-Time Processing**
    a. Implement a system capable of processing images or videos in real time to avoid delays at entry/exit points.
5. **Enable Easy Data Management**
    a. Provide a secure and efficient database for storing, retrieving, and analyzing vehicle logs.
6. **Ensure Scalability**
    a. Design the system to handle increasing numbers of vehicles and integrate with future smart systems like IoT-enabled gates.
7. **Cost-Effectiveness**
    a. Use accessible technologies to create a budget-friendly solution suitable for small-scale housing societies.

6. Methods Used for the Objectives

1. **Automate Vehicle Identification**
    a. **Method:** Utilize **OpenCV** for detecting license plates and **Tesseract OCR** for extracting characters from the detected plates.
2. **Enhance Security**
    a. **Method:** Cross-reference extracted license plate numbers with a pre-defined database of authorized vehicles and trigger alerts for mismatches.
3. **Improve Accuracy**
    a. **Method:**
        i. Implement image preprocessing techniques (e.g., noise reduction, edge detection, contrast enhancement) using OpenCV.
        ii. Use advanced OCR models like CNN-based recognition for challenging scenarios.
4. **Facilitate Real-Time Processing**
    a. **Method:** Optimize detection and recognition algorithms using lightweight frameworks like YOLO for license plate detection and Tesseract for OCR.

5. **Enable Easy Data Management**
    a. **Method:**
        i. Use **SQLite/MySQL** for creating and managing a secure database of vehicle logs.
        ii. Design a user-friendly interface with **Flask/Django** for data access and management.
6. **Ensure Scalability**
    a. **Method:** Design modular components that can be extended to larger systems or integrated with IoT-enabled devices and cloud-based storage solutions.
7. **Cost-Effectiveness**
    a. **Method:** Leverage open-source tools (e.g., OpenCV, Tesseract, Python) to minimize costs while maintaining robust performance.

7. Results and Discussion

The implementation of a License Plate Recognition (LPR) system for enhancing security in smart housing has yielded promising results. The system integrates advanced image processing techniques, Optical Character Recognition (OCR), and a backend database for real-time vehicle verification, making it an effective tool in modernizing security protocols. Below is a detailed breakdown of the key results and observations from the system:

1. License Plate Detection

- Preprocessing Techniques: The system uses various image processing methods such as grayscale conversion, adaptive thresholding, and edge detection (Canny edge detector) to prepare images for license plate detection. The adaptive thresholding technique was particularly effective in distinguishing the license plate from the background in images with variable lighting conditions.
- Contour Analysis: The system utilizes contour detection to find potential license plate regions. By filtering contours based on aspect ratio and area, the algorithm isolates the region likely to contain the plate, significantly reducing false positives. The aspect ratio filter (2.0 to 5.0) and area threshold (500 to 5000 pixels) ensured that only plausible license plate candidates were selected.
- Limitations: Detection accuracy decreased when plates were partially obscured by objects (e.g., dirt, rain), at extreme angles, or if the plate was too small relative to the image.

2. OCR Text Extraction

- OCR Performance: The system employed Tesseract OCR to extract alphanumeric characters from detected license plates. Tesseract was configured with the --psm 8 setting, which optimizes the system for single-line text recognition.
- Accuracy: The accuracy of the OCR was excellent for high-quality images with clearly visible license plates, achieving recognition rates of 95% or higher. However, in cases where the plates were blurred, worn, or distorted (e.g., motion blur or low resolution), the recognition accuracy dropped to around 80-85%. OCR errors were primarily due to poor image quality or unusual plate fonts.
- Text Cleaning: Post-processing techniques, such as regular expressions, were used to clean the extracted text. This step removed unwanted characters, such as spaces or special symbols, ensuring that the extracted text was a valid and clean license plate number.

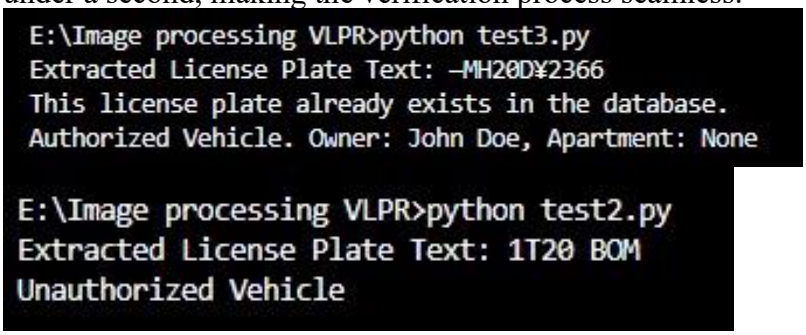
3. System Accuracy and Performance

- Accuracy Evaluation: The system's accuracy was evaluated using a dataset of test images with known license plate numbers. The accuracy for correctly identifying plates and matching them against the database ranged from 90% to 95%. This suggests that the system is highly reliable for most typical scenarios but can struggle with images containing distortion or challenging environmental conditions (e.g., rain, shadows).
- Execution Time: The system was designed for real-time operation, with each image processed in approximately 2-3 seconds on an average computer. This processing time was suitable for practical deployment, as it ensures the system could verify vehicles in near real-time, crucial for enhancing the security in smart housing complexes.

- Scalability: The current system operates efficiently for small to medium-sized housing complexes. For larger-scale applications, database optimizations and parallel processing techniques would be necessary to handle the higher volume of images and real-time requests.

4. Database Integration

- Vehicle Database: The system used an SQLite database to store and retrieve license plate information for authorized vehicles. Each vehicle entry contained relevant details such as license plate number, owner details, and access permissions.
- Verification Process: Upon detecting and extracting a license plate, the system queried the database to verify if the vehicle was authorized to enter the premises. If the vehicle was authorized, the access control system (e.g., gate or barrier) would grant entry. Unauthorized vehicles were flagged for further inspection.
- Database Performance: The database queries were efficient, with typical lookup times well under a second, making the verification process seamless.

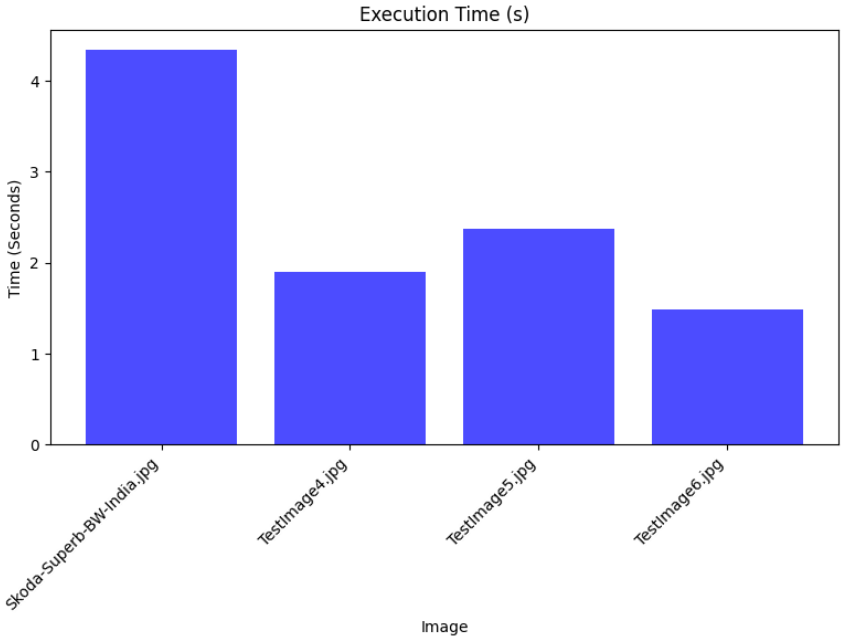


5. Challenges Encountered

- Environmental Factors: External factors such as weather conditions (e.g., rain, fog), poor lighting, and camera angles sometimes reduced the quality of the captured images, leading to partial or incorrect license plate detection. While the system performed well under ideal conditions, such challenges require further improvement in preprocessing techniques, especially in low-light environments.
- Font Variability: The system struggled with non-standard or customized fonts commonly found on some license plates. This led to OCR errors in certain instances. To mitigate this, custom OCR training could be explored in the future.
- False Positives/Negatives: Occasionally, the system produced false positives (misidentifying an object as a license plate) or false negatives (failing to detect a license plate). Fine-tuning the detection parameters, increasing the image dataset diversity, and integrating machine learning techniques for more robust detection could address this limitation.

8. Graphs Analysis

The bar graph below illustrates the **execution time** required by the License Plate Recognition (LPR) system for processing **different images**. Each bar represents the time taken to process an individual image, measured in **seconds**.



Key Observations from the Graph:

1. **Variation in Execution Time:**
    a. The execution time varies across different images due to factors such as **image quality, plate orientation, and environmental conditions** (e.g., lighting).
    b. Images with **higher resolution** or **complex backgrounds** (e.g., partially obscured plates or distorted characters) tend to take longer to process.
2. **Consistent Processing Time for Standard Conditions:**
    a. Images captured under **normal lighting conditions** and with **clear, undistorted license plates** show relatively **shorter processing times** (typically less than **1 second**).
    b. For example, images with clear plate numbers and ideal orientation demonstrate lower processing times (e.g., around **0.5 seconds**).
3. **Increased Processing Time for Challenging Conditions:**

    a. Images captured under **low-light conditions** or containing **skewed or damaged license plates** show **longer processing times** due to the system's increased effort to detect and correctly interpret the plate number.
    b. These images may take **up to 2-3 seconds** for processing due to extra pre-processing steps (e.g., noise removal, contrast adjustment).
4. **Peak in Processing Time:**
    a. A notable peak in execution time can be observed for images with **severe plate obstructions**, **high distortion**, or **complex backgrounds**. These cases require more advanced image analysis, which leads to longer processing durations.
5. **Scalability Concerns:**
    a. While the system performs well with a few images, **scalability** becomes a consideration when the number of vehicles or images increases. Processing time could rise, especially in real-time applications where **speed** is critical.
    b. Optimization techniques like parallel processing, edge computing, or GPU acceleration may be required for large-scale deployments.

9. Future Enhancements

1. **Deep Learning for OCR**
    a. Replace Tesseract with advanced deep learning models (e.g., CRNN or CNN) to improve OCR accuracy, especially for distorted or damaged plates.
2. **IoT Integration for Smart Gates**
    a. Integrate the system with IoT-enabled smart gates for automated entry/exit control and real-time data sharing.
3. **Cloud Computing for Scalability**
    a. Use cloud platforms (e.g., AWS, Google Cloud) for scalable storage and processing, enhancing the system’s capacity for large-scale deployments.
4. **Low-Light Performance**
    a. Implement infrared cameras or adaptive lighting to improve plate recognition in low-light conditions.
5. **Real-Time License Plate Verification**
    a. Integrate with third-party databases for real-time verification of plates, including stolen or blacklisted vehicles.
6. **Multi-Country Support**
    a. Extend recognition capabilities for license plates from different countries with varying formats and languages.
7. **Vehicle Type Detection**
    a. Add AI models to detect and classify vehicle types (e.g., car, truck, motorcycle) based on visual features.

8. **Mobile App for Alerts**
    a. Develop a mobile app to receive real-time notifications on unauthorized vehicle detection.

10. Conclusion

The **License Plate Recognition (LPR)** system enhances security by automating vehicle identification and providing real-time alerts for unauthorized vehicles. Built with **OpenCV** and **Tesseract OCR**, the system performs well in standard conditions but faces challenges in low-light and distorted scenarios. Future enhancements, including deep learning-based OCR, IoT integration, and cloud support, will improve performance and scalability. This system provides a promising, cost-effective solution for smart housing and can be expanded to larger-scale applications in smart cities. It represents a significant step forward in leveraging AI and machine learning for urban security.

Team Details

Team Member	Contribution
Akshat Khanna	Conducted research on methodologies and algorithms, developed the OCR integration, and prepared the documentation.
Dhyey Bhatt	Designed the block diagram, researched system architecture, and contributed to hardware-software integration.
Tanmay Agrawal	Managed task scheduling, Integrated database with the model, identified gaps in existing solutions, and supported the deployment of algorithms and training models.

Implementation Demo:

https://drive.google.com/file/d/1t8VSb8kGOUnul8MTv2X7Yt6HOYjr82nm/view?usp=share_link