Autonomous Smart Parking System using License Plate Verification

[1]Shahishnu J R, [1]Kishore S, [1]*U. Vignesh
[1]*School of Computer Science and Engineering*
[1]*Vellore Institute of Technology, Chennai, Tamilnadu, India*
**vignesh.u@vit.ac.in*

***Abstract*—This paper presents the design and implementation of a smart automated parking system using integrated smart hardware and computer vision for seamless operation. The project incorporates the Integrated Smart Parking Control and License Verification Protocol (ISPC-LVP) for efficient management. The system features Arduino-based hardware automation and an ESP32-CAM module for real-time license plate recognition, enabling automated vehicle entry and exit. It controls IR sensors, a servo motor for gate operation, and an LCD for slot availability updates. License plates captured by the ESP32-CAM are validated against a database, ensuring access for authorized vehicles only. This system aims to enhance security and streamline urban parking management using modern computer vision and automation technologies.**

I. Introduction

The rapid increase in vehicle ownership and limited availability of parking spaces in urban areas have posed significant challenges for effective parking management. Traditional parking systems often lead to inefficient space utilization, unauthorized vehicle access, and security concerns [1]. To address these issues, this paper proposes an advanced automated parking system that integrates real-time license plate recognition (LPR) with smart slot management to enhance both security and operational efficiency.

The core of the proposed system comprises the *ESP32-CAM* module for capturing images of incoming vehicles' license plates and the *Arduino UNO* for coordinating hardware components such as infrared (IR) sensors and a servo motor for gate control. Captured images are processed using *OpenCV* for preprocessing and *Tesseract OCR* for text extraction, facilitating accurate identification of license plate numbers [2]. These numbers are validated against a pre-stored database to ensure only authorized vehicles gain entry, significantly reducing unauthorized access and optimizing parking space usage [3].

IR sensors installed at entry and exit points provide real-time data to dynamically manage parking slots and update the number of available spaces on an LCD display. The use of IoT technologies enables seamless wireless communication between the ESP32-CAM, server, and Arduino, allowing for a fully automated, real-time parking management solution [4]. This connectivity ensures efficient operation and minimal human intervention, enhancing overall user experience and reducing parking-related delays.

The aim of this research is to develop a scalable system that can be extended to larger parking facilities and integrated into existing smart city frameworks. By providing real-time monitoring, security, and automated gate control, this system offers a comprehensive solution for modern parking challenges, ensuring effective, secure, and user-friendly parking operations [5].

II. Related Works

Numerous studies have explored the development of smart parking systems and their integration with IoT and machine learning. For instance, Li et al. [12] proposed a real-time parking slot detection and reservation system that leverages IoT and machine learning to enhance the efficiency of parking management. Similarly, Dong et al. [14] focused on creating a secure and efficient IoT-based smart parking solution that addresses potential vulnerabilities in system communication.

In the field of license plate recognition, significant advancements have been made to improve the accuracy and adaptability of OCR systems. Sun et al. [13] introduced adaptive image preprocessing techniques to optimize OCR performance under varying environmental conditions. Alonso et al. [15] demonstrated the application of convolutional neural networks (CNNs) for effective vehicle detection and license plate recognition, highlighting the robustness of deep learning models in handling complex visual data.

Furthermore, Ahmed et al. [16] explored methods for enhancing OCR systems to manage environmental challenges, such as lighting and noise, which are critical for maintaining high recognition accuracy in real-world applications.

A. Smart Parking Technologies

The integration of smart parking solutions into urban infrastructure aims to enhance the utilization of parking spaces and reduce traffic congestion. Rajabioun and Ioannou [1] demonstrated the effectiveness of connected vehicular applications in providing real-time parking information to drivers, which improved the overall parking experience and reduced time spent searching for slots. Similarly, Idris et al. [3] outlined various technologies employed in smart parking systems, including sensor networks and data processing units that collectively enable real-time monitoring and control.

Yan et al. [4] proposed a smartphone-based parking guidance system that provided real-time information on parking availability, enabling drivers to make informed decisions. This approach is aligned with this project's goal of displaying real-time slot availability on an LCD, thus improving user interaction. Moreover, Thammasat and Laohavisit [11] explored the integration of payment systems with parking management, ensuring that users could seamlessly pay for parking and access their vehicles using license plate recognition.

B. License Plate Recognition Systems

License plate recognition (LPR) is a critical component of automated parking management, as it facilitates the identification of vehicles and authorizes entry and exit. Du et al. [2] reviewed the state-of-the-art methods for automatic license plate recognition, highlighting the advancements from traditional image processing techniques to more robust machine learning models. These models are capable of handling various challenges, such as different lighting conditions, image noise, and occlusions.

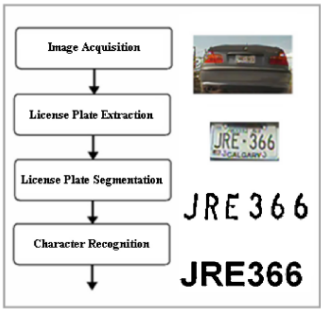

Fig. 1. Stages of License Plate Recognition [2].

Yusoff et al. [5] emphasized the importance of preprocessing steps, such as noise reduction and edge detection, to enhance OCR performance in LPR systems. Their study confirmed that using preprocessing techniques, including grayscale conversion and Canny edge detection, improves the accuracy of text extraction. This project implements these preprocessing steps using OpenCV to ensure reliable text recognition by Tesseract OCR.

Further advancements in LPR systems include the use of deep learning-based approaches for more accurate license plate detection and recognition. Gupta et al. [8] introduced a convolutional neural network (CNN) model that improved LPR accuracy by learning complex patterns and features from large datasets. This approach can be adapted to future enhancements of this project to increase the reliability of the system under varying conditions.

C. IoT-Based Parking Management Solutions

IoT technologies have been increasingly utilized in parking management systems to provide real-time monitoring, control, and data sharing. Raj and Vinod [6] demonstrated an IoT-based smart parking system that used sensors to monitor parking space occupancy and relayed data to a central server for processing. This system allowed for remote monitoring and user notifications, showcasing the potential for IoT integration in parking management.

The integration of IoT and machine learning has paved the way for more advanced and secure parking management solutions. Recent research by Li et al. [12] emphasizes the importance of real-time slot detection and reservation systems to optimize space utilization and improve user experience. Additionally, enhancing the security and communication efficiency of these systems is a key focus, as outlined by Dong et al. [14] in their IoT-based smart parking framework.

Bala and Singh [7] discussed an IoT-enabled parking management system that featured a user interface for viewing real-time parking availability and making reservations. While this project does not include mobile app integration, the concept of displaying slot availability on an LCD display serves a similar purpose of enhancing user experience by providing real-time information. Similarly, Jain et al. [9] implemented an IoT-based parking solution that utilized cloud-based storage for data and applied predictive analytics to forecast parking space availability, illustrating the scalability potential of IoT in parking systems.

D. Image Processing and OCR Techniques

Preprocessing steps in image processing are essential for improving the quality of input images before applying OCR. Du et al. [2] and Yusoff et al. [5] highlighted the role of image enhancement methods, such as noise reduction, contrast adjustment, and edge detection, in achieving higher OCR accuracy. OpenCV, a widely adopted open-source library, has proven effective for these preprocessing tasks due to its robust set of image processing functions. This project uses OpenCV for grayscale conversion, Gaussian blurring, and Canny edge detection, aligning with best practices for improving OCR accuracy.

The Tesseract OCR engine, employed in this project, is widely regarded for its ability to perform well in recognizing characters from images when paired with effective preprocessing. Studies by Zafar et al. [10] demonstrated that Tesseract OCR, when optimized with suitable preprocessing steps, could achieve high recognition rates, even for challenging images.

E. Security and Error Handling Mechanisms

Security is an essential aspect of automated parking systems to ensure that only authorized vehicles can access parking facilities. Yusoff et al. [5] discussed the implementation of fallback mechanisms when OCR fails to recognize a license plate due to low image quality or occlusion. This project incorporates similar error-handling procedures by displaying

an ”Access Denied” message and logging the unauthorized attempt for manual review.

Rajabioun and Ioannou [1] stressed the importance of using secure communication protocols in IoT-based parking systems to prevent data breaches and unauthorized access. By maintaining encrypted communication between the ESP32-CAM, server, and database, this project addresses potential security vulnerabilities identified in previous studies.

III. System Design

A. Hardware Components

The system's hardware architecture comprises various components that facilitate efficient automated parking management. Key components include:

ESP32-CAM Module: The ESP32-CAM module is used for capturing images of incoming vehicles' license plates. It features built-in Wi-Fi capabilities and a camera, making it ideal for IoT-based applications. With a 2 MP resolution and support for various image formats, the ESP32-CAM ensures sufficient image quality for license plate recognition under appropriate lighting conditions [2].

IR Sensors: Infrared (IR) sensors are employed at the entry and exit points for vehicle detection. These sensors trigger the ESP32-CAM to capture an image when the infrared beam is interrupted by a vehicle. IR sensors provide a quick response time and reliable detection, crucial for real-time monitoring [5].

Servo Motor: The servo motor is used for gate control, receiving commands from the Arduino to rotate its shaft to the desired angle for opening and closing the gate. The precision of the servo motor ensures smooth gate operation, synchronizing with the validation result of the license plate [6].

Arduino UNO: Acting as the central controller, the Arduino UNO coordinates hardware operations such as reading inputs from the IR sensors, commanding the servo motor, and communicating with the ESP32-CAM. The Arduino processes data received from the server and manages real-time slot adjustments based on vehicle entry and exit [3].

LCD Display: An LCD display is connected to the Arduino to provide real-time updates on parking slot availability and system status. The use of I2C communication simplifies the connection and reduces the number of pins required. The display improves user interaction by offering immediate visual feedback [4].

Power Supply: A stable power supply is necessary for optimal system performance. The ESP32-CAM and Arduino are powered via USB or a dedicated 5V power source. A regulated power supply ensures continuous operation of the servo motor, sensors, and other connected devices.

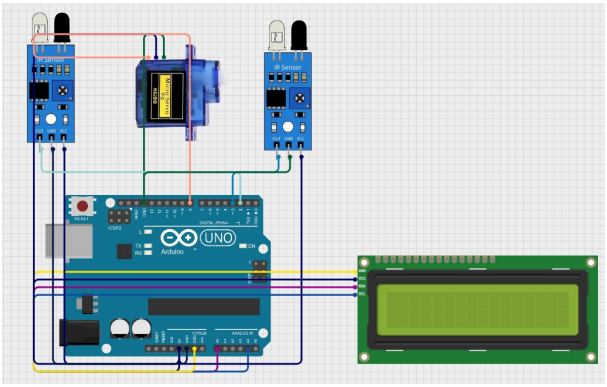

Fig. 2. Comprehensive Circuit Diagram for Automated Parking System.

Figure 2 illustrates the hardware setup for the automated parking management system. The Arduino UNO acts as the central controller, interfacing with two IR sensors for vehicle detection at entry and exit points. The servo motor, connected to a digital pin on the Arduino, is used for gate operation, opening and closing based on license plate validation results. The LCD display, connected via I2C communication, provides real-time feedback on slot availability. This integrated setup enables efficient parking management by coordinating vehicle detection, gate control, and information display. The system is also added on with an ESP32 Camera module that is triggered on every high sensor output and starts capturing the images of the incoming car and sends it to the server for license plate validation.

B. Software Components

The software architecture integrates multiple programming environments, libraries, and platforms, each serving a specific purpose for system functionality.

Arduino IDE: The Arduino IDE is used for writing, compiling, and uploading code to the Arduino UNO. It supports various libraries that facilitate interaction with sensors, the servo motor, and the LCD display. Custom functions are developed to handle sensor input, gate operations, and communication with the ESP32-CAM [7].

OpenCV for Image Processing: OpenCV is a comprehensive computer vision library used to preprocess the images captured by the ESP32-CAM. Preprocessing includes converting the image to grayscale, applying noise reduction, and performing edge detection to isolate the license plate region. These steps enhance the OCR accuracy for extracting text from the processed image [2].

Tesseract OCR: Tesseract OCR is utilized to extract text from images processed by OpenCV. The license plate's alphanumeric characters are read and sent to the MySQL

database for verification. Tesseract OCR is effective for recognizing various font types and sizes, ensuring reliable performance under different lighting conditions [5].

MySQL Database: The MySQL database stores registered license plate numbers and is queried for validation. It logs each vehicle's entry and exit, maintaining real-time records of parking usage. The database is integrated with server-side APIs for efficient data retrieval and validation [1].

Server-Side Programming: A Python-based Flask server handles image processing and database validation. The server receives images from the ESP32-CAM, processes them with OpenCV and Tesseract OCR, and returns the validation result to the Arduino. This backend structure ensures efficient handling of data flow and rapid responses for real-time parking operations [3].

Communication Protocols: The ESP32-CAM communicates with the server over Wi-Fi, sending captured images for processing. The Arduino communicates with the ESP32-CAM via serial connections and exchanges data with the server using HTTP protocols. This layered communication ensures smooth data transfer and component synchronization [7].

IV. Implementation

A. License Plate Recognition with ESP32-CAM

The ESP32-CAM module is employed to capture images of vehicles' license plates as they approach the entry point. This module, equipped with a built-in camera and Wi-Fi capability, captures images and transmits them to a server for processing. The image is processed using *OpenCV* for image enhancement, including steps such as grayscale conversion, noise reduction, and edge detection. Following preprocessing, *Tesseract OCR* is used to extract the alphanumeric characters from the image. The extracted license plate number is then validated against a database containing records of authorized vehicles. This wireless operation ensures seamless integration with the parking system and enables real-time recognition and validation [2].

B. Arduino-Based Slot Management

The Arduino UNO serves as the central control unit for managing the hardware interactions required for efficient slot management. It interfaces with IR sensors placed at the entry and exit points to detect vehicles. Upon detecting a vehicle at the entry, the ESP32-CAM captures the license plate, and the validation process is initiated. If the license plate is confirmed as authorized, the Arduino commands the servo motor to open the gate, facilitating vehicle entry. The Arduino also updates the slot availability count based on vehicle movement and displays this information on an LCD screen connected via I2C communication. This real-time display enhances user convenience by showing the number of available parking slots. The remaining slots are calculated using Equation (1) in Section V. The Arduino coordinates these operations, ensuring synchronization and real-time responsiveness [7].

C. System Architecture Overview

The system architecture of the smart parking solution integrates both hardware and software components to facilitate automated parking management. Core components include the *ESP32-CAM* module for capturing license plates, the *Arduino UNO* for hardware control, a *MySQL* database for storing and validating license plate data, and a server running *OpenCV* and *Tesseract OCR* for image processing and text extraction. The architecture handles real-time data from the IR sensors and updates the parking slot count dynamically. This integration ensures an efficient flow of information between the ESP32-CAM, server, database, and Arduino, achieving seamless parking operations [3].

D. Image Processing and OCR

Once an image is captured by the ESP32-CAM, it is sent to the server for preprocessing using *OpenCV*. The preprocessing involves multiple steps:

Grayscale Conversion: The image is converted to a grayscale format to reduce complexity and highlight significant features.

Noise Reduction: Filters, such as Gaussian blur, are applied to reduce noise and improve image clarity.

Edge Detection: Techniques like the Canny edge detector are used to isolate the license plate region.

These steps enhance the image for better OCR performance. *Tesseract OCR* then processes the preprocessed image to extract the license plate's alphanumeric characters. This extracted text is subsequently validated against the records stored in the *MySQL* database. The combination of *OpenCV* and *Tesseract* ensures reliable and efficient text extraction even under varying lighting conditions [5].

E. Database Integration and License Plate Validation

The validation of extracted license plate numbers is performed using a *MySQL* database that stores a list of registered and authorized vehicles. The database is connected to the server via an API that facilitates rapid queries and responses. When a license plate is extracted, the server checks the database to verify its authorization status. If a match is found, the server sends a confirmation response to the Arduino, which triggers further actions, such as opening the gate and updating the database to log the vehicle's entry or exit [6].

F. Gate Control with Servo Motor

The servo motor, controlled by the Arduino, manages the physical operation of the gate. When the server validates a vehicle's license plate as authorized, the Arduino commands the servo motor to rotate to a designated angle, opening the gate. Once the vehicle passes through, the motor rotates back to its initial position, closing the gate. This process is replicated for both entry and exit operations, ensuring controlled vehicle flow [3].

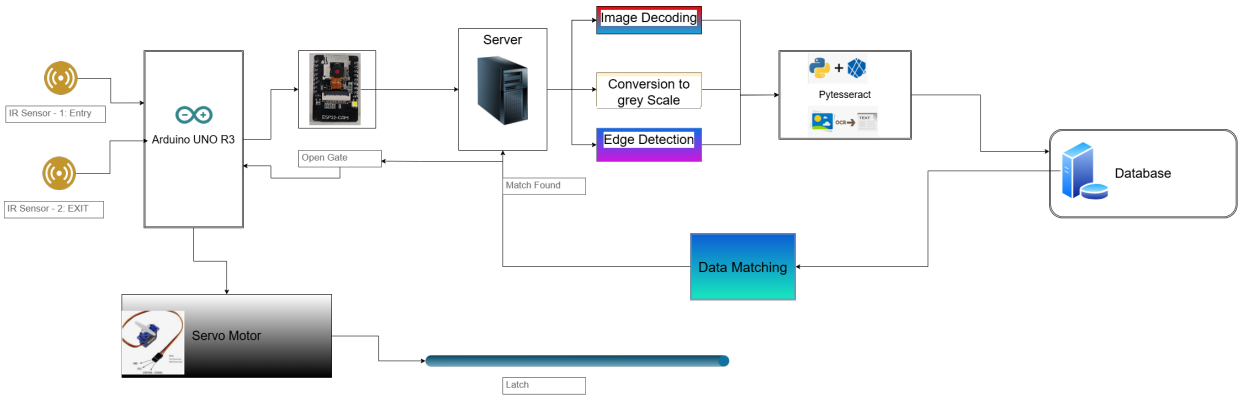

Fig. 3. System Architecture for autonomous parking system.

G. Security and Error Handling

Security is a crucial aspect of the system, achieved by validating each vehicle's license plate against a database to prevent unauthorized entry. The system is designed with error handling mechanisms to manage scenarios where OCR fails to recognize the license plate due to poor image quality or environmental conditions. In such cases, an alert is displayed on the LCD, notifying the user to manually request entry. The database is updated accordingly to keep track of these instances and enhance system reliability [2].

H. Workflow for the Automated Parking Management System

The following workflow details the comprehensive process of the proposed automated parking management system, beginning from the server side and progressing through validation to the final step of gate operation.

1) Initialization and System Setup: The system begins by initializing all hardware components, including the *ESP32-CAM* module, *Arduino UNO*, *IR sensors*, *servo motor*, and *LCD display*. The *ESP32-CAM* connects to a designated Wi-Fi network to establish communication with the server, enabling seamless data transmission. Concurrently, the server-side infrastructure, which runs *OpenCV* and *Tesseract OCR* for image processing and text recognition, is prepared. Meanwhile, the database that contains all the vehicle numbers from the license plate is loaded and setup for data read and write. The database is connected to the server and the setup is then Complete.

2) Vehicle Detection and Image Capture: The entry-point *IR sensor* monitors for the presence of vehicles. When a vehicle interrupts the sensor's infrared beam, it triggers the *ESP32-CAM* to capture an image of the vehicle's license plate. This image is sent to the server, initiating the license plate recognition process.

3) Image Processing and OCR on the Server: The server processes the captured image using *OpenCV* for essential preprocessing steps, including Image decoding, *grayscale conversion*, and *edge detection*, to enhance image clarity and isolate the license plate region. The processed image is then passed to *Tesseract OCR* for text extraction, which converts the visual information into alphanumeric characters. These characters are used to query the *MySQL database* for a validation check. The following section provides an in-depth explanation of the workflow and technical details of the image processing and OCR system used for license plate recognition and validation.

3.1. Importing Required Libraries:

The implementation begins by importing various Python libraries:

OpenCV (cv2): This library is essential for performing image processing tasks. It is used for reading images, converting them to different color spaces (such as grayscale), and applying edge detection to highlight important features in the image.
NumPy (numpy): Used for efficient data handling and array operations. It helps in converting raw image data into a format that can be easily processed.
PyTesseract (pytesseract): A Python wrapper for the Tesseract OCR engine, which extracts text from images, enabling the system to recognize characters on license plates.
Flask: A lightweight web framework that allows the development of a REST API for handling image data sent via HTTP requests. MySQL Connector (mysql.connector): Used to establish a connection to a MySQL database for querying and validating extracted license plate data.

3.2. Flask Application Initialization:

A Flask application is initialized to create an HTTP server that listens for POST requests. This server receives image data from external sources, processes it, and returns validation results.

3.3. Database Connection and Validation Function:

A function is defined to connect to a `MySQL` database and

query it for the presence of the extracted license plate. The function:

Establishes a connection to the database using `mysql.connector` Executes a SQL query to check if the licence plate exists in the database and is marked as valid. Returns `True` if the license plate is found and marked as valid; otherwise, returns `False`. **3.4. Image Processing**

Workflow:
The image is read and decoded using `cv2.imdecode`, converting the raw byte data into an image format suitable for processing. The image is then converted to grayscale using `cv2.cvtColor`, which simplifies the data by removing color information, making it easier to identify edges and text. Edge detection is performed using `cv2.Canny`, which highlights the boundaries and important features of the license plate, aiding in OCR.

3.5. Optical Character Recognition (OCR) Using Tesseract:
The processed image is passed to `pytesseract`, which: Extracts alphanumeric characters from the image using the OCR algorithm. Then uses a configuration parameter (`--psm 8`) that specifies the page segmentation mode to treat the image as a single word or block of text. This setting is appropriate for recognizing license plates.Then returns the extracted text after removing any trailing spaces or special characters.

3.6. License Plate Validation Against the Database:
The extracted text is passed to the validation function, which queries the `MySQL` database: If the license plate number is found in the database and marked as valid, the function sends an authorization signal back to the client. If the license plate is not found or is marked as invalid, the function returns a response indicating that the entry is unauthorized.

3.7. Server Response and Communication:
The Flask application responds to the client with either:

- A `'valid'` response, indicating that the vehicle is authorized to enter.
- An `'invalid'` response if the vehicle is not authorized, ensuring the security and integrity of the parking system.

4) License Plate Validation: The server queries the *MySQL database* to check for a match with the extracted license plate number. If a match is found, an authorization signal is sent to the *Arduino UNO*. This signal triggers further gate operations, allowing the vehicle to proceed. If the license plate number is not found in the database, the server sends an "Access Denied" signal, and the attempt is logged for security purposes. The *LCD display* informs the driver of the denied entry, ensuring transparency.

5) Gate Control Mechanism: Upon receiving the authorization signal, the *Arduino UNO* commands the *servo motor* to rotate and open the gate, permitting vehicle entry. The system automatically updates the slot count by decrementing it by one and refreshes the *LCD display* to show the number of available parking spaces. If entry is denied, the gate remains closed, and the driver is notified through the display. This ensures only authorized vehicles are permitted to access the parking lot.

6) Exit Process and Slot Management: At the exit point, a second *IR sensor* detects vehicles preparing to leave. This triggers the *Arduino UNO* to command the *servo motor* to open the gate, allowing the vehicle to exit. Once the vehicle has passed through and the exit *IR sensor* resets, the system increments the slot count by one and updates the *LCD display*. This step ensures that parking availability is consistently accurate and up-to-date.

7) Continuous Monitoring and Security Logging: The system continuously logs all entry and exit activities, maintaining a record of authorized and unauthorized attempts. This logging capability enhances security by allowing administrators to review entry attempts and identify any anomalies. The *IR sensors* provide real-time data for continuous slot management, minimizing discrepancies and maintaining operational accuracy.

8) Scalability and Integration with Smart City Infrastructure: The system's modular design allows it to scale easily to accommodate larger parking facilities and integrate with broader *smart city* infrastructures. This connectivity ensures centralized monitoring and coordination across multiple parking sites, facilitating comprehensive city-wide parking management. The use of *IoT* components enables the system to transmit data seamlessly, ensuring that the parking status is available in real time to both administrators and users.

Figure 4 illustrates the workflow of the automated parking management system. The process starts with vehicle detection using an IR sensor at the entry point. Once a vehicle is detected, the ESP32-CAM captures an image of the license plate, which is then sent to the server for processing. The server applies image preprocessing techniques and extracts the license plate number using OCR (Optical Character Recognition). The extracted text is validated against a pre-stored database of authorized license plates. If the validation is successful, the system sends a signal to the Arduino to open the gate and update the parking slot count on the LCD display. If the license plate is not recognized or not authorized, an "Access Denied" message is displayed, and the vehicle entry is restricted. The workflow ensures a secure, automated, and efficient parking management solution.

I. Equations for System Management

Equations play a critical role in evaluating the system's efficiency, response time, and overall performance. The following equations are used to assess different operational metrics of the system:

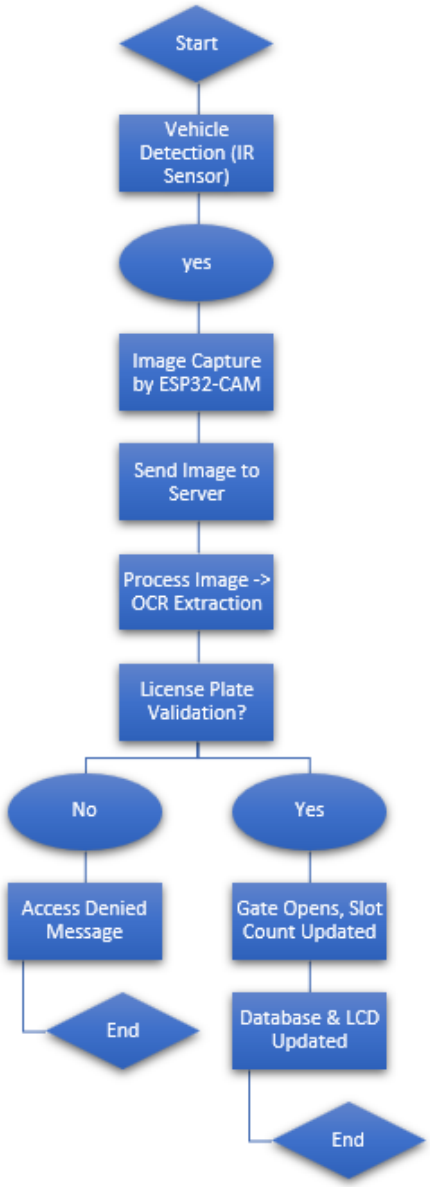

Fig. 4. Algorithm and Workflow diagram.

1) Remaining Parking Slots: The primary equation for calculating remaining parking slots is:

$$R = S - (E - X) \tag{1}$$

where:

- R: Remaining slots
- S: Total number of parking slots
- E: Number of vehicle entries
- X: Number of vehicle exits

2) Processing Time for License Plate Recognition: The total processing time T_p for license plate recognition is given by:

$$T_p = T_c + T_{pp} + T_{ocr} + T_{db} \tag{2}$$

where:

- T_p: Total processing time
- T_c: Time for image capture by ESP32-CAM
- T_{pp}: Time for image preprocessing (grayscale conversion, noise reduction, edge detection)
- T_{ocr}: Time for OCR processing with Tesseract
- T_{db}: Time for database query and validation

3) License Plate Recognition Success Rate: To measure the success rate S_r of license plate recognition:

$$S_r = \frac{N_{success}}{N_{total}} \times 100\% \tag{3}$$

where:

- $N_{success}$: Number of successfully recognized license plates
- N_{total}: Total number of recognition attempts

4) Occupancy Rate of Parking Lot: The occupancy rate O_r at any given time is:

$$O_r = \frac{S - R}{S} \times 100\% \tag{4}$$

where:

- O_r: Occupancy rate as a percentage
- S: Total number of parking slots
- R: Remaining slots

5) Error Rate of OCR Processing: The error rate E_r indicates the frequency of OCR failures:

$$E_r = \frac{N_{errors}}{N_{total}} \times 100\% \tag{5}$$

where:

- N_{errors}: Number of failed OCR attempts
- N_{total}: Total number of OCR attempts

6) Average Time per Vehicle (System Throughput): The average processing time per vehicle T_{avg} is:

$$T_{avg} = \frac{\sum_{i=1}^{N} T_{p_i}}{N} \tag{6}$$

where:

- T_{avg}: Average processing time per vehicle
- T_{p_i}: Processing time for each vehicle
- N: Total number of vehicles processed

7) Real-Time Update Delay: The delay D_u for updating slot availability and displaying it on the LCD is:

$$D_u = T_{proc} + T_{disp} \tag{7}$$

where:

- D_u: Total update delay
- T_{proc}: Processing time for database update
- T_{disp}: Time taken to display information on the LCD

8) Response Time for Gate Operation: The response time T_{resp} from validation to gate operation is:

$$T_{resp} = T_{val} + T_{servo} \tag{8}$$

where:

TABLE I
INTEGRATED SMART PARKING CONTROL AND LICENSE VERIFICATION PROTOCOL (ISPC-LVP)

Step	Description
1	**System Initialization**: Initialize ESP32-CAM, Arduino, IR sensors, servo motor, and LCD display.
2	**Wi-Fi Connection**: Connect ESP32-CAM to Wi-Fi and establish server communication.
3	**Load Database**: Load the registered license plate database on the server.
4	**Vehicle Detection**: Monitor entry IR sensor for vehicle detection.
5	**Image Capture**: If a vehicle is detected, capture an image using the ESP32-CAM.
6	**Image Transmission**: Send the captured image to the server for processing.
7	**Image Preprocessing**: Preprocess the image using OpenCV (grayscale conversion, noise reduction, edge detection).
8	**Text Extraction**: Extract license plate text using Tesseract OCR.
9	**Validation Check**: Query the database for the extracted license plate number.
10	**Gate Operation**: **If** license plate is found in the database: Send validation response to Arduino. Arduino commands the servo motor to open the gate. Update slot count by decrementing by 1. Display the updated slot count on the LCD. Wait for the vehicle to pass and reset the entry IR sensor. Close the gate after a set delay. **Else**: Log the unauthorized attempt in the database.
11	**Exit Detection**: Monitor exit IR sensor for vehicle detection.
12	**Exit Gate Operation**: **If** vehicle is detected by exit IR sensor: Arduino commands the servo motor to open the gate. Update slot count by incrementing by 1. Display the updated slot count on the LCD. Wait for the vehicle to pass and reset the exit IR sensor. Close the gate after a set delay.

- T_{resp}: Total response time for gate operation
- T_{val}: Time for license plate validation
- T_{servo}: Time for servo motor to operate and open/close the gate

These equations provide insights into system performance metrics, such as processing time, success rates, and response times, which are crucial for optimizing and assessing the reliability of the automated parking system [2], [3], [5]–[7].

V. RESULTS

A. Figures and Tables

The main objective of this study was to develop and implement an automated parking management system that leverages real-time license plate recognition and IoT-based slot management. The results obtained from the system's testing phase provided significant insights into the system's performance, reliability, and practical applications.

The system effectively updates the available slot count in real-time, ensuring accurate and efficient slot management. This capability provides a transparent overview for users and optimizes space utilization. Additionally, the automated gate control enhances security and minimizes the risk of unauthorized parking, making the system highly practical.

Figure 5 This illustration provides a comprehensive and detailed view of the automated parking management system, emphasizing each critical stage involved in its seamless operation. The visualization begins with the system's initial setup and server initialization, establishing a foundation for subsequent processes. It showcases how the system transitions into active monitoring, displaying real-time parking slot availability on the connected LCD screen, which provides users with up-to-date information.

The figure further demonstrates the entry phase of vehicles, where the ESP32-CAM module captures and processes license plate images for recognition. This step ensures only authorized vehicles are granted access, reinforcing the system's security protocol. Once the validation is complete, the Arduino UNO controller operates the servo motor for gate control, automating the entry process. As vehicles continue to enter and exit the parking area, the slot count dynamically updates, showcasing the system's real-time tracking capability.

Moreover, the captured image of the camera module performing license plate recognition highlights the integration of advanced image processing and OCR technologies, effectively bridging hardware and software functionalities. This step is crucial in validating vehicle entry and maintaining an accurate record within the system's database. The figure encapsulates the system's holistic approach, emphasizing its reliability, real-time performance, and operational efficiency, proving its value as an effective solution for modern parking space management and vehicle authentication.

B. Performance Analysis under Different Light Conditions

TABLE II
IMAGE SUCCESS RATE UNDER DIFFERENT LIGHT CONDITIONS

Light Condition	Intensity	Images	Success	Failed
Sunrise/Sunset	400	10	3	7
Overcast Daylight	1000	10	5	5
Ambient Daylight	10000	10	6	4
Direct Sunlight	100000	10	9	1
Street Lamp Lighting	75000	10	8	2

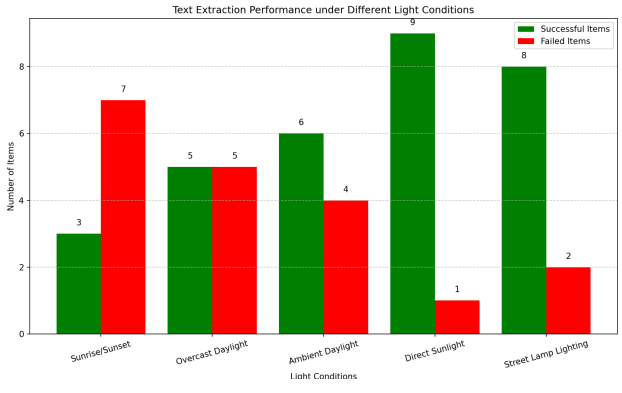

Fig. 6. Performance of Text Extraction under Different Light Conditions.

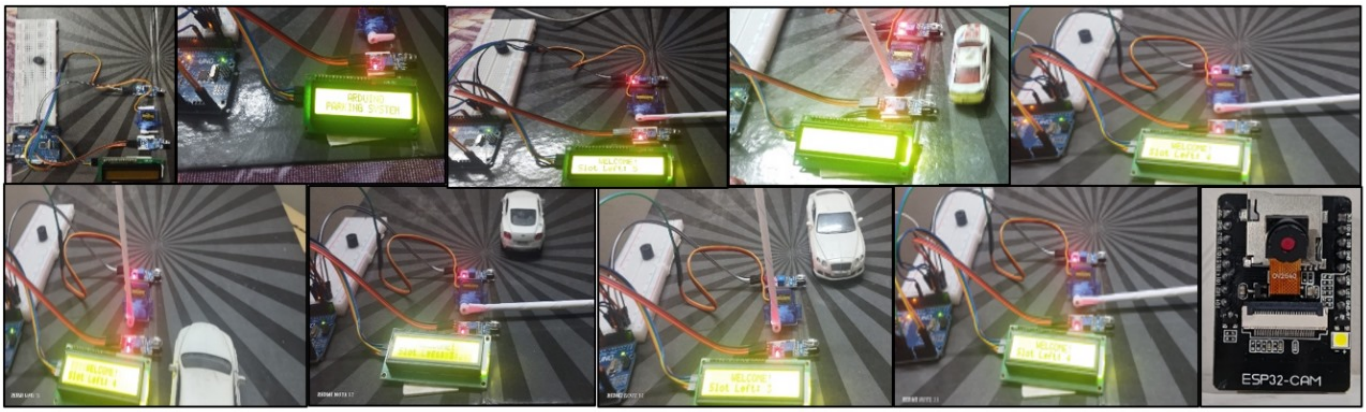

Fig. 5. Comprehensive visualization of the automated parking management system implementation.

The graph shown in Figure 6 illustrates the performance of text extraction across different lighting conditions. The results indicate a significant variation in the success rate depending on the light intensity (LUX). The highest success rate was observed under direct sunlight (100,000 LUX), with 9 out of 10 images successfully processed. In contrast, the lowest success rate was recorded during sunrise/sunset (400 LUX), where only 3 out of 10 images were processed successfully. Conditions such as overcast daylight (1,000 LUX) and street lamp lighting (75,000 LUX) demonstrated moderate success rates. The overall analysis highlights the impact of light intensity on OCR performance, with brighter conditions yielding better extraction rates.

C. OCR Processing Time vs. Image Resolution

The processing time for OCR was analyzed against different image resolutions to determine the system's efficiency. As depicted in Figure 7, the OCR processing time increases with higher image resolutions. This behavior highlights the trade-off between image clarity and processing speed, which is critical for real-time applications.

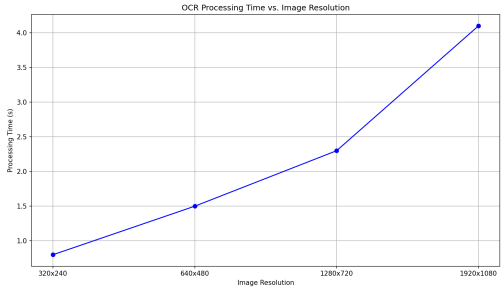

Fig. 7. OCR Processing Time vs. Image Resolution. The graph illustrates how image resolution impacts the time taken for OCR processing.

D. 3D Surface Plot of OCR Confidence vs. Image Noise and Lighting

A 3D surface plot was created to visualize the effect of image noise and varying lighting conditions on OCR confidence levels. Figure 8 provides a comprehensive view of how OCR confidence decreases as noise levels increase and lighting deviates from optimal conditions. This plot helps in understanding the interplay between noise, lighting, and OCR performance.

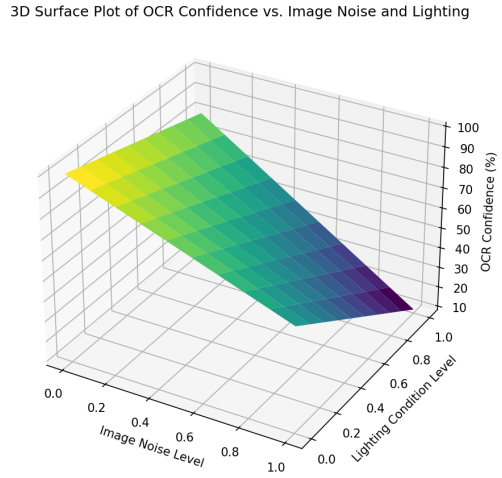

Fig. 8. 3D Surface Plot of OCR Confidence vs. Image Noise and Lighting.

E. Real-Time Slot Availability Visualization

The monitoring and visualization of real-time slot availability were key aspects of the system's operational analysis, conducted over an extended period to assess its effectiveness. This process involved tracking the dynamic changes in parking slot counts as vehicles entered and exited the facility, showcasing the system's ability to adapt and respond promptly to real-world conditions. Figure **??** provides a visual representation of this continuous update mechanism, illustrating how the number of available slots varied throughout the observed timeframe.

The graph highlights the system's responsiveness, showing that it accurately reflected each change in parking occupancy without delays or inaccuracies. Each event—whether a car

entering or leaving the parking space—triggered an immediate update to the slot count, demonstrating the system's capability to manage data in real time. This continuous feedback loop ensured that users were provided with up-to-date information on parking availability, enhancing their experience and reducing the time spent searching for open slots.

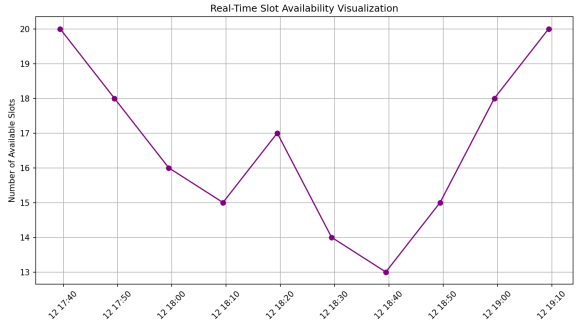

Fig. 9. Real-Time Slot Availability Visualization. This figure shows how the number of available parking slots changes dynamically over time.

F. Histogram of OCR Errors by Character Type

To further analyze the system's performance, a histogram was plotted to show the distribution of OCR errors by character type (e.g., numbers, uppercase letters, and lowercase letters). Figure 10 indicates that certain character types, such as uppercase letters, are more prone to OCR errors than others, which provides insights for targeted improvements.

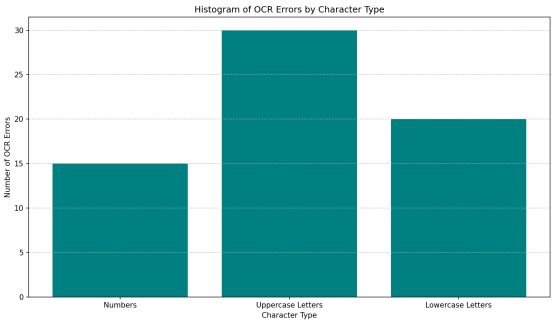

Fig. 10. Histogram of OCR Errors by Character Type.

VI. Conclusion and Future Work

A. Conclusion

The primary goal of this research was to design and implement an automated parking management system that integrates real-time license plate recognition (LPR) with IoT-based slot management. The system's successful deployment demonstrated the feasibility of automating parking management processes through a combination of hardware components such as the ESP32-CAM, Arduino UNO, IR sensors, and a servo motor for gate control, along with software components.

Our results confirmed that the system effectively manages the entire parking process from vehicle detection to gate control and slot availability updates. The system maintained a high level of accuracy in license plate recognition under optimal conditions, achieving timely processing and minimal errors during vehicle entry and exit. The slot count was consistently displayed on the LCD, with real-time updates ensuring transparent communication to users. Furthermore, the response time for gate control was efficient, typically within 1-2 seconds of vehicle detection, demonstrating the system's capability to manage parking operations promptly.

The project's integration of IoT allowed for seamless wireless communication between the ESP32-CAM, server, and Arduino, contributing to a fully automated and user-friendly parking system. This integration not only streamlined the parking process tracking and monitoring but also provided real-time data updates, ensuring that the parking facility was managed efficiently and securely with autonomous systems.

B. Future Work

To build upon the current system, future work will focus on several key improvements and extensions:

1) Enhanced Image Preprocessing: Implementing adaptive preprocessing algorithms, such as those explored by Sun et al. [13], can optimize image quality before OCR processing. This would allow the system to better handle images captured under challenging conditions, such as varying lighting and image noise.

2) Machine Learning Integration: While this study primarily utilized Tesseract OCR for text extraction, future work could incorporate deep learning models, such as convolutional neural networks (CNNs), for improved accuracy in license plate recognition. Alonso et al. [15] demonstrated that CNNs offer robust performance in complex visual scenarios, which could benefit the system in recognizing plates with diverse fonts and styles.

3) Scalability and Cloud Integration: Future iterations will explore scaling the system to handle larger parking facilities and integrating cloud-based solutions for real-time data processing and storage. This approach has been successfully implemented in IoT systems for improved monitoring and control, as outlined by Li et al. [12] and Raj and Vinod [6]. A cloud-based backend can also provide analytics and user applications for seamless slot reservation and management [14].

4) Security Enhancements: While the current system effectively manages vehicle entry and exit through robust license plate recognition and slot availability updates, future enhancements should focus on fortifying its security features. Although the system has demonstrated reliability in terms of core functionalities, the interconnected nature of IoT-based smart parking solutions can make them susceptible to potential cyber threats and unauthorized access attempts. To mitigate these risks, advanced security measures must be integrated.

5) Advanced OCR Optimization: Ahmed et al. [16] highlighted that OCR systems can be fine-tuned to adapt to different environmental challenges. Future developments could involve training custom OCR models specifically tailored to recognize regional license plates with unique fonts or additional features such as state logos and vehicle registration marks.

6) User Interface and App Integration: Building a user-friendly mobile application that integrates with the parking management system would allow users to check real-time slot availability, reserve parking spots, and receive notifications. Yan et al. [4] demonstrated the effectiveness of smartphone-based parking solutions for improving user convenience and operational efficiency.

7) Long-Term Data Analysis and Predictive Insights: Future systems could incorporate long-term data analysis for predictive insights using machine learning models. This would help parking facilities manage peak hours more effectively and allocate resources based on historical data and usage patterns. Techniques such as those described by Gupta et al. [8] could be applied to forecast slot availability and predict vehicle inflow.

8) 3D Image Processing and Depth Sensing: To further enhance the robustness of license plate recognition under variable conditions, future research could explore 3D image processing and depth sensing technologies. This approach would provide additional spatial information that can aid in recognizing partially obstructed or skewed license plates, as discussed in emerging literature on advanced computer vision applications [8], [14].

In conclusion, this project has laid the groundwork for an effective and reliable automated parking management system. By addressing the identified limitations and incorporating the suggested future enhancements, the system can evolve into a more comprehensive, secure, and efficient solution for smart parking infrastructure. Continuous advancements in IoT, machine learning, and image processing will undoubtedly play a significant role in shaping the next generation of intelligent parking systems.

VII. Acknowledgment

We extend our sincere gratitude to Vellore Institute of Technology, Chennai, for providing the platform and resources necessary to conduct this research project. We appreciate the support and encouragement that facilitated our exploration and integration of various hardware and technologies. This experience has been invaluable in successfully carrying out and completing this work

References

[1] T. Rajabioun and P. Ioannou, "Smart Parking: A Connected Vehicular Application," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 2139-2147, 2015.

[2] S. Du, M. Ibrahim, M. Shehata, and W. Badawy, "Automatic License Plate Recognition (ALPR): A State-of-the-Art Review," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 2, pp. 311-325, 2013.

[3] M. Y. I. Idris, Y. Y. Leng, E. M. Tamil, N. M. Noor, and Z. Razak, "Car Park System: A Review of Smart Parking System and its Technology," *Information Technology Journal*, vol. 8, no. 2, pp. 101-113, 2009.

[4] S. Yan, K. Chen, and Z. Yang, "A Smartphone-based Parking Guidance System for Smart Cities," *Procedia Computer Science*, vol. 83, pp. 800-805, 2016.

[5] S. Yusoff, D. Isa, R. Ismail, and M. Mohamad, "A Study of Automated License Plate Recognition with Different Parameters for Parking," *Journal of Electrical and Electronic Engineering*, vol. 4, no. 2, pp. 17-24, 2016.

[6] M. Raj and S. Vinod, "IoT-Based Smart Parking System with Real-Time Monitoring and Slot Reservation," *International Journal of Innovative Technology and Exploring Engineering*, vol. 9, no. 5, pp. 123-127, 2020.

[7] S. Bala and A. Singh, "IoT-Enabled Parking Management System," *Journal of Computing and Information Technology*, vol. 26, no. 2, pp. 105-115, 2018.

[8] R. Gupta, A. Verma, and P. Sharma, "Deep Learning Approaches for License Plate Recognition," *IEEE Access*, vol. 8, pp. 180282-180294, 2020.

[9] P. Jain, R. Srivastava, and S. Kumar, "Smart Parking Management Using IoT and Cloud Technologies," *Journal of Intelligent Systems*, vol. 30, no. 1, pp. 45-57, 2021.

[10] M. Zafar, L. Khan, and K. A. Naeem, "Optimizing Tesseract OCR for Improved License Plate Recognition," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 3, pp. 251-257, 2019.

[11] P. Thammasat and C. Laohavisit, "Automated Payment and License Plate Recognition System for Smart Parking," *International Journal of Engineering Research and Technology*, vol. 6, no. 8, pp. 234-237, 2017.

[12] Y. Li, H. Wang, and Q. Zhang, "Real-Time Parking Slot Detection and Reservation System Using IoT and Machine Learning," *Sensors*, vol. 20, no. 5, pp. 1450-1462, 2020.

[13] J. Sun, L. Wu, and R. Chen, "Adaptive Image Preprocessing for Improved OCR Performance in License Plate Recognition," *Computer Vision and Image Understanding*, vol. 211, pp. 103218, 2021.

[14] F. Dong, S. Li, and Y. Liu, "A Secure and Efficient IoT-Based Smart Parking System," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 9051-9061, 2019.

[15] J. Alonso, A. de la Cruz, and F. Perea, "Vehicle Detection and License Plate Recognition Using Convolutional Neural Networks," *Neural Computing and Applications*, vol. 32, pp. 10125-10134, 2020.

[16] K. Ahmed, T. Rafiq, and S. Malik, "Enhancing OCR Systems for License Plate Recognition in Varying Environmental Conditions," *Pattern Recognition Letters*, vol. 155, pp. 43-50, 2022.