VEHICLE LOGGING SYSTEM WITH AUTOMATIC NUMBER PLATE RECOGNITION

S Manjula[1], Amith Kumar[2], Bebikananda Waikhom[3], Triveni Wahengbam[4]
JSS Science and Technology University (Formerly known as SJCE)
SJCE Campus, Manasa Gangotri Mysuru, Karnataka 570006, India

Abstract: With an increase in urbanization and the rise in the number of automotive vehicles, the need for an autonomous vehicle logging and Parking Management System is becoming vital for the urban landscape. India is the third-largest road network with booming megacities and an ever-increasing amount of private automobiles. This induces a need for efficient parking systems in urban areas. We propose a layered solution architecture for a Parking Management System. Our proposed solution is a cloud-based application providing an effective user interface. It automates the process of parking management using machine learning techniques for real-time automated number plate recognition, analytics, etc. The primary focus of this project is to build modularized and layered architecture for parking management and an Automated Vehicle logging mechanism.

Keywords: Smart Parking, Check-in, Check-out, Real-time database, Notification.

I. INTRODUCTION

In cities, large scale car parking areas with hundreds/thousands of spaces are getting more and more common with the increasing vehicle population. The existing "pay and park" parking spaces are under-used, mismanaged and under-developed leading to an ineffective parking experience for the driver (parking consumer), and an unprofitable and ineffective scheme for the parking space owner. This deficiency in the existing parking infrastructure leads to accidents, thefts, mismanagement, losses, reduced transparency in the parking process, ineffective space management, etc. People now face problems parking their vehicles. The person parking in these parking-spaces has the concerns regarding the security, transparency and reliability of the existing parking infrastructure. The main idea of this project is to create a system that effectively handles this deficiency in the parking infrastructure by using modern technologies like machine learning, scalable microservices, effective android interfaces, connected devices to increase the level of automation and efficiency.

II. BACKGROUND AND MOTIVATION

The existing methods of parking management use human interaction in detecting and logging number plates. The existing methods mainly depend on human interaction to generate the bills, and it is time-consuming. They need to maintain data of all the vehicles by physically entering the information. This is error-prone and less effective than an automated solution. There have been a few solutions proposed and researched and prototyped by individuals and companies that have explored the using ANPR in real-time for parking management. In addition to that, there have been some solutions being developed and tested in artificial environments for the same. Some drawbacks are lesser transparency in the parking system, and precious time wasted due to the inconvenient and ineffective methods at parking places and more consumption of fuel while idling or driving around the parking places, users have little knowledge about nearby parking spaces, the uncertainty due to the lack of connected infrastructure and a platform, proves to be less profitable to the parking space owners and the parking space consumers. We propose a solution that is a fully automated computer vision-based smart parking system and implement an interface for the driver providing real-time notifications regarding the status of his vehicle. The solution we propose automates and guides the user in finding appropriate parking spaces using GPS based navigation. It automates the Vehicle entry and exit systems using machine learning techniques like Number plate recognition. It automates the billing, vehicle check-in and checks out processes, tracking vehicles, etc. It provides a connected solution consisting of devices like cameras that detect the vehicle entry and exit and send the data to the cloud. The solution we propose is decoupled, modularized, scalable, which improves the maintainability and future enhancement of the solution based on changing requirements. The automated parking management system is made up of 2 stations. One is at entry, and the other is at the exit at the parking places. These stations are linked to the main processing center hosted in the cloud, which provides the various parking management functionalities like synchronous updates of the available slots, nearby parking spaces, automated check-in and check-out logs, automated billing, etc. Advantages of the proposed method include increased transparency in the system, automated processes reduce the latency of the parking process, if well implemented, users can find nearby parking spaces increasing the visibility and the profits of the parking space owner, digitalization of the generation of parking bills and automated billing provides a seamless, dynamic and effective parking process, providing systematic and efficient parking space management, additional features like space availability, tracking of vehicles inside parking area, analytics etc. provide extra benefits for the parking management infrastructure.

III. LITERATURE SURVEY

We undertook a comprehensive literature survey which covered the following papers and articles:
K.M. Sajjad[1] discusses about using real-time embedded

systems for number plate detection. The author has also emphasized on using open source software and tools like open computer vision library for development of effective ALPR System. The author also briefly states different applications of such a ALPR software ranging from urban traffic control to parking admission.

According to Sergey Zherzdev and Alexey Gruzdev[2], Automatic License Plate Recognition is a challenging and important task which has various applications in the field of mobility solutions like traffic management, digital security surveillance, vehicle recognition, parking management of big cities. This paper tackles the License Plate Recognition problem and introduces the LPRNet , a Deep Convolutional Neural Network algorithm, designed to work without pre-segmentation and the consequent recognition of characters. The authors further discuss techniques like pruning and quantization by which the overall efficiency and processing can be improved.

Yann Lecun, Patrick Haffner, Leon Bottou and Yoshua Bengio[3] gives a detailed description and implementation of Object recognition using convolutional neural networks. It gives a detailed analysis of topics like local receptive fields, shared weights, spatial sub-sampling etc. The paper also briefly discusses LeNet-5 architecture used in character classification. It also gives a comparison between Convolutional Neural Networks and other neural networks like Support Vector machines etc. Their paper also gives a detailed analysis of gradient descent-based learning in complex systems. The paper also talks about Graph Transformer networks and their application in object detection.

Joseph Redmon and Ali Farhadi[4] focus on detailed analysis on the working of the yolov3 architecture and its various components used in object detection in their paper. It also compares the performance of yolov3 architecture with its predecessors, R-CNN, RetinaNet, Resnet and other architectures. It gives a brief about the bounding box predictions and its implementation. It also explains about feature extraction and Prediction across scales. Also, the advantages and disadvantages of Yolo architecture over other architectures is mentioned.

IV. WORKING METHOD OF APPLICATION

System Requirements used for implementation are as specified. Hardware Requirements: i3 Processor and above, 4GB minimum but 8GB RAM recommended, Hard Disk of 4GB minimum but 8GB recommended, Android Mobile, App Version is Marshmallow and above, Google Maps to get the real-time location. Software Requirements: Operating System of Windows 7 or Higher, Coding Languages such as Java, Python, TypeScript, Firebase for Back End, Java Software of JDK 1.8 or above, tools like Android, Python3, Google Colab, Google Cloud Platform (GCP), Raspberry-Pi, VS Code, Android Jetpack, Git and GitHub. System design defines the architecture, components, interfaces and data-flow for a system which satisfies the system's requirements. The system design for this project is aimed toward following the microservices architecture wherein each module of the system is loosely coupled but cohesive enough to independently develop, test, deploy and maintain while being scalability on changing system requirements and other demands.

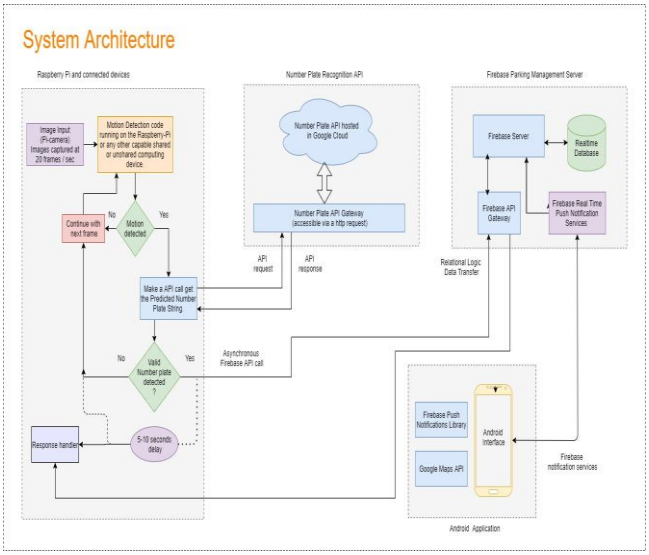


Fig-1: System architecture.

The Implementation of the System can be broken down into 4 major modules:

- Image capturing mechanism by motion detection.
- The Number Plate Recognition API.
- The Firebase Server.
- The Android Application.

Image capturing mechanism by motion detection.

We used the Raspberry Pi accompanied by its camera for detecting motion and capturing images, which are then sent to the number plate recognition API for Number plate recognition.

The Motion detection code works as follows:

Each frame is pre-processed as a grayscale image having the values of pixel intensities between 0 and 255. The frame is then compared with the previous frame and an intermediate image consisting of the absolute difference in corresponding pixel intensities of the two frames is constructed. From this image, the number of pixels for which the difference in pixel intensities is above a predefined threshold is considered as Pt. The original frame is sent to the Number plate Recognition API hosted on an external server via an HTTP request. Meanwhile, the image capturing mechanism is temporarily paused for 10 seconds in order to avoid duplicate requests made to the server for the same event. For every request passed by the camera, the server processes the image, and it returns the predicted number plate string, which is then sent to the firebase server handling the parking management and other associated trigger mechanisms.

The Implementation of Number Plate Recognition API comprises of 5 submodules:

- Number Plate Detection and Localization from the given input image.
- Character Segmentation from the detected number plate image.
- Character Recognition

- Finding the closest matching number plate from a set of registered number plates.
- The Flask Microservice

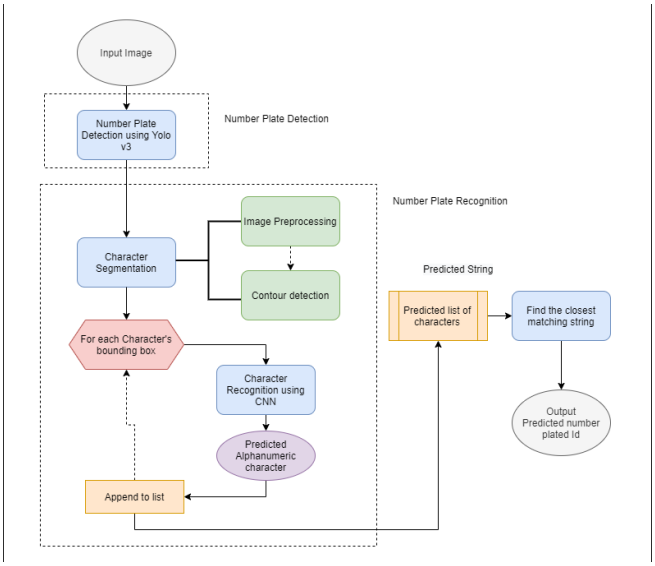


Fig-2: Block Diagram for Number Plate API

Number Plate Detection and Localization from the given input image.
We used the darknet open-source framework and the Yolo v3 architecture to train the Number plate detection model. The dataset we used to train our Plate detection model consisted of about 1800 images. The dataset was split up as 1600:200 images as train and test images. We took the dataset from various sources like Dataturks, Kaggle. We used tools like label-img for annotating the datasets. The annotation are of the form :
<object-class> <x> <y> <width> <height>
Example: 0 0.716797 0.395833 0.216406 0.147222
The model was trained for about 7 hours, for about 2000 epochs. The model stopped training with
-an average loss of 0.07
-0.5R of 0.96 (this corresponds to the percentage of samples with a recall having the confidence like 50% or more)
-0.75R of 0.87 (this corresponds to the percentage of samples with a recall having the confidence like 75% or more).

Character Segmentation from the detected number plate image.
We experimented with the following different approaches to accomplish this:

- Using the yoloV3 to train a single class character detection and localization model.
- Using image processing techniques like pixel projections.

Using image processing techniques like image filling and contour detection

Fig-3: Plate detection Input

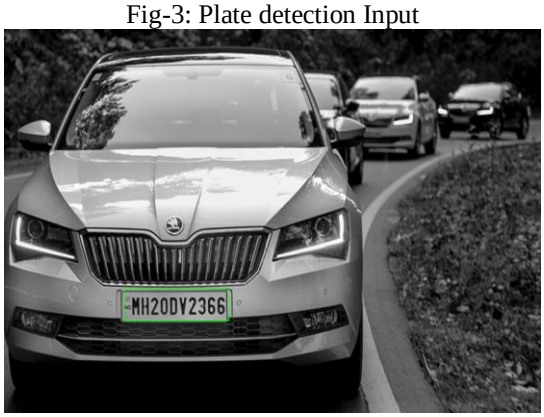


Fig-4: Plate detection Output

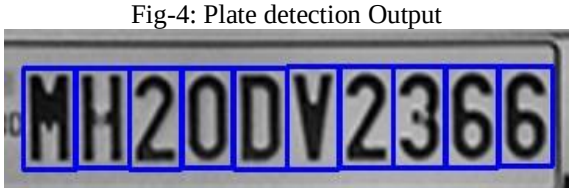


Fig-5: Character Segmentation output.

Character Recognition.
We used a Convolutional Neural Network to classify the characters from the number into one of the 35 classes. The 35 classes correspond to the alphanumeric characters (A-Z 0-9) with the character 'o' and the number 0 merged into one class. The model was trained on a dataset comprising about 60000 images (about 1800 images per class). The model was evaluated on a test set consisting of about 2000 images. The accuracy for this model (i.e. the character classifier) was found to be about 95.66%.

Finding the closest matching number plate from a set of registered number plates.
In order to improve the overall accuracy of the system, we used a string matching algorithm to find the closest number plate id.
After the prediction module returns the predicted string, the string is compared with a list of existing (registered) number plate strings. We used the Levenshtein distance as a measure to find the similarity between strings. The Levenshtein distance between two words is the minimum number of single-character edits (i.e. insertions, deletions or substitutions) required to change one word into the other.

The Flask Microservice.
We combined all the existing submodules into a single class.

We then coupled this with the Flask framework to create an ANPR microservice. This microservice (API) takes the input as an HTTP request (consisting of the image) and returns the predicted string as the HTTP response. We hosted this microservice in a Google Cloud Platform on a public domain so that it could be accessed by all devices on the internet having required authorization.

The Firebase Server.
Firebase is used as the Back-end Server to store the database. It stores the data in JSON-like documents. We use Node.js with Firebase CLI to develop our cloud functions to communicate with the mobile app.
In the firebase schema, there are four tables, namely :

- The History table.
- The User table.
- The Parking table.
- The Parking Slot table.

The History table:
The history table is used to store the parking history of all the users. It contains the user Id of the user who had parked his/her vehicle in the parking slot previously. This helps the app in identifying the users that have already signed in to this app before. The users' previous data and records are maintained neatly in the database ranging from the license no, the status of the vehicle, i.e. it has parked or left the parking slot, the time of entering and exiting the parking slot, how long the vehicle was parked and the allocated bill for the parked time.
The User table:
User Table is used to store the new user data and records. It contains the user Id of the new user who has signed in to the app. The license number of the user along with the user's name and a valid password to sign in to the account is noted in this table. This table also contains the current parking spot id of the parking area where the user is parking his/her vehicle. And lastly, the table collects the unique device token generated by each mobile phone to send notifications to the user's app.

The Parking table:
Parking Table is used to store the details about the parking area. It includes the unique parking-id of the area where the vehicle is parked. Also, the address, the area where the parking zone is located is included. The table also contains the title of the parking area, an image of the parking zone and also the price allocated for parking in the parking zone.

The Parking Slot table:
Parking Slot Table indicates whether a parking spot is available for parking or not. It contains the unique value of the parking spot chosen by the user. If the chosen spot is available for parking, then the user will be notified that he/she can park the vehicle in the spot. If the chosen spot is not available then the user will get notified that the parking spot has been already allocated to someone.

In the Vehicle Check-In process, when a vehicle checks in to the parking, the vehicle check-in function is called. It receives a parking Id and license number of the vehicle. If the parking id is not valid, then a status 204 with the message "Parking not found" will be returned. And if the Parking Id is valid, then using the vehicle license number, the user account is identified from the database. If the license number does not match with the one entered by the user, then a status 404 with User not found is returned. After this, we will check if the user is currently in the parking. If it is true, then a status 401 with the message "User is in the parking" will be returned. If it is false, then a new history document is created with status PARKING and sends a notification to the user. The current parking id in the user's document is updated with the id of the created history, and a status 200 will be returned for a success.
The specific user can be identified by the device token in the user's document, which was generated by the app and a notification is sent to the user using firebase cloud messaging.

In the Vehicle Check-Out process, when a vehicle leaves the parking, the vehicle check-out function is called. It receives a parking Id and license number of the vehicle. If the parking Id is not valid, status 204 with a message "Parking not found" will be returned. If it is valid, then using the vehicle license number, the user account is identified from the database. If the license number does not match with the one entered by the user then a status 404 with the message User not found will be returned. Now, after the user account is identified, we will check if the user is currently in the parking if it is false, then a status 404 with the message "User is not in the parking" will be returned. If it is true, then a check-out notification is sent to the user, and the current parking id in the user's document is updated to null indicating there is no current park for the user. Then a status 200 will be returned for the success of the process.

The duration of the parking is calculated from the time the history table document was created until this process is called, and the price of the park is calculated using the time duration and parking charge.

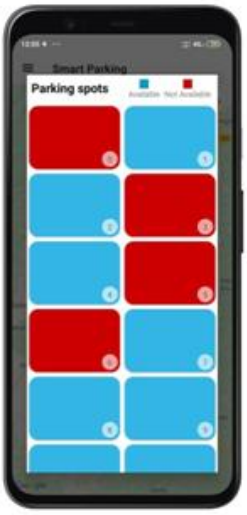
The Android Application.
The App consists of four modules:

Authentication Module:
This module consists Sign-In and Sign-Out features. These sections take care of creating a user's account as well as authenticating the existing user's account.

Map Module:
Map module takes the user's current device location and shows the parking areas which are closer to the user's current location. This module also shows the direction towards the parking area on the user's mobile app using Google Maps.

Fig-6: Map Interface

Fig-7: Parking Slot Selection.



Parking Module:
This module shows details about any parking, their current availability status of the parking slot. It gives a feature to the user to be able to select a parking slot when they enter a parking lot.

History Module:
This module displays all the parking history of the user. It consists of parking details such as Price, Duration, Selected Parking Slot etc.

IV. TESTING AND RESULTS

Testing for the Object detection module:
The object detection module was tested on 98 unseen images for which the model returned a 0.75R of 0.79 (this corresponds to the percentage of samples with a recall having the confidence as 50% or more). In other words, this indicates that for about 79 % of the test images (i.e. 78/98 images) the IOU of the predicted bounding box and the actual bounding box was greater than 0.5.
Testing the Character classifier:
The Character classifier model tested on about 4000 images from the same training dataset gave an avg loss of 0.058 and accuracy of about 93.67 %.

Testing the Number plate recognition API:
The Number plate Recognition API was manually tested on 40 images containing valid image samples in different orientations, out of those the predicted string and actual string matched for 33 samples of distance 0 and 2 samples with Levenshtein distance 1 and 4 samples with Levenshtein distance greater than 2.Therefore giving an accuracy of (34/40), i.e. 0.85 or 85%.
The average response time for the API was calculated to be about 4.8 seconds. This time is inclusive of the time taken for image upload, processing image and returning the number plate string to the client.
Note: The Levenshtein distance is the minimum number of single-character edits required to change one string into the other.
The following gives a summarization of the various test cases that were passed.

- Test case: To check user creation with empty mandatory fields
- Test case: To check unmatched password on user creation
- Test case: To check conflict on vehicle license number
- Test case: To check user sign-in with empty mandatory fields
- Test case: To identify an unknown user
- Test case: To verify a user
- Test case: To check runtime permission request
- Test case: To show nearby parking in the map interface
- Test case: To check parking details and availability
- Test case: To check unavailability of a parking
- Test case: To display parking direction
- Test case: To open the side navigation drawer
- Test case: To check navigation clicks
- Test case: To see more content of a history
- Test case: To show check-in notification
- Test case: To display parking slot selection screen
- Test case: To select a parking spot
- Test case: To show check-out notification
- Test case: To logout current user

Result analysis:
The Number plate recognition API performs as expected but lacks the accuracy due to the quality of the available datasets used. The Number plate recognition API performs below par in unnatural or custom lighting conditions, unconventional number plate fonts, misplaced orientations. The accuracy of the Number plate recognition API can be vastly improved by improving the quality of the dataset and quantity of samples in the dataset. The microservice can be further scaled up in order to increase the response time and parallelize the processing of the image.

V. CONCLUSION AND FUTURE WORK

The paper presents the vehicle logging system with automatic number plate recognition. It studies the license plate recognition of the vehicles based on Machine learning. The key elements of the system are successfully designed and implemented. The proposed system recognizes the license plate and generates the parking bills along with its entry-time and exit-time of the vehicles.
The Smart Parking system based on Number plate recognition of the vehicles is designed and implemented. The proposed system comprising various technologies is designed, implemented and tested. The various findings (advantages and shortcomings) of the technologies and the system were found and documented in the paper. The process of Parking of the vehicle becomes transparent, automated and efficient.

The scope of future work includes:
Increasing the efficiency and accuracy of the Number Plate Recognition API. This could be accomplished using better quality datasets, using ensemble techniques to improve efficiency, etc. Improving the Parking Management System with features like analytics, an improved portal for the System administrator, improved concurrency in the slot distribution mechanism, etc.

REFERENCES

[1] "Automatic License Plate Recognition using Python and OpenCV" Author: K.M. Sajjad.
[2] "LPRNet: License Plate Recognition via Deep Neural Networks" Author: Sergey Zherzdev, Alexey Gruzdev.
[3] "Object Recognition with gradient-based learning" Author: Yann Lecun, Patrick Haffner, Leon Bottou and Yoshua Bengio.
[4] "YoloV3: an incremental approach" Author: Joseph Redmon and Ali Farhadi.
[5] "Information Push Technology and Its Application in Network Control System" Author: Junman Sun, Huajing Fang, Ganyi Wang, Zhendong He.
[6] "License Plate Recognition Using Convolutional Neural Network" Author: Shrutika Saunshi, Vishal Sahani, Juhi Patil, Abhishek Yadav, Dr Sheetal Rathi.
[7] "Proposal for Automatic License and Number Plate Recognition System for Vehicle Identification" Author: Hamed Saghaei.
[8] "License Plate Detection and Recognition in Unconstrained Scenarios" Author: Sergio Montazzolli Silva and Cl´audio Rosito Jung.
[9] "Automatic Number Plate Recognition System" Author: Amr Badr, Mohamed M. Abdelwahab, Ahmed M. Thabet, and Ahmed M. Abdelsadek.
[10] "Automatic Number Plate Recognition System: Machine Learning Approach" Author: Mrs J. V. Bagade, MSukanya Kamble, Kushal Pardeshi, Bhushan Punjabi, Rajpratap Singh.

