An intelligent Raspberry-Pi based parking slot identification system

Raghav Agarwal[1], Gaurav Sharma[2], Nirdesh Singh[3], Hrishikesh S Nair[4], Yash Daga[5],

D. Venkata Lakshmi[6]

[1, 2, 3, 4, 5, 6]School of Computer Science & Engineering (SCOPE),

VIT-AP University, Amravati, Andhra Pradesh, India

raghav.20bce7383@vitap.ac.in[1], gaurav.20bce7443@vitap.ac.in[2], nirdesh.20bce7062@vitap.ac.in[3], hrishikesh.20bes7080@vitap.ac.in[4], yash.20bce7323@vitap.ac.in[5], venkatalakshmi.d@vitap.ac.in[6]

Abstract: A growing population necessitates more transportation, which puts more pressure on car parking spots. Parking is a problem for those who attend public places in cities, such as theatres, malls, parks, and temples. Even though several techniques have been suggested in publications, manual parking systems are still used in most places. For large locations where it is challenging to find open spaces, traditional parking arrangements need to be more archaic and convoluted. This might lead to heavy traffic, minor mishaps, and widespread accidents. In the modern era of sophisticated parking management systems, the automatic parking spot-detecting system has been introduced in a new format. Experts in computer vision are being drawn to this emerging field to contribute. The system could be able to tell if the automobile is fully or partially parked. Neither during the process nor subsequently, human oversight is not required. The system can classify cars as well as people or walkers, enabling accurate decisions to be made in circumstances when people or other obstructions have taken over the slots. As parking management enters a new era, computer vision is becoming increasingly important. The parking system will not only make it easier for drivers to identify parking spaces, but it will also enhance parking administration and monitoring. Vehicles will be able to observe available parking spots due to the technology that will monitor parking spaces. India and other emerging nations, as well as industrialized ones, have recently shown an interest in smart cities. This article's smart auto parking system was conceived and implemented utilizing a Raspberry Pi and cameras placed in various parking spaces. Using a website and an Android app, this project creates and deploys a real-time system that enables vehicles to efficiently find and reclaim open parking spaces.

Keywords: Parking Occupancy Detection, Computer Vision, Smart parking, Android App, Raspberry Pi

1. Introduction

With an increase in vehicles on the road, parking has become a significant problem in urban areas. Drivers will eventually look for an open parking space, park improperly, or park in a space allocated for people with disabilities due to the lack of real-time monitoring of parking spaces, especially close to the busiest locations. The parking issue causes traffic congestion, especially during rush hour, and wastes the driver's time and fuel [1]. Air pollution and disease are caused by an increase in CO emissions due to traffic congestion [2]. Governments searched for an efficient parking system to manage and control parking lots while easing traffic congestion as a result of this problem. Therefore, the management and maintenance of parking slots in both public and private parking spaces would be made simpler by the presence of a parking system that directs vehicles to the best available parking spot in a monitored and controlled manner. Vehicles will be able to hunt for parking spots using a smartphone app.

In today's packed cities, finding a parking space might be challenging. Unfortunately, there aren't enough parking spots to accommodate all the automobiles. As a result, effective parking management systems are becoming more and more in demand. There is no mechanism in place to detect if the spaces are unoccupied, and the majority of the parking slot is not automated. An intelligent parking management system may speed up traffic flow while automatically or manually deporting automobiles. In today's world, where technology and people are growing quickly, there is a higher need for parking cars, especially in public areas. The available slots are unknown to the drivers.

The article will show how a parking management system built on the Internet of Things can maximize the use of available parking spaces. There are websites in several nations where individuals may learn more about parking slots. This system can provide users with parking-related information. Through the website and Android app, you can check the availability of parking spaces in various parking zones from a distance.

In Section II of this study, various projects and studies concerning smart parking systems in smart cities will be evaluated, and in Section III, the methodology for smart parking systems will be highlighted. The findings and discussion will also be displayed in Section IV. The conclusion and future remarks will be covered in Section V.

2. Related Work

This section will examine the evaluation and assessment of various Smart Parking systems and efforts implemented in Smart Cities. Due to increased traffic, the time that drivers waste looking for available parking spaces, air pollution, and other factors, smart parking systems are becoming more and more popular in smart cities. One of these options is the Smart Parking system, which makes use of a variety of technologies and network infrastructures [3].

Current large-city study indicates that there are several methods for managing parking, including the amount of traffic on the roads. When trying to park their cars, drivers become frustrated due to the difficulty in finding a parking space. It is common for drivers to waste time and effort looking for parking spaces and wind up parking on the street, which congests the parking lot. In the worst-case situation, people might not be able to get a parking spot, especially during holidays or other busy times.

[4] described a smart parking system where users could use a website to check for open parking spaces and reserve them if they weren't previously reserved using LED lights and sensors. The recommended architecture can cut down on the amount of time spent searching for parking and can ease traffic congestion, but it necessitates that the user be connected to the same WIFI network in order to utilise the system.

[5] introduced SPARK, an automated, cost-efficient, real-time wireless sensor network-based smart parking system. A single parking place is monitored and reserved for the user by this method. The technology uses spot data and light sensors to detect cars. The information is then sent by radiofrequency to a subsystem.

[6] developed a smart, automated parking system using sensors based on the Raspberry Pi that is less expensive than expensive video cameras. It suggested creating a mobile application as a potential future project to showcase the availability of parking spaces and to notify users when parks are open nearby.

To identify parking space openings, outdoor parking systems often take the shape of sensor-based or computer vision-based systems, including camera-based surveillance systems.

[7] described a system that uses cameras and Classifier Neural Networks, requiring a specialist server with a lot of processing capacity and the ability to analyze and foresee real-time data at extremely fast rates. This system cannot scale since the amount of data generated increases exponentially with infrastructure.

Smart Parking System with IoT Support [8] This paper includes a KNN approach for slot security and availability. A sensor module, a raspberry pi module, and an Arduino module are used for both reserved and non-reserved vehicles.

Intelligent Parking Management System Based on Image Processing [9] It has been claimed that image processing methods can be used to locate a parking space. The RGB colours, which are regarded as fundamental hues, were used to identify empty spaces. The green colour denotes when a car is not parked in its designated spot, the blue colour denotes an entirely empty space, and the red colour denotes an empty lot.

Our system uses a small amount of computer resources. Because the modules are capable of performing all fundamental and necessary computations, a simple server is sufficient to manage the database, and our system is scalable without an increase in processing demands. Our recommended smart outdoor parking solution does this without the need for staff, providing orderly and secure outdoor parking.

3. Methodology

3.1. System requirement:

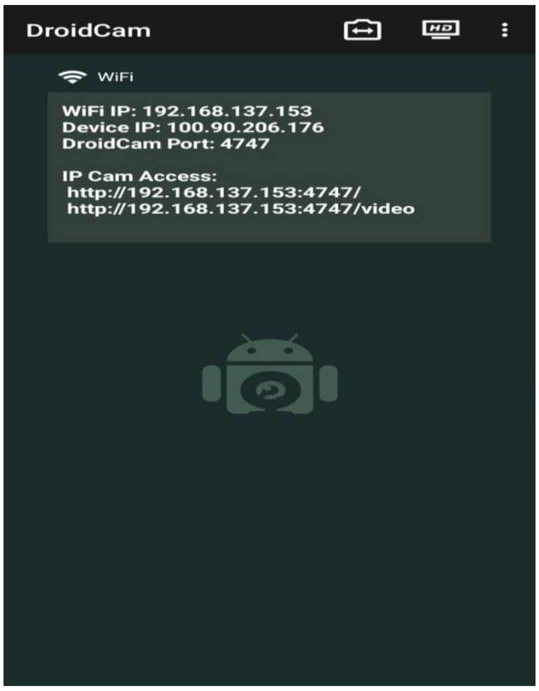
(a) DroidCam

With the help of the DroidCam programme, users may use their Android tablet or smartphone as a wireless camera for their PC. Both a free and a premium version of the app are available for download from the Google Play Store.

Users may utilise their Android device as a camera for video chats, online meetings, live streaming, and other purposes by utilising DroidCam to connect their Android handset to their computer via Wi-Fi or USB. Numerous customization options, such as changing the camera settings and video quality, are provided by the programme. A variety of video software products are also supported.

DroidCam is an all-around beneficial and practical option for anyone who needs a camera, possibly saving consumers the trouble of getting separate webcam equipment.

We use the WIFI-connected DroidCam application to generate a distinct IP address for each parking slot, which we then feed into the ML model which runs on the Raspberry Pi. The in-use demo of DroidCam is given in Fig.1. (a).

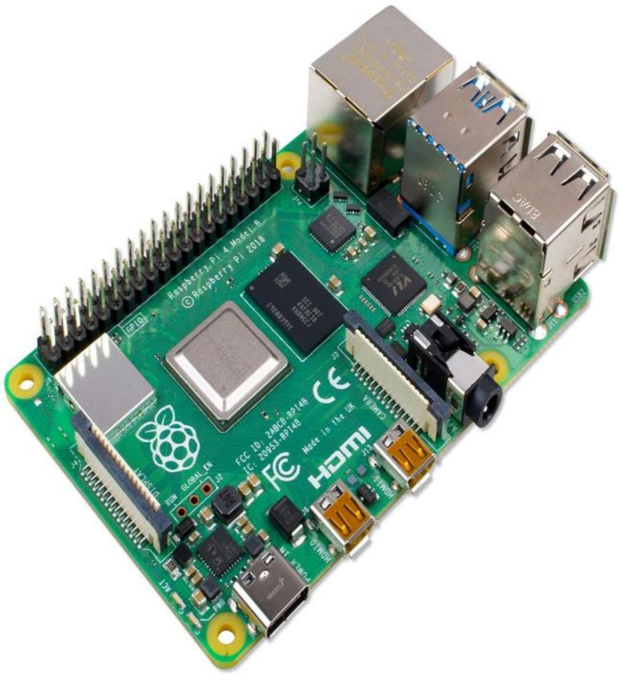


DroidCam

WiFi

WiFi IP: 192.168.137.153
Device IP: 100.90.206.176
DroidCam Port: 4747

IP Cam Access:
http://192.168.137.153:4747/
http://192.168.137.153:4747/video



(a) (b)

Fig.1 System Requirement (a) DroidCam (b) Raspberry Pi

(b) Raspberry Pi

We are using Raspberry Pi 3B+ , The Raspberry Pi 3B+ is a single-board computer designed for experts, academics, and hobbyists alike. This powerful computing platform contains a quad-core ARM Cortex-A53 Broadcom BCM2837B0 CPU and 1GB of RAM.

With built-in Wi-Fi, Bluetooth, Ethernet, and HDMI, the device may easily be connected to a wide range of peripherals and gadgets. For further expansion options, it also contains a 40-pin GPIO header, 4 USB ports, and a microSD card slot.

The Raspberry Pi 3B+ may be used for a wide range of tasks, from robotics and Internet of Things devices to home automation and media centres, and it can run a number of operating systems, including Linux-based versions like Raspbian. Due to its low price and adaptability, it is an excellent choice for anyone who wants to learn about or experiment with computing and electronics.

Our Raspberry Pi 3B+ is running an ML model that accepts video input using the distinct IP addresses produced by the DroidCam application.

The number of filled and unfilled parking spaces in a specific parking spot is then generated and pushed into the Cloud Firestore (Firebase). Fig.1. (b) represents the raspberry pi.

3.2. Detection Mechanism

By utilising computer vision techniques to identify occupied parking spaces, our ML model develops a smart parking system. Two colors—green and blue—are recognised by the system using OpenCV in images taken by cameras placed at parking lot entrances. The percentage of green and blue tones in the image is computed after colour detection using the HSV colour system. The code evaluates the parking lot's occupancy condition based on the proportion of green and blue colours seen and changes the status in a Firestore database.

The model begins by importing the Firebase Admin SDK, time, OpenCV, and other necessary libraries. The method detectobj() takes an image as input and produces the number of completely occupied parking spots by employing a colour detection approach. The code uses the cv2.inRange() method to construct a mask for the green and blue hues in the image, and then counts the number of green and blue pixels in the mask to calculate the percentage of each colour in the image. After determining whether it has picked up both colours or only one, the function modifies the occupancy status accordingly.

The Firebase Admin SDK is then initialised by providing the location of the service account key file. Then, three cameras are positioned at the entrances to three parking slots, and a loop is set up to record images from those cameras. To assess if the room is inhabited, the code gathers an image from each camera, converts it to RGB colour space, and then passes it to the detectobj() function. The occupancy status in the Firestore database is then updated.

Until the user closes the application or the cameras stop taking pictures, the code keeps running. The software updates the occupancy status of each parking lot every five seconds.

3.3. Frontend

We utilise Flutter for the front-end of our mobile application. Google developed the free and open-source Flutter mobile app development framework. It enables programmers to create high-performance, natively designed programmes from a single codebase for desktop, web, and mobile platforms. Developers may design stunning and responsive user interfaces with the help of Flutter, a programming framework that makes use of the Dart programming language and offers a wide variety of pre-built and customised widgets.

One of Flutter's key advantages is its rapid reload functionality, which lets programmers update the code and see the results right away without having to restart the application. This feature allows developers to iterate fast and test out different designs and functionalities while also significantly speeding up the development process.

Flutter offers a wide range of pre-built widgets, thorough documentation, and a growing online developer and contributor community, among other tools and resources, to developers. As a result, it is now simpler for programmers to get started with Flutter and employ all of its capabilities to develop mobile apps that are aesthetically beautiful and intriguing. In general, Flutter is a robust and adaptable framework for creating mobile apps that is rapidly gaining favour with both developers and companies.

One screen in our smartphone app essentially displays the three parking spaces: Parking A, Parking B, and Parking C. The app displays the number of open and occupied parking spaces in each parking spot. The information on available and occupied parking spaces is retrieved from the back-end (Firebase).

3.4. Backend

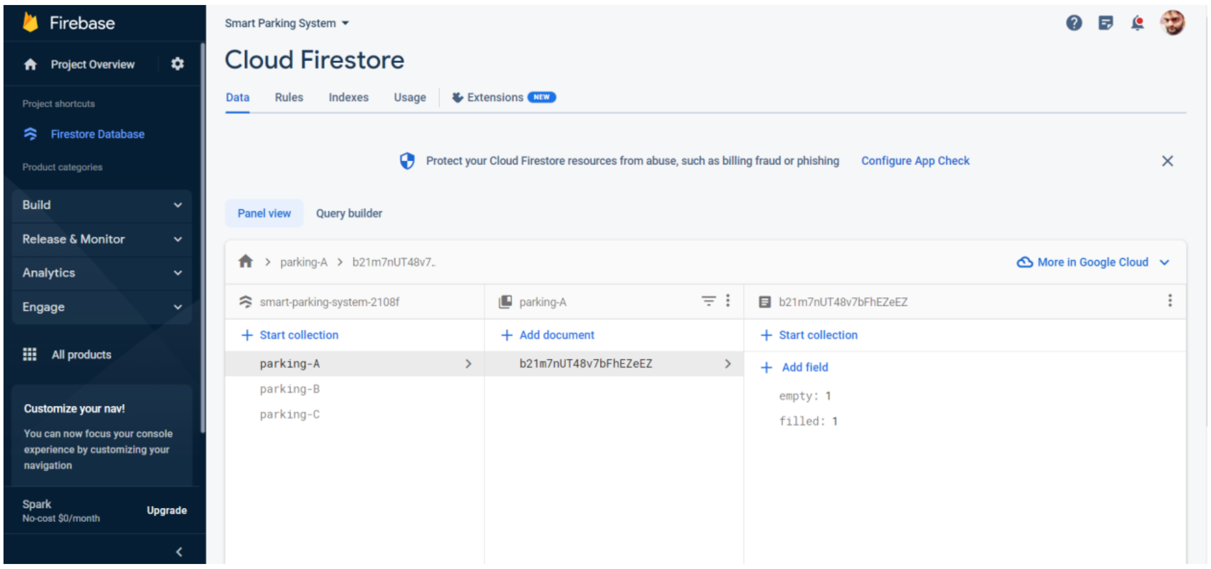
A NoSQL cloud database, Cloud Firestore is a component of Google's Firebase platform. It offers a scalable and adaptable solution for real-time data storage and retrieval for developers of mobile and web apps. For better speed, Cloud Firestore stores data in a document-oriented manner as opposed to traditional SQL databases.

Real-time synchronisation is one of Cloud Firestore's distinctive characteristics. All connected clients immediately receive updates whenever data is added, updated, or removed from the database. Therefore, it is simple to create cooperative applications such as chat apps, real-time dashboards, and multiplayer games.

As an added bonus, Cloud Firestore offers a powerful querying mechanism that spares developers from having to write time-consuming server-side code in order to filter, sort, and paginate data on the client-side. This enables the speedy and efficient retrieval of data from huge databases. Additionally supported by Cloud Firestore are transactions and batch writes, which ensure data consistency and reduce server round trips.

A further benefit is offered by the way Cloud Firestore is integrated with the Firebase platform. As a result, developers can easily include authentication, cloud communications, and cloud features into their programmes without having to build up a challenging infrastructure.

The Cloud Firestore is used to store the number of occupied and vacant parking spaces that we get from the ML model. The real-time data is then fetched by the mobile app and shown to users. The same is displayed in Fig.2.



Customize your nav!
You can now focus your console experience by customizing your navigation

Protect your Cloud Firestore resources from abuse, such as billing fraud or phishing

parking-A > b21m7nUT48v7..

smart-parking-system-2108f | parking-A | b21m7nUT48v7bFhEZeEZ
parking-A | b21m7nUT48v7bFhEZeEZ | empty: 1
parking-B | | filled: 1
parking-C | |

Fig.2. Backend support of the proposed system

3.5. System Design

Here’s the, step by step algorithm for the identification of available or occupied parking slots:

(a) Import required libraries: cv2, numpy, time, os, firebase_admin, credentials, and firestore.
(b) Define the detectobj() function, which takes an image as input and returns the number of parking spaces that are occupied.
    (i) Change the image's colour space from BGR to HSV.
    (ii) Set lower and upper limits for the HSV space's green and blue hues.
    (iii) Use the inRange() method of the cv2 module to create two masks for the colours green and blue.
    (iv) Calculate the percentage of green and blue color in the image.
    (v) See if you can distinguish the colours green and blue, one of them, or none of them.
    (vi) Using the bitwise_or() method, combine the green and blue masks.
    (vii) Using the bitwise_and() method, apply the mask to the original picture.
    (viii) Return the number of full parking slots based on the number of colors detected.
(c) Specify 5 frames per second as the fpsLimit.
(d) Set the startTime to the current time.

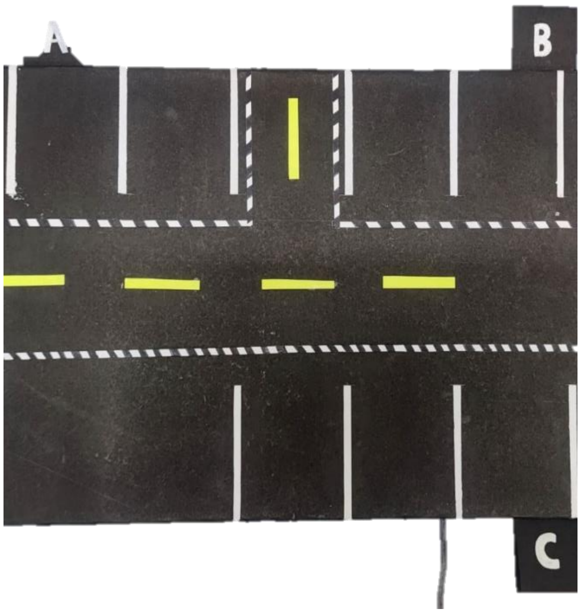
(e) Utilising cv2, launch three video streams.Their corresponding IP addresses are used using the VideoCapture() method.
(f) To read frames from the three video sources, start an endless loop.
    (i) Read the frame from parking A in the first video feed.
    (ii) See if the difference between the current time and the start time exceeds the fpsLimit.
    (iii) If so, change the frame's colour space from BGR to RGB.
    (iv) To determine the number of available parking spaces, use the converted picture and the detectobj() method.
    (v) Subtract the number of full parking spaces from the total number of spaces to determine the number of unoccupied spaces.
    (vi) Utilise the initialize_app() and connect to the Firebase Firestore database operations.
    (vii) Add the number of empty and filled parking spaces for parking A to the Firestore database.
    (viii) For parking B and parking C, repeat steps 1 through 7.
    (ix) Close all windows after releasing the video capture.
(g) End of the algorithm

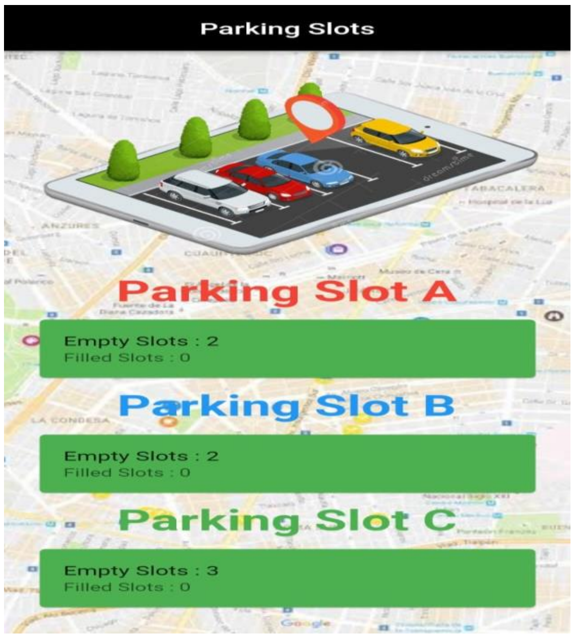
4. Results and Discussion

We are considering three parking spaces -Parking A, Parking B and Parking C. Three scenarios might occur:

Case 1: Any parking place that is completely empty (let's assume parking A as an example).

The ML model in this instance fails to identify a sizable fraction of green or blue colour. Thus, no slots are detected as being occupied. The app shows the same thing. The parking spot widget in the app stays green, suggesting that there are still spaces available. Fig.3. represents the input(parking condition) and output(Empty and filled slots detection) for case-1.




Parking Slots

Parking Slot A
Empty Slots : 2
Filled Slots : 0

Parking Slot B
Empty Slots : 2
Filled Slots : 0

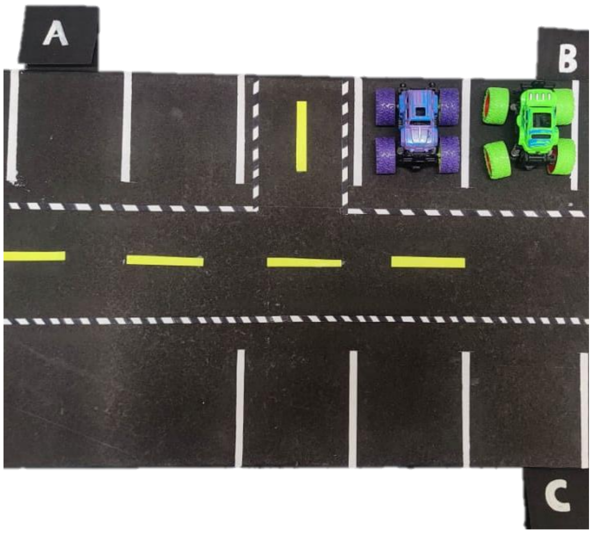
Parking Slot C
Empty Slots : 3
Filled Slots : 0

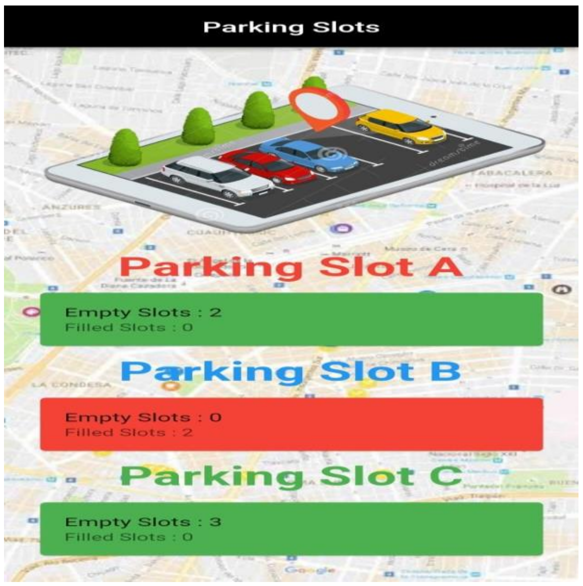
(a) (b)

Fig.3. Empty parking (a) Parking condition (b) Empty and filled slots detection

Case 2: When all parking spaces are fully occupied (let's assume parking B as an example).

In this instance, a sizable percentage of green and blue colors are detected by the ML model. Therefore, the number of vacant slots is reported as zero. The app shows the same thing. When there are no parking spaces available, the parking spot widget in the app turns red. Fig.4. represents the system input and output for case-2.




Parking Slots

Parking Slot A

Empty Slots : 2
Filled Slots : 0

Parking Slot B

Empty Slots : 0
Filled Slots : 2

Parking Slot C

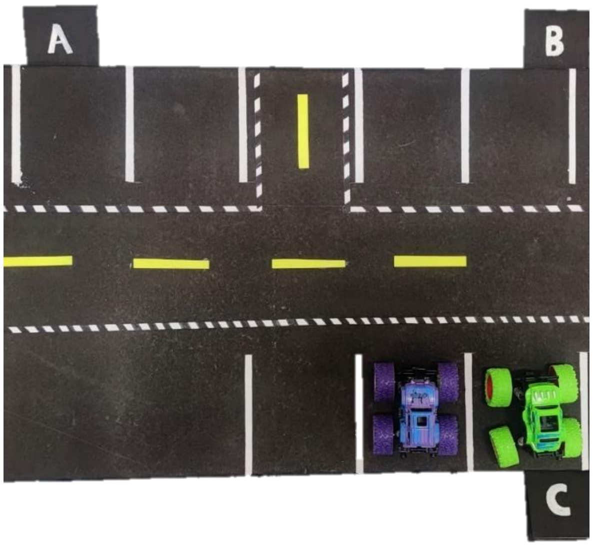
Empty Slots : 3
Filled Slots : 0

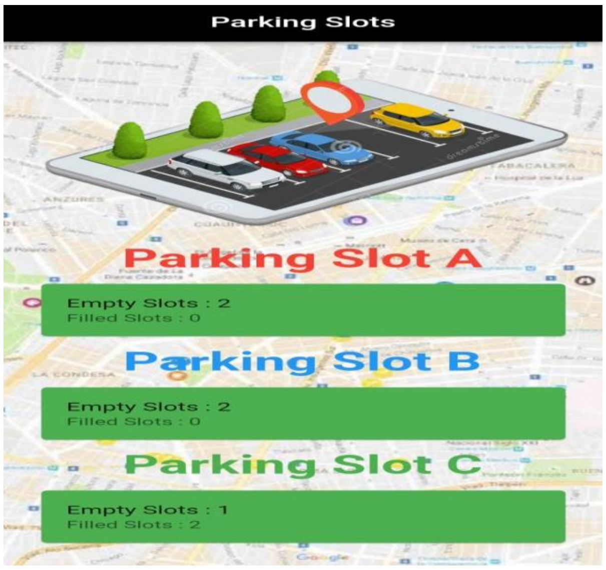
(a) (b)

Fig.4. Fully filled parking (a) Parking condition (b) Empty and filled slots detection

Case 3: When any parking space is partially occupied (let's assume parking C as an example).

The ML model in this instance finds a large percentage of green or blue color. Thus, the numerical value of vacant and filled slots is determined. The app shows the same thing. The parking spot widget in the app stays green, suggesting that there are still spaces available. Fig.5 represents the condition for case3.




Parking Slots

Parking Slot A

Empty Slots : 2
Filled Slots : 0

Parking Slot B

Empty Slots : 2
Filled Slots : 0

Parking Slot C

Empty Slots : 1
Filled Slots : 2

(a) (b)

Fig.5. Partially filled parking (a) Parking condition (b) Empty and filled slots detection

Hence, the proposed system is detecting the free and occupied slot effectively as shown in Table 1.

Table1. System observation for all the cases

Case	Parking Slot	Percentage of Green	Percentage of Blue	Occupied Slots	Unoccupied slots
1	A	0.0	0.01041	0	2
2	B	11.3541	3.1647	2	0
3	C	9.7434	8.3541	2	1

5. Conclusion and future remarks

Everybody's life now depends on their vehicles. However, everyone drives because traffic is so common. These days, it takes time to find a parking spot while the user is outside. A crucial source is also to provide parking for the car. This proposed effort's main aims are to help reduce petrol costs and find the best available space.

IoT-based smart cities are a ground-breaking innovation that has been built and is continuously developing and enhancing the services provided. One of these services, Smart Parking, exemplifies the new parking procedure, which differs significantly from the old network in terms of time savings, reduced traffic congestion, and reduced pollution. The new system lowers labor costs by reducing system complexity.

Automatic Parking Slot Occupancy Detection is a useful concept for implementing smart parking since it accurately and instantly assesses vacant and occupied parking spaces. This study's main objective was to identify cars and open parking spots in a stream of video images to raise the city's parking infrastructure and reduce emissions while also boosting locals' quality of life. The structure developed helps to reduce pollution since navigation is regulated. Time is saved, and movement is made easier, thanks to technology. The Android app offers accurate location data.

The experimental investigation suggests that additional modifications to the current model can be used to enhance the parking system. Future implementations will make it simple to locate outdoor parking spaces more precisely.

References:

[1] Pham, T. N., Tsai, M. F., Nguyen, D. B., Dow, C. R., & Deng, D. J. "A cloud-based smart-parking system based on Internet-of-Things technologies", IEEE Access, 3, pp. 1581- 1591, (2015).

[2] Zhang, Kai & Batterman, Stuart. "Air pollution and health risks due to vehicle traffic", Science of The Total Environment, 450-451. 307-316, (2013).

[3] Z. Faheem, S. A. Mahmud, G. M. Khan, M. Rahman, and H. Zafar. "A survey of intelligent car parking system," J. Appl. Res. Technol., vol. 11, no. 5, pp. 714–726, (2013).

[4] Khanna and R. Anand. "IoT based smart parking system,", International Conference on Internet of Things and Applications (IOTA), Pune, pp. 266-270, (2016).

[5] S. Srikanth, P. Pramod, K. Dileep, S. Tapas, M. U. Patil et al., "Design and implementation of a prototype smart parking (SPARK) system using wireless sensor networks," in Advanced Information Networking and Applications Workshops, 2009. WAINA'09. International Conference on. IEEE, pp. 401–406, (2009).

[6] Mr. Basavaraju S R "Automatic Smart Parking System using the Internet of Things (IoT)", in International Journal of Scientific and Research Publications, Volume 5, Issue 12, ISSN 2250-3153, December (2015).

[7]. Jermsurawong Jermsak .; Ahsan Umair .; Haidar Abdulhamid.; Dong Haiwei.; Mavridis Nikolaos: Statistical analysis to observe the parking demand for vacancy detection using a single camera for one day. J Transpn Sys Eng & IT, 2014, 14(2), 33-4. Available at - https://www.researchgate.net/publication/259190070_One Day_Long_Statistical_Analysis_of_Parking_Demand_by_Using_Single-Camera _Vacancy_Detection.

[8] Ashutosh Kumar Singh, Mohit Prakash et al., "Smart Parking System using IoT", in International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056 Volume: 06 Issue: 04, pp. 2970- 2972, Apr 2019.

[9] Hilal Al-Kharusi, Ibrahim Al-Bahadly, "Intelligent Parking Management System Based on Image Processing", World Journal of Engineering and Technology, Scientific Research, 2, 55-67, 2014. http://dx.doi.org/10.4236/wjet.2014.22006