EAI Endorsed Transactions
on Industrial Networks and Intelligent Systems Research Article EAI.EU

An Intelligent Raspberry-Pi-Based Parking Slot Identification System

Raghav Agarwal[1], Gaurav Sharma[1], Nirdesh Singh[1], Hrishikesh S Nair[2], Yash Daga[1] and D. Venkata Lakshmi[1]*

[1]School of Computer Science & Engineering (SCOPE), VIT-AP University, Amravati, Andhra Pradesh, India
[2]School of Electronics Engineering (SENSE), VIT-AP University, Amravati, Andhra Pradesh, India

Abstract

A growing population necessitates more transportation, which pressures car parking spots. Parking is a problem for 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, an automatic parking spot-detecting system has been introduced in an innovative format. Experts in computer vision are drawn to this emerging field to contribute. The system could tell if the automobile was fully or partially parked. Neither during the process nor afterward, human oversight is required. As parking management enters the modern era, computer vision is becoming increasingly critical. The parking system will not only make it easier for drivers to identify parking spaces but also enhance parking administration and monitoring. Vehicles will be able to observe available parking spots due to technology that monitors parking spaces. India and other emerging nations, as well as industrialized ones, have recently shown 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 Detection, Smart Parking, Android App, Raspberry Pi

Received on 28 August 2023, accepted on 27 October 2023, published on 06 November 2023

Copyright © 2023 R. Agarwal *et al.*, licensed to EAI. This is an open access article distributed under the terms of the CC BY-NC-SA 4.0, which permits copying, redistributing, remixing, transformation, and building upon the material in any medium so long as the original work is properly cited.

doi: 10.4108/eetinis.v10i4.4294

*Corresponding author. Email: venkatalakshmi.d@vitap.ac.in

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. This is due to the lack of real-time monitoring of parking spaces, especially close to the busiest locations. Parking causes traffic congestion, especially during rush hour, and wastes time and fuel [1]. Air pollution and disease are caused by increased CO emissions due to traffic congestion [2]. Governments searched for an efficient parking system to manage and control parking lots while easing traffic congestion due to 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 automobiles. As a result, effective parking management systems are becoming more in demand. Most parking spots are not automated, and there is no mechanism to detect if they are occupied. 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 high need for parking cars, especially in public areas. The available slots are unknown to drivers.

The article will show how a parking management system built on the Internet of Things can maximize available parking spaces. There are websites in several

nations where individuals may learn more about parking slots. This system can provide 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 intelligent cities will be evaluated. In Section III, the methodology for smart parking systems will be highlighted. Section IV will also display findings and discussion. Section V covers the conclusion and future remarks.

2. Literature Survey

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

Current large-city studies indicate that there are several methods for managing parking, including the amount of traffic on the roads. When parking their cars, drivers become frustrated by the difficulty of finding a parking space. Drivers often waste time and effort looking for parking spaces and parking on the street, which congests the parking lot. In the worst-case scenario, people might not be able to get a parking spot, especially during the holidays or other busy times.

[4] described a smart parking system where users could use a website to check for open parking spaces. They could also reserve them if they weren't previously reserved using LED lights and sensors. The recommended architecture can cut down on time spent searching for parking and ease traffic congestion. However, the user must be connected to the same WIFI network to use 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 radiofrequency to a subsystem.

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

To identify parking space openings, outdoor parking systems often use sensor-based or computer vision-based systems, including camera-based surveillance systems.

[7] described a system that uses cameras and Classifier Neural Networks. This requires a specialist server with a lot of processing capacity and the ability to analyze and predict 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] includes a KNN approach to 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] has claimed that image processing methods can be used to locate a parking space. RGB colors, which are fundamental hues, were used to identify empty spaces. The green color denotes when a car is not parked in its designated spot, the blue color denotes a space, and the red color denotes no parking lot.Alenezi et al. (2021) developed a novel Convolutional Neural Network (CNN) integrated with a block-greedy algorithm to enhance underwater image dehazing. The method addresses color channel attenuation and optimizes local and global pixel values. By employing a unique Markov random field, the approach refines image edges. Performance evaluations, using metrics like UCIQE and UIQM, demonstrated the superiority of this method over existing techniques, resulting in sharper, clearer, and more colorful underwater images [12]. In their study presented at the 13th International Conference on Computing Communication and Networking Technologies, Vikas et al. (2022) examined the classification of agricultural land using machine learning algorithms. The research, illustrated using Zhashui County as an example, established a Rating Factor System through methods like the Delphi method and straight-line method. They assessed various factors related to the land, calculating a classification index and achieving preliminary land classification. Furthermore, the study emphasized the role of weather and soil characteristics in agricultural decision-making, highlighting the potential for more accurate and cost-effective land classification through satellite images and machine learning algorithms [13].

Our system uses a small amount of computer resources. Because the modules perform all fundamental and necessary computations, a simple server manages the database. Our system is scalable without increasing processing demands. Our recommended smart outdoor parking solution does this without staff, providing orderly and secure outdoor parking.

3. Methodology

3.1. System requirement

DroidCam

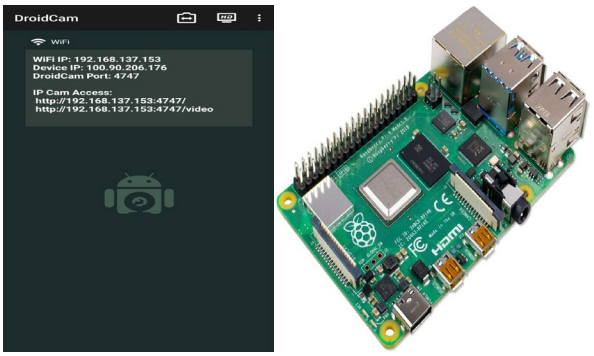
With the help of the DroidCam program, users may use their Android tablet or smartphone as a wireless camera for their PC. Both free and premium versions are available from the Google Play Store.

Users may use their Android device as a camera for video chats, online meetings, live streaming, and other purposes. This is done by using DroidCam to connect their

Android handset to their computer via Wi-Fi or USB. The program provides numerous customization options, such as changing camera settings and video quality. A variety of video software products are also supported.

DroidCam is a beneficial and practical option for anyone who needs a webcam, 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 running on the Raspberry Pi. DroidCam in-use demo is shown in Fig. 1. (a).



(a) (b)

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

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 centers. It can run several 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+ runs 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 utilizing computer vision techniques to identify occupied parking spaces, our ML model develops a smart parking system. Two colors—green and blue—are recognized 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 color detection using the HSV color system. The code evaluates the parking lot's occupancy condition based on the proportion of green and blue colors seen. It 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 color detection approach. The code uses the cv2.inRange() method to construct a mask for the green and blue hues in the image. It then counts the number of green and blue pixels in the mask to calculate the percentage of each color in the image. After determining whether it has picked up both colors or only one, the function modifies the occupancy status accordingly.

The Firebase Admin SDK is then initialized by providing the service account key file location. 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 occupied, the code gathers images from each camera. It converts it to RGB color 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 utilize Flutter for the front-end of our mobile applications. Google developed the free and open-source Flutter mobile app development framework. It enables programmers to create high-performance, natively designed applications from a single codebase for desktop, web, and smartphone platforms. Developers may design stunning and responsive user interfaces with Flutter, a programming framework that uses the Dart programming language. It offers a wide variety of pre-built and customized widgets.

One of Flutter's key advantages is its rapid reload functionality. This lets programmers update the code and see the results without restarting 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 take advantage of all of its capabilities. This will enable them to develop beautiful and intriguing mobile apps. In general, Flutter is a robust and adaptable framework for creating mobile apps that are rapidly gaining favor with both developers and companies.

One screen in our smartphone app 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. Available and occupied parking spaces are 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 format rather than traditional SQL databases.

Real-time synchronization is one of Cloud Firestore's distinctive characteristics. All connected clients receive automatic 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 a bonus, Cloud Firestore offers a powerful querying mechanism that spares developers from writing time-consuming server-side code to filter, sort, and paginate data on the client side. This enables speedy and efficient data retrieval from huge databases. Cloud Firestore also supports transactions and batch writes, which ensures data consistency and reduce server round trips.

Cloud Firestore is integrated with Firebase, offering another benefit. As a result, developers can easily include authentication, cloud communications, and cloud features into their programs without building a challenging infrastructure.

The Cloud Firestore stores the number of occupied and vacant parking spaces from the ML model [10]. Mobile apps fetch real-time data and show it to users. Fig. 2 shows the same.

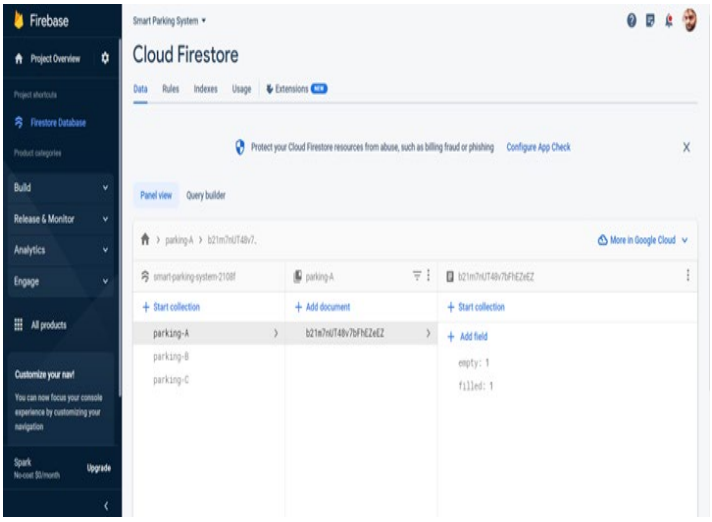


Figure 2. Backend support of the proposed system

3.5. System Design

Here's the step-by-step algorithm for identifying 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 color 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 colors green and blue.

(iv) Calculate the percentage of green and blue color in the image.

(v) See if you can distinguish the colors 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) Utilizing 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 color 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) Utilize 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

The diagrammatical structure of the brief system design is shown in Fig. 3.

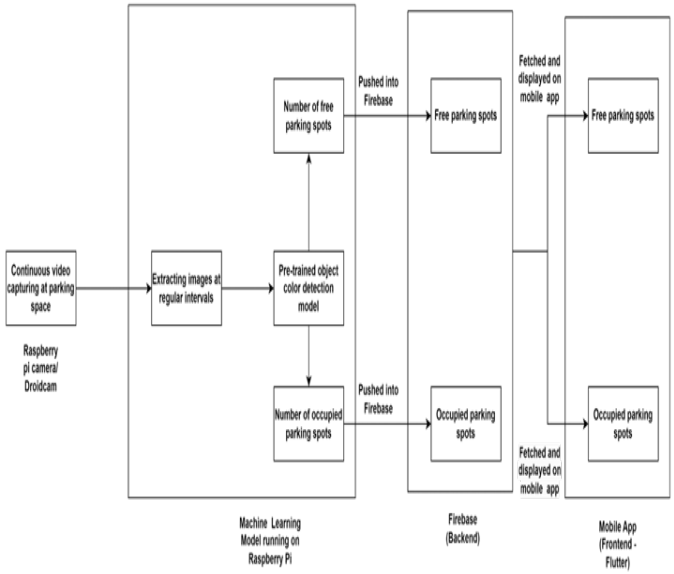


Figure 3. System Design

4. Cost And Comparison with existing systems

This section presents a detailed breakdown of the estimated costs associated with implementing the intelligent Raspberry Pi-based parking slot identification system. By accurately assessing the expenses, we aim to provide an understanding of the financial implications and feasibility of the proposed system [11].
This section also presents a comparative analysis of the cost associated with implementing an intelligent Raspberry Pi-based parking slot identification system in contrast to existing systems.

4.1. System Components

The Hardware Components which are used to make the following identification system are:

- Raspberry Pi boards
- Hard Cardboard plank
- Black, white, and yellow Chart papers.
- SD card
- High-speed micro-SD card reader
- Miniature toy Cars

4.2. Quantity and Cost of Components

The Quantity and Cost of Components are shown in Table 1.

Table 1. Quantity and Cost of Components

Component	**Quantity**	**Unit Cost (₹)**	**Total Cost (₹)**
Raspberry Pi board	1	₹ 5,256	₹ 5,256
Hard cardboard plank	1	₹150	₹150
Chart papers (Black, white and yellow)	3	₹12	₹36
Miniature toy Cars	2	₹199	₹398
SD Card	1	₹459	₹459
High-speed micro-SD card reader	1	₹59	₹59
Sum Total			₹6,358

4.3. Software and Development Cost

The Software's Components that are used to make the following identification system are:

- DroidCam Android App (Free Google Play Store application)
- Firebase Realtime Database (Open-Source real-time database)
- Flutter (Open-Source mobile app development platform)

As all of the above Software components are free of cost and are open source, the cost relating to software components is zero.

From sections 4.2 and 4.3 we deduce the total cost of the prototyping model to be INR ₹6,358/-.

4.4. Comparison with existing systems

In the prototyping scale, various existing systems can be compared to an intelligent Raspberry Pi-based parking slot identification system. Here are a few examples along with their approximate costs in INR:

Ultrasonic Sensor-based System

Description: A system that utilizes ultrasonic sensors to detect the presence of vehicles in parking slots.
Cost: The cost of an ultrasonic sensor-based system can range from ₹2,500 to ₹6,000, including the cost of sensors, Arduino, or similar microcontrollers, wiring, and other necessary components.

Infrared Sensor-based System

Description: A system that employs infrared sensors to detect vehicle occupancy in parking slots.
Cost: The cost of an infrared sensor-based system can range from ₹3,000 to ₹7,000, depending on the number of sensors, microcontrollers, wiring, and other associated components.

Arduino-based Parking System

Description: A system based on Arduino microcontrollers that can detect and monitor parking slot occupancy.
Cost: The cost of an Arduino-based parking system can range from ₹3,000 to ₹8,000, depending on the number of parking slots, sensors used, and additional components required.

All these systems mentioned above require multiple sensors in each parking slot. This will cause the overall cost to rise much more than its expected price range.

This also increases the complexity of these systems, introducing many problems such as Delay, Challenges in System Integration, Greater Risk of Errors, and reduced ease of use.

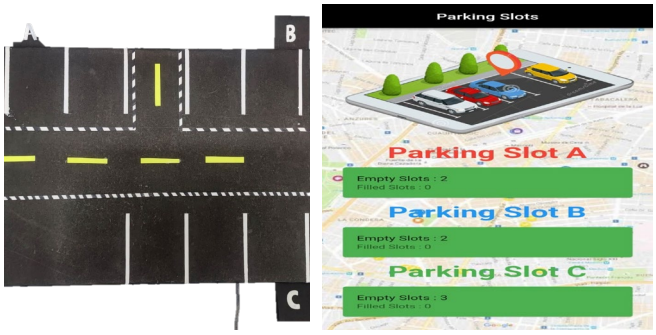
Comparing the costs, it is evident that the intelligent Raspberry Pi-based parking slot identification system (on a small scale) priced at ₹6,358/- (INR) offers a competitive advantage over the existing systems mentioned above in terms of affordability and functionality.

5. Results and Discussion

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

5.1. Completely empty parking place (let's assume parking A as an example)

The ML model in this instance fails to identify a sizable fraction of green or blue color. Thus, no slots are detected as occupied. The app shows the same thing. The parking spot widget in the app stays green, suggesting there are still spaces available. Fig. 4. represents the input (parking condition) and output (Empty and filled slots detection) for case-1.

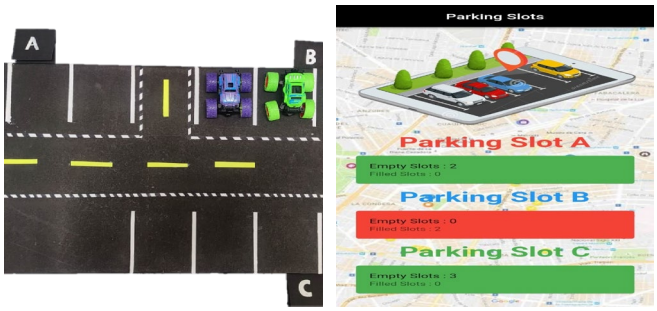


(a) (b)

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

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

In this instance, a lot of green and blue colors are detected by the ML model. Therefore, the number of vacant slots is zero. The app shows the same thing. When no parking spaces are available, the parking spot widget turns red. Fig. 5. represents the system input and output for case-2.

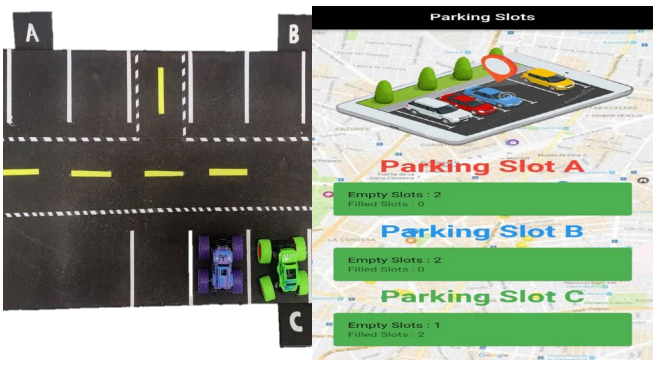


(a) (b)

Figure 5. Fully filled parking (a) Parking condition (b) Empty and filled slots detection.

5.3. 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 there are still spaces available. Fig. 6 represents the case 3 condition.



(a) (b)

Figure 6. Fully filled parking (a) Parking condition (b) Empty and filled slots detection.

Hence, the proposed system is detecting the free and occupied slots effectively as shown in Table 2.

Table 2. System observation for all the cases

Case	Parking Slot	% of Green	% of Blue	Occupied Slots	Empty 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

6. Conclusion and future remarks

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

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

Automatic Parking Slot Occupancy Detection is an excellent concept for 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 from a stream of video images. This was to improve the city's parking infrastructure and reduce emissions while boosting locals' quality of life. The structure developed helps reduce pollution since navigation is regulated. Time is saved, and movement is easier, thanks to technology. The Android app offers accurate location data.

The experimental investigation suggests that additional modifications to the current model can enhance the parking system. Future implementations will make it easier 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, 14(2), 33-4, (2014).

[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, (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).

[10] Agarwal, R., Suthar, J., Panda, S. K., & Mohanty, S. N. (2023). Fuzzy and Machine Learning based Multi-Criteria Decision Making for Selecting Electronics Product. *EAI Endorsed Transactions on Scalable Information Systems*, *10*(5). https://doi.org/10.4108/eetsis.3353.

[11] Agarwal, R., & Godavarthi, D. (2023). Skin Disease Classification Using CNN Algorithms. *EAI Endorsed Transactions on Pervasive Health and Technology*, *9*. https://doi.org/10.4108/eetpht.9.4039.

[12] Alenezi, F.; Armghan, A.; Mohanty, S.N.; Jhaveri, R.H.; Tiwari, P. Block-Greedy and CNN Based Underwater Image Dehazing for Novel Depth Estimation and Optimal

Ambient Light. Water 2021, 13, 3470. https://doi.org/10.3390/w13233470

[13] K. Vikas, K. V. Goud, S. Ponaganti, A. N. Reddy, K. Nagasai and S. N. Mohanty, "Agricultural Land Classification Based on Machine Learning Algorithms," 2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kharagpur, India, 2022, pp. 1-4, doi: 10.1109/ICCCNT54827.2022.9984254.