WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

REAL TIME PARKING LOT MONITORING USING ADAPTIVE IMAGE PROCESSING

Rushi Eshwar Reddy Neelam [1] ***,K. Rupa Sri*** [2] ***,K.Rushindra*** [3] ***,D. Rushika*** [4] ***,Thala Rohith*** [5] ***,Dr. Anjaiah Pole*** [6]

[1,2,3,4,5] B.Tech Scholars, [6] Assistant Professor

Department of CSE - Artificial Intelligence and Machine Learning Malla Reddy University, Hyderabad, India

ABSTRACT:

This parking spot detection system employs classical computer vision techniques to efficiently monitor and manage parking space occupancy using real-time video processing. Unlike machine learning-based approaches that require extensive datasets and training, this system relies on traditional image processing methods, making it lightweight, cost-effective, and easy to deploy. The system analyzes live video frames and determines the status of each parking space by examining grayscale intensity values within predefined parking regions. These coordinates are stored in a YAML configuration file, allowing for easy customization and scalability. The occupancy detection mechanism is based on statistical analysis—if the mean grayscale intensity within a parking spot surpasses a specified threshold and the standard deviation remains below a certain value, the system classifies the spot as occupied. By leveraging this rule-based approach, the system effectively differentiates between empty and filled parking spots without requiring deep learning models or extensive computational power. To improve user experience, the system overlays visual indicators onto the video feed, marking occupied spaces with one color and available spots with another, ensuring immediate recognition of parking availability. Additionally, a real-time counter dynamically updates to reflect the number of vacant spaces, allowing for quick decision-making by drivers or parking lot administrators. The system also includes interactive controls, enabling users to pause the video for detailed analysis, capture specific frames for reference, or exit the application as needed. These features provide flexibility in monitoring parking lot activity while maintaining an intuitive interface. One of the major advantages of this approach is its efficiency in processing video feeds without the need for high-end hardware, making it suitable for real-time applications. Unlike deep learning-based parking detection systems that require extensive training and substantial computational resources, this system operates effectively on low-power devices, making it an ideal choice for smart parking solutions in commercial, residential, and public spaces. Additionally, it offers seamless integration with existing parking management systems, enabling better space utilization, reduced congestion, and improved traffic flow in urban areas. The system's adaptability also makes it valuable for security surveillance, as it can assist in monitoring vehicle movement and unauthorized parking. Beyond parking lot management, this system can be extended to support more advanced features such as automated ticketing, predictive analytics for peak-hour estimations, and integration with mobile applications to help drivers locate available spots in real time. Future enhancements could also incorporate sensor-based validation, night-time detection improvements, and cloud-based storage for historical parking data analysis. By utilizing traditional image processing techniques instead of complex machine learning models, this system provides a scalable, cost-efficient, and practical solution for real-world parking management, making it a valuable asset for urban infrastructure development.

Keywords : Parking Space Detection, Computer Vision, Image Processing, Real-time Monitoring, Artificial Intelligence (AI)

I.Introduction

Parking space management has become a significant challenge in densely populated urban areas, where the increasing demand for parking spaces often surpasses the available supply. This imbalance contributes to severe traffic congestion, higher fuel consumption, increased pollution levels, and overall inconvenience for both residents and visitors. Inefficient parking allocation can also lead to wasted time for drivers searching for available spots, further exacerbating urban traffic issues. To address these pressing concerns, we propose a comprehensive parking management system that integrates advanced technologies, real-time data analytics, and strategic policy interventions to optimize parking space utilization. By leveraging smart sensors, Internet of Things (IoT) devices, and machine learning algorithms, the proposed system will enable dynamic monitoring of parking availability, guiding drivers to vacant spots efficiently while reducing unnecessary vehicle movement. Real-time data processing and predictive analytics will further enhance the system's capability to anticipate peak-hour demand, allowing for better space allocation and traffic flow optimization. In addition to technological solutions, the project will explore the effectiveness of policy interventions such as dynamic pricing strategies, shared parking initiatives, and incentives for alternative modes of transportation like public transit, cycling, and ride-sharing services. Implementing dynamic pricing can help regulate demand by adjusting parking fees based on occupancy levels, encouraging more balanced space utilization. Shared parking agreements between commercial and residential properties can also maximize the use of existing infrastructure. Furthermore, offering incentives for drivers who opt for eco-friendly transport options can contribute to reducing the number of private vehicles on the road. The combination of cutting-edge technology and well-designed policy measures aims to create a sustainable, efficient, and driver-friendly parking ecosystem. By minimizing congestion, improving parking accessibility, and

reducing environmental impact, this project seeks to enhance urban mobility, promote sustainability, and improve the overall quality of life for city dwellers.

II.Literature Review

Several research studies have explored various techniques for improving object detection and parking surveillance through advanced computational methods. In the study titled *"Technological Improvement Rate Predictions for All Technologies: Use of Patent Data and an Extended Domain Description" (2021)* by Anuraag Singh, Giorgio Triulzi, and Christopher L. Magee, the authors employed the Classification Overlap Method (COM). This technique is widely used in object detection to enhance classification accuracy by evaluating overlapping bounding boxes. It selects the bounding box with the highest confidence score, reducing redundant detections and improving object localization efficiency. Similarly, in the research paper *"A Smart, Efficient, and Reliable Parking Surveillance System With Edge Artificial Intelligence on IoT Devices" (2020)* by R. Ke, Y. Zhuang, Z. Pu, and Y. Wang, the Single Shot Multibox Detector (SSD) was utilized for vehicle and object detection. SSD enhances parking surveillance by efficiently identifying vehicles and obstacles using a system of default bounding boxes with multiple aspect ratios and scales. By predicting class scores and positional offsets for each box, SSD enables rapid detection of vehicles, unauthorized entries, and potential safety hazards, ensuring improved parking space monitoring. Another study, *"An Image Feature-Based Method for Parking Lot Occupancy" (2019)* by Tatulea, Paula, Florina Calin, Remus Brad, Lucian Brancovean, and Mircea Greavu, implemented the Histogram of Oriented Gradients (HOG) for background subtraction, a fundamental step in computer vision applications. Before applying this method, edge detection is performed using the Canny algorithm, and images are resized to 64 × 128 pixels. This preprocessing step enhances the accuracy of parking occupancy detection. Moreover, in *"A Smart Image Processing-Based System for Parking Space Vacancy Management" (July 2018)* by Kommey, Benjamin, Addo, Ernest, and Agbemenu, Andrew, the Canny Edge Detection Algorithm was employed to identify the edges of vehicles within parking spaces. This algorithm effectively detects vehicle boundaries, determining whether a parking spot is occupied or vacant by analyzing edge presence within designated regions. Additionally, the research paper *"Computer Vision-Based Parking Optimization System" (2021)* by Siddharth Chandrasekaran, Jeffrey Matthew Reginald, Wei Wang, and Ting Zhu introduced the Space Identification Algorithm to enhance parking space utilization. These algorithms play a crucial role in real-time detection of unoccupied spaces by processing sensor data and image feeds. By analyzing camera footage or sensor readings, the system updates the parking database with real-time availability information, which is then relayed to drivers via mobile applications or electronic displays. This approach streamlines the parking process, reduces congestion, and maximizes the efficiency of parking facilities. These studies collectively demonstrate how advanced image processing and computer vision techniques contribute to the development of intelligent parking management systems. By leveraging methods such as COM, SSD, HOG, Canny Edge Detection, and Space Identification Algorithms, researchers are improving parking surveillance, optimizing space utilization, and enhancing the overall efficiency of urban parking solutions.

III. Methodology

OpenCV (cv2)

OpenCV (*cv2*) is an open-source computer vision library widely used for real-time image and video processing. It provides various functions for reading, processing, and displaying video frames, making it an essential tool for applications like object detection, motion tracking, and automated surveillance. OpenCV includes powerful operations such as Gaussian blurring for noise reduction, color space conversions (e.g., RGB to grayscale or HSV), contour detection to identify object boundaries, and shape-drawing functions for marking detected objects within an image or video. These features make OpenCV a fundamental library for developers working on machine vision and artificial intelligence projects.

NumPy (np)

NumPy (*np*) is a core library for numerical computing in Python that provides support for large, multi-dimensional arrays and matrices. It includes a vast collection of mathematical functions that enable efficient computation, matrix transformations, and array manipulations. In computer vision applications, NumPy plays a crucial role in optimizing data processing and performance, particularly when handling large image datasets. It facilitates seamless mathematical operations such as filtering, transformations, and feature extraction, which are critical in applications like image recognition and real-time video analysis.

YAML (Yet Another Markup Language)

YAML is a lightweight and human-readable data serialization format commonly used for configuration files. In video-based applications such as parking spot detection, YAML files store predefined coordinates for specific regions of interest, such as parking spaces or object locations. This structured format allows developers to easily manage and modify configuration data without altering the main code. The readability and simplicity of YAML make it an excellent choice for storing structured data, enabling efficient parameter tuning in machine vision applications.

VideoCapture (cv2.VideoCapture)

The *VideoCapture* class in OpenCV allows for capturing video from various sources, including files, webcams, and external cameras. It reads frames from an input video file or live stream, enabling real-time processing for applications such as parking space detection, traffic monitoring, and security

surveillance. The ability to extract and process frames from a continuous video feed is essential for detecting objects, tracking motion, and performing automated analysis in smart surveillance systems.

Displaying and Controlling Video Playback

Functions like *cv2.imshow()* and *cv2.waitKey()* are crucial for displaying processed images and videos in OpenCV applications. *cv2.imshow()* creates a window to display frames, while *cv2.waitKey()* listens for keyboard input, allowing users to pause, resume, or exit the application. These functions enable interactive control of video playback, making them essential in applications requiring user input, such as parking space monitoring or interactive surveillance.

Saving Frames with cv2.imwrite()

The *cv2.imwrite()* function allows users to save specific frames as image files. In video processing applications, capturing key frames can be beneficial for documentation, later analysis, or debugging. In an interactive system, this feature is often triggered by user input, such as pressing the 'c' key to capture and store a frame for further evaluation. This capability is useful in applications like automated parking surveillance, where snapshots of detected vehicles can be stored for record-keeping.

By integrating these libraries and functions, developers can build powerful and efficient image-processing applications, making OpenCV, NumPy, and YAML essential tools in the field of computer vision and artificial intelligence.

IV.Architecture

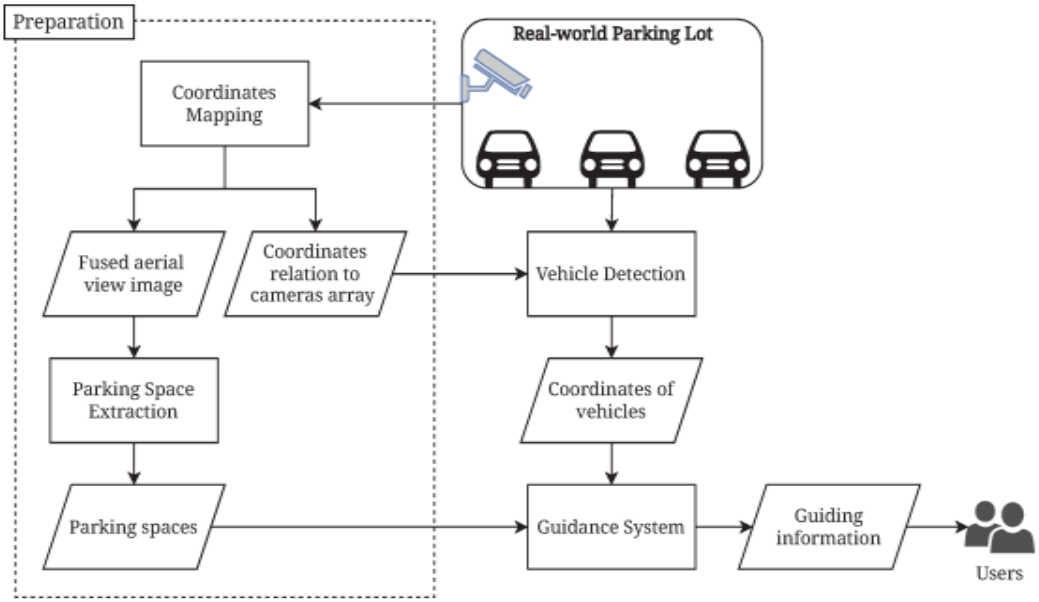

fig 1: architecture

1. Input Handling

The system begins by processing a video file (*video.mp4*), which contains footage of a parking area. This video acts as the primary data source for detecting available and occupied parking spots. The input video can be captured in real-time from a CCTV camera or a pre-recorded file, allowing flexibility in different parking environments such as commercial parking lots, residential complexes, and public garages.

2. Frame-by-Frame Video Processing

The video is read and processed frame by frame using OpenCV's *VideoCapture* class. This ensures that each frame is analyzed independently while maintaining continuity in detecting changes over time. By processing the video in real-time, the system can monitor parking space occupancy dynamically and update the display as vehicles enter or leave.

3. Preprocessing Techniques

To improve the accuracy of detection, each frame undergoes several preprocessing steps:

- *Gaussian Blurring:* This step helps smooth out noise in the video, reducing unnecessary details that could interfere with the detection algorithm. The function *cv2.GaussianBlur()* applies a filter to reduce image variations and ensure a more uniform intensity distribution.
- *Grayscale Conversion:* Since color information is not required for parking space detection, each frame is converted to grayscale using *cv2.cvtColor()*. This transformation simplifies the analysis by focusing only on intensity variations within the image.
- *Edge Detection (Optional):* To enhance feature detection, additional techniques such as Canny edge detection may be applied. This helps highlight boundaries and contours of vehicles, making it easier to differentiate between occupied and empty spaces.

4. Parking Spot Detection Algorithm

The core functionality of the system relies on an intelligent algorithm that detects and analyzes parking spots. The detection workflow includes:

- *Loading Parking Spot Coordinates:* The system reads pre-defined parking spot coordinates from a YAML file, which stores the specific locations of parking spaces within the frame. This ensures a consistent and reusable detection framework.
- *Defining Regions of Interest (ROIs):* Each parking space is assigned a specific region in the frame. The system extracts these ROIs to analyze the content within them.
- *Analyzing Pixel Intensity Variations:* The algorithm examines pixel intensity within each ROI to determine whether a vehicle is present. By comparing intensity changes over time, the system can differentiate between occupied and vacant spaces. If a significant change in pixel values is detected, it indicates a vehicle entering or leaving the space.

5. Visual Representation of Parking Status

To provide real-time feedback to users, the system overlays visual indicators on the processed frames. The following techniques are used for visualization:

- *Bounding Rectangles:* The system draws rectangular outlines around parking spots to highlight their positions.
- *Contours and Overlays:* When a parking spot is occupied, it is marked with a filled contour or colored overlay (e.g., red for occupied and green for vacant).
- *Text Labels:* The occupancy status of each parking spot is displayed on the video frame, making it easy for users to interpret the results at a glance.

6. User Interaction and Interface

The system provides an interactive interface that allows users to monitor the parking status and control specific functions:

- *Displaying Processed Video Frames:* Using *cv2.imshow()*, the system presents a live feed with parking spot overlays.
- *User Controls with Keyboard Input:* Users can interact with the system through keyboard commands using *cv2.waitKey()*. This enables functionalities such as pausing the video, exiting the application, or capturing specific frames for review.
- *Customization Options:* The system can be configured to allow users to adjust parameters such as detection sensitivity, visualization styles, and the ability to add or modify parking spot coordinates.

7. Frame Capture Feature

To facilitate further analysis or record-keeping, users can capture individual frames by pressing the 'c' key. This action invokes *cv2.imwrite()*, saving the current frame as an image file. This feature can be useful for documenting parking violations, generating reports, or debugging the detection process.

8. Output and Data Representation

The system outputs a processed video feed containing:

- *Real-time Parking Spot Overlays:* This provides users with an immediate understanding of space occupancy.
- *Live Occupancy Updates:* As vehicles enter or exit, the status of each parking space is updated dynamically.
- *Data Storage Options:* The system can be extended to log parking occupancy data, enabling historical analysis and optimization of parking space utilization.

V. Design:

5.1 System Overview

The parking space detection system is designed to monitor and analyze parking occupancy using computer vision techniques. It processes a video feed in real-time to determine whether parking spaces are vacant or occupied. The system consists of two main components: *Parking Space Picker* – A utility for manually selecting and storing parking spots, and *Main Detection Module* – The core module that processes video frames and detects parking space occupancy.

5.2 System Architecture

The system follows a modular approach with distinct functionalities.

5.2.1 Input Handling

The system processes a video file (carPark.mp4) as the primary input. A static image (carParkImg.png) is used for manual parking spot selection. Parking spot coordinates are stored and retrieved using a serialized file (CarParkPos).

5.2.2 Preprocessing Module

Converts each video frame to *grayscale* to simplify analysis. Applies *Gaussian Blurring* to remove noise. Uses *adaptive thresholding* to enhance contrast for improved detection.

5.2.3 Parking Spot Detection Algorithm

Loads predefined parking space coordinates from a *Pickle file*. Defines and extracts *Regions of Interest (ROIs)* corresponding to parking spaces. Analyzes *pixel intensity variations* to determine parking occupancy.

5.2.4 Visualization Module

Overlays *bounding rectangles, contours, and text labels* on detected parking spots. Uses *green markers* for vacant spaces and *red markers* for occupied spaces. Displays the *total number of available parking slots* in real-time.

5.2.5 User Interaction Module

Uses *cv2.imshow()* to display processed frames. Waits for user inputs via *cv2.waitKey()* to enable interactions: 'c' key captures and saves a frame for documentation, 'r' key allows resetting parameters.

5.2.6 Output Handling

Continuously updates and displays processed video frames. Saves selected frames as images for further analysis.

5.3 Software Components

5.3.1 Programming Language

Python is used for developing image processing and OpenCV functionalities.

5.3.2 Libraries and Frameworks

OpenCV (cv2) – Facilitates image and video processing. *NumPy* – Supports numerical operations and matrix manipulations. *cvzone* – Enhances visualization and text overlays. *Pickle* – Handles storage and retrieval of parking space coordinates.

5.4 Functional Workflow

5.4.1 Parking Space Selection (ParkingSpacePicker.py)

Loads an image of the parking area. Allows users to select parking spots manually. Saves selected parking spots to a serialized file for future use.

5.4.2 Real-time Parking Detection (Main.py)

Reads video frames sequentially. Applies *grayscale conversion, blurring, and adaptive thresholding*. Loads stored parking spot coordinates and evaluates *pixel intensity*. Determines and classifies parking spaces as *occupied or vacant*. Displays the detection results in an interactive window. Updates the count of *available parking spaces dynamically*.

5.5 Design Considerations

5.5.1 Performance Optimization

The system processes frames *in real-time* for seamless monitoring. *Adaptive thresholding* ensures accuracy under varying lighting conditions. *Median blur and dilation* help refine detection accuracy.

5.5.1.1 Real-time Processing

Uses optimized OpenCV methods to handle video frames efficiently. Reduces unnecessary computations to maintain smooth performance.

5.5.1.2 Noise Reduction Techniques

Gaussian blurring minimizes random noise. Median filtering removes unwanted pixel intensity fluctuations.

5.5.2 Scalability

The system can be adapted to different *parking layouts and environments*. It supports future extensions such as cloud integration and mobile app support.

5.5.2.1 Customizable Parking Spot Selection

Users can define parking spots manually using a graphical interface. The system stores these locations for automated future detection.

5.5.2.2 Expansion for Large Parking Areas

The detection algorithm can be extended to *process multiple video feeds simultaneously*. Further optimizations can be introduced for large-scale parking areas.

5.5.3 Limitations & Future Enhancements

Current Limitation – The system does not use deep learning models, which might affect accuracy in complex conditions. *Future Improvements*: Integrating *AI-based models (YOLO, SSD, Faster R-CNN)* for higher detection precision. Implementing a *mobile notification system* for real-time alerts. Deploying on *edge devices (Raspberry Pi, Jetson Nano)* for practical implementation.

5.6 Conclusion

The parking space detection system successfully integrates *image processing* and *computer vision* techniques to provide an efficient parking management solution. By leveraging *real-time monitoring*, *user interaction*, and *visual feedback*, the system enhances operational efficiency and reduces parking congestion. Future improvements, such as AI integration and IoT connectivity, could further elevate the system's capabilities, making it a robust solution for intelligent urban parking management.

fig 2: output screen 1

fig 3: output screen 2

VI. Conclusion

The image-driven parking space optimization project successfully leveraged advanced computer vision techniques to enhance parking lot management and efficiency. By implementing sophisticated image processing algorithms, the system accurately identified vacant and occupied parking spaces, reducing congestion and improving overall organization within parking facilities. This solution addresses a critical urban mobility issue by offering real-time parking space monitoring, ensuring that drivers receive up-to-date information about available spots.

A major strength of this project lies in its ability to process video footage dynamically, enabling automated detection of parking occupancy without the need for additional hardware such as sensors. Instead of relying on costly infrastructure modifications, the system utilizes existing surveillance cameras or video feeds, making it a cost-effective and scalable solution for various environments, including shopping malls, office complexes, residential areas, and public parking facilities.

Through real-time monitoring and data analytics, the project provides valuable insights into parking patterns, peak usage hours, and space utilization efficiency. This information can assist authorities, parking lot managers, and city planners in making informed decisions about infrastructure development, resource allocation, and traffic flow optimization. Additionally, historical data analysis can help in predicting demand trends, enabling proactive strategies such as dynamic pricing models, reserved parking systems, and improved traffic routing to reduce unnecessary vehicle movement.

Another key aspect of this project is the enhanced user experience it delivers. By integrating parking detection with digital interfaces, such as mobile applications or electronic signage, drivers can receive live updates on available spaces, minimizing the time spent searching for parking and reducing fuel consumption. This not only improves convenience but also contributes to environmental sustainability by cutting down on carbon emissions resulting from unnecessary idling and circling for parking.

Furthermore, the system's adaptability allows for future enhancements. Machine learning algorithms can be integrated to improve detection accuracy, distinguishing between different types of vehicles and identifying special-purpose spaces such as handicapped spots or reserved areas. Additionally, by incorporating Internet of Things (IoT) connectivity, the system can be linked to smart city infrastructure, enabling seamless communication between vehicles, parking facilities, and traffic management systems.

By utilizing technology to streamline parking operations, this project highlights the transformative potential of intelligent solutions in addressing urban mobility challenges. With continued development, such image-based parking optimization systems can play a vital role in building more efficient, sustainable, and user-friendly urban environments. The success of this approach demonstrates that harnessing artificial intelligence and computer vision can significantly enhance parking management, improve traffic flow, and contribute to the broader vision of smart cities.

VII. REFERENCES:

[1] **Singh, A., Triulzi, G., & Magee, C. L. (2021).** Technological improvement rate predictions for all technologies: Use of patent data and an extended domain description.

[2] **Ke, R., Zhuang, Y., Pu, Z., & Wang, Y. (2020).** A Smart, Efficient, and Reliable Parking Surveillance System With Edge Artificial Intelligence on IoT Devices.

[3] **Tatulea, P., Calin, F., Brad, R., Brancovean, L., & Greavu, M. (2019).** An Image Feature-Based Method for Parking Lot Occupancy.

[4] **Kommey, B., Addo, E., & Agbemenu, A. (2018).** A Smart Image Processing-based System for Parking Space Vacancy Management.

[5] **Chandrasekaran, S., Reginald, J. M., Wang, W., & Zhu, T. (2021).** Computer Vision Based Parking Optimization System. ResearchGate

[6] **Boda, V. K., Nasipuri, A., & Howitt, I. (2007).** Design considerations for a wireless sensor network for locating parking spaces.

[7] **Fabian, T. (2008).** An Algorithm for Parking Lot Occupation Detection.arXiv

[8] **Ichihashi, H., Katada, T., Fujiyoshi, M., Notsu, A., & Honda, K. (2010).** Improvement in the performance of camera-based vehicle detectors for parking lots.

[9] **Yusnita, R., Norbaya, F., & Basharuddin, N. (2012).** Intelligent Parking Space Detection System Based on Image Processing.
[10] **True, N. (2007).** Vacant Parking Space Detection in Static Images.
[11] **Zabawi, N. H. B., Sunardi, & Ghazali, K. H. (2013).** Parking lot detection using image processing method.
[12] **Banerjee, S., Choudekar, P., & Muju, M. K. (2011).** Real-time car parking system using image processing.
[13] **Haque, S. M. T., Scielzo, S. A., & Wright, M. (2013).** A study of user password strategy for multiple accounts.
[14] **Sood, S. K., Sarje, A. K., & Singh, K. (2010).** Inverse Cookie-based Virtual Password Authentication Protocol.
[15] **Li, Y., Mao, H., Yang, W., Guo, S., & Zhang, X. (2023).** Research on Parking Space Status Recognition Method Based on Computer Vision. *Sustainability*, *15*(1), 107. MDPI+1MDPI+1
[16] **Nyambal, J., & Klein, R. (2021).** Automated Parking Space Detection Using Convolutional Neural Networks. arXiv
[17] **Almeida, P. R. L. de, Alves, J. H., Oliveira, L. S., Hochuli, A. G., Fröhlich, J. V., & Krauel, R. A. (2023).** Vehicle Occurrence-based Parking Space Detection. arXiv
[18] **Wang, L., Musabini, A., Leonet, C., Benmokhtar, R., Breheret, A., Yedes, C., Burger, F., Boulay, T., & Perrotton, X. (2023).** Holistic Parking Slot Detection with Polygon-Shaped Representations. arXiv
[19] **Grbić, R., & Koch, B. (2023).** Automatic Vision-Based Parking Slot Detection and Occupancy Classification. arXiv
[20] **Li, Y., Mao, H., Yang, W., Guo, S., & Zhang, X. (2023).** Research on Parking Space Status Recognition Method Based on Computer Vision. *Sustainability*, *15*(1), 107. MDPI
[21] **Almeida, P. R. L. de, Alves, J. H., Oliveira, L. S., Hochuli, A. G., Fröhlich, J. V., & Krauel, R. A. (2023).** Vehicle Occurrence-based Parking Space Detection. arXiv
[22] **Wang, L., Musabini, A., Leonet, C., Benmokhtar, R., Breheret, A., Yedes, C., Burger, F., Boulay, T., & Perrotton, X. (2023).** Holistic Parking Slot Detection with Polygon-Shaped Representations.
[23] **Grbić, R., & Koch, B. (2023).** Automatic Vision-Based Parking Slot Detection and Occupancy Classification.
[24] **Nyambal, J., & Klein, R. (2021).** Automated Parking Space Detection Using Convolutional Neural Networks. arXiv