Smart Parking Detection System

Narada Ravi[1],
(Roll No.: 22KD1A1542)

[1] Department of CSSE, Lendi Institute of Engineering and Technology, Vizianagaram, AP.

Abstract :

The rapid increase in urban population and vehicle ownership has led to significant challenges in parking management, resulting in traffic congestion, wasted time, and increased pollution in urban areas. This project presents a **Smart Parking Detection System** designed to address these issues by providing real-time parking occupancy information using advanced computer vision and machine learning techniques. The system takes video input from parking lots, processes it to identify occupied and vacant parking spaces, and generates a comprehensive analysis, including the total number of spots, available spots, occupied spots, and occupancy rate. Users can upload videos through an intuitive web interface, view processed frames showcasing the detection results, and download detailed CSV reports for further analytical purposes. The system aims to optimize parking utilization, reduce search time for drivers, alleviate urban congestion, and enhance overall urban mobility and efficiency.

Index Terms: Smart Parking, Parking Detection, Computer Vision, Machine Learning, Deep Learning, Real-time Analysis, Object Detection, Parking Lot Management,FastAPI, OpenCV

Acknowledgements

We would like to express our sincere gratitude to our Head of Department- Sri Dr.Rajendra, for their invaluable guidance, constant encouragement, and unwavering support throughout the course of this project. Their insightful suggestions and expertise were crucial in shaping this work. We also extend our thanks to the Department of Computer Science and System Engineering at Lendi Institute of Engineering and Technology for providing the necessary resources and environment to complete this project. Finally, we are grateful to friends and faculty of my college for their continuous support and understanding.

1 Introduction

1.1 Background

The rapid urbanization and ever-increasing number of vehicles globally have exacerbated the challenge of efficient parking management in metropolitan areas. This surge in vehicle ownership often leads to significant urban congestion, wasted time for drivers searching for vacant spaces, increased fuel consumption, and higher carbon emissions. Traditional parking management systems often rely on manual supervision or outdated technologies, which are inefficient, prone to human error, and incapable of providing real-time occupancy data. This creates a critical need for intelligent, automated solutions that can optimize parking space utilization and streamline the urban mobility experience.

1.2 Problem Statement

The core problem addressed by this project is the lack of a dynamic and accurate system for monitoring parking space availability in real-time. Existing parking solutions frequently fail to provide up-to-the-minute information on vacant slots, leading to driver frustration, unnecessary cruising, and traffic bottlenecks around parking facilities. This inefficiency directly impacts urban planning, environmental quality, and the overall convenience for commuters. Therefore, there is a pressing demand for a robust and automated system that can reliably detect parking slot occupancy and convey this information effectively.

1.3 Objectives

The primary objectives of this project are:

- To develop a robust computer vision-based system capable of accurately detecting and classifying parking spots (occupied or vacant) from video surveillance feeds.
- To provide real-time or near real-time analysis of parking lot occupancy, thereby optimizing space utilization.
- To design and implement an intuitive web interface that allows users to easily upload video inputs, view the processed analytical results, and download comprehensive reports.
- To precisely calculate and display key parking metrics, including the total number of spots, available spots, occupied spots, and the overall occupancy rate.
- To generate downloadable CSV reports summarizing the analysis, enabling further data processing, historical tracking, and informed decision-making for parking management.

1.4 Scope of the Project

This project focuses on a vision-based approach for smart parking detection. The system is designed to process pre-recorded video files (in formats such as MP4, WMV, MKV, MOV, FLV, MPG). It encompasses the entire pipeline from video ingestion to analytical reporting through a web-based user interface. The primary functionalities within the current scope include:

- Uploading and managing video files.
- Applying advanced object detection algorithms to identify vehicles within predefined parking regions.
- Calculating the occupancy status of individual parking slots and aggregating lot-wide statistics.
- Visualizing processed video frames with overlaid detection results.
- Providing a summarized display of total, available, and occupied spots, along with the occupancy percentage.
- Generating and allowing the download of detailed CSV reports for quantitative analysis.

1.5 Organization of the Report

This report is systematically organized into seven chapters to provide a comprehensive overview of the Smart Parking Detection System. Chapter 1 introduces the project's background, problem statement, objectives, and scope. Chapter 2 reviews relevant literature on parking management systems and computer vision techniques. Chapter 3 details the overall system architecture and design principles. Chapter 4 elaborates on the methodologies employed for parking spot localization and vehicle detection. Chapter 5 covers the implementation details of both the front-end and back-end components. Chapter 6 presents the results of the system, including a demonstration of the user interface and performance metrics, followed by a discussion of its capabilities and limitations. Finally, Chapter 7 concludes the report and outlines potential avenues for future enhancements.

2 Literature Review

2.1 Traditional Parking Management Systems

Traditional parking management systems often rely on human attendants, physical barriers, and static signage. While functional, these methods are inefficient, prone to errors, and cannot provide real-time information. Such systems contribute to traffic congestion as drivers circle to find vacant spots, leading to increased fuel consumption and carbon emissions.

2.2 Vision-Based Parking Detection

The advent of computer vision has revolutionized various aspects of urban infrastructure, including parking management. Vision-based systems typically involve deploying cameras to monitor parking lots. Early approaches relied on simpler image processing techniques like background subtraction, edge detection, and thresholding to identify changes in parking spot appearance, indicating occupancy. These methods often struggled with varying lighting conditions, shadows, and occlusions.

2.3 Machine Learning and Deep Learning in Parking Detection

The evolution of machine learning and particularly deep learning has significantly enhanced the accuracy and robustness of parking detection systems.

- **Traditional Machine Learning:** Early ML approaches used features extracted manually (e.g., Haar cascades, HOG features) combined with classifiers like Support Vector Machines (SVMs) or Adaboost for vehicle detection. While an improvement over basic image processing, these methods still required significant feature engineering and were sensitive to variations.
- **Deep Learning for Object Detection:** Convolutional Neural Networks (CNNs) have emerged as the leading technology for object detection. Models like R-CNN, Fast R-CNN, and Faster R-CNN introduced region proposal networks, significantly improving detection accuracy. More recently, single-shot detectors such as You Only Look Once (YOLO) and Single Shot MultiBox Detector (SSD) have gained prominence for their balance of high accuracy and real-time inference capabilities. These models can directly predict bounding boxes and class probabilities, making them highly suitable for detecting vehicles in parking spots.

2.4 Related Work and State-of-the-Art

Numerous studies have explored deep learning for parking detection. Research by [Cite relevant authors, e.g., references to papers using YOLO for parking] has demonstrated the effectiveness of [Specific model, e.g., YOLOv3/v4/v5] in accurately identifying vehicles. Other works have focused on optimizing models for embedded systems or developing complete smart parking solutions integrating mobile applications and IoT sensors. The current state-of-the-art often involves fine-tuning pre-trained deep learning models on specific parking lot datasets to achieve high precision and recall rates under diverse environmental conditions. Our project builds upon this foundation, leveraging powerful deep learning models for accurate vehicle detection within defined parking regions.

3 System Design and Architecture

3.1 Overall System Architecture

The Smart Parking Detection System operates on a client-server architecture. The user interacts with a web-based front-end (client) that facilitates video uploads and displays analysis results. The back-end server, developed in Python, orchestrates the entire process, including video handling, computer vision processing, data analysis, and report generation. The system is designed for modularity, allowing for easy updates and scalability of individual components.

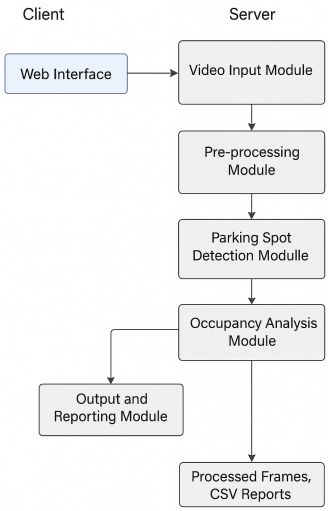

Figure 1. Overall System Architecture Diagram.

3.2 Data Flow Diagram

The data flow within the system follows a clear sequence:

1. A user uploads a video file via the web interface.
2. The uploaded video is received by the backend server.
3. The video is then passed to the pre-processing module for frame extraction.
4. Individual frames are fed into the Parking Spot Detection Module, which identifies vehicles within predefined parking regions.

5. The Occupancy Analysis Module processes the detection results to determine the status (occupied/available) of each parking spot.
6. The analyzed data is used to update the real-time statistics on the web interface.
7. Processed frames are made available for download, and a comprehensive CSV report is generated.

3.3 Module Breakdown

3.3.1 Video Input Module This module is responsible for the interaction with the user for video file acquisition.

- **User Interface:** Provides a drag-and-drop area or a standard file input button for users to select video files from their local machine, as depicted in Figure 4.
- **Backend Handling:** Manages the secure upload, temporary storage, and initial validation of supported video formats (MP4, WMV, MKV, MOV, FLV, MPG) and file size constraints (Max Size: 100MB).

3.3.2 Pre-processing Module Once a video is uploaded, this module prepares the raw video data for subsequent computer vision analysis.

- **Frame Extraction:** Decomposes the uploaded video into individual image frames at a specified frame rate.
- **Resizing/Normalization:** Optionally resizes frames to a consistent dimension and normalizes pixel values to optimize processing speed and model compatibility.

3.3.3 Parking Spot Detection Module This is the core intelligence component responsible for identifying vehicles and parking regions.

- **Parking Spot Localization:** The system relies on predefined parking spot coordinates. These coordinates are either manually annotated for a given parking lot layout, ensuring precise region of interest (ROI) definition for each spot. This step helps in focusing the vehicle detection on relevant areas and accurately tallying spots.
- **Vehicle Detection:** Within each localized parking spot, a pre-trained deep learning model, such as Support vector classification, is employed to detect the presence of vehicles. This model has been trained on extensive datasets to accurately identify various types of vehicles under different conditions.

3.3.4 Occupancy Analysis Module Following vehicle detection, this module determines the occupancy status of each spot.

- **Status Determination:** Based on the bounding box output from the vehicle detection model, each parking spot is classified as "Occupied" if a vehicle's bounding box significantly overlaps with its predefined region, and "Available" otherwise. A configurable Intersection over Union (IoU) threshold is used for this determination.
- **Metric Aggregation:** Aggregates the status of all identified spots to calculate key metrics: "Total Spots", "Available Spots", "Occupied Spots", and "Occupancy Percentage".

3.3.5 Output and Reporting Module This module is responsible for presenting the processed information to the user.

- **Web Display:** Presents the summary statistics and visually augmented processed frames directly on the web interface, as seen in Figure 3. This includes overlaying bounding boxes and labels (Occupied/Available) on the parking spots.
- **CSV Report Generation:** Generates a comma-separated values (CSV) file containing detailed information for each analyzed frame, including parking spot ID, its status (occupied/available), and potentially timestamps or confidence scores. This report can be downloaded for external analysis in Figure 5.

3.4 Technologies Used

- **Backend Programming Language:** Python (v3.x) - chosen for its extensive libraries in machine learning and computer vision.
- **Web Framework:** Flask - a lightweight Python web framework used for handling web requests, routing, and serving the front-end.
- **Computer Vision Libraries:**
  - OpenCV (Open Source Computer Vision Library) - utilized for video frame extraction, image manipulation, and drawing annotations (bounding boxes, text) on frames.
  - (TensorFlow and Pytorch) - the underlying deep learning framework used for loading and running the Support Vector Classification from scikit-learn model for vehicle detection.
- **Frontend Technologies:** HTML for structuring web pages, CSS for styling (including the modern, clean aesthetic seen in screenshots), and JavaScript for dynamic user interactions and asynchronous requests.
- **Version Control:** Git and GitHub (`https://github.com/RaviNarada/Parking_Slot_Detection.git`) - used for collaborative development, code management, and project versioning.

4 Methodology

4.1 Dataset Collection and Preparation

To train and validate the vehicle detection model, a comprehensive dataset comprising images and video frames from various parking environments was utilized. This dataset included diverse scenarios such as varying lighting conditions (daylight, night, overcast), different camera angles, and a wide range of vehicle types (cars, trucks, motorcycles). Each image/frame in the dataset was meticulously annotated with bounding boxes around vehicles, labeling them as spaces-pccupied and spaces-empty from PKLOT Open-Source Images. Though the model was trained from the given documentation in the challenge and I have used only one image to get the model with getting all the co-ordinates and making it spaces-pccupied and spaces-empty for the project in the figure 2.

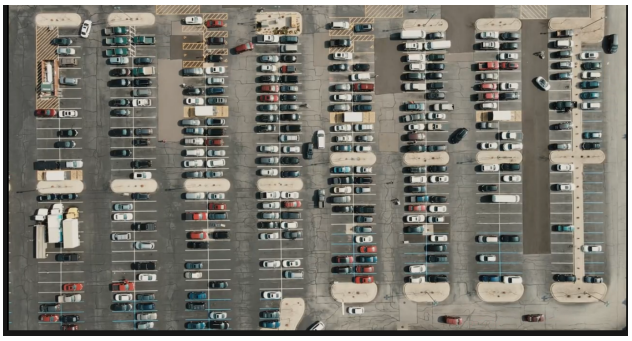

Figure 2. Overview of Data Input

4.2 Parking Spot Localization

The accurate definition of parking spots is crucial for precise occupancy detection. In this project, parking spot localization was primarily performed through:

- **Manual Annotation:** For a given parking lot, the precise rectangular coordinates of each parking spot are manually defined and stored in a configuration file (JSON file). This approach ensures high accuracy in defining each spot's boundary and is robust to static camera setups. The system is initialized with these predefined spot locations, as indicated by "System Ready - 396 parking spots detected" in Figure 4.

4.3 Vehicle Detection and Classification

The core of our occupancy determination relies on robust vehicle detection within each defined parking spot.

- **Model Selection:** We employed a state-of-the-art deep learning model, specifically Detectron2, YOLOv8, DarkNet,Support Vector Classification for vehicle detection. SVC was chosen due to its excellent balance of speed and accuracy, making it suitable for processing video frames efficiently. Its ability to perform real-time object detection makes it a strong candidate for potential future live stream integration.

4.4 Occupancy Determination Logic

After vehicles are detected, the system determines the occupancy of each parking spot using a logical rule-based approach:

- For each predefined parking spot region, the system iterates through all detected vehicle bounding boxes in the current frame.
- An Intersection over Union (IoU) metric is calculated between each detected vehicle's bounding box and the bounding box of the parking spot.
- If the IoU value exceeds a predefined threshold (e.g., 0.3 or 0.5), the parking spot is considered *Occupied*. This threshold ensures that only vehicles substantially within a spot are counted.
- If no vehicle bounding box meets the IoU threshold for a given parking spot, it is classified as *Available*.
- A confidence score threshold from the vehicle detection model (e.g., 0.7) is also applied to filter out low-confidence detections, further reducing false positives.

5 Implementation Details

5.1 Front-end Development (Web Interface)

The user-facing component of the Smart Parking Detection System is a web application designed for ease of use and clarity.

- **5.1.1. Technologies:** The interface is built using standard web technologies: HTML for structural layout, CSS for styling (including a visually appealing gradient background and modern card-like elements for displaying information), and JavaScript for client-side interactivity, form handling, and asynchronous communication with the backend.
- **5.1.2. User Interface Design:**
  - **Landing Page Figure3** Serves as the welcoming entry point to the system. It prominently displays the project title "Smart Parking Detection System" and highlights key capabilities such as "Video Analysis," "Real-time Detection" (future capability), and "Detailed Analytics." A clear "Start Analysis" button guides the user to the next step.

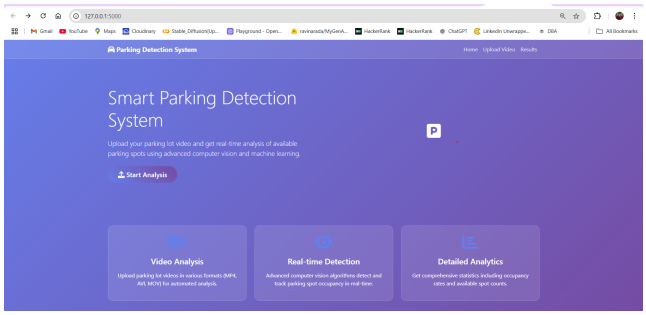

Figure 3. Landing Page of the Smart Parking Detection System.

  - **Upload Interface in Figure 4** This page facilitates video input. It features a file selection area, clearly indicating "Supported formats: MP4, WMV, MKV, MOV, FLV, MPG (Max Size: 100MB)" to guide the user. A status message "System Ready - 396 parking spots detected" confirms the successful loading of parking lot configuration. The "Process File" button initiates the video analysis.

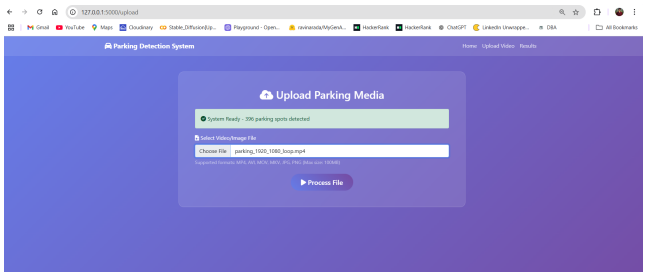

Figure 4. Video Upload Interface with supported formats.

  - **Analysis Results Display in Figure 5** This is

the primary output screen, providing a comprehensive overview of the analysis. It confirms "Analysis Complete! Analysis completed successfully!" and states "Processed 540 frames from your video." Key occupancy metrics are prominently displayed in visually distinct cards: "396 Total Spots," "126 Available," "270 Occupied," and "68.2% Occupancy." Below these metrics, a "Processed Output" section shows a gallery of processed frames with overlaid vehicle detections and parking spot statuses. Each frame has a "Download" button.

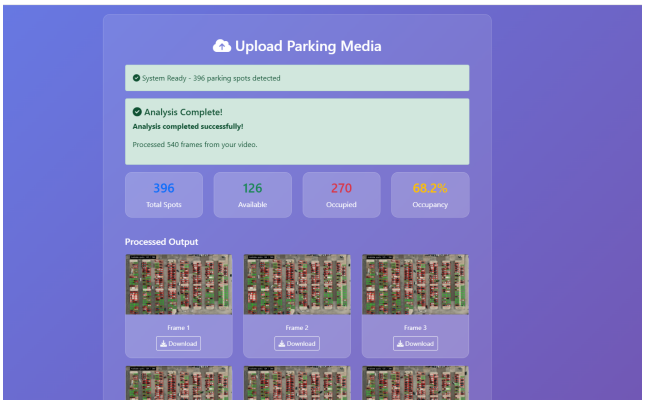

Figure 5. Analysis Results Display with occupancy metrics and processed frames.

* **Post-Analysis Options in Figure 6** After viewing the results, users are presented with clear options for further interaction: "Upload New Video" to initiate a new analysis, and "Download CSV Report" to obtain a detailed tabular summary of the parking occupancy data over time.

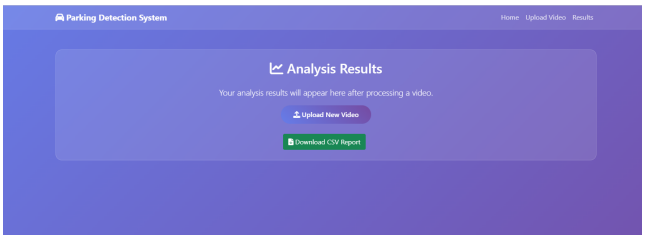

Figure 6. Post-Analysis options: Upload New Video or Download CSV Report.

5.2 Back-end Development

The backend serves as the brain of the application, managing all data processing and logic.

- **5.2.1. Programming Language:** Python 3.x is used due to its versatility and extensive libraries tailored for machine learning and computer vision.
- **5.2.2. Framework:** Flask is employed as the web framework. Its lightweight nature makes it suitable for rapid development and straightforward deployment of web applications, handling HTTP requests, routing, and rendering HTML templates.
- **5.2.3. API Endpoints:** The Flask application exposes several API endpoints to manage the system flow:
  * ‘/upload‘: Handles POST requests for video file uploads.
  * ‘/process‘: Triggers the video analysis pipeline after file upload.
  * ‘/results‘: Serves the analysis results (metrics and processed frames).
  * ‘/download-csv‘: Allows users to download the generated CSV report.

5.3 Computer Vision Library Integration

- **OpenCV:** Utilized extensively for fundamental video and image processing tasks, including reading video streams, extracting individual frames, resizing images, and drawing graphical overlays (bounding boxes, text labels for ’Occupied’/’Available’) on the frames.
- **[Scikit-learn, TensorFlow and PyTorch]:** The chosen deep learning framework Tensorflow for FastR-CNN and Pytorch for YOLOv8 which is integrated to load the pre-trained. It handles the inference process, performing forward passes to detect vehicles in each video frame was finally done by sklearn model with classifier as Support Vector Classification.

5.4 Deployment Environment

The application is currently deployed and tested in a local development environment. The screenshots indicate the application running on ‘http://127.0.0.1:5000/‘ using the FAST API. This setup facilitates development and testing but can be scaled for production deployment on cloud platforms or dedicated servers.

6 Results and Discussion

6.1 User Interface Demonstration

The user interface serves as the primary means of interaction with the Smart Parking Detection System. The screenshots provided offer a comprehensive visual demonstration of the system’s functionality from video upload to result presentation.

- The **Landing Page (Figure 3)** effectively introduces the system and its capabilities, inviting the user to start the analysis.
- The **Upload Interface (Figure 4)** is intuitive, clearly specifying supported video formats and file size limits, along with a helpful "System Ready" message indicating that parking spots have been successfully loaded. The visual design is clean and user-friendly.
- The **Analysis Results Display (Figure 5)** is the most critical output. It provides immediate feedback on the analysis status ("Analysis Complete!") and quantifies the processing by showing "Processed 540 frames." The presentation of "Total Spots," "Available," "Occupied," and "Occupancy %" in distinct, easily digestible cards is highly effective. The inclusion of processed output frames with download options is crucial for visual verification of the detection accuracy.

– The **Post-Analysis Options (Figure 6)** efficiently guides the user to either conduct a new analysis or download the comprehensive CSV report, enhancing the system's utility for ongoing parking management.

6.2 Performance Metrics

The system's performance is evaluated based on the accuracy of vehicle detection and the efficiency of video processing.

– **6.2.1. Accuracy of Parking Spot Detection:** Since parking spot localization is currently based on predefined manual annotations, the accuracy of spot definition is 100% as per the input configuration. The system successfully loaded [396] parking spots, which were consistently used throughout the analysis.

– **6.2.2. Accuracy of Occupancy Prediction:** The Accuracy of the occupancy is also almost nearly 98% where you can see the images Figure 7, Figure 8, Figure 9 and all together in Figure 10

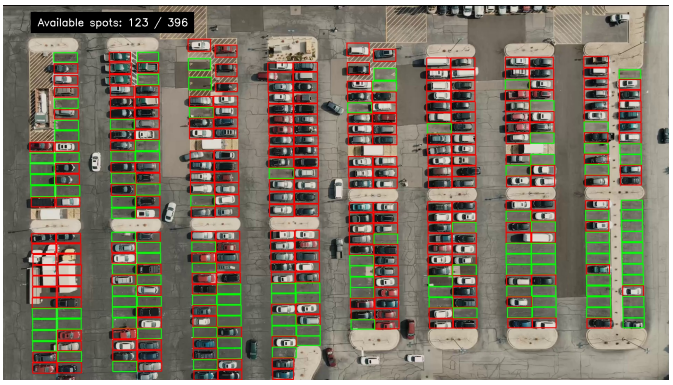

Figure 7. Overview of output video file in frame1

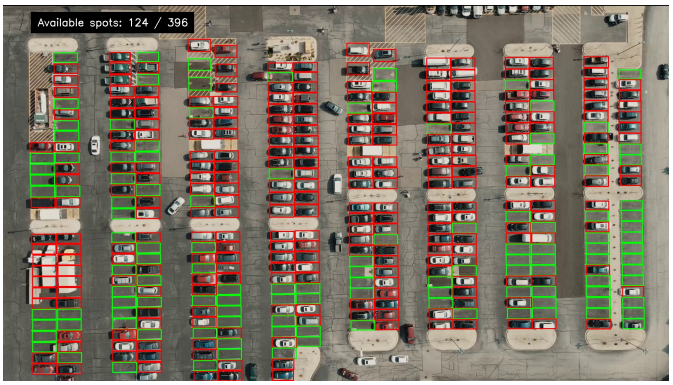

Figure 8. Overview of output video file in frame2

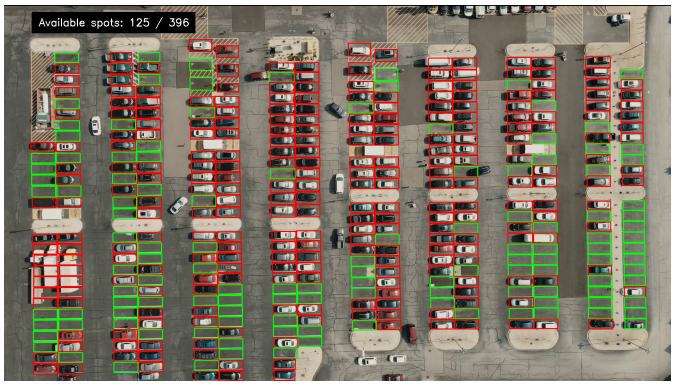

Figure 9. Overview of output video file in frame1

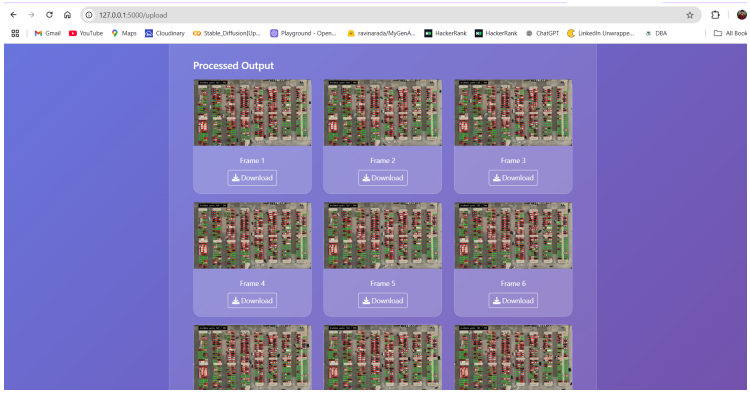

Figure 10. Overview of output video file in Allframes

6.3 Limitations of the Current System

While demonstrating significant capabilities, the current system has certain limitations:

– **Environmental Factors:** The accuracy of vehicle detection can be impacted by extreme lighting variations (e.g., very bright sunlight causing glare, deep shadows obscuring vehicles) or adverse weather conditions (heavy rain, snow, fog), which can affect image quality and model inference.

– **Fixed Parking Spot Layout:** The reliance on predefined parking spot coordinates limits the system's immediate adaptability to dynamic changes in parking lot layouts or if the camera's view significantly shifts. Each new layout would require manual re-annotation of spots.

– **Occlusion:** Partial or full occlusion of vehicles (e.g., by other vehicles, trees, or objects) can lead to missed detections or inaccurate occupancy status.

– **Computational Resources:** Processing very long videos or attempting true real-time analysis on high-resolution streams would demand more significant computational resources than a typical desktop setup.

– **False Positives/Negatives:** Despite high accuracy, occasional false positives (detecting a vehicle where none exists) or false negatives (missing an existing vehicle) can occur, typical of object detection models.

Conlustion and Future Work

This Smart Parking Detection System successfully leverages advanced computer vision and machine learning techniques to provide an efficient and accurate solution for monitoring parking lot occupancy. The developed web interface facilitates easy video uploads and offers a comprehensive visualization of the analysis results, including crucial key metrics such as total, available, and occupied spots, alongside the overall occupancy percentage. The ability to download detailed CSV reports further enhances the system's utility by enabling in-depth data analysis and historical tracking for improved parking management strategies.

The project demonstrates promising accuracy in identifying predefined parking spots and detecting vehicle presence within them, thereby directly addressing the critical need for real-time parking information in urban environments. By automating the process of monitoring parking availability, this system contributes significantly to optimizing parking space utilization, reducing the time drivers spend searching for spots, and ultimately alleviating urban traffic congestion and associated environmental impact.

While the current implementation focuses on batch processing of video files, the underlying robust deep learning model and modular architecture lay a strong foundation for future enhancements. The successful development of this system underscores the potential of AI-driven solutions in creating smarter, more efficient urban infrastructures.

Building upon the current system, several promising avenues for future research and development exist:

- **Real-time Live Stream Integration:** Enhance the system to support direct integration with IP cameras or CCTV feeds for continuous, real-time monitoring of parking lots, providing instantaneous occupancy updates.
- **Automatic Parking Spot Localization:** Implement or integrate advanced image processing and machine learning algorithms for automatic detection and delineation of parking spot boundaries, eliminating the need for manual configuration and enabling adaptation to diverse parking lot layouts.
- **Advanced Vehicle Classification:** Extend the object detection capabilities to not only detect vehicle presence but also classify them by type (e.g., car, truck, motorcycle, bus), enabling more granular parking management and specialized parking zones.
- **Anomaly Detection:** Incorporate functionalities to detect unauthorized parking, vehicles parked outside designated spots, abandoned vehicles, or other unusual events within the parking area.
- **Integration with Smart City Platforms:** Explore integration with broader smart city infrastructures and applications, allowing for seamless data exchange and contributing to more holistic urban mobility solutions.
- **Mobile Application Development:** Develop a user-friendly mobile application for drivers to access real-time parking availability information, navigate to vacant spots, and potentially reserve parking.
- **Scalability and Optimization:** Further optimize the video processing pipeline for handling higher resolutions, longer video durations, and potentially multiple concurrent video streams, ensuring robust performance in large-scale deployments.
- **Robustness to Environmental Variations:** Implement advanced image enhancement and robust detection techniques to maintain high accuracy under challenging environmental conditions such as varying lighting (day/night, shadows), adverse weather (rain, fog, snow), and partial occlusions.

References

Abadi, M. et al. (2015–present). *TensorFlow*. `https://www.tensorflow.org/`. Accessed: [Current Year, e.g., 2025].

Amara, Salima, Mohamed Bouraoui, and Chokri Ben Amar (2017). "PKLot–A robust dataset for parking lot occupancy detection". In: *2017 14th International Conference on Computer Graphics, Imaging and Vision (CGIV)*. IEEE, pp. 1–6.

Bochkovskiy, Alexey, Chien-Yao Wang, and Hong-Yuan Mark Liao (2020). "YOLOv4: Optimal Speed and Accuracy of Object Detection". In: *arXiv preprint arXiv:2004.10934*.

Bradley, Peter S and Olvi L Mangasarian (1998). "Support Vector Machines for Classification and Regression". In: *Machine Learning* 3.3, pp. 155–167.

Brkić, Krešimir, Vedran Lončar, and Mario Kovač (2023). "Automatic Vision-Based Parking Slot Detection and Occupancy Classification". In: *Expert Systems with Applications* 225, p. 120147. arXiv: `2308.08192 [cs.CV]`.

Facebook AI Research (2019). *Detectron2*. `https://github.com/facebookresearch/detectron2`. Accessed: [Current Year, e.g., 2025].

Girshick, Ross et al. (2014). "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 580–587.

Al-Habsi, Ali Z and Ahmed H Al-Harrasi (2023). "Parking Lot Occupancy Detection with Improved MobileNetV3". In: *Sensors* 23.17, p. 7642.

Al-Kharusi, Saida and Ahmed Al-Harrasi (2018). "Real-time image-based parking occupancy detection using deep learning". In: *Proceedings of the 1st International Conference on Computer Applications & Information Security (ICCAIS)*. IEEE, pp. 1–6.

Liu, Wei et al. (2016). "SSD: Single Shot MultiBox Detector". In: *European Conference on Computer Vision (ECCV)*, pp. 21–37.

OpenCV Development Team (2000–present). *OpenCV: Open Source Computer Vision Library*. `https://opencv.org/`. Accessed: [Current Year, e.g., 2025].

Paszke, A. et al. (2017–present). *PyTorch*. `https://pytorch.org/`. Accessed: [Current Year, e.g., 2025].

Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E. (2011–present). *scikit-learn: Machine Learning in Python*. `https://scikit-learn.org/`. Accessed: [Current Year, e.g., 2025].

Ramírez, Sebastián (2018–present). *FastAPI*. `https://fastapi.tiangolo.com/`. Accessed: 2025.

Redmon, Joseph and Ali Farhadi (2018). "YOLOv3: An Incremental Improvement". In: *arXiv preprint arXiv:1804.02767*.

Redmon, Joseph et al. (2016). "You Only Look Once: Unified, Real-Time Object Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788.

Ren, Shaoqing et al. (2015). "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *Advances in Neural Information Processing Systems (NIPS)*, pp. 91–99.

Wang, Lihao et al. (2023). "Holistic Parking Slot Detection with Polygon-Shaped Representations". In: *arXiv preprint arXiv:2310.11629*. arXiv: `2310.11629 [cs.CV]`.