Real-Time Parking Lot Occupancy Monitoring using YOLOv8 with Parking Slot Alignment Transformation Algorithm

Shuban M S[1], Aneek Kumar Saha[1], Jervis Francis Lopes[1], Vijayarajan Rajangam[2][0000−0003−0562−4472] (Fellow IETE), and Sangeetha Nagarajan[3]

[1] School of Electronics Engineering, Vellore Institute of Technology, Chennai, India
{shuban.ms2021, aneekkumar.saha2021, jervisfrancis.lopes2021}@vitstudent.ac.in
[2] CHAIR, School of Electronics Engineering, Vellore Institute of Technology, Chennai, India
viraj2k@gmail.com (Corresponding Author)
[3] School of Computer Science and Engineering, Vellore Institute of Technology, Chennai, India
sangeethana2016@gmail.com

Abstract. This study presents a camera-based approach for real-time parking occupancy monitoring across a campus parking lot using minimal hardware — a single pre-existing Pan-Tilt-Zoom closed circuit television camera mounted on a nearby building. The system captures video footage, extracts frames at one-second intervals, and compiles them into a dataset, which is then divided into training, validation, and test sets. Data augmentations including resizing to 640 x 640 and horizontal flipping were applied to enhance model performance to generalize on the dataset. A comparative study on various You Look Only Once (YOLO) model versions identified YOLOv8 as the top performer, achieving a precision of 99.6%, mAP50 of 99.4%, and mAP50-95 of 93.6%, while maintaining real-time processing capabilities. We developed a custom Parking Slot Alignment Transformation algorithm (PSAT) to convert detection outputs into a slot-occupancy map specific to the parking layout. Testing of the slot center transformation achieved up to 87.9% accuracy on average and when occupancy was high achieved up to 80.8% accuracy, with performance adjusting dynamically based on the number of cars in the lot. Results confirm that PSAT is both efficient and scalable, maintaining accuracy and responsiveness even at near-maximum capacity.

Keywords: parking occupancy · YOLOv8 · Parking Alignment Transformation

1 Introduction

Effective parking management is a critical component in urban and campus environments [1], where limited parking space and high vehicle volumes often lead to congestion and frustration. Traditional systems, such as sensor-based or manually monitored methods, fall short in addressing

these challenges due to high costs, scalability issues, operational delays, and limited adaptability to dynamic parking conditions. For example, sensor-based systems require expensive hardware installations and maintenance, while manually monitored systems are prone to human error and inefficiency. These limitations emphasize the need for an adaptive, scalable solution that can handle the dynamic nature of modern parking lots.

In this study, we use a camera-based parking occupancy monitoring solution that operates using only one Pan-Tilt-Zoom (PTZ) closed circuit television camera (CCTV) [2], already installed and functional on a nearby building, eliminating the need for new hardware installations and enabling seamless integration with existing software. The single-camera setup posed unique challenges like perspective distortion or perspective scaling, which are particularly present in parking monitoring, as vehicles at different distances may appear with significant size variations, complicating detection and slot assignment. Another challenge faced was the lack of predefined slot markings on the ground or uniform parking orientation for vehicles.

Thus, there was a need to build a custom algorithm capable of dynamically mapping vehicles to virtual slot areas based on vehicle position and size. To address this, we developed an adaptive slot mapping algorithm that detects and maps vehicles to available spaces with an average accuracy of 87.9%. This algorithm works in hand with YOLOv8 [3, 4], using the predicted bounding boxes to map them to the available slots.

Our dataset was built from the CCTV footage, capturing a frame every second. Data augmentations such as horizontal flipping was applied to improve model generalization. A comparative study of various YOLO model versions shows that YOLOv8 had the best-performing metrics [5], achieving a precision of 99.6%, mAP50 of 99.4%, and mAP50-95 of 93.6%, making it ideally suited for real-time deployment.

In addition to high accuracy, the proposed system offers practical advantages in accessibility and real-time interaction. Occupancy data is updated in real-time on a web-based platform accessible via desktop and mobile devices, providing users with a convenient, up-to-date view of available parking spaces. By integrating with existing hardware and infrastructure, this solution not only reduces operational costs but also contributes to improved parking efficiency, reducing search times and optimizing lot usage.

2 Literature Review

YOLO models have revolutionized object detection due to their high speed and accuracy. This section reviews relevant literature on object detection models, parking lot monitoring systems, and techniques for handling angled camera setups.

2.1 Object Detection with YOLO Models

Object detection techniques serve as an important piece in the field of artificial intelligence, the YOLO framework since its introduction, has undergone numerous advancements resulting in versions such as YOLOv5, YOLOv8, YOLOv10, and the most recent YOLOv11.

One prominent application of YOLO models is vehicle detection [6]. Comparative studies reveal that YOLOv5 exhibits slightly higher precision and recall values than YOLOv8, with a precision improvement of 1.63% and a recall improvement of 2.49%. When evaluated using the F1 score, YOLOv5 achieved a score of 0.958, whereas YOLOv8 scored 0.938 [7].

2.2 Parking Lot Monitoring Systems

Efficient parking management has become crucial due to the growing number of vehicles and limited urban parking spaces. Smart Parking Management Systems (SPMS) [8] leverage Internet of Things (IoT), Arduino components, infrared (IR) sensors, WIFI module, and mobile applications to enable users to check availability and reserve parking spots.

2.3 Camera Limitations and Transformation Techniques

Object detection [9] identifies objects like cars or people in images, often requiring pose information, such as bounding boxes or transformations. Angled camera setups introduce challenges like perspective distortion, complicating object alignment.

Homography techniques [10] address these issues by mapping image elements from different perspectives on the same plane. Widely used for tasks like pose reconstruction and image alignment, these methods improve detection accuracy and mapping in complex parking scenarios.

2.4 Gaps in Existing Research

While object detection techniques, including YOLO algorithms, have achieved remarkable success, existing research often overlooks real-world complexities such as angled camera perspectives, varying lighting conditions, and occlusions in parking lots.

Additionally, limited work has been done to integrate robust transformation techniques for accurate mapping of parking spaces. Current studies also lack solutions tailored for dynamic, real-time monitoring of unstructured parking lots without predefined slot markings, further highlighting the need for customized algorithms to address these challenges effectively.

2.5 Research Contribution

This paper proposes a novel parking lot monitoring system that combines YOLOv8 with a custom PSAT algorithm to address real-world parking challenges. Our approach overcomes perspective distortions, maps bounding boxes to parking slots accurately, and dynamically identifies free and occupied spaces.

The system operates with a single angled camera, eliminating the need for additional hardware or predefined slot markings. Through this integration, we deliver a scalable, efficient, and real-time solution for urban parking management, contributing a significant advancement in smart city applications.

3 Methodology

3.1 Data Collection and Pre-Processing

The dataset for this study was collected from footage of the campus parking lot captured using a pre-installed PTZ (Pan-Tilt-Zoom) CCTV camera. The camera was positioned on a nearby building, providing an angled view of the entire parking lot as shown in Fig. 1. This angle enabled coverage of all parking spaces, although it introduced a perspective effect that made objects appear smaller at greater distances.

To create a detailed dataset, four hours of continuous footage was recorded in 4K resolution, yielding frames of 3840x2160 pixels. From this footage, we extracted frames at one-second intervals, forming a dataset that spans various lighting conditions, from direct overhead sunlight to

lower, angled sunlight as the sun began to set. This diverse lighting allowed the model to handle the wide range of brightness and shadows typically encountered in real-world parking scenarios. Once frames were extracted, each image was annotated. Bounding boxes were drawn around each visible car to create precise labels for model training.

The annotations were saved in YOLO format, generating a corresponding .txt file for each image. Each .txt file contains the coordinates of the bounding boxes, which define the positions of vehicles within the frame. This format is compatible with YOLO models and facilitates streamlined training. Additionally, a YAML configuration file was created to manage dataset paths and class labels for the model training phase.

After annotation, the dataset was divided into training, validation, and test subsets to facilitate model evaluation and prevent overfitting. To prepare the images for YOLOv8, we resized each frame to 640x640 pixels, as required by the model's input specifications. Additionally, a percentage of the images were randomly selected for horizontal flipping to introduce variability and reduce any model bias.

Fig. 1. Angled camera view of the Parking Lot.

3.2 YOLO Model Selection and Training

Model selection We initially conducted a comparative study using various YOLO versions, including YOLOv5, YOLOv8, YOLOv10, and

YOLOv11, to identify the most effective model for real-time parking occupancy detection. Each model was trained on the entire dataset and validated on the test set, with performance assessed based on accuracy, speed, and overall model error (loss). Among these versions, YOLOv8 consistently outperformed others, displaying optimal precision and mAP scores while also maintaining lower latency during inference.

YOLO was selected for this project over other object detection models due to its well-established balance of accuracy and speed in real-time applications. YOLO's architecture is particularly optimized for quick detection tasks, which is essential for real-time monitoring. Unlike other models that may sacrifice speed for accuracy, YOLOv8's updated architecture provides both. YOLO's single-shot detection approach and efficient architecture make it ideal for this application, where rapid frame processing and reliable object detection are crucial.

Training parameters For training, we configured YOLOv8 with a batch size of 16 over 100 epochs. The image input size was set to 640x640 pixels, and a consistent learning rate of 0.001 was applied throughout training. We observed an average epoch time of approximately 15 seconds on a GPU, allowing efficient iterative updates without significant computational constraints. Key training configurations included:

- Batch Size and Epochs: A batch size of 16 was chosen to balance memory efficiency and model convergence, while 100 epochs were sufficient for model stability and high detection accuracy.
- Augmentation: To introduce variability, mosaic and random affine transformations were also utilized, increasing robustness against minor changes in orientation and car positioning.
- Early Stopping: Early stopping mechanisms were implemented to monitor validation metrics, avoiding overfitting by halting training if improvements were minimal over five consecutive epochs.

Model Evaluation and Deployment Optimization The YOLOv8 model [11] was evaluated on several key metrics, including precision, mAP50, mAP50-95, and recall, achieving high scores that validated its suitability for real-time detection. In particular, YOLOv8 achieved a precision of 99.6%, an mAP50 of 99.4%, and an mAP50-95 of 93.6%.

For deployment, we fine-tuned the model to ensure efficient integration with the web-based application, optimizing it for minimal latency.

The model's detection on the given frame is shown in Fig. 2 with all the detected cars and their bounding boxes in blue along with their confidence scores. The final model achieved a response time of approximately 200 milliseconds per frame, making it well-suited for live applications on both desktop and mobile platforms. This low-latency model enables users to view parking availability in real time.

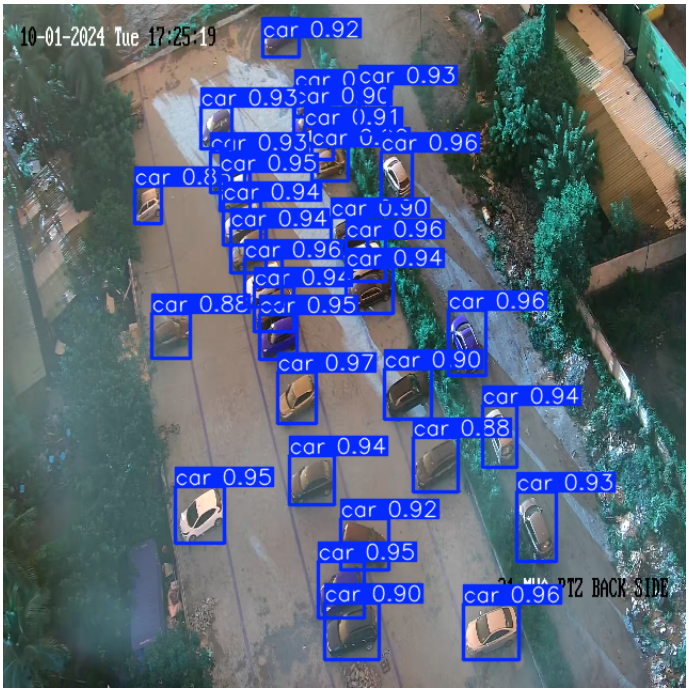

Fig. 2. Parking lot with detected cars bounding boxes.

3.3 Parking Slot Alignment Transformation

The PSAT algorithm is pivotal for mapping detected car bounding boxes to virtual parking slots in lots without predefined markings. This adaptable approach re-calibrates dynamically to accommodate changing parking layouts, camera positions, and angles, ensuring reliable occupancy detection in varied environments.

PSAT addresses challenges like perspective distortion, occlusion from overlapping cars, and irregular parking orientations. By integrating our

trained YOLOv8 model and it's precise detection, bounding boxes are aligned with virtual slot boundaries using geometric transformations, maintaining accuracy even in crowded or obstructed conditions.

The algorithm's flexibility enables it to function across diverse parking lot configurations and adapt to dynamic changes, offering a robust solution for real-time parking monitoring.

Section Lines and Slot Positioning Each parking section is defined by two boundary points, which are the start and end coordinates of the section line. There are three section lines in the parking lot, the middle section is a bit shorter holding up to 20 cars while the other two sections hold up to 22 cars making the total parking lot capacity 64 cars. These boundary points are given by:

$$\text{start} = (x_{\text{start}}, y_{\text{start}}), \quad \text{end} = (x_{\text{end}}, y_{\text{end}}) \tag{1}$$

These points represent the coordinates of the parking section's start and end along a straight line as show in Fig. 3. The section line are given in purple, the parking lot boundary is given in red and the detected cars bounding boxes are marked in green along with their center point in blue. Each of the section lines are evenly spaced apart giving way for cars to drive and use the parking lot. There is also some space at either ends of the middle section line.

For each slot s, where s ranges from 0 to num_slots $-$ 1, we calculate the center position c_{slot} of each slot. The formula for determining the center of slot s is:

$$c_{\text{slot}} = \left(x_{\text{start}} + s \cdot \frac{x_{\text{end}} - x_{\text{start}}}{\text{num\_slots}},\ y_{\text{start}} + s \cdot \frac{y_{\text{end}} - y_{\text{start}}}{\text{num\_slots}} \right) \tag{2}$$

This equation ensures that the slots are evenly distributed along the section line. The distance between consecutive slot centers is calculated by dividing the total length of the section by the number of slots in the section. This helps in precisely mapping the detected cars to their corresponding parking slots.

Slot Dimensions Each parking slot has a defined width (w_{slot}) = 35.58 and length (l_{slot}) = 43.49, which form the boundaries of the slot. To represent each slot's bounding box, we define the slot's center coordinates

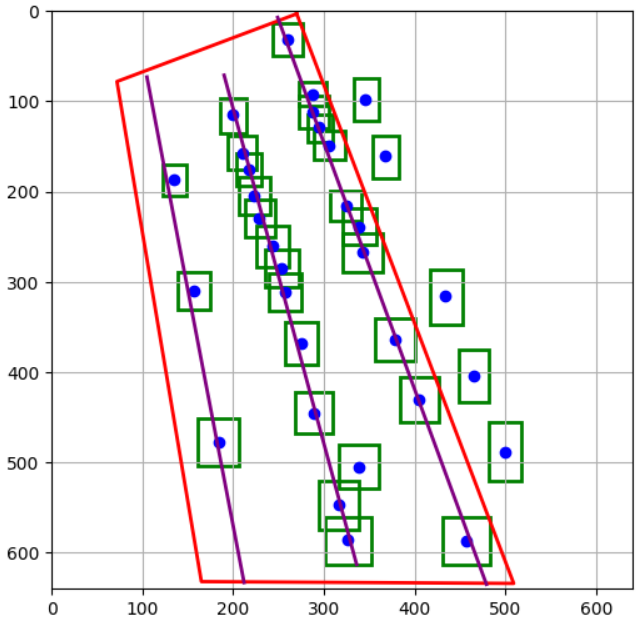

Fig. 3. Parking lot boundary with section lines.

$c_{\text{slot}} = (x, y)$ and calculate the coordinates of the slot's bounding box as follows:

$$\text{slot\_box} = \left(x - \frac{w_{\text{slot}}}{2},\ y - \frac{l_{\text{slot}}}{2},\ x + \frac{w_{\text{slot}}}{2},\ y + \frac{l_{\text{slot}}}{2} \right) \tag{3}$$

Here, the bounding box is defined by the top-left and bottom-right corners. The slot's width and length are used to compute the extent of the box, ensuring that it accurately represents the area of each parking slot. This bounding box is used to check if a detected car overlaps with the slot.

Bounding Box Rotation and Transformation In our parking occupancy detection system, the camera captures an angled view of the parking lot. As a result, the parking slots are not aligned with the standard horizontal and vertical axes of the image, making it challenging to directly compare the detected car bounding boxes with parking slot boundaries. To address this issue, we apply a rotation transformation to each detected car's bounding box, ensuring it aligns with the predefined ori-

entation of the parking slots. This transformation is performed around the center of the bounding box, using a rotation angle θ calculated based on the relative alignment between the bounding box and the parking slot grid [12].

The first step in this process is to determine the center of the bounding box, which serves as the pivot point for rotation. Given a bounding box defined as bbox $= (x_1, y_1, x_2, y_2)$, where (x_1, y_1) and (x_2, y_2) are the coordinates of the top-left and bottom-right corners, the center c_{bbox} is computed as:

$$c_{\text{bbox}} = \left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2} \right) \tag{4}$$

This center point c_{bbox} represents the geometric midpoint of the bounding box and is critical for performing the rotation.

Next, we calculate the angle θ, which represents the inclination of the bounding box relative to the parking slot grid. This angle is derived from the slope of the diagonal of the bounding box. Using the coordinates of the bounding box corners, the angle θ is computed as:

$$\theta = \arctan \left(\frac{y_2 - y_1}{x_2 - x_1} \right) \tag{5}$$

The calculated angle θ provides the measure of rotation needed to align the bounding box with the orientation of the parking slots, accounting for the camera's angled perspective.

Once the center and rotation angle are determined, we apply the rotation transformation to each corner of the bounding box. For a point (x, y) representing one corner of the bounding box, the rotated coordinates (x', y') are computed using the following equations:

$$x' = \cos(\theta) \cdot (x - c_x) - \sin(\theta) \cdot (y - c_y) + c_x \tag{6}$$

$$y' = \sin(\theta) \cdot (x - c_x) + \cos(\theta) \cdot (y - c_y) + c_y \tag{7}$$

Here, c_x and c_y are the x- and y-coordinates of the bounding box center c_{bbox}. This transformation rotates each corner of the bounding box around its center by the angle θ, aligning the bounding box with the parking slot layout. This rotation is applied to all corners, ensuring that the bounding box is correctly transformed while preserving its dimensions.

By aligning the bounding boxes with the parking slots through this rotation process, the system can accurately determine the overlap between detected cars and parking slot regions. This alignment is essential for achieving precise slot occupancy determination, even in challenging scenarios where the camera's perspective introduces angular distortions in the image.

Intersection over Union Calculation To determine whether a car occupies a specific slot, we calculate the Intersection over Union (IoU) between the car's bounding box and the slot's bounding box [13]. The IoU metric is used to quantify how much the car's bounding box overlaps with the parking slot's bounding box.

The IoU is given by:

$$\text{IoU} = \frac{\text{Area of Intersection}}{\text{Area of Union}} \tag{8}$$

To compute the area of intersection between the two bounding boxes, we first determine the coordinates of the intersection region:

$$\begin{aligned} \text{inter\_x1} &= \max(x_{\text{slot\_box1}}, x_{\text{bbox1}}), \\ \text{inter\_y1} &= \max(y_{\text{slot\_box1}}, y_{\text{bbox1}}) \end{aligned} \tag{9}$$

$$\begin{aligned} \text{inter\_x2} &= \min(x_{\text{slot\_box2}}, x_{\text{bbox2}}), \\ \text{inter\_y2} &= \min(y_{\text{slot\_box2}}, y_{\text{bbox2}}) \end{aligned} \tag{10}$$

The area of intersection is then:

$$\begin{aligned} A_{\text{intersection}} = \max(0, \text{inter\_x2} - \text{inter\_x1}) \cdot \\ \max(0, \text{inter\_y2} - \text{inter\_y1}) \end{aligned} \tag{11}$$

The areas of the bounding boxes are computed as:

$$A_{\text{slot}} = (x_{\text{slot\_box2}} - x_{\text{slot\_box1}}) \cdot (y_{\text{slot\_box2}} - y_{\text{slot\_box1}}) \tag{12}$$

$$A_{\text{bbox}} = (x_{\text{bbox2}} - x_{\text{bbox1}}) \cdot (y_{\text{bbox2}} - y_{\text{bbox1}}) \tag{13}$$

Finally, the union area is:

$$A_{\text{union}} = A_{\text{slot}} + A_{\text{bbox}} - A_{\text{intersection}} \tag{14}$$

The IoU is then calculated using (8) as:

$$\text{IoU} = \frac{A_{\text{intersection}}}{A_{\text{union}}} \tag{15}$$

This IoU value helps determine if a detected car is within a parking slot. If the IoU exceeds a certain threshold, we consider the car to be occupying the slot.

Pseudocode for PSAT Algorithm Below is the pseudocode for the PSAT algorithm.

Algorithm 1 Parking Slot Alignment Transformation Algorithm

Require: Detected bounding boxes (`detected_bboxes`), slot positions (`slot_positions`), slot dimensions (`slot_dimensions`)
Ensure: Slot status (occupied or free)

for each bounding box *bbox* in `detected_bboxes` **do**
  rotated_bbox ← `rotate_bbox`(*bbox*, *angle*)
  for each slot in `slot_positions` **do**
    slot_box ← `create_slot_box`(*slot*, `slot_dimensions`)
    iou ← `calculate_iou`(*rotated_bbox*, *slot_box*)
    if *iou* > 0.5 **then**
      `mark_slot_as_occupied`(*slot*)
    else
      `mark_slot_as_free`(*slot*)
    end if
  end for
end for

3.4 System Architecture

The proposed system is designed to use existing infrastructure and object detection models along with the PSAT algorithm to deliver an adaptable solution for parking space management. The architecture shown in Fig. 4 integrates camera feeds, YOLO models, PSAT algorithms, cloud services, and end user applications.

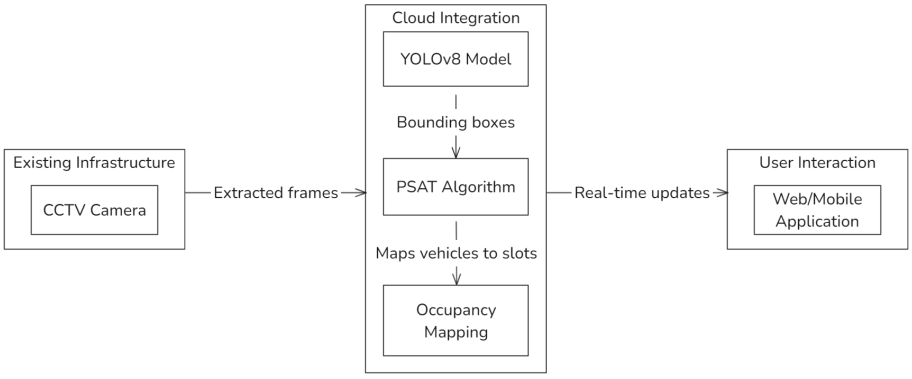

Fig. 4. Proposed system architecture.

- **CCTV Camera (Input Source):** The system uses an existing Pan-Tilt-Zoom (PTZ) CCTV camera installed on a nearby building. This eliminates the need for additional hardware installation.
- **YOLOv8 Model (Detection Module):** Captured frames are processed by the YOLOv8 object detection model, which identifies vehicles and generates bounding boxes.
- **PSAT Algorithm (Mapping Module):** The PSAT algorithm aligns the detected vehicle bounding boxes with predefined virtual parking slots. This step compensates for the angled camera view and dynamic slot configurations, ensuring accurate mapping irrespective of parking lot layouts or changes in camera positioning.
- **Occupancy Mapping (Processing Module):** The output of the PSAT algorithm is used to classify parking slots as either occupied or free. This information is updated in real-time and prepared for user access.
- **Web/Mobile Application (User Interface):** Real-time occupancy data is presented to end-users through an intuitive web and mobile application as shown in Fig. 5 and Fig. 6. Users can view available parking slots. This interface offers accessibility and convenience, promoting efficient parking utilization.

This architecture forms the backbone of the proposed parking lot monitoring system, enabling efficient operation under diverse and dynamic conditions.

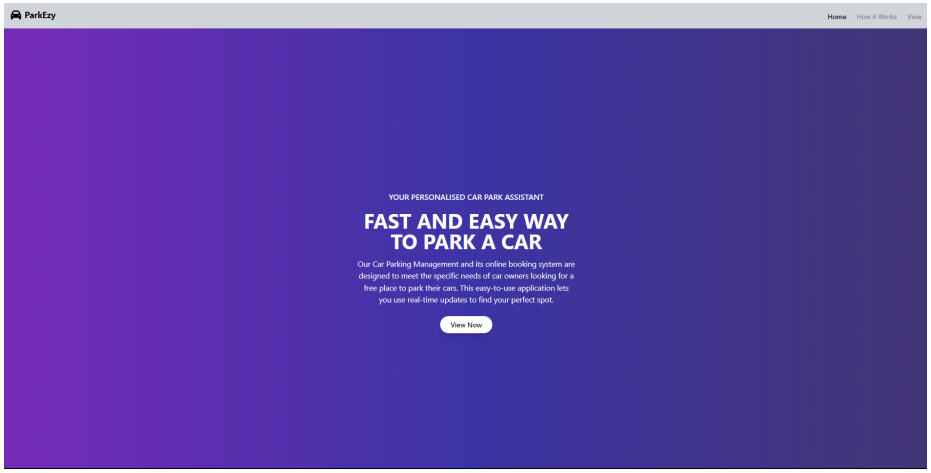

Fig. 5. Website front page.

Fig. 6. Website parking lot section.

4 Results and Discussions

4.1 Training Metrics of YOLO Models

The YOLO models were trained using the custom dataset captured from the parking lot under various lighting and environmental conditions. The dataset includes 4 hours of video footage sampled at 1-second intervals. The training was conducted on a high-performance GPU. The learning performance metrics on the train subset of all the YOLO models over 100 epochs is shown in Fig. 7

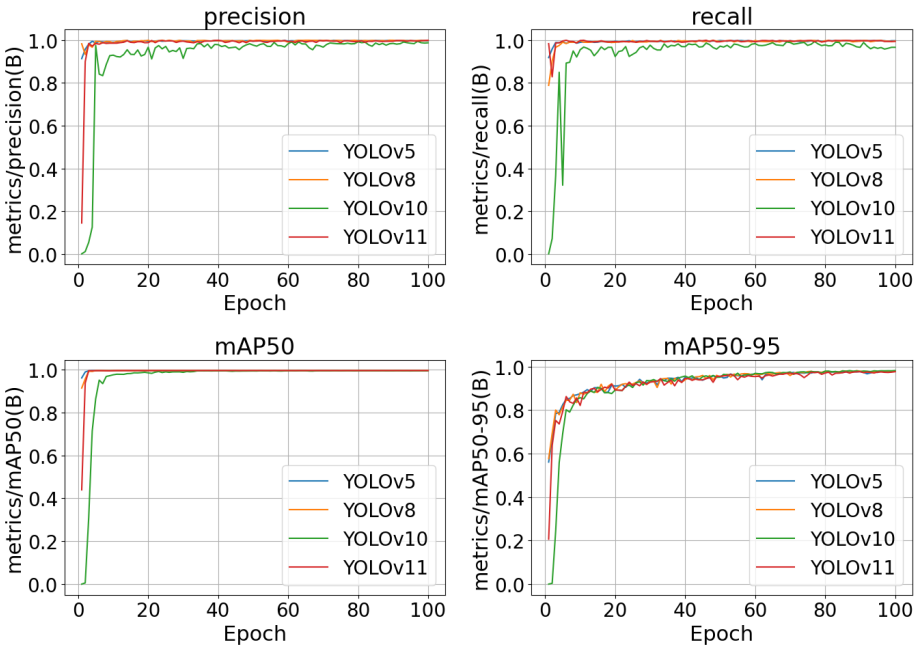

Fig. 7. Train metrics over 100 epochs of all models.

4.2 Test Metrics of YOLO Models

To assess real-world performance, the trained YOLO models were evaluated on a test set comprising parking lot images with varying conditions, including occlusions, perspective distortions, and environmental factors. Table 1 provides the test metrics of each model. Fig. 8 shows a visual comparison of the test metrics of all the models

Table 1. Test Metrics of YOLO Models

Model	Precision	Recall	mAP50	mAP50-95
YOLOv5	0.999	1.000	0.995	0.974
YOLOv8	0.999	1.000	0.995	0.977
YOLOv10	0.992	0.989	0.994	0.986
YOLOv11	0.999	1.000	0.995	0.976

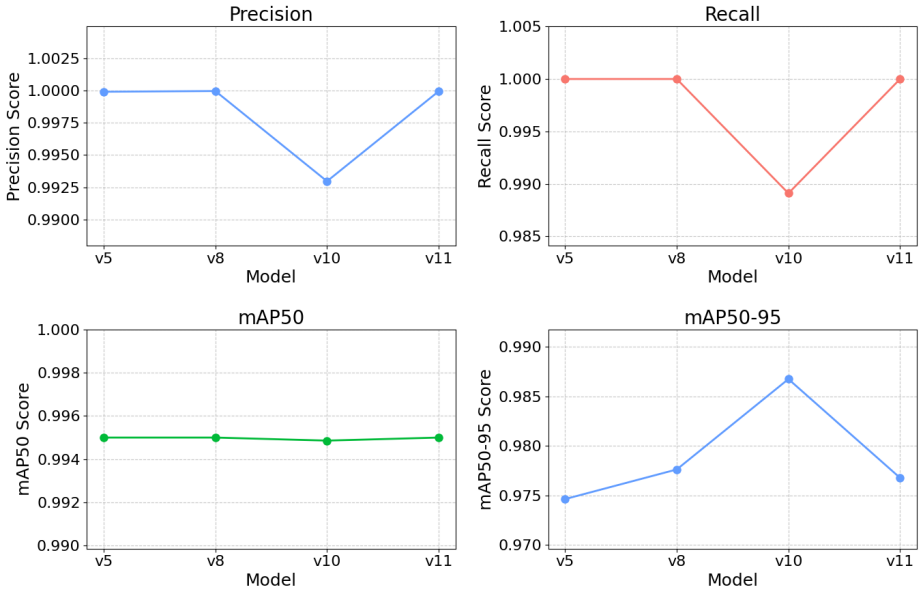

Fig. 8. Test metrics comparison.

The results indicate that YOLOv8 consistently performs well in all the metrics, YOLOv10 appears to perform the worst as compared to other models. Another reason YOLOv8 would be the best pick is the ease in testing as we can extend it to also use video files for training removing the need for frame extraction completely.

4.3 PSAT Metrics

The PSAT algorithm was evaluated using the metric of accuracy, using a labeled ground class of which slots are occupied and which ones are free. The PSAT algorithm's accuracy was calculated to be on average of 87.9%. The accuracy of the algorithm varied over the count of cars, when the parking lot was full the algorithm could achieve an highest accuracy

of 80.8%. The performance of the algorithm with respect to the number of cars is shown in Fig. 9

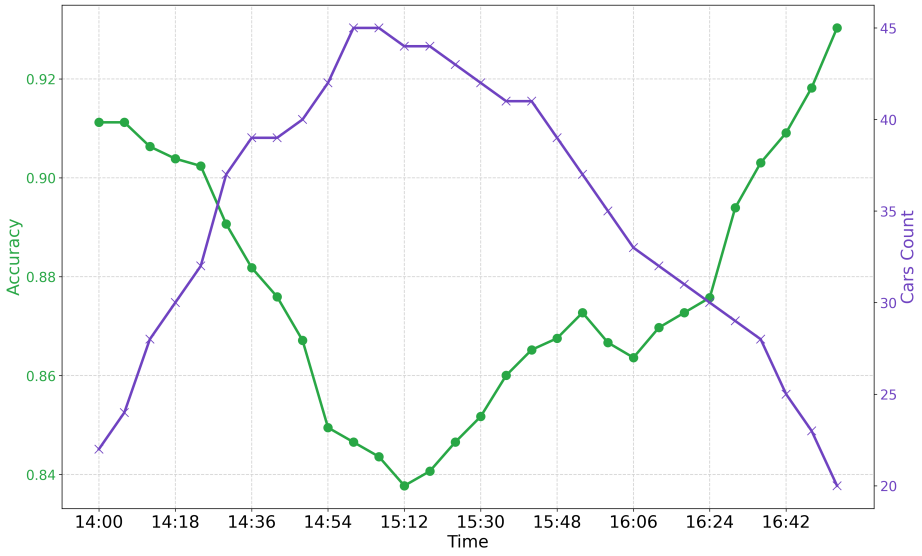

Fig. 9. Accuracy vs Cars count over time.

5 Conclusion and Future Scope

This research presents an innovative solution for parking lot monitoring using YOLOv8 combined with the PSAT algorithm. The PSAT algorithm effectively addresses real-world challenges such as occlusion and overlapping vehicles, which often lead to inaccurate detection, especially in scenarios where cars park at angles or in non-marked parking spaces. By leveraging YOLOv8's object detection capabilities and applying PSAT, our system provides accurate identification and alignment of parking slots, even in difficult situations where traditional methods might struggle. This approach not only improves the detection of parked cars but also ensures a more precise mapping of available and occupied parking spaces, significantly enhancing parking management in urban environments.

Future work could incorporate advanced data analytics and predictive analysis to optimize parking space utilization and improve system efficiency. Additionally, enhancing the PSAT algorithm's accuracy could be

achieved by adjusting camera placement to reduce the angle between the camera and parked cars. Ensuring that all vehicles are captured at a minimal angle would significantly improve the algorithm's precision, further enhancing detection and mapping of parking slot occupancy.

References

1. J. Vera-Gómez, A. Quesada-Arencibia, C. García, R. Suárez Moreno, and F. Guerra Hernández, 'An Intelligent Parking Management System for Urban Areas', Sensors, vol. 16, no. 6. MDPI AG, p. 931, Jun. 21, 2016. doi: 10.3390/s16060931.
2. S. Hanoun et al., 'A framework for designing active Pan-Tilt-Zoom (PTZ) camera networks for surveillance applications', 2017 Annual IEEE International Systems Conference (SysCon). IEEE, Apr. 2017. doi: 10.1109/syscon.2017.7934744.
3. H. Lou et al., 'DC-YOLOv8: Small-Size Object Detection Algorithm Based on Camera Sensor', Electronics, vol. 12, no. 10. MDPI AG, p. 2323, May 21, 2023. doi: 10.3390/electronics12102323.
4. V. V. Kukartsev, R. A. Ageev, A. S. Borodulin, A. P. Gantimurov, and I. I. Kleshko, 'Deep Learning for Object Detection in Images Development and Evaluation of the YOLOv8 Model Using Ultralytics and Roboflow Libraries', Lecture Notes in Networks and Systems. Springer Nature Switzerland, pp. 629–637, 2024. doi: 10.1007/978-3-031-70285-3_48.
5. S. A. Sanchez, H. J. Romero, and A. D. Morales, 'A review: Comparison of performance metrics of pretrained models for object detection using the TensorFlow framework', IOP Conference Series: Materials Science and Engineering, vol. 844. IOP Publishing, p. 012024, Jun. 30, 2020. doi: 10.1088/1757-899x/844/1/012024.
6. N. Gonthina, S. Katkam, R. A. Pola, R. T. Pusuluri, and L. V. N. Prasad, 'Parking Slot Detection Using Yolov8', 2023 3rd International Conference on Mobile Networks and Wireless Communications (ICMNWC). IEEE, pp. 1–7, Dec. 04, 2023. doi: 10.1109/icmnwc60182.2023.10435799.
7. F. N. Kılıçkaya, M. Taşyürek, and C. Öztürk, 'Performance evaluation of YOLOv5 and YOLOv8 models in car detection', Imaging and Radiation Research, vol. 6, no. 2. EnPress Publisher, p. 5757, Jul. 01, 2024. doi: 10.24294/irr.v6i2.5757.
8. Amira. A. Elsonbaty and M. Shams, 'The Smart Parking Management System', arXiv, 2020, doi: 10.48550/ARXIV.2009.13443.
9. Y. Amit, P. Felzenszwalb, and R. Girshick, 'Object Detection', Computer Vision. Springer International Publishing, pp. 875–883, 2021. doi: 10.1007/978-3-030-63416-2_660.
10. G. Babbar and R. Bajaj, 'Homography Theories Used for Image Mapping: A Review', 2022 10th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO). IEEE, pp. 1–5, Oct. 13, 2022. doi: 10.1109/icrito56286.2022.9964762.
11. N. Sharma, S. Baral, M. P. Paing, and R. Chawuthai, 'Parking Time Violation Tracking Using YOLOv8 and Tracking Algorithms', Sensors, vol. 23, no. 13. MDPI AG, p. 5843, Jun. 23, 2023. doi: 10.3390/s23135843.
12. B. X. Chen and J. K. Tsotsos, 'Fast Visual Object Tracking with Rotated Bounding Boxes', arXiv, 2019, doi: 10.48550/ARXIV.1907.03892.
13. Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, 'Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression', Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 07. Association for the Advancement of Artificial Intelligence (AAAI), pp. 12993–13000, Apr. 03, 2020. doi: 10.1609/aaai.v34i07.6999.