YOLO-Park: Adaptive Parking Space Detection Using Hybrid NMS and Otsu Thresholding

Abstract:
We present an automated vision-based parking monitoring system that dynamically detects occupied and vacant spaces without requiring predefined slot coordinates . Our computer vision-based parking monitoring system automatically detects occupied and available parking spaces using camera feeds. It works in 3 main steps:

- A YOLOv8 object detection model: Detects cars using AI (YOLO model)
- Dynamic size filtering (0.3-2.0× mean vehicle area) removes detection outliers: Filters results to remove mistakes and group close-together cars
- optimized Non-Maximum Suppression (IoU=0.1) resolves overlapping detections in crowded parking scenarios.

The system autonomously defines parking zones through spatial clustering of vehicle positions and identifies vacant spots using adaptive thresholding (Otsu's method) combined with contour-based size validation.

1. **Key Features:**
   - **YOLOv8-powered detection**: Accurate vehicle identification without specialized hardware
   - **Adaptive filtering**: Auto-adjusting size thresholds (0.3-2.0× mean area) for robust performance
   - **Density-optimized NMS**: Custom IoU=0.1 threshold handles tightly-packed vehicles
   - **Self-calibrating zones**: Automatic parking area detection via spatial clustering
   - **Hybrid vacancy analysis**: Combines Otsu's binarization with contour validation
   - **Practical outputs**: Visual overlays + CSV reports for easy integration

2. Objective of the Project

The objective of this project is to develop an automated, vision-based parking slot detection and occupancy classification system using deep learning and image processing techniques. The system aims to accurately detect vehicles in overhead parking lot images and identify empty and occupied slots with minimal manual intervention.

By leveraging a YOLOv8-based object detector trained on a custom dataset of 9,200 parking images, the method ensures reliable vehicle localization. Subsequent filtering, non-maximum suppression (NMS), and contour analysis are applied to refine the detections and assess parking slot availability effectively. This approach supports real-time monitoring, reduces human effort, and enables integration with smart parking solutions.

3. Introduction

With the increasing demand for intelligent transportation systems, smart parking management has become a crucial component of modern urban infrastructure. Traditional parking monitoring methods rely heavily on physical sensors or manual supervision, which are often costly, error-prone, and

difficult to scale. In contrast, computer vision offers a scalable and automated alternative for analyzing parking lot occupancy using surveillance footage or drone imagery.

In this project, we leverage YOLOv8, a state-of-the-art object detection model, to detect vehicles from top-down parking images. By combining deep learning with classical image processing techniques such as thresholding and contour analysis, the system not only detects parked vehicles but also estimates the number of empty and occupied slots accurately and efficiently. This fusion of AI and vision-based automation supports the development of smart, low-maintenance parking systems.

4. Methodology

The block diagram represents the sequential pipeline of the proposed vision-based parking slot detection and classification system. It comprises the following key stages

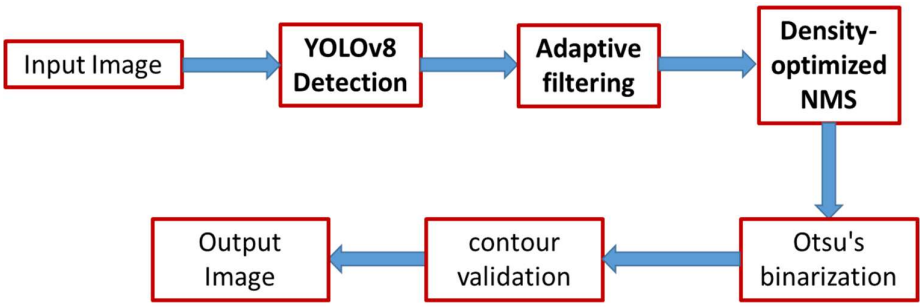

Figure 1: Block diagram of the proposed parking slot detection and classification pipeline. It illustrates the flow from YOLOv8-based car detection through size filtering, NMS, thresholding, and contour validation to final output generation.

4.1 Input Image:

The Fig 2 shows the original input image before any processing. The system begins by receiving a parking lot image captured from a top-down or elevated perspective using a surveillance camera or drone. This image serves as the primary input for the entire detection pipeline. The image is expected to:

- Contain multiple vehicles and visible parking slots.
- Be in a standard format such as JPEG or PNG.
- Have sufficient resolution for detecting objects like cars and slot boundaries.

Fig. 2. The original input image before any processing

4.2 Yolov8Detection:

The Fig. 3 illustrates the output of the YOLOv8 detection stage, where bounding boxes are drawn around detected vehicles. This step involves processing the input image using the YOLOv8 (You Only Look Once version 8) object detection model. The model has been trained on 9,200 annotated parking lot images, allowing it to accurately detect cars under varying lighting and density conditions.

The YOLOv8 model processes the entire image in a single pass, enabling fast and efficient real-time detection. It outputs bounding boxes for all detected vehicles along with associated confidence scores.

The detections produced in this stage form the foundation for subsequent filtering and analysis, ensuring that only relevant vehicle regions are passed through the pipeline.

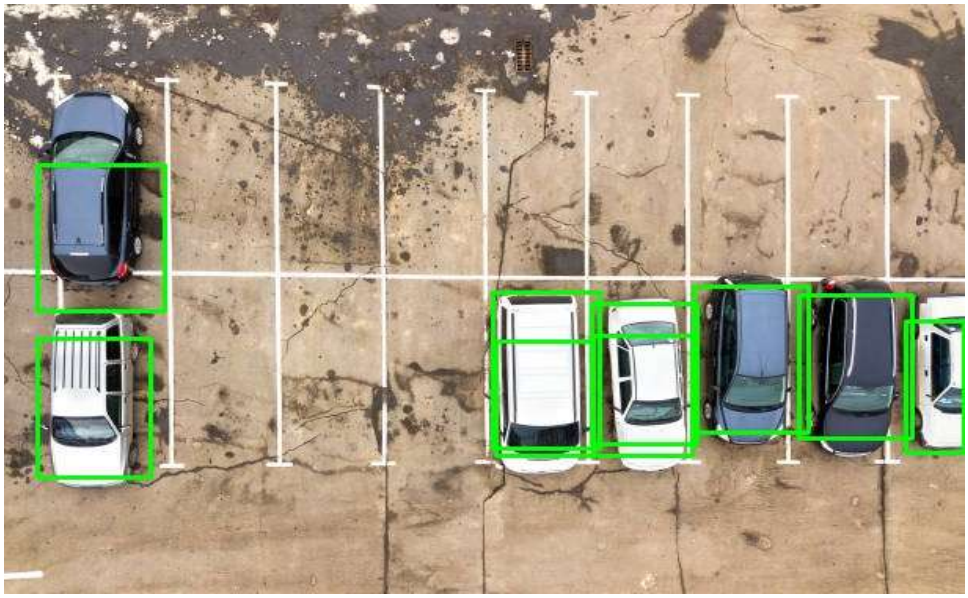

Fig. 3: Output of the YOLOv8 Detection stage showing bounding boxes around the detected vehicles in the parking lot. This stage serves as the first step in identifying occupied slots based on object-level localization.

4.3 Size Filtering

The Fig. 4 demonstrates the result after applying size-based filtering on the detected bounding boxes. While YOLOv8 provides initial car detections, some boxes may be too small (e.g., shadows, objects) or too large (e.g., overlapping detections or artifacts) to represent real vehicles accurately.

To address this, we calculate the area of each bounding box and compute a reference threshold based on the mean or median area of all valid detections. Bounding boxes that fall outside a defined range (e.g., less than 30% or more than 20% of the average area) are discarded.

This step enhances the robustness of detection by retaining only vehicle-sized regions and reduces the number of false positives passed into further stages.

Fig. 4: Output after size-based filtering. Irregular detections such as extremely small or large bounding boxes are removed to retain only valid vehicle-sized regions, ensuring more accurate occupancy analysis in subsequent steps. (In this example image, no such Bounding boxes that fall outside a defined range)

4.4 NMS (IoU = 0.1)

The Fig. 5 shows the result after applying Non-Maximum Suppression (NMS) with an Intersection over Union (IoU) threshold of 0.1. Despite the use of a robust detector like YOLOv8, overlapping bounding boxes around the same vehicle are common — especially in dense parking lots or near vehicles with complex shadows.

To address this, NMS is applied as a post-processing step. For each pair of overlapping bounding boxes:

- The IoU is calculated, which measures the ratio of the intersection area to the union area of the two boxes.
- If the IoU exceeds the chosen threshold (0.1 in this case), the box with the lower confidence score is suppressed.

A low IoU threshold like 0.1 ensures strict filtering, aggressively eliminating boxes that might only slightly overlap, which is especially useful in parking scenes where precision in vehicle boundary detection is critical. By removing redundant detections, this stage ensures that each vehicle is represented uniquely, which is essential for correctly estimating the number of occupied slots.

Fig. 5: Result of applying Non-Maximum Suppression (NMS) with a strict IoU threshold of 0.1. Redundant overlapping bounding boxes are removed, improving the precision of detected vehicle locations and preventing over-counting.

4.5 Otsu's Thresholding

The Fig. 6 presents the result of applying Otsu's thresholding on the grayscale version of the input image. After vehicle detections are finalized, a pixel-level analysis is performed to identify potential empty spaces in the parking lot, particularly inside the defined usable area.

The image is first converted to grayscale, and then a Gaussian blur is applied to reduce noise and smoothen the intensity variations. Subsequently, Otsu's thresholding, an adaptive binarization technique, is applied. This method automatically determines the optimal threshold value to separate foreground (cars or objects) from background (road or empty slots). The output is a binary image where:

- Foreground objects (including cars and shadows) appear in white.
- Background (such as pavement or unused space) appears in black.

This thresholded image is essential for identifying contours corresponding to unoccupied slots in the later stages.

Fig. 6: Binary image obtained using Otsu's thresholding. Foreground and background regions are segmented adaptively based on pixel intensity, enabling clear separation of potential empty spaces and vehicle regions.

4.6 Contour Validation

The Fig. 7 shows the output after applying contour detection and validation on the thresholded image. This stage is crucial for identifying potential empty parking slots, especially those not occupied by any vehicle but located within the usable area.

Contours are extracted from the binary image using OpenCV's findContours() function. Each contour is treated as a candidate region for an empty slot. However, to ensure accuracy and prevent false positives, each contour is validated using the following criteria:

- Area Filtering: Only contours with areas falling within ±50% of the average detected vehicle area are retained.
- Aspect Ratio Filtering: The bounding rectangle of each contour is calculated. Contours are accepted only if their aspect ratio (width/height) lies within a realistic range (typically between 1.0 and 3.5) based on actual parking slot dimensions.

Contours that pass both checks are considered as estimated empty slots within the usable area. This process helps identify vacant spaces even when no predefined slot layout is available.

Fig. 7: Result of contour validation showing estimated empty slots. Contours are filtered by area and aspect ratio to ensure only vehicle-sized vacant regions are identified within the usable parking area.

4.7 Output Generation

The Fig. 8 displays the final annotated output image where each detected and classified parking slot is visually marked. At this stage, all results from previous steps are consolidated to produce a clear, interpretable output for the end user or system integrator.

Key elements in the output include:

- Green bounding boxes around detected and validated vehicles labelled as occupied slots.
- Placeholder labels (E1, E2, ...) for estimated empty slots detected through contour validation.

- Additional labels (OE1, OE2, ...) for empty regions identified outside the main usable area, when applicable.
- A summary overlay indicating the total number of:
    - Occupied slots
    - Estimated empty slots (inside)
    - Estimated empty slots (outside)

Additionally, the system exports:

- A CSV file (slot_summary.csv) listing slot IDs, coordinates (if applicable), and status.
- A second CSV (contest_summary.csv) containing summarized slot counts and metadata (e.g., usable area ratio, outside region status).

This output supports further integration with parking management dashboards or IoT-based parking systems.

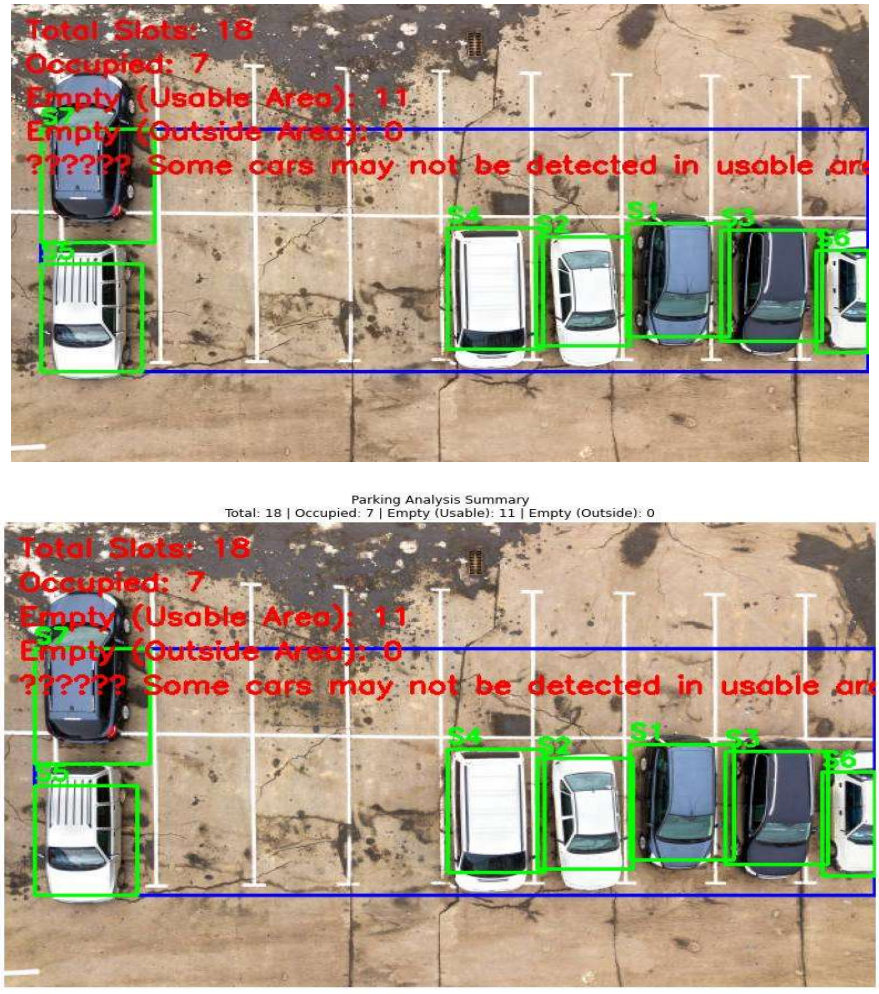

Fig. 8: Final annotated output showing occupied, empty, and outside-empty slots with labeled bounding boxes. The image also includes summary statistics for easy interpretation and system integration.

5. Conclusion

This work presents a complete pipeline for vision-based automatic parking slot detection and occupancy classification using a combination of deep learning and classical image processing techniques.

Starting with an input image captured from a top-down camera, the system uses a YOLOv8 model trained on 9,200 parking lot images to detect vehicles accurately. Subsequent steps — including size filtering, non-maximum suppression (NMS), Otsu's thresholding, and contour validation — refine the detections and allow for estimation of empty slots even in the absence of a predefined parking layout.

The final output provides:

- Visual annotations for occupied and estimated empty slots.
- Tabular data (CSV format) for integration with smart parking systems.
- Summary information such as total slot counts and usable area metrics.

This method is highly adaptable, cost-effective, and suitable for both real-time and offline analysis. It reduces manual effort and enhances the automation of parking management systems, supporting scalable deployment in urban and smart city environments.