

Parknest: A Lightweight Computer Vision Framework for Real-Time Parking Space Detection and Visualization

¹*Nitin Panwar, ²Shreya Priyal, ³Riya Raj, ⁴Kundan Prasad

Department of Computer Science and Engineering, Shivalik College of Engineering,
Dehradun.

⁵Ms. Vandana Bansla, Assistant Professor, Department of Computer Science and Engineering,
Shivalik College of Engineering, Dehradun.

Abstract: Managing parking availability in urban spaces is a constant struggle for drivers and facility managers alike. Traditional smart parking setups usually rely on expensive physical hardware, like ground sensors installed under every vehicle slot, which are prone to damage and costly to maintain. This paper presents a software-driven alternatives: a lightweight computer vision framework designed to monitor parking occupancy in real-time using standard surveillance feeds, the proposed system integrates an interactive spot-mapping module, an openCV-based image processing pipeline, and a flask-based API for data distribution. Instead of relying on computationally heavy machine learning models, the system applies adaptive thresholding and pixel-density calculation to accurately determine space availability. Experimental evaluation demonstrates that by evaluating non-zero pixel counts against a calibrated threshold, the system provides high accuracy with very low latency. The Proposed Framework Contributes toward cost-effective urban tech by repurposing existing CCTV infrastructure into intelligent parking management systems.

I. INTRODUCTION

The rapid expansion of urban areas has made finding an open parking spot a frustration and mostly ineffective process. Drivers often circle lots aimlessly, which increases localized traffic congestion and wastes fuel [1]. To fix this, automated parking surveillance has become a major focus for smart city initiatives. But right now, most commercial lots use ultrasonic, magnetic, or infrared sensors drilled directly into the pavement. While accurate, these physical sensors networks are heavily limited by their installation cost, wiring requirements, and maintenance overhead[2].

With the continuous improvements in Computer Vision, using standard optical cameras to monitor wide areas has become a much more realistic alternative. However, existing camera-based platforms tend to rely on oversized deep convolutional neural networks (like YOLO or SSD) to identify specific vehicles[3]. While these models are powerful, they require dedicated GPUs to run smoothly, which shuts out smaller facilities that just want to run the software on basic edge computers or standard desktop setups.

To address these limitations, this research proposes a simplified, highly efficient computer vision framework. Our system calculates occupancy by evaluating the structural layout of the parking spaces themselves rather than attempting complex object classification. By converting video frames into binary threshold maps and counting structural pixel density, the system acts as a virtually free alternative to hardware sensors, assisting facility operators and drivers with instant availability updates.

II. Literature review

Parking space detection has evolved from analog gating mechanisms to advanced digital monitoring. Early works in cameras-based tracking heavily utilized manual feature extraction, like Harris feature corners and histogram of oriented gradients (HOG), to try to figure out if cars were present[4]. However, these methods often struggled when the lighting changed throughout the day or when shadows moved across the asphalt.

More recent advancements have swung aggressively toward Deep Learning. Frameworks adopting Mask R-CNN or YOLO have effectively solved the object identification problem, allowing systems to easily recognize specific cars,

trucks and motorcycles in a frame [5]. However, the issue with heavy deep learning systems in a surveillance context is computational bloat. Running real-time inference on a high-definition CCTV feed requires expensive hardware accelerators.

Limited work has explored optimizing these systems down to their bare essentials for low-end hardware. The gap exists because finding a middle ground between “too simple to handle outdoor lighting” and “too complex to run cheaply” is mathematically difficult. The proposed framework addresses this by applying Adaptive Gaussian Thresholding and region-specific Pixel counting [6]. This provides the environmental resilience of an advanced system while keeping calculation times completely minimal.

III. Research Methodology

The application-based research methodology used here was centered around shifting the detection logic away from “finding a car” and towards “checking if a designated piece of ground is empty.” The framework is split into three primary mechanics: visual mapping, images transformation, and spatial evaluation.

First, to make the system flexible for any camera angle, a positioning script (`parkingSpacePicker.py`) allows operators to manually trace the exact coordinates of every parking slot over a static reference image. These coordinates are saved directly to a serialized local file (`CarParkPos`). This eliminates the need to hard-code locations, meaning the system can be deployed to new parking lot in less than five minutes.

During live monitoring, the main computer vision engine processes incoming video feeds frames-by-frames. To handle the variable lighting of an outdoor environment, the frame is converted to grayscale, softened with a Gaussian Blur to drop camera noise, and passed through an adaptive threshold filter. Standard thresholding would fail if half the lot was wrapped in shadows, but adaptive thresholding recalculates contrast values localized to smaller block regions, perfectly isolating the sharp edges of vehicles regardless of the weather. A median blur and a dilation matrix are then applied to thicken the visible structural lines.

Finally, the system clips the processed binary images into the specific rectangles drawn during the mapping phase. It counts the active (white) pixels inside each box. An empty patch of pavements registers very few active pixels. A complex geometric object like a car registers thousands by setting a hard threshold boundary limit (e.g., 855 pixels), the system evaluates the slot. If the count remains underneath the thresholds, the spot is painted green and marked as free. If it exceeds it, it is painted red.

IV. System Design

The architecture of our framework is built on modular Python scripts, separating the user interface, the visual processing loop, and the backend data handling. The system handles real-time data seamlessly on consumer-grade hardware.

The Configuration Layer handles the environment layout. Operators use a point-and-click interface to draw and delete slot boundaries. This ensures that the detection system has an accurate geometric map of the environment to evaluate, effectively telling the AI strictly where to look and ignoring irrelevant background movement like walking pedestrians.

The Vision Processing Engine acts as the core controller. Utilizing OpenCV and NumPy, it pulls the video stream, applies the adaptive thresholding pipeline, loops through the designated slot coordinates, and executes the pixel-count logic. This layer also handles the real-time HUD (Head-Up Display) via `cvzone`, actively drawing the green and red bounding boxes and printing the live count of available spaces directly onto the monitor for security personnel to observe.

The Data Distribution Layer is a crucial component for making the system actually useful to everyday drivers. Built using Flask, this lightweight web server reads the occupancy state generated by the vision engine and packages it into JSON format. By hosting a `/status` API endpoint, the system allows external mobile applications, digital street signs, or web dashboards to request the live parking data asynchronously without interfering with the video processing performance.

Integrating these layers creates a tight, highly functional ecosystem capable of replacing thousands of dollars in sensor hardware with a single camera and a basic computer.

V. Conclusion

This paper presented an efficient, software-based smart parking framework designed for real-time occupancy detection. The proposed system integrates manual coordinate mapping, an adaptive OpenCV thresholding pipeline, and a Flask-based API into a single operational architecture. Experimental testing shows that by prioritizing basic architectural pixel density over heavy neural network classification, the framework achieves high detection accuracy while maintaining extremely low latency. The study highlights how clever applications of fundamental computer vision techniques can solve complex urban infrastructure problems cheaply, allowing scalable smart-city deployment without the massive hardware costs.

References

- [1] R. S. Pressman, *Software Engineering: A Practitioner's Approach*, 2020.
- [2] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2021.
- [3] J. Redmon et al., "You Only Look Once: Unified, Real-Time Object Detection," *Proceedings of CVPR*, 779-788, 2016.
- [4] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 886-893, 2005.
- [5] G. Jocher et al., "Ultralytics YOLOv8 Documentation and Architecture," *Ultralytics Technical Documentation*, 2023.
- [6] OpenCV Development Team, "OpenCV Documentation," 2024. Available: <https://opencv.org/>
- [7] G. Amato, F. Carrara, F. Falchi, C. Gennaro, C. Meghini, and C. Vairo, "Deep learning for decentralized parking lot occupancy detection," *Expert Systems with Applications*, vol. 72, pp. 327-334, 2017.
- [8] T. Lin, H. Chu, and Z. Chen, "A vision-based parking lot management system," *IEEE International Conference on Systems, Man and Cybernetics*, pp. 2897-2902, 2006.
- [9] N. True, "Vacant parking space detection in static images," *University of California, San Diego*, 2007.
- [10] M. Grinberg, *Flask Web Development: Developing Web Applications with Python*, 2nd ed., O'Reilly Media, 2018.
- [11] G. Bradski, "The OpenCV Library," *Dr. Dobbs's Journal of Software Tools*, vol. 25, no. 11, pp. 120-123, 2000.
- [12] S. Suzuki, "Topological structural analysis of digitized binary images by border following," *Computer Vision, International Journal of Advanced Computer Science and Applications*, vol. 11, no. 4, 2020.
- [13] A. Al-Khatib, T. Al-Saeed, and S. Al-Mutairi, "Smart Parking System based on Internet of Things and Computer Vision," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 4, 2020.
- [14] V. Paidi, H. Fleyeh, J. Hakansson, and R. G. Nyberg, "Smart parking sensors, technologies and applications for open parking lots: A Review," *IET Intelligent Transport Systems*, vol. 12, no. 8, pp. 735-741, 2018.
- [15] I. Bazzani, N. Conci, and G. C. F. Fusi, "Towards an urban smart parking system," *IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pp. 635-640, 2012.
- [16] W. W. Ahmed, O. Ali, R. R. Alomari, "IoT-based Smart Parking System Implementation Conference on Electrical and Computing Technologies and Applications (ICECTA), 2019.
- [17] M. Abdi and Y. Jin, "Image-processing based robust parking space detection," *IEEE 15th International Conference on Mobility and Transport*, pp. 1-6, 2018.