If life is a highway, what is parking? A Rascal Flatts approach to parking spot detection

Abstract

On many streets, particularly in urban cities, on-street parking has long proved to be a hassle. Our proposed solution uses a two-step framework to identify spots and classify whether they are full or empty. Aerial images of streets on College Hill were collected using an Unmanned Aerial Vehicle. The dataset was then processed using a combined Convolutional Neural Network framework, which detected spots using a combined MaskRCNN-YOLO World pipeline. These detections were then temporally manipulated and geometrically manipulated using principal component analysis to align the parking spots. These spots were cropped and trained on various Convolutional Neural Networks, including MobileNet V2 Classifier and ResNet-18 to classify whether the spots were empty or full. Both MobileNetV2 and ResNet-18 demonstrated high accuracy with accuracies of 98% and 97%, respectively. ResNet18 demonstrated rapid convergence, whereas the MobileV2Net classifier showed high peak performance. Further research should explore the accuracy of this model on a variety of footage and should factor in adjustments for shadows and other noisy objects that may disrupt the frame.

1 Introduction

Looking for parking is a waste of time, fuel, and emotional energy. The average person takes an average of 7.8 minutes per drive when blind searching for parking. These consequences quickly compound and have significant impacts on traffic congestion and pollution, particularly in urban areas. Previous research has explored the capabilities of a spot management system in contained and visually-defined ecosystems, such as parking lots, but there is limited exploration in the field of unbounded parking spots. Existing solutions currently use a combination of sensor-based devices, manual detection, and real-time traffic video analysis. This paper proposes a two-stage approach to classifying whether a parking spot is full or empty, separating the task into two sub-tasks: parking spot detection and parking spot classification. The ultimate goal of this project is to act as a service to drivers on a mobile platform to minimize the amount of time drivers spend looking for parking. Ultimately, this project aims to offer a comprehensive solution to a common problem, saving drivers fuel, frustration, and most importantly, time.

2 Related Work

There is very limited research that focuses on unbounded, on-street parking. Most approaches to parking spot detection typically explore techniques to detect parking spots in parking lots, which have clear lines that allow for a more simple image classification technique. These approaches are often used in conjunction with sensor-based devices to develop a deployable parking management system.

Most parking lot detection approaches used the PKLot Dataset — a dataset containing 12,416 images of parking lots extracted from surveillance camera frames — to train their models. These models are then used in real-time to detect whether or not a parking spot was full or empty. Some also used UAVs to determine the number of cars in the parking lot, while others used cameras from the driver's seat to determine an available parking spot. Very few ap-

proaches collected their own data.

Aajal et. al conducted a review of current on-street parking slot detection. The study found that there were three main types of image scan types — around view, bird's-eye view, and driver's point of view — four main datasets — the PS2.0 dataset, the PPSD Dataset, the Semantic Segmentation Dataset, and the SNU Dataset — that were run on various modifications to existing deep learning moodels.

Several previous studies used CNNs to detect certain features within an image. Typically, the CNNs used a form of YOLO for object and line detection. Directional Marking-Point Regression for Parking Slots (DMPR-PS), for example, used a CNN to generate feature maps which were used to regress the corners of each of the parking slots. The Vacant Parking Slot Detection Network (VPS-Net) uses YOLOv3 to detect the marking points and the slothead, and infers the type of parking slot by analyzing the geometric distance between the paired marking points. Geometric rules are further applied to infer hidden corners. The second stage independently classifies the occupancy status of each detected slot. Deep Parking-Slot Detection (DeepPS) uses a pre-trained YOLOv2-based object detector to identify all marking points within the input image, then a custom DCNN analyzes pairs of detected marking points to determine whether they form a valid entrance line. DeepPS used data augmentation techniques to address potential issues with rotational invariance. The technique also used a template-matching approach to handle slanted parking slot detection. Another study explored an end-to-end parking slot detection method that operates on Around View Monitor (AVM) images to extract global and local features via a grid-based CNN structure. The aforementioned review also discussed a study that used a specialized two-stage parking slot detection framework, demonstrating their efficiency of two-stage models. A region proposal network generated region proposals to determine the orientation of the spot, then precise coordinates were learned to estimate location, which allowed for a more efficient classification.

When working with images from a bird's-eye view camera as input, a fine-tuned YOLOv3 model was often used to estimate the width, height, and center coordinates of parking slots. After a spot is detected, one method used a temporal inference mechanism across multiple frames to determine whether the spot is full or empty. Other approaches using bird's-eye view camera used morphological filtering, grayscale conversion, and edge enhancement using the Canny operator to preprocess the images and eliminate any unnecessary noise. Sequential RANSAC algorithms were also used to detect linear structures and generate candidate parking slots, which were further validated using a Naive Bayes classifier.

Of the models that dealt with unbounded parking spaces, the pipelines tended to be one-stage classifiers. Goren et. al used a CNN, but did not find substantial results.

Mask RCNNs were used with some frequency, and Coleiro et. al found great success in using the model to develop a lost-cost solution to automatically annotate and detect parking spaces using a public camera feed. These spots were not labeled, but were found to be detected with relatively high frequency and accuracy. This was the ultimate inspiration for the combined MaskRCNN-YOLOWorld model.

Previous researchers also explored the utility of graph neural networks and transformer-based approaches to approach parking spot detection and classification. One study created a fully connected graph where each node corresponds to a detected-marking point. Then, a multi-head attention GNN aggregates features from adjacent nodes to capture the geometric and contextual dependencies.

3 Method

We used 2 hours of raw video footage across 4 streets and 11 camera angles, trimmed to moments when cars moved in or out of spots. Since most footage lacked painted spot boundaries, we treated spot detection and occupancy classification as two separate tasks combined into one pipeline.

After testing YOLOv8, YOLOv11, YOLO-World, MaskRCNN, MaskRCNNYOLO-World, and MaskRCNN-YOLOv11, the MaskRCNN-YOLOWorld Pipeline was determined to be the most comprehensive at initial object detection. (See Figure

2).

After each of the trimmed video clips was run through a combined MaskRCNN-YOLOWorld pipeline (See Figure 3), there were still many spots that were not detected because a car never filled the space or the model did not detect the car. For each boundary box returned by the combined pipeline — in which MaskRCNN scans were given priority — any boxes significantly above or below the median height and width were removed from the detections. Then, boxes were clustered together using Density-Based Spatial Clustering of Applications with Noise (DBSCAN) to group similarly located objects into the same location. Across each frame, the dwell, mean velocity, and number of observations were calculated to determined the "parkedness" of a car. The clusters were then split into rows using Principal Component Analysis to fit a line along the street, then K-means clustering on the perpendicular distance to fit two additional lines that would serve as the mathematical curb. The function also ensures that all cars are parked along their horizontal axes. If this model were to address working with parking at a slant, this particular addition should be reconsidered. Each parking spot along the curb line was then processed to fit the car to the line. Using RANDom Sample Consensus — an iterative algorithm to estimate model parameters with data that contains a large number of outliers — each of the spots were fit to the curb line that was previously found. After this, any significant outliers were removed, and RANSAC again determined the best spots that fit along the curb line. (See Figure 4)

At this point in the pipeline, we experimented with the use of open clip to verify that the two endpoints were in fact cars. This is because these end points served as the two bookends for which parking spots were inferred, so it was important that these points be accurate in their classification. Ultimately, the addition provided some utility, but it was not enough to outweigh its computational load.

For each spot, if the cosine angle between the car and the curb exceeded 5 degrees, then it was considered to be an outlier or an unparked car. Then, each of the spots were mapped on an XY plane, with the curb as an axis and each spot was sorted based on their Y-distance to the curb. If there was a gap that exceeded the (manually-defined) gap tolerance, then a spot would be inferred based on the median height and width of a car. If there was any overlap between two different types of spots, inferred spots or parking spots were given detection priority. (See Figure 5)

Of the models tested for object detection, the combined MaskRCNN-YoloWorld proved to be the most comprehensive object classifier (See Figure 6). The final spot coordinates were then used as inputs for the second stage classifier to determine whether a parking spot was full or empty. (See Figure 7)

In order to train our classifier, spots were defined manually by drawing bounding boxes, with coordinates applied across all frames for that angle to crop individual 224×224 patches. All 1,377 patches were manually labeled as occupied or empty, yielding 584 labeled patches (389 occupied, 195 empty) after filtering noisy or obstructed spots. To prevent data leakage, we split at the clip level rather than the patch level, resulting in a roughly 80/20 split of 395 training and 189 validation patches, augmented with random flips, rotations up to 30°, and color jitter.

The inputs were run on a fine-tuned ResNet18 classifier and MobileNetV2 classifier (See Figure 8 and 9). Both models were initialized with ImageNet weights and fine-tuned in two phases, first training only the classification head, then fine-tuning the full network with a lower learning rate.

An additional branch classifier was developed to detect whether or not footage had lines. The line detection provides for a problem that more closely aligns with previous research. The branch classifier is a MobileNetV2 CNN trained on 53 aerial frames (25 lined, 28 unlined) from our drone footage. Using the classical CV pipeline we detected spot boundaries automatically for all lined parking spot footage. This pipeline consisted of a Hough transform used on a white-pixel mask, which identifies long horizontal parking lane lines. We also applied a 50-pixel window on each side of each candidate line which counts the white vertical lines to decide which side is the parking lane. Dividers are then found by scanning for brightness peaks along the parking lane line. The points where vertical spot dividers intersect the horizontal parking lane line create local maxima in the column-wise brightness profile. These peaks are used

directly as divider positions, with any gaps (caused by parked cars covering painted lines) filled by interpolation using the median divider spacing. This produced a complete set of spot bounding boxes which could then be fed into our spot classification model.

4 Results

The training and validation accuracy curves for ResNet18 appeared to demonstrate rapid convergence (See Figure 10), whereas the training and validation for MobileNetV2 showed high performance (See Figure 11).

Method	Accuracy
ResNet Classifier	0.98
Mobile VNet2 Classifier	0.97

Table 1: ResNet18 proved to have higher overall absolute accuracy.

For the most part, models were able to detect where a spot was and whether or not it was full or empty (See Figure 12). Gradient camera heatmaps show that the model was accurately identifying the portions of parking spaces that were filled by cars (See Figure 13).

4.1 Technical Discussion

Throughout the development of this framework, we encountered a perpetual tradeoff between accuracy and computational efficiency. This was particularly highlighted when exploring the accuracy of the models for object detection. Previous computer vision techniques evaluate parking lots with spots that are often bounded by lines with little to no noise in the images. Given the simpler image classification task, these models worked well with lightweight CNNs. Using aerial footage for no-line spot detection requires significantly more data to allow for object detection models to not only pick up on the objects in the frame, but also analyze the temporal relationships between each detected object. The model's integration into live parking detection is also limited by its current computational load. The combined MaskRCNN-YOLO pipeline is run on GPUs in OSCAR.

The object detection for images with more shadows and noise still remains relatively poor. There are many edge cases in parallel parking that are not accounted for by this two-stage pipeline. For example, there are different shapes of curbs (i.e.: indented, slanted, etc.) that do not follow the above logic of fitting a line to a curb (See Figure 14).

The comparatively worse object detection could be improved with increased data or a more fine-tuned object detection model, especially one that deals with drone footage. This accuracy limited the classifier's ability to comprehensively full and empty spots (See Figure 15).

5 SRC Considerations

One of the largest considerations when using self-collected aerial footage is that of the privacy of the pedestrians and vehicles who were captured without consent. The total dataset contains around 70GB of footage. The trimmed clips are around 13GB. The footage does not track the movement of specific cars, but rather stays in one location for around 15 minutes. Although our study did not deal with license plate detection specifically, if our dataset was repurposed for further research, this would certainly introduce more critical privacy implications. Before sharing our footage with any other parties, then, the license plates and identities of the vehicles and persons, respectively, in the footage will be blurred.

In further iterations of this project, parking standards will also be included. That is, how long a driver can legally park or whether or not a spot is labeled as handicapped parking. The classification of these spots would be executed through the model, but the user's status would be denoted in the front-end.

The next steps for implementation involve experimenting with a static bird's-eye view camera over dense parking streets, such as Thayer St. When this footage is used, it is critical that it is kept for a short period of time to deliver real-time parking analytics while maintaining privacy. The data should not

be shared with any other party. This practice aims to optimally ensure the privacy of pedestrians and drivers.

Our current data collection method is explicitly a research prototype and not a sustainable infrastructure. We flew the drone manually for fixed sessions, limited by battery life and weather constraints, and were therefore able to obtain discrete clips used to train our models instead of a constant data stream. This bounds the privacy exposure of our data to a finite dataset, and means our current system does not require the location-aware app interface, continuous flight/camera operations, or persistent storage infrastructure that a real-time service would demand. Any transition toward real-time deployment would require a separate ethical review, for which we would take into consideration all concerns outlined above.

6 Conclusion

The two-stage pipeline provided a relatively successful framework for accurately classifying whether a parking spot was full or empty (See Figure 16). The combined MaskRCNN-YOLO-World pipeline was found to be the most comprehensive. Further geometric and temporal analysis can be conducted to explore the mathematical relationships associated with parking. The final CNN classification showed slightly different training and validation accuracy: ResNet18's learning plateaued while MobileNetV2's had a more consistent training curve.

There are various existing solutions (i.e.: ParkZen, weight sensors, parking lot spot classification) to determine parking availability, but very few — if any — use computer vision to give an accurate picture of what a region's parking looks like.

Similar to the length it can take to find parking, the history of the space itself is quite long. The vehicle industry in general has largely flourished, with a notable increase following the introduction of automobile design from 1900 to 1929. In accordance with this relative lack of popularity, traffic regulations were almost nonexistent in cities in the United States. The need for parking was first realized during WWI when the presence of vehicles in major cities grew too large to ignore — cars left unattended on the streets were a major problem for cities. Although the landscape of driving has changed, the problem of available parking remains.

Many urban cities may not have the physical capital to build more parking spaces, but this project should serve as a framework for deploying this detection to help drivers same time, money and frustration.

References

1. Azough, Ahmed, et al. "Autonomous On-Street Parking Slot Detection: A Comprehensive Systematic Review of Vision-Based Approaches." 2025, doi:10.1109/ACCESS.2025.3623566.

2. Chandrasekaran, Siddharth, et al. "Computer Vision Based Parking Optimization System." 2022, doi:10.48550/arXiv.2201.00095.

3. Dixit, Madhur, et al. "Smart Parking with Computer Vision and IoT Technology." *2020 43rd International Conference on Telecommunications and Signal Processing (TSP), IEEE, 2020, pp. 170–174.

4. Xu, Zhong, et al. "Research on Parking Space Detection and Prediction Model Based on CNN-LSTM." vol. 12, 2024, pp. 30085–30100, doi:10.1109/ACCESS.2024.3368521.

5. Kaur, R., et al. "A Hybrid Deep Learning CNN-ELM Approach for Parking Space Detection in Smart Cities." Neural Computing and Applications, vol. 35, no. 18, 2023, pp. 13665–13683, doi:10.1007/s00521-023-08426-y.

6. Nithya, R., et al. "A Smart Parking System: An IoT Based Computer Vision Approach for Free Parking Spot Detection Using Faster R-CNN with YOLOv3 Method." Wireless Personal Communications, vol. 125, no. 4, 2022, pp. 3205–3225, doi:10.1007/s11277-022-09705-y.

7. Gkolias, K., and E. I. Vlahogianni. "Convolutional Neural Networks for On-Street Parking Space Detection in Urban Networks."

IEEE Transactions on Intelligent Transportation Systems*, vol. 20, 2018, pp. 4318–4327, doi:10.1109/TITS.2018.2882439.

8. Gören, S., et al. "On-Street Parking Spot Detection for Smart Cities." 2019 IEEE International Smart Cities Conference (ISC2), IEEE, 2019, pp. 292–295, doi:10.1109/ISC246665.2019.9071760.

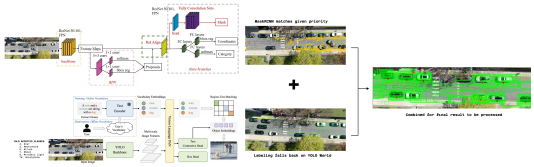

Figure 3: Combined MaskRCNN-YoloWorld Pipeline

Appendix

Figure 1: Ka-chow!

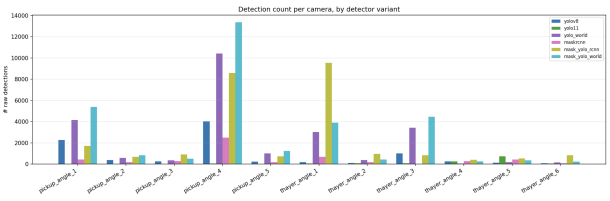

Figure 2: Overall, the MaskRCNN-YOLOWorld model performed best in initial object detection.

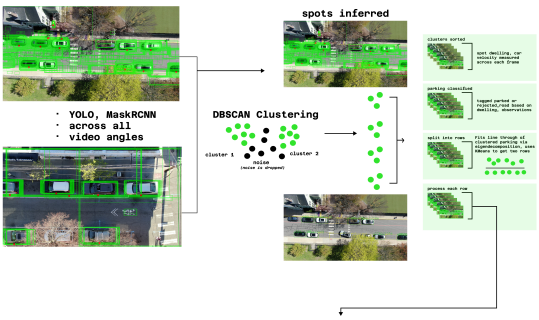

Figure 4: Boundary boxes are clustered based on dwell frames, number of observations.

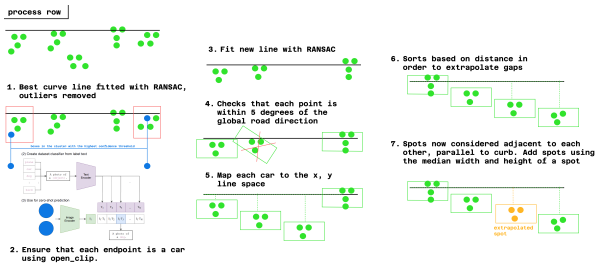

Figure 5: Each spot along the curb line is processed to see if it is an outlier. In any gaps, spots are then inferred.

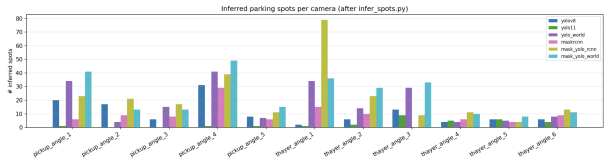

Figure 6: The combined MaskRCNN-YoloWorld and MaskRCNN-Yolov11 proved to most abundantly infer spots

Figure 7: The coordinates of the final spots were then used as inputs for the classification of full or empty.

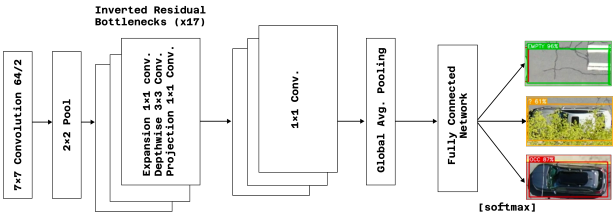

Figure 8: ResNet18 Classifier Architecture

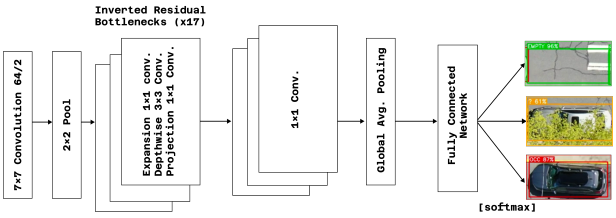

Figure 9: Mobile VNet2 Classifier Architecture

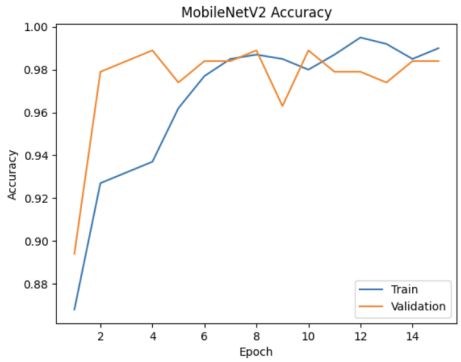

Figure 11: MobileNetV2 showed high average performance.

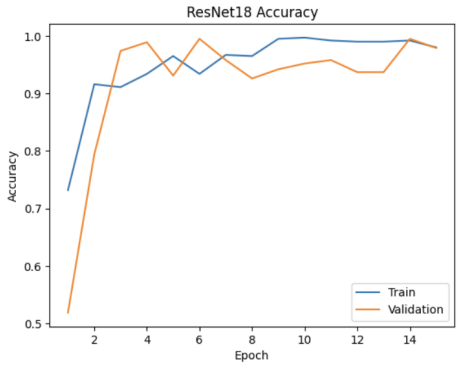

Figure 10: ResNet18 demonstrated rapid convergence.

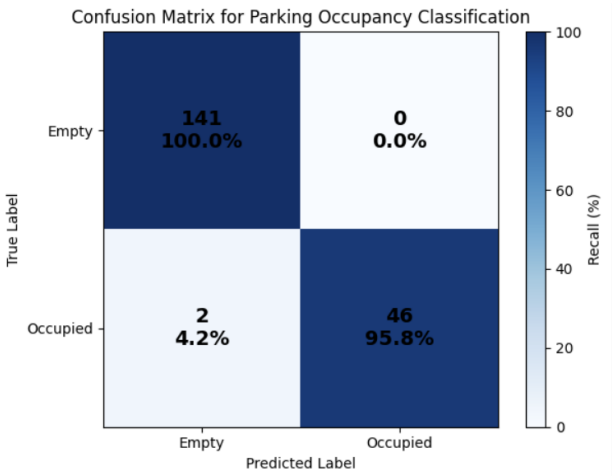

Figure 12: Confusion matrix illustrating model performance for parking occupancy classification.

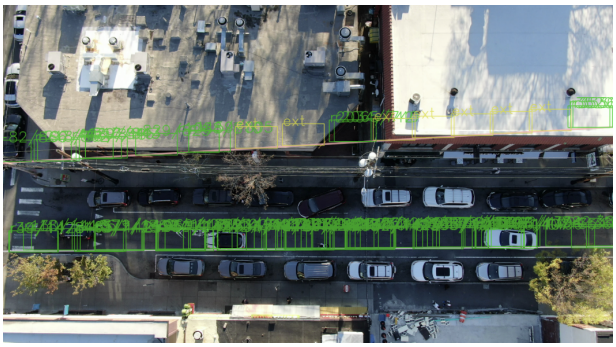

Figure 14: This indented curb shows the improper fitting of a curb line. We aimed to circumvent this issue by using car velocity to determined "parkedness."

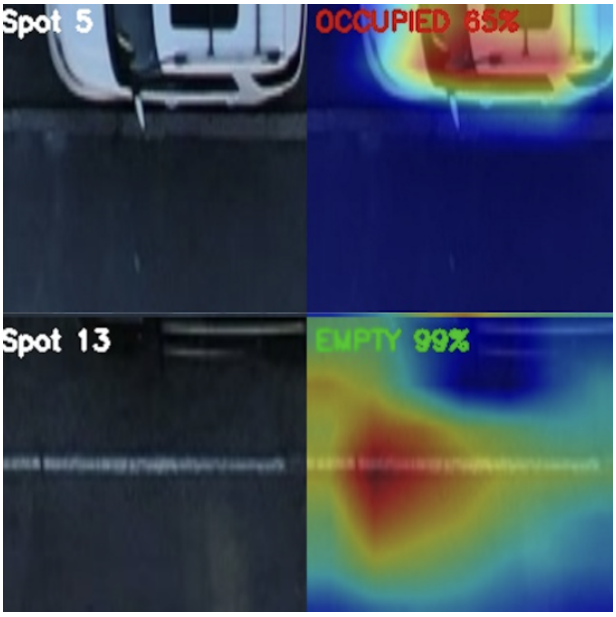

Figure 13: Grad-CAM heatmaps show model attention for accuracy predictions.

Figure 15: This image shows success line fitting on one side of the street, but no significant objects were detected on the other side. Further exploration should be done to manipulate the shadows of a street.

Figure 16: Final detection at the end of the CNN pipeline.