UrbanPark: Street Parking Detection through Computer Vision

Cynthia Chen , Abhinav Agarwal
Stanford University
cchen24@stanford.edu, abhinav4@stanford.edu

1 Introduction

In the field of object detection, one common method is parking space detection and classification. Upon doing research, we found several online datasets relating to parking space detection, several of which are on Kaggle. These datasets include images of original parking lots, boxes - which are the bounding box labels for the original images, and finally an annotations.xml file, which contain the coordinates of the bounding boxes and labels. Typically, work has been done in terms of training a CV model on these images and predicting the annotations and bounding boxes for the images. The previous research/data that has been done uses pytorch via a R-CNN model to predict bounding boxes for parking spots, and indicate if they are red or green depending on if there is an open or closed parking space. This method has been found to be very effective so far, in some applications. However, my partner and I found that there was a lack of research done on street parking classification and identification, so we have decided to make that the focus of our final project.

2 Background and Related Work

The first thing that we did was to find the datasets that could work for our street parking detection problem. After looking through a few datasets on Kaggle and other websites, we encountered one on Github that contained a large repository of 450+ images. (We used the starter code from this github repository to print the images and set up the baseline RCNN model). What we discovered upon looking at these images was that it contained a mix of both parking lot images and street images, which was good because we were initially looking to create street images in the first place, and after discovering these we just needed to change the angle of them so that they would look more similar to the parking lot datasets. As such, related work is mostly centered on tackling parking lot datasets but not street parking datasets. Additionally, one of our goals was to make the RCNN model more robust and see whether adding more complexity to the model would help it to garner better results, which is something that other works have not done in great depth. Finally, we have not seen past works perform perspective warping on this type of problem, so we aim to introduce this as a layer of novelty.
First, we created a Google Colaboratory notebook to load the training and validation datasets for the parking and street images. We were able to separate the data into a training dataset, validation dataset, and test dataset. Each image has three components: an image batch, rois batch, and labels batch. (We were able to set this up using the starter code that the Github repository provided, and load them into three variables per image.) The rois batch represented the red and green bounding boxes that would be overlaid on top of the cars in the image, and the labels represented whether those spaces/bounding boxes represented empty or occupied parking spots.

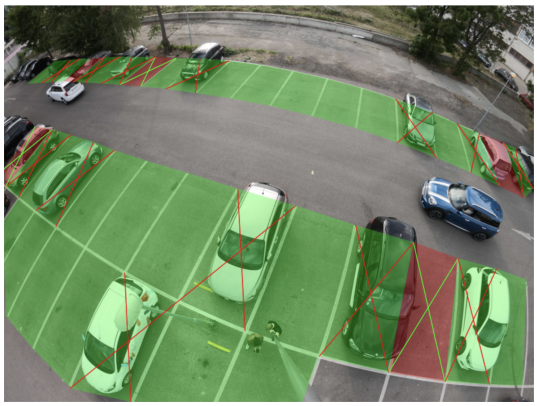

Figure 1: Demonstrative Example of Bounding Boxes

3 UrbanPark

3.1 Baseline RCNN Model

We used the RCNN model from the github repository as our baseline for evaluation. However, we wrote a function to compute the accuracy of the model on the testing dataset by taking the percent of bounding boxes that differed from the ground truth bounding boxes of the images.
Overall, this baseline RCNN model leverages a pre-trained ResNet-50 backbone for feature extraction, customizes ROI pooling, and allows fine-tuning of specific layers to adapt the model for this specific task. The ResNet-50 backbone is used as a feature extractor, and the model is initialized with pre-trained weights obtained from training on the large-scale dataset. The last fully connected layer (fc layer) of the ResNet-50 is modified to output 2 classes, for

Figure 2: K-Means for Four Different Features

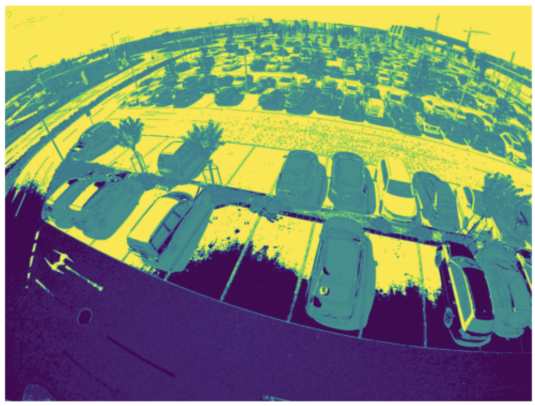

Figure 3: K-Means for Four Different Features

classifying whether a bounded parking spot was occupied or not occupied by a car in the image.
We trained this model on the training dataset (around 250 images from the original dataset) and then tested its performance on the test set (around 90 images). As mentioned earlier, we wrote a function to evaluate the error of the model on the test dataset by calculating the proportion of occupancy classes it predicted wrong, out of all of the examples. The accuracy we got for the baseline RCNN model was 0.737.

3.2 Updated RCNN Model

Overview

The updated RCNN model represents an effort to enhance the existing RCNN architecture by introducing an additional fully connected (FC) layer. This adjustment aims to capture more complex features from the region of interest (ROI) warps, with the intention of improving the classification accuracy of parking space occupancy.

Architecture

The enhanced model, `RCNN_UPDATED`, incorporates an additional FC layer after the backbone ResNet50's average pooling layer. The layer is configured with 1024 neurons, a ReLU activation function, and a dropout of 0.5 for regularization. The output of this layer then feeds into the final classification layer, which predicts the occupancy status. The backbone's final FC layer is also adapted to accommodate the reduced feature dimensionality output by the new FC layer.

Training Challenges

During the training process, we encountered significant computational challenges. The Jupyter notebook environment consistently crashed due to excessive RAM usage, likely caused by the increased complexity and size of the model. As a result, we could not complete the training with the newly initialized weights for the additional FC layer.

Testing with Random Weights

Due to the aforementioned constraints, we proceeded to test the model's initial behavior by instantiating random weights and biases for the additional FC layer. While these random weights do not carry meaningful learned representations, this step was necessary to verify the updated model's forward pass functionality. The average percent test error on the train dataset, noted as 0.0736977177718558, indicates the randomness of the initial weights and underlines the necessity of proper training. For future work, the `RCNN_UPDATED` model requires comprehensive training with adequately provisioned computational resources to prevent environment crashes and ensure the model's convergence. It would also be prudent to explore model optimization techniques to reduce RAM usage, possibly through model pruning or the implementation of a more efficient training pipeline.

4 Experiments

4.1 K-Means

We also wanted to do some exploration on different types of features that we could extract from the parking lot datasets. We were able to individually extract color, postion, gradient, and edge features and run k-means clustering on them to get assignments and segments, and apply them accordingly to the original image and visualize the results. [See Figures 2 and 3].

4.2 Perspective Warping

What We Tried to Do

We looked into using perspective warping to transform high-angle bird's eye view parking lot images into lower angle views, similar to what you'd see from a camera mounted on a vehicle. The idea was to create a test dataset that closely matched what our parking lot detection and analysis model would encounter in the real world.

Our Approaches

Automated point selection We tried to automatically identify key features and points in the bird's eye view images and map them to corresponding points in the low angle view. We used techniques like edge detection, corner detection, and line intersections to find parking space boundaries and other reference points that could help guide the transformation.

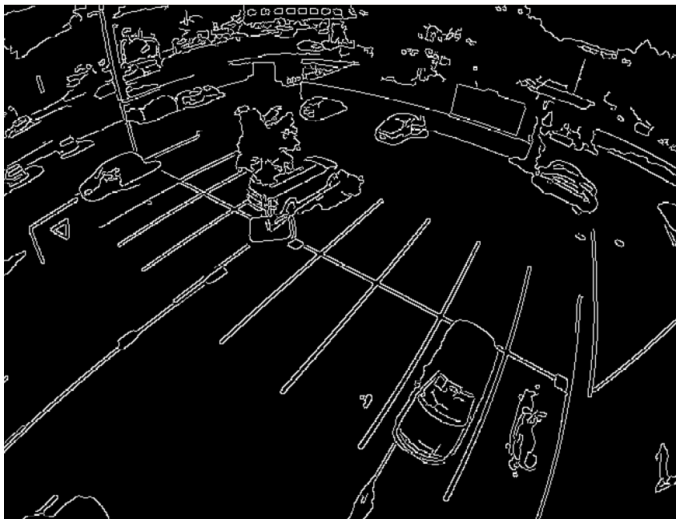

Figure 4: Edge Detection

Using GitHub dataset annotations We found a GitHub dataset [https://github.com/martin-marek/parking-space-occupancy/tree/main] that had annotated parking space coordinates in an annotations.csv file. We thought these could be used as source points for the perspective transformation.

Two-step transformation: affine and perspective Since the bird's eye view and low angle view are quite different, we tried breaking down the transformation into two steps. First, an affine transformation to adjust the overall orientation and proportions of the image, and then a perspective transformation to simulate the depth and foreshortening effects.

Camera model and inverse perspective mapping Inspired by MATLAB's birdsEyeView feature, we tried to model the perspective warp using camera parameters like focal length, principal point, camera height, and pitch. We then applied an inverse perspective mapping to project the bird's eye view onto a virtual camera plane.

Challenges We Faced

Inconsistent and inaccurate point correspondences Automatically detecting matching points between the bird's eye and low angle views turned out to be error-prone. The features looked quite different and were often occluded in the low angle view, making it hard to establish reliable point correspondences.

Limitations of annotation data Using the parking space coordinates from the GitHub dataset annotations as source points for the perspective warping didn't give us good results. The coordinates didn't directly correspond to the desired low angle perspective, so the transformed images ended up distorted and unrealistic.

Oversimplification of 3D to 2D transformation The two-step transformation approach (affine followed by perspective) wasn't enough to fully represent the complex 3D to 2D mapping. The affine step could only approximate the desired viewpoint, and the perspective warp didn't fully capture the nonlinear effects. This led to artifacts like stretched regions and weird depth discontinuities.

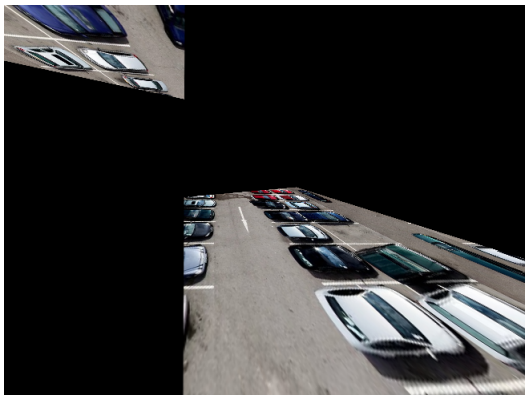

Figure 5: Distorted image when attempting warping

Figure 6: One of the images from the curated street dataset.

Mismatch between camera model and real-world variations Using fixed camera parameters to model the perspective warp had its drawbacks. In the real world, parking lots have uneven terrain, obstacles at different heights, and cameras mounted at different positions on vehicles. The simplifying assumptions of our camera model just didn't hold up well, resulting in unrealistic distortions.

4.3 Creating the street dataset out of the original dataset

Given the challenges we faced with perspective warping, we ultimately decided to take a different approach for creating our test dataset.

We first attempted to find some street images from the internet and creating a dataset that way, but we realized that the ROI information was essential input for calculating the scores during the forward pass.

Instead of trying to transform the bird's eye view images, we carefully selected a subset of twelve images from the full training parking lot dataset that most closely resembled the street parking scenario we were interested in testing.

To prepare this abridged test dataset for our model, we created a corresponding dataset of regions of interest (ROIs) for

each image. These ROIs highlight the specific areas in each image that we wanted our model to focus on during the testing phase [See Figure 6].

By using this approach, we were able to create a more realistic and reliable test dataset that closely matched our desired street parking scenario. This allowed us to effectively evaluate our model's performance without the need for error-prone perspective warping techniques.

While this test dataset was smaller in size compared to our original plan, it provided a more accurate representation of the real-world conditions our model would encounter. This strategy enabled us to proceed with testing and validating our parking lot detection and analysis model using data that we could trust.

4.4 Results and Discussion

We were able to run both of our RCNN and RCNN_Updated models on this smaller street dataset and get their accuracies as well. The average test error per image for the RCNN model was 0.0617 and, for the RCNN_Updated file, it was 0.3449.

5 Conclusion and Future Work

Throughout this paper, we have summarized our quantitative results from running the two models on both the original dataset and also the specifically curated street dataset of 12 images. Using the original dataset, we saw that the testing error had improved slightly with the updated RCNN model but was mostly the same between the two models. Interestingly, the RCNN model did not perform significantly better despite adding another layer to it. However, because we had to test with random weights, this could have negatively impacted the performance of the updated model, and in the future, we plan to improve upon this. Finally, we found that the original model still performed well on the street dataset that we created, with the RCNN updated model performing much worse, however. Despite this, we were able to target our RCNN model towards the street detection problem and identified one area of improvement, which was to create a dataset and model such that the model would not be dependent on ROIs data to make predictions.

Ultimately, we found that adding the additional complexity did not necessarily improve performance in certain scenarios. By performing perspective warping and k-means, we encountered a few challenges and visualizations that will be useful for future work.

In the future, we plan to create models that do not depend on ROI information for testing, as well as seeing how we can apply this work to AutoPark from Tesla. Recent developments include advancing AutoPark capabilities without reliance on an adjacent car requirement, and we hope to help enhance the solution to this problem with UrbanPark.

6 Contributions

Cynthia wrote the k-means visualizations of the images, created the street parking dataset and wrote the percent error evaluation function. Abhi wrote the RCNN_Updated model and the code and visualizations for perspective warping. Both group members set up the data loading and baseline RCNN model and contributed equally in writing the report.

7 Appendix

The code for our project is linked here: https://drive.google.com/drive/folders/1DU3zl0NQvP7K7cpRJr9VVMZB9nfqJYTV?usp=sharing

References

[Acharya *et al.*, 2018] Debaditya Acharya, Weilin Yan, and Kourosh Khoshelham. Real-time image-based parking occupancy detection using deep learning. 04 2018.

[Amato *et al.*, 2017] Giuseppe Amato, Fabio Carrara, Fabrizio Falchi, Claudio Gennaro, Carlo Meghini, and Claudio Vairo. Deep learning for decentralized parking lot occupancy detection. *Expert Systems with Applications*, 72:327–334, 2017.

[Anuj Khandelwal, 2021] Anuj Khandelwal. Bird's eye view transformation. https://gist.github.com/anujonthemove/7b35b7c1e05f01dd11d74d94784c1e58, 2021. Accessed: 2023-MM-DD.

[Marek, 2021] Martina Marek. Image-based parking space occupancy classification: Dataset and baseline. *ArXiv*, abs/2107.12207, 2021.

[Matsuda *et al.*, 2021] Akihiro Matsuda, Tomokazu Matsui, Yuki Matsuda, Hirohiko Suwa, and Keiichi Yasumoto. A system for real-time on-street parking detection and visualization on an edge device. In *2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, pages 227–232, 2021.