# 2021 SM1 GEOM90038 Advanced Imaging

## Lab Assignment 4: Parking occupancy detection from CCTV images

1087032 Chen-An Fan

## 1. Introduction:

### 1.1.Motivation

Smart city is the prime focus of the next generation urbanisation as metropolises grow rapidly in the 21st century. With the advancement of the Internet of Thing (IoT), data acquired by sensors around the city can be quickly gathered and processed to provide a comprehensive view of the city. The smart parking technique is one of the IoT applications using real-time data to detect the occupancy of parking slots. This can prevent the traffic-congestion and benefit the route planning as users can know the parking occupancy near their destinations in advance.

The infrared or magnetic sensors are the traditional way to directly measure the parking occupancy, however, they require additional effort to install. On the other hand, CCTV cameras are abundant and inexpensive nowadays. Real-time images can be easily acquired at a low cost by them. This facilitates the research in the field of computer vision that using image-based approaches to automatically detect parking occupancy.

In this assignment, I will implement an object detection model to automatically delineate the parking slots in the CCTV images of specific areas, and a classification model to automatically predict if a delineated parking slot is occupied or empty. These two models are implemented in MATLAB. Evaluation will be performed on the object detection model and then be used as metrics to post-process the bounding boxes to reach better parking slots delineation. The limitation of this method will also be discussed.

### 1.2.History of Machine Learning, Computer Vision, and Object Detection

Machine learning is the technique that makes machines (e.g. computers) automatically learn and improve through experience [1]. It is first purposed by Arthur Samuel in 1952. Arthur designed a set of mechanisms that allowed his program to learn playing checkers through reward function [2]. In 1957, Frank Rosenblatt first proposed the concept of perceptron which combined the human brain model with Arthur's machine learning concept [3]. The perceptron is the simplest unit of the modern neural network. One perceptron takes multiple inputs, weights those inputs, and passes the weighted value to an activation function to produce one output. An activation function acts like a threshold that takes the weighted inputs and decides whether to generate an output or not (generate 1 or 0). A perceptron is similar to a neuron in the brain. In the 1960s, researchers discovered the use of multilayer perceptron. This brought the concept of the neural network to the world [2]. However, due to the limited computational power at that time, the AI winter came. Research and funding of machine learning were decreased until the 1990s. In recent decades in which the computational power improved dramatically, the research on neural networks and related applications sprang up again. Those applications include computer vision, speech recognition, natural language processing, robot control, etc. [1]. In this assignment, I will focus the discussion on object detection, a sub-field of computer vision.

Computer vision is a research field that deals with high-level information extraction from digital images or videos [4]. And object detection, a sub-field of computer vision, is focused on detecting instances of objects of a certain class in digital images or videos [5]. There are multiple approaches to object detection based on convolutional neural networks, such as R-CNN [6], Fast R-CNN [7], Faster R-CNN [8], and YOLO [9]. In this assignment, I will implement Faster R-CNN as our object (cars) detection model.

### 1.3.Evaluation Matrices

In this section, I will discuss the evaluation matrices for object detection. Average Precision (AP) is defined as the area under the precision-recall curve. Where precision and recall are defined as:

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

The true positive (TP) is the instance that prediction and ground truth are both positive. The false positive (FP) is the instance that prediction is positive while the ground truth is negative. The false negative is the instance that prediction is negative while the ground truth is positive.

Therefore, precision means "among those predicted positive instances, how many of them have positive

ground truth". And the recall means "among all the instances with positive ground truth, how many of them have successfully been detected". In this assignment, I plotted the recall and precision of every parking slot to form the precision-recall curve and then calculated the AP from it. Intersection over Union (IoU) is used as the threshold to determine a prediction is positive or negative. IoU measures the predicted boundary and the ground truth boundary of the target object [10].

## 2. Methods

In this section, I will discuss the procedures and steps to complete each of the tasks.

### 2.1.Visualise Dataset

In this task, a training dataset contains images from a publicly available dataset (PKLot) and a test dataset contains images from Barry street data are provided. These datasets include images of occupied and empty parking spaces. Ground truth that includes the occupancy and the annotations of the parking slots delineations are also provided. The visualisation of the PKLot dataset is shown in Figure 1. Examples of empty and occupied parking slots are shown in Figure 2. The visualisation of Barry Street dataset is shown in Figure 3.

There are few concerns here. First, the image scale of the training set (PKLot) and the test set (Barry St.) are different. Second, in the training set, all the parking slots are oriented vertical, while in the test set, the parking slots are oriented both vertical and horizontal. These two concerns need to be further addressed to prevent challenges in the object detection model.

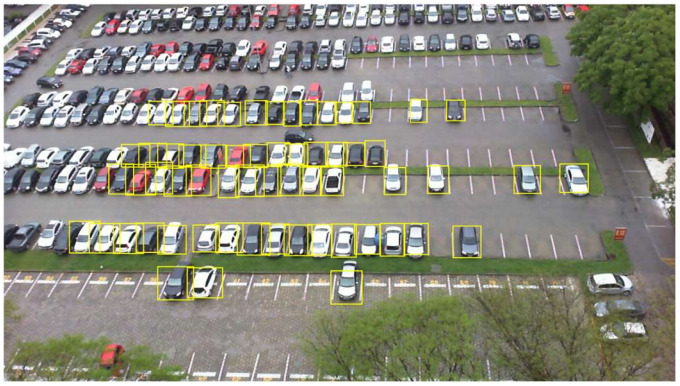

Figure 1. PKLot Parking Visualisation (the ground truth)


Figure 2. Segmented Images of PKLot

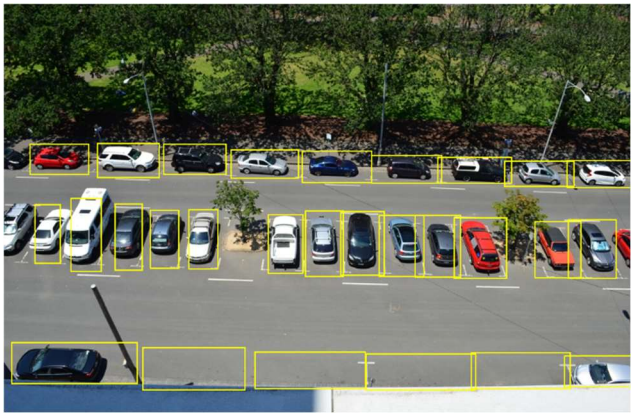

Figure 3. Barry Street Parking Visualisation (the ground truth)

**2.2.Create a Classifier (for classifying a parking slot is occupied or empty)**

*Input: cropped individual parking slot image*
*Output: 'Empty' or 'Occupied'*

In this task, a pre-trained Resnet50 CNN model is provided. This pre-trained Resnet50 CNN model contains 1000 classes, which is not compatible with our task. Therefore, we need to modify its classification and fully connected layers to two classes. Make it able to handle binary classification tasks. We will use 3000 segmented images with labels from PKLot (1500 empty slots, 1500 occupied slots, see Figure 2) to fine-tune the model. 70% of the segmented images will be used as training set, and 30% of the segmented images will be used as validation set. The classifier takes the segmented images (images of parking slots) as input (see Figure 2) and classified if the input is "Empty" or "Occupied".

During the training, the input images of park slots would be reflected, rotated, or resized to address the two concerns I mentioned in section 2.1. Due to the limitation of computational resources, I will directly use the provided fine-tuned model. The training curve is shown in Figure 4.

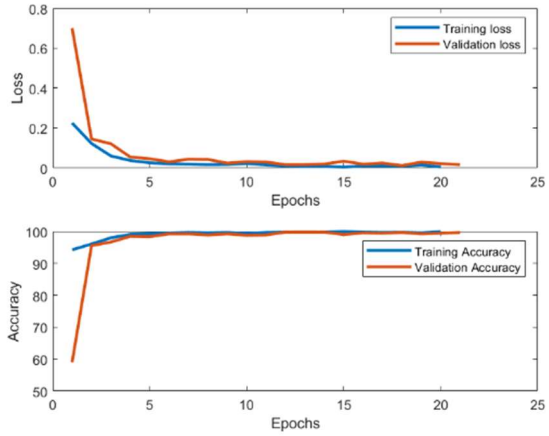


Figure 4. Training Curve

As we can see in Figure 4, the loss decreases, and accuracy increases within a few epochs. This means that the model fits the training data well.

After fine-tuning this model, the model is tested with the Barry Street dataset. The test accuracy is 99.21%. The confusion matrix of the test set is shown in Figure 5. And some of the wrong classified segmented images are shown in Figure 6. Then we stitch the segmented images (the individual parking slots) and their predicted result to visualise the whole Barry Street occupancy. The predicted occupied slots are coloured in red, and the predicted empty slots are coloured in green (see Figure 7).

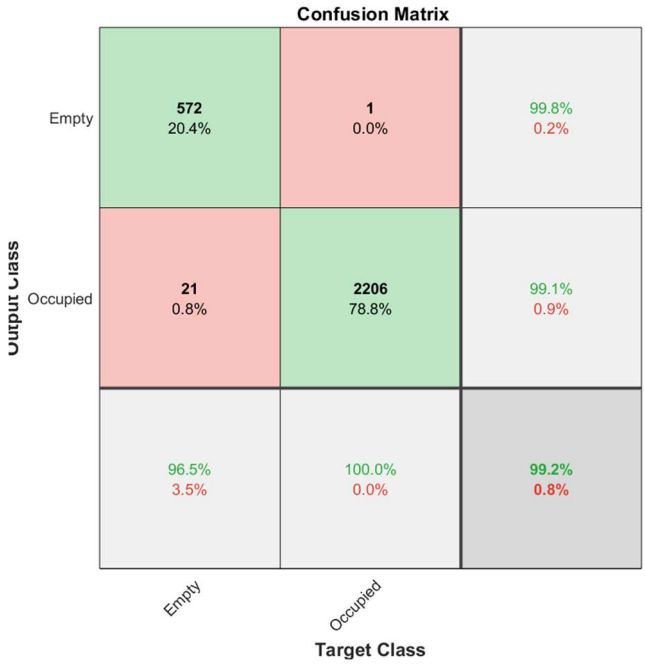


Figure 5. Confusion Matrix of the Test Set

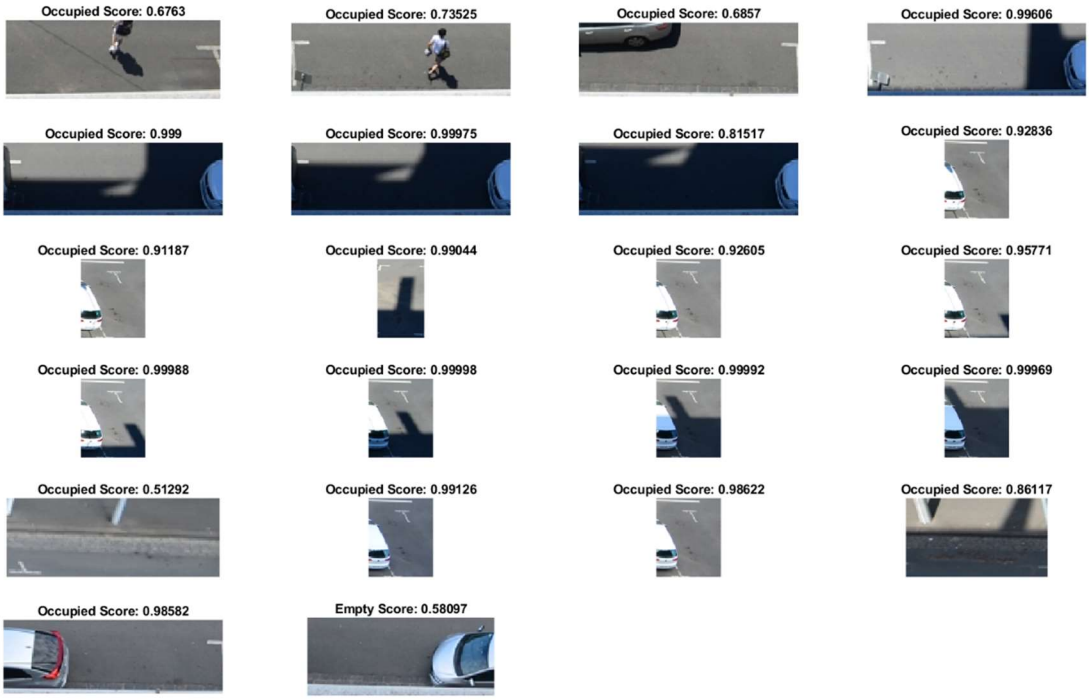


Figure 6. Wrong Classified Images of Test Set

As we can see the Figure 5, the overall accuracy is 99.2%, and the main error comes from classifying empty slots into occupied. The precision of the empty class is 99.8% and the recall is **96.5%**. The precision of the occupied class is 99.1% and the recall is almost 100%. In Figure 6Figure 5, there are total of 22 wrong classified images, 21 of them are actually empty but classified as occupied and 1 of them are actually occupied by classified as empty. This means that this model is less precise in classifying empty parking slots. We can observe from Figure 6 that if there is a part of a car in the wrong classified images, then it is likely to predict as "Occupied". Sometimes the shadows and humans in the images may affect the prediction as well.


Figure 7. Predicted Result of Barry Street

As Figure 7 shows, our model tends to make wrong classifications if there is a shadow in the parking slot. In this case, it would likely predict the empty parking slot as occupied. This is caused by the variance between training and test data. In the training dataset, there are no images with shadows. Therefore, this model performs badly when the input images contain tree shadows.

### 2.3.Automatic Parking Slots Delineation

*Input: the whole image of the parking area*
*Output: the delineation of the parking slots*

In section 2.2, the classifier takes the individual delineated parking slots as input and classifies them to empty or occupied. However, the images captured by CCTV are often the whole parking area instead of a single parking model. Hence, an object detection model that can automatically delineate the parking slots from images of the whole parking area is needed. The whole process of the parking occupancy detection will be:

a) Feed multiple images of the **whole parking area** taken from different time into an **object detection** model as inputs *(this Section)*

b) The objection model detects the **cars** in the images, then, outputs the bounding boxes of cars and their respective scores *(this Section)*

c) Cluster the **detected cars** into individual parking slots by density-based clustering algorithm [11] which uses spatio-temporal analysis *(this Section)*

d) Estimate the coordinates of the parking slots by weighing the coordinates of individual **detection** with the score of the detection in each cluster *(this Section)*

e) Improve the coordinates of the parking slots by statistics of the **detections** *(Section 2.5)*

f) Use the coordinates of the parking slots to crop the images of the whole parking area into multiple images of individual parking slot

g) Feed the images of individual parking slot into a **classifier** to **predict** if a parking slot is empty or occupied (Section 2.2)

h) Annotate the prediction results back to the original image of the whole parking area for visualisation

In this section, we will build a model to automatically draw the parking delineation on the image of the whole parking area (above step a-d). Here, a pre-trained object detection model (Faster-RCNN) is provided. This pre-trained detector is trained to detect cars on the highway by using images captured by camera mounted on a vehicle. Since our task is to detect cars in CCTV images, which is different from the highway environment especially from the angle of view. Therefore, we need to fine-tune this detector by using the images and the ground truth bounding box from the PKLot dataset. The training information about fine-tuning is shown in Figure 8. As we can observe from Figure 8, the training RMSE continues to improve. In this task, IoU is used to determine a detection is positive or negative. IoU between 0 to 0.3 will be treated as negative detection, and the IoU between 0.6 to 1 will be treated as positive detection.

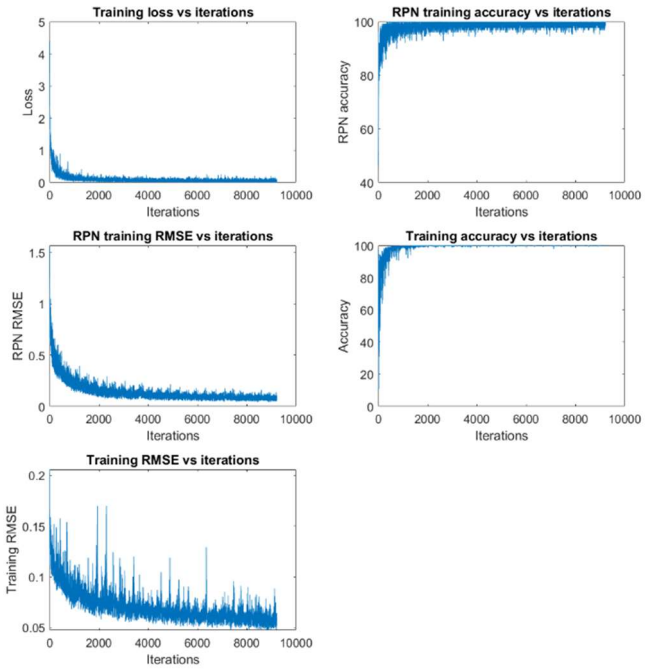


Figure 8. Training Info of the Fine-tuned Detector

Then, we use this fine-tuned detector to make predictions on one of the images from the test Barry street dataset. The result visualisation is shown in Figure 9. The input is the whole image of the parking area and the output is the bounding box of the cars.

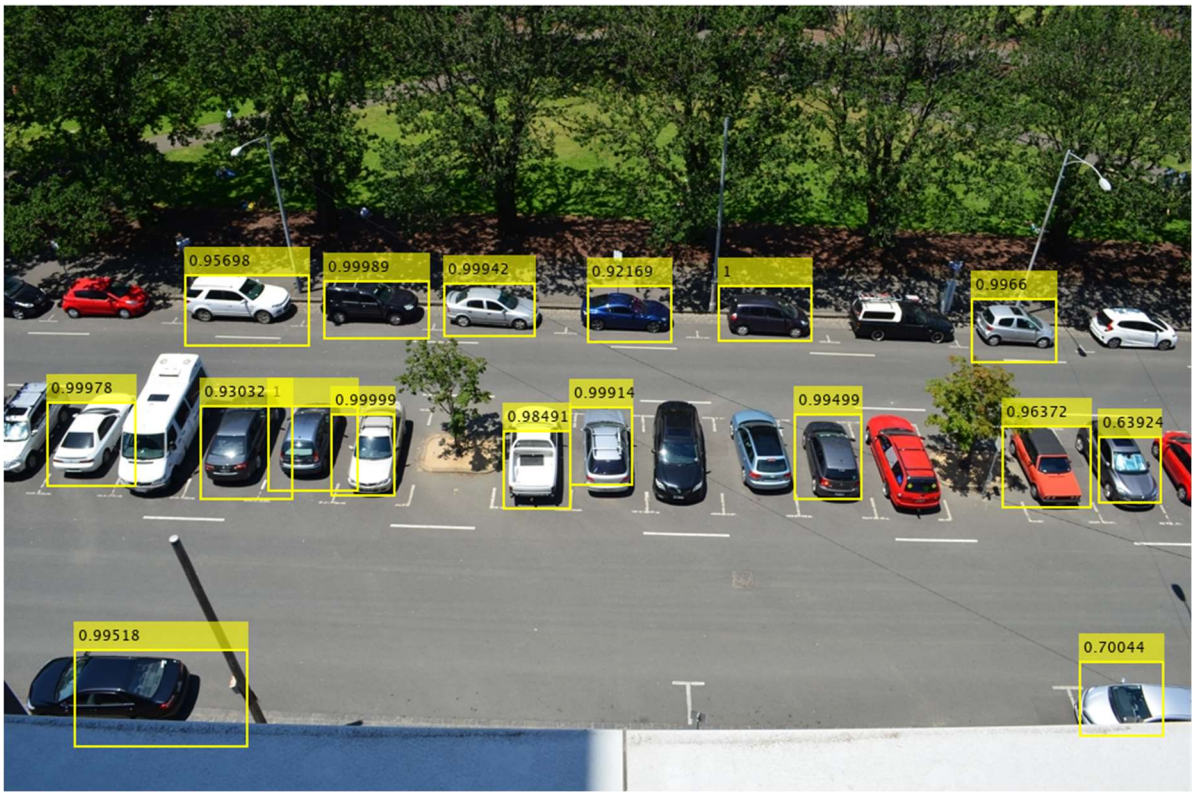


Figure 9. Detector Predicted Result on Test Set

As we can see in Figure 9, not all the cats are detected in this image. Hence, we run the detector to make predictions on 100 images from the test Barry Street dataset to get the average results. The result is shown in Figure 10 and Figure 11.

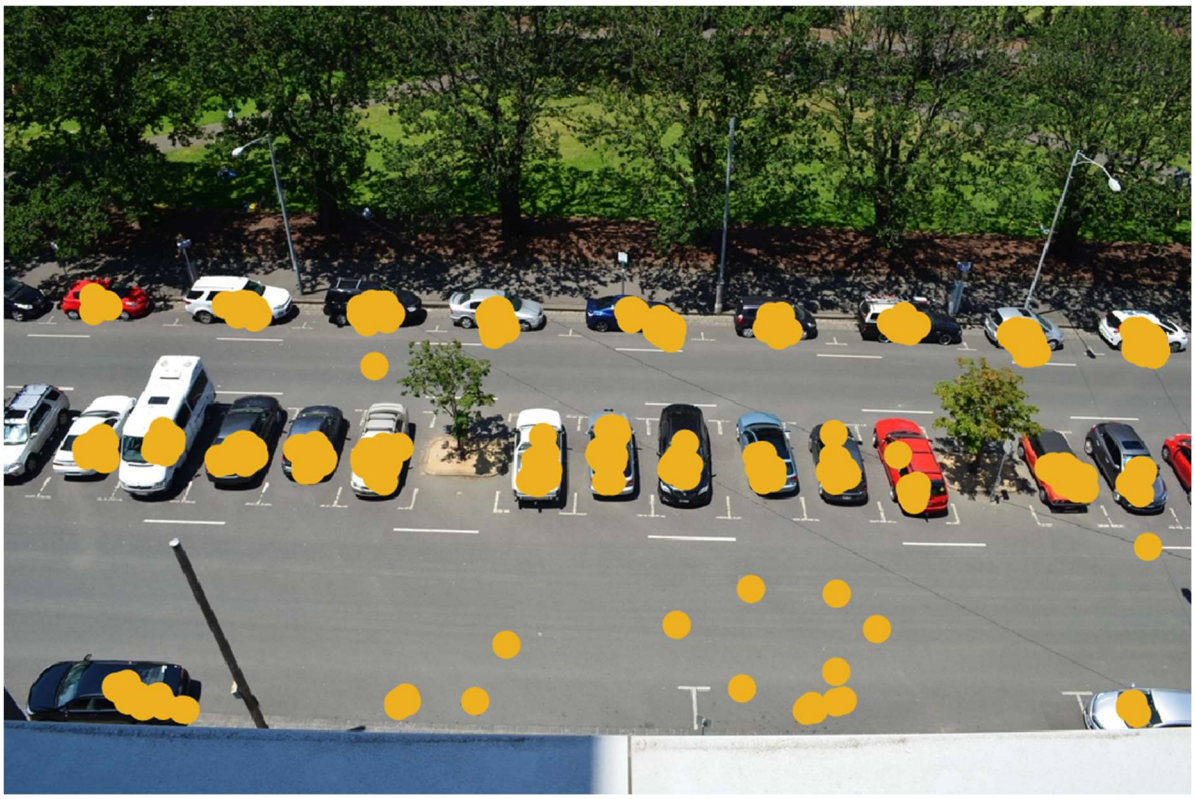

Figure 10. The Centre of Detections of 100 Barry Street Images with Faster-RCNN Fine-tuned with PKLot Dataset [12]

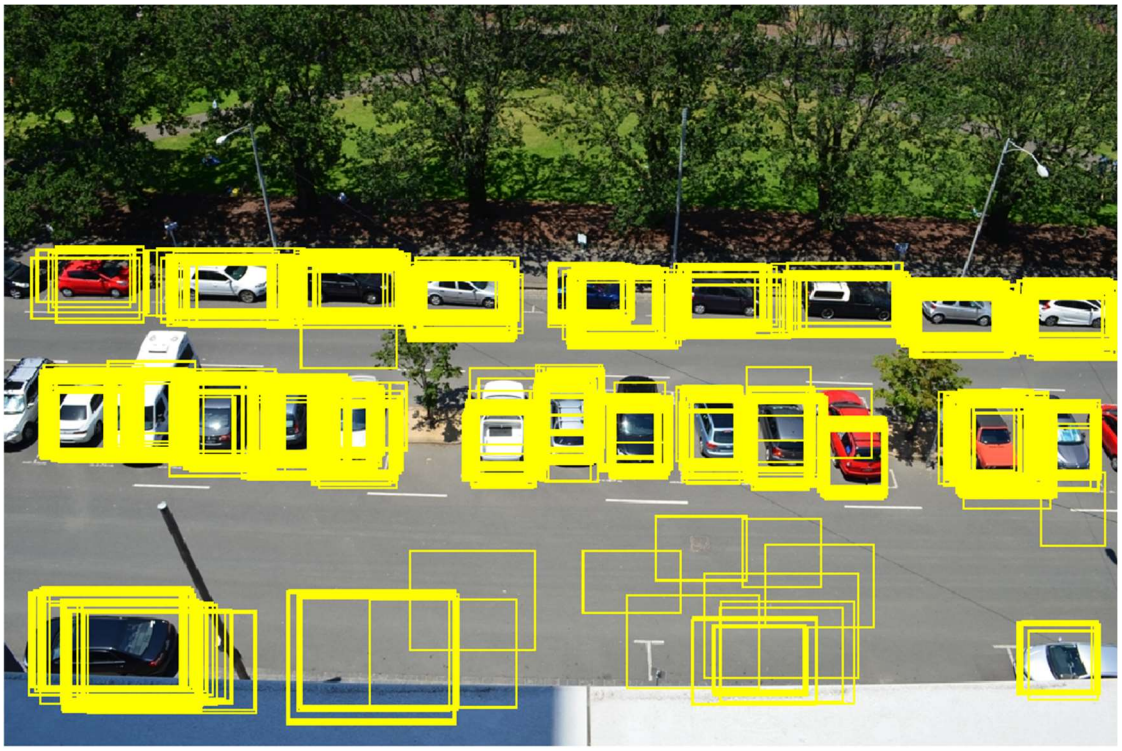

Figure 11. The Detections of 100 Barry Street Images with Faster-RCNN Fine-tuned with PKLot Dataset [12]

Then, the density-based clustering algorithm [11] is implemented on these detections. This algorithm estimates the number of clusters based on the distance between centres of the detections and the number of neighbours of a detection. We set the number of neighbours to 4 to filter out the detected cars on the road in few images; these sparse detections on the road will not form a cluster. We called it spatio-temporal analysis because those 100 images were captured at different times and the distance between centres is spatial. The output of this algorithm is the number of clusters which indicates the number of parking slots in this area. Here, 26 clusters are identified on the Barry Street image.

Then, we estimate the coordinates of each parking slot by weighing the coordinates of every detection with their detection scores in each cluster. In such a method, we can get the mean coordinates of the 26 parking slots. The bounding boxes of these 26 detected parking slots with their average scores are shown in Figure 12.

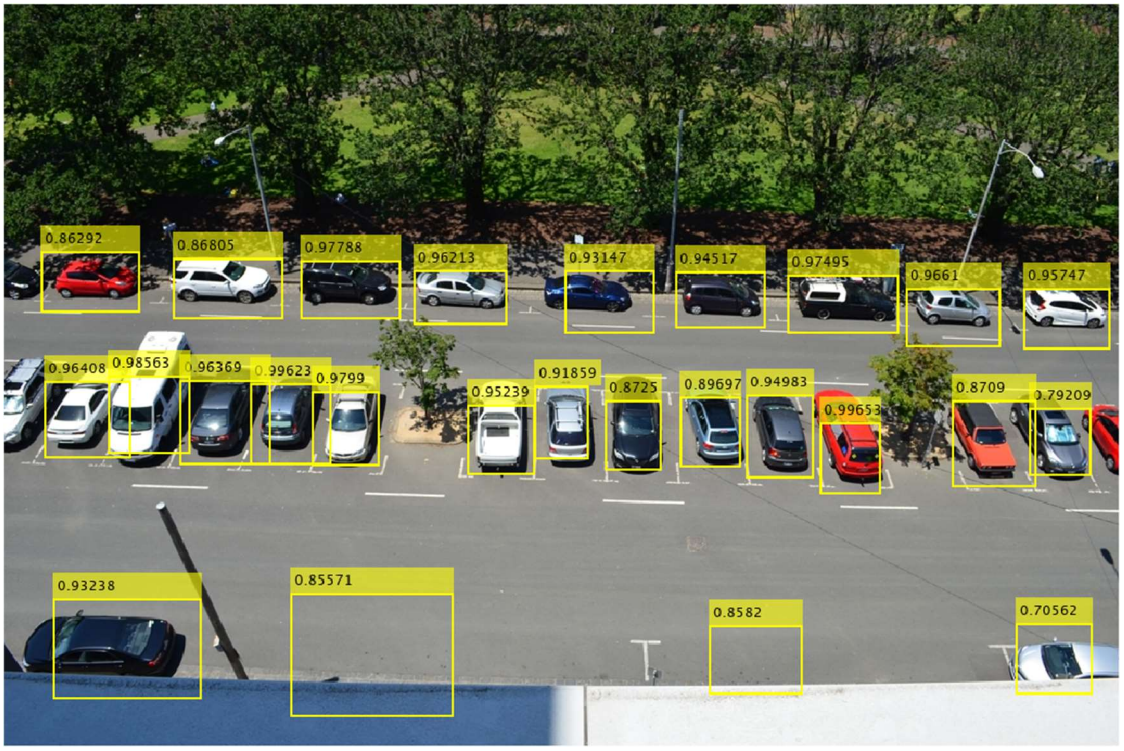


Figure 12. Bounding Boxes Derived from Each Cluster Representing Individual Parking Slots [12]

### 2.4.Perform Evaluation

In this case, I will evaluate the performance of the parking slot delineations from the previous section by comparing the bounding boxes in Figure 12 with the ground truth. My code is shown below:

```
% Calculate the average precision
load('GroundTruthBarryStreet.mat')
% Transform the resized prediction output back to original coordinates
classifiedMeanTransformed(:,1) = classifiedMean(:,1) - 141;
classifiedMeanTransformed(:,2) = classifiedMean(:,2) - 58;
classifiedMeanTransformed(:,3) = classifiedMean(:,3);
classifiedMeanTransformed(:,4) = classifiedMean(:,4);

predictedBox = classifiedMeanTransformed;
predictedScore = classifiedScoreMean;
predictionResults = table('Size',[1,2], ...
    'VariableTypes',{'cell','cell'}, ...
    'VariableNames',{'Boxes','Scores'});
predictionResults(1,'Boxes') = {predictedBox};
predictionResults(1,'Scores') = {predictedScore};
Vehicles{1} = ParkingSlots;
Groundtruth = table(Vehicles);
[ap, recall, precision] = evaluateDetectionPrecision(predictionResults, Groundtruth, 0.5);

figure;
plot(recall,precision);
grid on
title(sprintf('Average Precision = %.3f',ap));
xlabel('Recall');
ylabel('Precision');

fprintf('The mean precision of the bounding boxes is %.3f\n', mean(precision(2:end)));
%exclude the class "-1"
fprintf('The mean recall of the bounding box is %.3f\n', mean(recall(2:end))); %exclude the
class "-1"
```

The precision-recall curve is shown in Figure 12. The average precision with 50% IoU (AP50) is 37.1%. And the mean precision and recall of the 26 detected parking slots are 60.7% and 31.3%.

The precision is low due to two reasons. First, the size of detection is not uniform as we can see in Figure 12. This is counterintuitive with the fact that all the parking slots should have the same size. Second, the object detection we trained in section 2.3 is for car detection not for parking slot detection, therefore, the bounding boxes will be smaller than actual parking slots. Hence, post-process is needed to improve the precision.

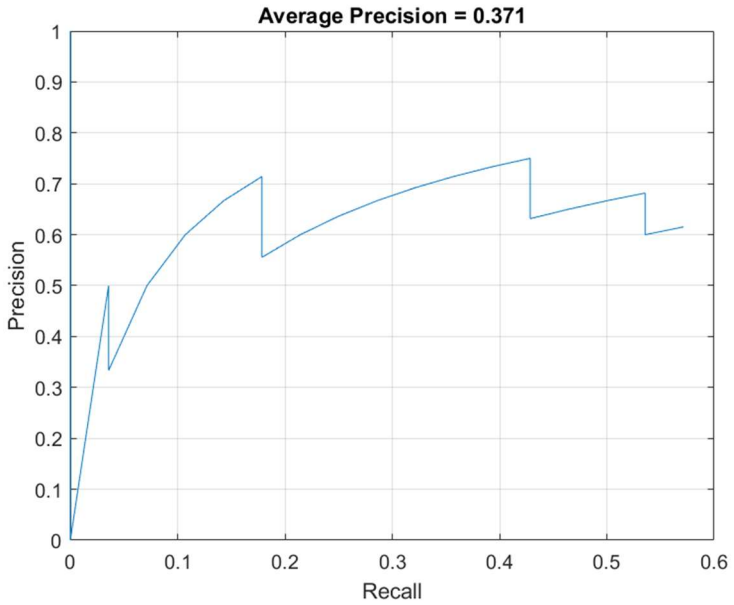


Figure 13. Precision-Recall Curve

### 2.5.Improve the Delineation Accuracy

#### 2.5.1. Plot the Statistics of Bounding Boxes

In this task, I will plot all the 26 bounding boxes for visualisation (see Figure 14). The bounding boxes are rotated to maintain an aspect ratio larger than 1. My code is shown below:

```
% Plot the statistics of the bounding boxes and calculate average length
% and width of parking slots
classifiedMeanTransformed_AspectRatio = classifiedMeanTransformed;

% As there are two main clusters with aspect rations >1 and <1 we rotate
% them to bring in the same direction
for i=1:length(classifiedMeanTransformed_AspectRatio)
    % if x direction is larger y direction then rotate
    if classifiedMeanTransformed_AspectRatio(i,3) > classifiedMeanTransformed_AspectRatio(i,4)
        classifiedMeanTransformed_AspectRatio(i,3) = classifiedMeanTransformed(i,4);
        classifiedMeanTransformed_AspectRatio(i,4) = classifiedMeanTransformed(i,3);
    end
end

% Parking slots of different aspect ration displayed together
figure;
hold on;
for i=1:length(classifiedMeanTransformed_AspectRatio)
    rectangle('Position', [0, 0, classifiedMeanTransformed_AspectRatio(i,3), ...
        classifiedMeanTransformed_AspectRatio(i,4)])
end
hold off;
xlabel('Width of parking slots in pixels')
ylabel('Length of parking slots in pixels')
axis equal
```

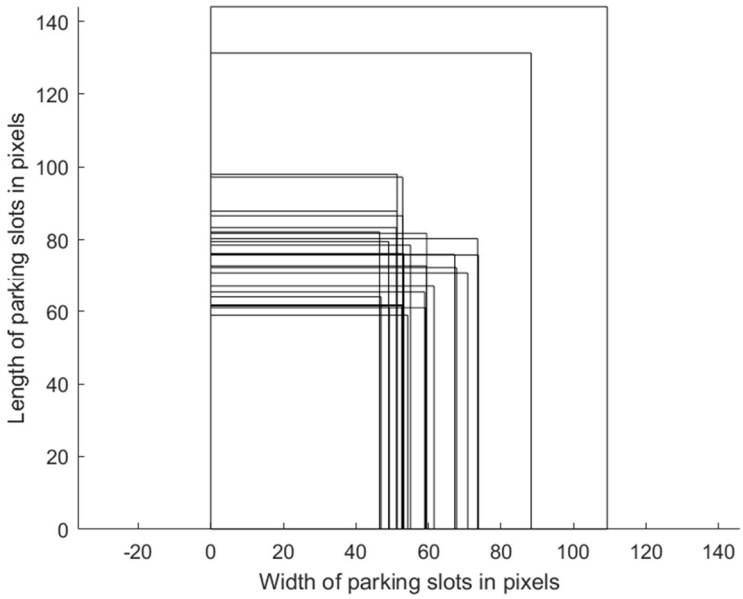


Figure 14. Visualisation of the Delineated Parking Slots after Clustering

#### 2.5.2. Calculate the Average Length and Width

The code for calculating the average length and width of all the parking slots is shown below:

```
avgLength = mean(classifiedMeanTransformed_AspectRatio(:,4));
avgWidth = mean(classifiedMeanTransformed_AspectRatio(:,3));
fprintf('The average length and width of the parking slots are %.3f and %.3f
pixels.\n',avgLength, avgWidth)
fprintf('The average aspect ratio is %.3f\n', avgLength/avgWidth)
```

The average length and width of parking slots are 79.808 and 60.335 pixels. And the average aspect ratio is

1.323.

### 2.5.3. Post Process the Parking Slots

To address the two concerns (the size of parking slots is not uniform and the bounding boxes are smaller than the actual parking slots) addressed in section 2.4. We first assume all the length and width of the parking slots remain constant in the images. Then, we assume the cars take 80% of the length of the parking slots. Therefore, we post-process the bounding box, making all of them have the length and width equal to:

$$length\ of\ every\ bounding\ box = 1.25 \times average\ length\ of\ all\ bounding\ boxes = 99.76\ pixels$$

$$width\ of\ every\ bounding\ box = average\ width\ of\ all\ bounding\ boxes = 60.335\ pixels$$

The code is shown below:

```
% Post-process the length and width of the parking slots

% Use average length and width of the parking slots as constraints
% Use the vehicle occupied 80% of the parking slot assumption

load('GroundTruthBarryStreet.mat')

classifiedMeanTransform_postprocess = classifiedMeanTransformed;

for n=1:length(classifiedMeanTransform_postprocess)
    % x direction is the length
    if classifiedMeanTransform_postprocess(n,3)/classifiedMeanTransform_postprocess(n,4) >=
(avgLength/avgWidth)

        differenceX = classifiedMeanTransform_postprocess(n,3) - avgLength*1.25;
        differenceY = classifiedMeanTransform_postprocess(n,4) - avgWidth;

    else % x direction is width
        differenceX = classifiedMeanTransform_postprocess(n,3) - avgWidth;
        differenceY = classifiedMeanTransform_postprocess(n,4) - avgLength*1.25;

    end
    classifiedMeanTransform_postprocess(n,1) = classifiedMeanTransform_postprocess(n,1) +
(differenceX/2);
    classifiedMeanTransform_postprocess(n,2) = classifiedMeanTransform_postprocess(n,2) +
(differenceY/2);
    classifiedMeanTransform_postprocess(n,3) = classifiedMeanTransform_postprocess(n,3) -
differenceX;
    classifiedMeanTransform_postprocess(n,4) = classifiedMeanTransform_postprocess(n,4) -
differenceY;

end
```

### 2.5.4. Recalculate the Precision-Recall Curve and Show Final Bounding Boxes

After post-process, I recalculate and plot the precision-recall curve (see Figure 15).

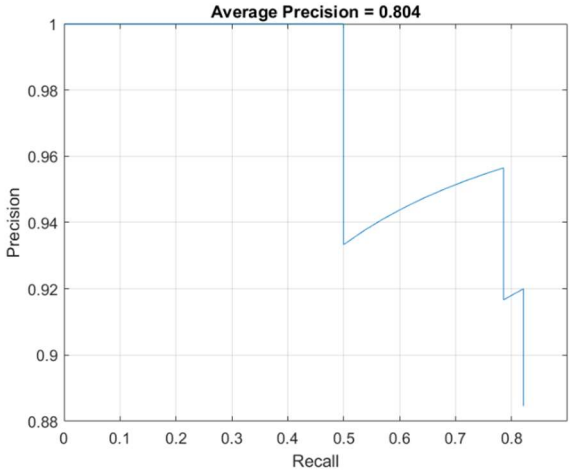


Figure 15. Precision-Recall Curve after Post Process

We can observe the average precision AP50 increases from 37.1% to 80.4%, which is around a 43.3% improvement. The code for plotting the precision-recall curve is shown below:

```
redictedBox = classifiedMeanTransform_postprocess;
predictedScore = classifiedScoreMean;
predictionResults = table('Size',[1,2], ...
    'VariableTypes',{'cell','cell'}, ...
    'VariableNames',{'Boxes','Scores'});
predictionResults(1,'Boxes') = {predictedBox};
predictionResults(1,'Scores') = {predictedScore};
Vehicles{1} = ParkingSlots;
Groundtruth = table(Vehicles);

[ap, recall, precision] = evaluateDetectionPrecision(predictionResults, Groundtruth, 0.5);

figure;
plot(recall,precision);
grid on
title(sprintf('Average Precision = %.3f',ap));
xlabel('Recall');
ylabel('Precision');

% exclude the class "-1"
fprintf('The mean precision of the bounding boxes is %.3f\n', mean(precision(2:end)));
fprintf('The mean recall of the bounding boxes is %.3f\n', mean(recall(2:end)));
```

The mean precision and recall of the 26 detected parking slots are 97.1% and 46%.

Then, I will visualise the delineated parking slots after the post process with the ground truth bounding boxes of the Barry Street dataset. The code is shown below:

```
figure;
hold on;
% Plot the ground truth bounding box
for i=1:length(ParkingSlots)
    rectangle('Position', [ParkingSlots(i,1), ParkingSlots(i,2), ...
        ParkingSlots(i,3), ParkingSlots(i,4)],'EdgeColor','r','LineWidth',2)
end

% Plot the prediction bounding box
for i=1:length(classifiedMeanTransform_postprocess)
    rectangle('Position', [classifiedMeanTransform_postprocess(i,1), ...
        classifiedMeanTransform_postprocess(i,2), ...
        classifiedMeanTransform_postprocess(i,3), ...
        classifiedMeanTransform_postprocess(i,4)],'EdgeColor','b','LineWidth',2)
end

% reverse the y axis, because the image coordinate start from the upper
% left corner while the matlab rectangle start from the lower left corner
set(gca, 'ydir', 'reverse');
xlim([0 1020]); ylim([0 663]);

% create empty line for generating legend
rline = line(NaN,NaN,'LineWidth',2,'Color','r');
bline = line(NaN,NaN,'LineWidth',2,'Color','b');
legend('Ground Truth','Detections')

hold off;
```

Note that, the image coordinates start from the upper-left corner while the rectangle plot coordinates start from the lower-left corner. Therefore, we need to reverse the y-axis after plotting the bounding boxes. The result is shown in Figure 16 and Figure 17.

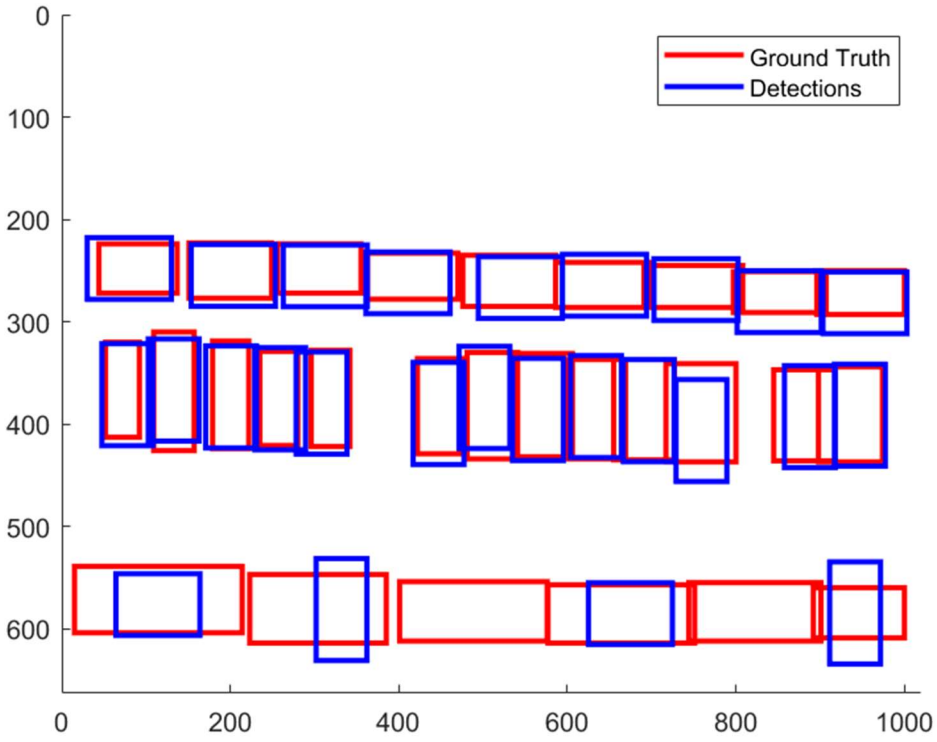


Figure 16. Visualisation of the Delineated Parking Slots and the Ground Truth for the Barry Street Dataset

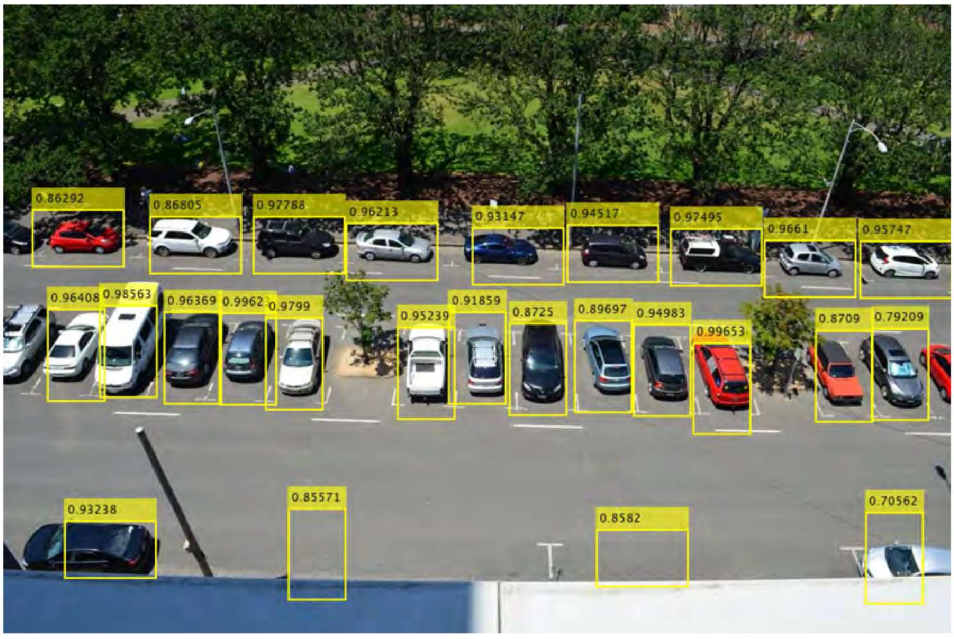


Figure 17. The Final Parking Slot Delineations for Berry Street Dataset

The code for showing the final parking slot delineations on the Berry Street Image is shown below:

```
% Transform back to the resized images coordinates
classifiedMean_postprocess = classifiedMeanTransform_postprocess;
classifiedMean_postprocess(:,1) = classifiedMeanTransform_postprocess(:,1) + 141;
classifiedMean_postprocess(:,2) = classifiedMeanTransform_postprocess(:,2) + 58;

BarryStreetImageAnnBoxes = insertObjectAnnotation(BarryStreetImageProcessed,'rectangle',...
    classifiedMean_postprocess,classifiedScoreMean, 'LineWidth', 2);
figure
imshow(BarryStreetImageAnnBoxes)
xlim([141 1140]); ylim([58 720]);
```

# 3. Discussion:

## 3.1.Delineation

As we discussed in section 2.3, the delineation is done by first detecting cars in the images and then using a density-based clustering algorithm to cluster the detection into classes (the parking slots). This method is based on an assumption that the number of cars on the road is much fewer than the number of cars parked in the parking slots. Therefore, if the traffic of Berry Street is busy, the CCTV may capture many cars on the road. This will cause the model to think the area on the road may also be valid parking slots. Moreover, if few cars are parked in the parking slots, or some of the parking has few cars parked during the image capturing period, this model may not be able to find the delineations of all parking slots.

As we can observe from Figure 17, there are two parking slots in the bottom row of the parking area that are missing due to the fewer cars parked there causing a lack of detection in that area. Moreover, in Figure 17, there are two parking slots oriented in the wrong direction because the wall occludes part of the parking slots/cars, and the parking slot at the lower-right corner is cut by the image. Therefore, to get a good delineation, we should ensure the whole parking area is captured and not be partially occluded.

Furthermore, the post-process of the delineation is based on two assumptions. First, it assumes all parking slots are shown with the same width and length in the images. Second, it assumes all cars take 80% of the area of parking slots. These assumptions may be problematic. First, the images are usually captured non-vertical. This will cause the scale variance within the image, therefore, contradicts the first assumption. Second, the size of cars is not constant, some cars may take more or less than 80% of the length of parking slots, hence, cause a problem on the second assumption. Consequently, the limitation of this method is that the parking area should not be too large, and the size of cars parked should not vary too much. In this case, the scale variance within the image and the size variance among cars could be ignored.

## 3.2.Classification

Our classifier takes the images of delineated parking slots as input and predicts if the parking slots are occupied or empty. As we can see in Figure 7, the limitation of this classification model is that it is easier subjected to the environment such as shadow. Shadows in Figure 7 make the model predict empty parking slots as occupied. To address this issue, good lighting in the parking area or proper image pre-process to handle the shadows is required.

## 3.3.Effect of Post Process for Parking Slot Delineation

As we can see in Table 1, the post-process can largely increase the delineation performance because the bounding boxes drawn by the object detection model are not accurately aligned with the boundary of the cars and also the object detection model is trained for cars detection, not for parking slot detection.

If we use other computer vision techniques to delineate the parking slots such as boundary detection which directly extracts the boundary of the parking slots, we may be able to save the effort on post-process. On the other hand, since the parking slots are usually fixed, human annotating the coordinates of parking slots is also possible. It only needs to be done once, then we can always get the accurate result.

| | Before Post Process | After Post Process |
|---|---|---|
| AP50 | 37.1% | 80.4% |
| Mean precision of 26 parking slots | 60.7% | 97.1% |
| Mean recall of 26 parking slots | 31.3% | 46% |

Table 1. Effect of Post Process

# 4. Conclusion

In this assignment, we have introduced the method to automatically delineate the parking slots and then automatically classify the occupancy of all the parking slots. We also found that the post-process can improve the performance of delineation from the less accurate bounding boxes drawn by the object detection model. The limitation of this method such as environment variance and the three assumptions (few cars on the road, no scale variance within image, cars take 80% of the length of parking slot) are also discussed. This automatic parking occupancy detection model could benefit the traffic and route planning in the city. Also, it is easy to implement. The delineation only needs to be done once. After that, this task becomes a simple classification problem. However, there is always a trade-off between automation and accuracy. We need to weigh the pros and cons carefully to select the best approach.

## 5. Reference:


[1] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science,* vol. 349, no. 6245, pp. 255-260, 2015.

[2] K. D. Foote, "A Brief History of Machine Learning," ed, 2019.

[3] F. Rosenblatt, *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[4] C. t. W. projects. "Computer vision - Wikipedia." https://en.wikipedia.org/w/index.php?title=Computer_vision&oldid=1024857806 (accessed 30, 05, 2021).

[5] C. t. W. projects. "Object detection - Wikipedia." https://en.wikipedia.org/w/index.php?title=Object_detection&oldid=1020628489 (accessed 30, 05, 2021).

[6] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580-587.

[7] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440-1448.

[8] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *arXiv preprint arXiv:1506.01497,* 2015.

[9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779-788.

[10] J. Hui. "mAP (mean Average Precision) for Object Detection - Jonathan Hui - Medium." Medium. https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173 (accessed 30, 05, 2021).

[11] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Kdd*, 1996, vol. 96, no. 34, pp. 226-231.

[12] D. Acharya and K. Khoshelham, "Real-time image-based parking occupancy detection and automatic parking slot deliniation using deep learning: A tutorial."