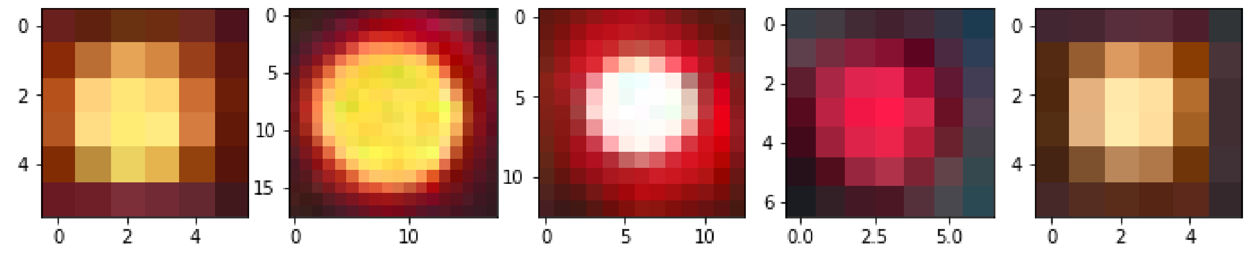
EE/CNS/CS 148 Homework 2 Report

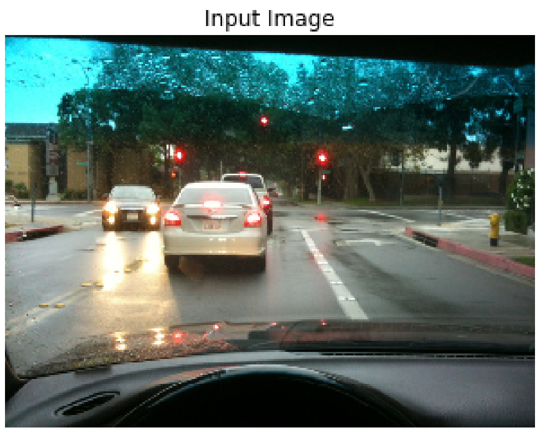
Albert Zhai

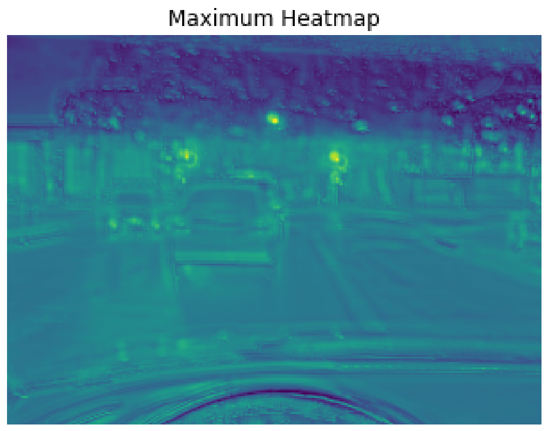
23 April 2020

My algorithm uses matched filtering with 5 templates, which were slices of images that I hand-selected from the dataset to try to cover some different scales and lighting conditions. The templates were each normalized by their mean and then centered at zero. Each input image was cross-correlated with all of the templates, producing heatmaps for each template. The pixelwise maximum across the 5 heatmaps was then calculated to obtain a final heatmap. This was then divided into a grid of 30 × 30 squares, and the highest-score pixel in each square was retrieved. These pixels were collected into a list and sorted by descending score. Some final non-maximum suppression was performed by iterating through the sorted list in order and selecting pixels which are at least a taxicab distance of 30 pixels away from all of the already selected ones. Only the top 4 points were kept for each image. The bounding boxes were calculated by combining each selected pixel with the size of the template which had the highest correlation there, and confidence values were produced by applying a sigmoid function to the heatmap score.

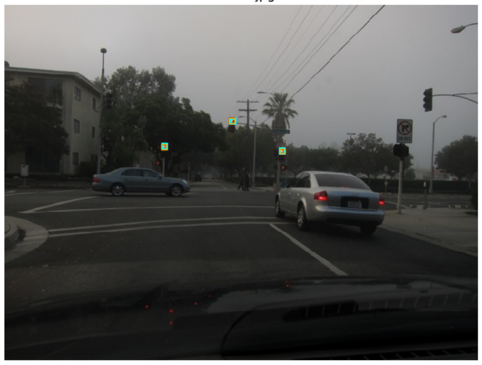
The templates used for matched filtering and an example heatmap (maximum over templates) are visualized below. The box locations were localized by adding to each selected pixel the size of the template which had the highest correlation there.

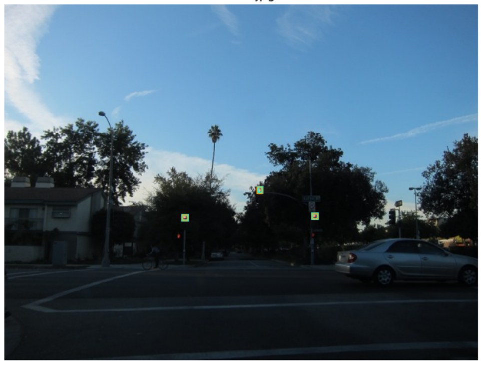


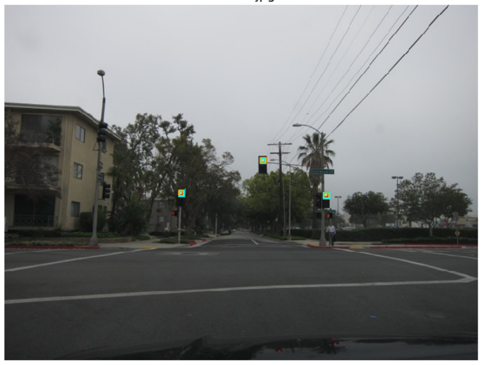


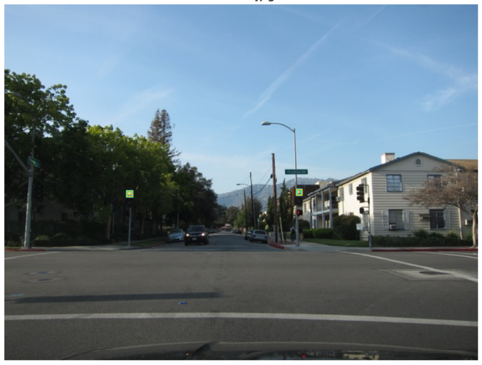


Some examples where the algorithm succeeded are shown below (detected bounding boxes drawn in cyan, ground truth in orange):


RL-092.jpg

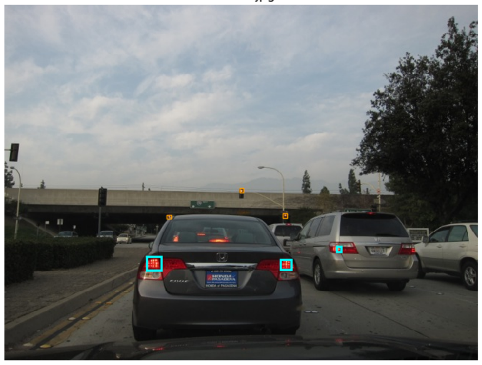

RL-115.jpg

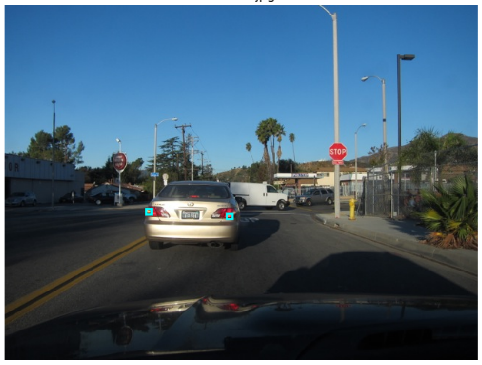

RL-271.jpg

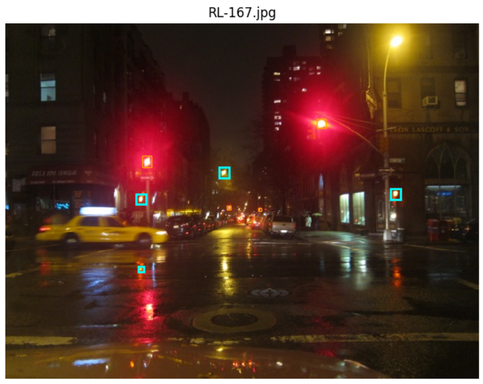

RL-326.jpg

These examples were easy because the appearance of the traffic lights were similar to those used as the filter templates, and there were not any similar patterns which could cause false alarms.

Some examples where the algorithm failed are shown below:

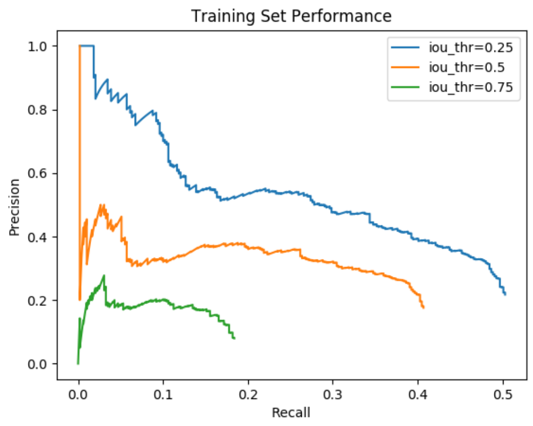

RL-030.jpg

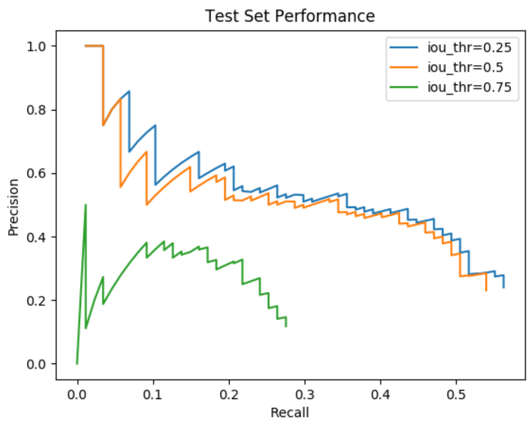

RL-140.jpg



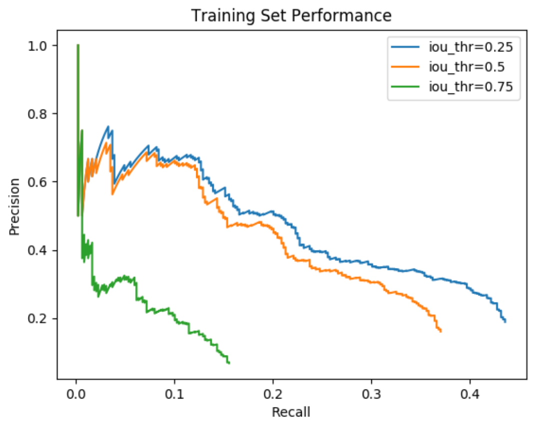
In example 167, the rainy nighttime conditions caused the red lights to look significantly different from any of the selected templates and thus be missed. In the other examples, misleading patterns were formed from non-target objects such as tree shadows and dark-colored car taillights. Also, when the traffic lights are extremely far away and span regions much smaller than any of the templates, they cannot be detected.

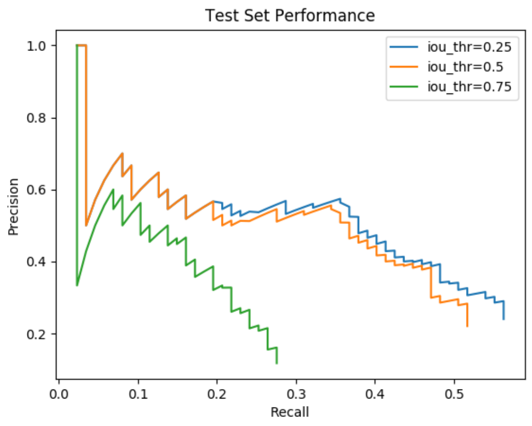
The precision-recall curves for the full algorithm are shown below:





The algorithm was weakened by using only the first filter template instead of all 5. The PR curves for this weakened version are shown below:





As the IoU threshold increases, both precision and recall become strictly worse, as the criterion for a detection being correct becomes more difficult. Surprisingly, for the full algorithm, test performance is slightly better than training performance, especially for the 0.5 IoU threshold. This can only be a chance occurrence resulting from the test set happening to have easier images than the training set. Since the algorithm was tweaked to perform well on the training set, we expect the test performance to be lower. The single-filter version of the algorithm performed similarly to the full algorithm except in the low-recall, high-precision regions of the 0.5 and 0.25 IoU threshold curves. The sharper template distribution makes the weakened algorithm more targeted towards a certain type of red light appearance, making it better in some situations. Comparing to a weakened version of the algorithm gives some insight into what components of the algorithm are most crucial to its performance, helping to guide directions of improvement.

The GitHub repository is located at https://github.com/ajzhai/caltech-ee148-spring2020-hw02. The code for the detection algorithm is in the "run_predictions.py" file, and the code for generating PR curves is in the "eval_detector.py" file. The JSON files of train and test predictions are at "hw02_preds/preds_train.json" and "hw02_preds/preds_test.json" respectively.