EE/CNS/CS 148 Homework 1 Report

Albert Zhai

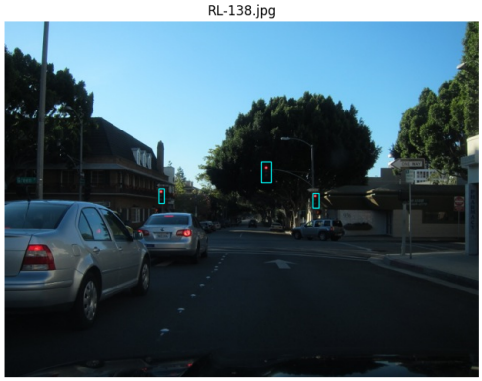
14 April 2020

The algorithms I tried can all be considered variations on matched filtering. To evaluate performance, I ran the algorithm on sample images and roughly kept track of precision and recall in my head. I tried to maximize recall while maintaining an approximate false alarm rate of under 10 percent, a somewhat arbitrary value. The best evaluation metric depends on the overall system that the detection algorithm is a part of and the downstream task(s) it is used for. For example, in a self-driving car, detecting at least one light in an intersection is very important but detecting additional lights beyond the first is less important as they carry redundant information. The precision threshold should probably also be higher than 90 percent, as falsely perceiving a red light could be disastrous.

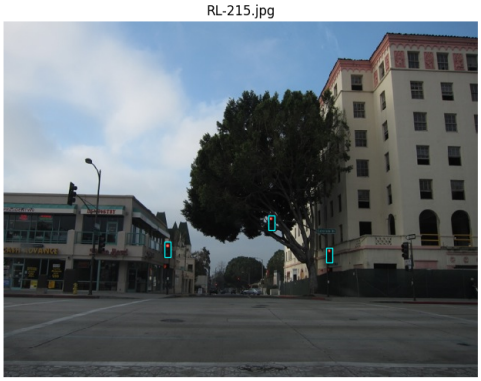
My best-performing algorithm used matched filtering with 3 kernels, which were slices of images that I hand-selected from the dataset to try to cover some different scales and lighting conditions. The kernels were each normalized by their mean and then centered at zero. Each input image was cross-correlated with all of the kernels (normalizing each window by its mean), producing score maps for each kernel. All of the pixels with score above 0.74 were collected into a list and sorted by descending score. A form of non-maximum suppression was implemented by iterating through the sorted list in order and selecting pixels which are at least a Euclidean distance of 25 pixels away from all of the already selected ones. The final bounding boxes were calculated by combining each selected pixel with the size of its corresponding kernel.

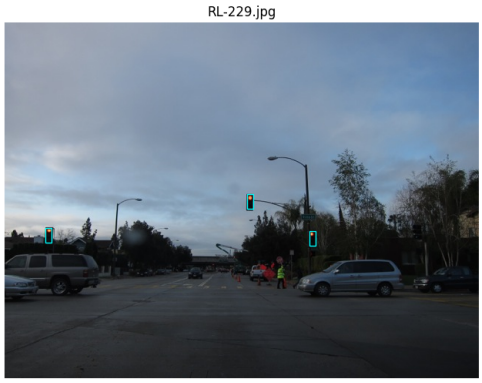
Some examples where the algorithm succeeded are shown below (detected bounding boxes drawn in cyan):

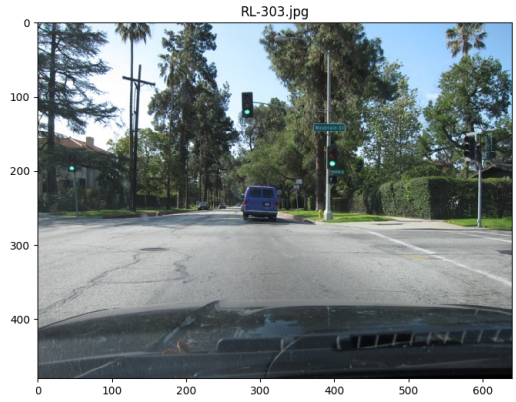
RL-138.jpg



RL-215.jpg

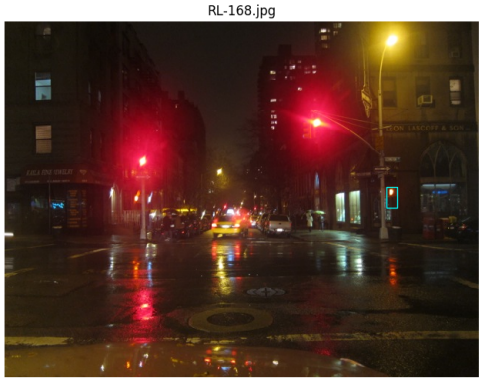


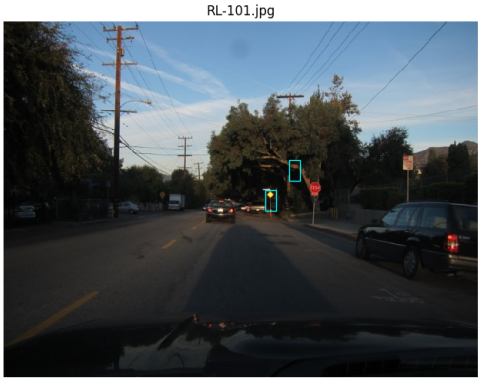


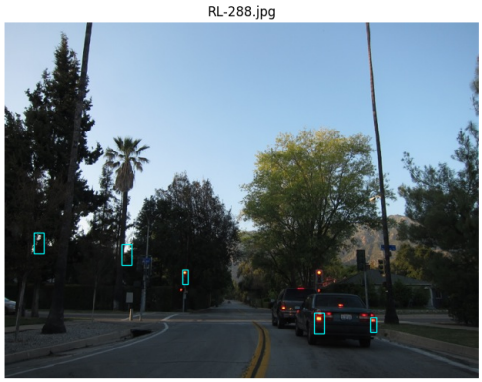


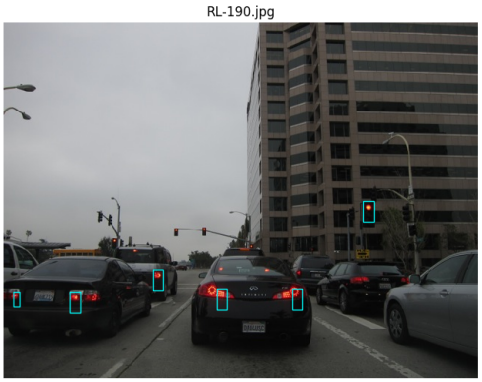
These examples were easy because the appearance of the traffic lights were similar to those used as the filter kernels, and there were not any similar patterns which could cause false alarms.

Some examples where the algorithm failed are shown below:









In example 168, the rainy nighttime conditions caused the red lights to look significantly different from any of the selected kernels 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.

One problem of the current approach is that it does not adapt well to different scales (distances) and rotations of the traffic lights. This could possibly be fixed in a "shotgun" manner by using multiple scales and orientations of the kernels for matching.

The GitHub repository is located at https://github.com/ajzhai/caltech-ee148-spring2020-hw01. The code for the detection algorithm is in the "run_predictions.py" file, and the code for visualization is in the "visualize.py" file. The JSON file of predictions is at "hw01_preds/preds.json".