Vehicle Detection Project

The goals / steps of this project are the following:

- Perform a Histogram of Oriented Gradients (HOG) feature extraction on a labeled training set of images and train a classifier Linear SVM classifier
- Optionally, you can also apply a color transform and append binned color features, as well as histograms of color, to your HOG feature vector.
- Note: for those first two steps don't forget to normalize your features and randomize a selection for training and testing.
- Implement a sliding-window technique and use your trained classifier to search for vehicles in images.
- Run your pipeline on a video stream (start with the test_video.mp4 and later implement on full project_video.mp4) and create a heat map of recurring detections frame by frame to reject outliers and follow detected vehicles.
- Estimate a bounding box for vehicles detected.

Project Rubric Points As below

###Here I will consider the rubric points individually and describe how I addressed each point in my implementation.

###Writeup / README

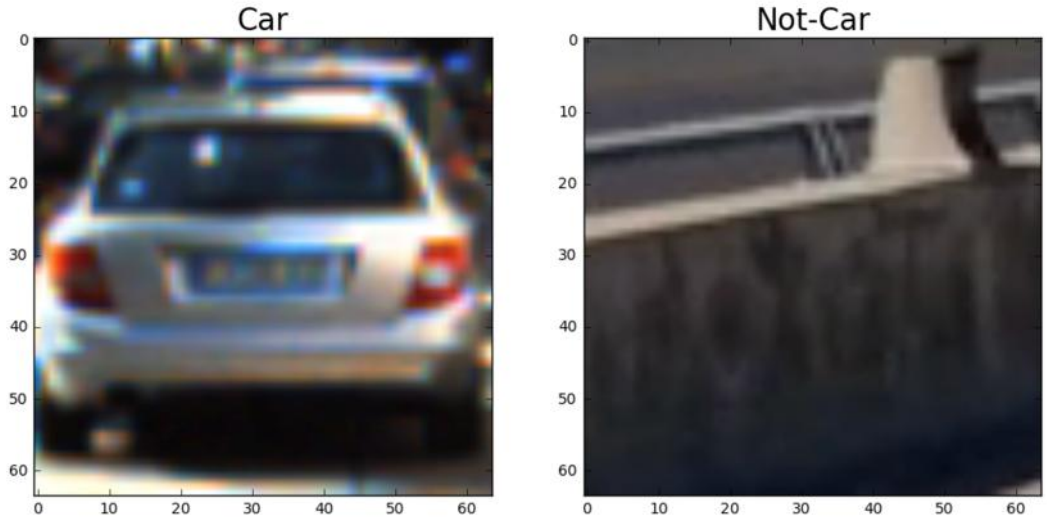
####1. Provide a Writeup / README that includes all the rubric points and how you addressed each one. You can submit your writeup as markdown or pdf. Here is a template writeup for this project you can use as a guide and a starting point.

You're reading it! And uploaded same at GitHub repo

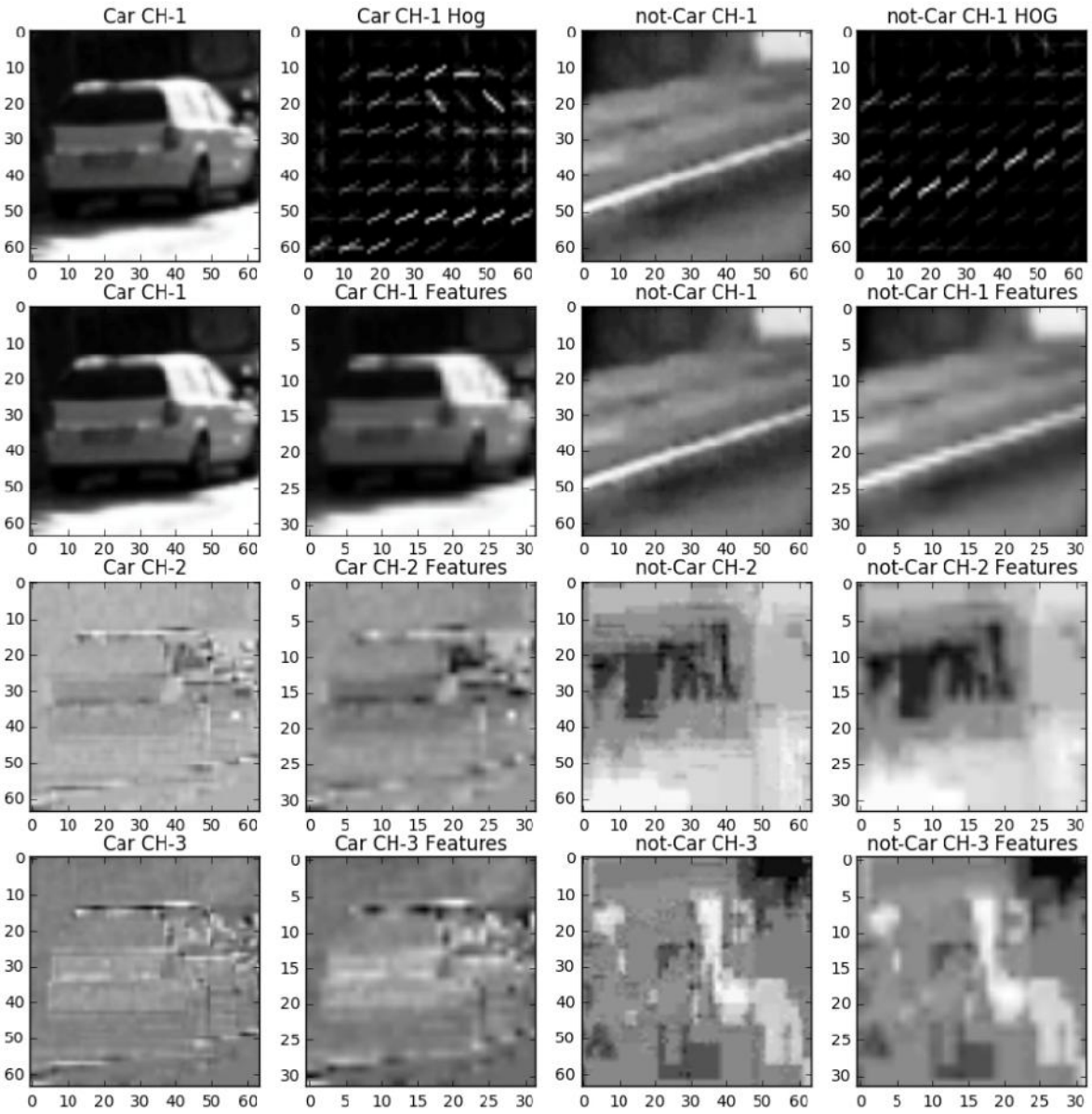
###Histogram of Oriented Gradients (HOG)

####1. Explain how (and identify where in your code) you extracted HOG features from the training images.

The code for this step is contained in the first code cell of the Project_Vechicle_Detection.ipynb (or in lines # through # of the file called `some_file.py`). I started by reading in all the `vehicle` and `non-vehicle` images. Here is an example of one of each of the `vehicle` and `non-vehicle` classes:



I then explored different color spaces and different `skimage.hog()` parameters (`orientations`, `pixels_per_cell`, and `cells_per_block`). I grabbed random images from each of the two classes and displayed them to get a feel for what the `skimage.hog()` output looks like.
Here is an example using the `YCrCb` color space and HOG parameters of `orientations=8`, `pixels_per_cell=(8, 8)` and `cells_per_block=(2, 2)`:



####2. Explain how you settled on your final choice of HOG parameters.

I tried various combinations of parameters

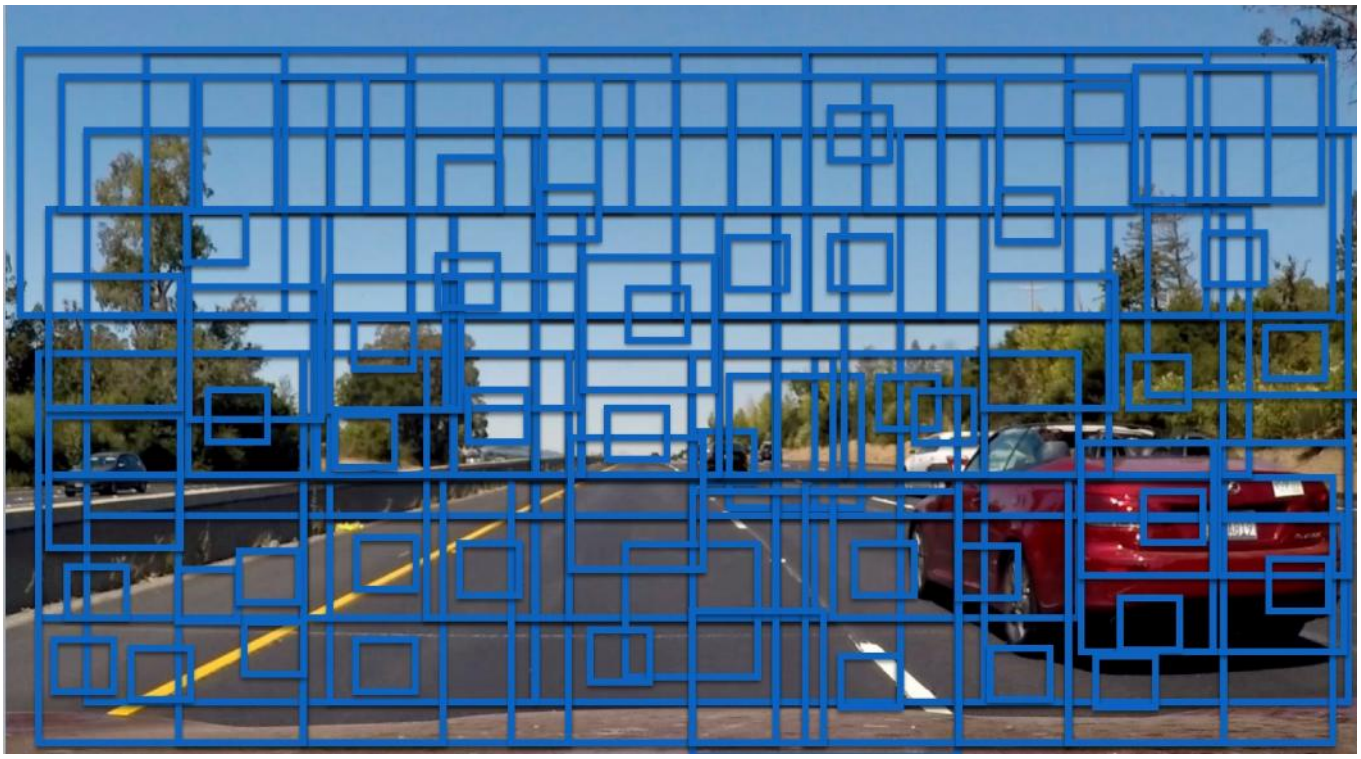
####3. Describe how (and identify where in your code) you trained a classifier using your selected HOG features (and color features if you used them).

I trained a linear SVM using...

###Sliding Window Search

####1. Describe how (and identify where in your code) you implemented a sliding window search. How did you decide what scales to search and how much to overlap windows?

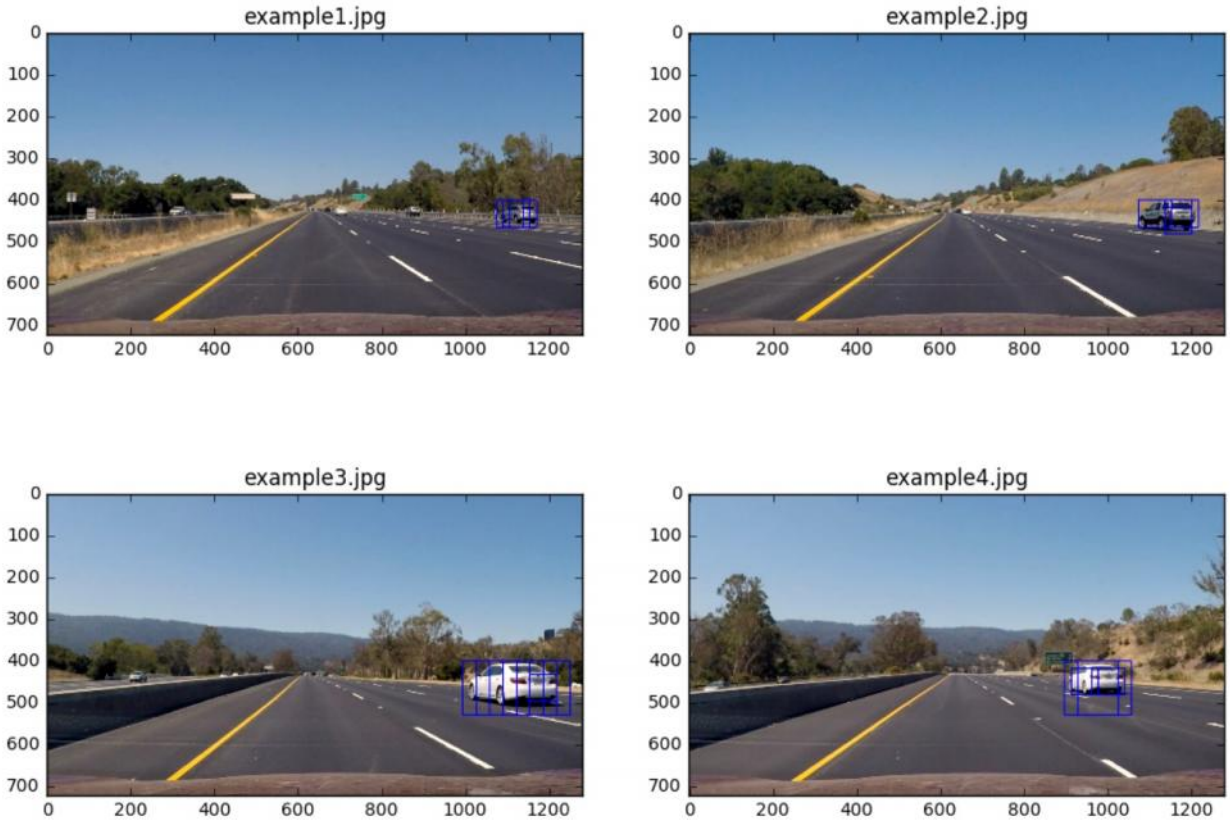
I decided to search random window positions at random scales all over the image and came up with this



####2. Show some examples of test images to demonstrate how your pipeline is working. What did you do to optimize the performance of your classifier?

Ultimately, I searched on two scales using YCrCb 3-channel HOG features plus spatially binned color and histograms of color in the feature vector, which provided a nice result.

Here are some example images:



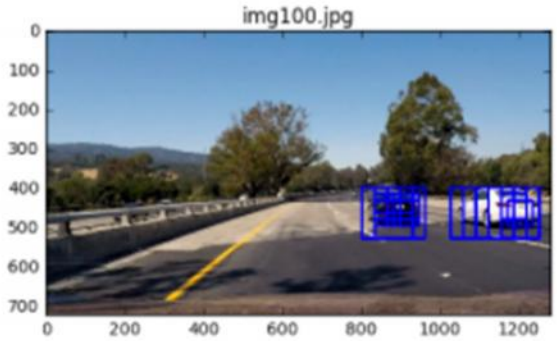
Video Implementation

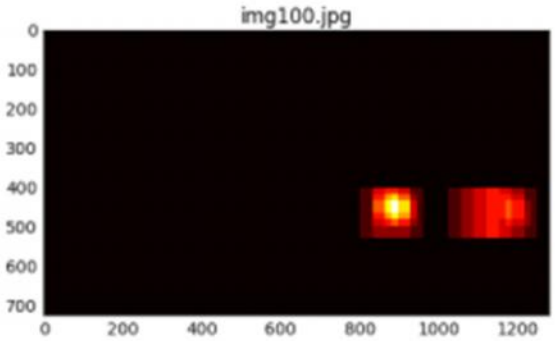
####1. Provide a link to your final video output. Your pipeline should perform reasonably well on the entire project video (somewhat wobbly or unstable bounding boxes are ok as long as you are identifying the vehicles most of the time with minimal false positives.) Here's a link to my video result

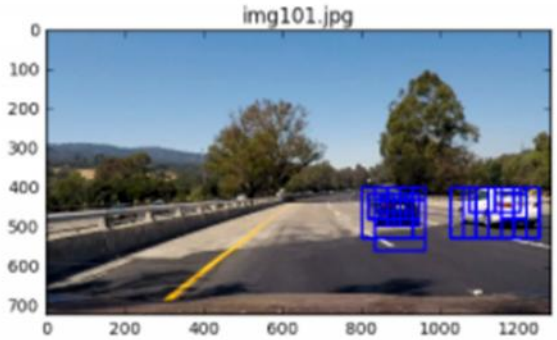
####2. Describe how (and identify where in your code) you implemented some kind of filter for false positives and some method for combining overlapping bounding boxes.

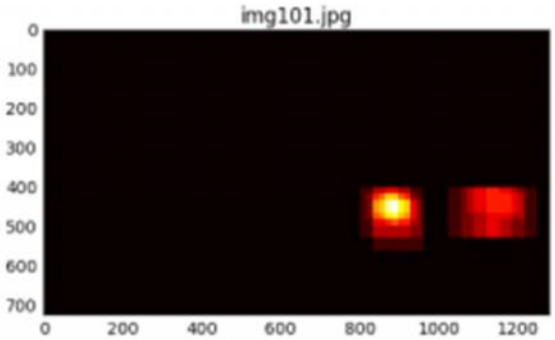
I recorded the positions of positive detections in each frame of the video. From the positive detections I created a heatmap and then thresholded that map to identify vehicle positions. I then used `scipy.ndimage.measurements.label()` to identify individual blobs in the heatmap. I then assumed each blob corresponded to a vehicle. I constructed bounding boxes to cover the area of each blob detected.
Here's an example result showing the heatmap from a series of frames of video, the result of `scipy.ndimage.measurements.label()` and the bounding boxes then overlaid on the last frame of video: Here are six frames and their corresponding

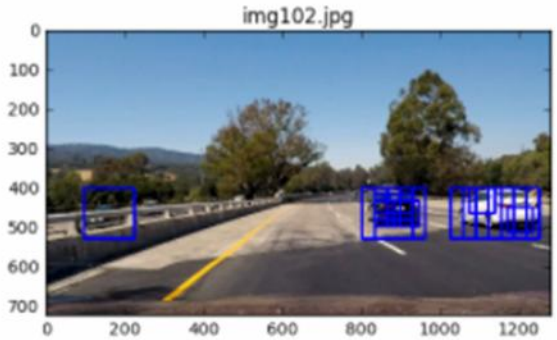
heatmaps:

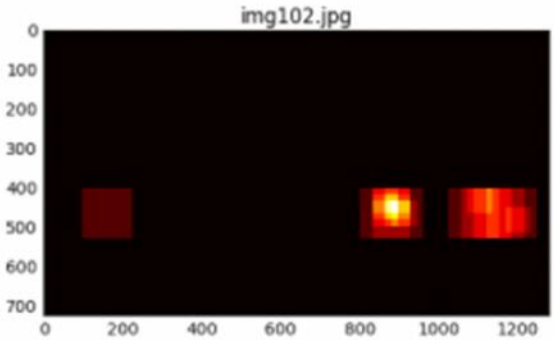

img100.jpg
0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200

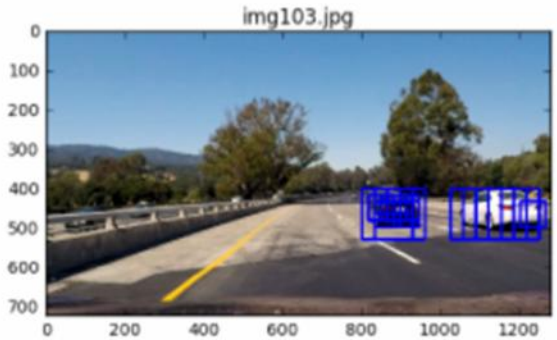

img100.jpg
0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200

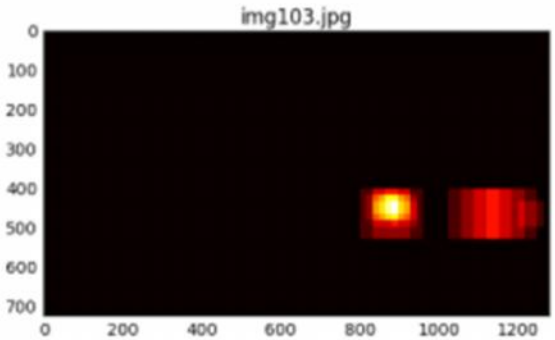

img101.jpg
0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200

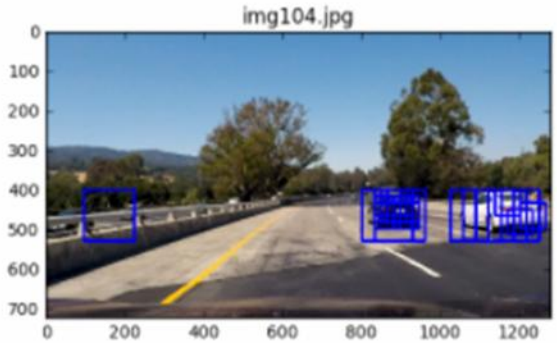

img101.jpg
0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200

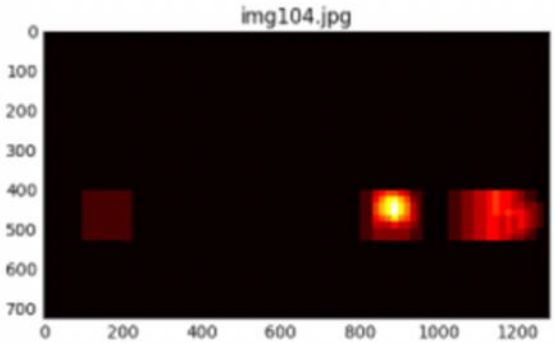

img102.jpg
0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200


img102.jpg
0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200

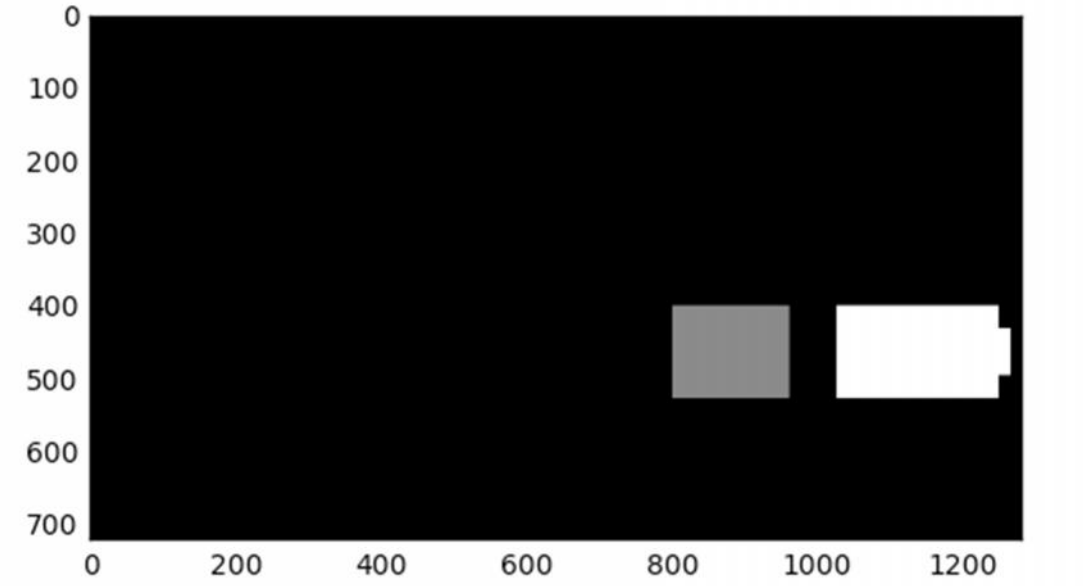

img103.jpg
0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200


img103.jpg
0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200


img104.jpg
0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200


img104.jpg
0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200

Here is the output of `scipy.ndimage.measurements.label()` on the integrated heatmap from all six frames:



Here the resulting bounding boxes are drawn onto the last frame in the series: