Describe your algorithm and any decisions you made to write your algorithm a particular way.

get_positive_features.m:
In this function, the images in the file path are read in and converted to single image. Resize the image to the same as the desired template size. Use vl-hog function ot build the gradient histogram. At last return all the histograms as feature_pos.

get_random_negative_features.m
In this function, I first calculate the amount of sub images to take from each input non-face iamge. Calculate the random pixel index that will be use to crop the sub image each round, then use imcrop to get the random sub image. Calculate histogram for each sub non-face scene image and return them together as features_neg

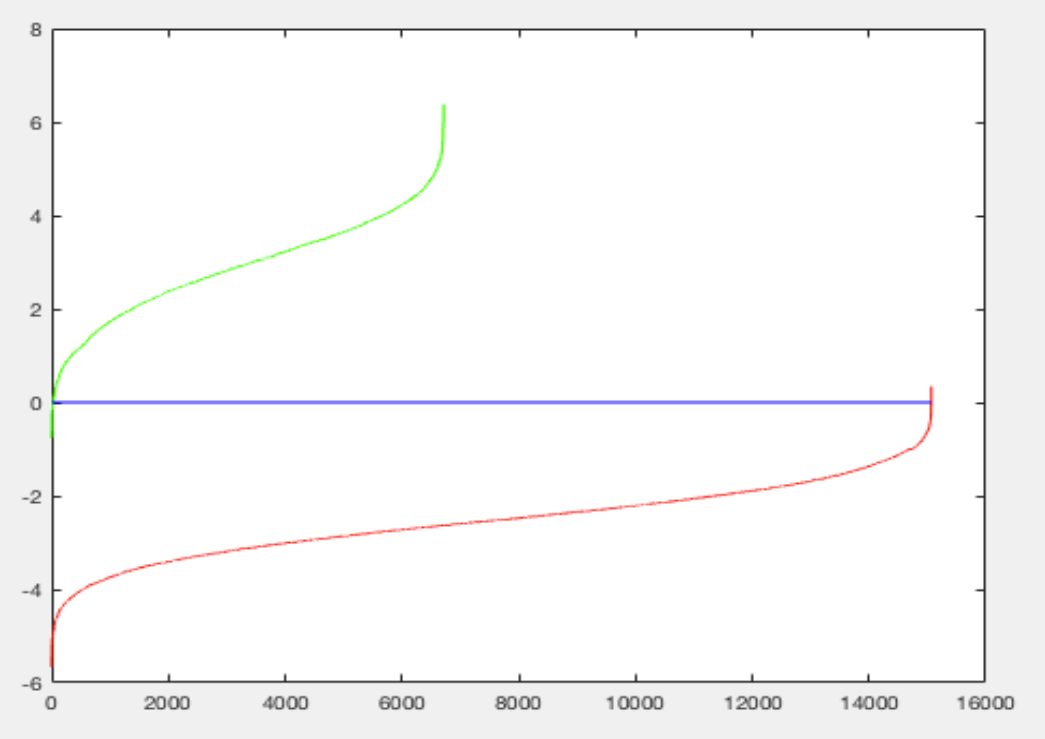
classifier training
In this part, I set the lambda value be 0.0001. Label each positive features HOG as 1 and each negative feature HOG as -1. Then input both feature matrixes along with their labels to function vl_svmtrain. The returned values are the wanted SVM classifier.

run_detector.m
In this function, I first set the threshold of score and the different scales to run for test image. For each test image, resize it to a scale, calculate its HOG using vl_hog. After this I calculate the window indexes for sliding window checking. During this process, the size of window is equal to $(\text{template\_size/hog\_cell\_size})^2$ get the corresponding histogram from the step above. At the same time, calculate the x_min, y_min, x_max and y_max values of window corner values in resized image. At last, test the scores of all histograms with the svm classifier. Find the confidence scores above thresh value, store the image name, [x_min, y_min, x_max y_max] as a matrix record. Do the steps above for each test images and return the final results.

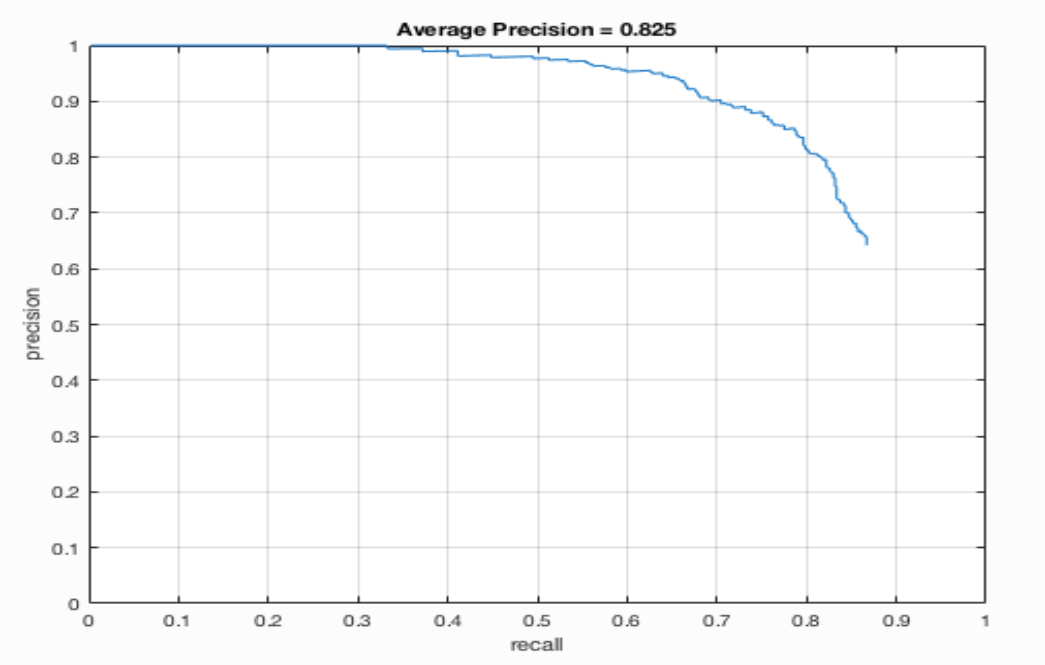
Show and discuss the results of your algorithm.

With the setting of:

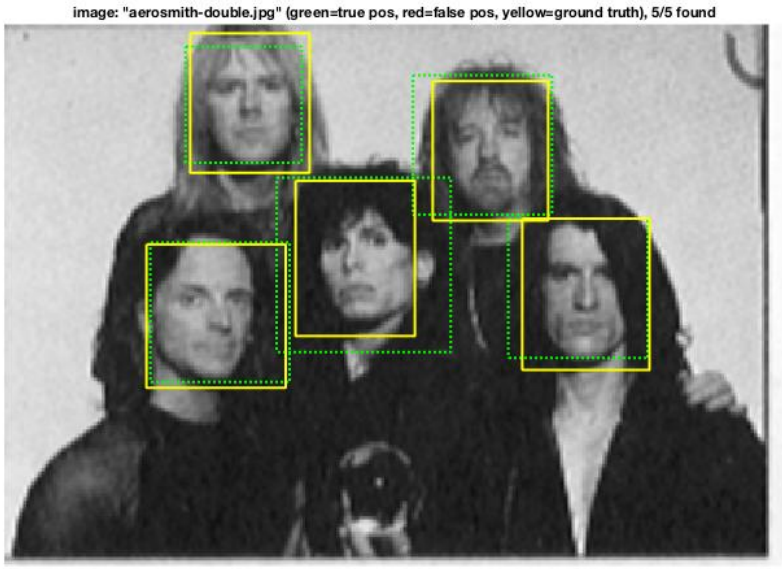
num_negative_examples = 15000
lambda = 0.0001
thresh = 0.88
SCALE = 0.1:0.1:1.1
`hog_cell_size = 6`

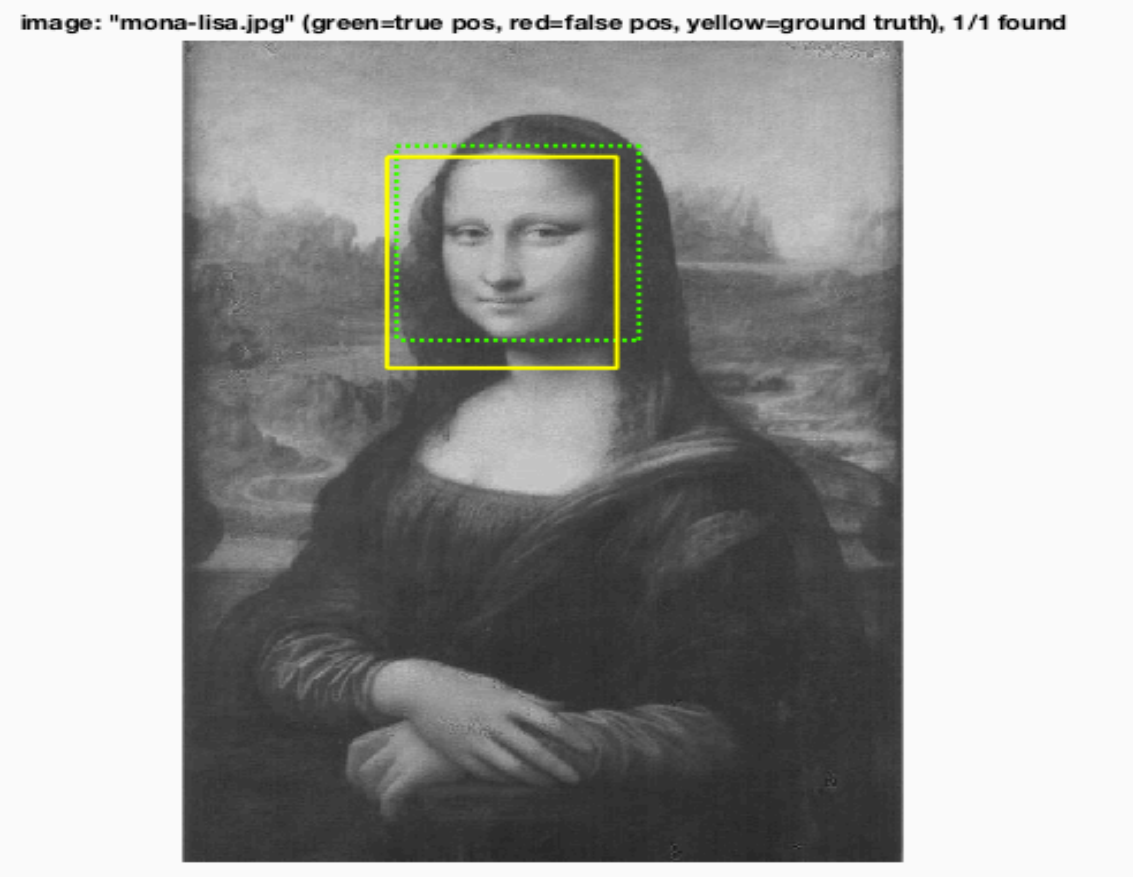


precision-recall curve



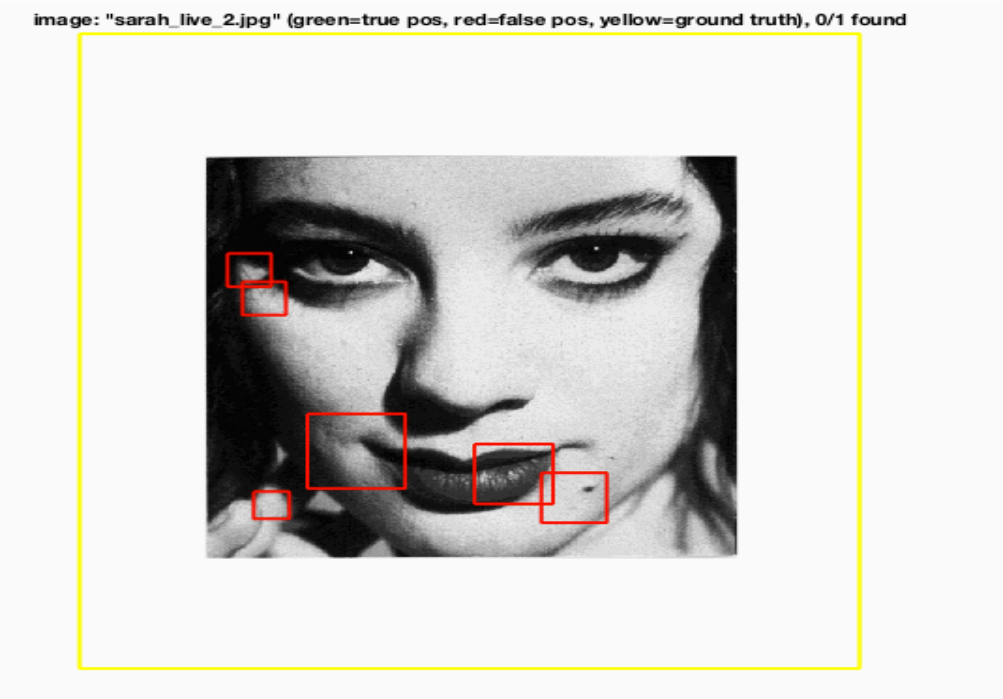
Some good results:





The above tow images had 100% faces found, the result only has true positive faces.

Also a bad results:



The whole image above is a decent and clear face. But the function doesn't detect it. It might because the confidence score get is somehow or a lot smaller than the set threshold value, which causes the detection box being filtered out.

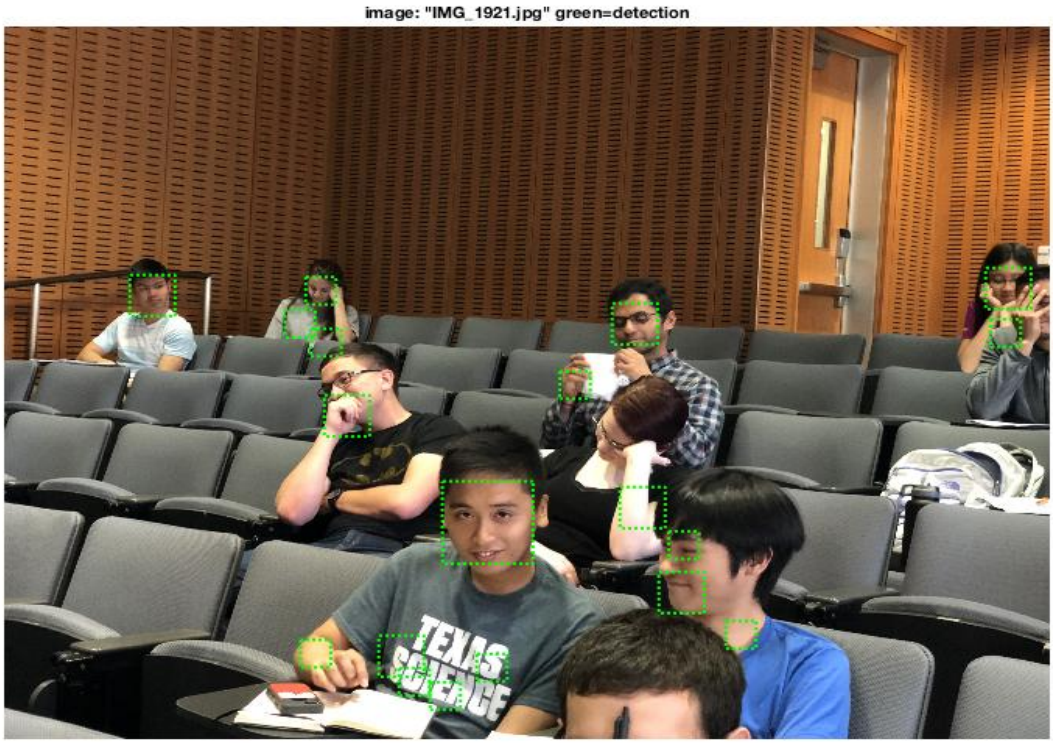
Output of extra test scenes in class:

Above are two of the outputs I get for extra test scenes in class:

```
Set SCALE = 0.1:0.2:1
```

image: "IMG_1928.jpg" green=detection

image: "IMG_1926.jpg" green=detection



Above shows that the detection accuracy one the first image is pretty high since students are showing their front faces without covering. But the results on the second and the third images tends to be much worse. Hard to detect some of the students' faces especial the ones covering papers or hands in front of their faces.

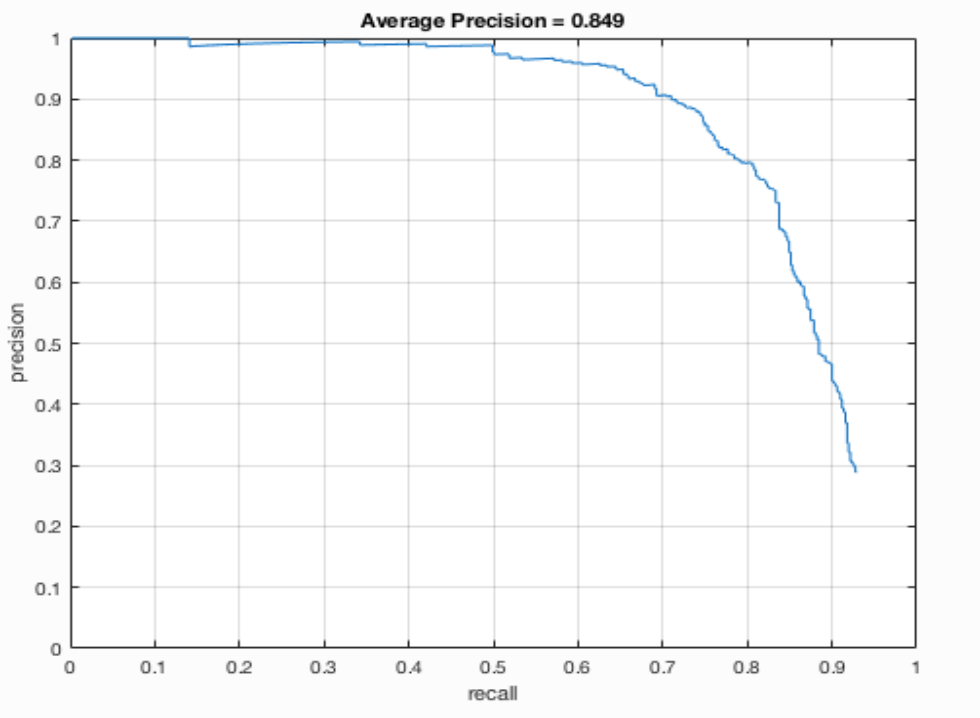
Extra credit:

1. **up to 5 pts: Find and utilize alternative positive training data to improve results. You can either augment or replace the provided training data.**

In this assignment I augmented additional training data. The additional images are from **At&T The Database of Faces**. Url: https://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html. To make the data format fits in the functions in this assignment, I wrote a script named pgm2jpg.m to convert all the images in .pmg to the wanted .jpg format. Also, I cast the images size to the desired 36*36. Then in the main script proj4.m, the additional images are read in and augmented to the positive feature matrix.

But the final precision rate doesn't change much compared with the rate using original smaller positive image set.

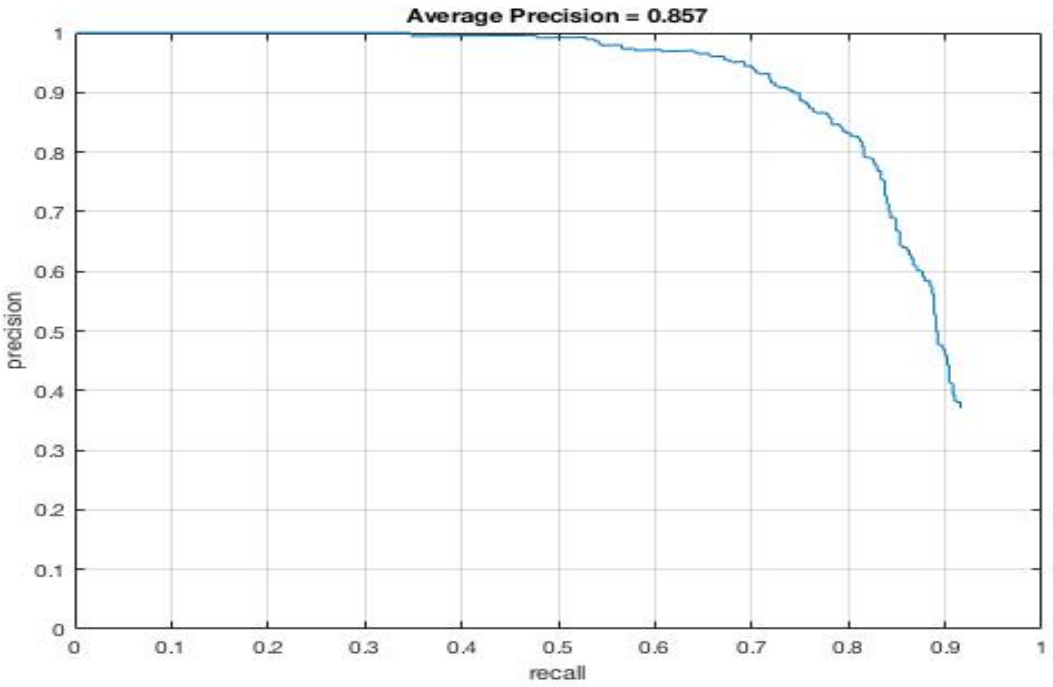
Output average precision:

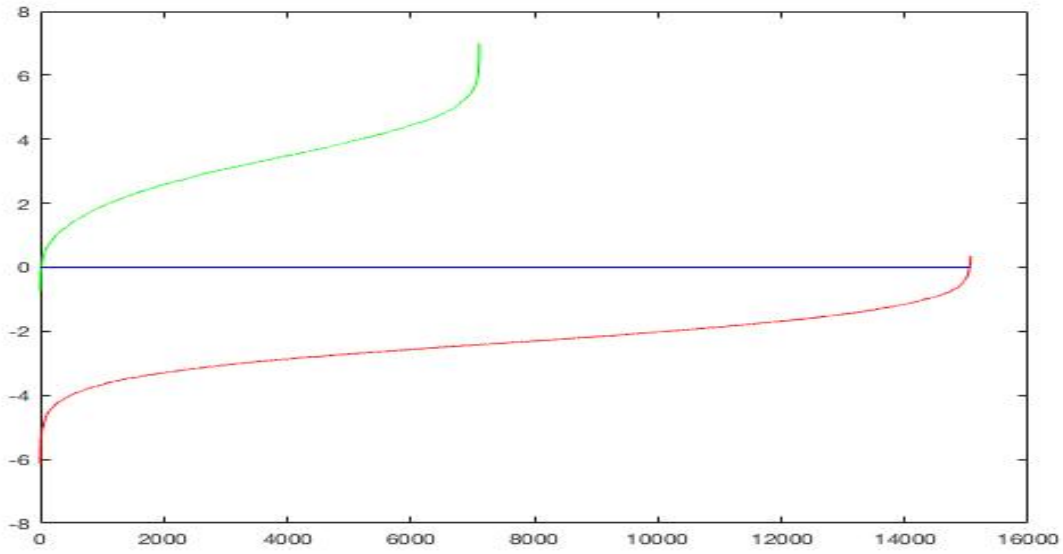


Compared with the average precision of 0.825, the new average precision has a slightly improvement which now is 0.849. In my opinion, the little improvement is because of the increment of input true face data size. The AT&T face database contains faces of a same person by with different factors changing. For some subjects, the images were taken at different times, varying the lighting, facial expressions (open / closed eyes, smiling / not smiling) and facial details (glasses / no glasses). Those increase the variety in face feature set data, making the prediction more accurate.

2. **up to 5 pts: Implement hard negative mining, as discussed in the Dalal and Triggs paper, and demonstrate the effect on performance**

In this part, I used the non-face features to get the confidence scores which are greater than the threshold I set. In for loop, calculate the scores round by round and add these false positive again to the feature matrix and train the SVM model again. After 200 rounds, the new SVM is done. Run precision detection, I get the output:





Just as the assignment page said, hard negative mining is somewhat unnecessary for face detection. The average precision doesn't change much compared with the previous result. But it is true that less false positive results were generated.