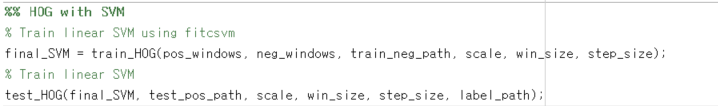
ICV Assignment #4 (Final Project)

**Sliding Window-based Object Detection**

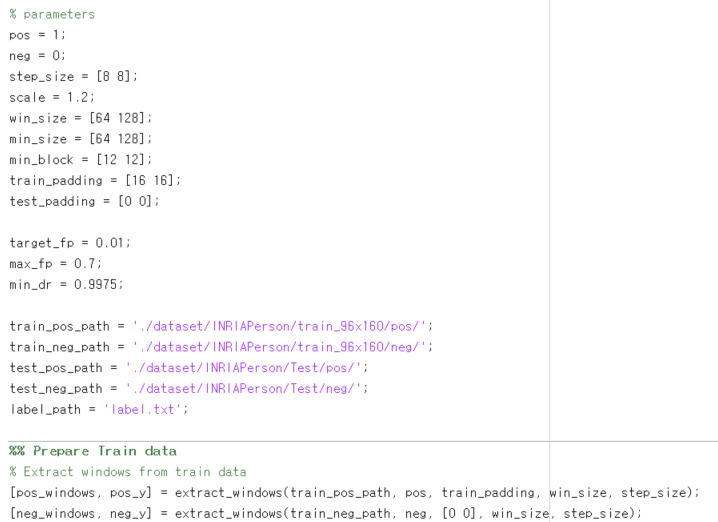
2018-10457 Junseo Lee

**1. Human detection algorithm using HOG and linear SVM**



- Parameter setting

First, we need pedestrian image dataset for training and testing the classifier. I converted png images to jpg images in ‘dataset/INRIAPerson/train\_64x128\_H96/{pos, neg}’ due to PNG library error(neg was referenced using symbolic links). It can be changed as INIRAPerson/train\_64x128\_H96/pos/, INIRAPerson/Train/neg/.



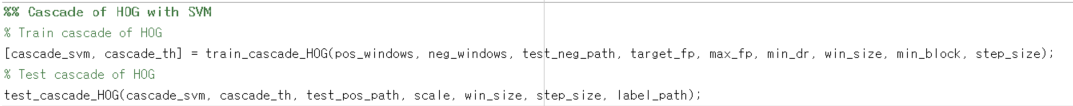
- Brief description of algorithm

In this project, the goal is to find the windows that show pedestrian. The smallest window size is 64x128 and by using multiscale detection, we can find various size of pedestrians. First, the HOG features of windows should be extracted to train the linear SVM classifier. Using ‘extractHOGFeatures(I)’ with default cell size(8x8) and default block size(2x2), HOGs of windows are extracted and stored in X\_train and true decision label is stored in y\_train. extract\_windows() function extracts all windows from pos. In case of neg, extract\_windows() chooses 10 random windows for each image. After preparing HOG features and true decision label, I trained linear SVM model using fitcsvm() function. Trained SVM model is tested through detectMultiScale() function. I referred to the parameters of C++ gpu::HOGDescriptor::detectMultiScale. At refinement stage, it removes duplicated detections for the final output using nms() function. nms() function is copied from Tomasz Malisiewicz’s github repository, <https://github.com/quantombone/exemplarsvm>, and I modified some part of code according to my intentions. It removes low score detection windows. In addition, after training the first SVM model the FP windows from training negative images are added to train final SVM model.

- Optimization

After training the SVM model using windows of pos/randomly sampled neg, I added false positive windows(train\_with\_hard\_example), which are extracted from neg’s multiscale images, to training dataset. The final SVM model is trained using original dataset + FP windows. For more fast processing, I used parfor(parallel for loop) for extractHOGFeatures of test windows. It was very time-consuming process before optimization. When I test the SVM model and pause to see how much it progressed, it was at extractHOGFeatures in nine out of ten times. To reduce the number of function call, I first concatenate the multiscale windows(extract\_multiscale\_windows) and call the function only once. It was much faster than before. In addition, I only selected windows the scores are higher than 1.5.

**2. Cascade of HOG algorithm (AdaBoost)**



- Parameter setting

For AdaBoost training algorithm, target FP rate for the final cascade classifier, maximum FP rate & minimum detection rate for each level of cascade are selected. In each round, n\_sample is the number of randomly chosen blocks among blocks of different sizes(12x12 ~ 64x128), locations(step size = 4, 6, 8) and aspect ratios(1:1, 1:2, 2:1). I used the parameters proposed in the paper [3].

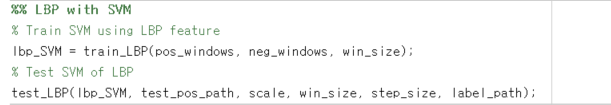
- Brief description of algorithm

train\_cascade\_HOG() function exactly do the same as the algorithm proposed in the paper [3], 3p. I trained and compared at the same time for choosing the best SVM, instead of storing all 250 trained linear SVMs and choosing the best after that. make\_block\_set() function makes block set consists of 7000+ blocks of different sizes, locations and aspect ratios. 250 blocks for each round are sampled randomly from the block set. After each level execution is finished, false negative windows are removed for faster computation of next cascade level. The total number of cascade levels was 15.

- Optimization

For more optimization, the weak classifiers from each round were stored in the form of beta and bias of SVM model due to easy calculation of distance(score) for updating threshold and less memory usage. The cascade SVM classifier stores the name of block(size, position), beta and bias of weak classifier, and threshold of weak classifier for holding minimum detection rate per cascade level. The cascade threshold holds the thresholds of the final strong classifiers of each level. Also, I added negative samples each round for more accurate training.

**3. Human detection algorithm using LBP and linear SVM**



- Parameter setting

All parameters except cell size of feature are same. The cell size is set to 16x16, which is four times larger than the HOG feature extraction.

- Brief description of algorithm

It is just as same as the HOG with linear SVM. The only one difference is that we use the LBP(local binary patterns) feature instead of HOG.

- Optimization

I also used parfor for extract LBP features of test positive windows. In my case, how to extract the features quickly was the main problem of this project. Also, I only selected windows the scores are higher than 1.5.

**4. Result**

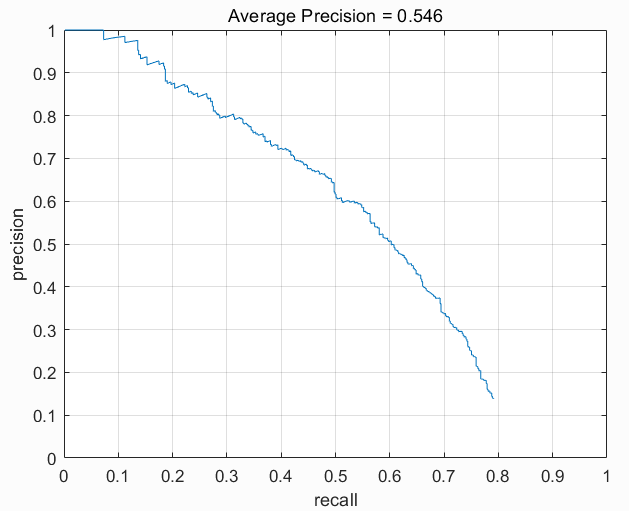
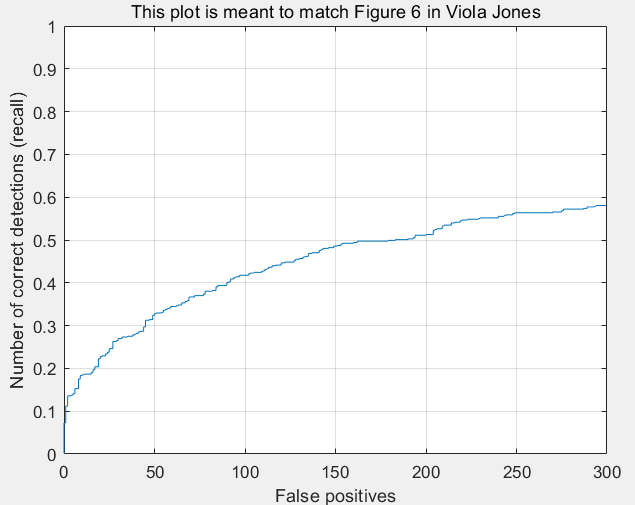
 

Figure 1, 2. Result of using HOG (no cascade)

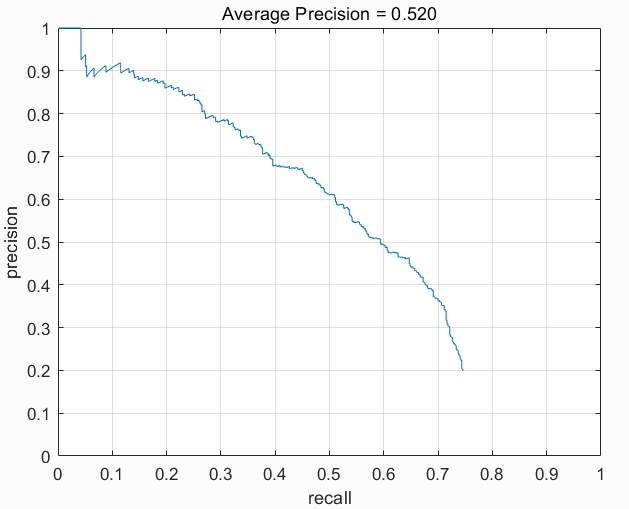
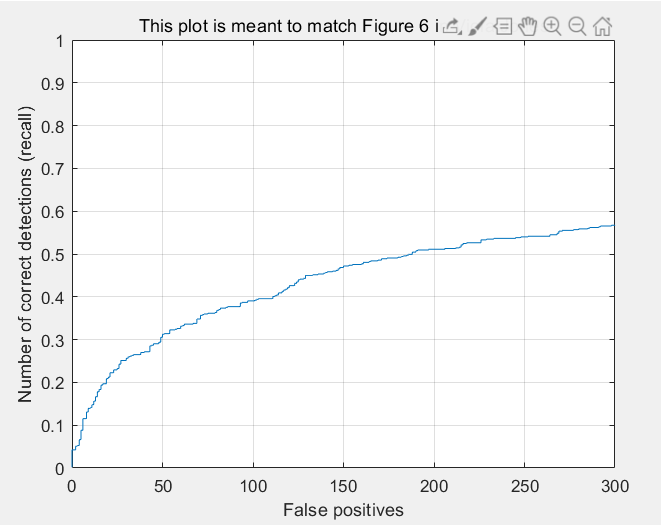
 

Figure 3, 4. Result of using cascade of HOG

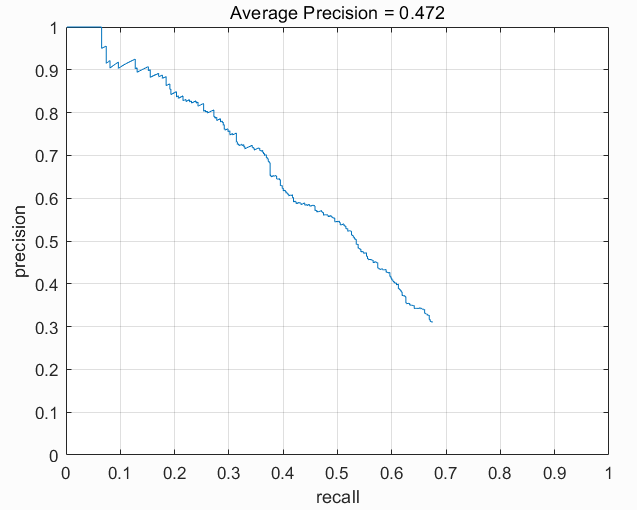
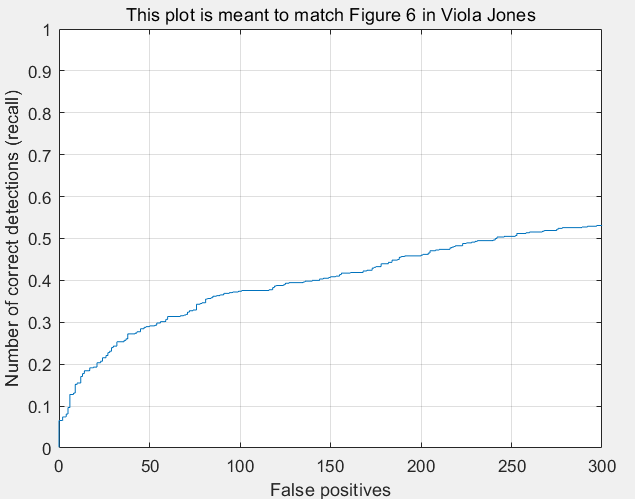
 

Figure 5, 6. Result of using LBP

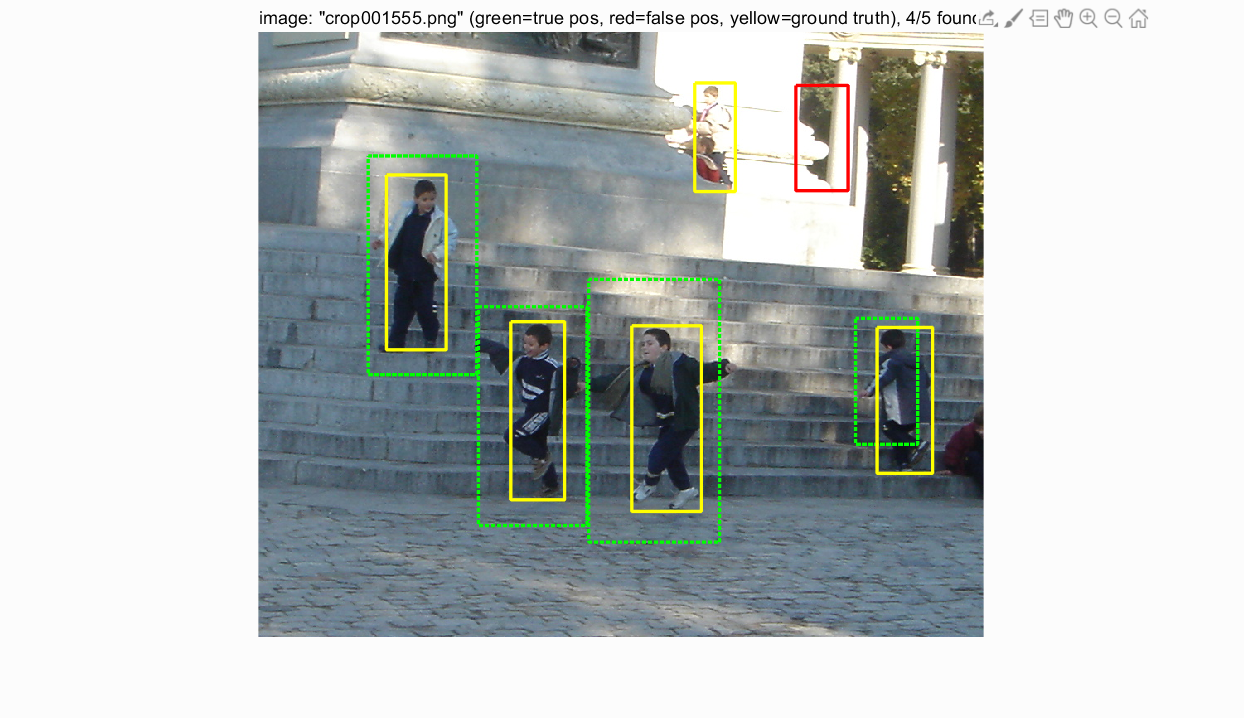


Figure 7. Example detection result of HOG

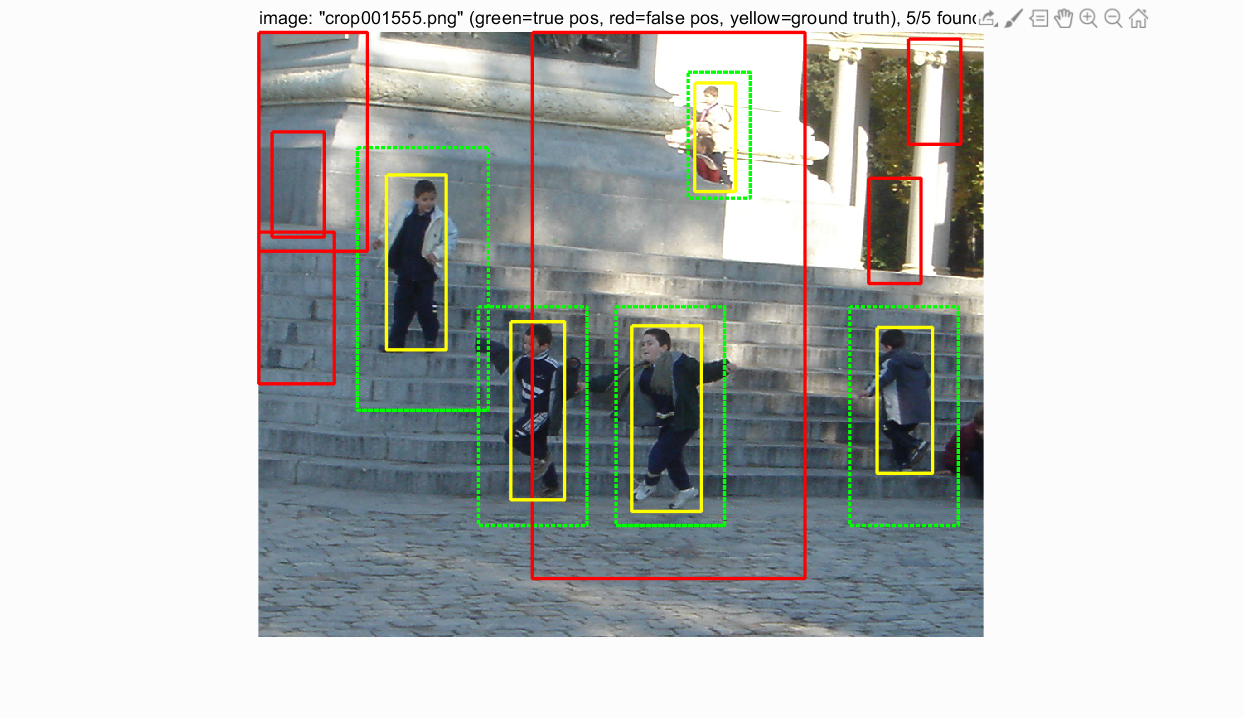


Figure 8. Example detection result of cascade of HOG

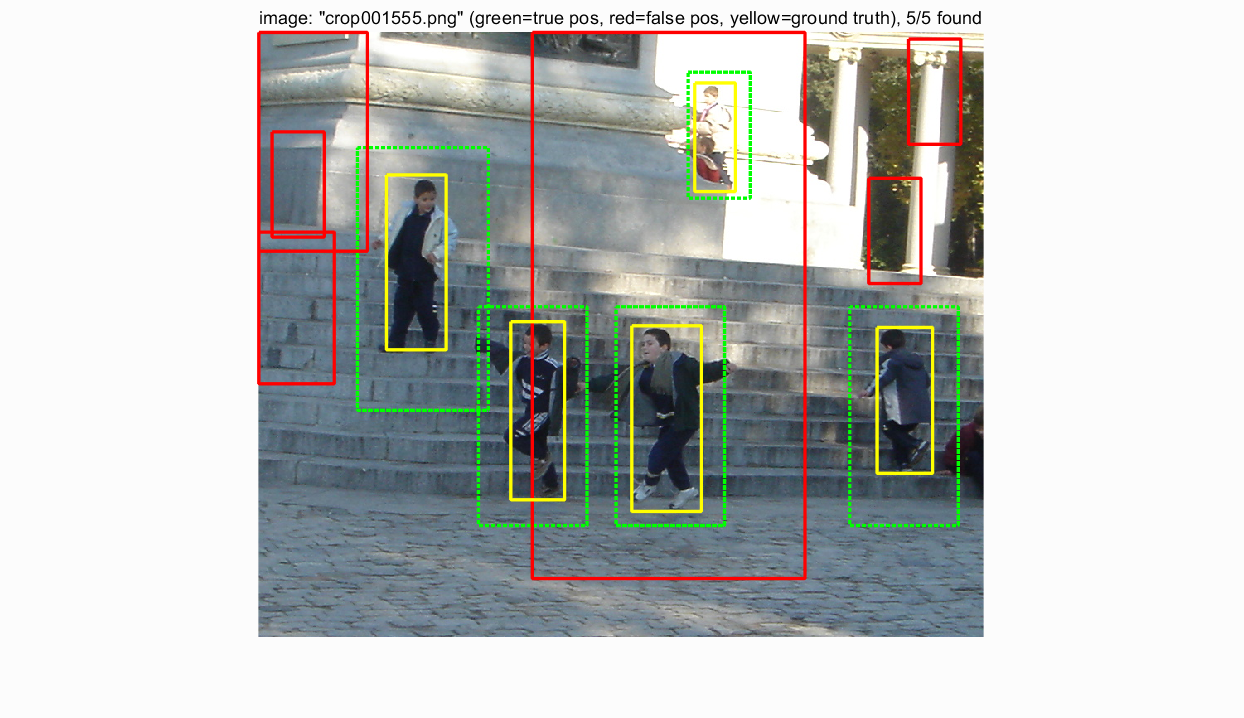


Figure 9. Example detection result of LBP

The average precision results are quite similar when using HOG features, which are better for precision than using LBP features. The reason of why the result of using LBP features shows lower average precision than using HOG would be that I did not added negative samples when training SVM model of LBP. Adding hard examples to train dataset was good optimization as we can see. Some detection result images are in ‘visualization’ folder. The models tend not to detect the pedestrians whose color of clothes is quite similar to background. In addition, some results show that all models tend to predict pillars or power poles as a human, so if we add more train data that can separate human and non-human but standing object the result would be better.

**5. Lesson Learned**

Even though understanding full algorithm of human detection using HOG, cascade of HOG, and LBP features was tough, extracting features and training (especially for cascade version..) were a lot more time-consuming in this project. The results were not satisfied for me compared to the time I spent, but for this class I learned many things and it would be helpful for my later research related to computer vision.

**References**

[1] Paul Viola and Michael Jones, Rapid Object Detection using a Boosted Cascade of Simple Features, CVPR 2001

[2] Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR 2005

[3] Qiang Zhu et al., Fast Human Detection Using a Cascade of Histograms of Oriented Gradients, CVPR2006

[4] Y. Mu, S. Yan, Y. Liu, T. Huang, B. Zhou, Discriminative local binary patterns for human detection in personal album, CVPR 2008