Introduction to Computer Vision Assignment #4

전기정보공학부2018-14643 김민성

I . 과제설명

INRIA Person dataset을 이용하여 classifier를 training 한 후 sliding window-based object detection algorithm을 이용하여 사진 속에서 사람 이미지를 detect한다.

II. Implenmention detail

1. Human detection algorithm using HOG and SVM

```
%% ----------------  dir setting ----------------------
%directories
images_train_pos_dir = 'INRIAPerson/train_64x128_H96/pos/';
images_train_neg_dir = 'INRIAPerson/Train/neg/';
images_test_pos_dir = 'INRIAPerson/Test/pos/';
annos_test_pos_dir = 'INRIAPerson/Test/annotations/';
%% ------------------------ training ------------------------

h = 128;
w = 64;
[pos_list, npos] = getimglist_for_train(images_train_pos_dir,h,w,1);
[neg_list, nneg] = getimglist_for_train(images_train_neg_dir,h,w,0);
```

먼저 INRIA data set에서 데이터를 불러 드린 후 getimglist_for_train 함수를 이용하여 positive train set과 negative train set을 얻었다. getimglist_for_train은 아래와 같다.

```
function [imglist, n] = getimglist_for_train(file_dir,h,w,is_pos)
files = dir(file_dir);
files = files(3:end);
if(is_pos == 1)
    n = size(files,1);
    imglist = cell(n,1);
    for i = 1:n
      img = imread([file_dir,files(i).name]);
        img = rgb2gray(img);
        r = centerCropWindow2d(size(img),[h w]);
        img = imcrop(img,r);
        imglist{i} = img;
    end
else
    n = size(files,1)*10;
    ni = 1;
    imglist = cell(n,1);
    for j = 1:size(files,1)
        img = imread([file_dir,files(j).name]);
        img = rgb2gray(img);
        for i =1:10
           %random crop
           r = randomCropWindow2d(size(img),[h w]);
           img = imcrop(img,r);
           imglist{ni} = img;
           ni = ni + 1;
        end
    end
end
end
```

이는 파일 경로와 window size를 input으로 받아서 positive train set일 경우에는 파일경로에 있는 128*64의 사람 이미지들을 추출하여 return하고 negative train set일 경우에는 이미지를 랜덤하게 잘라 한 파일당 10조각의 128*64 이미지를 return한다.

```
%% positive sample's HOG feature
pos_feature_list = getHOGs(pos_list,[8 8],[2 2]);
%% negative sample's HOG feature
neg_feature_list = getHOGs(neg_list,[8 8],[2 2]);
```

이후 HOG feature들을 추출하였다.

getHOGs 함수는 아래와 같다.

```
function features = getHOGs(imglist,cellsize,blocksize)
n = size(imglist,1);
feature_size = size(extractHOGFeatures(imglist{1},'CellSize',cellsize, 'BlockSize', blocksize),2);
features = zeros(n,feature_size);
for i = 1:n
    features(i,:) = extractHOGFeatures(imglist{i},'CellSize',cellsize, 'BlockSize', blocksize);
end
end
```

이미지에서 extractHOGFeatures함수를 통해 HOG feature를 추출한다.

```
%% Train svm
%%dataset setting
X = [pos_feature_list; neg_feature_list];
y = zeros(npos+nneg,1);
y(1:npos) = 1;

%%fitting
svm_model = fitcsvm(X,y,'BoxConstraint',1);
%%
w = svm_model.Beta;
b = svm_model.Bias;
```

이후 matlab의 svm라이브러리를 이용하여 classfier를 학습시켜주었다.

```
%% ------------------- Testing ------------------------
%ground truth 정보를 담은 파일을 생성하였다.
write_gt_from_anno(annos_test_pos_dir);
```

이후 annotatinos의 파일을 읽어서 testset에서 ground_truth값들을 기록하였다.

write_gt_from_anno 함수는 아래와 같다.

```
function write_gt_from_anno(annodir)
 delete('gt.txt');
 files = dir(annodir);
 files = files(3:end);
 n = size(files,1);
 fileID = fopen('gt.txt','w');
for i = 1:n
     record = info_from_anno([annodir,files(i).name]);
     nobj = length(record.objects);
     for j = 1:nobj
         name = record.imgname;
         bbox = record.objects(j).bbox;
         fprintf(fileID, '%s %d %d %d %d₩n', name(10:end) , bbox);
     end
 end
 fclose(fileID);
 end
```

Info_from_anno함수는 다음과 같다.

```
function record = info_from_anno(filename)
fd = fopen(filename,'rt');

record.imgname = [];
record.objects.bbox = [];
namestr = 'Image filename : %q';
bboxstr = 'Bounding box for object %d "PASperson" (Xmin, Ymin) - (Xmax, Ymax) : (%d, %d) - (%d, %d)';
notEOF=1;
while(notEOF)
    line=fgetl(fd);
    notEOF=ischar(line);
    if(strncmp(line,namestr,14))
        name = strread(line,namestr);
        record.imgname = char(name);
    end
    if(strncmp(line,bboxstr,8))
        [obj,xmin,ymin,xmax,ymax] = strread(line,bboxstr);
        record.objects(obj).bbox = [xmin ymin xmax ymax];
    end
end
fclose(fd);
end
```

annotations 폴더에서 파일이름과 bbox를 읽어서 이를 gt.txt에 기록한다.

```
%% detect
tic
[test_bboxes, test_confidences, test_image_ids] = detect(w,b,images_test_pos_dir, 32, [128 64], 10, 0.8, 1);
toc
```

그리고 detect함수를 이용하여 test data set에서 사람이미지를 detect해보았다.

detect함수는 아래와 같다.

```
function [test_bboxes, test_confidences, test_image_ids] = detect(w,b,test_dir, stride, window_size, ...
    n_scale, reduce_ratio, ftype)
%initialize
test_bboxes = [];
test_confidences = [];
test_image_ids = cell(0,1);


testfiles = dir(test_dir);
testfiles = testfiles(3:end);
n = size(testfiles,1);
for i = 1:n
    if rem(i,20) == 0
        i/n
    end
    test_img = rgb2gray(imread([test_dir,testfiles(i).name]));
    [samples, bboxes] = get_sliding_window_samples(test_img,  stride, window_size,n_scale, reduce_ratio, ftype);
    all_confidences = samples*w+b;

    cur_test_bboxes = [];
    cur_test_confidences = [];
    cur_test_image_ids = cell(0,1);

    for j = 1:size(all_confidences,1)
        if(all_confidences(j) > 0)
            cur_test_confidences = [cur_test_confidences; all_confidences(j)];
            cur_test_bboxes = [cur_test_bboxes; bboxes(j,:)];
            cur_test_image_ids = [cur_test_image_ids; [testfiles(i).name]];
        end
    end
```

```
    if (~isempty(cur_test_confidences))
        is_maximum = nmss(cur_test_bboxes, cur_test_confidences);
        cur_test_confidences = cur_test_confidences(is_maximum,:);
        cur_test_bboxes = cur_test_bboxes(is_maximum,:);
        cur_test_image_ids = cur_test_image_ids(is_maximum,:);

        test_bboxes = [test_bboxes; cur_test_bboxes];
        test_confidences = [test_confidences; cur_test_confidences];
        test_image_ids = [test_image_ids; cur_test_image_ids];
    end
end
end
```

먼저 test data set의 사진들을 읽어드린 후 각 이미지를 get_sliding_window_samples 함수를 통해 window size로 잘게 쪼개진 이미지들의 특징(HOG or LBF or image)들을 얻는다. 그리고 svm model의 parameter들을 이용하여 쪼개진 특징들을 classify한다. 그리고 positive로 판정한 것들을 모은다. 그후 nmss함수를 이용하여 duplicated detections들을 제거한다.

위에서 사용된 get_sliding_window_samples 함수는 아래와 같다.

```
function [samples, bboxes] = get_sliding_window_samples(test_img, stride, window_size, n_scale, reduce_ratio, ftype)
%test_img is graysclae
 % if ftype is 1 -> hog
 % if ftype is 2 -> lbf
 % if ftype is 3 -> jsut img
%initializing
 img_ho = size(test_img,1);
 img_wo = size(test_img,2);
 win_h = window_size(1);
 win_w = window_size(2);
 img_hi = img_ho;
 img_wi = img_wo;
 r = 1; %total reduce_ratio
 samples = [];
 if ftype ==3
     samples = {};
 end
 bboxes = [];
 test_img_i = test_img;
 %window의 스케일을 키워가며 반복
for i = 1:n_scale
     %window의 크기가 이미지보다 크면 종료
     img_hi = size(test_img_i,1);
     img_wi = size(test_img_i,2);
     if(img_hi < win_h || img_wi < win_w)
         break
     end

     rend = img_hi - win_h +1;
     cend = img_wi - win_w + 1;
     sizeofsamples_i = ceil(rend/stride)*ceil(cend/stride);
```

```
    if ftype==1
    samples_i = zeros(sizeofsamples_i,3780);
    end
    if ftype==2
    samples_i = zeros(sizeofsamples_i,6195);
    end
    if ftype==3
    samples_i = cell(sizeofsamples_i,1);
    end
    bboxes_i = zeros(sizeofsamples_i,4);
    n = 1;
    for ri = 1:stride:rend
        for ci = 1:stride:cend
            sample = test_img_i(ri:ri+win_h-1, ci:ci+win_w-1);
            bbox = [round((ci)/r) round((ri)/r) round((ci+win_w-2)/r) round((ri+win_h-2)/r)];
            if ftype == 1
            sample = extractHOGFeatures(sample);
            samples_i(n,:) = sample;
            end
            if ftype == 2
            sample = extractLBPFeatures(sample,'CellSize',[6 12]);
            samples_i(n,:) = sample;
            end
            if ftype == 3
            samples_i{n} = sample;
            end
            bboxes_i(n,:) = bbox;
            n = n + 1;
        end
    end

    samples = [samples; samples_i];
    bboxes = [bboxes; bboxes_i];
    r = r*reduce_ratio;
    test_img_i = imresize(test_img_i,reduce_ratio);
end
end
```

이 함수는 test_img를 stride에 따라 이동하며 window size로 잘게 짜른다. 그리고 자른 이미지의 특징(HOG or LBP or image)을 추출하여 모은다. 그리고 test_img의 사이즈를 줄인다. 그리고 다시 stride에 따라 이동하며 window_size로 잘게 자른 후 특징을 모으는 과정을 반복한다.

non-maximum suppression step을 시행하는 nnms함수는 다음과 같다.

```
function [valid] = nmss(bboxes, confidences)
[confidences, ind] = sort(confidences, 'descend');
bboxes = bboxes(ind,:);
is_valid = ones(1,size(bboxes,1));
for i = 1:size(bboxes,1)
    bb1 = bboxes(i,:);
    for j=find(is_valid)
        if i<j
            bb2 = bboxes(j,:);
            %iou가 0.3이상이면 기각
            if(getIOU(bb1,bb2)>0.3)
                is_valid(1,j) = 0;
            end
        end
    end
end
rev(ind) = 1:size(bboxes,1);
valid = logical(is_valid(rev));
end

function f = getIOU(bb1, bb2)
bi=[max(bb1(1),bb2(1)); max(bb1(2),bb2(2)); min(bb1(3),bb2(3)); min(bb1(4),bb2(4))];
iw=bi(3)-bi(1)+1;
ih=bi(4)-bi(2)+1;
if(iw>0 && ih>0)
    a = (bb1(3)-bb1(1)+1)*(bb1(4)-bb1(2)+1)+(bb2(3)-bb2(1)+1)*(bb2(4)-bb2(2)+1);
    f = iw*ih/(a-iw*ih);
else
    f = 0;
end
end
```

bbox들을 입력 받아 이를 confidence에 대한 내림차순으로 정렬한 후 차례로 IOU를 계산하여 중복된 detection들을 제거하고 유효한 bbox들의 index를 return한다.

```
%% evaluate
[gt_ids, gt_bboxes, gt_isclaimed, tp, fp, duplicate_detections] = evaluate_detections(test_bboxes, test_confidences, test_image_ids, 'gt.txt',1);
%%
visualize_detections_by_image(test_bboxes, test_confidences, test_image_ids, tp, fp, images_test_pos_dir, 'gt.txt');
```

그리고 evaluate_detections와 visualize_detections_by_image함수를 이용하여 결과를 기록하였다.

2. cascade architecture

먼저 특징들을 정의할 block들의 bbox 목록을 getblocks_bbox함수를 통해 가져왔다.

```
%% ---------------------- cascade -------------------------------------------
blocks = getblocks_bbox();
```

getblocks_bbox() 함수는 아래와 같다.

```
function blocks = getblocks_bbox()
b=[];
for i = 12:2:64
    b=[b; i i;i 2*i];
end
for i = 12:2:32
    b=[b; 2*i i];
end
blocks=[];
n=size(b,1);
for i = 1:n
    if (mod(64-b(i,1), 4)== 0 && mod(128-b(i,2), 4)==0)
        sx = 1:4:64-b(i,1)+1;
        sy = 1:4:128-b(i,2)+1;
        [sp, n_sp] = getStratingPoint(sx,sy);
        for j=1:n_sp
            blocks = [blocks; sp(j,:) sp(j,:)+b(i,:)-[1 1]];
        end
    end
    if (mod(64-b(i,1), 6)==0 && mod(128-b(i,2), 6)==0)
        sx = 1:6:64-b(i,1)+1;
        sy = 1:6:128-b(i,2)+1;
        [sp, n_sp] = getStratingPoint(sx,sy);
        for j=1:n_sp
            blocks = [blocks; sp(j,:) sp(j,:)+b(i,:)-[1 1]];
        end
    end
end
blocks = unique(blocks,'rows');
end
function [sp, n] =getStratingPoint(sx,sy)
nx = size(sx,2);
ny = size(sy,2);
sx = repmat(sx,ny,1);
sx = reshape(sx,[],1);
sy = repmat(sy,1,nx);
sy = reshape(sy,[],1);
sp = [sx sy];
```

```
n=nx*ny;
end
```

먼저 block의 사이즈를 참고문헌[2]에서와 같이 12*12 부터 64*128의 사이즈에 달하는 다양한 크기의 block 사이즈들을 정의하였다. 이 block들을 4 또는 6 pixel씩 stride하여 특징들을 정의하였고 이에 대한 bbox들을 return 하였다. 총 6342개의 특징 구역의 bbox들을 얻었다.

```
%% training cascade
cascade_classifier = training_cascade_classifier(pos_list, neg_list, blocks, 0.01, 0.7, 0.9975);
```

그리고 training_cascade_classifier 함수를 이용하여 cascade_classifier를 training 하였다.

참고문헌[2]에서와 같이 f_max는 0.7 d_min은 0.9975로 하였고 F_target은 0.001로 하고 싶었지만 training할 충분한 시간이 없어 0.01로 설정하였다.

training_cascade_classifier 함수는 아래의 알고리즘을 따르게 구현하였다.

Algorithm 1 Training the cascade

Input: F_{target}: target overall false positive rate
f_{max}: maximum acceptable false positive rate per cascade level
d_{min}: minimum acceptable detection per cascade level
Pos: set of positive samples
Neg: set of negative samples

initialize: i = 0, D_i = 1.0, F_i = 1.0
loop $F_i > F_{target}$
$i = i + 1$
$f_i = 1.0$
loop$f_i > f_{max}$
1) train 250 (%5 at random) linear SVMs using Pos and Neg samples
2) add the best SVM into the strong classifier, update the weight in AdaBoost manner
3) evaluate Pos and Neg by current strong classifier
4) decrease threshold until d_{min} holds
5) compute f_i under this threshold
loop end
$F_{i+1} = F_i \times f_i$
$D_{i+1} = D_i \times d_{min}$
Empty set Neg
if $F_i > F_{target}$ then evaluate the current cascaded detector on the negative,i.e. non-human,images and add misclassified samples into set Neg.
loop end

Output: A i-levels cascade
each level has a boosted classifier of SVMs
Final training accuracy: F_i and D_i

Figure 1 Training the Cascade

- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \ldots, T$:
  1. Normalize the weights,
  $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
  so that w_t is a probability distribution.
  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$.
  3. Choose the classifier, h_t, with the lowest error ϵ_t.
  4. Update the weights:
  $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$
  where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- The final strong classifier is:
$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$
where $\alpha_t = \log \frac{1}{\beta_t}$

Figure 2 Adaboost

Figure1 의 알고리즘에 따라 구현하였고 adaboost manner는 figure 2에서 확인할 수 있다.

```
function cascade_classifier =training_cascade_classifier(pos_list, neg_list, blocks,F_target, f_max, d_min)

%initialize
cascade_classifier = {};
D_i = 1;
F_i = 1;
i = 0;

while(F_i>F_target)
    i = i+1;
    fprintf('Level %d: \n',i);
    f_i = 1;

    %adaboost
    npos = size(pos_list,1);
    nneg = size(neg_list,1);
    nall = npos + nneg;
    w_t = [ones(npos,1)./(2*npos); ones(nneg,1)./(2*nneg)];

    strong_classifier = [];
    alpha_list = [];
    strong_confiednce = zeros(nall,1);
    img_list = [pos_list; neg_list];
```

adaboost manner에 따라 weak classifier들을 합친 strong classifier가 cascade 구조의 한 level이 되는데 , 먼저 F_i, D_i, weight를 초기화 시켜주고 F_i가 F_target보다 낮아질 때까지 cascade level을 올리게 하였다.

```
    while (f_i > f_max)
        w_t = w_t ./ sum(w_t);
        now_blocks = datasample(blocks,250,1,'Replace',false);
        lowest_error = inf;
        % 250 개의 random으로 고른 block 에 의한 svm
        for j = 1:250
            bbox = now_blocks(j,:);
            cellsize = [(bbox(4)-bbox(2)+1)/2 (bbox(3)-bbox(1)+1)/2];
            windowsample_list = cell(nall,1);

            for ni = 1:nall
                windowsample_list{ni} = img_list{ni}(bbox(2):bbox(4),bbox(1):bbox(3));
            end
            X = getHOGs(windowsample_list,cellsize,[2 2]);
            y = zeros(nall,1);
            y(1:npos) = 1;
            svm_model = fitcsvm(X,y);
            w = svm_model.Beta;
            b = svm_model.Bias;
            cur_confidence = X * w + b;
```

그리고 f_i가 0.7보다 낮아 질때까지 blocks 에서 random하게 250개를 고른 것 중 가장 낮은 error를 갖는 block에 대한 weak_classfier를 adaboost manner에 따라 strong_classfier에 추가하였다.

```
        % 모든 가능한 weak_threshold별로 최적의 weak_threshold 검색 후 error 측정
        error_list_temp = zeros(npos,1);
        for nw = 1:npos
            weak_threshold = cur_confidence(nw);
            predicted = (cur_confidence >= weak_threshold);
            error_list_temp(nw,1) = sum(w_t.*abs(predicted-y));
        end
        [error, idx] = min(error_list_temp);
        best_weak_threshold = cur_confidence(idx);
        if error < lowest_error
            lowest_error = error;
            weak_classifier =  [bbox'; w; b; best_weak_threshold];
            predicted_now = predicted;
        end
    end
    beta_t = lowest_error / (1 - lowest_error);
    ei = abs(predicted_now-y);
    w_t = w_t .* (beta_t .^ (1 - ei));
    alpha_t = log(1/beta_t);
    strong_confiednce = strong_confiednce + alpha_t * predicted_now;

    min_detect_pos = round(d_min * npos);
    temp = maxk(strong_confiednce(1:npos), min_detect_pos);
    threshold = temp(end);

    strong_predicted = (strong_confiednce >= threshold);

    d_i = sum(strong_predicted(1:npos))/npos;
    f_i = sum(strong_predicted(npos+1:end)) / nneg;

    strong_classifier = [strong_classifier weak_classifier];
    alpha_list = [alpha_list alpha_t];
    fprintf('now f_i -> %f₩n', f_i);
end
```

Weak_classifier를 찾는 과정에서 가능한 weak_threshold를 모두 적용해보며 error를 계산하며 weak_classifier를 찾았다. 그리고 이를 strong_classfier에 추가해 adaboost manner에 따라 strong_confidence들을 계산하고 d_min 이 0.9975가 되도록 strong_threshold를 정해 주었다. 이를 f_i가 f_max보다 작아질 때까지 반복하였다.

```
    D_i = D_i * d_i;
    F_i = F_i * f_i;

    fprintf('now Fi ->  %f₩n', F_i);
    cascade_classifier{i} = struct('strong_cf',strong_classifier,'alpha',alpha_list,'strong_Th',threshold);

    % set neg 을 비우고 다시 채운다.
    neg_list = {};
    if F_i > F_target
        neg_list = getimglist_for_train('INRIAPerson/Train/neg/',128,64,0);
        [neg_list, ~, ~] = detect_cascade_samples(samples, bboxes, cascade_classifier);
    end

end
```

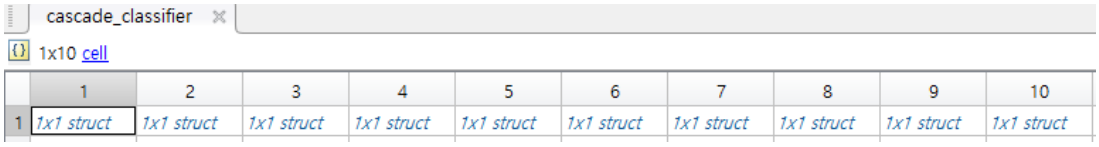
그리고 f_i가 f_max보다 작아지면 현재 level의 strong_classfier를 cascade_classifier{i}의 구조에 담아 놓는다. 다음 cascade level로 넘어가기 전에 neg set을 비워준 후 다시 neg set을 만들어 이를 현재까지의 cascade classifier로 detect하여 positive로 detect한 것들만 모아 neg set이 되

게 하였다. Cascade classifier로 samples를 detect하는 detect_cascade_samples함수는 아래와 같다.

```
function [samples, bboxes, confidences] = detect_cascade_samples(samples, bboxes, cascade_classifier)

if ~exist('bboxes','var'), bboxes = 0; end
num_levels = size(cascade_classifier,1);
num_samples = size(samples,1);
for i = 1:num_levels

    strong_classifier = cascade_classifier{i}.strong_cf;
    block_bboxes =  strong_classifier(1:4,:);
    w_list = strong_classifier(5:40,:);
    b_list = strong_classifier(41,:);
    weak_threshold = strong_classifier(42,:);
    alpha = cascade_classifier{i}.alpha;
    threshold = cascade_classifier{i}.strong_Th;
    confidences = 0;

    num_weak_cf = size(w_list,2);

    for j = 1:num_weak_cf
        w = w_list(:,j);
        b = b_list(:,j);
        weak_Th = weak_threshold(j);
        alpha_t = alpha(j);
        bbox = block_bboxes(:,j);
        cellsize = [(bbox(4)-bbox(2)+1)/2 (bbox(3)-bbox(1)+1)/2];
        cut_samples = cell(num_samples,1);
        for ns = 1:num_samples
            cut_samples{ns} = samples{ns}(bbox(2):bbox(4),bbox(1):bbox(3));
        end
        X = getHOGs(cut_samples,cellsize,[2 2]);
        weak_confidence = X * w + b;
        weak_predicted = (weak_confidence >=  weak_Th);
        confidences = confidences + weak_predicted * alpha_t;
    end
    strong_predicted = (confidences >= threshold);
    samples = samples(strong_predicted);
    bboxes = bboxes(strong_predicted,:);
    confidences = confidences(strong_predicted);
```

입력받은 Samples들을 입력받은 cascade_classifier를 이용하여 detect한다. 그리고 positive로 detect한 sample들과 이에 대한 bbox들 그리고 confidece들을 return한다.

이렇게 training 시킨 cascade_classifier는 10개의 level을 가졌다.



이를 이용하여 test set에서 detect_cascade함수를 이용하여 detect하였다.

```
function [test_bboxes, test_confidences, test_image_ids] = detect_cascade(cascade_classifier, test_dir, stride, window_size,n_scale, reduce_ratio)
%initialize
test_bboxes = [];
test_confidences = [];
test_image_ids = cell(0,1);


testfiles = dir(test_dir);
testfiles = testfiles(3:end);
n = size(testfiles,1);
n = 1;
for i = 1:n
    if rem(i,20) == 0
        i/n
    end
    test_img = rgb2gray(imread([test_dir,testfiles(i).name]));
    [samples, bboxes] = get_sliding_window_samples(test_img,  stride, window_size,n_scale, reduce_ratio, 3);
    [~, bboxes, confidences] = detect_cascade_samples(samples, bboxes, cascade_classifier);
    cur_test_bboxes = bboxes;
    is_maximum = nmss(cur_test_bboxes, confidences);
    cur_test_confidences = confidences(is_maximum,:);
    cur_test_bboxes = cur_test_bboxes(is_maximum,:);
    n_c = size(cur_test_bboxes,1);
    cur_test_image_ids = cell(n_c,1);
    for c = 1:n_c
    cur_test_image_ids{c} = [testfiles(i).name];
    end
    test_bboxes = [test_bboxes; cur_test_bboxes];
    test_confidences = [test_confidences; cur_test_confidences];
    test_image_ids = [test_image_ids; cur_test_image_ids];

end
end
```

Test set을 훑으면서 test 이미지를 get_sliding_window_samples 함수를 통해 multiwindow size로 잘린 image samples을 얻고 이 samples를 detect_cascade_samples함수를 통해 detect하고 nmss 함수를 통해 중복된 detection들을 삭제하고 남은 sample에 대한 정보들을 return한다.

```
%% evaluate
[gt_ids, gt_bboxes, gt_isclaimed, tp, fp, duplicate_detections] = evaluate_detections(test_bboxes, test_confidences, test_image_ids, 'gt.txt',1);
%%
visualize_detections_by_image(test_bboxes, test_confidences, test_image_ids, tp, fp, images_test_pos_dir, 'gt.txt');
```

그리고 evaluate_detections와 visualize-detections_by_image함수를 이용하여 결과를 기록하였다.

3. LBP feature

HOG feature와 svm을 이용한 detection의 과정을 거의 따른다. 다만 특징을 HOG가 아닌 LBP를 이용한다.

```
%% ------------------------------------LBP-------------------------------------------
pos_feature_list = getLBPs(pos_list);
neg_feature_list = getLBPs(neg_list);
%% Train svm
%%dataset setting
X = [pos_feature_list; neg_feature_list];
y = zeros(npos+nneg,1);
y(1:npos) = 1;

%%fitting
svm_model = fitcsvm(X,y,'BoxConstraint',1);
%%
w = svm_model.Beta;
b = svm_model.Bias;

%% detect
tic
[test_bboxes, test_confidences, test_image_ids] = detect(w,b,images_test_pos_dir, 8, [128 64], 10, 0.8,2);
toc
%% evaluate
[gt_ids, gt_bboxes, gt_isclaimed, tp, fp, duplicate_detections] = evaluate_detections(test_bboxes, ...
    test_confidences, test_image_ids, 'gt.txt',1);
%%
visualize_detections_by_image(test_bboxes, test_confidences, test_image_ids, tp, fp, images_test_pos_dir, 'gt.txt');
```

이미지에서 LBP특징을 뽑아내는 getLBPs함수는 다음과 같다.

```
function features = getLBPs(imglist)
n = size(imglist,1);
feature_size = size(extractLBPFeatures(imglist{1},'CellSize',[6 12]),2);
features = zeros(n,feature_size);
for i = 1:n
    features(i,:) = extractLBPFeatures(imglist{i},'CellSize',[6 12]);
end
end
```

4. Bagging HOG svm and LBP svm

HOG에 대한 svm과 LBP에 대한 svm에 대한 confidence들을 평균을 내어 detect해 보았다.

```
%% 추가 구현 ----------------------LBF + HOG---------------------------------
pos_feature_list_H = getHOGs(pos_list,[8 8],[2 2]);
neg_feature_list_H = getHOGs(neg_list,[8 8],[2 2]);
pos_feature_list_L = getLBPs(pos_list);
neg_feature_list_L = getLBPs(neg_list);
```

```
%% svm_H
%%dataset setting
X = [pos_feature_list_H; neg_feature_list_H];
y = zeros(npos+nneg,1);
y(1:npos) = 1;
svm_model_H = fitcsvm(X,y,'BoxConstraint',1);
```

```
%% svm_L
%%dataset setting
X = [pos_feature_list_L; neg_feature_list_L];
y = zeros(npos+nneg,1);
y(1:npos) = 1;
svm_model_L = fitcsvm(X,y,'BoxConstraint',1);
```

```
%%
w_H = svm_model_H.Beta;
b_H = svm_model_H.Bias;
w_L = svm_model_L.Beta;
b_L = svm_model_L.Bias;
```

```
%%
tic
[test_bboxes, test_confidences, test_image_ids] = detect_myextra(w_H,b_H,w_L,b_L,images_test_pos_dir, 8, [128 64], 10, 0.8);
toc
```

```
%% evaluate
[gt_ids, gt_bboxes, gt_isclaimed, tp, fp, duplicate_detections] = evaluate_detections(test_bboxes, ...
    test_confidences, test_image_ids, 'gt.txt',1);
```

```
%%
visualize_detections_by_image(test_bboxes, test_confidences, test_image_ids, tp, fp, images_test_pos_dir, 'gt.txt');
```

Detect할 때 사용된 detect_myextra함수는 아래와 같다.

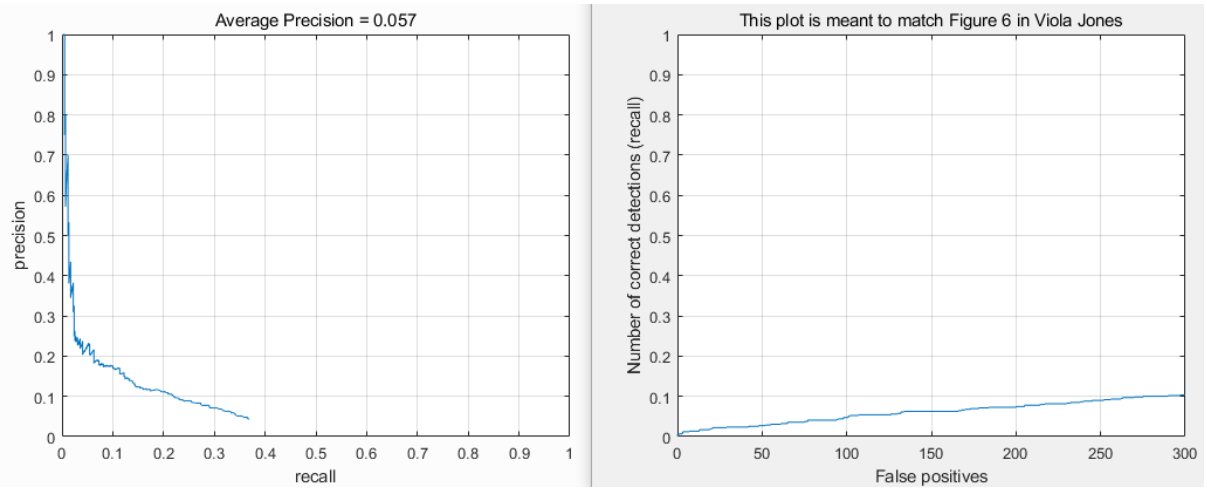
```
function [test_bboxes, test_confidences, test_image_ids] = detect_myextra(w_H,b_H,w_L,b_L,test_dir, stride, window_size, ...
    n_scale, reduce_ratio)
%initialize
test_bboxes = [];
test_confidences = [];
test_image_ids = cell(0,1);
testfiles = dir(test_dir);
testfiles = testfiles(3:end);
n = size(testfiles,1);
for i = 1:n
    if rem(i,20) == 0
        i/n
    end
    test_img = rgb2gray(imread([test_dir,testfiles(i).name]));
    [samples_H,samples_L,bboxes_H, bboxes_L] = get_sliding_window_samples_myextra(test_img, stride, window_size, n_scale, reduce_ratio);
    all_confidences = (samples_L*w_L+b_L+samples_H*w_H+b_H)/2;
    cur_test_bboxes = [];
    cur_test_confidences = [];
    cur_test_image_ids = cell(0,1);
    for j = 1:size(all_confidences,1)
        if(all_confidences(j) > 0)
            cur_test_confidences = [cur_test_confidences; all_confidences(j)];
            cur_test_bboxes = [cur_test_bboxes; bboxes_H(j,:)];
            cur_test_image_ids = [cur_test_image_ids; [testfiles(i).name]];
        end
    end
    if (~isempty(cur_test_confidences))
        is_maximum = nmss(cur_test_bboxes, cur_test_confidences);
        cur_test_confidences = cur_test_confidences(is_maximum,:);
        cur_test_bboxes = cur_test_bboxes(is_maximum,:);
        cur_test_image_ids = cur_test_image_ids(is_maximum,:);

        test_bboxes = [test_bboxes; cur_test_bboxes];
        test_confidences = [test_confidences; cur_test_confidences];
        test_image_ids = [test_image_ids; cur_test_image_ids];
    end
```

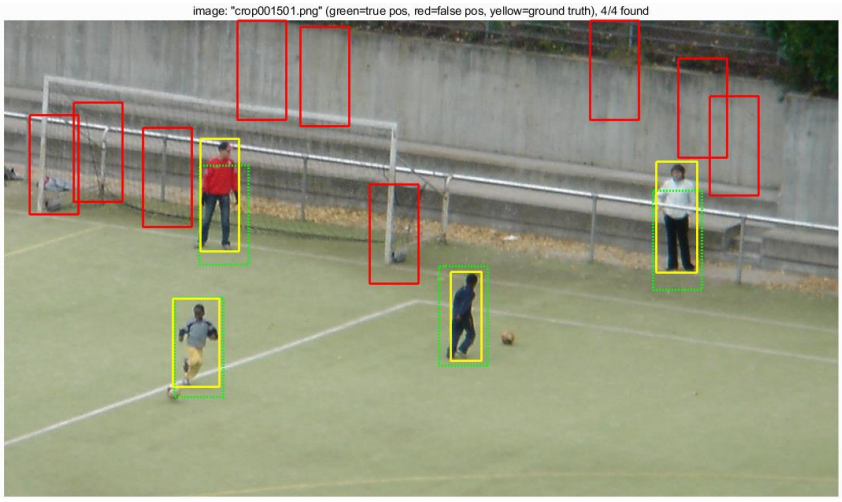
위에서의 detect함수와 달리 hog와 lbp 특징을 모두 뽑아낸 다음, 각각 svm을 실행시킨후 confidence 값을 평균 내는 것을 볼 수 있다.

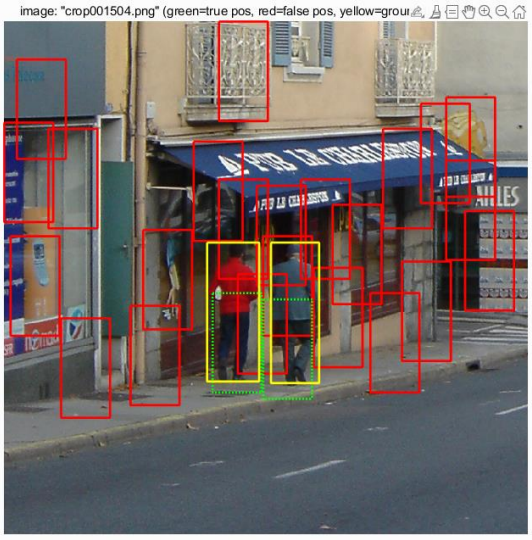
Hog와 lbp 특징은 get_sliding_window_samples_myextra함수에서 뽑아냈는데 이는 위 get_sliding_window_samples와 같이 동작하고 단지 samples에서 hog와 lbp 특징을 모두 뽑아낸다.

IV. RESULTS

1. svm with HOG(simgle scale)

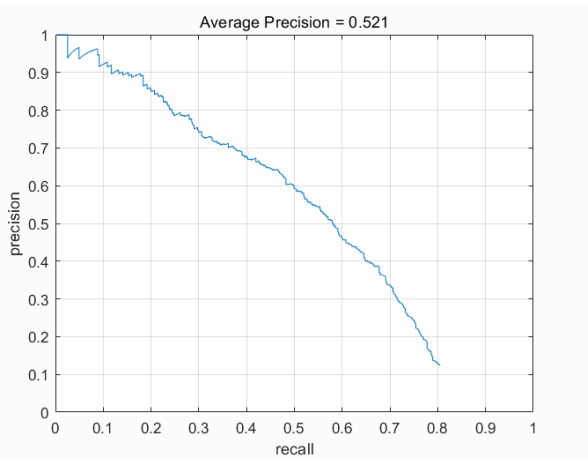


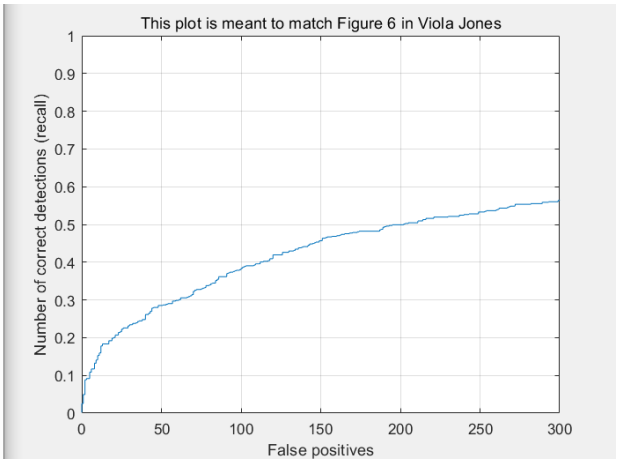


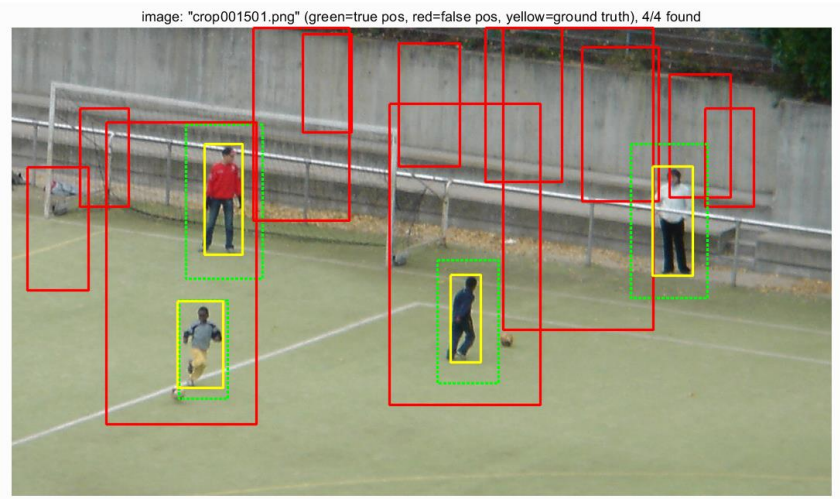


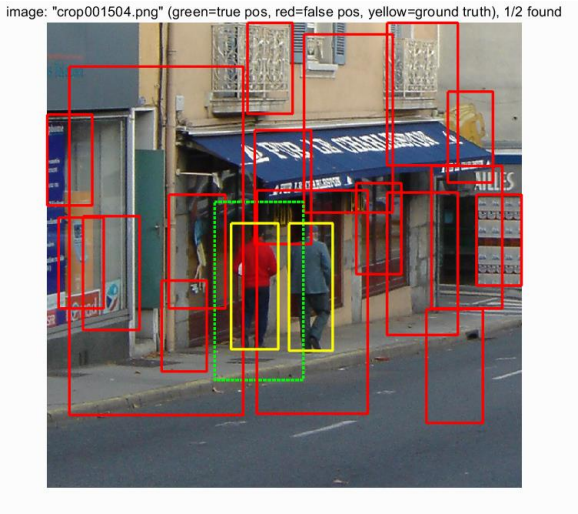
test data set 전체를 detect하는데 걸린 시간: 1117.367170초

2. svm with HOG(multiscale)



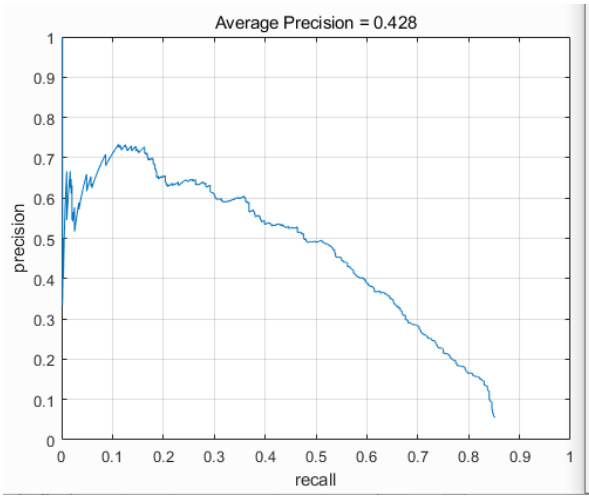


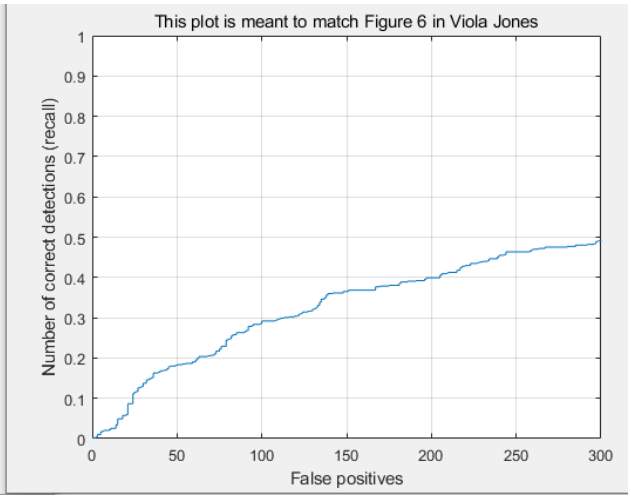


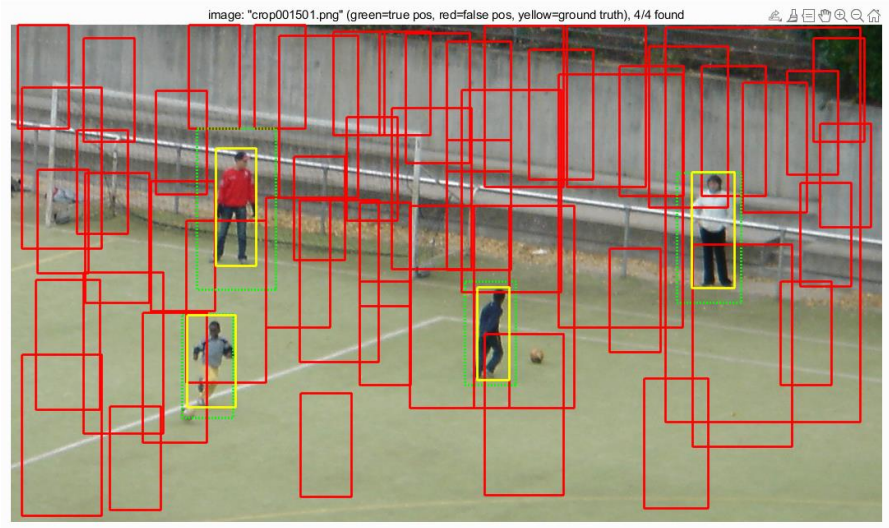


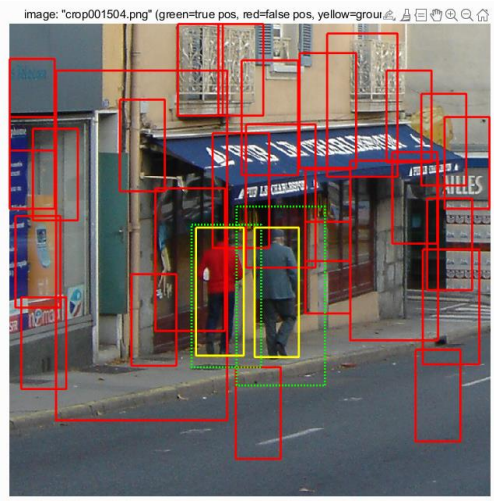
test data set 전체를 detect하는데 걸린 시간: 3082.041322초

3. Cascade architecture with HOG



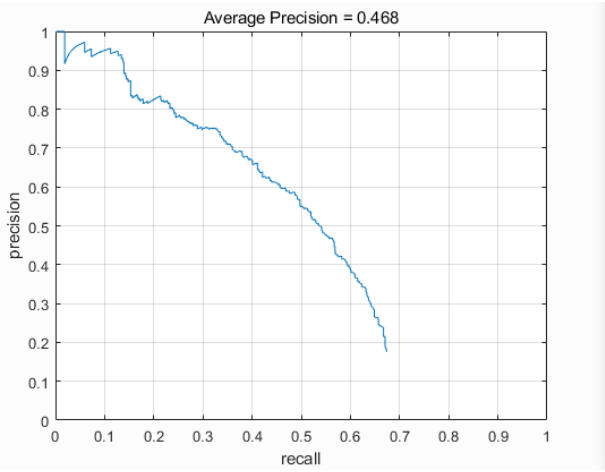


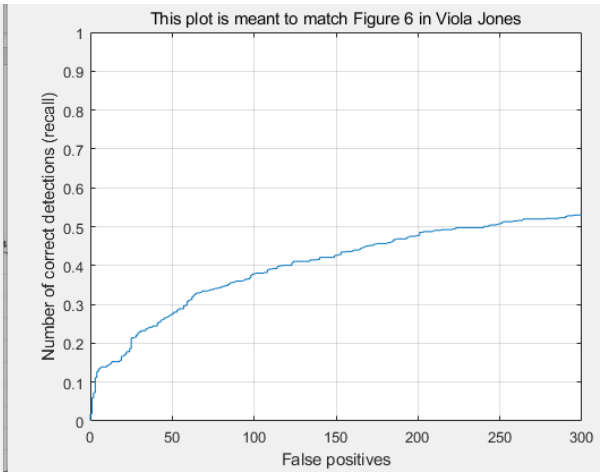


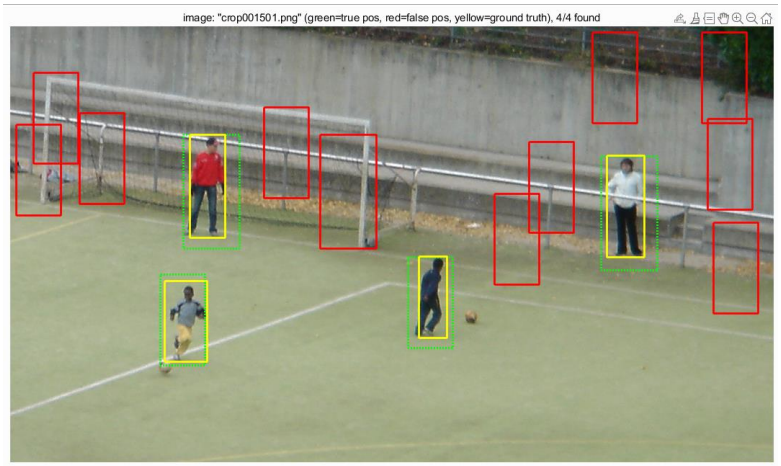


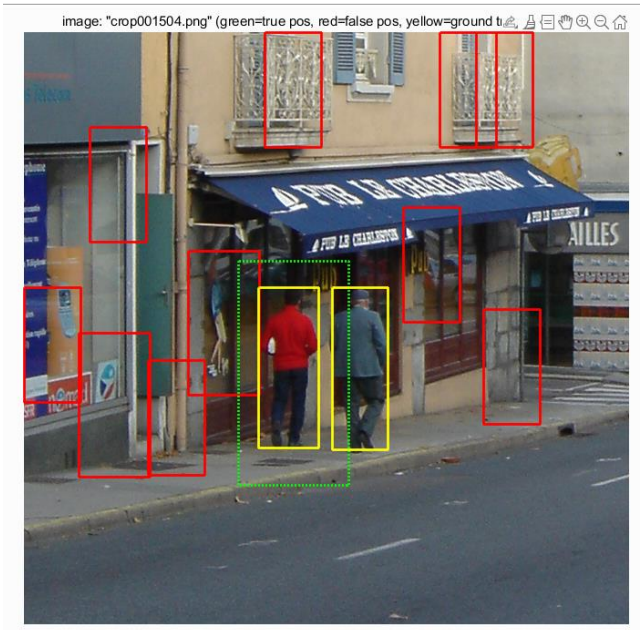
test data set 전체를 detect하는데 걸린 시간: 31054.276663초

4. svm with LBP



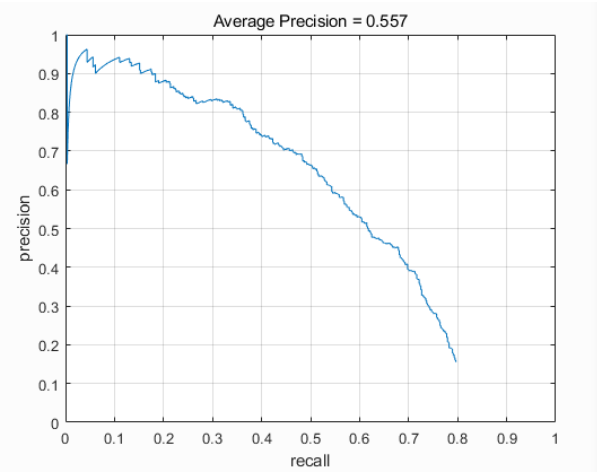


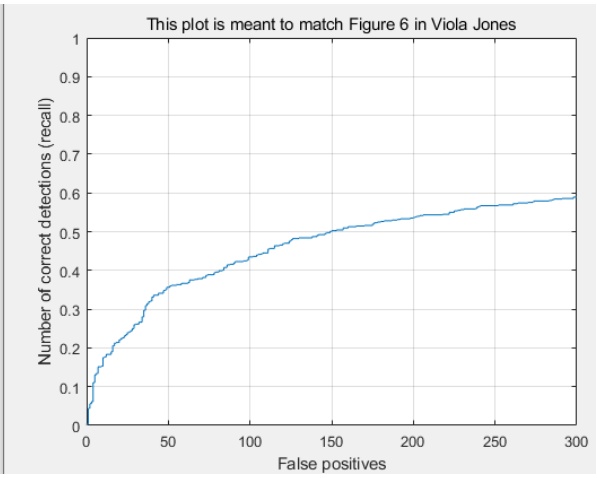


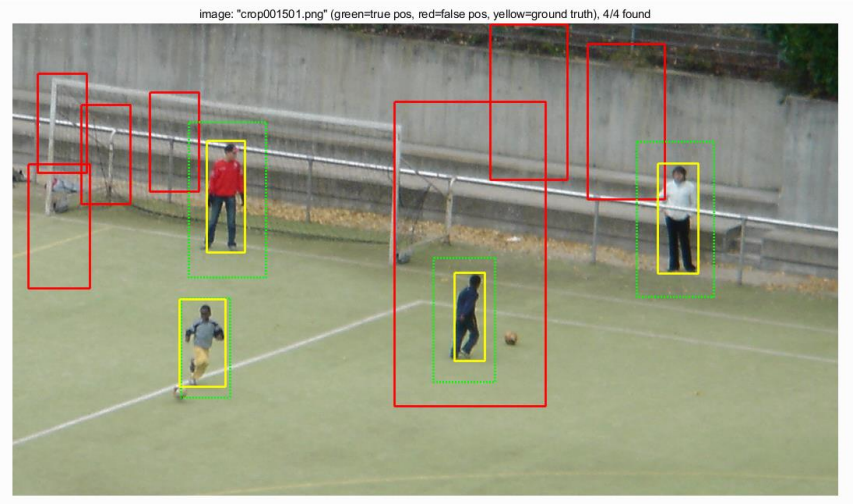


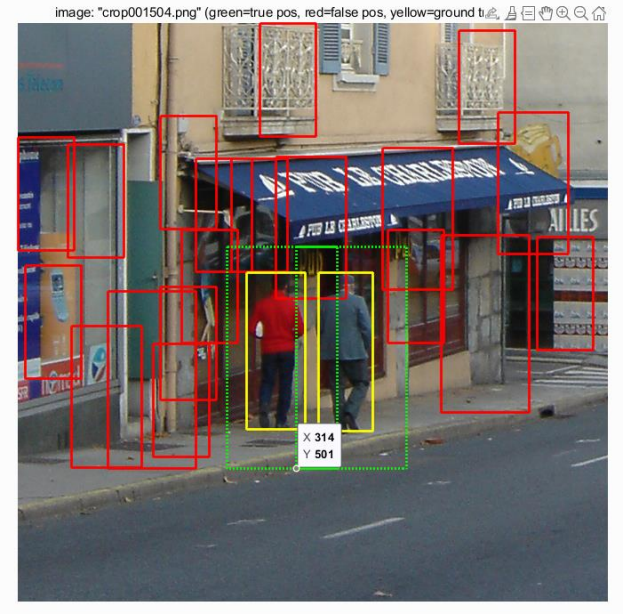
test data set 전체를 detect하는데 걸린 시간: 3788.605023초

1. Bagging HOG svm and LBP svm









test data set 전체를 detect하는데 걸린 시간: 6764.208148초

V. 결과 분석

	Average precision	Test set 전체에 대한 run time
svm with HOG(simgle scale)	0.057	1117.367170초
svm with HOG(multi scale)	0.521	3082.041322초
Cascade architecture with HOG	0.428	31054.276663초
svm with LBP	0.468	3788.605023초
Bagging HOG svm and LBP svm	**0.557**	6764.208148초

1. svm with HOG에서 multi scale의 window를 사용하는 것이 single scale의 윈도우를 사용했을 때 보다 훨씬 좋은 결과를 나타냈다.

2. Cascade with HOG의 결과가 생각보다 나쁜 것을 알 수 있다. 먼저 참고문헌[2]에서는 F_target을 0.001로 training하여 30 level의 cascade 구조를 얻었지만 여기서는 충분한 시간이 없어서 F_target을 0.01로 설정하여 10 level의 cascade 구조를 얻었다. 이로 인해 false positive가 훨씬 많이 검출되었다고 추측할 수 있다. 그리고 run time이 svm with HOG(multi scale)보다 10배이상 길었는데, 이는 HOG계산이 연속적으로 일어나게 코딩된 반면 cascade detect는 HOG 계산을 단편적으로 여러 번 일어나게 코딩하였기 때문에 이렇게 크게 차이가 났다고 추측된다.

3. LBP와 HOG의 결과를 비교해보면 average precision이 HOG가 더 높은 것을 알 수가 있다. 그러나 LBP도 human detection에 상당히 준수한 결과를 보여줌을 알 수 있다. 그래서 이 둘을 bagging한 구조를 사용하여 detect한 결과 accuracy 면에서 향상된 결과를 얻을 수 있었다. 반면에 run time은 길어졌다.

VI. 참고문헌

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

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