

## 一、实验项目名称：

利用滑动窗口进行面部识别

## 二、实验原理：

本次实验首先分别提取正负类训练集的 Hog 特征，将提取得到的训练集 Hog 特征用于训练 SVM 分类器。再提取测试集的 Hog 特征，按照 step 滑动窗口，用刚刚训练得到的 SVM 分类器筛出人脸窗口。最后利用 NMS 移除重叠候选框。

### (1) Hog 特征原理

HOG 特征的全称为 Histogram of Oriented Gradients，该算法先计算图片某一区域中不同方向上梯度的值，然后进行累积，得到直方图，这个直方图就可以代表这块区域，即作为特征输入到分类器里进行分类。

### (2) Hog 特征实现

#### 1. 图像预处理

为了减少光照因素的影响，首先需要将整个图像进行归一化。对于灰度图像，一般为了去除噪点，所以会先利用高斯函数在不同平滑的尺度下进行对灰度图像进行平滑操作。然而，实验表明不做高斯平滑的人体检测效果更好，可能是由于 HOG 特征是基于边缘的，而平滑会降低边缘信息的对比度，从而减少图像中的有用信息，导致效果下降。

#### 2. 梯度计算

分别计算每个像素点的梯度，计算方法如下：

$$G_x(x, y) = I(x + 1, y) - I(x - 1, y)$$

$$G_y(x, y) = I(x, y + 1) - I(x, y - 1)$$

$$G(x, y) = \sqrt{G_x(x, y)^2 + G_y(x, y)^2}$$

$$\alpha(x, y) = \tan\left(\frac{G_y(x, y)}{G_x(x, y)}\right)^{-1}$$

\* $G_x(x, y)$ ,  $G_y(x, y)$  分别表示输入图像在像素点  $(x, y)$  出水平方向和垂直方向的梯度

\* $G(x, y)$  为像素点  $(x, y)$  的梯度幅值

\* $\alpha(x, y)$  为像素点  $(x, y)$  的梯度方向

#### 3. 得到梯度统计直方图

将图像分成若干个单元格 (cell)，每个 cell 边长为 hog\_cell\_size 的像素。假设采用 9 个 bin 的直方图来统计其中一个 cell 的梯度信息，即将 cell 的 0~180 度梯度方向 (若考虑正负则是 0~360 度) 分成 9 个方向块。例如，某个像素的梯度方向落在 20-40 度，那么直方图第 2 个 bin 即的计数就加 1，这样对 cell 内每个像素用梯度方向在直方图中进行加权投影，将其映射到对

应的角度范围块内,就可以得到这个 cell 的梯度方向直方图了,该直方图就是该 cell 对应的 9 维特征向量。以上为不加权的方法,而加权投影的方法就是用当前点的梯度幅值做为权值进行加权。

4. 将每个 cell 的特征合并并且归一化  
把各个单元格(cell)组合成大的、空间上连通的区域(blocks),一个 block 内所有 cell 的特征向量串联起来便得到该 block 的 HOG 特征。再将 HOG 特征归一化,归一化之后的块描述符(向量)就称之为 HOG 描述符。

以上,得到图像的 HOG 描述符后,就可以用于训练 SVM 分类器了。

### 三、实验目的:

创建一个多尺度的滑动窗口面部检测器,该探测器使用 HoG 特征来训练线性 SVM。

### 四、实验内容:

#### (1) 获取训练集

预先准备好存在人脸的正样本和随机选择的负样本(通常为场景等图片)作为训练集,将他们都裁减成大小一致。所有正负样本都将作为特征矩阵中的样本转换到 HoG 空间。

#### (2) 训练 SVM

所有窗口都将转换到 HoG 空间后,标记样本为人脸与非人脸。用该训练数据作为输入以训练 SVM 分类器。

#### (3) 运行窗口检测器

图像被转换为 HoG 空间后,用一个设定好大小的滑动窗口“模板”在 Hog 空间中移动。计算每个窗口的置信度,记录边界框并将其添加为潜在最大值。然后将图像缩小,继续重复上述步骤,直到图像小于“模板”为止。在所有比例的所有边界框上执行非最大值抑制 NMS,以找到最可靠的框,借此消除多个比例尺对同一位置的检测。多尺度的检测能极大地提高探测器的性能。

#### (4) 利用负类特征强化分类器性能

为了提高 SVM 的准确性,可以将它在负样本上运行,以挖掘难以区分(容易与正类混淆)的负面特征。将所有假阳性特征附加到特征矩阵,然后使用更新的特征矩阵和标签再次训练线性 SVM。这样可以降低假阳性率(fp\_rate)从而提高准确性。

### 五、实验步骤:

- (1) project4.m

```

close all
clear
[~,~,~] = mkdir('visualizations');
data_path='D:\..\proj4\data\';
train_path_pos=fullfile(data_path,'caltech_faces/Caltech_CropFace
s');
non_face_scn_path = fullfile(data_path, 'train_non_face_scenes');
test_scn_path = fullfile(data_path,'test_scenes/test_jpg'); %测试集
CMU+MIT test scenes
label_path=fullfile(data_path,'test_scenes/ground_truth_bboxes.txt
');
feature_params = struct('template_size', 36, 'hog_cell_size', 3);

features_pos=get_positive_features(train_path_pos,feature_params
);
num_negative_examples = 12000; %Higher will work strictly
better, but you should start with 10000 for debugging
features_neg = get_random_negative_features( non_face_scn_path,
feature_params, num_negative_examples);
X = [features_pos;features_neg];%正负类的特征
Y = [ones(size(features_pos,1),1); -1*ones(size(features_neg,1),1)];
lambda = 0.0005;
[w,b] = vl_svmtrain(X', Y', lambda);

fprintf('Initial classifier performance on train data:\n')
confidences = [features_pos; features_neg]*w + b; %计算的结果
label_vector      =      [ones(size(features_pos,1),1); -
1*ones(size(features_neg,1),1)];
[tp_rate, fp_rate, tn_rate, fn_rate] = report_accuracy( confidences,
label_vector );
non_face_confs = confidences( label_vector < 0);
face_confs     = confidences( label_vector > 0);
figure(2);
plot(sort(face_confs), 'g'); hold on
plot(sort(non_face_confs),'r');
plot([0 size(non_face_confs,1)], [0 0], 'b');
hold off;
n_hog_cells = sqrt(length(w) / 31); %specific to default HoG
parameters
imhog = vl_hog('render', single(reshape(w, [n_hog_cells n_hog_cells
31])), 'verbose') ;
figure(3); imagesc(imhog) ; colormap gray; set(3, 'Color',
[.988, .988, .988])
pause(0.1) %let's ui rendering catch up

```

```

hog_template_image = frame2im(getframe(3));
imwrite(hog_template_image, 'visualizations/hog_template.png')
[hn_bboxes, hn_con, hn_id] = run_detector(non_face_scn_path, w, b,
feature_params);

hn_features= zeros(length(hn_con), (feature_params.template_size
/ feature_params.hog_cell_size)^2 * 31);

for i = 1: length(hn_con)
    hn_bbox = hn_bboxes(i, :);
    img = rgb2gray(imread(fullfile(non_face_scn_path, hn_id{i})));

    x_min = max(floor(hn_bbox(1)),1);
    y_min = max(floor(hn_bbox(2)),1);
    x_max = min(ceil(hn_bbox(3)),size(img,2));
    y_max = min(ceil(hn_bbox(4)),size(img,1));

    scaled_img = imresize(img(y_min:y_max, x_min:x_max),
[feature_params.template_size, feature_params.template_size]);
    hog = vl_hog(single(scaled_img),
feature_params.hog_cell_size);
    hn_features(i,:) = reshape(hog, [1,
(feature_params.template_size / feature_params.hog_cell_size)^2 *
31]);
end
[w, b] = vl_svmtrain([features_pos; features_neg; hn_features]',
[ones(1, size(features_pos, 1)), -ones(1, size(features_neg, 1)), -ones(1,
size(hn_features, 1))], lambda);
[bboxes, confidences, image_ids] = run_detector(test_scn_path, w,
b, feature_params);

[gt_ids, gt_bboxes, gt_isclaimed, tp, fp, duplicate_detections] = ...
evaluate_detections(bboxes, confidences, image_ids,
label_path);

visualize_detections_by_image(bboxes, confidences, image_ids, tp,
fp, test_scn_path, label_path)
visualize_detections_by_image_no_gt(bboxes, confidences,
image_ids, test_scn_path)

visualize_detections_by_confidence(bboxes, confidences,
image_ids, test_scn_path, label_path);

```

(2) get\_positive\_features.m 得到正类的 hog 特征

```

function features_pos = get_positive_features(train_path_pos,
feature_params)
    image_files = dir( fullfile( train_path_pos, '*.jpg' ) ); %Caltech Faces
stored as .jpg
    num_images = length(image_files);
    features_pos = [];
    % placeholder to be deleted
    % features_pos = rand(100, (feature_params.template_size /
feature_params.hog_cell_size)^2 * 31);

    for i = 1:num_images
        image_path = fullfile(train_path_pos,image_files(i).name);
        image = imread(image_path);
        if(size(image,3) > 1)
            image = rgb2gray(image);
        end
        HOG = vl_hog(single(image), feature_params.hog_cell_size);
        features_pos=cat(1,features_pos,
reshape(HOG,[1,(feature_params.template_size/feature_params.hog_ce
ll_size)^2 * 31]));

    end
end

```

- (3) get\_random\_negative\_features.m——在多个尺度上随机选取负类样本，得到负类的 hog 特征

```

function features_neg = get_random_negative_features
(non_face_scn_path, feature_params, num_samples)

image_files = dir( fullfile( non_face_scn_path, '*.jpg' ) );
num_images = length(image_files);

features_neg=[];

for i = 1:num_images
    image_path = fullfile(non_face_scn_path,image_files(i).name);
    image = imread(image_path);
    if(size(image,3) > 1)
        image = rgb2gray(image);
    end
    [h,w]=size(image);
    for j = 1:num_samples/num_images
        index_i = randperm(h-feature_params.template_size,1);
        index_j = randperm(w-feature_params.template_size,1);
        HOG
    end
end

```

```

vl_hog(single(image(index_i:index_i+feature_params.template_size-
1,index_j:index_j+feature_params.template_size-1)),...
        feature_params.hog_cell_size);
        features_neg = cat(1,features_neg,
reshape(HOG,[1,(feature_params.template_size
feature_params.hog_cell_size)^2 * 31]));
    end
end
end
end

```

(4) report\_accuracy.m 得到精度

```

function [tpr, fpr, tnr, fnr] = report_accuracy( confidences,
label_vector )

    correct_classification = sign(confidences .* label_vector);
    accuracy=1-
sum(correct_classification<=0)/length(correct_classification);
    fprintf(' accuracy:   %.3f\n', accuracy);

    true_positives = (confidences >= 0) & (label_vector >= 0);
    tpr = sum( true_positives ) / length( true_positives);
    fprintf(' true  positive rate: %.3f\n', tpr);

    false_positives = (confidences >= 0) & (label_vector < 0);
    fpr = sum( false_positives ) / length( false_positives);
    fprintf(' false positive rate: %.3f\n', fpr);

    true_negatives = (confidences < 0) & (label_vector < 0);
    tnr = sum( true_negatives ) / length( true_negatives);
    fprintf(' true  negative rate: %.3f\n', tnr);

    false_negatives = (confidences < 0) & (label_vector >= 0);
    fnr = sum( false_negatives ) / length( false_negatives);
    fprintf(' false negative rate: %.3f\n', fnr);

```

(5) run\_detector.m 窗口检测器函数，固定大小的窗口对多个尺度的图像进行滑窗检测，将多个尺度计算得到的矩形框都还原成原图尺寸，再进行非极大值抑制

```

function [bboxes, confidences, image_ids] = ...
    run_detector(test_scn_path, w, b, feature_params)

test_scenes = dir( fullfile( test_scn_path, '*.jpg' ));

bboxes = zeros(0,4);
confidences = zeros(0,1);
image_ids = cell(0,1);

```

```

iter = 0.8;
scale_vector = [1,iter,iter^2,iter^3,iter^4,iter^5,iter^6];
threshold = 0.75;

for i = 1:length(test_scenes)

    fprintf('Detecting faces in %s\n', test_scenes(i).name)
    img = imread( fullfile( test_scn_path, test_scenes(i).name ));
    img = single(img)/255;
    if(size(img,3) > 1)
        img = rgb2gray(img);
    end

    cur_bboxes = zeros(0,4);
    cur_confidences = zeros(0,1);
    cur_image_ids = cell(0,1);

    for scale = scale_vector
        scaled_img = imresize(img,scale);
        [height,width] = size(scaled_img);
        HOG = vl_hog(scaled_img,feature_params.hog_cell_size);
        window_x_limit =
floor(width/feature_params.hog_cell_size) - ...

feature_params.template_size/feature_params.hog_cell_size + 1;
        window_y_limit =
floor(height/feature_params.hog_cell_size) - ...

feature_params.template_size/feature_params.hog_cell_size + 1;

        D =
(feature_params.template_size/feature_params.hog_cell_size)^2*31;
        features = zeros(window_x_limit * window_y_limit,D);
        for x=1:window_x_limit
            for y=1:window_y_limit
                features((x-
1)*window_y_limit+y,:)=reshape(HOG(y:(y+feature_params.template_s
ize/feature_params.hog_cell_size-1),...

x:(x+feature_params.template_size/feature_params.hog_cell_size-
1),:),...

[1,D]);
            end
        end
    end
end

```

```

        scores = features * w + b;
        indices = find(scores>threshold);
        scale_confidences = scores(indices);

        x_hog = floor(indices./ window_y_limit);
        y_hog = mod(indices,window_y_limit)-1;
        scale_x_min = x_hog*feature_params.hog_cell_size+1;
        scale_x_max = x_hog*feature_params.hog_cell_size+feature_params.template_size;
        scale_y_min = y_hog*feature_params.hog_cell_size+1;
        scale_y_max = y_hog*feature_params.hog_cell_size+feature_params.template_size;
        scale_bboxes = [scale_x_min, scale_y_min, scale_x_max, scale_y_max]./scale;
        scale_image_ids = repmat({test_scenes(i).name}, size(indices,1), 1);

        cur_bboxes = [cur_bboxes; scale_bboxes];
        cur_confidences = [cur_confidences; scale_confidences];
        cur_image_ids = [cur_image_ids; scale_image_ids];

    end

    [is_maximum] = non_max_supr_bbox(cur_bboxes, cur_confidences, size(img));

    cur_confidences = cur_confidences(is_maximum,:);
    cur_bboxes = cur_bboxes(is_maximum,:);
    cur_image_ids = cur_image_ids(is_maximum,:);

    bboxes = [bboxes; cur_bboxes];
    confidences = [confidences; cur_confidences];
    image_ids = [image_ids; cur_image_ids];

end

```

(6) non\_max\_supr\_bbox.m 非极大值抑制函数

```

function [is_valid_bbox] = non_max_supr_bbox(bboxes, confidences, img_size, verbose)

    if(~exist('verbose', 'var'))
        verbose = false;
    end

    x_out_of_bounds = bboxes(:,3) > img_size(2); %xmax is greater

```



```

than x dimension
    y_out_of_bounds = bboxes(:,4) > img_size(1); %ymax is greater
than y dimension

    bboxes(x_out_of_bounds,3) = img_size(2);
    bboxes(y_out_of_bounds,4) = img_size(1);

    num_detections = size(confidences,1);

    [confidences, ind] = sort(confidences, 'descend');
    bboxes = bboxes(ind,:);

    is_valid_bbox = logical(zeros(1,num_detections));

    for i = 1:num_detections
        cur_bb = bboxes(i,:);
        cur_bb_is_valid = true;

        for j = find(is_valid_bbox)

            prev_bb=bboxes(j,:);
            bi=[max(cur_bb(1),prev_bb(1)) ; ...
                max(cur_bb(2),prev_bb(2)) ; ...
                min(cur_bb(3),prev_bb(3)) ; ...
                min(cur_bb(4),prev_bb(4))];
            iw=bi(3)-bi(1)+1;
            ih=bi(4)-bi(2)+1;
            if iw>0 && ih>0

                ua=(cur_bb(3)-cur_bb(1)+1)*(cur_bb(4)-cur_bb(2)+1)+
                (prev_bb(3)-prev_bb(1)+1)*(prev_bb(4)-prev_bb(2)+1)-iw*ih;
                ov=iw*ih/ua;
                if ov > 0.3 %If the less confident detection overlaps too much with
the previous detection
                    cur_bb_is_valid = false;
                end
                center_coord = [(cur_bb(1) + cur_bb(3))/2,
(cur_bb(2) + cur_bb(4))/2];
                if( center_coord(1) > prev_bb(1) && center_coord(1)
< prev_bb(3) && ...
                    center_coord(2) > prev_bb(2) &&
center_coord(2) < prev_bb(4))

```

```

                                cur_bb_is_valid = false;
                                end

                                if(verbose)
                                    fprintf('detection %d, bbox: [%d %d %d %d], %f overlap with %d
[%d %d %d %d]\n', i, cur_bb(1), cur_bb(2), cur_bb(3), cur_bb(4), ov, j,
prev_bb(1), prev_bb(2), prev_bb(3),prev_bb(4))
                                end
                                end
                                end

                                is_valid_bbox(i) = cur_bb_is_valid;

                                end

                                reverse_map(ind) = 1:num_detections;
                                is_valid_bbox = is_valid_bbox(reverse_map);

                                fprintf(' non-max suppression: %d detections to %d final bounding
boxes\n', num_detections, sum(is_valid_bbox));

```

(7) evaluate\_all\_detections.m 将运行结果进行可视化

```

function [gt_ids, gt_bboxes, gt_isclaimed, tp, fp,
duplicate_detections] = ...
    evaluate_detections(bboxes, confidences, image_ids,
label_path, draw)

    if(~exist('draw', 'var'))
        draw = 1;
    end

    fid = fopen(label_path);
    gt_info = textscan(fid, '%s %d %d %d %d');
    fclose(fid);
    gt_ids = gt_info{1,1};
    gt_bboxes = [gt_info{1,2}, gt_info{1,3}, gt_info{1,4}, gt_info{1,5}];
    gt_bboxes = double(gt_bboxes);

    gt_isclaimed = zeros(length(gt_ids),1);
    npos = size(gt_ids,1); %total number of true positives.

    [sc,si]=sort(-confidences);
    image_ids=image_ids(si);

```

```

bboxes    =bboxes(si,:);

nd=length(confidences);
tp=zeros(nd,1);
fp=zeros(nd,1);
duplicate_detections = zeros(nd,1);
tic;
for d=1:nd

    if toc>1
        fprintf('pr: compute: %d/%d\n',d,nd);
        drawnow;
        tic;
    end
    cur_gt_ids = strcmp(image_ids{d}, gt_ids); %will this be slow?

    bb = bboxes(d,:);
    ovmax=-inf;

    for j = find(cur_gt_ids')
        bbgt=gt_bboxes(j,:);
        bi=[max(bb(1),bbgt(1))    ;    max(bb(2),bbgt(2))    ;
min(bb(3),bbgt(3)) ; min(bb(4),bbgt(4))];
        iw=bi(3)-bi(1)+1;
        ih=bi(4)-bi(2)+1;
        if iw>0 && ih>0
            ua=(bb(3)-bb(1)+1)*(bb(4)-bb(2)+1)+...
                (bbgt(3)-bbgt(1)+1)*(bbgt(4)-bbgt(2)+1)-...
                iw*ih;
            ov=iw*ih/ua;
            if ov>ovmax %higher overlap than the previous best?
                ovmax=ov;
                jmax=j;
            end
        end
    end
end

if ovmax >= 0.3
    if ~gt_isclaimed(jmax)
        tp(d)=1;
        gt_isclaimed(jmax)=true;
    else
        fp(d)=1;
    end
end

```

```

        duplicate_detections(d) = 1;
    end
    else
        fp(d)=1;
    end
end

% compute cumulative precision/recall
cum_fp=cumsum(fp);
cum_tp=cumsum(tp);
rec=cum_tp/npos;
prec=cum_tp./(cum_fp+cum_tp);

ap=VOCap(rec,prec);

if draw
    % plot precision/recall
    figure(12)
    plot(rec,prec,'-');
    axis([0 1 0 1])
    grid;
    xlabel 'recall'
    ylabel 'precision'
    title(sprintf('Average Precision = %.3f',ap));
    set(12, 'Color', [.988, .988, .988])

    pause(0.1) %let's ui rendering catch up
    average_precision_image = frame2im(getframe(12));

    imwrite(average_precision_image,
'visualizations/average_precision.png')

    figure(13)
    plot(cum_fp,rec,'-')
    axis([0 300 0 1])
    grid;
    xlabel 'False positives'
    ylabel 'Number of correct detections (recall)'
    title('This plot is meant to match Figure 6 in Viola Jones');
end

reverse_map(si) = 1:nd;
tp = tp(reverse_map);
fp = fp(reverse_map);

```

```
duplicate_detections = duplicate_detections(reverse_map);
```

(8) VOCap.m

```
function ap = VOCap(rec,prec)
mrec=[0 ; rec ; 1];
mpre=[0 ; prec ; 0];
for i=numel(mpre)-1:-1:1
    mpre(i)=max(mpre(i),mpre(i+1));
end
i=find(mrec(2:end)~=mrec(1:end-1))+1;
ap=sum((mrec(i)-mrec(i-1)).*mpre(i));
```

(9) visualize\_detections\_by\_image.m

```
function visualize_detections_by_image(bboxes, confidences,
image_ids, tp, fp, test_scn_path, label_path, onlytp)

if(~exist('onlytp', 'var'))
    onlytp = false;
end

fid = fopen(label_path);
gt_info = textscan(fid, '%s %d %d %d %d');
fclose(fid);
gt_ids = gt_info{1,1};
gt_bboxes = [gt_info{1,2}, gt_info{1,3}, gt_info{1,4}, gt_info{1,5}];
gt_bboxes = double(gt_bboxes);

gt_file_list = unique(gt_ids);

num_test_images = length(gt_file_list);

for i=1:num_test_images
    cur_test_image = imread( fullfile( test_scn_path, gt_file_list{i}));
    cur_gt_detections = strcmp( gt_file_list{i}, gt_ids);
    cur_gt_bboxes = gt_bboxes(cur_gt_detections,:);

    cur_detections = strcmp(gt_file_list{i}, image_ids);
    cur_bboxes = bboxes(cur_detections,:);
    cur_confidences = confidences(cur_detections);
    cur_tp = tp(cur_detections);
    cur_fp = fp(cur_detections);

    figure(15)
    imshow(cur_test_image);
    hold on;
```

```

num_detections = sum(cur_detections);

for j = 1:num_detections
    bb = cur_bboxes(j,:);
    if(cur_tp(j)) %this was a correct detection
        plot(bb([1 3 3 1 1]),bb([2 2 4 4 2]),'g','linewidth',2);
    elseif(cur_fp(j))
        plot(bb([1 3 3 1 1]),bb([2 2 4 4 2]),'r-','linewidth',2);
    else
        error('a detection was neither a true positive or a false
positive')
    end
end

num_gt_bboxes = size(cur_gt_bboxes,1);

for j=1:num_gt_bboxes
    bbgt=cur_gt_bboxes(j,:);
    plot(bbgt([1 3 3 1 1]),bbgt([2 2 4 4 2]),'y-','linewidth',2);
end

hold off;
axis image;
axis off;
title(sprintf('image: "%s" (green=true pos, red=false pos,
yellow=ground truth), %d/%d found',...
              gt_file_list{i},                sum(cur_tp),
size(cur_gt_bboxes,1)), 'interpreter','none');

set(15, 'Color', [.988, .988, .988])
pause(0.1) %let's ui rendering catch up
detection_image = frame2im(getframe(15));

imwrite(detection_image,
sprintf('visualizations/detections_%s.png', gt_file_list{i}))

fprintf('press any key to continue with next image\n');
pause;
end

```

(10) visualize\_detections\_by\_image\_no\_gt.m

```

function visualize_detections_by_image_no_gt(bboxes,
confidences, image_ids, test_scn_path)

test_files = dir(fullfile(test_scn_path, '*.jpg'));

```

```

num_test_images = length(test_files);

for i=1:num_test_images
    cur_test_image = imread( fullfile( test_scn_path,
test_files(i).name));

    cur_detections = strcmp(test_files(i).name, image_ids);
    cur_bboxes = bboxes(cur_detections,:);
    cur_confidences = confidences(cur_detections);

    figure(15)
    imshow(cur_test_image);
    hold on;

    num_detections = sum(cur_detections);

    for j = 1:num_detections
        bb = cur_bboxes(j,:);
        plot(bb([1 3 3 1 1]),bb([2 2 4 4 2]),'g','linewidth',2);
    end

    hold off;
    axis image;
    axis off;
    title(sprintf('image:          "%s"          green=detection',
test_files(i).name),'interpreter','none');

    set(15, 'Color', [.988, .988, .988])
    pause(0.1) %let's ui rendering catch up
    detection_image = frame2im(getframe(15));

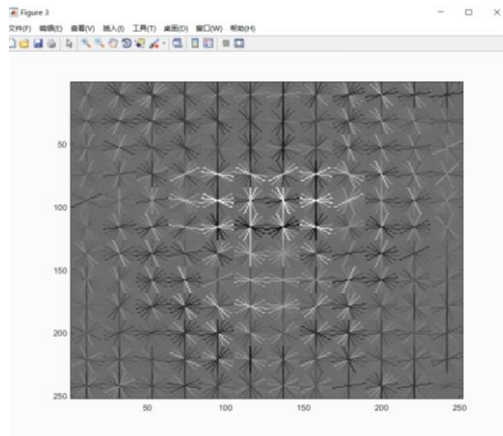
    imwrite(detection_image,
sprintf('visualizations/detections_%s.png', test_files(i).name))

    fprintf('press any key to continue with next image\n');
    pause;
end

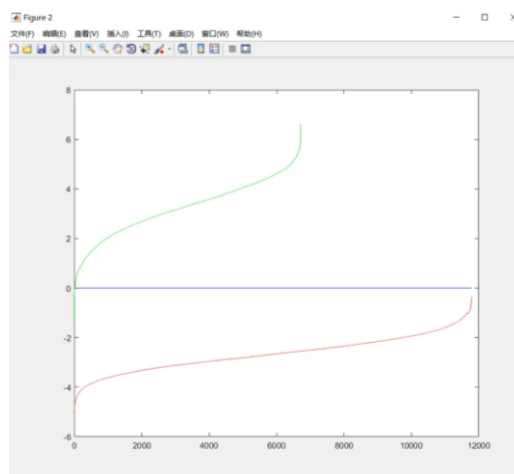
```

## 六、实验数据及结果分析：

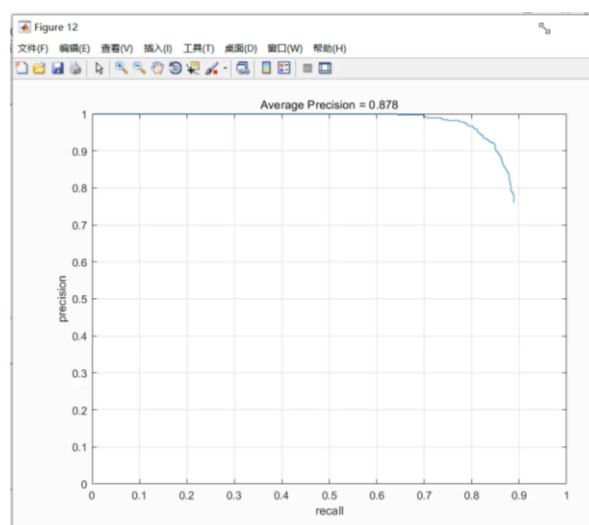
将训练得到的 HOG template 可视化如下：



人脸特征和非人脸特征的 **confidences** 分布如下：

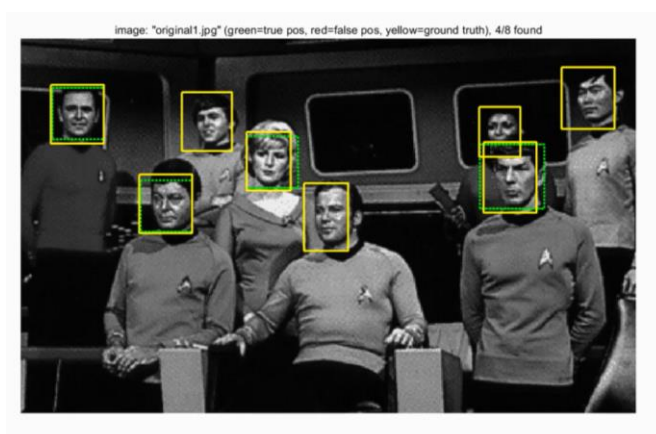
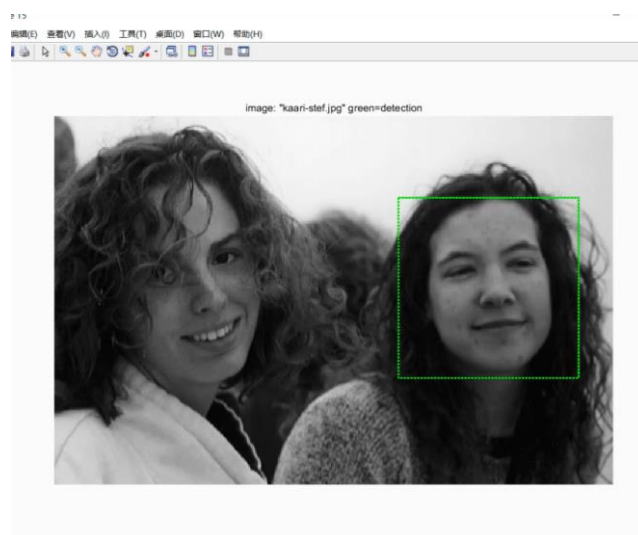
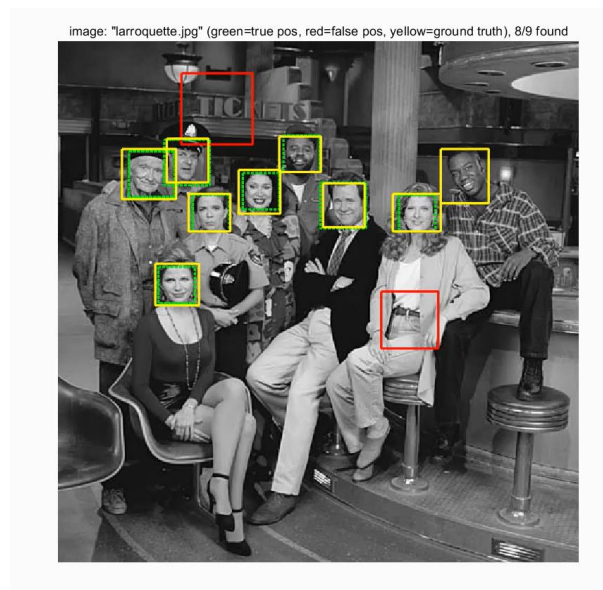


PR 曲线如下所示，说明当召回率设置为 0.7 时，能在保证精确度的基础上尽可能的找全正样本。



对图片中人脸识别的结果如下：

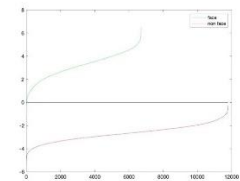
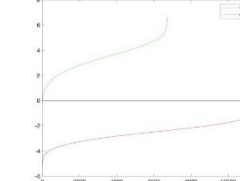
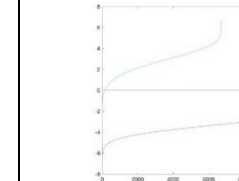
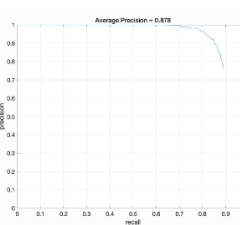
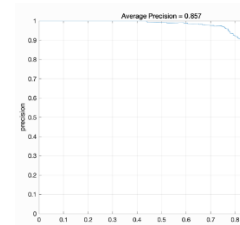
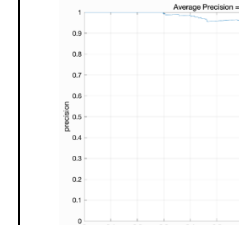
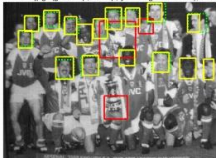
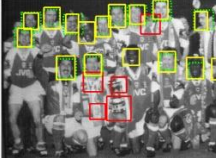
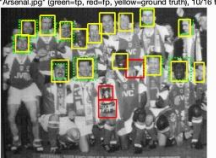




可见，当人脸存在遮挡或旋转时，精度会明显下降。

#### (1) 探讨 `hog_cell_size` 对结果的影响

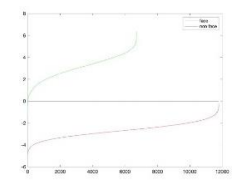
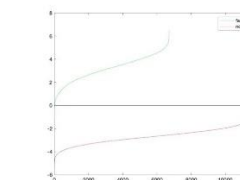
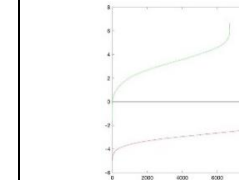
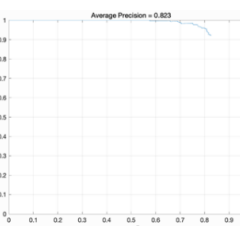
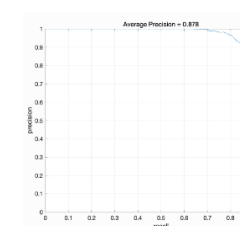
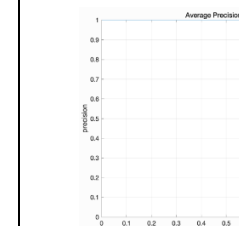
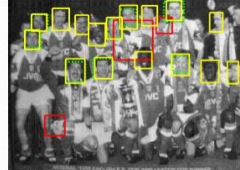

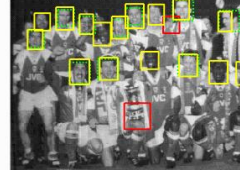
控制  $\lambda=0.0005$ ,  $\text{threshold}=0.75$ , 当 `hog_cell_size` 分别取以下值时，得到的实验结果如下：

hog_cell_size	3	4	6
precision	0.878	0.857	0.762
confidences			
Precision-recall curve			
实例结果	image: "Arsenal.jpg" (green=tp, red=fp, yellow=ground truth), 13/16 found 	image: "Arsenal.jpg" (green=tp, red=fp, yellow=ground truth), 13/16 found 	image: "Arsenal.jpg" (green=tp, red=fp, yellow=ground truth), 10/16 found 

由实验结果可见，随着 hog\_cell\_size 的增大，精度不断下降，相应的计算量变小。

## (2) 探讨 lambda 对结果的影响

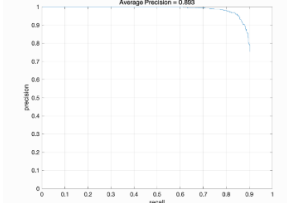
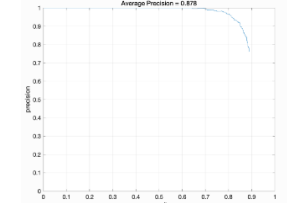
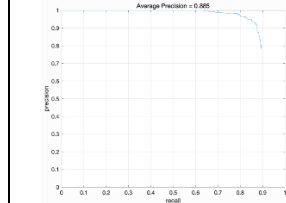
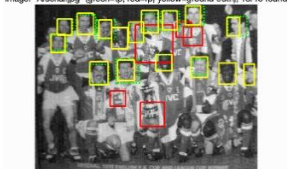
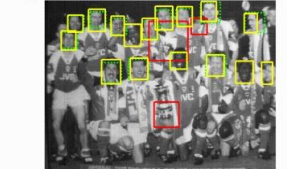
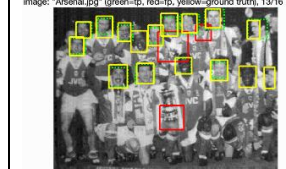
控制 hog\_cell\_size=3, threshold=0.75, 当 hog\_cell\_size 分别取以下值时，得到的实验结果如下：

Lambda	0.0001	0.0005	0.001
precision	0.823	0.878	0.795
confidences			
Precision-recall curve			
实例结果	image: "Arsenal.jpg" (green=tp, red=fp, yellow=ground truth), 8/16 found 	image: "Arsenal.jpg" (green=tp, red=fp, yellow=ground truth), 13/16 found 	image: "Arsenal.jpg" (green=tp, red=fp, yellow=ground truth), 10/16 found 

由实验结果可见，随着 lambda 的增大，结果精度先增大后减小。Lambda 增大，精度先增大可能是因为训练集和测试集之间存在差

异，在训练集中得到的样本并不能适应测试集的人脸。Lambda 增大，精度后减小是因为 SVM 分类器的特性。

- (3) 探讨 threshold 对结果的影响  
控制 hog\_cell\_size=3, lambda=0.0005, 当 hog\_cell\_size 分别取以下值时，得到的实验结果如下：

threshold	0.7	0.75	0.8
precision	0.893	0.878	0.885
Precision-recall curve			
实例结果			

由实验结果可见，随着 threshold 的增大，结果精度先减小后增大。threshold 增大，精度先减小可能是因为将一些正类也判别为负类了。

七、实验结论：

由以上结果可见，本次实验实现了利用滑动窗口进行人脸检测的目的，从图片中提取的 HOG 特征也基本反映了人脸的特征，SVM 分类器也能将人脸和非人脸进行区分。

八、总结及心得体会：

本次实验主要有获取训练集、训练 SVM、运行窗口检测器、利用负类特征强化分类器性能几个步骤。经过本次实验，我对 HOG 特征检测有了更深入的理解，跟 SIFT 相比，HOG 没有选取主方向，所以它本身不具有旋转不变性，如果想要让 hog 具有旋转不变性，就需要利用不同旋转方向的训练样本来实现。此外，HOG 本身也不具有尺度不变性，其尺度不变性是通过缩放检测窗口图像的大小来实现的。而且 HOG 在遮挡问题的处理上表现的并不是很好。而且由于 HOG 的特性，一般进行高斯平滑后会降低准确度，故有时不进行平滑，这样就导致了图片中噪点的很容易影响结果。

## 九、对本实验过程及方法的改进建议：

HOG 特征还可以与 pyramid 相结合成为 PHOG，即对同一幅图像进行不同尺度的分割，然后计算每个尺度中 patch 的小 HOG，最后将他们连接成一个很长的一维向量作为特征。例如：对一幅  $512 \times 512$  的图像先做  $3 \times 3$  的分割，再做  $6 \times 6$  的分割，最后做  $12 \times 12$  的分割。接下来对分割出的 patch 计算小 HOG，假设有 12 个 bin 即 12 维。那么就有  $9 \times 12 + 36 \times 12 + 144 \times 12 = 2268$  维。需要注意的是，在将这些不同尺度上获得的小 HOG 连接起来时，必须先对其做归一化，因为  $3 \times 3$  尺度中的 HOG 任意一维的数值很可能比  $12 \times 12$  尺度中任意一维的数值大很多。PHOG 与传统 HOG 相比，它可以检测到不同尺度的特征，表达能力更强，但是数据量和计算量都比 HOG 大了不少。之后的实验可以尝试利用 PHOG 替代传统 HOG 提升精度。