Computer Vision: EECE 5639
Project 4
November 15, 2019
Due on December 6, 2019
Object Category Detection

By, John Nguyen and Jiayuan Zhang

Abstract:
The objective of Project 4 is detect objects of a given class using a generalized hough transform approach, a bin voting method. Given a set of training images which are examples of a car dataset, a bag of words method is used to extract key features. Harris corner detection is used to find key features and an image patch centered at these corners are recorded. From there the displacement vectors are stored between the patch and the training image center. This is repeated for the training images and with the visual vocabulary, the object position is used to index votes where each visual word is expected to be a car part. Each part uses the recorded displacement and using the peaks in the voting space leads to one or more objects being detected.

Experiments and Algorithms:
To begin the experiment, the training dataset is 550 cropped images of cars, the object of interest. With the training image, the first step performed is Harris Corner Detection. As mentioned iin Project 2 and Project 3, the following is used…

"The first algorithm used, the Harris Corner Detector, extracts corners from the input frame. Corners are useful for finding the correspondences between small sections of images so that the similarities between images can be observed. The basic methodology of corner detection is to find two adjacent strong edges. The Harris corner detector determines the amount of change in intensity in a small window when moving in directions:

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u,y+v) - I(x,y)]^2 \quad \text{[Eq 1.]}$$

Using the first order approximation of f(x,y), the previous equation can be rewritten as

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \quad \text{[Eq 2.]}$$

The matrix M is calculated from the derivatives at each pixel. The presence of corners can be determined by calculating the eigenvalues λ_1, λ_2 of M: there is a corner when both are large. The corner response can be calculated as the following. R is large when there is a corner while it is negative when there is an edge. By thresholding R, we can get the corner pixels.

$$R = det\, M - k(trace\, M)^2 \quad \text{[Eq 3.]}$$

The R value matrix is rotation invariant which is advantageous since it does not need any pre alignment. Equation 3 describes the R value computation where k is set to .04. After getting the vertical and horizontal gradient vectors, the sums of the square individual gradients and the

product of the gradient is used to create the R matrix at each respective pixel. A corner is denoted by a high R value as shown in the figure below.

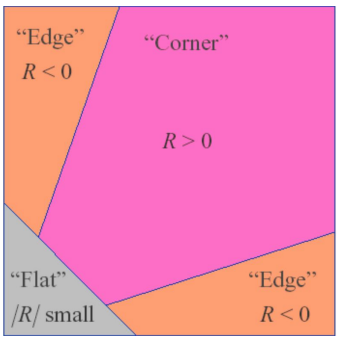

Figure 1 - R value response

The R values chosen are shown to be effective in identifying corners and edges." (Project 2).

After acquiring the corners, a struct is created for each image in the data set as follows.

```
img =

  struct with fields:

              OG_RGB: [112×167 uint8]
                 RGB: [112×167 uint8]
  corner_coordinates: [78×2 single]
             patches: [15×15×78 uint8]
                xywh: [78×4 single]
           vocab_idx: [1×78 double]
```

The image data structure stores the RGB image values, the corner coordinate values, the 25x25 patch surrounding the coordinate, the position vector to plot the patch rectangle and the vocabulary index which will be discussed later. The patch was found by offsetting the corner coordinate by half of the window size and acquiring the entire image with the corner detected being the center of the patch. A similar offset is used the position vector since the MATLAB rectangle function also takes in the top-left position of the rectangle as opposed to the center of the image. The following images are examples of the corner coordinates detected which is denoted by displayed bounding box around the features of interest.

Figure 2: Feature Patches Acquired

An important fact to notice from the harris corner detector is that since the training images were cropped and were small in resolution, 40x100, a possible source of error from our experiment were not enough corners were detected or missed detections. Preprocessing such as an averaging and blurring filter were used to no avail as an improvement in corner detection was not apparent. With the patch acquired, the vector of raw pixel values are stored in the data structure as descriptors. With the high amount of patches, the next step performed was K-means clustering. K means clustering is an iterative algorithm that randomly assigns K number of random points within the feature space. From there, each corner is determined to belong to a cluster based on its Euclidean distance from the centroid. This process is repeated until the best centroid are determined based on the smallest Euclidean distance measurement meaning each patch is closest to correct centroid and this becomes the patches label. A method to determine the best K values is the elbow method since K means is an unsupervised machine learning algorithm. The elbow method runs K-means for different values of K and produces a plot of the sum of the square errors vs number of clusters. The elbow of this plot is where the sum of the square error is small and tends to decrease towards 0 as k is increased. Essentially a small value of k with low SSE score is desired. The following chart is this procedure performed on the different k values.

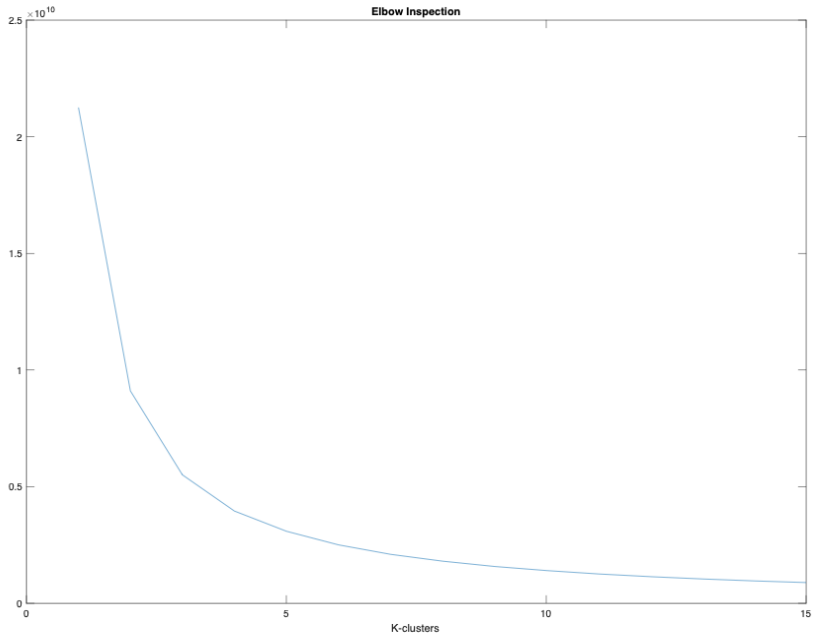

Figure 2: Elbow inspection

From inspecting the elbow plot, the best K was determined to be 6. From there the data is refit with 6 means clustering. This step significantly reduces the number of possible visual words and the clusters formed create a visual vocabulary for the process. Having found the vocabulary, each local patch is assigned to a visual word based on the Euclidean distance and labels are assigned to each patch. Next off, for each visual word occurence in the training example, each possible displacement vector between the occurrence and the center of the training gimage is computed.

The next stage of the experiment is to test the bag of words through using unseen testing images on the table of displacement vectors. Similar to the training image, the harris corner detector is used to find key features. At each interesting point, a 25x25 image patch of raw pixels is captured and similarly each patch is assigned a visual word. Each visual word occurrence votes for the potential object location and after all votes are casted, the top 10 maximum patches with corresponding displacement vectors of interest are used to determine the object location. From there the 10 points acquired from the voting scheme are used to determine the object location. Often, the highest voted point has a neighboring point of interest so redundancy based on euclidean distance is computed so that only one bounding box is created for each object. The following figures are the results of the predicted bounding box in blue vs the groundtruth bounding box which is denoted in yellow. The voting scheme is displayed using a normalized heat map to display the votes casted where the lightest/brightest points are the predicted object detection. An important assumption made is that scale and orientation of the object is the same as the ones using during training which is two dimensional.

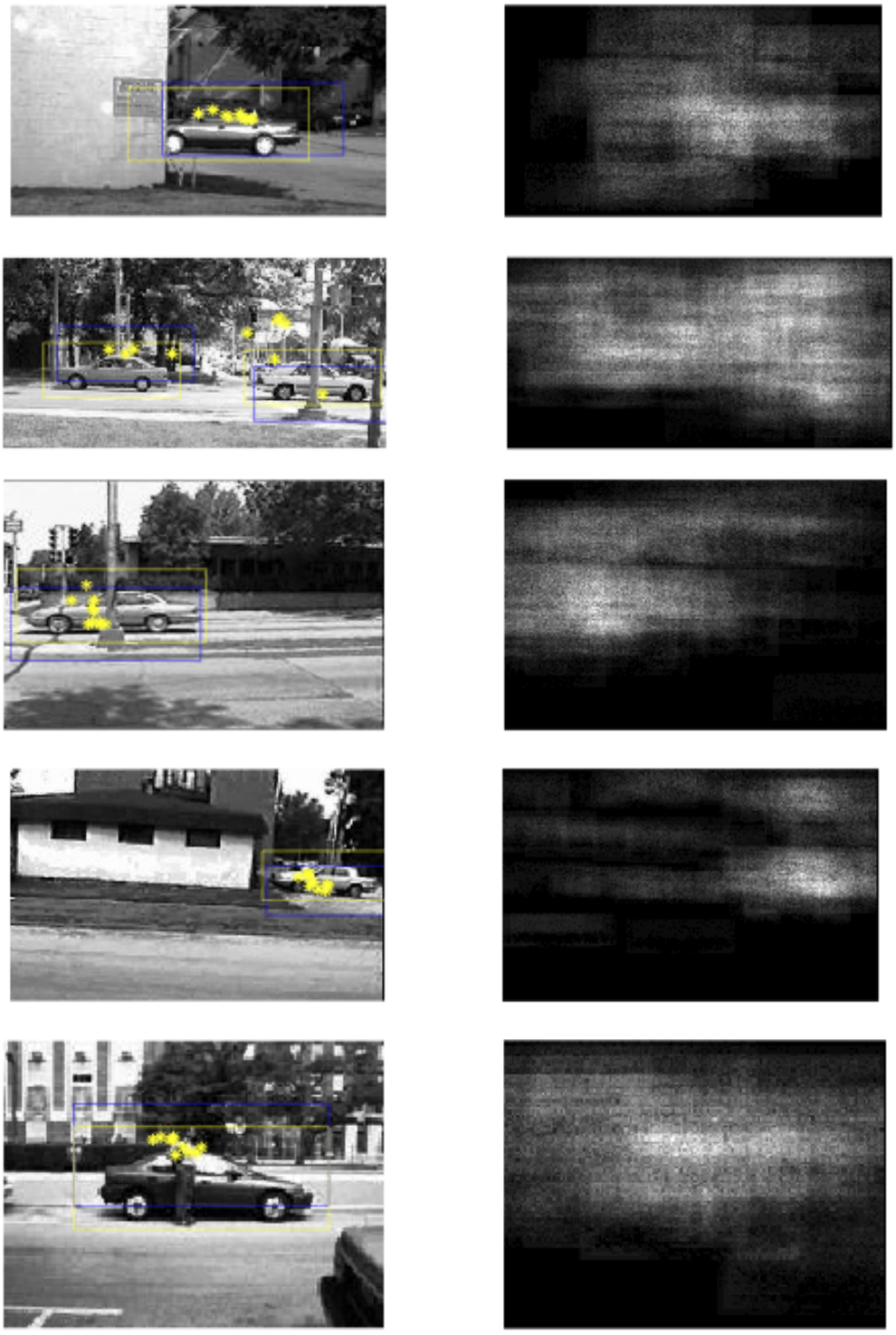

Figure 3: True bounding box(yellow), predicted bounding box(blue) and weighted segmentation mask

Figure 3 shows excellent results as the predicted bounding box matches the ground truth data. The figures show instance of scale variation, occlusion, and multiple objects in the same image which the object detector was able to decipher despite these obstacles which shows the robustness of the algorithm. However the model was not perfect as there was a large number of misprediction as well. Figure 4 shows instances of these mispredictions where either a bounding box was not centered correctly, or it missed a second vehicle, or when there are two cars and the predicted bounding box was thought to be between two vehicles.

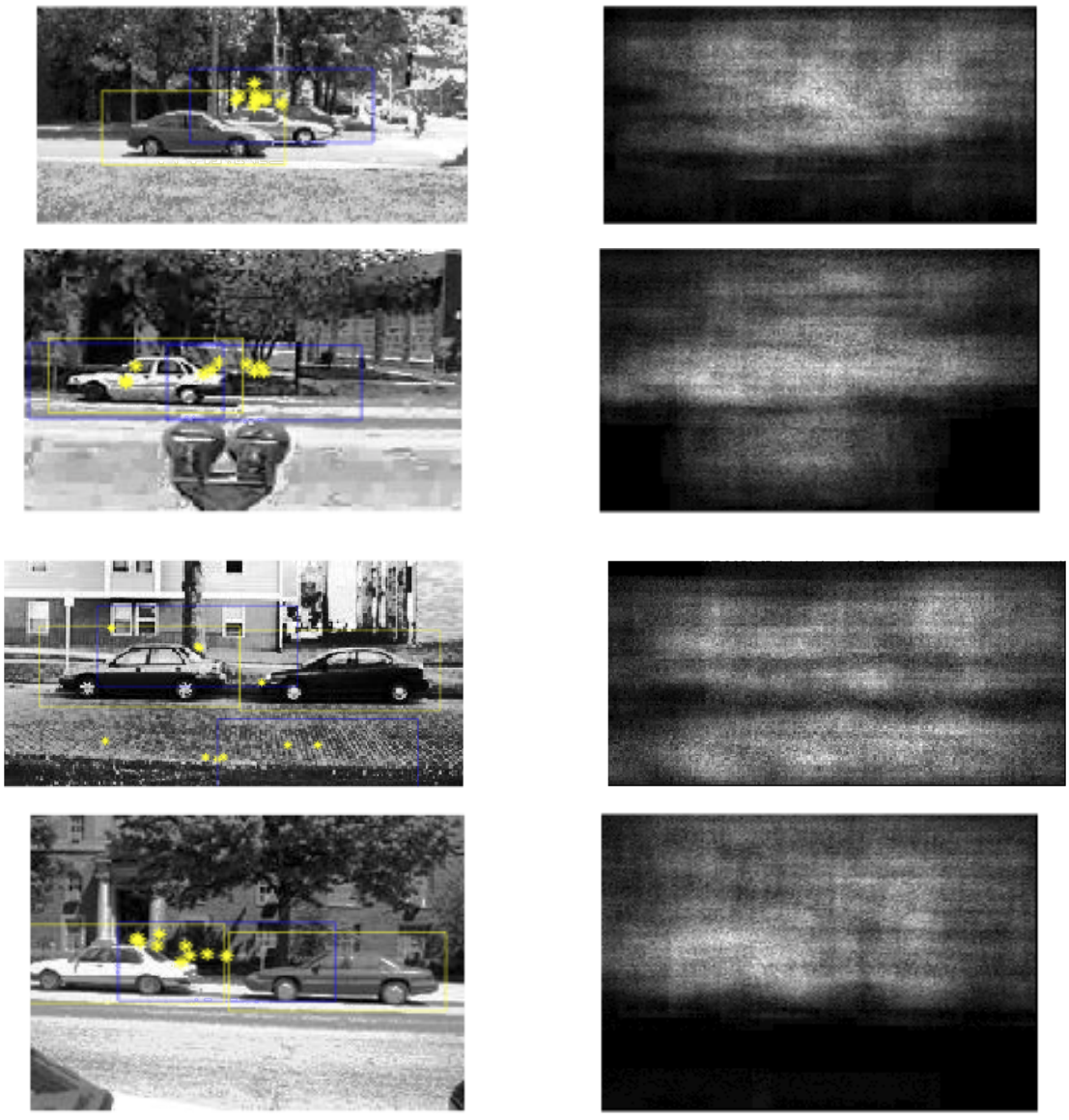

Figure 4: Mispredicted bounding boxes

The sources of these errors are believed to originate from the harris corner detector as it is believed that some key features are overlooked. In some images, other corners are determined to be the object so there appears to be some correlation between those patches and the patches of car words. Lastly, the bounding boxes sometimes gets stuck between two cars which is assumed to be from the box getting confused with the back wheel of one car and the car wheel of another car which constitutes one car. Lastly to determine the overall accuracy of the algorithm, a predicted detection is counted as correct if the area of the intersection of the boxes, normalized by the area of their union exceeds 0.5. The figure below visualizes the calculation.

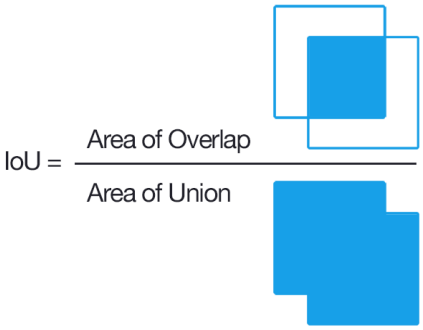

Figure 5: IoU calculation [1]

This was computed by using the rectint MATLAB function to compute the area of overlap between the predicted bounding box and the predicted bounding box and dividing is by the union of the two boxes.

$$IoU = area\_overlap/(area\_pred + area\_ground - (2*overlap)) \quad Eq[2]$$

The final tally of correct predictions was determined to be 62% which leaves a lot of room for improvement and improving the effectiveness of the BoW algorithm

Conclusion:
To summarize, the bag of words object category detection consists of two different stages. In the training stage, a vocabulary of patches i s built around extracted interest points from clustering. Each patch is then mapped around each interest point to the closest word. Finally for each word, all the displacement vectors are stored relative to the object center which is also the center of the training image. In the testing stage, given new test images, corners are determined and then patches are extracted and the patches are matched to vocabulary words. Votes are then cast for possible positions of the object center and the maxima are thresholded for in the voting space. Next the weighted segmentation mask is extracted based on the codebook occurrences and this is the predicted location of the object of interest. The classifier predicted many cars correctly but also struggles at time in images with multiple cars and in images with certain distinct features

that were voted for higher than a vehicle. Some possible sources of error are the effectiveness of the Harris Corner Detector and the possibly non optimal value of K clustering. For future improvements, another method would be to use Deep Learning and a convolutional neural network which looks for the best weights for the convolutional filter and linear classification weights to predict class labels with after training on a ground truth similar to the bag of words method. The project explored a classical computer vision technique of extracting hand crafted features from an image in order to make a prediction.

References

[1]https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

Appendix:

```
clear all; close all; clc;

%load frame paths and put in list
zips = ["CarTestImages", "CarTrainImages"];
test_folder="/Users/johnnguyen/Documents/CV/Projects/Project4/"+zips(1)+"/";
train_folder="/Users/johnnguyen/Documents/CV/Projects/Project4/"+zips(2)+"/";


path = '*.jpg';
im_test = dir(test_folder+path);
im_train = dir(train_folder+path);
numTest = length(im_test);
numTrain = length(im_train);

window_size=7;

for i = 1:numTrain
   fprintf("Training example %d loaded\n",i)
   RGB = imread(train_folder+im_train(i).name);
   train_data(i).OG_RGB = RGB;
   c = detectHarrisFeatures(RGB);
   train_data(i).RGB = RGB;
   [patches,xywh,xy] = patch_extractor(RGB,round(c.Location),window_size);
   train_data(i).corner_coordinates = xy;
   train_data(i).patches = patches;
   train_data(i).xywh = xywh;
end
center_row = size(RGB,1)/2;
center_col = size(RGB,2)/2;

% if want to concatenate all images along 3rd dimension... then cat(3,train_data.RGB)
%%
i = 1;
RGB = train_data(i).OG_RGB;
```

```
X = train_data(i).corner_coordinates;
patches = train_data(i).patches;
patch_xywh = train_data(i).xywh;
figure(1)
imshow(RGB)
hold on
plot(X(:,1),X(:,2),'y*')

% plot corners
for xywh = patch_xywh'
   rectangle('Position',xywh)
end
%% K-means
patches = double(cat(3,train_data.patches));

% resize patches for k-means
patches = reshape(patches,(window_size*2+1)*(window_size*2+1),size(patches,3));

k_fit = [];
for k = 1:15
   [~,~,sumd] = kmeans(patches',k);
   k_fit = [k_fit mean(sumd)];
end
%% elbow inspection
% figure(2)
% plot(1:15,k_fit)
% title("Elbow Inspection")
% xlabel("K-clusters")
% by visual inspection I think k=4 is ideal
%% refit K-means with ideal
K = 6
[c_idx,c_mean,~,D] = kmeans(patches',K);
%% visualize patches
% for patch = c_mean'
%    figure
%    imshow(reshape(uint8(patch),(window_size*2+1),(window_size*2+1)))
% end
%% assign labels to patches
n_start = 1;
```

```
for i = 1:numTrain
    n_patches = size(train_data(i).patches,3);
    train_data(i).vocab_idx = c_idx(n_start:(n_start+n_patches-1));
    n_start = n_start + n_patches;
end
%% record distances to center
for i = 1:numTrain
    train_data(i).displacement_vecs = train_data(i).corner_coordinates-[center_col,center_row]
end
%% displacement table
for k = 1:K
    idx = c_idx==k
    displacement_vec = cat(1,train_data.displacement_vecs);
    displacement_table{k} = displacement_vec(idx,:);
end
%% test time
for i = 1:numTest
    fprintf("Testing example %d loaded\n",i)
    RGB = imread(test_folder+im_test(i).name);
    test_data(i).OG_RGB = RGB;
    c = detectHarrisFeatures(RGB);
    test_data(i).RGB = RGB;
    [patches,xywh,xy] = patch_extractor(RGB,round(c.Location),window_size);
    test_data(i).corner_coordinates = xy;
    test_data(i).patches = patches;
    test_data(i).xywh = xywh;
end
center_row = size(RGB,1)/2;
center_col = size(RGB,2)/2;
% if want to concatenate all images along 3rd dimension... then cat(3,train_data.RGB)
%% assign labels to patches
for i = 1:numTest
    fprintf("Test point assignment %d\n",i)
    patch_img = test_data(i).patches;
    vocab_idx = [];
    for j = 1:size(patch_img,3)
        distances =
sum(sqrt((c_mean-double(reshape(patch_img(:,:,j),1,(window_size*2+1)*(window_size*2+1))))
.^2),2);
```

```
        [min_distance,k_pred] = min(distances);
        vocab_idx = [vocab_idx k_pred];
    end
    test_data(i).vocab_idx = vocab_idx;
end
%% voting time
tally = 0
for i = 1:numTest
    img = test_data(i);
    voting_scheme = zeros(size(img.RGB));
    xy = img.corner_coordinates;
    vocab_id = img.vocab_idx;
    for j = 1:size(xy,1)
        r = xy(j,2);
        c = xy(j,1);
        displace = displacement_table{vocab_id(j)};
        r = r + displace(:,2);
        c = c + displace(:,1);
        erase = ((r>0)&(c>0)) & (c<size(voting_scheme,2)) & (r<size(voting_scheme,1));
        idx = sub2ind(size(voting_scheme),r(erase),c(erase));
        voting_scheme(idx) = voting_scheme(idx)+1;
    end
    close all
    figure(3)
    minimum = min(voting_scheme(:));
    [maximum,idx] = maxk(voting_scheme(:),10);
    maximum = max(maximum);

    [y,x] = ind2sub(size(voting_scheme),idx);
    subplot(1,2,1)
    imshow(img.RGB)
    hold on
    plot(x,y,'y*')

    rectangle('Position',[max([x-50 y-20]) 100 40],'EdgeColor', 'b')


    for p =1:size(x,1)
        if abs(max(x)-x(p)) > 50 | abs(max(y)-y(p)) > 30
```

```
        rectangle('Position',[x(p)-50 y(p)-20 100 40],'EdgeColor', 'b')
        break
      end
   end

   for k = 1:size(groundtruth(i).topLeftLocs ,1)
      rectangle('Position',[groundtruth(i).topLeftLocs(k,:),groundtruth(i).boxW,
groundtruth(i).boxH],'EdgeColor', 'y')
   end

   %overlap results
   A = [max([x-50 y-20]) 100 40];
   B = [groundtruth(i).topLeftLocs(1,:),groundtruth(i).boxW, groundtruth(i).boxH]
   area = rectint(A,B);
   den = ((100*40*2)-(area*2))
   iou = round(area/den,2)
   if iou > .5
      tally = tally + 1;
   end
   subplot(1,2,2)
   imshow(voting_scheme/maximum)
%    subplot(1,3,3)
%    imshow(img.RGB)
%    hold on
%
%    for k = 1:size(groundtruth(i).topLeftLocs ,1)
%        rectangle('Position',[groundtruth(i).topLeftLocs(k,:),groundtruth(i).boxW,
groundtruth(i).boxH],'EdgeColor', 'y')
%    end
   %pause
end

result = tally/numTest
%%
figure
img = test_data(4)
imshow(img.RGB)
hold on
plot(xy(:,1),xy(:,2),'y*')
```

```
%% plot ground truth boxes on test data
for i = 1:numTest
   figure
   imgTest = test_data(i)
   imshow(imgTest.RGB)
   hold on
   for k = 1:size(groundtruth(i).topLeftLocs ,1)
      rectangle('Position',[groundtruth(i).topLeftLocs(k,:),groundtruth(i).boxW,
groundtruth(i).boxH],'EdgeColor', 'y')
   end
   pause
end
```

Patch extractor function

```
function [image_patches,xywh,xy] = patch_extractor(RGB,Corners,r_size)
   image_patches = [];
   xywh = [];
   xy = [];
   for c = Corners'
      y = c(1);
      x = c(2);
      try
         patch = RGB((x-r_size):(x+r_size),(y-r_size):(y+r_size));
         bx = y-r_size/2;
         by = x-r_size/2;
         box = [bx,by,r_size,r_size];
      catch
         continue
      end
      image_patches = cat(3,image_patches,patch);
      xywh = cat(1,xywh,box);
      xy = cat(1,xy,[y,x]);
   end
end
```