Assignment3

Name: Jing Liu

Student ID: 18231917

1. Introduction

This assignment majorly divided by three parts, the first part is circle detection and red detection; the second part is feature extraction, the main technical is HOS feature extraction; the third part is image classification, which majorly using Euclidean and SVM classifiers.

2. Circle Detection and Red Detection

For the circle detection, here I using Hough Transform， the code shown as figure 1.1, the function is imfindcircles, here detect circle automatically and then set the radius range, then return to the centre point and radius, according whether or not here has centre points to detect whether or not it is a circle.

For the red detection, firstly here converting the RGB image to HSV image, then through calculating the value of H and S to judge whether here has a red circle area, the threshold is for H is greater than 0.9 or smaller than 0.1 and the S should be smaller than 1.0 and greater than 0.5, then return to a binaries image for red area are white, rest of are black, the coding shown as figure 1.2.

```
function result = circle_detection(image)
    [centers, radius] = imfindcircles(image,[20 60],'ObjectPolarity','bright');
    viscircles(centers, radius,'Color','b');
    result = ~isempty(centers);
end
```

Figure1.1 Circle Detection

```
function outresult = red_detection(imageRGB)
        % Convert RGB image to HSV
        imageHSV = rgb2hsv(imageRGB);
        % Threshold image
        outresult = (imageHSV(:, :, 1) >= 0.9 | imageHSV(:, :, 1) <= 0.1) &...
            (imageHSV(:, :, 2) >=  0.5 & imageHSV(:, :, 2) <= 1.0);
end
```

Figure1.2 Red Detection

The output of image shown as follows 1.3:

Figure1.2 Red Detection Output

3. Image Pre-processing

For the image pre-processing, on the training (development) dataset, there are lots of image are very dark, so here I using HSV and RGB respectively to improve image brightness, for the total number pixels of V or R,G,B if the sum of pixel smaller than 11000, then let the V of HSV improve to 1.8*original pixel. It's same with the RGB to improve the colour so that make the image works better. The second step of image pre-processing is when you finish circle and red area detection, then according the binaries image to extract the digital content inside the red circle, here firstly using remove some small object using bwareopen and imclearborder to remove noises, and then using strel to get the disk shape and then dealing with the image with a set of operation, after clean the image, then using regionprops function to get the attribute we want, here I want get the box bounding of each image and within the circle and the then crop the corresponding image according to the box bounding, here we can get a binaries image within a circle, and using this image as input image.

```
hsv = rgb2hsv(A);
V = hsv(:,:,3);

if sum(V(:))<11000
%1.7062e+05
for i = 1:m
for j = 1:n
hsv(i,j,3) =1.8* hsv(i,j,3);
end
end
end
```

Figure 3.1 Increase brightness

```
exreact_number = imresize(image_extract, [450, 450]);
c = bwareaopen(exreact_number, 50);
c = imclearborder(c);
se = strel('disk', 5);
% imerode and imdilate work better
c = imerode(c, se);
c = imdilate(c, se);
% remove objects
cc = bwconncomp(c);
label_matrix = labelmatrix(cc);
label_extent = regionprops(cc, 'Extent');
indexe = [label_extent.Extent] >= 0.3;
mask = ismember(label_matrix, find(indexe));
mask=bwareafilt(mask,2);
statsBB = regionprops(mask, 'BoundingBox');
if ~isempty(statsBB)
   count=count+1;
   % Extract the ROI for the left digit by setting the index of the object
   % in the labelled image to '1'
   bbox = statsBB(1).BoundingBox;
   digitROI = mask(int16(bbox(2)):int16(bbox(2)+bbox(4)), int16(bbox(1)):int16(bbox(1)+bbox(3)),:)
```

Figure 3.2 Image pre-processing

The output image shown as figure 3.3:

Figure 3.3 Image pre-processing Output

4. Feature Detection

For the feature detection, here I only use HOG feature extraction, the method of extracting the HOG feature is to firstly divide the window image into cells and then divide the cell into blocks, Then, the direction gradient of each pixel in the block is extracted and the histogram is synthesized. Finally, the histograms of these blocks are combined to obtain the Hog feature of the image, the Figure4.1 and 4.2 is the source of HOG, and the output features looks like figure 4.3.Comapre with using the HOG feature detection and without using HOG feature detection, such as for the 20Kph, the accuracy improved from 77% to 79%, but for other image class such as 50Kph, the accuracy decreased.

```
function H=HOG(Im)
        nwin_x=3;%set here the number of HOG windows per bound box
        nwin_y=3;
        B=9;%set here the number of histogram bins
        [L,C]=size(Im); % L num of lines ; C num of columns
        H=zeros(nwin_x*nwin_y*B,1); % column vector with zeros
        m=sqrt(L/2);
        if C==1 % if num of columns==1
            Im=im_recover(Im,m,2*m);%verify the size of image, e.g. 25x50
            L=2*m;
            C=m;
        end
        Im=double(Im);
        step_x=floor(C/(nwin_x+1));
        step_y=floor(L/(nwin_y+1));
        cont=0;
        hx = [-1,0,1];
        hy = -hx';
        grad_xr = imfilter(double(Im),hx);
        grad_yu = imfilter(double(Im),hy);
        angles=atan2(grad_yu,grad_xr);
        magnit=((grad_yu.^2)+(grad_xr.^2)).^.5;
```

Figure 4.1 feature extraction1

```
for n=0:nwin_y-1
    for m=0:nwin_x-1
        cont=cont+1;
        angles2=angles(n*step_y+1:(n+2)*step_y,m*step_x+1:(m+2)*step_x);
        magnit2=magnit(n*step_y+1:(n+2)*step_y,m*step_x+1:(m+2)*step_x);
        v_angles=angles2(:);
        v_magnit=magnit2(:);
        K=max(size(v_angles));
        %assembling the histogram with 9 bins (range of 20 degrees per bin)
        bin=0;
        H2=zeros(B,1);
        for ang_lim=-pi+2*pi/B:2*pi/B:pi
            bin=bin+1;
            for k=1:K
                if v_angles(k)<ang_lim
                    v_angles(k)=100;
                    H2(bin)=H2(bin)+v_magnit(k);
```

Figure 4.1 feature extraction2

Figure 4.3 HOG Features Extraction Output

5. Classifier

5.1.Distance Classifier

For the distance classifier, here I tried Consine and Euclidean distance, the results is Eudliean works better, the distance function shown as figure 5.1, here is two way to calculate the distance, firstly is using the image matrix, read all training images and gold images after pre-processing, then you will get the matrix of the content of each speed limit sign, then calculate the distance of each image matrix with the image matrix of gold standard image, then compare all distance and get the smaller one with gold image, for the shortest distance, get the standard image label and compare with the training image label, then get the accuracy, different class image accuracy is definitely different, the performance of this system works better for 20Kph and poor for 50Kph, the accuracy of 20 Kph with image matrix is 0.77, another way is using HOS feature extraction, here the image matrix convert to feature vector, and the accuracy improved to 0.7905. The result shown as figure 5.2.

```
function speedLimit=distance_classify(digitROI)
         % Load digits extracted from the gold standard images
        myFolders='C:\Users\Owner\Desktop\mini-pro\result2';
        filePattern=fullfile(myFolders,'*.jpg');
        jpgFiles=dir(filePattern);
        for k = 1:length(jpgFiles)
            baseFileName=jpgFiles(k).name;
            filePath=fullfile(myFolders,baseFileName);
            goldImage = imread(filePath);
            goldImage=HOG(goldImage);
            % Calculate the percentage of black pixels in the binary image
            euclidean_dist(k) = sqrt(sum((im2double(digitROI(:))- im2double(goldImage(:))).^ 2));
            %cosine_dis(k) = pdist2(digitROI(:),goldImage(:),'cosine');
        end
        % The output image with the highest percentage of black pixels is the
        % best match
        [~, index] = min(euclidean_dist);
        % Return string of the best matched speed
        speedLimit = jpgFiles(index).name(1:end-2);
    end
```

Figure 5.1 Distance Classifier

```
>> mini_pro
accuracy for 20Kph without feature extraction:
    0.7762

accuracy for 20Kph with feature extraction:
    0.7905

accuracy for 30Kph without feature extraction:
    0.6394

accuracy for 30Kph with feature extraction:
    0.4727

accuracy for 50Kph without feature extraction:
    0.3697

accuracy for 50Kph with feature extraction:
    0.1939
```

Figure 5.2 Output of Euclidean Distance

5.2. SVM classifier

SVM is another classifier I used, the code shown as figure 5.3, SVM is a classification, for this part, firstly I read the image from dataset and then split it to training set and testing set with 8:2, then convert it to the HOG features and then feed all image HOS feature vector as the input vector, then call the function fitcecoc of SVM classifier and after training the image then using testing data to predict it, then compare the real label of testing and predict label of testing and get the final accuracy , the final accuracy shown as figure 5.4 is 81.86% .

```
function SVM_classify()
    imds = imageDatastore('C:\Users\Owner\Desktop\mini-pro\total2', 'IncludeSubfolders', true,
    tbl = countEachLabel(imds);
    [train,test] = splitEachLabel(imds, 0.8,'randomize');
    %trainingFeatures = extract_feature(train)
    trainLebal=train.Labels;
    train_num= numel(train.Files);
    vectors_train= zeros(train_num, 81);
    for i=1:train_num
        image = readimage(train,i);
        vectors_train(i,:)=HOG(image);
    end
    classifier = fitcecoc(vectors_train,trainLebal);
    testLabels=test.Labels;
    test_num=numel(test.Files);
    vectors_test=zeros(test_num, 81);

     for i=1:test_num
        image = readimage(test,i);
        vectors_test(i,:)=HOG(image);
    end
    predictedLabels =  predict(classifier,vectors_test);
    accuracy = sum(predictedLabels == testLabels)/numel(testLabels);
    disp(accuracy)
end
```

Figure 5.3 SVM Classifier

```
>> mini_pro
the accuracy for speed limit sign detection with SVM:
    0.8186
```

Figure 5.4 SVM Accuracy

6. Result

6.1. Development image result

This is all outputs for two different classifiers with HOG features and image pixel. The figure 6.1 is the output of gold standard, the processing way is keep all digits and white circle, the training development image is same with gold standard.

Figure 6.1 Gold Standard

The figure 6.2 is showing the Euclidean distance accuracy for 20Kph,30Kph and 50Kph respectively, from the result I can get that the 20Kph have the highest accuracy while the 50 get the poorest accuracy, compare with image pixel and HOS features, the 20Kph is improved while the accuracy of 30Kph and 50Kph is reduced.

```
>> mini_pro
accuracy for 20Kph without feature extraction:
    0.7762

accuracy for 20Kph with feature extraction:
    0.7905

accuracy for 30Kph without feature extraction:
    0.6394

accuracy for 30Kph with feature extraction:
    0.4727

accuracy for 50Kph without feature extraction:
    0.3697

accuracy for 50Kph with feature extraction:
    0.1939
```

Figure 6.2: Result of Euclidean Distance with HOG and without HOG respectively

The figure 6.3 is the accuracy of SVM classifier; from the result we can see that compare with Euclidean distance, the SVM has a better performance on average.

```
>> mini_pro
the accuracy for speed limit sign detection with SVM:
    0.8186
```

Figure 6.3: Result of SVM classifier with HOG feature

6.2. Street Speed Limit Sign Result

The street speed limit sign is firstly detect the circle, here are 5 images didn't works, the system cannot detect the circle within the image, and rest of 11 images can be detected successfully, for the image can be detect, there are 7 image are recognized successfully, the figure 6.4 below show the code for how to pre-processing image and detect circle and recognized. Here also firstly read image from dataset and then detect the image dimension, and then give a set of processing to get image and crop image with suitable box bounding, lastly convert it to HOS feature and calculate distance, for the street speed limit sign, the accuracy is 77.78%, which shown as figure 6.5.

```
function street_detection()
    imds = imageDatastore('C:\Users\Owner\Desktop\mini-pro\stress dataset', 'IncludeSubfolders', t
    label={'01','01','01','03','02','02','02','01','01','01','01','01','01','01','01','02'};
    street_acc=0;
    canbe_re=0;
    prediction={};
    for s = 1:numel(imds.Files)
        image = readimage(imds,s);
        if(size(image)==[480   640     3])
            imagelight_resize = imresize(image,[450 450]);
            img_gray= rgb2gray(image);
            x =  imsubtract(image(:,:,1), img_gray);
            x = medfilt2(x, [3,3]);
            x = imbinarize(x, 0.18);
            x = bwareaopen(x, 300);
            x = imresize(x, [450 450]);
            cc = bwconncomp(x);
                stats = regionprops(cc, 'BoundingBox');
            if size(stats) ~=[0 0]
                 bbox = stats.BoundingBox;
                 digitROI = imagelight_resize(int16(bbox(2)):min(int16(bbox(2)+bbox(4)), 450),...
                 int16(bbox(1)):min(int16(bbox(1)+bbox(3)), 450), :);
                 image_street= imbinarize(rgb2gray(digitROI));
                 number_street = imresize(image_street, [160, 120]);
```

Figure 6.4 coding for street speed limit sign extraction

```
the accuracy for street speed limit sign detection:
    0.7778
```

Figure 6.5 Accuracy of street speed limit sign detection

6.3. Problems

Here also get some problems, for some of image which are too blurred or brightness is not well, the digits identification effect is not very well even if try to pre-processing it. For example:

Figure 6.6 original(left) and after processed(right)

Figure 6.7 original(left) and after processed(right)

In conclusion, speed limit sign detection system can recognize most of images, the HOS feature extraction works better than normal image pixel matrix in some degree. For the classifier, the SVM is better than distance algorithm, which improved almost 10% on average.

Reference:

1. https://uk.mathworks.com/matlabcentral/answers/343645-how-to-find-the-euclidean-distance-between-two-images Euclidean Distance
2. https://uk.mathworks.com/matlabcentral/fileexchange/28689-hog-descriptor-for-matlab HOG descriptor for Matlab
3. https://blog.csdn.net/u013476464/article/details/41595209 how to random dataset
4. https://uk.mathworks.com/help/images/ref/imfindcircles.html Imfindcircle
5. https://uk.mathworks.com/help/matlab/ref/rgb2hsv.html convert RGB to HSV
6. https://uk.mathworks.com/help/images/ref/bwareaopen.html bwareaopen
7. https://uk.mathworks.com/help/images/ref/imclearborder.html imclearborder
8. https://uk.mathworks.com/help/images/ref/imdilate.html `imdilate`
9. https://uk.mathworks.com/help/images/ref/imerode.html imerode
10. https://uk.mathworks.com/help/images/ref/bwareafilt.html bwarefilt
11. https://uk.mathworks.com/help/images/ref/regionprops.html `regionprops`
12. https://uk.mathworks.com/help/stats/fitcecoc.html SVM classfier
13. https://uk.mathworks.com/help/matlab/ref/matlab.io.datastore.imagedatastore.spliteachlabel.html splitdataset