PROJECT-4 REPORT

UNIVERSITY OF MARYLAND

A. JAMES CLARK SCHOOL OF ENGINEERING

SUBMITTED BY:

ABHISHEK KATHPAL

UID: 114852373

SUBMITTED TO:

PROF. CORNELIA FERMÜLLER

CONTENTS

1) OVERVIEW OF PIPELINE

The pipeline of Traffic Sign Recognition Project can be divided into two sections- Traffic Sign Segmentation and Traffic Sign Classification. For Segmentation part, I used the Maximally Stable Extremal Region (MSER) pipeline as mentioned in the Project Guidelines. For the classification part, first the HOG Features of Training Data are extracted and using those features a multi-class support vector machine classifier is trained and then this classifier is tested using the test dataset provided. This classifier is further used for classifying the signs among the 8 traffic signs as shown in Fig. 1. The details of pipeline are discussed in the next sections.

2) TRAFFIC SIGN SEGMENTATION

The recognition of traffic signs is the most important part of this project. Initially, I tried using the HSV Color Space for thresholding the regions with traffic signs. But I was getting a lot of noise and because of that, my classifier is taking a long time to process each frame. So, I switched to another pipeline i.e. extracting Maximally Stable Extremal Region.

2.1) MSER ALGORITHM

For detection of MSER Regions, the steps followed are discussed in detail below:

1. **Noise Removal and contrast Normalization**
   For removing noise, “imgaussfilt” function is used with parameter sigma as 0.2. For contrast normalization, “strechlim” function is used with default parameters.

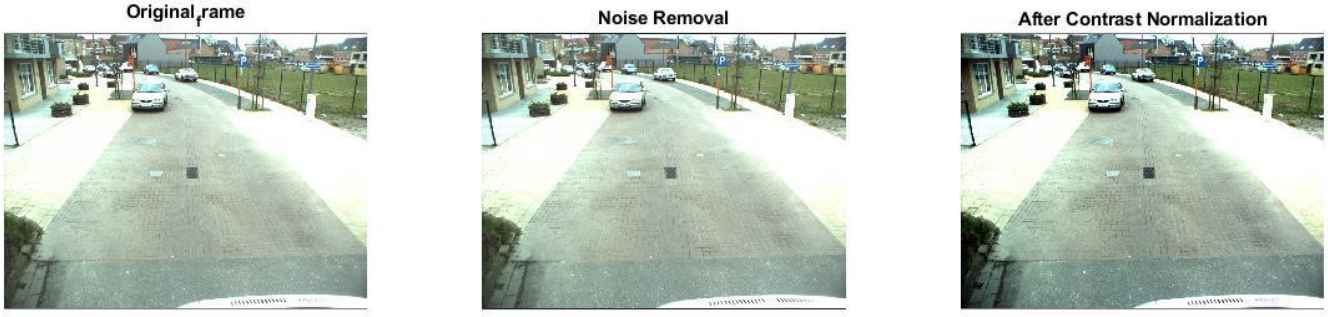

Fig. 1. Noise Removal and Contrast Normalization

2. **Normalizing Intensity of Image**
   The next step is normalizing the intensities. As mentioned in the project guidelines, I used those equations but I was getting better result without dividing the (R+G+B) factor. So, the equations I used for this step are –

   A. C = max (min (R-G, R-B),0) - for red signs
   B. C = max (min (B-G, B-R),0) - for blue signs

The output after this step is shown in Fig. 2. You can clearly see the Parking sign board in blue region and in red region image the back of one of the traffic signs.

Fig. 2. a) Blue Region and b) Red Region after Intensity Normalization

3. MSER Parameter Tweaking

The next step is to "detectMSERFeatures" function to get the MSER Regions. This step requires parameter tweaking. There are two important parameters – Threshold Delta and RegionAreaRange. If thresholdDelta is too low, it will give a lot of unwanted regions and if ThresholdDelta is too high, It will give very less regions. For blue and red traffic signs, I have used the following parameters –

A. Blue – "RegionAreaRange" = [400,4000] and "ThresholdDelta" = 2
B. Red – "RegionAreaRange" = [500,4000] and "ThresholdDelta" = 3

The output for red and blue MSER Regions is shown in Fig. 3.

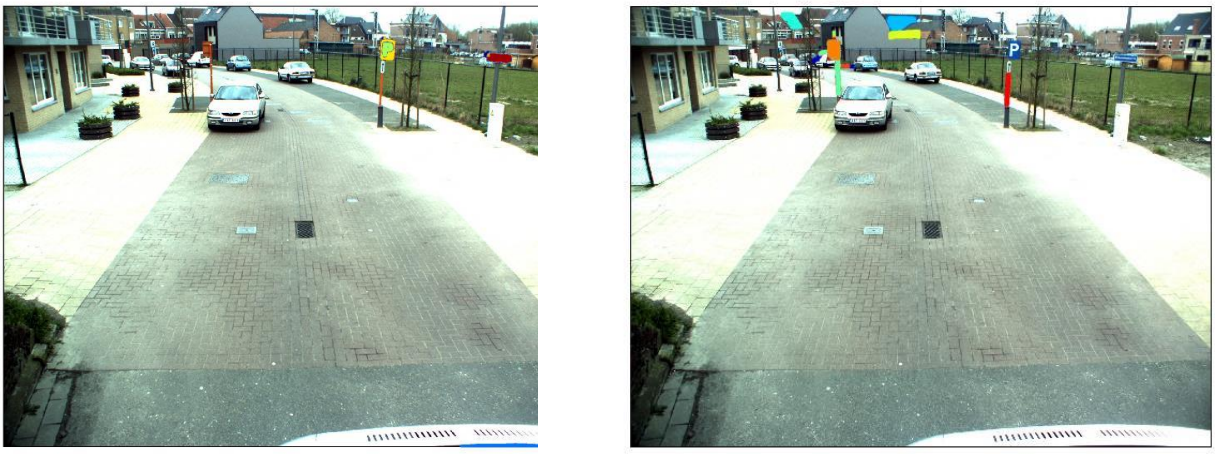

Fig. 3. A) Blue MSER Regions b) Red MSER Regions

To reduce the noise further, I have used the following tricks:

1) Limiting Aspect Ratio between 0.6 to 1.3. This is according to the paper mentioned in the Project Guidelines pdf.
2) To avoid overlapping boxes, initially I tried to implement the selectStrongestBbox MATLAB function which uses non-maximal suppression but later I switched to a more simpler approach that you can see from the code. Removing the addition of box if the corner point is closer to previously detected bounding box.

   The code for the MSER Extraction is shown below:

```
function bbox = mser_regions(img,str)

R = im2double(img(:,:,1));
G = im2double(img(:,:,2));
B = im2double(img(:,:,3));

%Danger and Prohibitory Signals
if str == "red"
red_norm = max(min(R-B,R-G),0);
[~, mserConnComp] = detectMSERFeatures(red_norm, ...
    'RegionAreaRange',[500 4000],'ThresholdDelta',3);
s = regionprops(mserConnComp, 'BoundingBox');
bbox = cell(1,0);
j = 1;
a = 1;
b = 1;
for k = 1:length(s)
    temp = s(k).BoundingBox;
    aspectRatio = temp(3)/temp(4);
    if temp(3) > 20 && temp(4) > 20 && aspectRatio < 1.3 && aspectRatio > 0.8
       if (temp(1)-a)>20 && (temp(2)-b)>20
        a = temp(1); b = temp(2);
        bbox{j} = s(k); j = j+1;
       end
    end
end

end
```

Fig. 4. MSER Red Regions Code

3) TRAFFIC SIGN CLASSIFICATION

After extracting the regions, the next goal is to train the classifier as per as the given training dataset and then use that classifier to find which sign it belongs. To train the classifier, first HOG features are extracted as shown in the fig below. Then, using MATLAB's inbuilt "fitceoc" function is used to train the classifier. Then I tested the classifier on the 8 folders of the test data and computed the confidence matrix. The code for training classifier is shown in Fig. 5.

After training the classifier, I combined the steps of MSER and classification to get the required output. For blue signs, I made a blue box around them, their label is printed above them. Similarly, for red signs, a red box is there and I am showing an image along side the sign as mentioned in the project guidelines.

```
clc;clear;close all;

cd ../training;
trainingSet = imageDatastore(pwd, 'IncludeSubfolders', true, 'LabelSource', 'foldername');
num = numel(trainingSet.Files);

trainingLabels=trainingSet.Labels;

cd ../code;

for i=1:num
    img=readimage(trainingSet,i);
    img=imresize(img,[64 64]);
    trainFeatures(i,:) = extractHOGFeatures(img,'CellSize',[8 8]);
end

classifier_8x8 = fitcecoc(trainFeatures,trainingLabels);
save('classifier_8x8.mat', 'classifier_8x8');
```

Fig. 5. Train Classifier using fitceoc

The final output for red and blue signs are shown in Fig. 6.

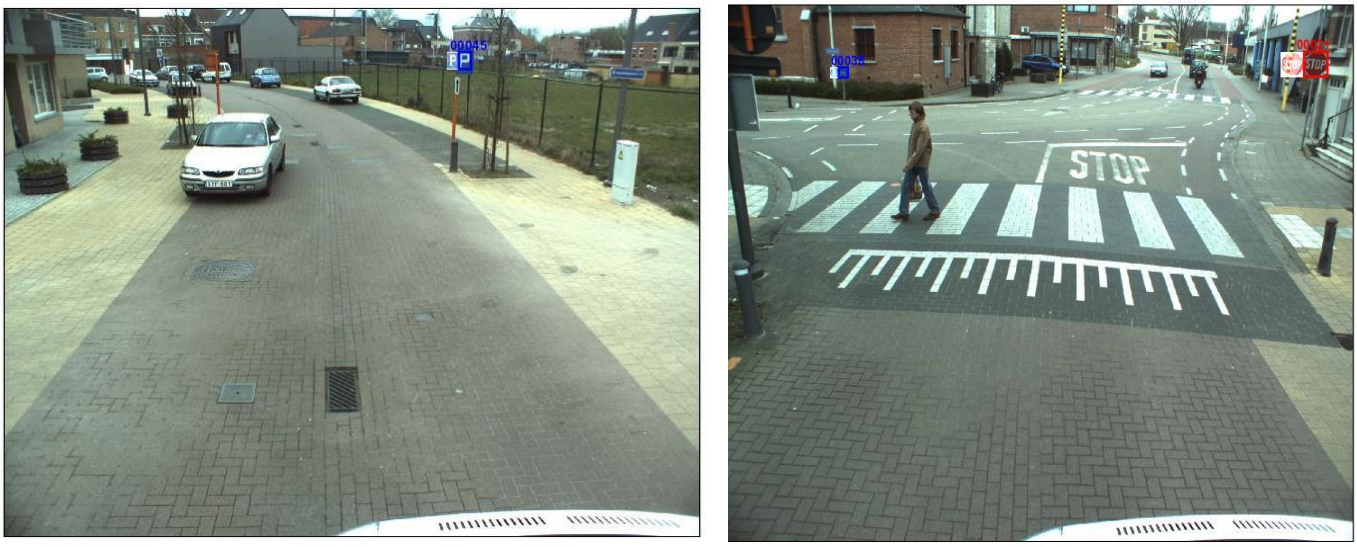

Fig. 6. a) final blue detection b) red stop detection

4) REFERENCES

1) Matlab tutorial for Text Recognition
2) Matlab documentation for detectMSERFeatures
3) Matlab tutorial for Digit Classification using HOG features
4) MSER Perception Lecture Slides
5) Traffic Sign Dataset