Karen Escareno
861205811
March 13, 2019

Street Sign Detection

The problem that the street sign detector would help with is to help avoid accidents from happening. Drivers tend to get distracted by many things (Talking, Technology, etc.), that having help would help avoid cars from passing stop signs or not adhering to the street signs. There are many accidents that are caused by drivers not noticing the speed limit or running a stop sign. This can all be avoided by having our cars do it for us. They would be able to notice the signs in time and act according to what sign the car has recognized.

The significance of the street sign detector is that it would help avoid cars from getting into accidents because of distractions which cause people to miss street signs. There are over 1.6 million crashes annually because of people using their cell phones. This would help lower the number because when the driver is distracted it would take over is the driver was to almost miss a stop sign or going over the speed limit. It would see signs like yield or slow down and be more precautious while the driver may have missed it which could cause it to lead to an accident. This could drastically reduce the number of accidents of people who are distracted or just may have missed a sign by accidents so that it doesn't cause it to lead to a serious accident.

The approach that I used was that it would take in an RGB image. It would then sort through the colors which in this case I used Red (stop and yield signs), Green (street names and freeway signs), Yellow (School, slow down and slipping when wet signs), White (U-turn and turn right signs).It would set a Low and High Hue, Saturation and Value so that it would filter out the color that we want. We would convert the RGB image to HSV which we would then mask with the Low and High values to get the right colors that we want. We would take the masks and multiply with the R,G,B of the original image to get the desired color we want.

We would combing them all together to get the desired results. We would then binarize the image to be able to determine the shapes within the image. Once we have the binary image we would use connected components to combine the shape in case of a separation. We would then blob measurements and their properties. The blob would give us the shapes we are looking for but with small colors outside caused by noise we filter out areas that are lower than a certain value so that we get only the large values we need. The larger values are the polygons which are typically larger than 800-20,000. The values vary because the areas of the polygons are different for each image. Based on the area we are able to see what type of street sign it is and we are also able to but a rectangle around the desired sign.

The data that we first obtained were when testing the High/Low Hue, Saturation and Value. We had to adjust the values so that we could get the colors we wanted since the colors would vary based on the environment and the type of lighting. We also obtained data when finding the area which also had to be modified because there were small details that were found when we outputted the color from the original image since there were small hints in the background. I created an if loop to get rid of the smaller areas because they were not relevant and so the larger areas were the street signs which helped find the street signs quicker. The data also helped determine the street signs based on the color and the area of the blobs we have.

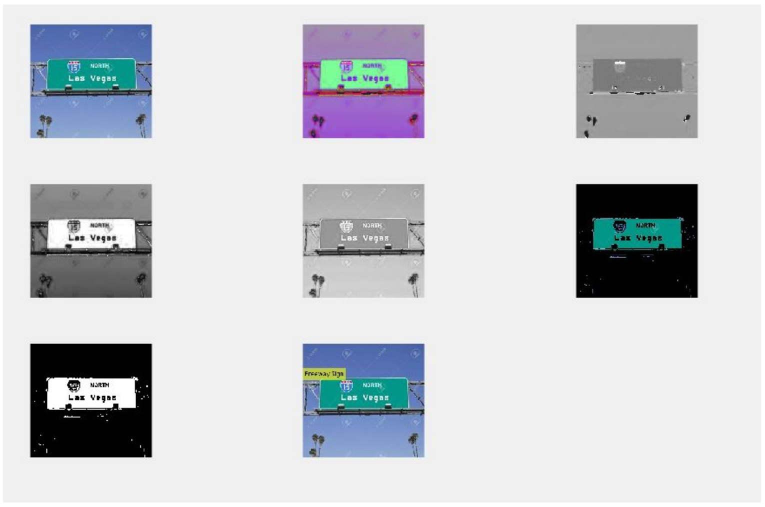
The results that we get are able to get are the HSV image, the hue,saturation and value images separately. We are also able to see the masked HSV with the original image R,G,B combined. We were also able to see the binarized image which was then how we determined the blob which contained the street sign. We then outputted out last image which was our original image with the blob (street sign boxed) which was labeled with the designated street sign that it corresponds to.

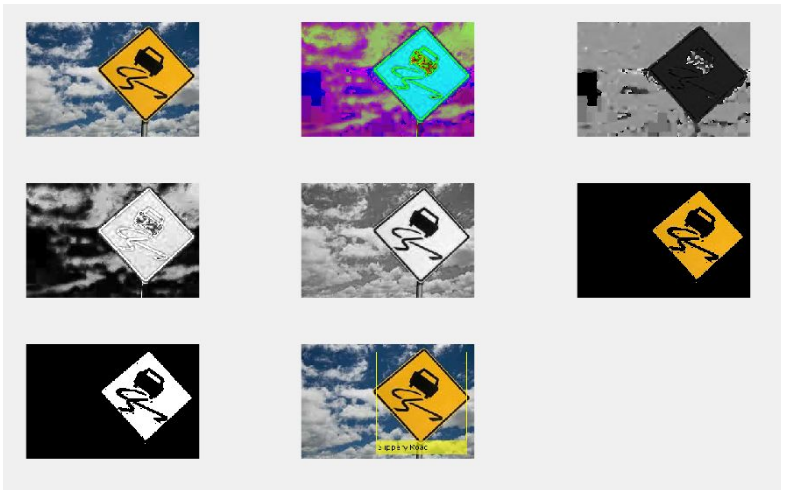
In conclusion, we were able to detect seven different types of street signs based on the color and of the area of the shape. We notice that there is a small error rate because not all stop signs and yield signs are doing to have the same area which also affects the perimeter and circularities which is why area was the best choice. It can sometimes cause the desired box to be in the incorrect place since it depends how large it is in the image we have acquired. Overall, we were able to find seven different street signs that were also the same color and distinguish them by the area of the shapes we found.

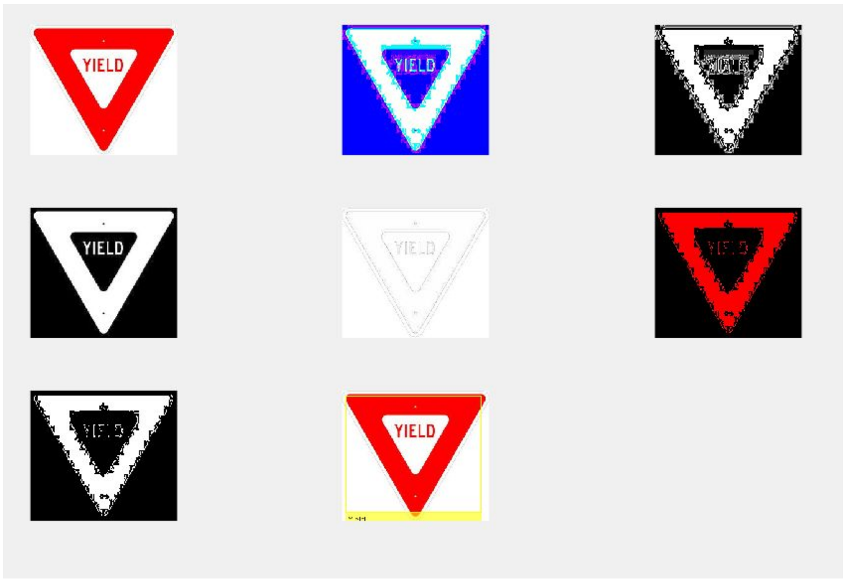
References:

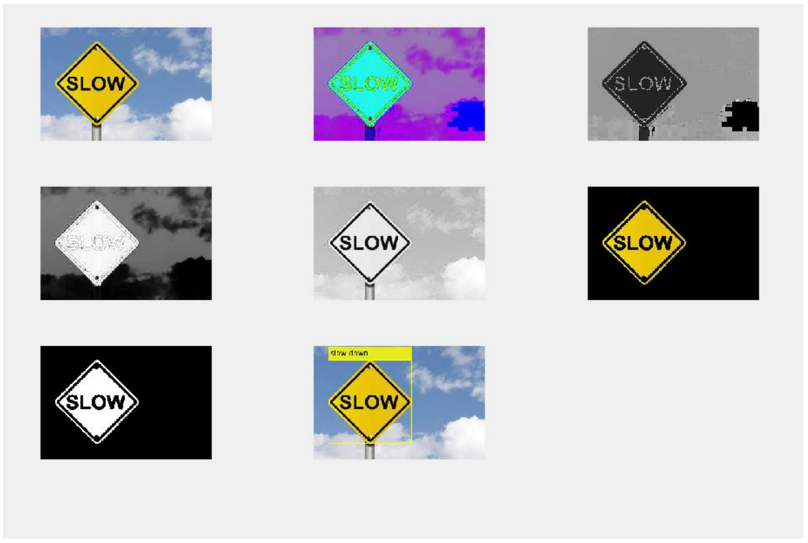
- Traffic Warning Sign Recognition. https://www.mathworks.com/help/vision/examples/traffic-warning-sign-recognition.html
- trainCascadeObjectDetector. https://www.mathworks.com/help/vision/ref/traincascadeobjectdetector.html

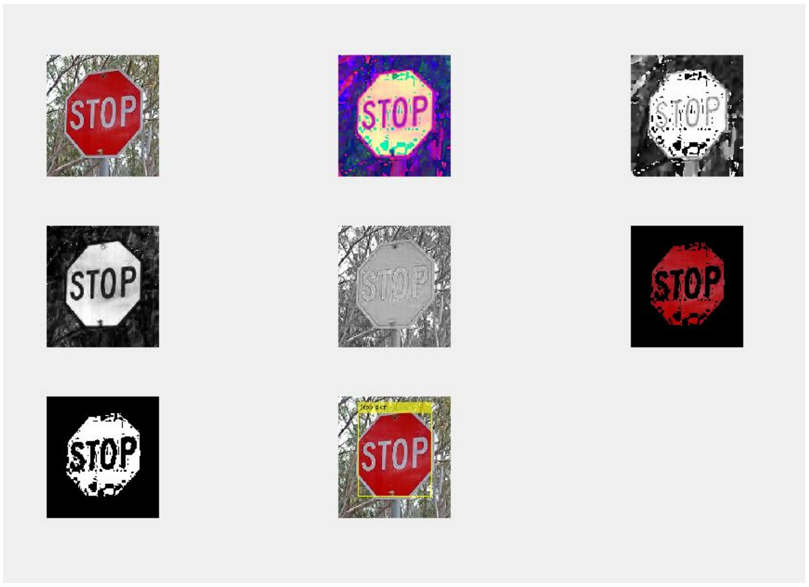
Results:








slow down



Code:

```
I = imread('slow.jpeg');
figure
subplot(4,3,1)
imshow(I)

choice = menu('Choose a color','Red','Yellow','Green','White')
switch choice
    case 1
        findred(I)
    case 2
        findyellow(I)
    case 3
        findgreen(I)
    case 4
        findwhite(I)

end
function findwhite(I)
        hueL = 0.0;
        hueH = 1;
        satL = 0;
        satH = 0.36;
        valL = 0.7;
        valH = 1.0;
originalImage = rgb2gray(I);
[r,c,z] = size(I)
hsv = rgb2hsv(I);
subplot(4,3,2)
imshow(hsv)
hue = hsv(:,:,1);
sat = hsv(:,:,2);
val = hsv(:,:,3);
subplot(4,3,3)
imshow(hue)
subplot(4,3,4)
imshow(sat)
subplot(4,3,5)
imshow(val)
%Set mask using the color HSV limits for colors
hueMask = (hue >= hueL) & (hue <= hueH);
saturationMask = (sat >= satL) & (sat <= satH);
valueMask = (val >= valL) & (val <= valH);
J = uint8(hueMask & saturationMask & valueMask);

maskedImageR = J .* I(:,:,1);
maskedImageG = J .* I(:,:,2);
maskedImageB = J .* I(:,:,3);
RGB = cat(3, maskedImageR, maskedImageG, maskedImageB);
subplot(4,3,6)
imshow(RGB)

G = rgb2gray(RGB);
BW = imbinarize(G);
subplot(4,3,7)
imshow(BW)
Image = bwlabel(BW, 8);
```

```
blobMeasurements = regionprops(Image, originalImage, 'all');
    % Display the image with informative caption.
    Area = cat(1, blobMeasurements.Area);
    [r,c] = size(Area)
    for r2 = 1:r
      if(Area(r2,1) < 500)
        Area(r2,1) = 0;
      end
    end
    Area
    for r3 = 1:r
      if(Area(r3,1) > 1000)
        subplot(4, 3, 8);
        I = insertObjectAnnotation(I,'rectangle',blobMeasurements(1).BoundingBox,'Turn sign');
        imshow(I)

      end
    end
end

function findgreen(I)
      hueL = 0.15;
      hueH = 0.60;
      satL = 0.36;
      satH = 1;
      valL = 0;
      valH = 0.8;
originalImage = rgb2gray(I);
[r,c,z] = size(I)
hsv = rgb2hsv(I);
subplot(4,3,2)
imshow(hsv)
hue = hsv(:,:,1);
sat = hsv(:,:,2);
val = hsv(:,:,3);
subplot(4,3,3)
imshow(hue)
subplot(4,3,4)
imshow(sat)
subplot(4,3,5)
imshow(val)
%Set mask using the color HSV limits for colors
hueMask = (hue >= hueL) & (hue <= hueH);
saturationMask = (sat >= satL) & (sat <= satH);
valueMask = (val >= valL) & (val <= valH);
J = uint8(hueMask & saturationMask & valueMask);

maskedImageR = J .* I(:,:,1);
maskedImageG = J .* I(:,:,2);
maskedImageB = J .* I(:,:,3);
RGB = cat(3, maskedImageR, maskedImageG, maskedImageB);
subplot(4,3,6)
imshow(RGB)

G = rgb2gray(RGB);
BW = imbinarize(G);
subplot(4,3,7)
```

```
imshow(BW)
Image = bwlabel(BW, 8);
blobMeasurements = regionprops(Image, originalImage, 'all');
    % Display the image with informative caption.
    Area = cat(1, blobMeasurements.Area)
    [r,c] = size(Area);
    for r2 = 1:r
      if(Area(r2,1) < 900)
        Area(r2,1) = 0;
      end
    end
    for r3 = 1:r
      if(Area(r3,1) > 13000)
        subplot(4, 3, 8);
        I = insertObjectAnnotation(I,'rectangle',blobMeasurements(1).BoundingBox,'Street Sign');
        imshow(I)
      elseif(Area(r3,1) > 8000)
        subplot(4, 3, 8);
        I = insertObjectAnnotation(I,'rectangle',blobMeasurements(1).BoundingBox,'Freeway Sign');
        imshow(I)


      end
    end
end

function findyellow(I)
      hueL = 0.10;
      hueH = 0.2;
      satL = 0.4;
      satH = 1;
      valL = 0.8;
      valH = 1.0;
originalImage = rgb2gray(I);
[r,c,z] = size(I)
hsv = rgb2hsv(I);
subplot(4,3,2)
imshow(hsv)
hue = hsv(:,:,1);
sat = hsv(:,:,2);
val = hsv(:,:,3);
subplot(4,3,3)
imshow(hue)
subplot(4,3,4)
imshow(sat)
subplot(4,3,5)
imshow(val)
%Set mask using the color HSV limits for colors
hueMask = (hue >= hueL) & (hue <= hueH);
saturationMask = (sat >= satL) & (sat <= satH);
valueMask = (val >= valL) & (val <= valH);
J = uint8(hueMask & saturationMask & valueMask);

maskedImageR = J .* I(:,:,1);
maskedImageG = J .* I(:,:,2);
maskedImageB = J .* I(:,:,3);
RGB = cat(3, maskedImageR, maskedImageG, maskedImageB);
```

```
subplot(4,3,6)
imshow(RGB)

G = rgb2gray(RGB);
BW = imbinarize(G);
subplot(4,3,7)
imshow(BW)
Image = bwlabel(BW, 8);
blobMeasurements = regionprops(Image, originalImage, 'all');
    % Display the image with informative caption.
    Area = cat(1, blobMeasurements.Area);
    [r,c] = size(Area)
    for r2 = 1:r
      if(Area(r2,1) < 700)
        Area(r2,1) = 0;
      end
    end
    Area
    for r3 = 1:r
      if (Area(r3,1) > 7000)
        subplot(4, 3, 8);
        I = insertObjectAnnotation(I,'rectangle',blobMeasurements(1).BoundingBox,'Slippery Road');
        imshow(I)
      elseif (Area(r3,1) > 1000)
        subplot(4, 3, 8);
        I = insertObjectAnnotation(I,'rectangle',blobMeasurements(1).BoundingBox,'slow down');
        imshow(I)

      end
    end
end
function findred(I)
      hueL = 0.80;
      hueH = 1;
      satL = 0.58;
      satH = 1;
      valL = 0.55;
      valH = 1.0;
originalImage = rgb2gray(I);
[r,c,z] = size(I)
hsv = rgb2hsv(I);
subplot(4,3,2)
imshow(hsv)
hue = hsv(:,:,1);
sat = hsv(:,:,2);
val = hsv(:,:,3);
subplot(4,3,3)
imshow(hue)
subplot(4,3,4)
imshow(sat)
subplot(4,3,5)
imshow(val)
%Set mask using the color HSV limits for colors
hueMask = (hue >= hueL) & (hue <= hueH);
saturationMask = (sat >= satL) & (sat <= satH);
valueMask = (val >= valL) & (val <= valH);
J = uint8(hueMask & saturationMask & valueMask);
```

```
maskedImageR = J .* I(:,:,1);
maskedImageG = J .* I(:,:,2);
maskedImageB = J .* I(:,:,3);
RGB = cat(3, maskedImageR, maskedImageG, maskedImageB);
subplot(4,3,6)
imshow(RGB)

G = rgb2gray(RGB);
BW = imbinarize(G);
subplot(4,3,7)
imshow(BW)
Image = bwlabel(BW, 8);
blobMeasurements = regionprops(Image, originalImage, 'all');
    % Display the image with informative caption.
    Area = cat(1, blobMeasurements.Area)
    [r,c] = size(Area);
    for r2 = 1:r
        if(Area(r2,1) < 700)
            Area(r2,1) = 0;
        end
    end
    for r3 = 1:r
        if(Area(r3,1) > 13000)
            subplot(4, 3, 8);
            I = insertObjectAnnotation(I,'rectangle',blobMeasurements(r3).BoundingBox,'Yield');
            imshow(I)
        elseif(Area(r3,1) > 1200)
            subplot(4, 3, 8);
            I = insertObjectAnnotation(I,'rectangle',blobMeasurements(r3).BoundingBox,'Stop sign');
            imshow(I)
        end
    end
end
```