Traffic signal labelling using OpenCV

The traffic signals are detected using Hough Circle transform which uses Hough gradient to find circular regions. The gradient helps in retracing the circle edges thereby forming a circle with radius r and center (X, Y).

In the code I have defined a function detect_signal (), which takes filepath, file and count as arguments, displays the image with traffic light circled and its state, and writes it back into the output folder.

Reading Image, Converting into HSV and choosing the range:

>>img = cv2.imread(filepath + '\\' + file)

>>hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

#Red color range

>>lower_red1 = np.array([0,100,100])

>>upper_red1 = np.array([10,255,255])

The RGB image is read from the path and is converted into HSV image. Then with reference to the HSV chart, the red, green, and yellow color upper and lower HSV ranges are taken as an array.

Masking:

>>mask_green = cv2.inRange(hsv, lower_green, upper_green)

The HSV image is masked to the range selected with cv2.inRange() function. The function takes image, upper and lower range as inputs and masks the image pixels in that range. Masking highlights or focus on the portion of the image which is the range specified.

cv2.HoughCircles():

The **Hough Transform (HT)** is a basic feature extraction technique used in digital image processing for detecting circles in imperfect images. The circle candidates are produced by Hough gradient.

The cv2,HoughCircles() function detects circular regions in the image file and returns vectors of all circles detected which has (X,Y) centers of the circle and r radius of the circle.

The circle vectors are converted into numpy array with (X, Y, r) as coordinates.

>>Bound = 4.0/10

Now bound is defined which limits the detection range only to the circles and ignores unwanted regions with unwanted detection (i.e., car breaking lights)

```
>>r = 5
```

r is defined to detect all the detected circles and ignores other small circular region. If r is too high, then none of the circles are detected and no labeling is done.

Now if a circle is detected the h and s values are updated as,

For red circle

```
>>h += mask_red[i[1]+m, i[0]+n]

>>s += 1
```

Now,

Based on the h/s range the circle is drawn and a label is written on it.

```
>>if h/s > 50

        >>cv2.circle(cimg, (i[0], i[1]), i[2]+10, (0, 255, 0), 2)

        >>cv2.circle(mask_red, (i[0], i[1]), i[2]+30, (255, 255, 255), 2)

        >>cv2.putText(cimg,'RED',(i[0], i[1]), font, 0.8,(255,0,0),2,cv2.LINE_AA)
```

Where, i[0] , i[1], i[2] are (X, Y, r) coordinates of circles.

Writing into a file:

```
>>cv2.imshow('Traffic_labels' + str(count), cimg)
```

Finally the output is displayed using imshow() method. The method takes two arguments, name of the file and the image file and displays the output in another window.

```
>>path = r"C:\Users\rajiv\Practise\Traffic Light Detection - OpenCV\Traffic_labels\Output"

>>cv2.imwrite(os.path.join(path, 'traffic_labels' + str(count) + '.jpg'), cimg)
```

Finally, the path where the output files are to be stored is given to imwrite() method.

The imwrite() method takes path and image as arguments and stores the images in that destination.

Application of Deep Learning:

Deep learning methods such as convolution neural networks can also be used to detect and classify the signal state.

In this method different images with traffic lights are trained y developing a convnet and thereby tuning the model to improve its performance. Once the model is trained, the image files given in the assignment can be passed as training set and the labels can be detected.

The best approach in solving this problem rather than developing a convolution neural network is to use transfer learning methods of imageNet.

We can use vgg16, inception, Resnet50 or any other methods where the model is pretrained on several images and we can take the advantage of trained weights to our model and train the dataset. Once trained we can use the image files give in the assignment as training dataset and predict the state of the traffic lights.

Embedded Application:

I have used OpenCV for the DRDO DRUSE competition during my bachelors where we developed a miniature version of MARS Curiosity rover using bogie suspense. The rover is mounted with several sensors and in addition to it, it carries a camera which detects the yellow lanes for locomotion. The heart of all these operations was a Raspberry pi which sits in the center of the rover.