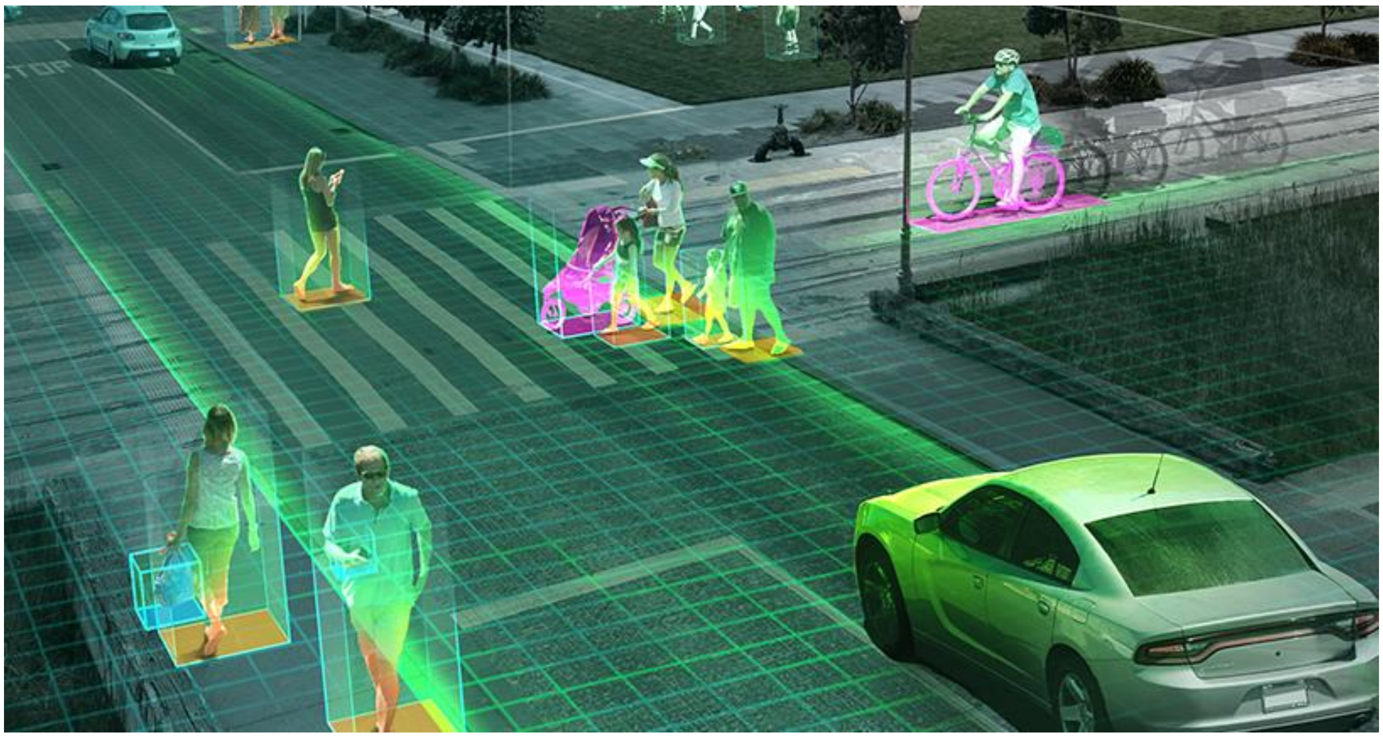
CS4053: COMPUTER VISION

ASSIGNMENT 2

Recognizing Traffic Lights



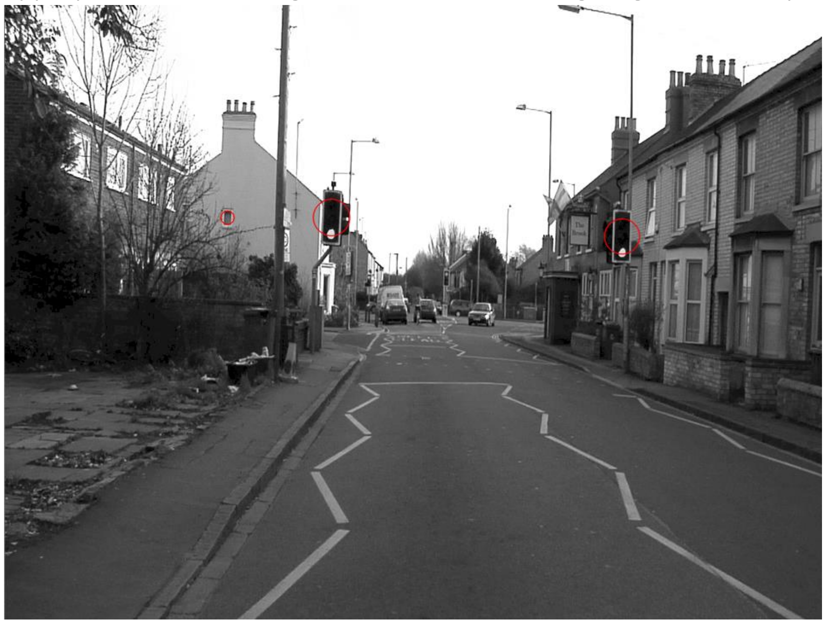
Submitted To: Kenneth-Dawson Howe

Submitted By: Kavin Gupta

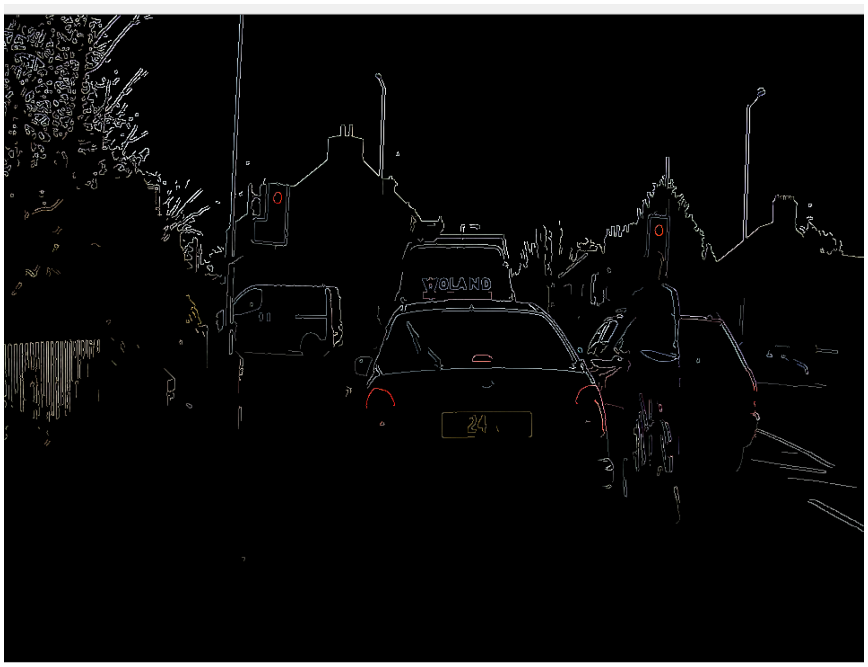
17317505

Methods tried:

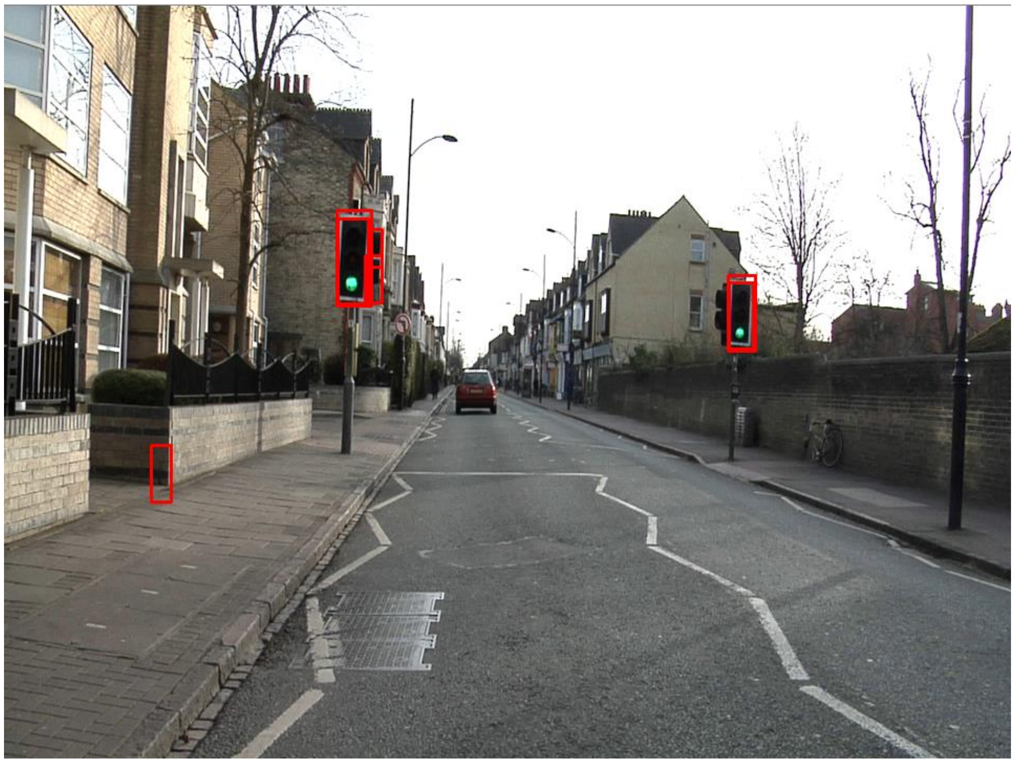
Blob Detection: The method is used mainly to find a group of pixels with similar characteristics by thresholding the image. Here in the given test images the traffic lights were mainly circular and with versatility of the method as it defines the circularity, convexity and inertia ratio of the blobs. With appropriate thresholding also, this method was giving some false positives.



Edge Detection: This method was great and with appropriate thresholding the traffic lights were getting detected. But in some of the images there were cars whose lights were acting as a traffic lights as shown below.



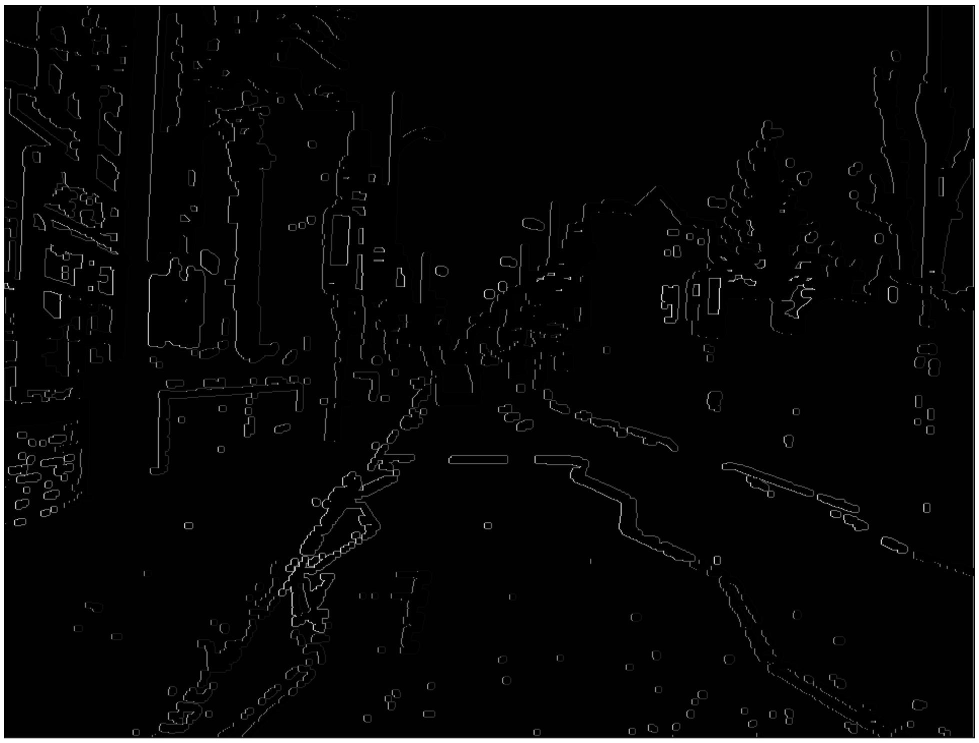
Template matching: The idea was template matching seemed to go with as the result was having a rectangle surrounding the traffic light. With appropriate thresholding the results were good, with some false positives.



Colour masking: The colour mask was introduced for each type of blob i.e. red, amber and green. The idea was to apply a mask that comprises of the lights in said color. After that I have used a bitwise_and on the input image and the threshold image so that only the red, yellow and green lights are highlighted and stored in result.

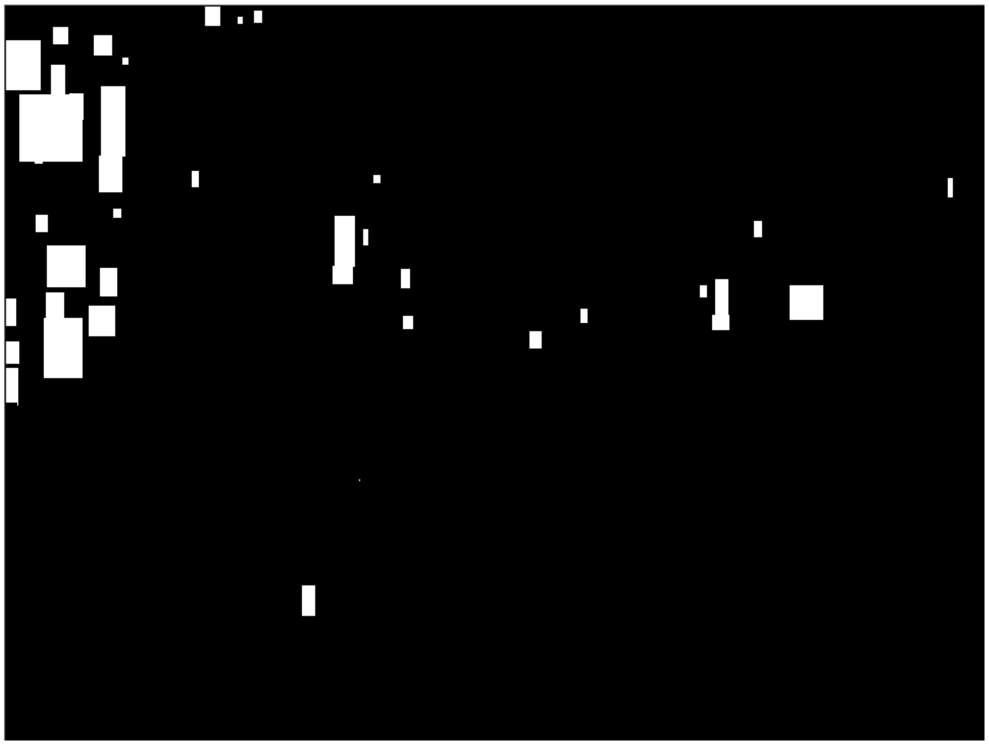
Histogram and Back Projection: The Back projection was used on the HSV image of the source image to get the rough idea of the traffic light positions. Back projection of the histogram over the test image will calculate the probability of every pixel belonging to the traffic light.

Contour Image: The contour provides a continuous flow of the image and similar to edge detection. The rectangles of the traffic lights were the one ones needed.



Contour Image

Morphological Segmentation: The segmentation was applied to separate the background from the foreground. The erosion and dilation is don appropriately according to the result needed.



Dilated Image after thresholding

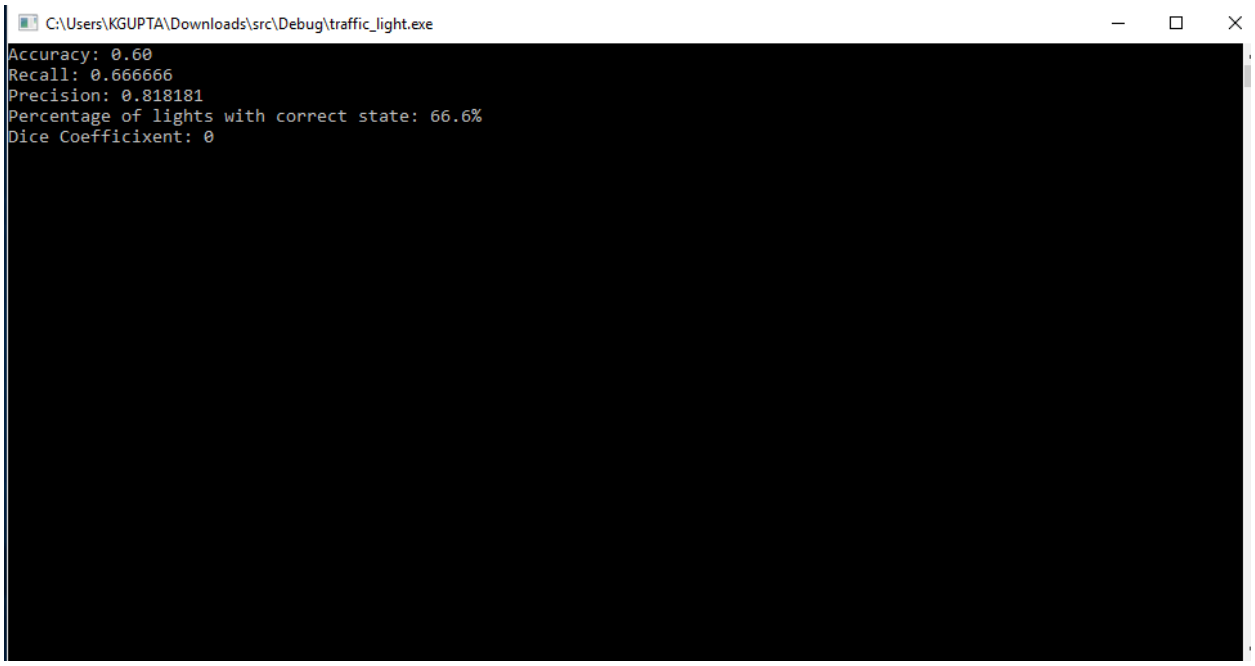
Proposed Solution:

The proposed solution to the project is to use the contour imaging, template matching, color masking and morphological segmentation. I believe these operations should be enough for the recognition of the traffic lights. The contour image of the source one will join all the continuous point shaving similar color. This will ease thongs for object recognition. The morphological segmentation will separate the background and the foreground. The idea is to find the rectangles formed by the traffic lights for more accuracy. With proper thresholding and applying erode and dilate appropriately the traffic light rectangles can be carved out. The images were masked on using HSV filter and SCALAR function and the colours were detected. The color masking is to detect each of the traffic lights and state of the traffic lights. The perceptually uniform colors will be masked and state of the traffic light would be detected. Then would be making rectangles surrounding the light by using rectangle command that would make a rectangle after detecting from morphological segmentation. The template would correspond to the binary image we will get after the image segmentation. With all the normalization and types of match it has the accuracy should be quite high.

Problems Faced

- Detection of false positives was a major problem as the template match would discover one thing again and again.
- Edge Detection is a fail when recognizing objects outside as the trees and building surrounding it would form a similar kind of the structure, hence usage of more liquid mask.
- Histogram of the image was not able to get the desired output by appropriate thresholding and hence the image segmentation used.
- The intensity of the light were not bright enough.
- Getting the accuracy high enough as there can never be a 100%. The color mask will sure be a plus point as the blobs of the lights are getting detected individually.
- Calculation of dice coefficient.
- Getting the colour state of a mixed light or in some case more than one light but different colours.

Performance Metrics



```
Accuracy: 0.60
Recall: 0.666666
Precision: 0.818181
Percentage of lights with correct state: 66.6%
Dice Coefficixent: 0
```

The performance metrics were a crucial part to obtain a positive result. The ground truths were given to us in a csv file where the area of the lights were somehow calculated. The rectangle area was calculate using Rect.area() for the every matched and unmatched light and was compared with the ground truth given with a 20% margin. Then these were considered as parameters to calculate the accuracy, recall and precision. The observation and calculation of getting True Positives, False Positives, True Negatives and False Negatives were a hoot. The true positive were the one that were correctly identified , false positives was the ones that were identified but were not on traffic lights or cover 80% of the area. The total number of lights to be recognized were 30 in 14 images given to us. The calculation of accuracy, recall and precision was done on these variables by the formulas:

Recall = True Positives / (True Positives + False Negatives)

Precision = True Positives / (True Positives +False Positives)

Accuracy = True Positives / 30

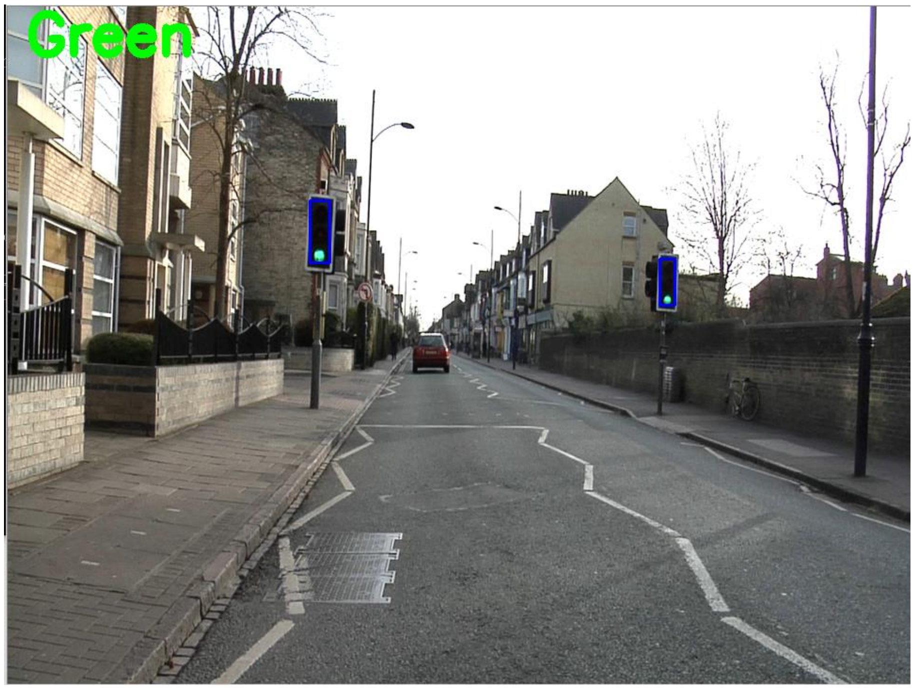
Average Dice coefficient = dice coefficient / 30

The TP's were 18, FP's were 4 and FN's were 9 due to the fact with ratio thresholding after contour image was eliminating some traffic lights as the computer misunderstands the height and width.

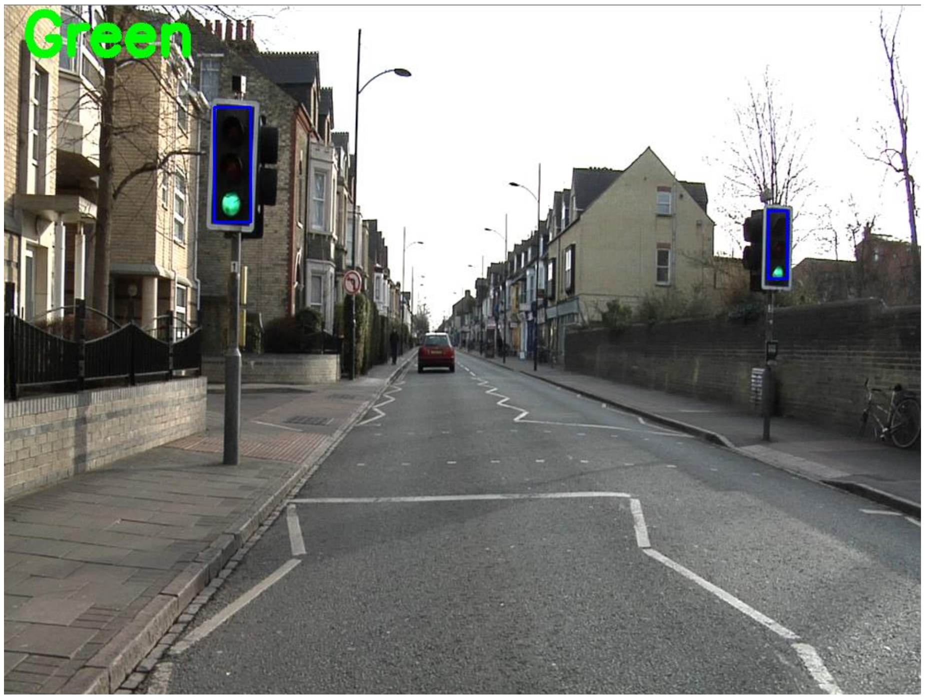
The Dice coefficient was the division of the area of the traffic lights that were recognized to check the similarity of the images. The lights that were not detected the area was assumed to be zero.

The percentage of the state of the light were recognized but for mixed lights i.e. Red + Amber my code was detecting only one and sometimes the wrong color.

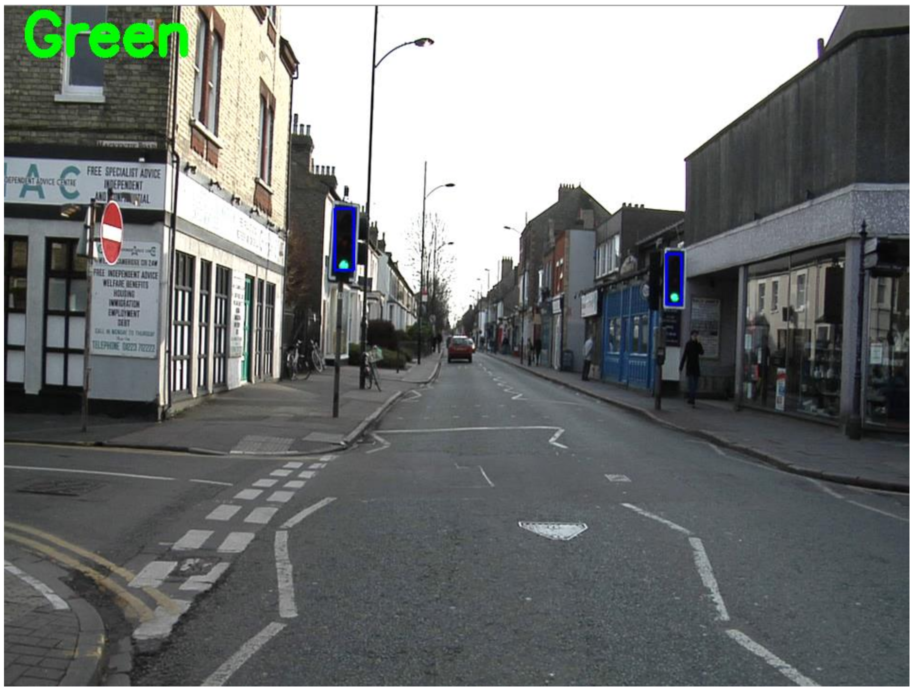
IMAGES



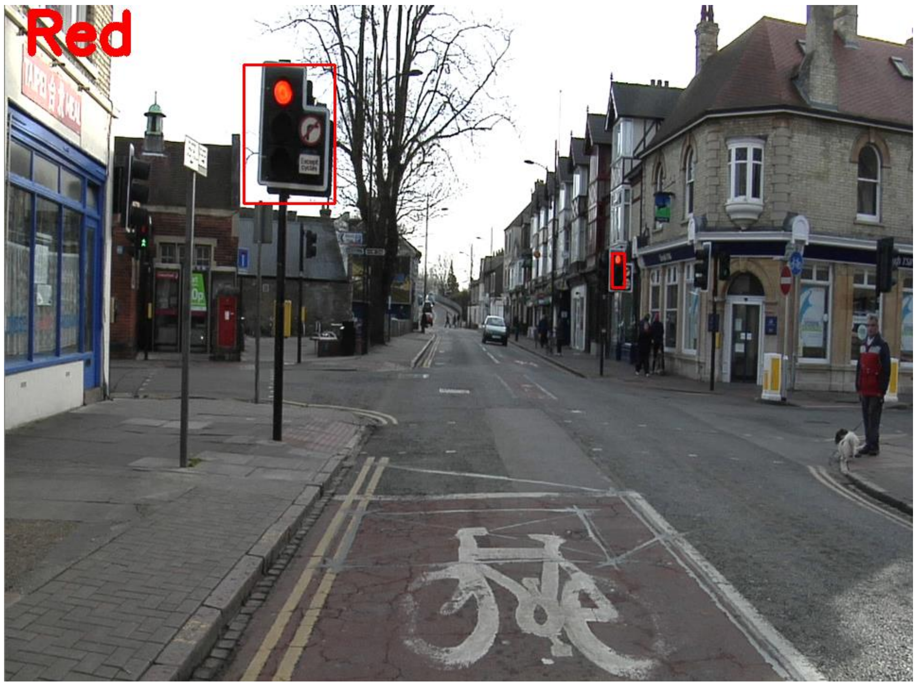
CAMVIDLIGHTS01



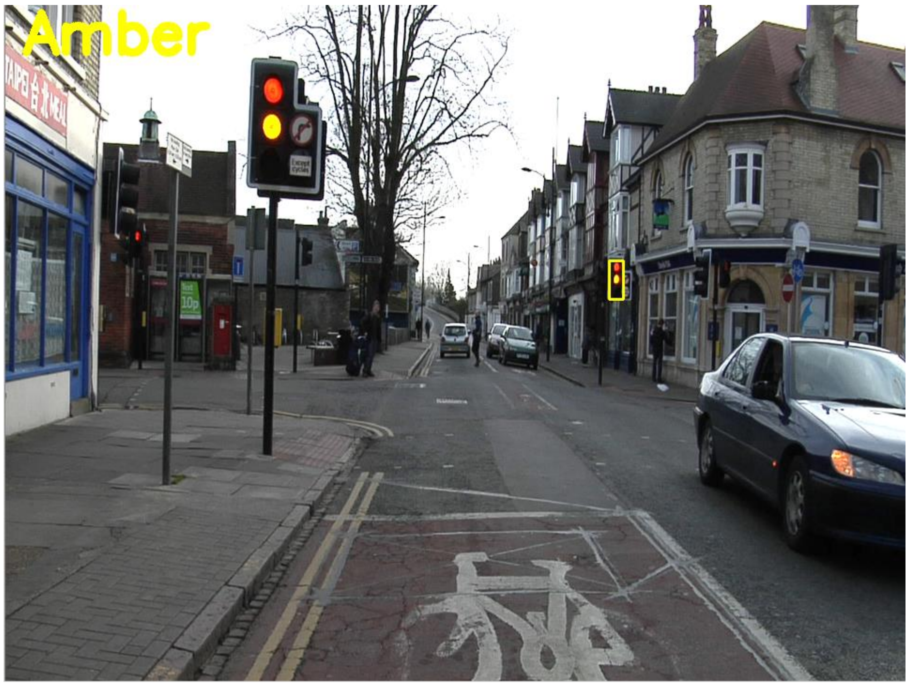
CAMVIDLIGHTS02



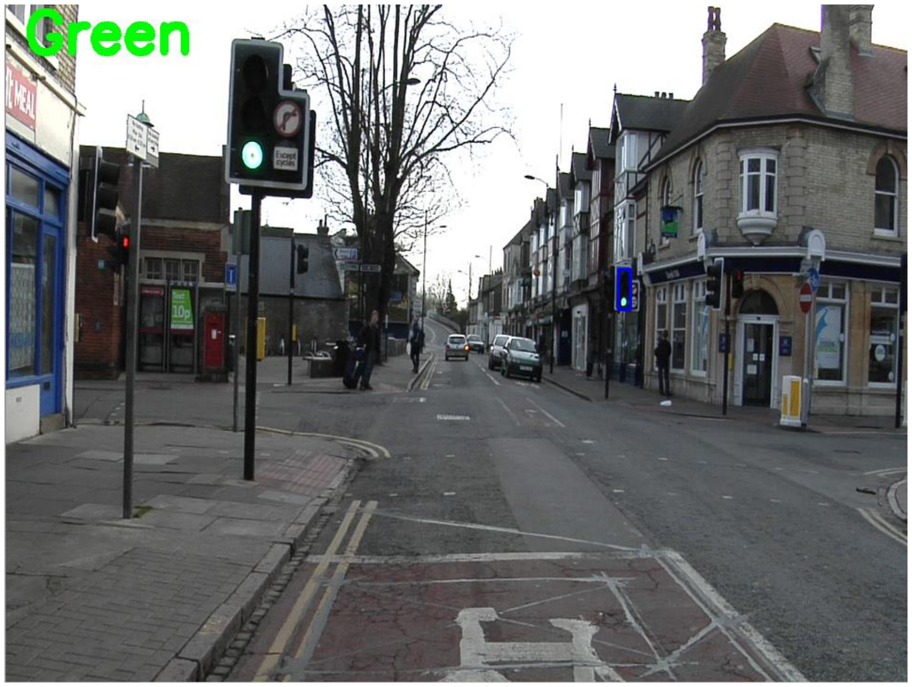
CAMVIDLIGHTS03



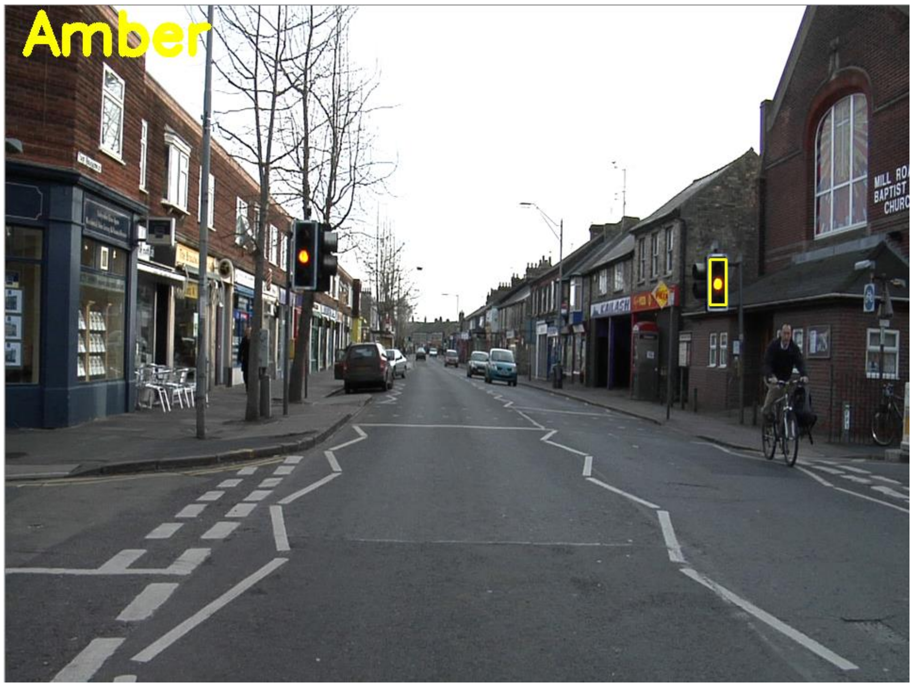
CAMVIDLIGHTS04



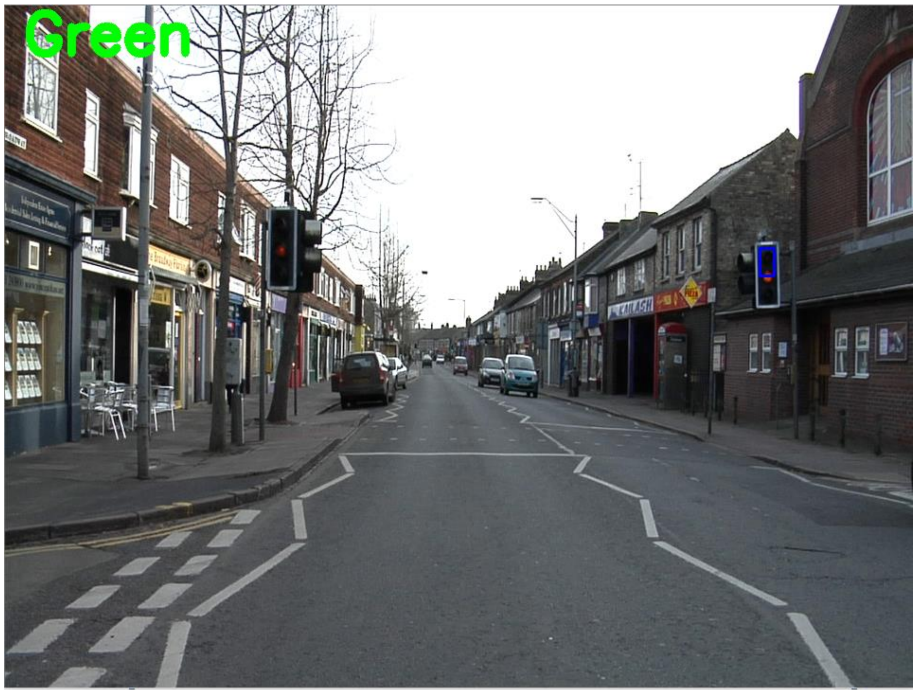
CAMVIDLIGHTS05



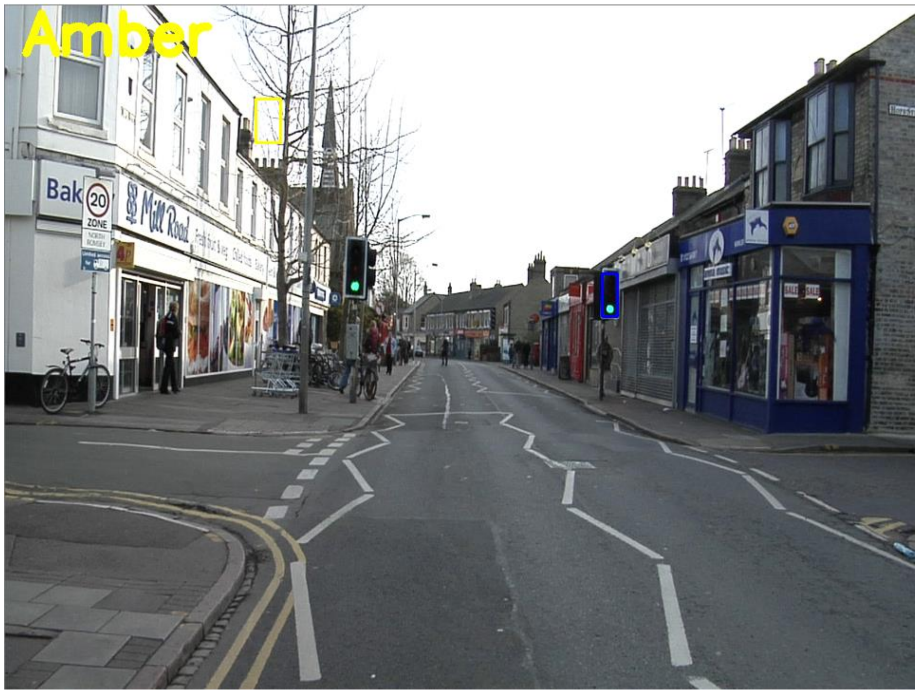
CAMVIDLIGHTS06



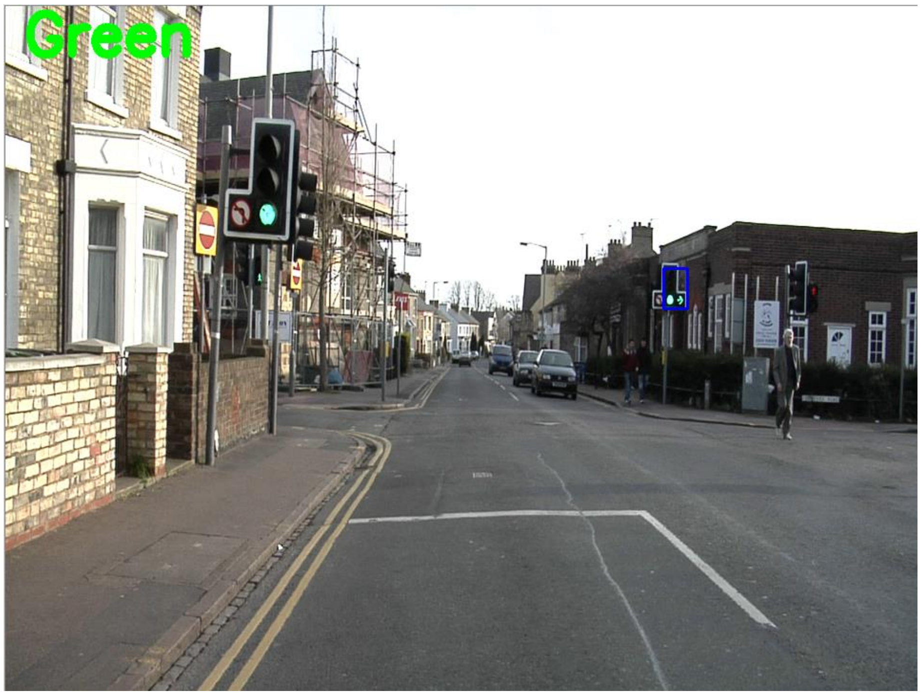
CAMVIDLIGHTS07



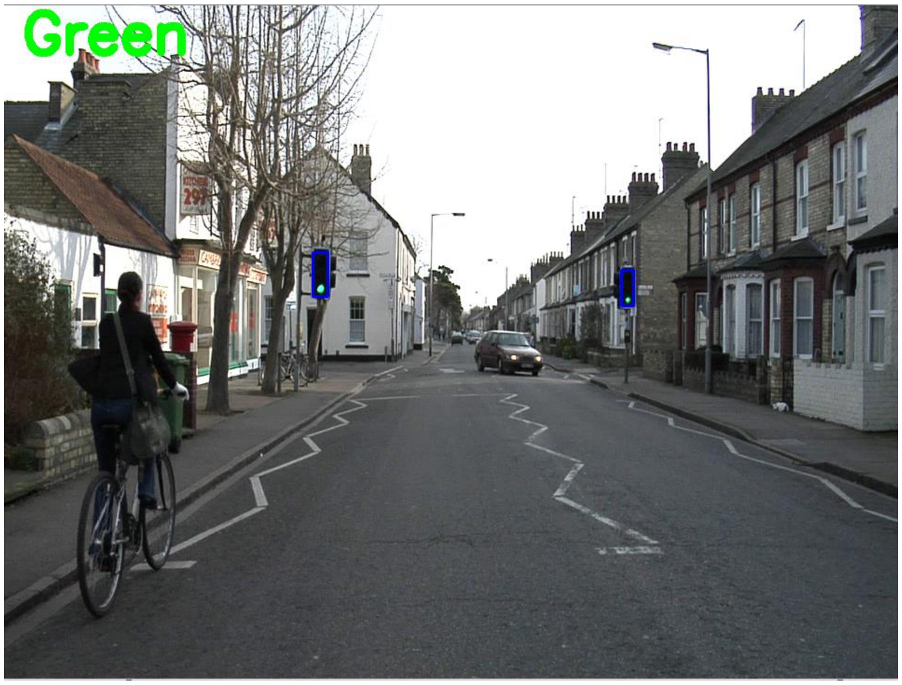
CAMVIDLIGHTS08



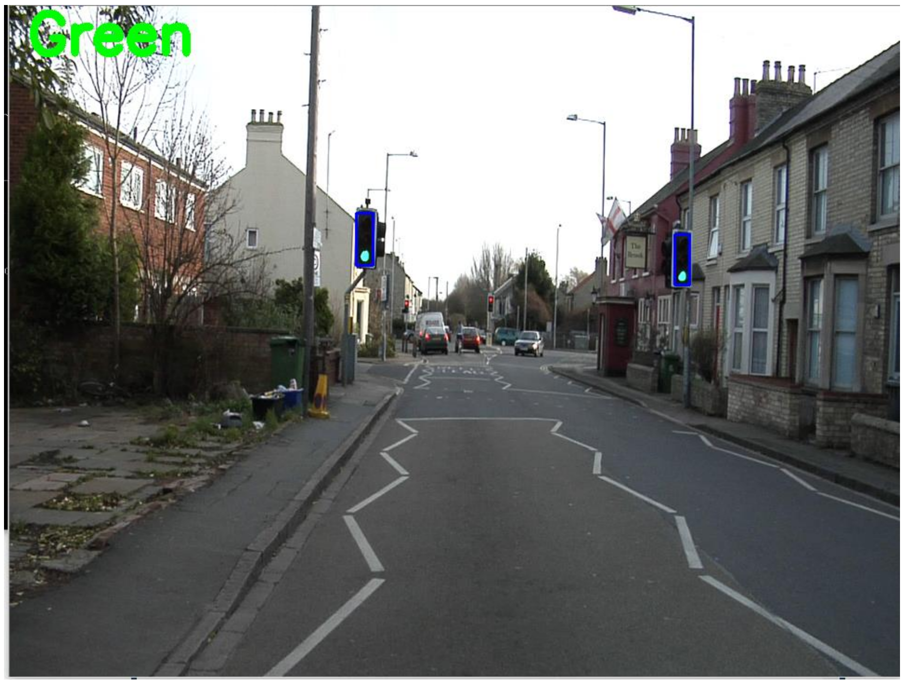
CAMVIDLIGHTS09



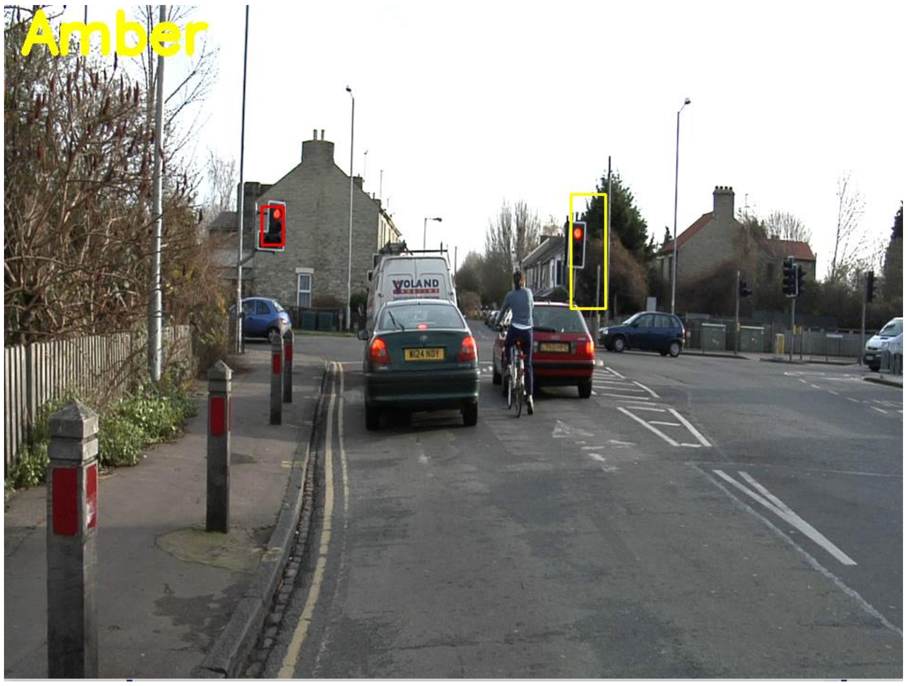
CAMVIDLIGHTS10



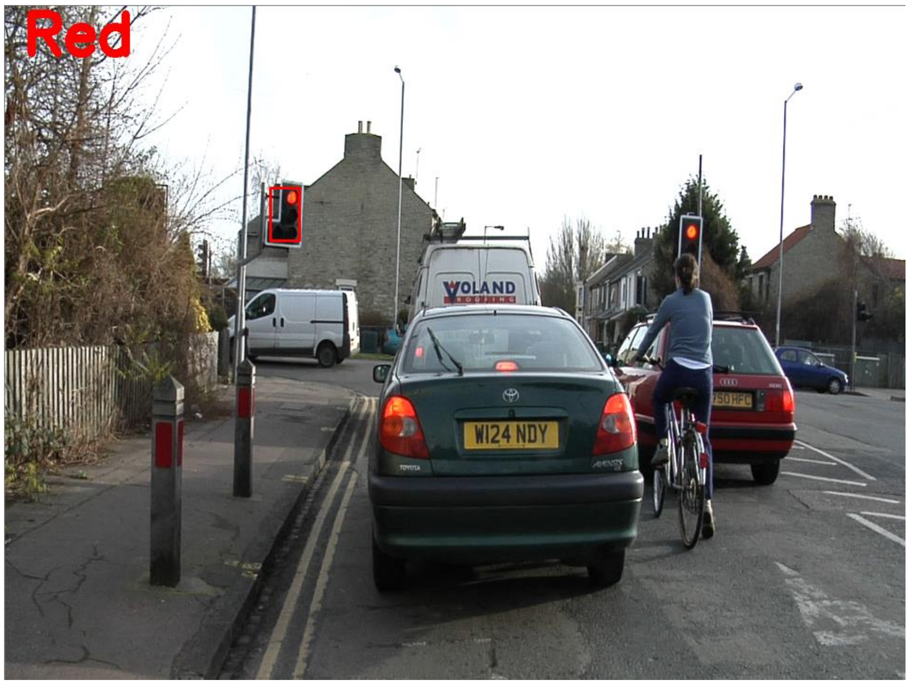
CAMVIDLIGHTS11



CAMVIDLIGHTS12



CAMVIDLIGHTS13



CAMVIDLIGHTS14