Traffic Signal State Detection

Amartya Singh, Parth Shah, Surya Teja Sharma

Motivation

Self driving cars can make road travel cheaper and safer, that being said, the process of them becoming robust and reliable is still a few decades away. With Machine Learning applications being pervasive, the fact that they are computationally expensive and heavily dependant on pre processing doesn't change. We have come up with a simple, yet effective solution to detect the state of a traffic light which will help the self driving car to make a decision about switching the engine on/off at the right time at a signal, utilizing fuel better and also contributing in small ways to make the environment a better place.

Approach

In order to figure the state of a traffic light in an image, it is crucial to detect and confirm the presence of a traffic light and also identify the signal color to know if its in Red or Green state. As detection of traffic light by image processing techniques alone may not give satisfactory results, we first look through the image with bounded intensity values for the colors Red ,Orange and Green. Once we find a match in color, we extract the region which is the area of interest. We then apply segmentation techniques to verify if a circular fit is possible. We also validate the same image by running canny and picking the strongest contours to see if the detected circular object is a part of a traffic signal.

Color Matching -
For each image we get as an input, the first step is to detect whether we get red, yellow or green. We set the intensity boundaries for each of the channel and try to see if the input image matches any of them. We have implemented three functions (one for each channel). Each of the three functions responds whether it was able to find the color in the image or not. If it does, it crops that part from the image for further analysis.

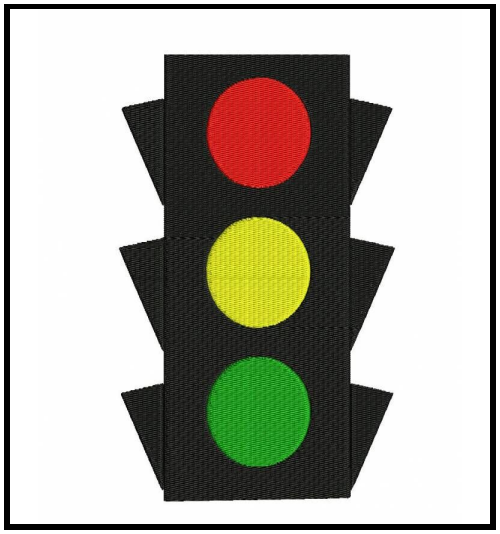
Hough Circles -
In order to find the traffic lights in an image, it is crucial to find the shape of the detected match in the above step. The shape in consideration is a circle since traffic light casings are circular by default. This is where Hough Circles are used to find the circular shape based on a voting mechanism. The grayscale image undergoes blurring in order to remove noise using the Gaussian Blur. The vote is accumulated based on the specified parameters such as the minimum, maximum radii of a circle and the threshold for the number of votes along with

specifying the gradients. The circular elements which receive the majority of the votes will correspond to the traffic lights.

Edge Detection and Shape fitting -
The captured image is passed as an input to the *Canny()* function defined in OpenCV. This step is crucial as it helps us decide if the Hough circle found is indeed that of a traffic light by checking if the circle exists within the detected object. We used the *findContours()* function to find a closed contour and *approxPolyDP()* function to decide if a contour is similar to a geometric shape with at least 4 vertices (assumption that a traffic signal box is of rectangular shape).

Data Used

Various images from the internet were used to tune the scripts and to arrive at decent parameter values to get best results. The data used to test the correctness and quality of the project can be classified into **Simple**, **Difficult** and **Hard** categories. The simple category comprised of images which had a frontal view of a signal with bright intensity of colors. Ex:



Simple Image

The Difficult category consisted of images where the traffic light had issues with the color intensities of the image. This is a difficult challenge as the color bounding is solely dependent of finding patches in image which have intensity within a defined bound, which varies from color to color. Furthermore, images where the traffic signal boxes are in yellow color, pose a potential issue for detecting yellow regions of an image. Example of such an image:

Difficult Image

In the case of Hard category, we include a good mix of random images to gauge the robustness of code and correctness of result by inputting images which may result in False Positive results or undesirable results. Also, images where the traffic lights are not frontal captures of a signal pose a threat of not being correctly detected as traffic lights as our process depends on the ability to find a geometric shape similar to a rectangle, which may not be found in such images. Examples are as follows:

Hard Images

Results

As mentioned in the approach, we have three crucial steps in the process of detecting the state of the traffic light, consider the following input image, we will walk through each of the steps with intermediate results:

Input Image

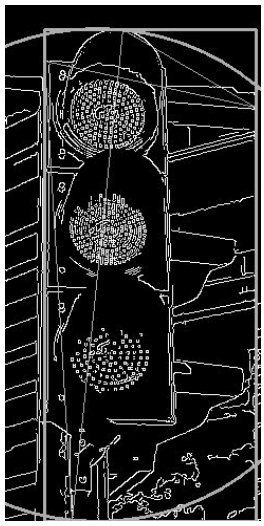
In the above input image, the first step is to locate the area of interest by detecting the colors Red, Yellow or Green, which will be the deciding factor of the state of the traffic light. The input image is passed as a parameter to the script to find which color of the three are found in the image. If a color is found, we box the area of interest and crop the image for further analysis. The cropped resultant image is as follows:

Area of interest

Once we have the area of interest, we would next want to know if there is a circle encapsulating the color patch, which is an assumption made on real world knowledge that lights inside a signal are usually circular in shape. To find out if a circle exists in the cropped image, we send the above image as an input to a script which tries to draw circles based on Hough transform. If a circle is found in the area of interest, the following result is returned:

Hough Circle drawn around signal

If a circle is found in the area of interest, we need to detect and validate if there is a traffic box, usually rectangular in shape, present in the original input image. We use Canny edge detection to find edges and then make geometric shapes out of the detected contours using the implementation of Douglas Peucker algorithm to find polygons with at least 4 vertices in the edge map. If we do find such an object which lies in the previously detected are of interest, we can confirm that a traffic light is detected. The output of this script will be as follows:

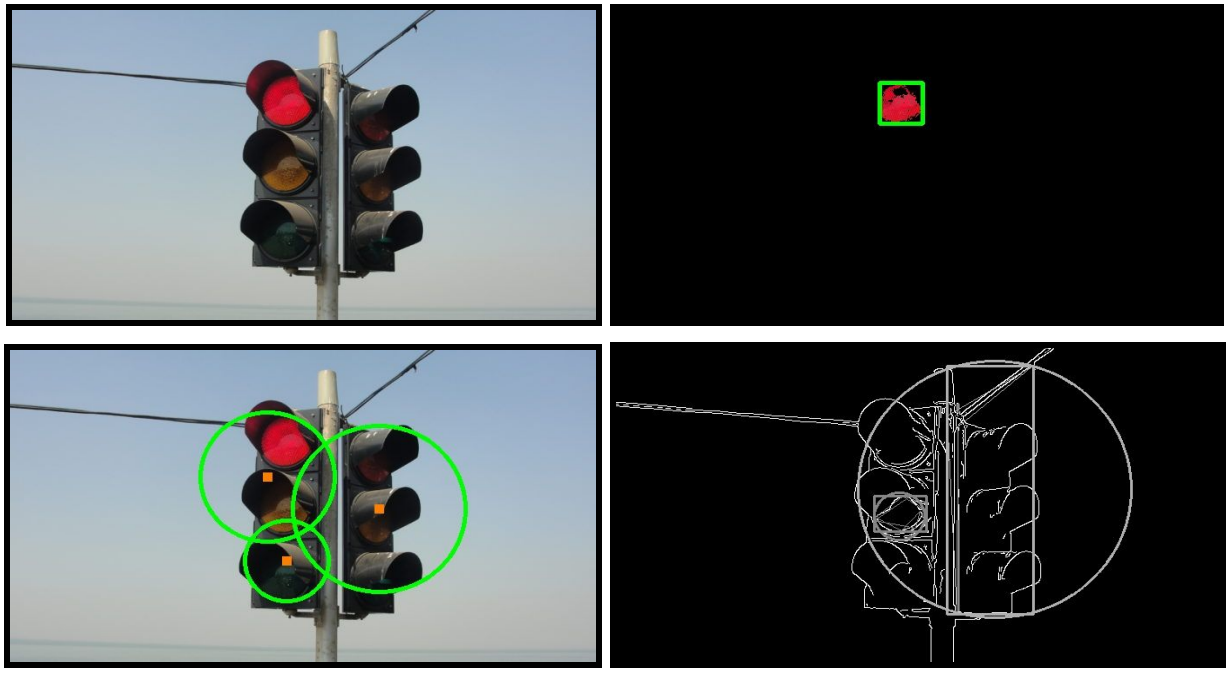


Rectangle & Circle drawn around found probabilistic geometric contours

Critical Discussion

The critical discussion around the state detection of a traffic light stems from the question of how do we detect if there is a traffic light in the input image? How to guarantee if we are in fact looking at a traffic light? Should we take surrounding environment settings into consideration? The answer to these questions are not simple and need some elaboration. Also, we need to keep in mind that the solution should be pragmatic in the computation sense of the word and yet

be useful. Without the backing of a trained data model, to have a success rate in detecting a traffic light is difficult. Few of the reasons for that are d*ifferent shapes may be used to build traffic lights, different shades of Red-Yellow-Green intensities can be used in lights or the picture may not always capture the signal in frontal view*. These are the challenges which we try to overcome by making a few assumptions to reduce the scope of the project. The assumptions are that the *traffic lights are in rectangle shapes and the signal inside the traffic light box is circular and that the input images will have a frontal view of the traffic light*. We can justify these assumptions as most of the traffic lights follow the standard of having circular lights placed inside rectangular boxes and the frontal view of traffic light is guaranteed in our use case of a self driving car as the camera that captures the image is placed on the top of the car,which gets a perfect frontal view of the traffic light. Also, a few images from the data set are in accordance with assumptions yet there are issues when the intensity of the color is not without the set static bounds, or if the hough circle is not rawn exactly on the signal as reflections may sometimes make it harder for correct detection of contours. And, the last hurdle of detecting if the object is a traffic light is heavily dependant on the edge detection and contour mapping to geometric shape. It approximates a contour shape to another shape with less number of vertices depending upon the precision we specify. The precision parameter in this project is set at 5 percent, which works for most images but definitely fails for certain images. A tentative solution is to take in a range of precision values and decide an output based on averages at different precision value, but this process will make the process computationally intensive. Following are a few cases where the project failed in recognition:



The above images show that Red color was identified successfully and was passed to the next script to fit a circle, but no hough circle was found in the cropped image. This image result would have been that no traffic light was found, but unfortunately, that is not the case.

Future Work

The future work on this project can involve:

- Understanding the environment of the traffic light better, implying that we can take into consideration the distance of the traffic light from the camera using camera calibration.
- Reduce failure cases by taking more identification factors than mere object detection.
- Dealing with different angles of traffic light to make the system more robust.
- If multiple traffic lights are present in an image, identifying the correct signal depending on distance from object and orientation of car/camera with respect to the object.