Intelligent Traffic Lights

Ayush Yadav 2017034
Tanish Jain 2017115
Vaibhav Jayant 2017270

Abstract **— In this project we are trying to create automated traffic lights which will be able to coordinate traffic according to the number of cars at any red light. Also there is a provision for the detection of emergency vehicles which should be allowed to pass when they are stuck in a red light.**

Keywords — Traffic lights, background removal, image processing, edge detection, vehicle detection, emergency vehicle beacon detection

I. Introduction

In India, around 24K people die daily simply because they are not able to reach the hospital on time.[1] A similar situation is present in many other countries of the world. Similarly, there are many cases where a lot of deaths and property damage occur because the fire brigade vehicles are not able to make it in time. We plan to change that. The traffic lights would be controlled based on the presence of emergency vehicles and the traffic on different sides of the roads. Also, it would help to reduce traffic jams when one lane is comparatively empty, then this system will stop this line longer for the easy flowing on the other line.

There have been many technologies that have tried to counter this problem, like magnetic loop detectors buried in the road, radar & infra-red sensing technology, etc. But these provide very little information, are not practical to use and are hard to code. We plan to install cameras with the traffic lights that would continuously capture images and process them to sense the presence of emergency vehicles and also count the number of vehicles on the road whose images it is capturing. It would require the use of Edge Detection, Background Subtraction, Red Light Detection possibly technologies like Machine Learning, Computer Vision, Artificial Intelligence, etc. and some more advanced image processing techniques also.

This technology would combine surveillance technology with the technology that is responsible for controlling traffic through traffic lights. This technology can be combined with many more technologies and greatly increase the control of traffic. We have added some extra code that shows some of these possibilities. For example, we have added a code that reads the number plates of the vehicles. This can be used to pinpoint vehicles with criminals in it or vehicles that were stolen and many more similar cases. We have also added a code that scans whether the vehicle is a car or a truck or something similarly distinctive.

With this technology, a lot of accurate data related to traffic, roads, accidents, etc. can be collected easily and various studies can be done on it.

II. Methodology

We have coded totally in the Python programming language. We used the following libraries: NumPy, OpenCV, ImageAI (which uses

TensorFlow, Keras and many more), Pillow, SciPy, Scikit-Image and many more.

A camera would be installed at all the traffic lights. This camera will continuously capture images or capture a video and send it to a connected processor where our image processing code will run and control the traffic lights accordingly.

The code first takes two consecutive images and then subtracts them to get an image in black and white where the white pixels are the edges of the cars. This is because in two consecutive images only the cars have moved (even when the traffic is moving slowly the cars would be at different pixels in two consecutive images) and the surroundings remain stationary. This image consists of only the edges that are the edges of the cars. This would be the case in an ideal situation but there was a lot of noise in the image, probably because there were minor movements in the surroundings and there is noise that is being added to the image while capturing the image. So, we used Canny filter with a higher T_L (low threshold) to remove those edges and finally we were left with a much more clear image with only the edges of the cars.

Sometimes, it is possible that two roads going in opposite direction are such that the camera's images are also capturing the other road. So, it is possible that the processor would perform wrong calculations. Thus, those cars would also have to be removed. We are removing them by simply traveling through the image. In the cases of both the roads, the other road would be at the left hand side of the road.

Now, with the image showing only the edges of the cars, some calculations are performed that finally control the traffic light. First of all, the distance of the last car from the top of the image is calculated. This can simply be done by calculating the first white pixel of the image along the rows of pixels of the image. This is done for all the roads that are intersecting. The maximum distance of the last car from traffic light is considered and the corresponding road's traffic light is turned green and all others are turned red. But this change is not done abruptly. There is a countdown of about 10 seconds on the road on which the traffic light was green before, assuming it was not the same road with the maximum distance that was just calculated. And finally the road with more traffic gets green light for more time. If instead the road with the green light before is the same as the road which has the maximum traffic, then that traffic light remains green. If the code is left at this, it would lead to one road getting green light and the other not getting green light for a long time if there were two roads with a lot of traffic. So, we added a maximum value of time for which a traffic light can stay green. The time that we fixed is 3 minutes as that is just enough to let a road with more traffic to empty the traffic and the remaining roads with less traffic can wait for that long and not any longer.

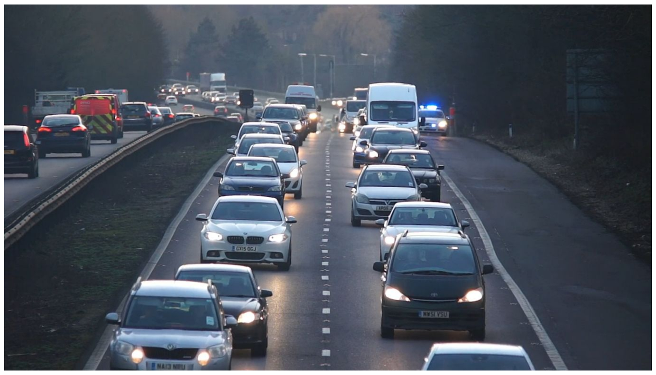


Fig.1 Image one

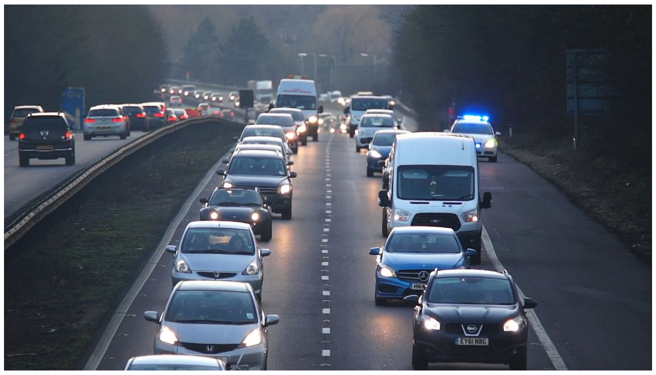


Fig. 2 Image two

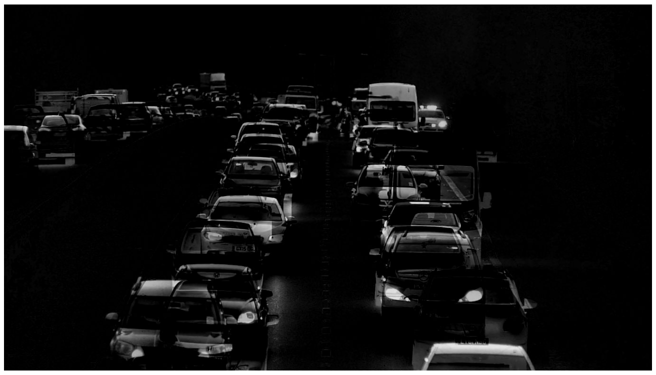


Fig. 3 Subtracted Image

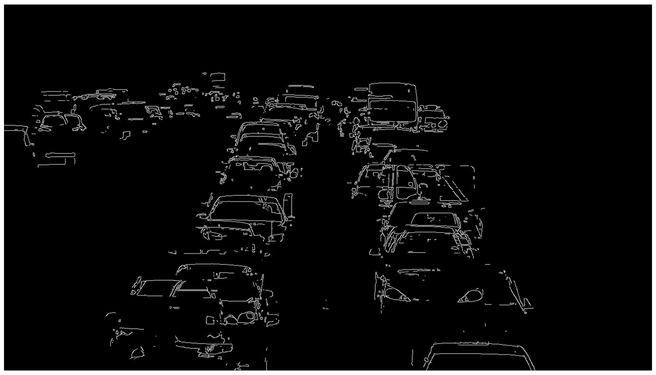


Fig. 4 Final Image with edges but with extra cars

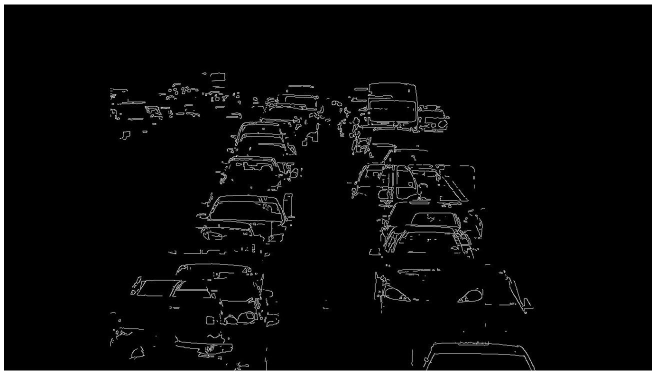
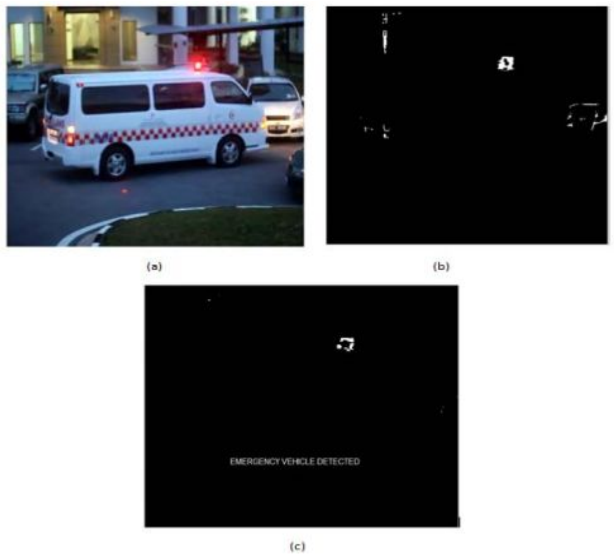


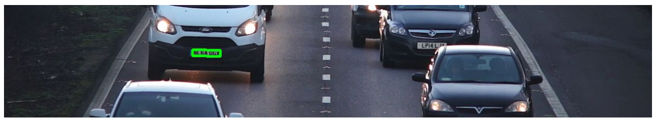
Fig. 5 Final Image with almost no extra cars

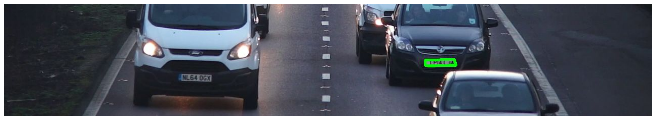
With all this technology and code added, we also added a system that gives priority to emergency vehicles like police cars, ambulances, fire vehicles, etc. We are scanning their beacons which rotate at a fixed speed and thus the images produced would have light in one image and no light in the next image if the images are captured with the right time difference. In other words, the light would be blinking. We would scan this change in light and after confirming that there is indeed an emergency vehicle present on one of the roads, we will turn the traffic light green and keep it green until the emergency vehicle has passed. In the absence of an emergency vehicle the code works normally but when there are emergency vehicles present, we give them more priority. In case there are multiple emergency vehicles present, the traffic lights would simply turn green for the first vehicle, wait for it to pass and then turn green for the next road and so on. This is because two traffic lights can't be turned green without the traffic running into difficulties. We are also checking whether the color of the light in the beacon of the emergency vehicle is of the right color or not.



[6]

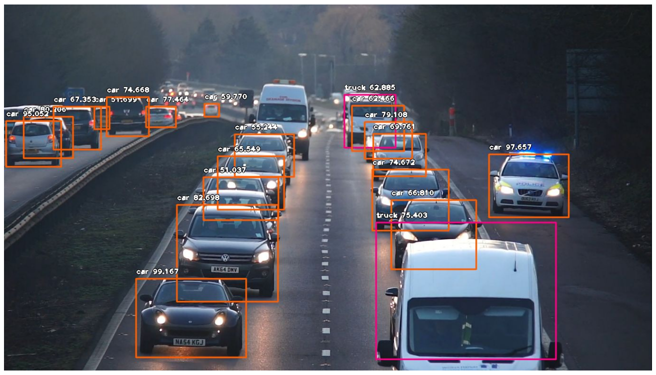
We have also added some extra codes to our project. These codes show the various possibilities that come with the use of our project. We added a code that scans the number plates of all the vehicles that cross roads with the traffic lights. It scans the small rectangles present in the images.





We have also added all kinds of filters that can be used including both the filter from the inbuilt libraries and the ones that we made manually.

We have also added a code that scans each vehicle for whether it is a car or a truck or a similarly distinctive vehicle. We used imageAI library for this.



III. Results & Discussion

With the method that we applied, we were able to fulfil the requirements for the project that we previously thought of having. The code does have some negative points but most of them are not significant. The code might change the traffic lights a little too frequently although many logics have been added that decreases the possibility of that happening. The filters used to filter the images that the camera gives might not be enough for low quality cameras, which might be used to decrease overall cost if this project is actually installed on the roads. In some cases, it is possible that the cameras are placed highly inaccurately, but we are not considering that possibility as that can't possibly be handled by code. The code might also not always have 100% accuracy in doing what it does. But that can probably be achieved by experts who can use our project.

IV. References

1. ImageAI Documentation - https://imageai.readthedocs.io/en/latest/detection/index.html
2. https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123
3. OpenCV Documentation
4. https://www.irjet.net/archives/V4/i4/IRJET-V4I4381.pdf
5. Video and Images Source - https://www.pexels.com/video/traffic-flow-in-the-highway-2103099/
6. Research Paper Reference - Vikramaditya Dangi, Amol Parab, Kshitij Pawar & S.S Rathod - http://interscience.ac.in/URJA/journals/urja_vol1no1/urja_paper4.pdf