|  |
| --- |
| University of Portsmouth |
| Coursework Report |
| Graphics and Computer Vision – UP918156 |

|  |
| --- |
| 12-20-2019 |

Task Two

## Introduction

For this task, it is required to design and implement a traffic warning system. This system monitors cars entering a tunnel and warns the authorities if a car is violating a traffic law. The project will be developed using Matlab and its image processing libraries.

## The Requirements

The following section will list the key requirements for the application.

* The system should warn the authorities in two conditions:
  + If the vehicle’s speed exceeds 30 miles/hour
  + If the vehicle’s width exceeds 2.5 meters
* Fire trucks are exempted from these checks
* The system will only take the input from a pair of images (by the camera)
* The system will output the speed and width outcomes
* The system will output the processed images, showcasing the key processing steps

## Assumptions

In order to develop this system, there are several key assumptions that need to be stated:

* The camera is fixed 7m above the ground at a 30-degrees angle with the horizon
* There is only one car in each image
* Blue cars are traffic cars, while red cars are fire trucks
* The resolution of the camera is 640X480
* Pixels of the camera are squared, and each sensor pixel is 0.042 degrees in the view angle
* Cars produce the largest blobs in an image, larger than 20000 pixels

## The Design & Implementation

This section will discuss the design and implementation of the key processing features of this application. First, a flow diagram will be provided to showcase the overall architecture of the system. Subsequentially, a brief description of each stage of the application, explaining the purpose and inner workings of that stage. Finally, a demonstration of the program will be provided presenting the final output of the system under various conditions.

### Flow Diagram

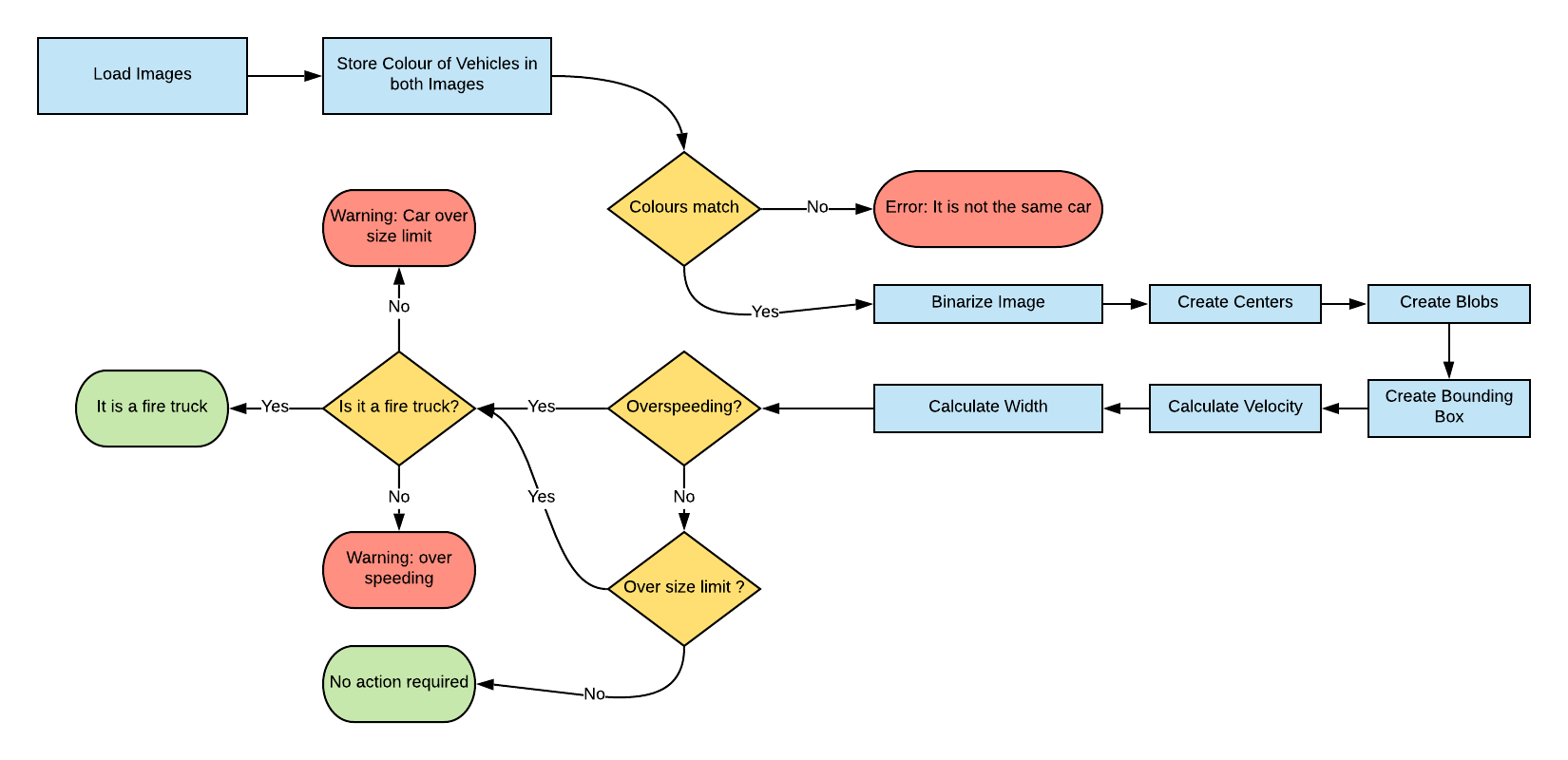
The following figure will demonstrate the flow of the application, highlighting the key processing steps:

Figure 1 Flow Diagram

### Setup Stage

In the setup stage the system loads the images from the camera and proceeds to analyse the colour of the vehicles. In order to do this, the system requests the user to input the name of the first image, followed by the second, through the command line. Moreover, the system proceeds to locate the reddest and bluest pixel in the image. If the reddest pixel is of a lower value than the bluest pixel, the system determines that the car is not a fire truck. Elsewise, if the reddest pixel is indeed of a higher value, then the system concludes that it is a fire truck. Afterwards, the system performs the first check, in order to determine if the two images are of the same car using their colours as a determining factor.

### Processing Stage

In this stage, the system starts to process the inputted pictures, in order to extract information regarding the speed and width of the vehicle. This is done in multiple key steps. First, the system binarizes the images, this is key in order to be able to extract the blobs in the image. Next the system proceeds to calculate and store the centres of the blobs, the area of each blob, and retrieve the coordinates of the bounding boxes around each blob. The application will then try to locate a blob with the largest area filled. A value of 20000 pixel is chosen to be the threshold of this test. That is to say, any blob with an area of 20000 pixels or larger is assumed to be the car. The rest of the blobs are then discarded as noise. These two stages are then repeated for the second image.

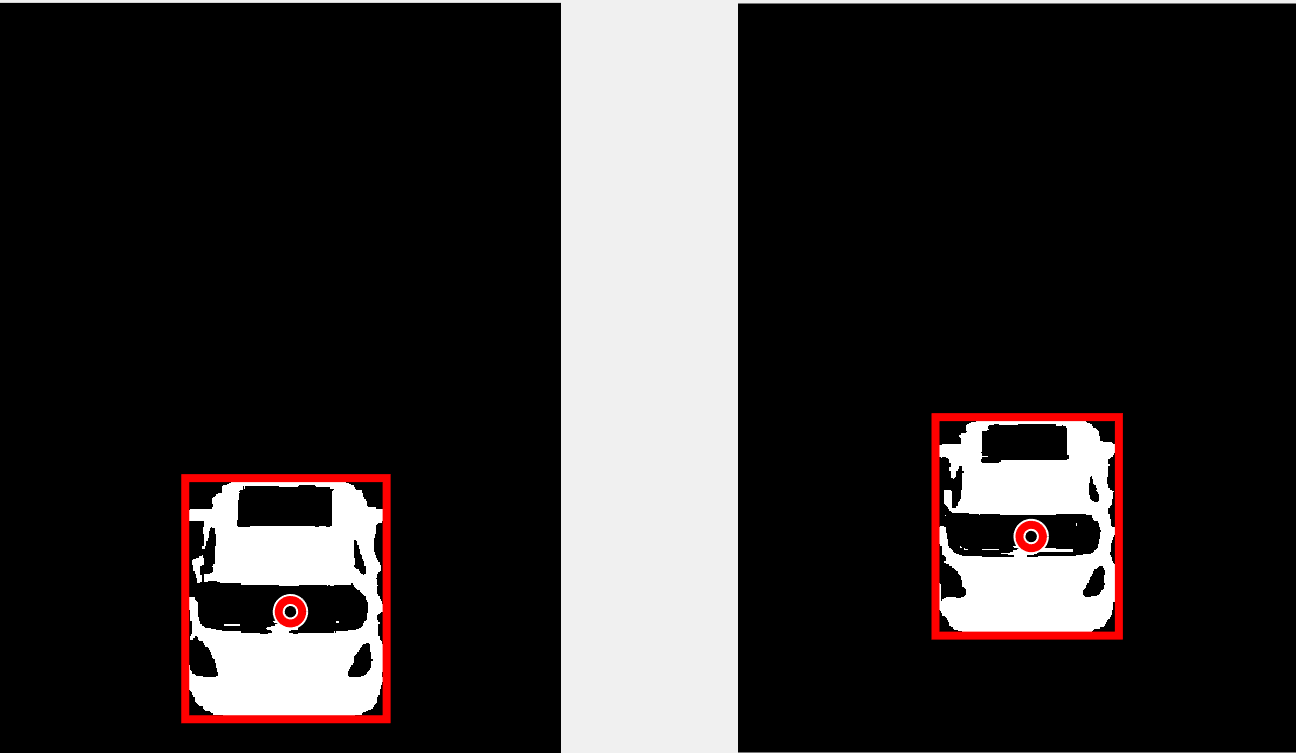
Now that both images have been processed, the system will display the processed images. The following figures showcase an example of this.

Figure 001.jpg vs 002.jpg processed

Figure 001.jpg vs 002.jpg unprocessed

### Analysis Stage

This final stage is concerned with determining if the vehicle is violating any of the two requirements.

First, the speed of the car is calculated. Using the following formula:

It is clear that in order to determine the velocity of the vehicle, a calculation of the difference of distance is needed. This can be done using trigonometry as follows:

Therefore,

Knowing that the camera is at an angle of 60-degrees and positioned 7 meters above the ground. The difference in distance between the vehicle position in both images is measured and used to determine the velocity of the vehicle.

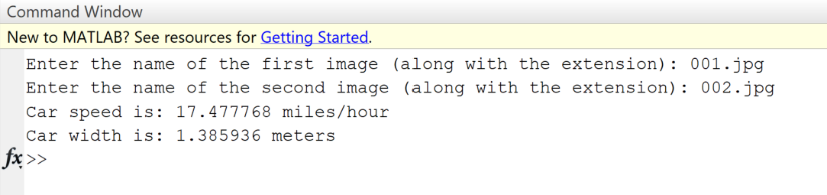
After that, the width of the car in pixels is calculated while accounting for the distance to the car. After the calculations are complete both are presented in the command window.

Figure 001.jpg vs 002.jpg output

### Warning Stage

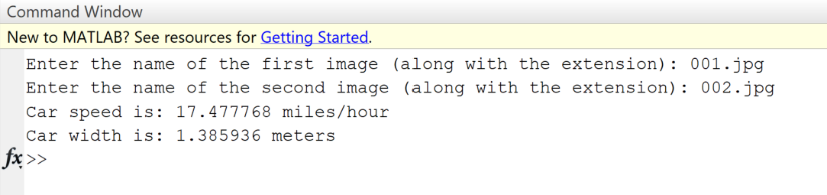
The final stage in this system is concerned with checking if the vehicle is committing any traffic violations according to the requirements presented. This is done through the use of simple if-else statements to determine if the car is over the speed limit, over the width limit and if it is exempted due to being a fire truck. The following figures will present all possible outcomes in the command window:

Figure 001.jpg vs 002.jpg (No warning)

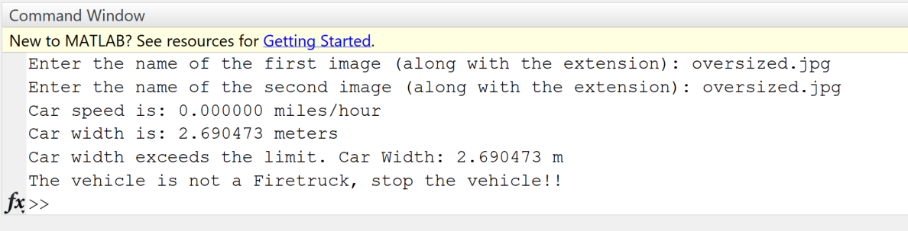
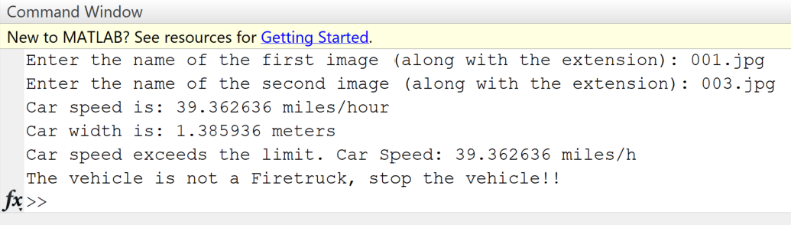


Figure oversized.jpg (Width Violation)

Figure 001.jpg vs 003.jpg (Speeding Violation)

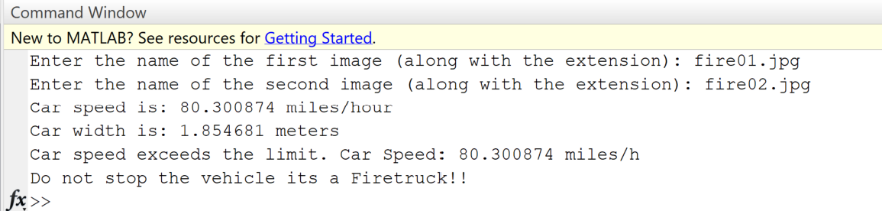


Figure fire01.jpg vs fire02.jpg (Fire Truck exception)

## Conclusion

This report showcases the design and implementation of task two. This task involves creating a computer vision application, specifically a traffic monitoring system, designed to monitor vehicles entering a tunnel. The system contains a command line that can be used to test it. The report further highlights the key processing steps used to achieve this feat. Examples of the outputs as well as the processed images are provided as proof of completing all the desired requirements. Not to mention, further tests can be easily done by running the program and following the command line.