M30242 Graphics and Computer Vision 2021-22 Coursework

Part 1

UP901354

Word count: 995 (excluding headings, image labels, etc.)

Contents

Assumptions

- Each frame contains only one vehicle
    - If this were not an assumption, we would have to match features between the 2 images to find the same vehicle in both. This would be similar to the process of feature matching in stereo vision – in stereo vision the difference between the 2 images is that they're taken from 2 different locations, with this approach the images are instead taken at 2 different times.
- Fire engines are mostly red, with a width/length ratio of approximately 1:3
    - If this were not an assumption, we would need another way of distinguishing fire engines, as with our current method other vehicles (such as buses) could be mistaken.
- Camera pixels are square and equal to 0.042 degrees
    - This allows us to more easily calculate angles used to find the size of objects. If this was not given, we would likely have to calibrate the camera before processing.

Flow Diagram

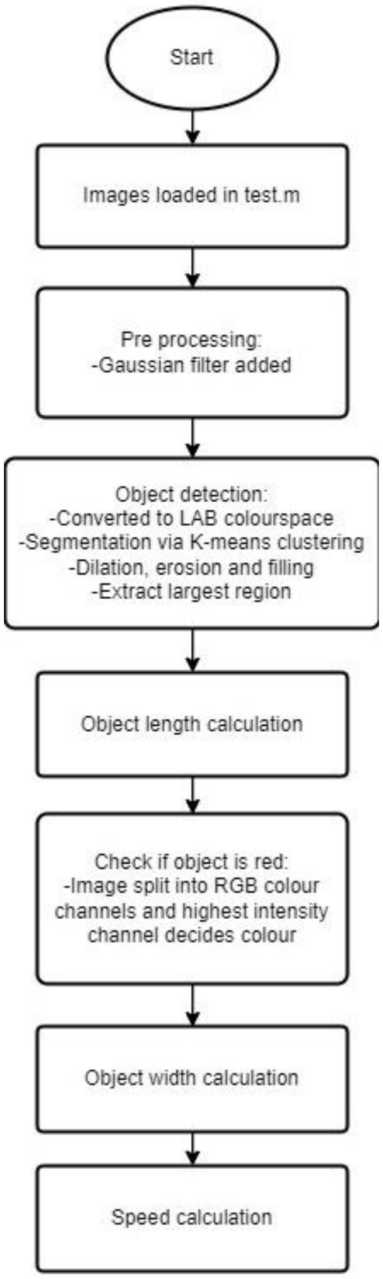

Start
Images loaded in test.m
Pre processing:
-Gaussian filter added
Object detection:
-Converted to LAB colourspace
-Segmentation via K-means clustering
-Dilation, erosion and filling
-Extract largest region
Object length calculation
Check if object is red:
-Image split into RGB colour channels and highest intensity channel decides colour
Object width calculation
Speed calculation

Figure 1 – Flow diagram

Main processing steps and overview of functions

Object detection (object_detection.m)

- This function takes the image and returns the bounding box and centroid.
- The images, which have had a gaussian filter applied to them, are passed to the object_detection function one at a time.
- The image is converted to the LAB colour space, and then we remove the L (lightness) channel, leaving us with the 2 colour channels (A and B).
- Imsegkmeans used to segment the image into a predefined number of colours (2).
    - This method was decided upon after testing a few methods of edge detection and getting less reliable results.
- Colours inverted and the image is dilated, eroded and filled to further isolate the object (and get as close to the original shape as possible).
- Image labelled (mostly to show the region to the user), and then the largest region is selected using bwareafilt.
- Bounding box and centroid calculated with regionprops.
- Bounding box applied over original image.

Object length estimation (get_length.m)

- This function takes the height (in pixels) of the object (C to D in figure 2) and the distance (in pixels) between the bottom of the object and the bottom of the frame (A to C in figure 2). It returns the length in meters.
- I used the following triangles along with trigonometric functions and the sine rule to calculate the length:

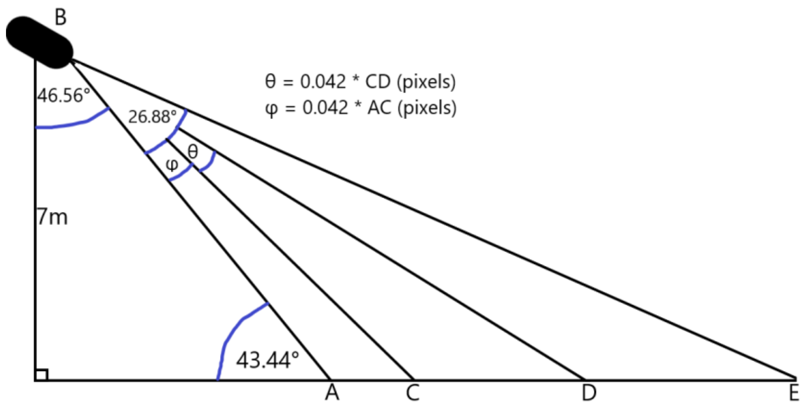


Figure 2 – Triangles used for length estimation

    - With A to E being the cameras field of view, B being the camera and C to D being the location of the vehicle.

```
CAB=136.56;
CBD=CD_pixels*0.042;
ABC=AC_pixels*0.042;
ACB=180-CAB-ABC;
BCD=180-ACB;
CDB=180-CBD-BCD;

AB = (7*(tand(46.46)))/(sind(46.56));
AC = (AB/sind(ACB))*sind(ABC);
CB = (AC/sind(ABC))*sind(CAB);
DB = (CB/sind(CDB)) * sind(BCD);
CD = (DB/sind(BCD)) * sind(CBD);
```

Figure 3 – Length estimation code

Check is object is red (checkRed.m)

- Image and bounding box are passed into this function.
- Image is cropped to bounding box coordinates and then split into RGB channels.
- Mean value for each channel calculated and stored in an array (meanRGB).
- Values compared against sample values (taken from the fire engine images provided).
    - Another method tried was checking if the largest value of the array is the red channel, then assuming that the object is red - this would be less reliable.

Object width estimation (get_width.m)

- To calculate width, I used the following method:

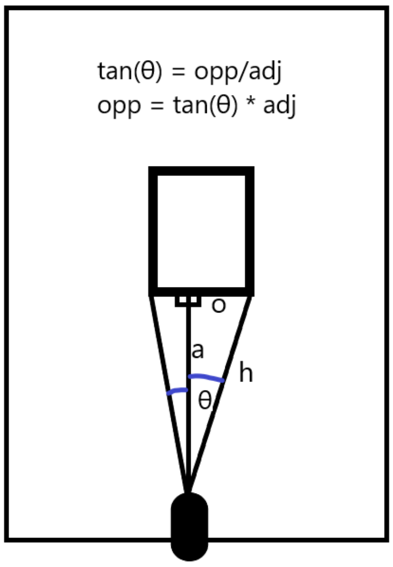
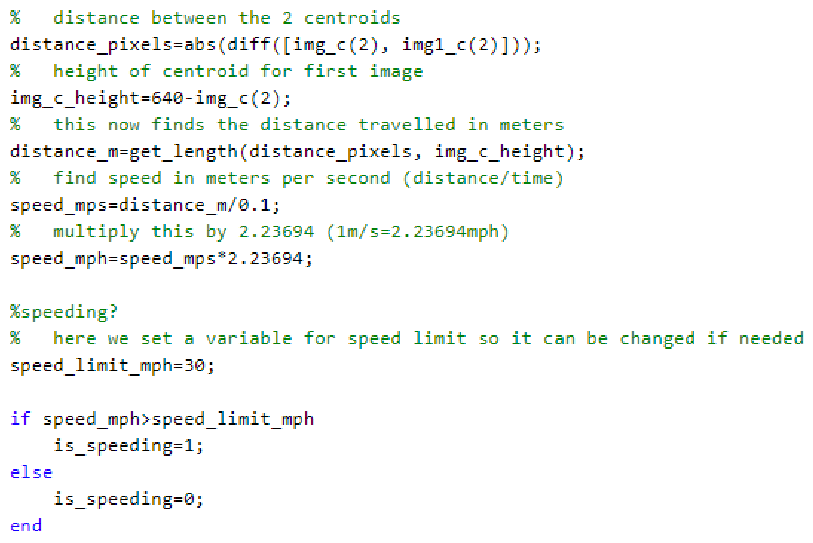


Figure 4 – Width calculation method

- The adjacent side here is the distance from the vehicle to the camera. During the length calculation (get_length.m), CB and DB are also calculated (please see figure 2) – these are then used in this width estimation method, along with the bounding box coordinates, to calculate the width at both the top and bottom of the bounding box. These are averaged to give the estimated width from the centre of the vehicle.

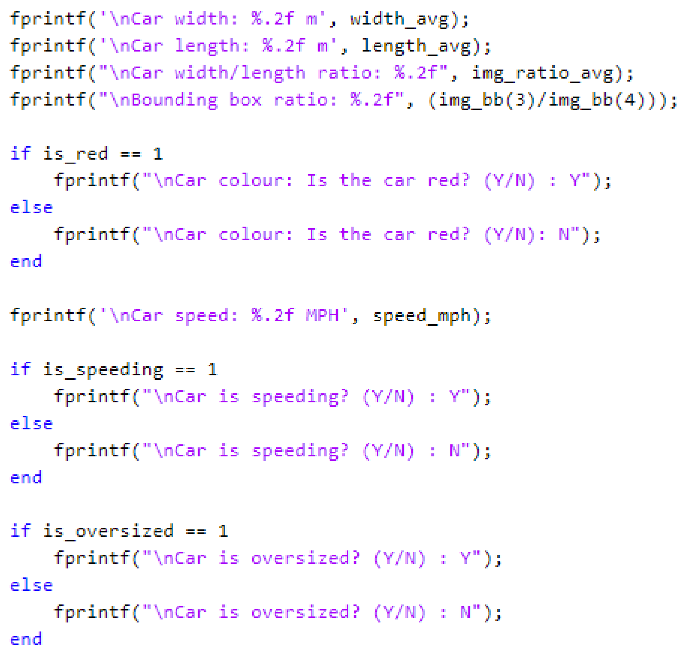
The main function (traffic_cam.m)

- This function is mostly used to call the other functions of this program, but also handles simple calculation as well as output.
- Some examples:



```
%   distance between the 2 centroids
distance_pixels=abs(diff([img_c(2), img1_c(2)]));
%   height of centroid for first image
img_c_height=640-img_c(2);
%   this now finds the distance travelled in meters
distance_m=get_length(distance_pixels, img_c_height);
%   find speed in meters per second (distance/time)
speed_mps=distance_m/0.1;
%   multiply this by 2.23694 (1m/s=2.23694mph)
speed_mph=speed_mps*2.23694;

%speeding?
%   here we set a variable for speed limit so it can be changed if needed
speed_limit_mph=30;

if speed_mph>speed_limit_mph
    is_speeding=1;
else
    is_speeding=0;
end
```

Figure 5 – Some of the simple logic of the main function



```
fprintf('\nCar width: %.2f m', width_avg);
fprintf('\nCar length: %.2f m', length_avg);
fprintf("\nCar width/length ratio: %.2f", img_ratio_avg);
fprintf("\nBounding box ratio: %.2f", (img_bb(3)/img_bb(4)));

if is_red == 1
    fprintf("\nCar colour: Is the car red? (Y/N) : Y");
else
    fprintf("\nCar colour: Is the car red? (Y/N): N");
end

fprintf('\nCar speed: %.2f MPH', speed_mph);

if is_speeding == 1
    fprintf("\nCar is speeding? (Y/N) : Y");
else
    fprintf("\nCar is speeding? (Y/N) : N");
end

if is_oversized == 1
    fprintf("\nCar is oversized? (Y/N) : Y");
else
    fprintf("\nCar is oversized? (Y/N) : N");
end
```

Figure 6 – Printing results from main function

Testing

- Gaussian filter added.

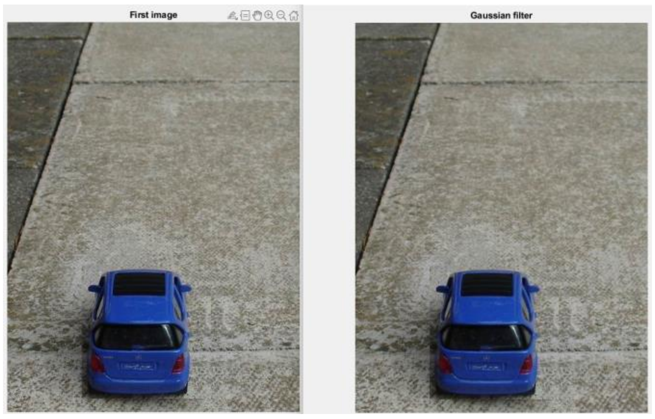


Figure 7 – Gaussian filter

  - A subtle difference but helps to merge some of the detail in the road - this makes it easier to separate the road and the car later on.
- K-means clustering used to separate colours of image.

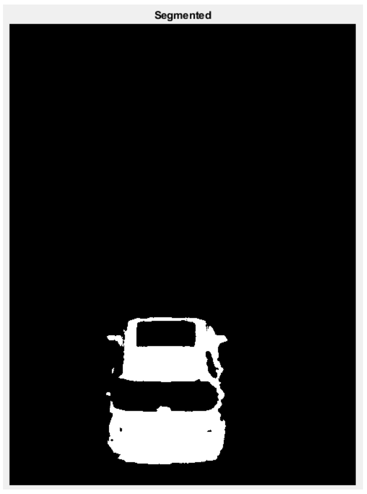


Figure 8 – Segmentation via k-means clustering result

- Further processing (dilating, filling, eroding) used to isolate object.

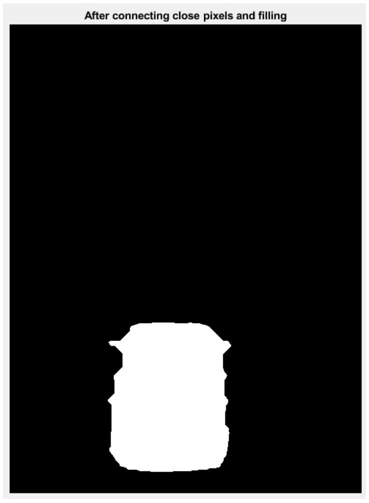


Figure 9 – Isolated object

- Draw bounding box of largest region on original image.
    - The k-means clustering searches for 2 colours (in this case the road and the vehicle) – this has proven to be effective for this set of images, however to ensure the correct region is picked (if it were used with other images), we use the largest one.

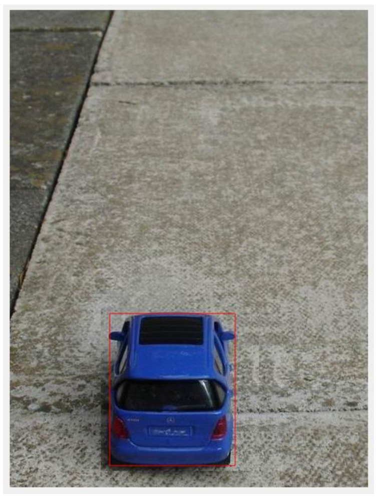


Figure 10 – Bounding box

- The length is calculated for both images (see figure 2 & 3) and then an average is used for the result and subsequent calculations. The average is used as this reduces the impact of any imperfections the images could have (for example, if one of the images is blurred).
- Decide if vehicle is red by separating object into 3 colour channels.

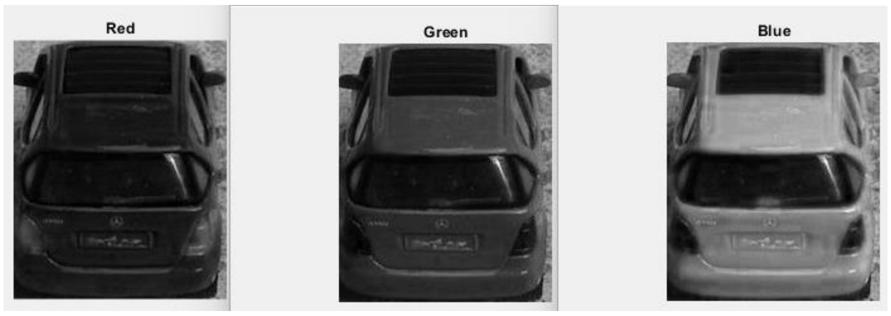


Figure 11– RGB channels

    - If the mean values for these channels match the sample colour (taken from fire01 and 02 images), vehicle is red.
    - Another approach was to check if the red channel was the largest value and assume based off that, but this was less accurate.
- Calculate width for both images (see figure 4) and then take an average of the two. As with the length, this average aims to reduce the impact of any imperfections with the images.
- Speed is calculated by using the centroids of the bounding boxes for both images to calculate the distance travelled (using the same method as calculating the length of the vehicle). This is then divided by 0.1 seconds (the interval for the images) and then multiplied by 2.23694 to convert from meters per second to miles per hour. Figure 5 shows the code for this.
- The results are printed to the console.

```
Car width: 1.39 m
Car length: 2.82 m
Car width/length ratio: 0.49
Car colour: Is the car red? (Y/N): N
Car speed: 20.07 MPH
Car is speeding? (Y/N) : N
Car is oversized? (Y/N) : N
Car is fire engine? (Y/N) : N

No issues found>>
```

Figure 12 – Console output

Problems

The average width/length ratio for the fire engine is calculated as 0.37 – while still close to 1/3rd (and the specification says approximately 1:3), it still seems slightly high. In my program I have set the width/length ratio required to be identified as a fire engine to above 0.22 and below 0.38.

Initially, the length was calculated from the below formula – however, this caused issues due to being both slightly incorrect and confusing to read in the code. A simpler approach was used.

$$CD = Sin(CBD)\left(\frac{(Sin(136.56)\left(\frac{10.18}{Sin(ACB)}\right)}{Sin(CDB)}\right)$$

Figure 13 - Original (incorrect) calculation

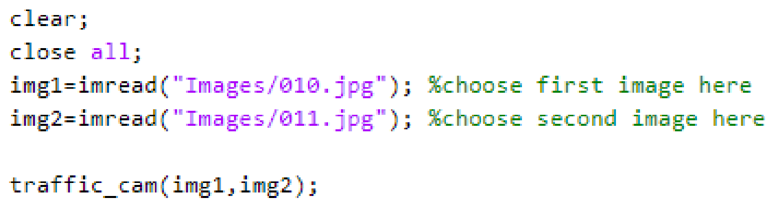
Additional Testing

```
clear;
close all;
img1=imread("Images/fire01.jpg"); %choose first image here
img2=imread("Images/fire02.jpg"); %choose second image here

traffic_cam(img1,img2);
```

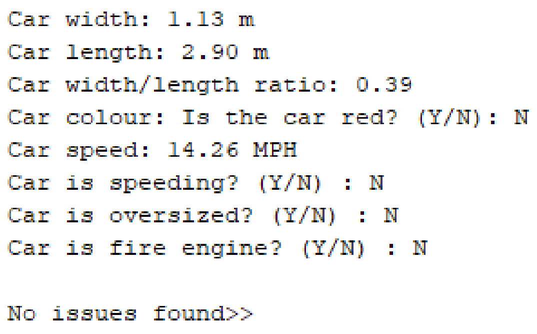
```
Car width: 2.07 m
Car length: 5.56 m
Car width/length ratio: 0.37
Car colour: Is the car red? (Y/N) : Y
Car speed: 96.24 MPH
Car is speeding? (Y/N) : Y
Car is oversized? (Y/N) : N
Car is fire engine? (Y/N) : Y

Vehicle is exempt from speeding
No issues found>>
```

```
clear;
close all;
img1=imread("Images/002.jpg"); %choose first image here
img2=imread("Images/003.jpg"); %choose second image here

traffic_cam(img1,img2);
```

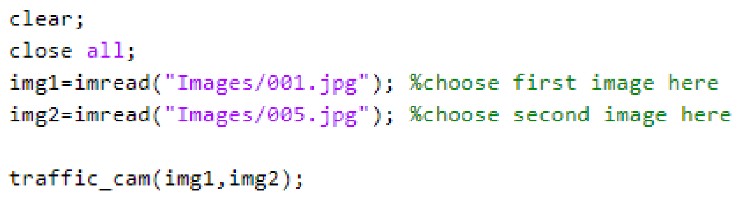
```
Car width: 1.34 m
Car length: 2.88 m
Car width/length ratio: 0.47
Car colour: Is the car red? (Y/N): N
Car speed: 24.60 MPH
Car is speeding? (Y/N) : N
Car is oversized? (Y/N) : N
Car is fire engine? (Y/N) : N

No issues found>>
```



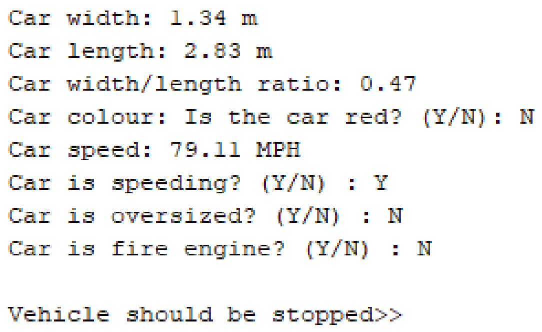
```
clear;
close all;
img1=imread("Images/010.jpg"); %choose first image here
img2=imread("Images/011.jpg"); %choose second image here

traffic_cam(img1,img2);
```



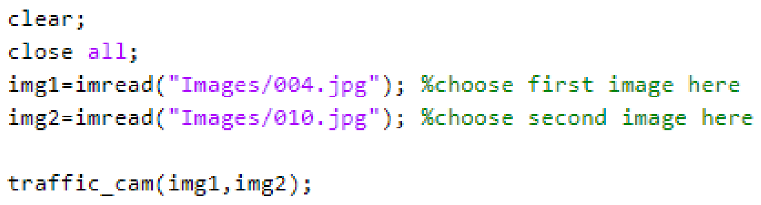
```
Car width: 1.13 m
Car length: 2.90 m
Car width/length ratio: 0.39
Car colour: Is the car red? (Y/N): N
Car speed: 14.26 MPH
Car is speeding? (Y/N) : N
Car is oversized? (Y/N) : N
Car is fire engine? (Y/N) : N

No issues found>>
```



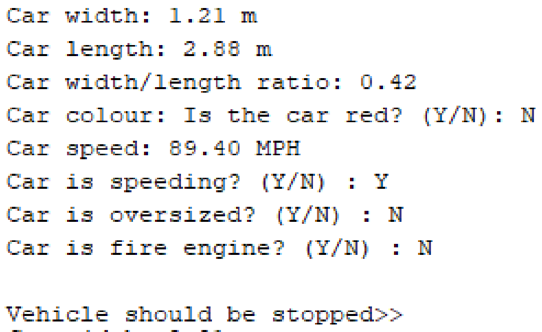
```
clear;
close all;
img1=imread("Images/001.jpg"); %choose first image here
img2=imread("Images/005.jpg"); %choose second image here

traffic_cam(img1,img2);
```



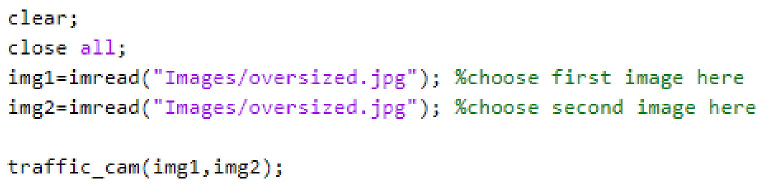
```
Car width: 1.34 m
Car length: 2.83 m
Car width/length ratio: 0.47
Car colour: Is the car red? (Y/N): N
Car speed: 79.11 MPH
Car is speeding? (Y/N) : Y
Car is oversized? (Y/N) : N
Car is fire engine? (Y/N) : N

Vehicle should be stopped>>
```



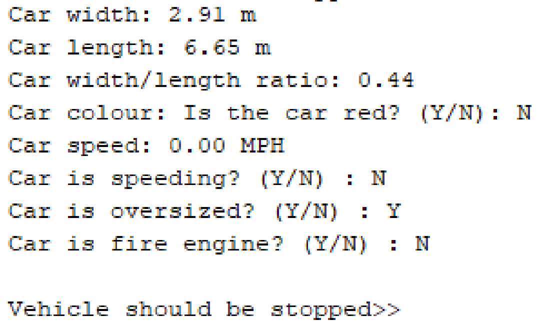
```
clear;
close all;
img1=imread("Images/004.jpg"); %choose first image here
img2=imread("Images/010.jpg"); %choose second image here

traffic_cam(img1,img2);
```



```
Car width: 1.21 m
Car length: 2.88 m
Car width/length ratio: 0.42
Car colour: Is the car red? (Y/N): N
Car speed: 89.40 MPH
Car is speeding? (Y/N) : Y
Car is oversized? (Y/N) : N
Car is fire engine? (Y/N) : N

Vehicle should be stopped>>
```



```
clear;
close all;
img1=imread("Images/oversized.jpg"); %choose first image here
img2=imread("Images/oversized.jpg"); %choose second image here

traffic_cam(img1,img2);
```



```
Car width: 2.91 m
Car length: 6.65 m
Car width/length ratio: 0.44
Car colour: Is the car red? (Y/N): N
Car speed: 0.00 MPH
Car is speeding? (Y/N) : N
Car is oversized? (Y/N) : Y
Car is fire engine? (Y/N) : N

Vehicle should be stopped>>
```

UP901354