Name: Datta Sainath Dwarampudi (dsd298)

REPORT ON DETECTION OF WRONG DIRECTION DRIVING VEHICLES

Note: This project is completely modeled for left hand driving countries like India.

Introduction

Need of a system which can detect the vehicles driving in the wrong direction is very important for developing countries like India where one-third of the accidents occur due to the wrong direction driving vehicles. Not only people driving in the wrong direction suffer during the accidents but also people driving in the right direction suffer too as the people driving in the wrong direction crash into them. So, I wanted to design a system which can alert the vehicles whenever it detects any vehicles moving in the wrong direction. This can be a very good system as it alerts the people moving in right direction and enforces them to be extra careful.

Motivation and Problem Identification Study:

My childhood memories in India have motivated me to consider the above stated problem and to solve it. In developing country like India most of the 2 lane roads do not have a concrete road divider which gives rise to reckless driving (driving vehicles in wrong direction). I have discussed in detail about two popular scenarios of people driving in wrong direction.

Scenario 1 (Overtaking):

IDEAL: According to the traffic rules of India, there should be no overtaking when there is a single lane for both the directions of traffic as shown in the figure 1.

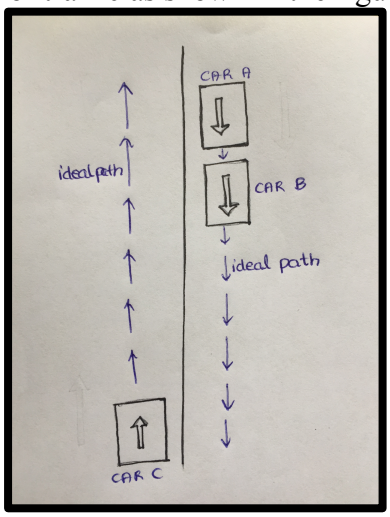
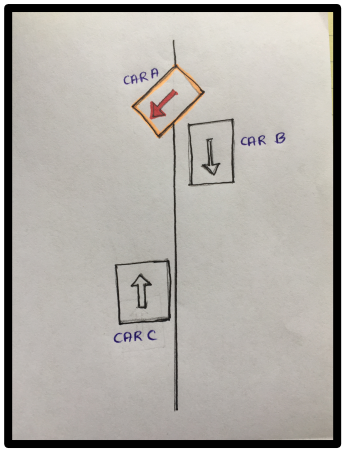
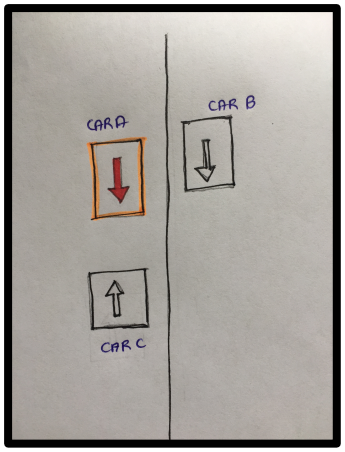


Figure 1 shows the ideal way of driving without overtaking vehicles

REALITY: Most of them do not adhere to that rule and overtake the vehicles by moving onto other lane (opposite traffic lane) as shown in figure 2. There is a very high probability that the overtaking car (Car A), the car being overtaken (Car B) and the car travelling on the opposite lane (Car C) can meet with an accident.

Name: Datta Sainath Dwarampudi (dsd298)





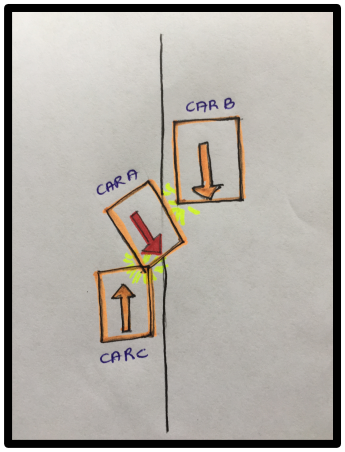


Figure 2(i) **Figure 2(ii)** **Figure 2(iii)**

Figure 2(i), 2(ii) & 2(iii) shows overtaking of a vehicle by driving in the wrong direction.

Scenario 2 (Lazy/Reckless Driving)

IDEAL: As shown in the figure 3, the Car A should cross over to the right lane and then cross into the desired street.

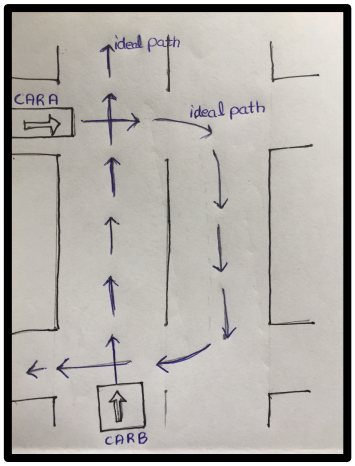
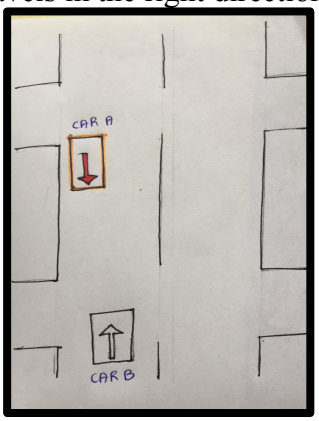


Figure 3 shows the ideal way of driving of vehicles.

REALITY: Lazy or reckless driving is one of the most common practices by common man used for saving fuel and time. As shown in figure 4, the Car A travels in the left lane (wrong lane) and tries to sneak into the adjacent lane. This can cause an accident if at the same time there is another car like Car B which travels in the right direction but ends up in an accident.



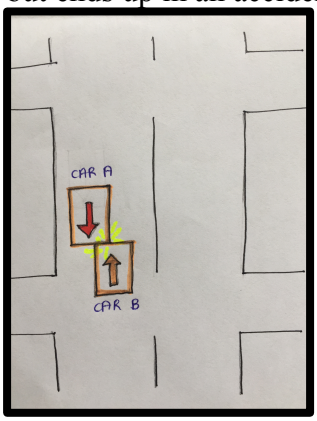


Figure 4 (i) **Figure 4(ii)**

Figure 4 (i) & 4(ii) illustrating the reckless driving by driving in wrong direction to save fuel and time

Design of solution:

We can design a model as a solution for the above stated problem by setting up a device which consist of an SBC (Single-Board Computer) like Raspberry Pi, a siren, a red LED light. The raspberry pi can handle the computer vision processing work and give inputs to siren and LED lights to function appropriately.

The solution model will consist of two blocks:

- **First Block**: Intelligence to detect whether there is any vehicle driving in wrong direction using Computer Vision.
- **Second Block**: Using the output generated from the first block , the second block can be used to ring the siren and glow the red LED light whenever wrong direction driving vehicle is detected on either/both sides of the road.

By implementing the above solution model, we can alert the vehicles on either side of the road to be extra careful. In this project, we will only deal with the first block (computer vision part)of the above-mentioned model.

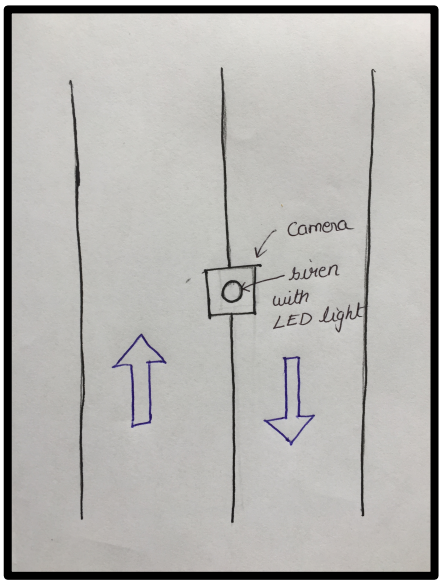


Figure 5(i)

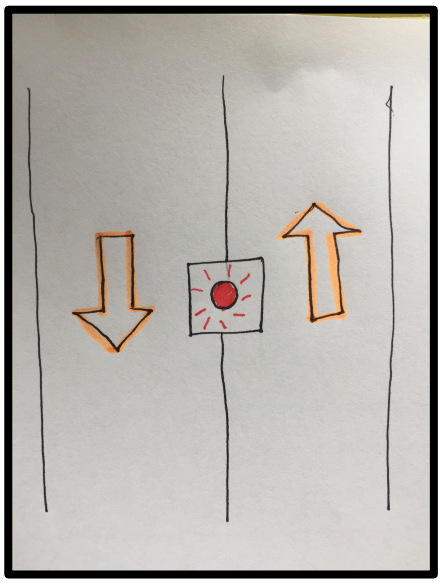


Figure 5 (ii)

Figure 5 (ii) shows that whenever the camera detects vehicle in wrong direction, the siren rings and the red LED light glows.

Name: Datta Sainath Dwarampudi (dsd298)

Outline of the complete project:

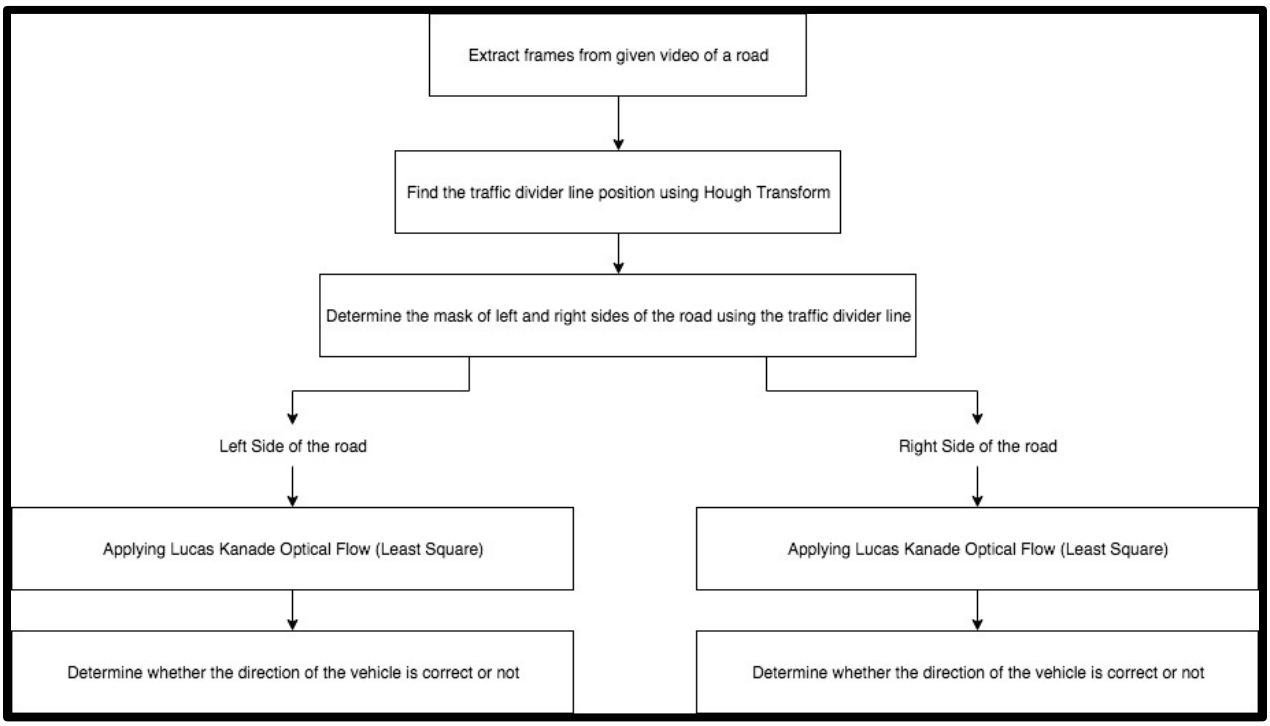


Figure 6 shows the outline of the complete project

The program initially extracts frames from the video of a road given to the program as input. We extract 2 frames from all the frames extracted from the video and then choose any one of the frame to determine the traffic divider line in the frames using Hough Transform. After determining the traffic divider line, we create masks for the left side of the road and right side of the road to individually analyze both the sides of the road. On each side of the road, I applied Lucas Kanade Optical Flow Method to extract the directions of each pixel wherever there was motion. After determining the directions of each pixel, I determined the overall direction of the vehicle on each side of the road and outputted whether the vehicles were travelling in the right direction or not.

Detailed Analysis of each image processing step:

STEP 1: Extract frames from a given video of a road:

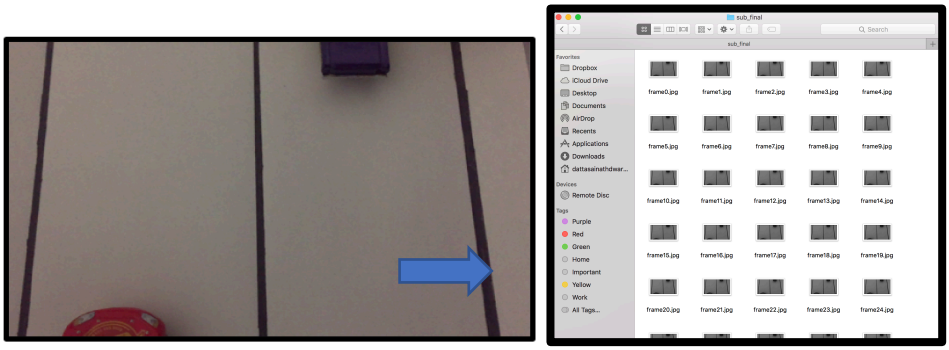


Figure 7 shows the video converted to frames in a folder.

To extract the frames from the video, I have used the built-in functions of OpenCV.

STEP 2: Find the traffic divider line position using Hough Transform

STEP 2(a): Choosing 2 frames for analysis from all the extracted frames.

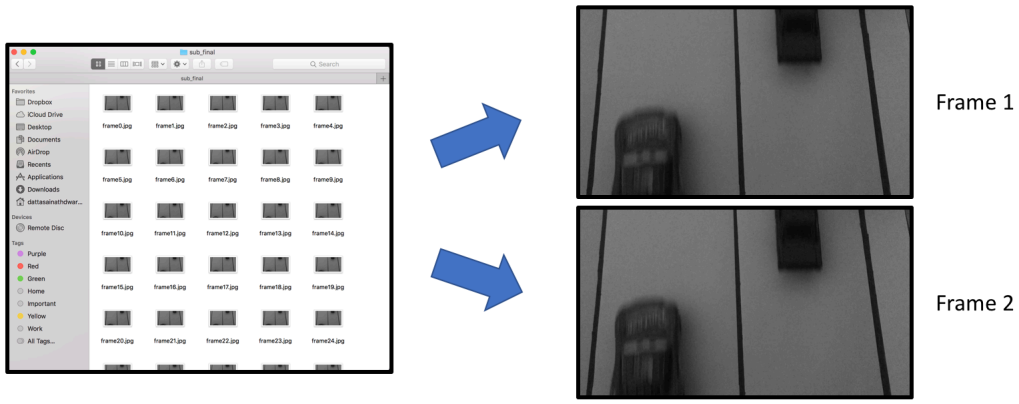


Figure 8 shows extraction of frames from the video.

Explanation: I have taken only 2 frames from the all the frames extracted from video. The logic applied to selection was to choose the 65th percent and 66th percent frames of all the frames extracted. I have chosen 65th and 66th percent frames because the cars were visible during those frames for the videos taken by me.
Choosing frames for a real time monitoring: Initially take the image of an empty road and capture every frame in real time. For every frame captured do a correlation with the empty road frame captured, then choose the frame which has the least correlation among all the frames collected around few seconds.

STEP 2(b): Resizing the frames from 1080 X 1920 to 200 X 400 size

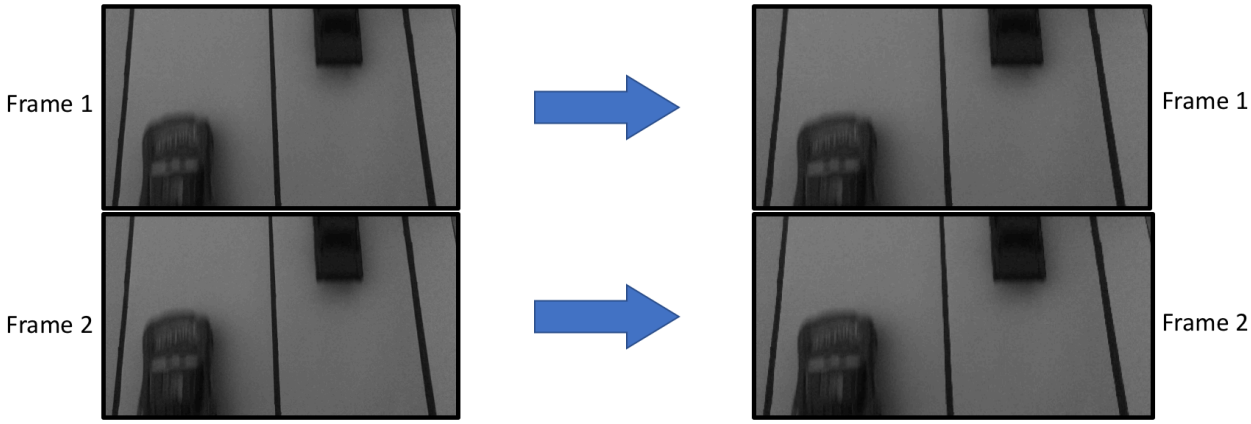


Figure 9 shows the resized image from 1080 X 1920 to 200 X 400.

Explanation: I have reduced the resolution of image from 1080 X 1920 to 200 X 400 because:

1. To reduce the computation time while computing the Hough Transform
2. By reducing the resolution of the images, the displacement across the corresponding pixels between the 2 frames will be less which is very important for Lucas Kanade Optical Flow Method because Lucas Kanade Optical Flow Method is fruitful only when displacement between frames is small.

STEP 2(c): Preparing the image for the Hough Transform

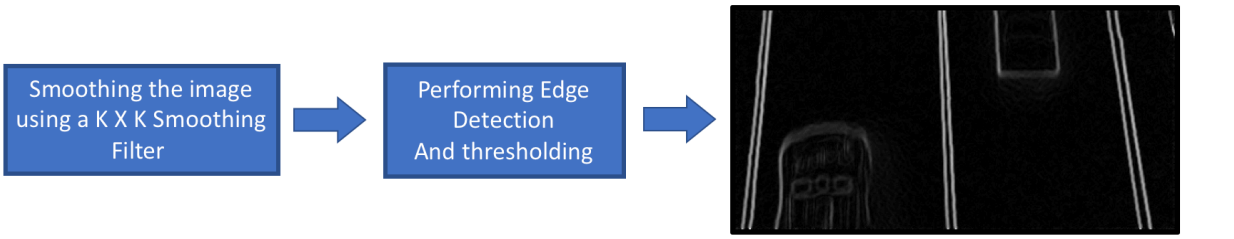


Figure 10 shows the pre-processing steps of Hough Transform

Explanation: Smoothing an image is very important before doing edge detection because it smoothens or removes many unwanted sharp peaks in the image.
I performed Edge Detection by calculating x gradient and y gradient. Using x and y gradients, I found the magnitude and orientation of edges.
The most important step is thresholding: I thresholded the edges obtained after edge detection because to find the single traffic divider line, we don't need edges of car/ vehicle and I found out that the edges on the car are usually of small magnitude and edges of traffic divider was of high magnitude. So, I choose a threshold value and filtered all the weak edges. I performed several experimentations on several videos before choosing a generalized threshold value.

STEP 2(d): Performing Hough Transform on the edge detected image:

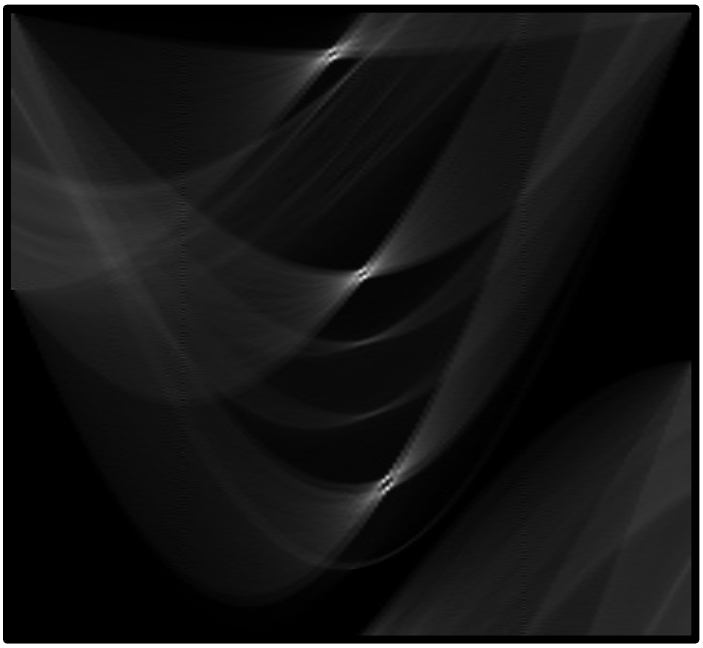


Figure 11 shows Hough Space generated from one of the selected frames

Explanation: I have added the (pixel_value/255) to the Hough space accumulator rather than adding 1 to plot points on accumulator. Since, we can see that the divider was carrying the maximum intensity. We can clearly see 3 pairs of bright spots in the accumulator. 3 pairs of spots represent one road divider in the middle of the road and two margin lines on either side of the road.

Optimization of Hough Transform:
I tried using the orientation of edges and optimizing the Hough Transform but I could get good results only when the Road divider line was clearly visible, if the image had lots of noise, it was tough to detect the divider. I tested this by adding gaussian to the original video and then run the optimized Hough Transform.

STEP 2(e): Detection of the divider line from the accumulator produced by Hough transform

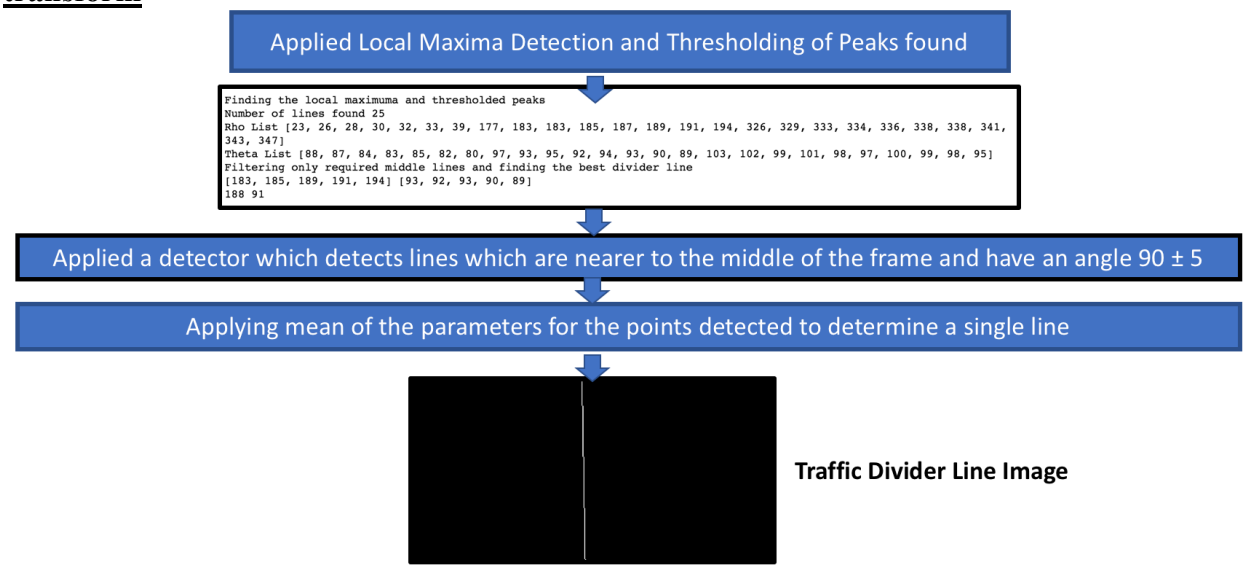


Figure 12 shows flowchart for detecting the traffic divider line from the Hough Transform

1. *Applied Local Maxima Detection and Thresholding of Peaks found:* I have chosen only the pixels which had more intensity than all of its N-8 neighborhood. I could get better results when I applied a threshold for all those pixels found. After this step, I ended up with lines which formed the traffic divider and other two margin lines on either side of the road.
2. *Detecting the Traffic Divider lines only*: Applied a detector which detects all the lines which have perpendicular distance around the middle of image and also has theta angle 90 ± 5 degrees. We can achieve this by looping over the lists obtained from the previous step and using if conditions which satisfy the previously stated conditions.
3. *Down sampling to a single Traffic Divider line:* After the 2nd Step, we obtain a group of points which satisfy the previously stated condition. To obtain only a single line, I have taken the arithmetic mean of the parameters for the lines obtained.

STEP 3(a): Determine the masks of left and right sides of the road using the traffic divider line

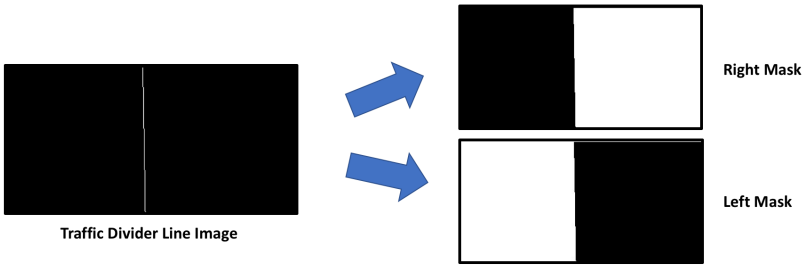


Figure 13 shows the formation of left and right masks using the divider line.

We can obtain the right and left masks from the divider line using the following algorithms.
Algorithm for left mask:
Initialize a left_mask_image of same size of traffic divider image with **zero values**
For i to total_rows_of_divider_image:
    For j in total_columns_of_divider_image:
        If(divider_image[i][j] = = 255):
            break;
        else:
            left_mask_image[i][j] = 1

Algorithm for right mask:
Initialize a right_mask_image of same size of traffic divider with **ones values**
For i to total_rows_of_divider_image:
    For j in total_columns_of_divider_image:
        If(divider_image[i][j] = = 255):
            break;
        else:
            left_mask_image[i][j] = 0

STEP 3(b): Applying Masks to both the frames

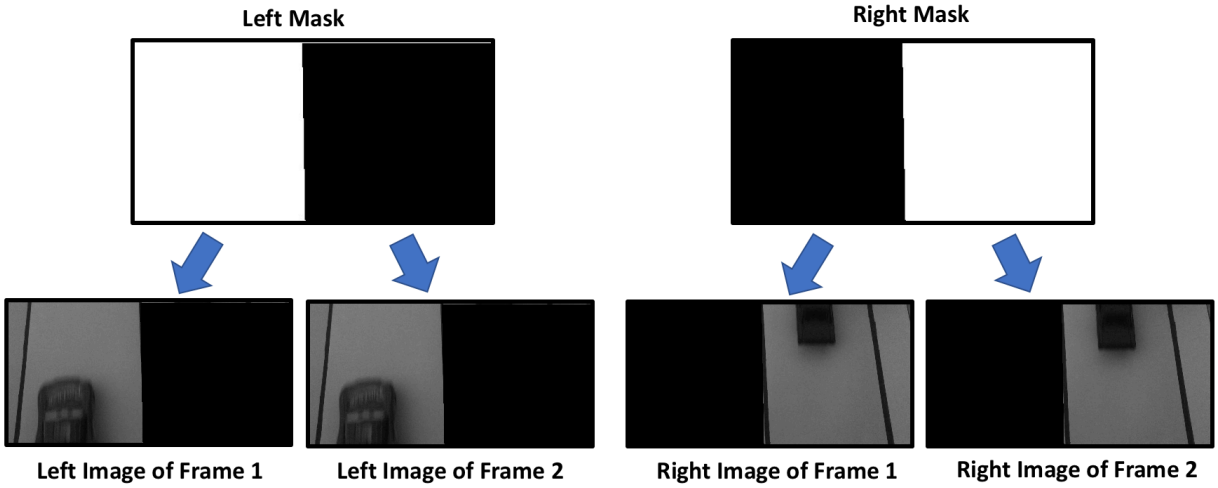


Figure 14 shows applying left and right mask on the 2 frames

We can obtain the resulting 4 images by multiplying left and right masks to each of the frames.
Logic:
left_image_of_frame1 = left_mask*frame1
left_image_of_frame2 = left_mask*frame2
right_image_of_frame1 = right_mask*frame1
right_image_of_frame2 = right_mask*frame2

Name: Datta Sainath Dwarampudi (dsd298)

STEP 4: Optical Flow on Right and Left sides of the road

STEP 4(a): Gaussian Smoothing on the images before applying optical flow method

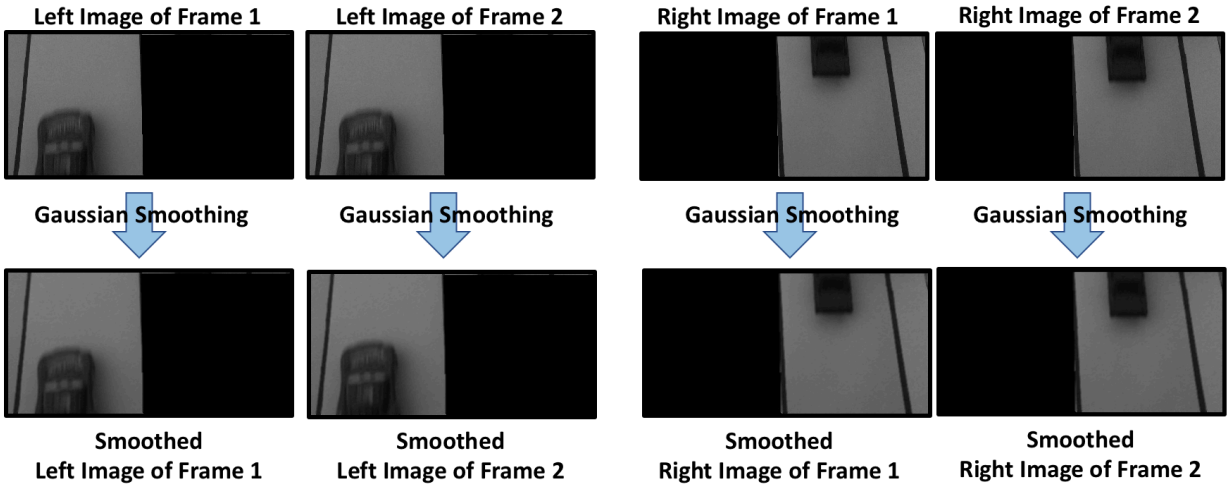


Figure 15 shows the gaussian smoothed images

Gaussian smoothing the images before applying Lucas Kanade Optical Flow Method *is the most important step*. By performing gaussian smoothing we are actually smoothing the intensities of the images which is very important otherwise during the calculation of u and v matrices of optical flow we will get lot of different uv magnitude and uv directions within a small region. By smoothing these images, we get good smoothed uv magnitude and uv directions all over. I used gaussian instead of a normal KXK smoothing filter as Gaussian is a true low-pass filter.

STEP 4(b): Determining x, y and t derivatives for both frame on left side of the road

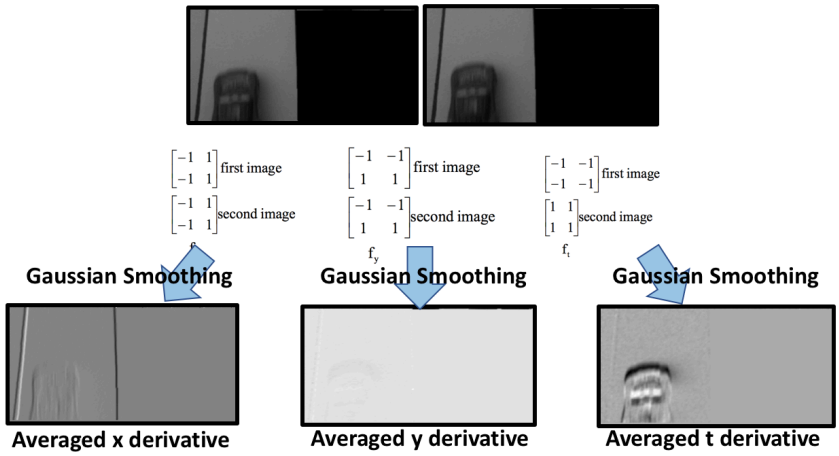


Figure 16 shows the procedure for computing the x, y and t derivatives for the left side of the road frames

Based on the recommendations of Prof. Mubarak Shah of UCF (University of Central Florida) from his lecture notes, applying a Robert's mask of 2X2 size on both the frames and then averaging the results will yield us good results for x and y derivatives. As averaging both the frames for x and y derivatives will be useful as it uses information of x, y derivatives from both the frames.

Name: Datta Sainath Dwarampudi (dsd298)

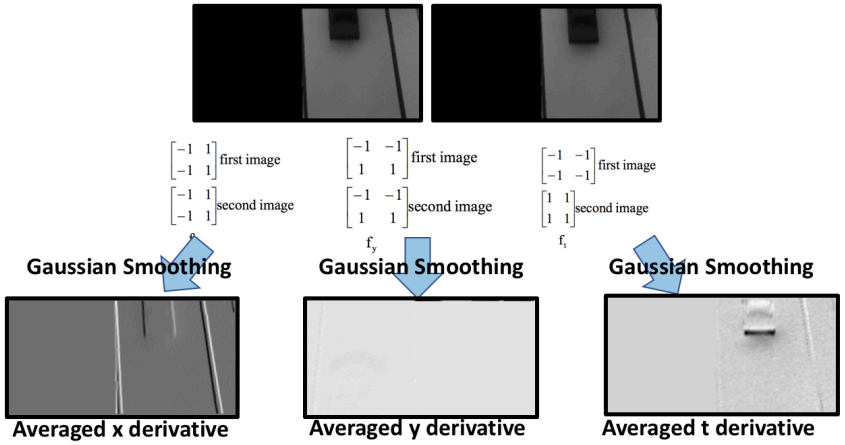


Figure 17 shows the procedure for computing the x, y and t derivatives for the right side of the road frames

STEP 4(C): Applying Lucas Kanade Method (Least Squares)

$$\sum(f_{xi}u + f_{yi}v + f_{ti})f_{xi} = 0$$
$$\sum(f_{xi}u + f_{yi}v + f_{ti})f_{yi} = 0$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum f_{xi}^2 & \sum f_{xi}f_{yi} \\ \sum f_{xi}f_{yi} & \sum f_{yi}^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum f_{xi}f_{ti} \\ -\sum f_{yi}f_{ti} \end{bmatrix}$$

$$u = \frac{-\sum f_{yi}^2 \sum f_{xi}f_{ti} + \sum f_{xi}f_{yi} \sum f_{yi}f_{ti}}{\sum f_{xi}^2 \sum f_{yi}^2 - (\sum f_{xi}f_{yi})^2}$$

$$v = \frac{\sum f_{xi}f_{ti} \sum f_{xi}f_{yi} - \sum f_{xi}^2 \sum f_{yi}f_{ti}}{\sum f_{xi}^2 \sum f_{yi}^2 - (\sum f_{xi}f_{yi})^2}$$

Most Important Formula used to calculate u and v

Figure 18 shows the important formulas which is used for Lucas Kanade Optical Flow Method.

I used a mask of 3X3 neighborhood for computation of u and v. Taking a bigger neighborhood, I found that u and v matrices calculated were very bad as it was not a smoothed u and v matrices.

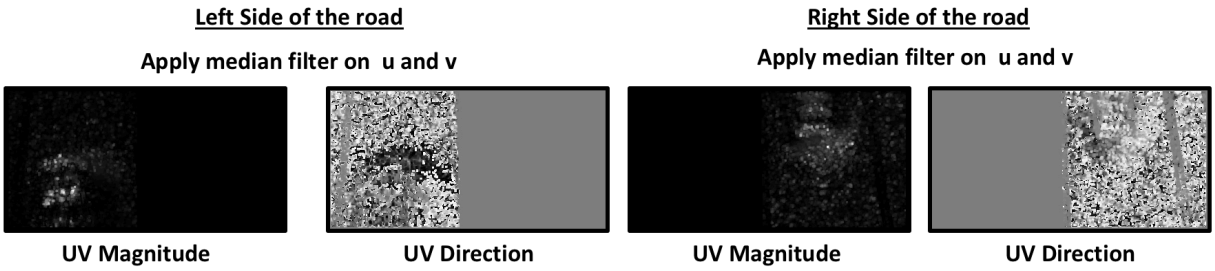


Figure 19 shows the uv Magnitudes and uv Direction obtained from both the sides of the road.

After obtaining the matrices u and v, *applying a median filter of kernel size 3X3 is the most important step* even though we have smoothed the images enough before implementing the Lucas Kanade Optical Flow Method applying a median filter is highly recommended.
We can calculate the magnitude of uv and direction also on both sides of the frame using the following formulas:

$$\text{uv\_magnitude} = \sqrt{u^2 + v^2} \qquad \text{uv\_direction} = \tan^{-1}(u/v)$$

STEP 5: Determine whether the direction of the vehicle is correct or not:

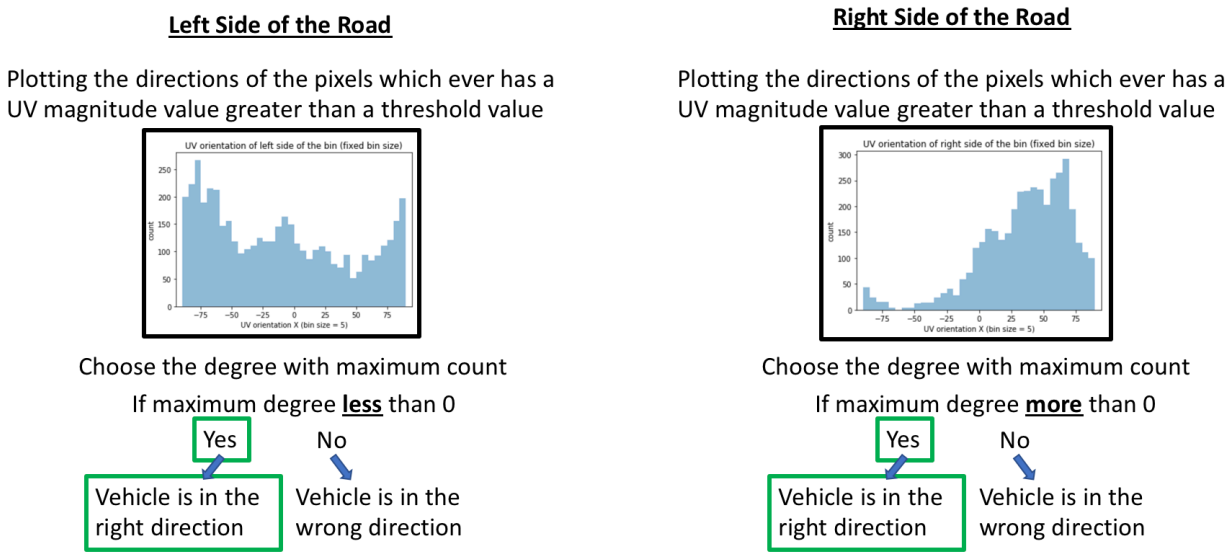


Figure 20 shows the flowchart of determining the direction of vehicles on both the sides of road.

Algorithm/Pseudocode to determine the direction of the vehicle on the road.

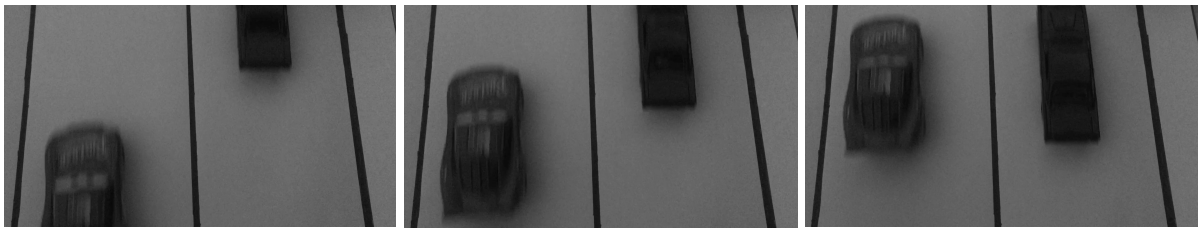
```
Filtered_directions_list = empty_list
For i to total_rows_of_u:
        For j in total_columns_of_u:
                If(uv_magnitude[i][j] >  10):
                        Add uv_direction[i][j] to Filtered_directions_list

Maximum_occourence_angle = determinte_maximum_occourence(Filtered_directions_list)
```

Explanation: I have chosen a threshold value which is just above zero after conducting several experiments. After determining the list of direction which have magnitude values greater than threshold values, choose the angle which has the maximum occurrences in that list.

Analysis on the left side of the road: If it is less than zero, than the vehicle is travelling in the right direction on the left side of the road otherwise it is travelling in the wrong direction.
Analysis on the right side of the road: If it is less than zero, than the vehicle is travelling in the right direction on the left side of the road otherwise it is travelling in the wrong direction.

Name: Datta Sainath Dwarampudi (dsd298)

Final Result:



Frame 87 Frame 92 Frame 97

Figure 21 shows few frames of a video to show the motion of vehicles in the video

Expected Results:

Left Side of the Road: Correct Direction *Right Side of the Road*: Correct Direction

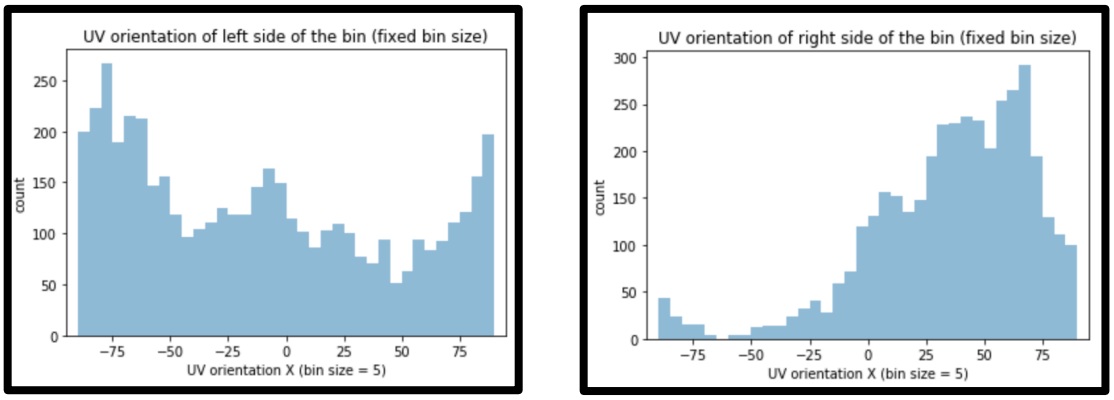


Figure 22 shows the uv directions on left and right side of the road.

```
FINAL RESULTS
The vehicle on the LEFT SIDE OF THE ROAD IS IN THE CORRECT DIRECTION
The vehicle on the RIGHT SIDE OF THE ROAD IS IN THE CORRECT DIRECTION
```

Figure 23 shows the output from the terminal

Other Results Obtained:

Scenario 2:

Frame 42 Frame 47 Frame 52

Figure 24 shows few frames of a video to show the motion of vehicles in the video

Expected Results:

Left Side of the Road: Wrong Direction *Right Side of the Road*: Correct Direction

Name: Datta Sainath Dwarampudi (dsd298)

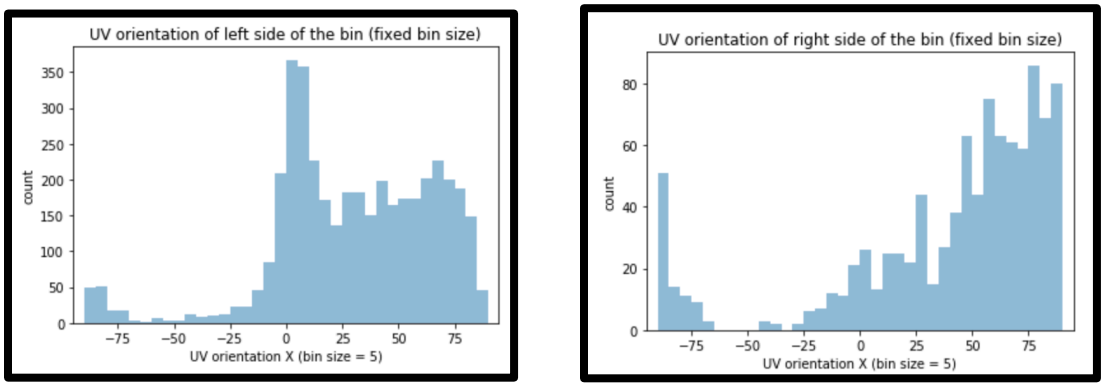
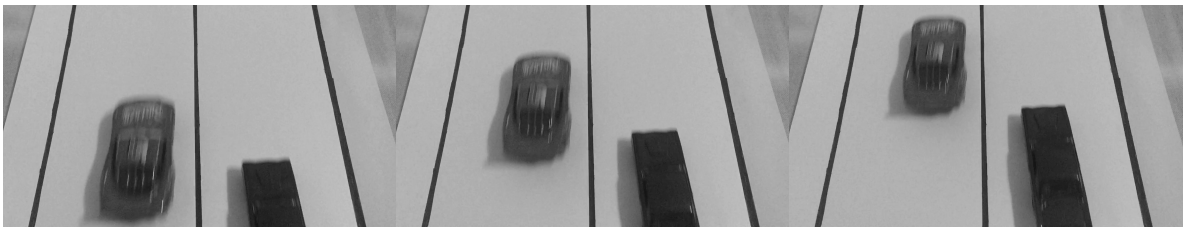


Figure 25 shows the uv directions on left and right side of the road.

```
FINAL RESULTS
The vehicle on the LEFT SIDE OF THE ROAD IS IN THE WRONG DIRECTION
The vehicle on the RIGHT SIDE OF THE ROAD IS IN THE CORRECT DIRECTION
```

Figure 26 shows the output from the terminal

Scenario 3:



Frame 87 Frame 92 Frame 97

Figure 27 shows few frames of a video to show the motion of vehicles in the video

Expected Results:

Left Side of the Road: Correct Direction *Right Side of the Road*: Wrong Direction

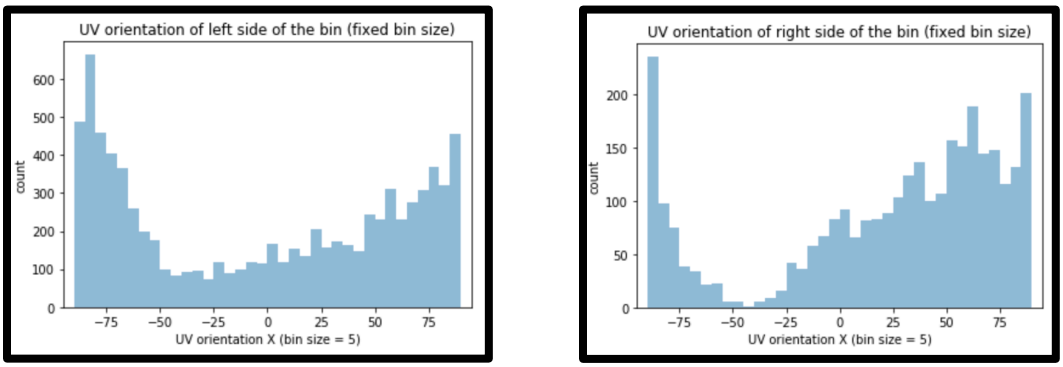


Figure 28 shows the uv directions on left and right side of the road.

```
FINAL RESULTS
The vehicle on the LEFT SIDE OF THE ROAD IS IN THE CORRECT DIRECTION
The vehicle on the RIGHT SIDE OF THE ROAD IS IN THE WRONG DIRECTION
```

Figure 29 shows the output from the terminal

Case where this algorithm fails and its solution:

Frame 62 Frame 67 Frame 72

Figure 30 shows few frames of a video to show the motion of vehicles in the video

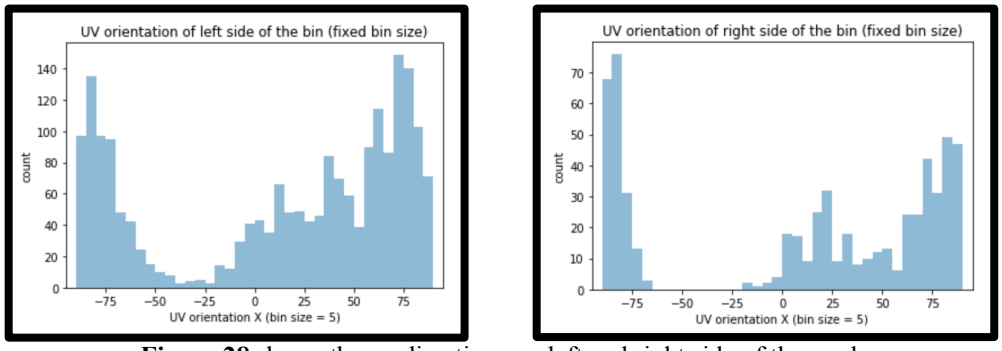


Figure 28 shows the uv directions on left and right side of the road.

Explanation about Right side of the road:
We could find that degrees were plotted on this side of the road because there is *shadow movement of the experimenter* when we were doing this experiment.

Explanation about Left side of the road:
We could find that degrees were plotted on positive side and negative side of the graph because there was *movement of vehicles in both the direction*. We got the result as wrong direction because the surface area of the red car is more, so more plots were plotted with respect to that objects optical flow.

Solution: *Reduce the scope of the camera* so that it can capture only a single car body and not multiple objects

Few other scenarios where this model can fail:

1. If there is lot of camera movement.
2. We should place our camera in the middle or somewhere near the middle otherwise we will not be able to find the divider line using this model.
3. Care has to be taken that the u and v displacement between frames is very small

Data Used:

I have used all the data created by toy cars on a sheet of paper (I have discussed it with the professor before employing this approach)

Future Work and Conclusion:

Successfully completed the first block of the complete solution model which can be used to detect wrong direction driving vehicles on both the sides of the road. This project can be extended to build the second block of the solution model to fully utilize this project work. We can avoid few scenarios where this project model fails by reducing the scope of the camera as discussed in the previous section. We can extend the computer vision block to do a real time monitoring rather than recording the video and then analyzing the video.

References:

[1] Guido Gerig (New York University), Class 10-Optical Flow lecture slides of Spring 2018 course material.

[2] Prof. Mubarak Shah (University of Central Florida), Computing Optical Flow Lecture-6 of Fall 2012 course material.

[3] Milan Sonka, Vaclav Hlavac and Roger Boyle, Image Processing, Analysis and Machine Vision, 4th Edition, Cengage Learning, Stamford, CT, 2015.

[4] A third of road accidents caused by vehicles plying on wrong side, Times of India, July 2017, https://timesofindia.indiatimes.com/city/delhi/a-third-of-road-accidents-caused-by-vehicles-plying-on-wrong-side/articleshow/59623945.cms