ENPM 661: Perception Robotics

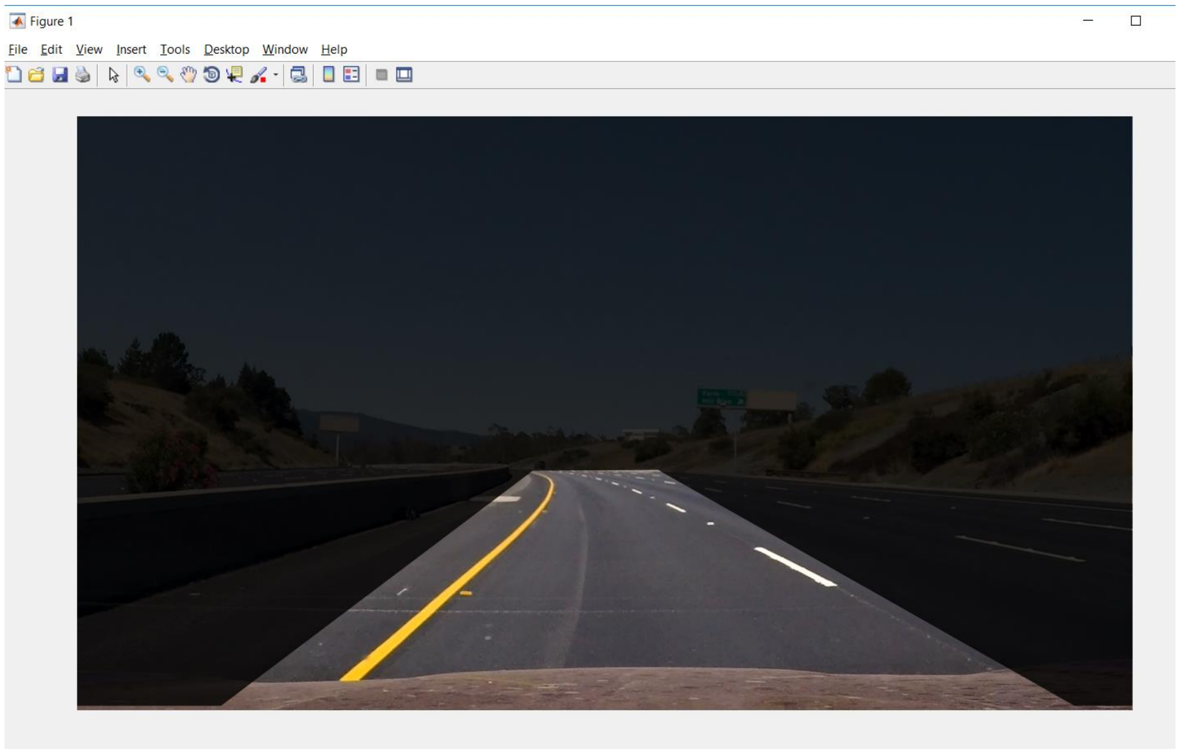
Project 1

Anirudh Topiwala (UID: 115192386)

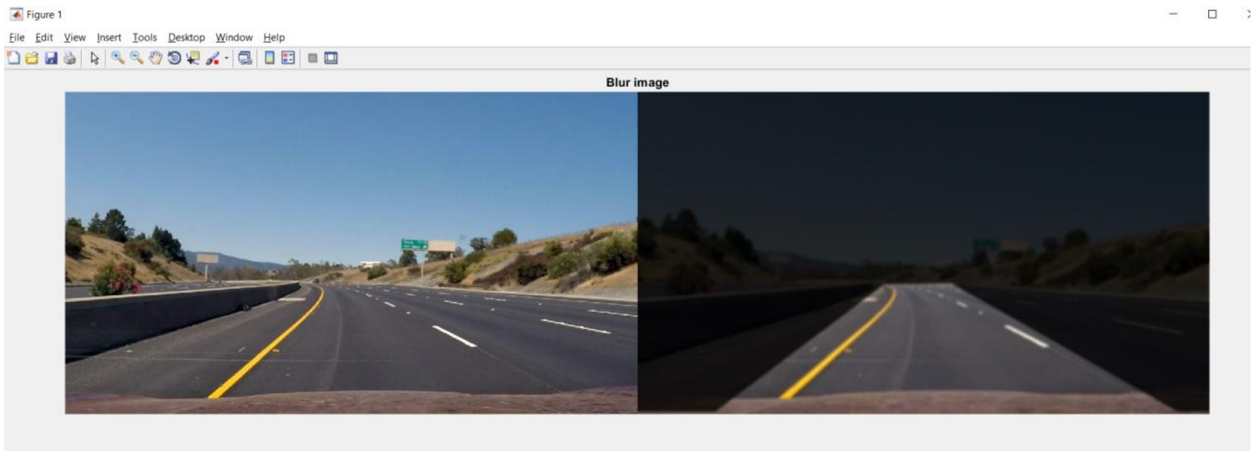
1) Lane Detection:

The first part of the project is detecting lanes in the project video. To do this we follow the given pipeline.

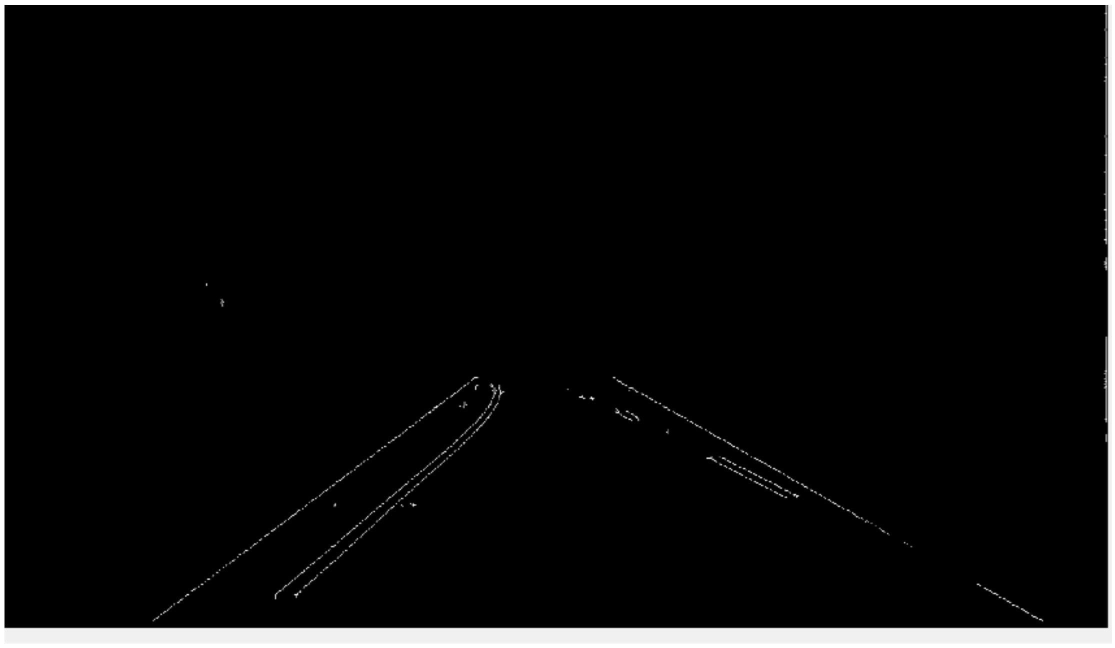
- We first read the frame from video. This is done by using readFame() and a while loop is used to continuously extract frames till a frame exist.
- Implementing ROI (Region of Interest)
  - This is done using inserting a shape which covers the entire image except the lanes. The estimated coordinates are defined and a polygon is created.
  - The shape is inserted using insertshape () and the opacity is set at 0.8. This is done because if the opacity is 1 then the edges of the polygon are also read using edge detection, which is not wanted.
  - The output is as follows:



- Blurring the Image
  - This is done in order to reduce the noise in the image. I have used a gaussian filter to blur the image.
  - The output is as follows:

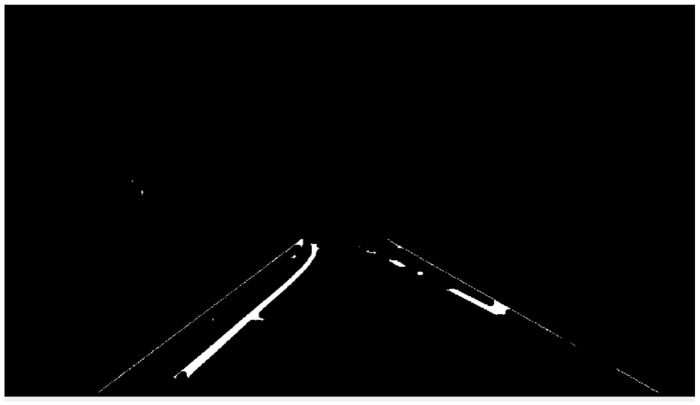


- Edge Detection:
  - This is the most crucial part of the project. Here I have used a Sobel method to carry out edge detection.
  - The thresholding value is found out by trial and error and the function is set to recognize only vertical edges.
  - The output is as follows:

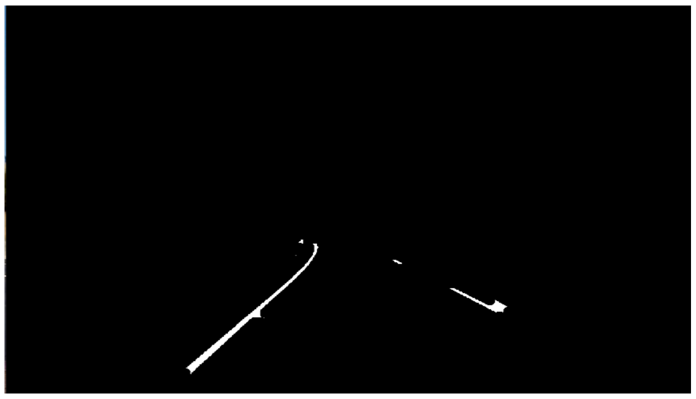


  - Here we see that even the edges of the black polygon shape which we inserted earlier is being detected, but that we will ignore using imclose and imerode as can be seen later.

- Joining the Edges
  - This is done using imclose and imerode. The strel element taken is disc. Imerode() will remove the unwanted noise in the image.
  - Output is as follows:

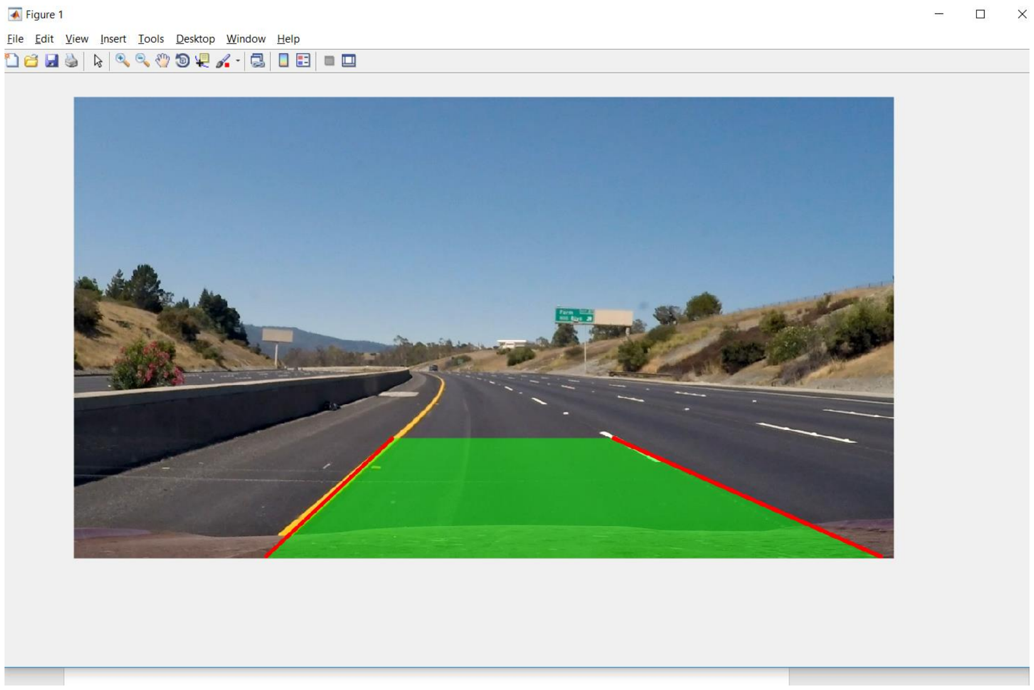


Imclose()



imerode()

- Detecting lines and Inserting green shape to show lane detected.
  - This is done by using houghpeaks and houghlines.
  - Once we get the lines, the line of greatest length is extracted. This is done just by comparing the length of lines.
  - Once we have the greatest length line, I extract the points of that line.
    - Once I have the points I calculate the slope and plot the line using any of those points.
    - Now I calculate the intersection of this line with the end of the image axis, that is with line y=rows; This will give me the x and y coordinate of the starting point of both the left and the right line.
    - Now I simply calculate the points which are at a x= x+200 distance. This distance is arbitrary and can be changed depending upon the user. This will just determine how long the green trapezoid should be created.
    - The above step is repeated on both left and right lines.
    - Once I have all the four points I insert the trapezoid figure using inertshape().
    - The output is as follows:



2. Reduction of Errors:
   - To reduce errors, I have also added try and catch block.
   - Here if the difference between the newly added points and the old points is more then a set threshold then the new points detected are discarded and the calculations are done with the previous set of points or lines.
   - This removes unwanted variation in both the left and the right turn.
   - The threshold here is set at 150. This value can be changed by the user to get better optimized output.

3. Predicting Turns
   - This done using both the vanishing point method and the slope method. But I got better results with vanishing point method.

   3.1 Slope method:

   - In this I have simply taken the slope of left and right lane and averaged them.
   - This averaged slope is again averaged with the last three slope inputs in the previous frames and the resultant slope is used to predict the turn. This is done cause sometimes, the lines detected can be slightly off and I don't want that to affect my turn prediction.
   - Using the output values generated for successive frames, left, straight and right turn thresholding values were set.

   3.2) Vanishing Point method:
   - Similar to previous method here I detect the vanishing point. This is done be finding the Intersection point of left and the right line.
   - Again, by using the output values generated for successive frames, left, straight and right turn thresholding values were set.
   - Here also, I average the intersection point values with the last three frames , to reduce the error in predicting turns.

   The Output is as follows:

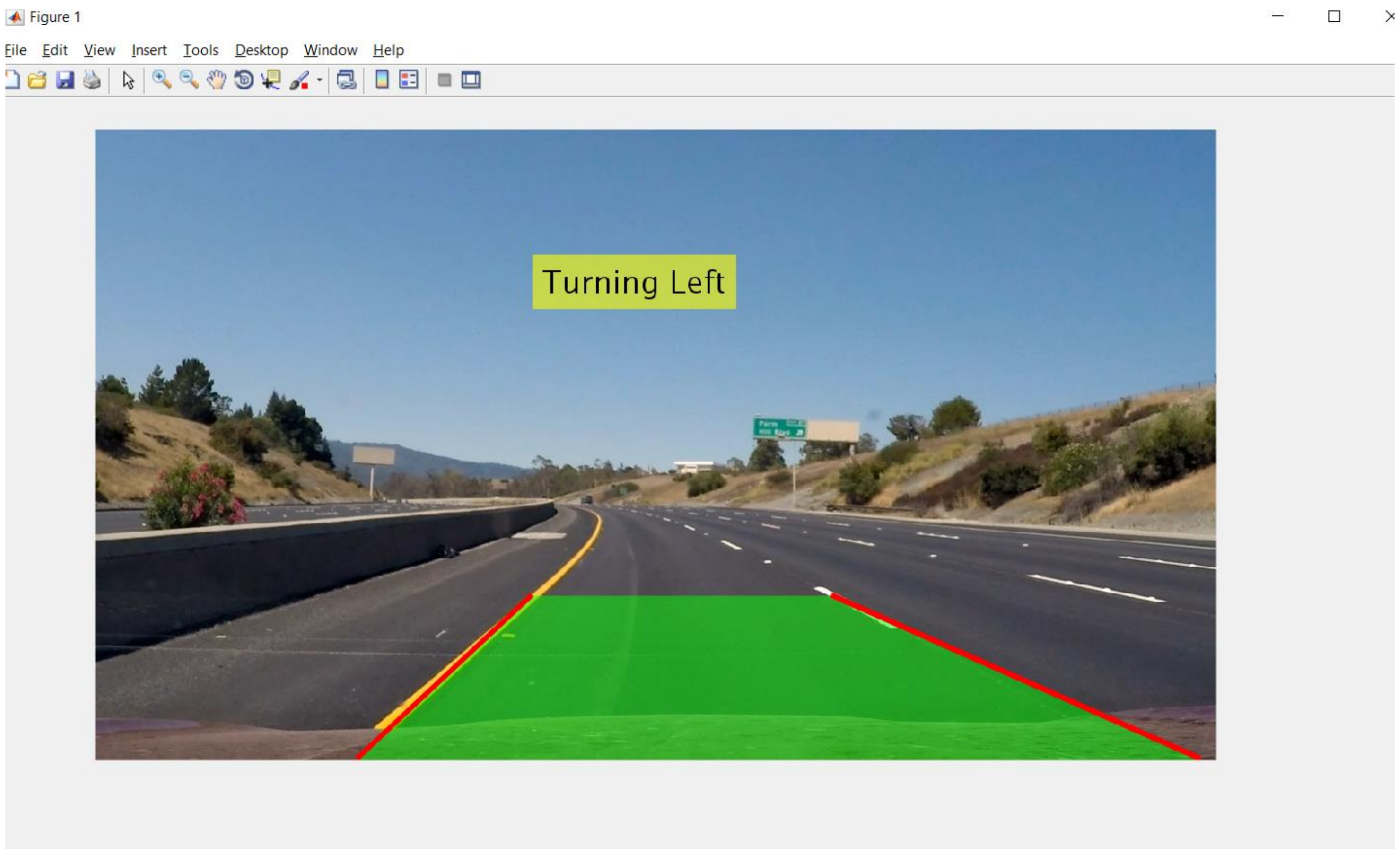


Figure 1
File
Edit
View
Insert
Tools
Desktop
Window
Help
Turning Left