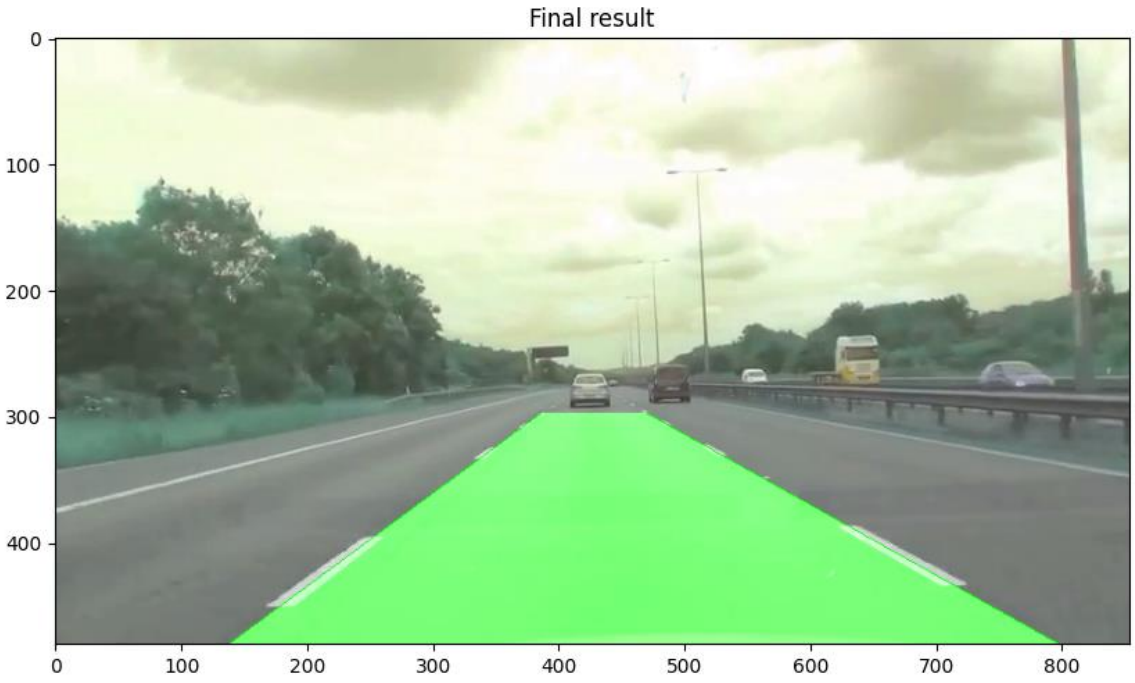
Project #1 – Lane Detection

Omri Gruman, 318965621

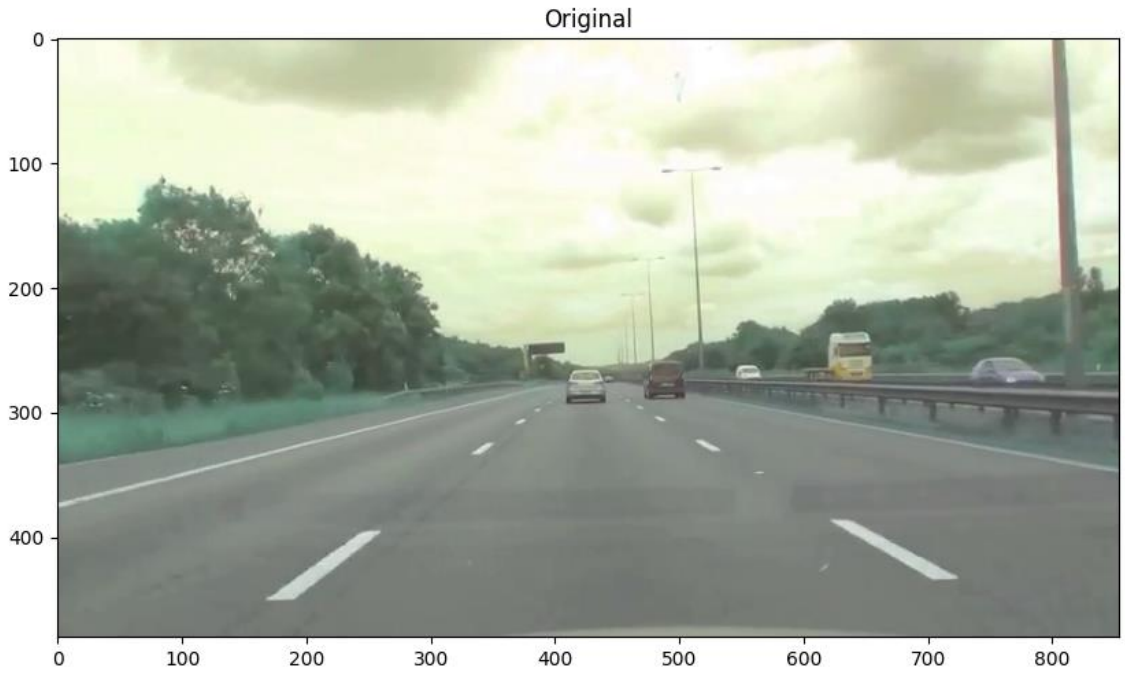


1 Overview

In this project, we are given the task of detecting lanes in a dashboard camera video. The process includes preprocessing the video in frame-by-frame manner, detecting the lane lines in the frame, as well as notifying the viewer of a lane transition when it occurs.

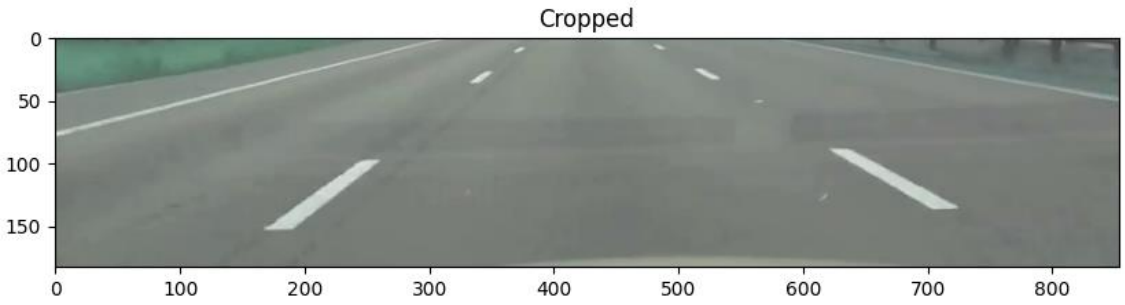
2 Preprocess Frame

In the preprocessing stage, we first attempt to focus on the lower part of the image. Then, we detect edges using Canny's algorithm and mask the given edge. At last, we run over the resulting image row-by-row to "squash" together pixels that we consider "neighbors".



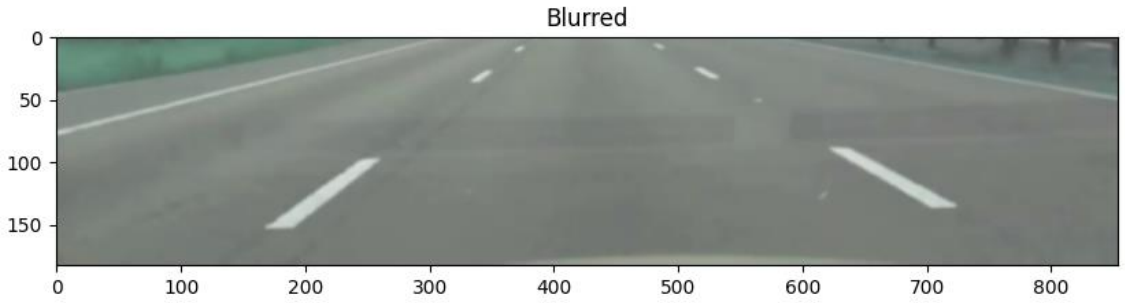
2.1 Cropping

In the first step of the preprocessing stage, we crop the image to observe only on the lower 38% of it. The crop size was chosen by trial and error while trying to capture a large part of the road as well as to avoid irrelevant parts of the image such as the sky, trees, further cars, etc.



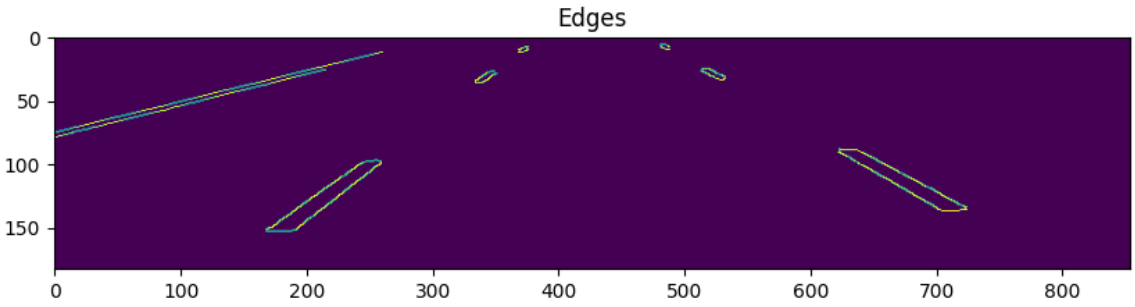
2.2 Gaussian Blurring

After cropping the image, we follow our actions by blurring the image using Gaussian filters of size 5x5 to prepare it for edge detection in the next step.



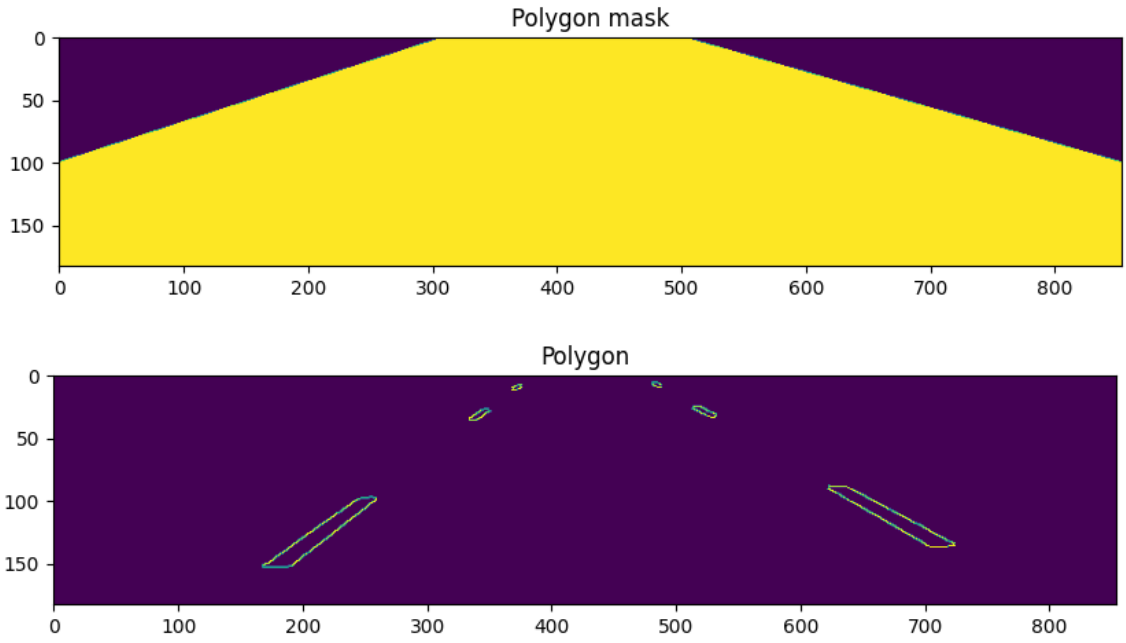
2.3 Canny edge detection

Next, we apply Canny's algorithm for edge detection on the blurred image in order to capture lane line segments that will help us detect those lane lines later in the "Lane Detection" phase.



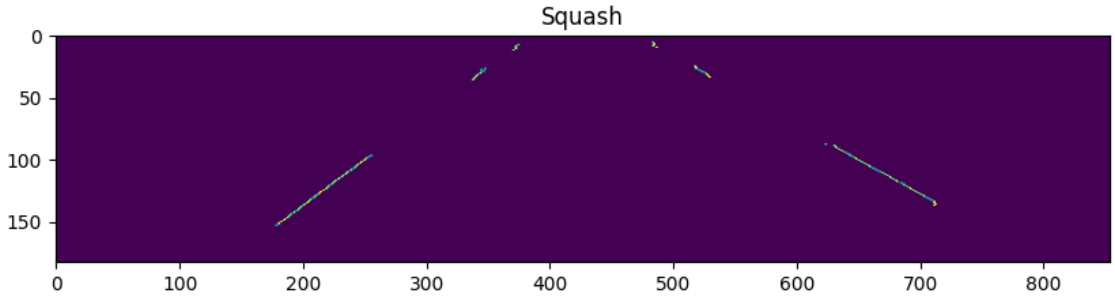
2.4 Polygon masking

Nevertheless, we do not stop with only detecting edges in the cropped image, as we mask the resulting edge image to focus on the predetermined polygon-shaped part that includes only the area of interest and will help us avoid further noise on both sides of the image.



2.5 “Squashing”

At last, we take a final shot at optimizing our preprocessing with an elegant attempt to take the lane line segments and suppress it row-by-row. That is, running over the binary edge image row-by-row and grouping together “neighboring” pixels. After the grouping is finished, we erase all the pixels in the group leaving only 2 pixels in the center of the group (e.g., if we group together pixels at indices 4, 5 and 9, the center of the group will be indices 6 and 7). This technique is done to reduce the number of pixels in the image as well as preserve the straight shape of the lane lines.



3 Lane Detection

In the lane detection stage, we use the RANSAC technique along with Least Squares algorithm to try and fit different lines to the preprocessed image and compare the selected lane lines to the lane lines of the previous frame to verify our choice.

3.1 RANSAC + Least Squares

3.1.1 Motivation

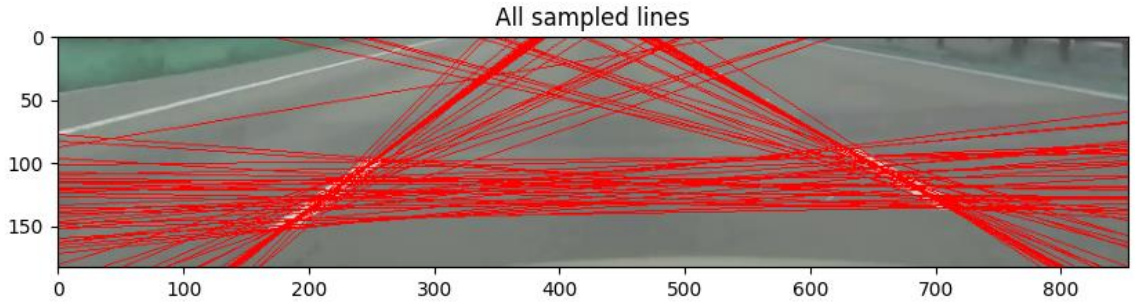
At first, an attempt was made to use Hough's algorithm to find different lines in the image. Unfortunately, the parameters of rho and theta were not intuitive enough to handle, and so the RANSAC technique was selected, primarily due to the nature of a slope and intersection point. They are easier to handle, and the line selection process was simpler.

3.1.2 "Dimensions Transposing"

After the RANSAC technique was chosen, there was still a problem regarding vertical lines. During the lane transition, we need to detect vertical lines to achieve a smooth transition. And so, we observed the image in a transposed manner. That is, we "flipped" the image 90 degrees counterclockwise (i.e., referred to the y-axis as the horizontal axis, and the x-axis as the vertical axis). This way, the slopes of the lines are distributed around 0, and the transition detection process can be done smoothly.

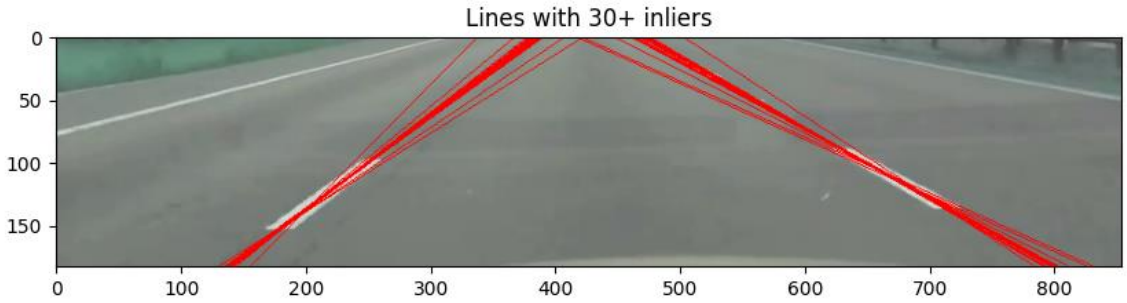
3.1.3 Random Sample

After preprocessing our image, we sample 100 random pairs of pixels from the image, each pair representing a line in the image space (We set a constant seed to our random generator to make the sampling process deterministic). Each line is evaluated by counting the number of pixels whose distance from the calculated line is not higher than 1.5 pixels (i.e., the number of inliers).



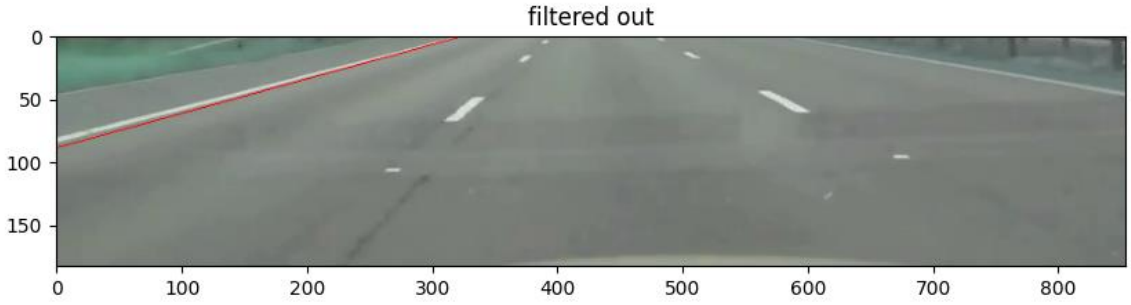
3.1.4 Optimal lines selection

After sampling and evaluating our lines, we drop lines with less than 30 inliers to maintain only lines that would optimally fit pixels of our lane lines. Then, we fit the lines to their respective inliers using the Least Squares algorithm.

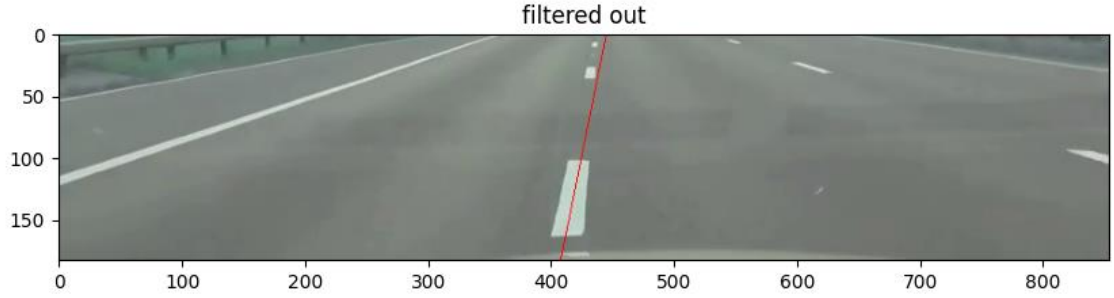


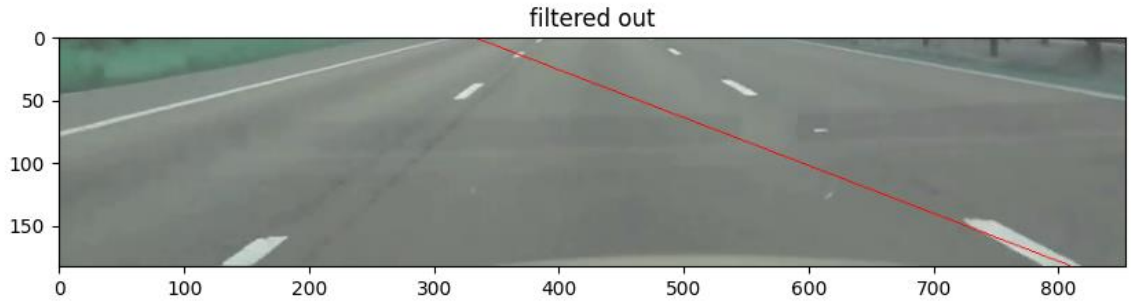
After fitting the remaining lines to our image, we decide to filter out lines that break the following conditions:

- A relevant lane line has a slope absolute value that is not higher than a certain threshold.



- A valid lane line has either: [1] a positive slope and an intersect higher than a *middle* value or [2] a negative slope and an intersect lower than the *middle* value.

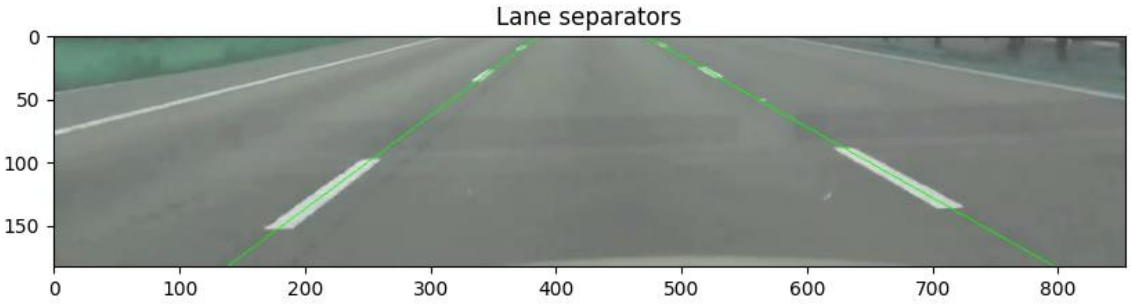


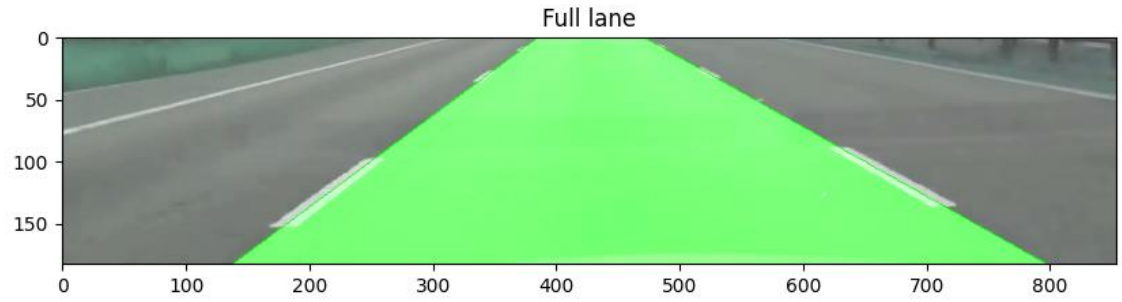


At last, we distinguish between left line and right line according to the slope of the line. Initially, the threshold slope between a left line and a right line is set to 0. During the transition process, it is set to a different value according to the state of the previous frame.

3.2 Previous frame comparison

After the lines selection was complete, we compare are new lines with the lines of the previous frame. Of course, if no new line was selected, the previous line of the respective side is selected as a default. In addition, the new line should be similar enough to the previous line regarding its slope and its intersection point.





4 Lane Transition

During the lane detection process, we must also look for certain conditions to determine if a lane transition occurs. Moreover, we need to keep track of the transition state to capture the new lane we one is detected as well as terminate the process when the transition ends.

4.1 Deviation detection

First, we look for a condition that would imply a lane transition. This condition determines that if a slope of a line gets close enough to 0, and the other slope gets far enough from 0, a transition starts.

4.2 New lane capture

Then, we must be ready to capture the new lane. The signal that we look for during the transition process is a new lane line that has a slope with significantly higher absolute value in the respective side. When such signal presents itself we adjust the other lane line as well.

4.3 Transition termination

At last, we terminate the transition process when the slopes of both lane lines reach within a normal range.

5 Assumptions

During the project we made several assumptions that would make the lane detection slightly easier.

- Most of the road is in the lower part of the video.
- Slopes of relevant lane lines are within a certain range. This is relevant to the line selection as well as the polygon masking.
- On the first frame, the lines that we select from the samples are a good enough basis to lean on for following detections, if needed.
- In order to initiate the transition process, we can detect deviation based on the current frame alone, as its properties are sufficient to determine a deviation.
- The video does not contain any shade that could have interfered with the algorithm, and the road is in good shape.
- The road itself does not curve in a way that would interfere with the algorithm.

6 Conclusion

In this project, we tried to solve the problem of detecting lanes in a dashboard camera video. During the process we preprocessed the video in frame-by-frame manner, used RANSAC technique to select lane lines in the frame, and even analyzed our lines to determine if a lane transition occurs.