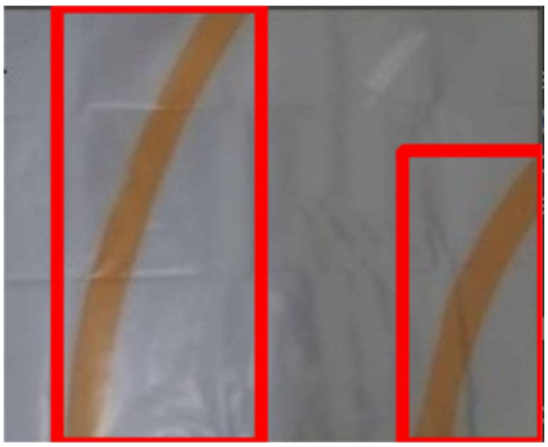
TASK 1

In our code, we have given a sense of direction by using the moments function to find the centroids of both the paths and then find the center of the whole lane using those two centroids. Comparing this with the center of the whole frame, the code gives an output of whether the robot should move left, right, or straight.

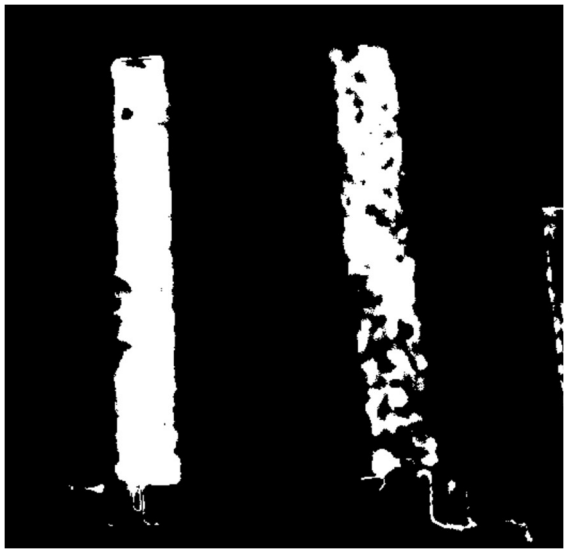
We tried approaching the task in a different way initially, trying to get the code to make the decision based upon the areas of the rectangles that bounded both the paths. Our deduction was that the robot should turn to the opposite direction of the path that has a bigger bounding rectangle.



But this wasn't a very convincing way to proceed since we assumed it could be prone to failing many test cases. So we decided to take on with the approach of finding centroids from contours.

The lower and upper limits used for masking were found using trackbars and for testing the webcam code, we used a blue pen to represent the path.
In some test cases, the code didn't work perfectly as expected, but from how much we could understand, it seemed like even if there was a left turn ahead but the robot was towards the left line of the lane, the centroid method first printed "Go right" since the center of the frame was still towards the right. We concluded that the code must be working as desired and the bot would remain within the lane, although it would be more efficient to get the code to make more approximate decisions.

In the videoclip that we've put while we were testing the webcam code, its fairly visible that the directions are not stable, but this is only due to excessive noise which we tried to reduce to a great extent. This noise was not observed in the video file codes since the path video was taken much more clearly. We have increased the number of iterations for erosion to 3 and for dilation to 5 from the default values, and this helped in decreasing noise to a great extent.



Initially, we didn't try applying the cropping function to our code since we didn't see the necessity. But after we received the video of the path taken from the robot's camera, we realized that to get better accuracy, we need to use only the lower half or two-thirds of the video and we cropped the frames we were using. It did help to a large extent, but the video was harder to process and required much more noise reduction and we could only do it to a certain extent due to time constraints.

Our code only gives an output mentioning whether the bot should go right, or left, or straight. But from how much we could understand, we concluded that when used practically, we can make a line joining the center of the frame to the center of the lane, and then see the angle it makes with the vertical, and we can make the motors turn accordingly, thus making the movement precise.