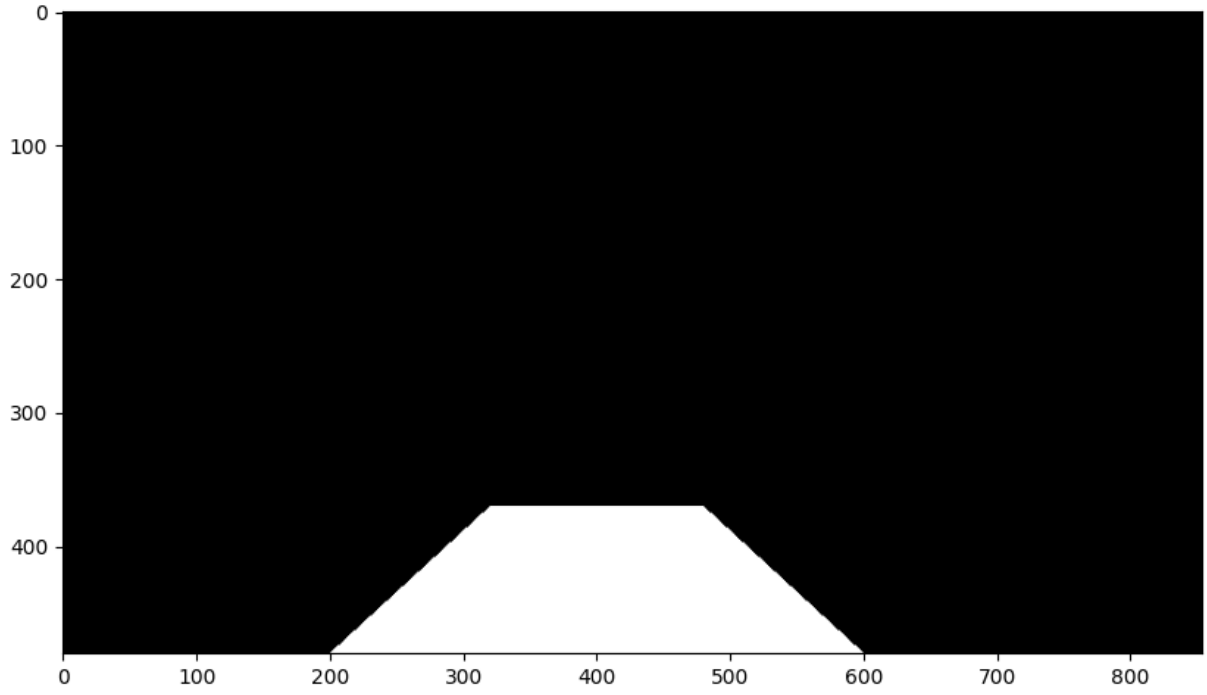
Lane Detection

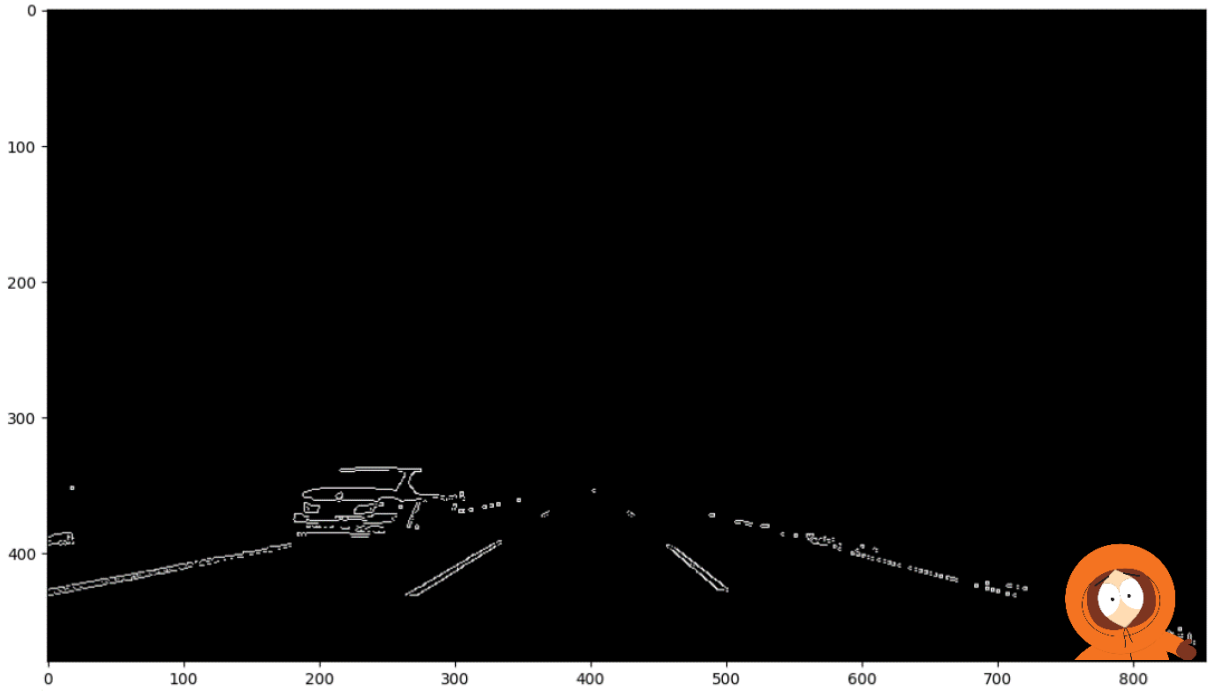
By:
Idan Oren 209007921
Sagi Goldfarb 316496173

First we created a mask in shape of the source image for the image that will only reveal the area we search for lanes



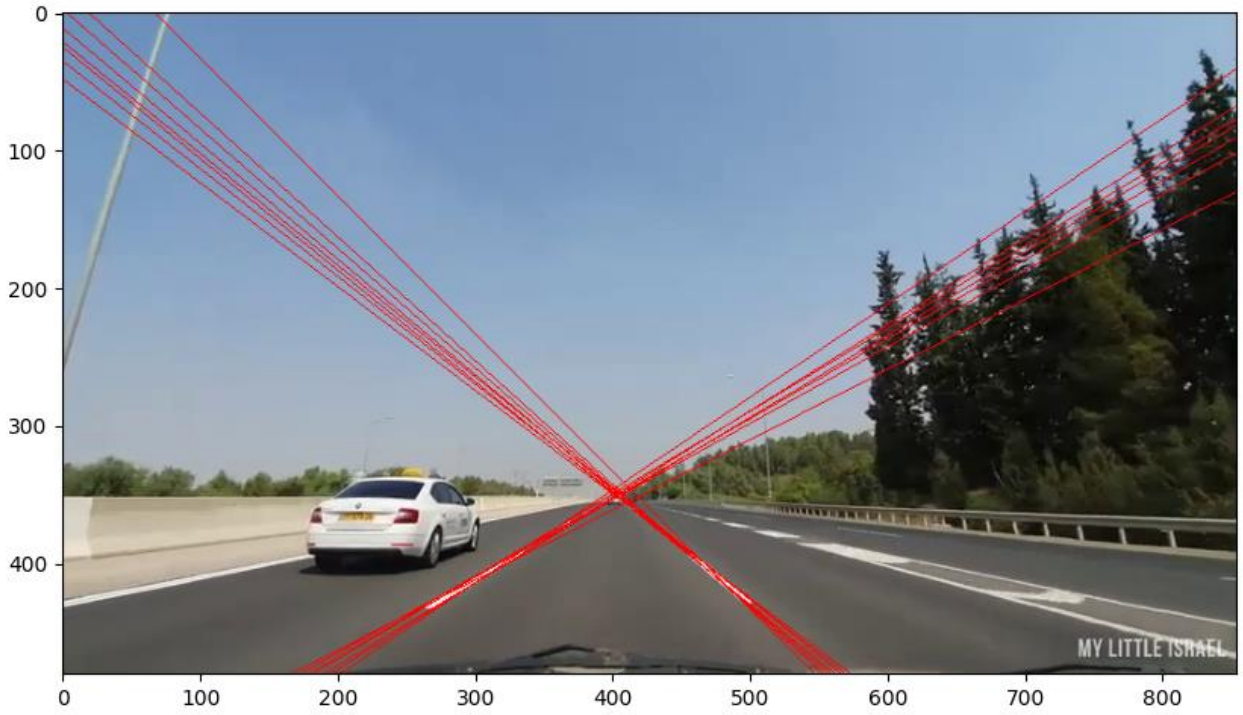
The mask is in shape of trapeziod in order to eliminate distractions.

For detecting the edges, we apply Canny Algorithm over the source image:

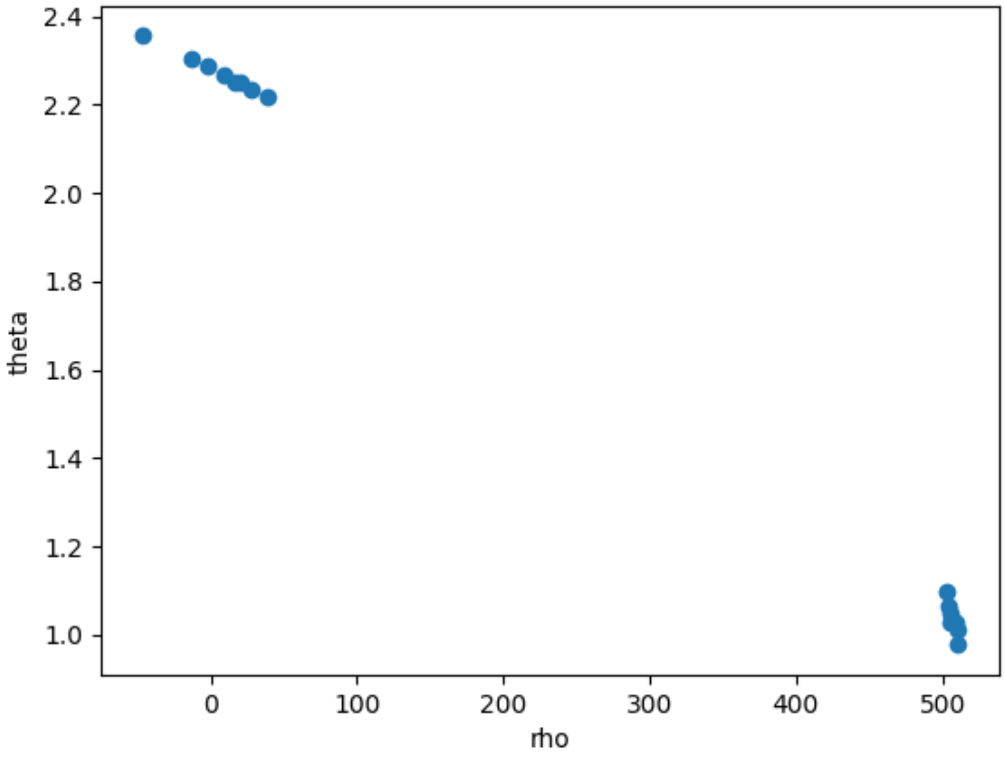


After masking the Canny'ed image (named mag_im in the code) we activated Hough lines algorithm.

Usually, the Hough algorithm will return us multiply lines (more than 2 as you can see here)



We used the next graph to analyze the data:



We noticed that the returned Hough lines parameters are grouping around two corners so we searched the average of each group:

- first we sorted the returned Hough lines by they ρ value using the next line:

```
lines_sorted = sorted(lines, key=lambda a_entry: a_entry[..., 0])
```

- then we used a tolerance function for calculating the average:
  1. insert the first line parameter to the result array
  2. while run over the sorted lines:

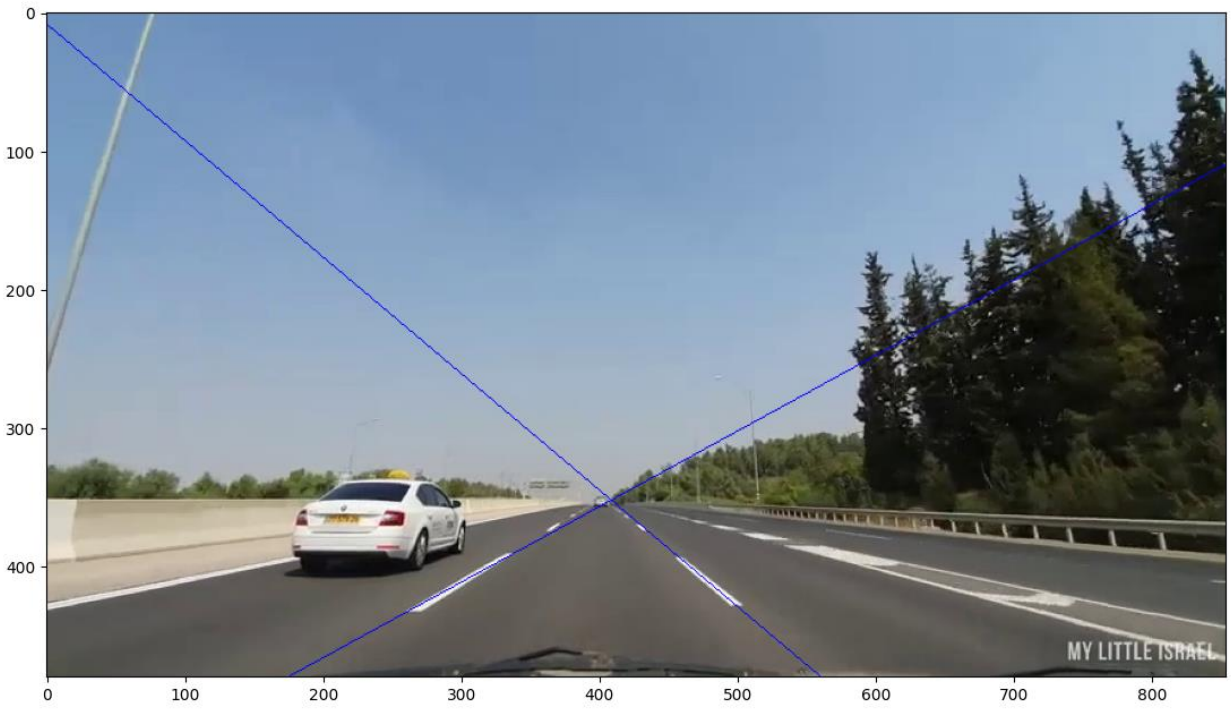
     if the next line parameter is in tolerance:

     calculate average

     if not:

     insert it in the end of the result array

by applying this algorithm we narrowed our lines from many to 2.



To beautify the marks we used a canvas that we draw the line over it, than we cropped it to the desired length and use cv2.addwheighted over the source image

In order to stop the lane detection flickering and to detect changing lane,

We saved the previous lanes (prev_left and prev_right)

If the new frame didn't detect any new lines, the previous lines will be applied.

If only one lane was detected we calculated its slop (positive => left, negative => right)

And updated the previous side.

```
tl_line = tolerance_lines(lines_sorted)
        if tl_line.shape[0] == 1:
            if slop(tl_line[0]) > 0:
                prev_left = tl_line[0]
            else:
                prev_right = tl_line[0]
            res =
markLanes(np.array([prev_left,prev_right]),res,cropped)
        else:
            prev_left = tl_line[0]
            prev_right = tl_line[1]
            res = markLanes(tl_line,res,cropped)
```

we find out that if prev_left slop is negative we change to the left lane a text was added: