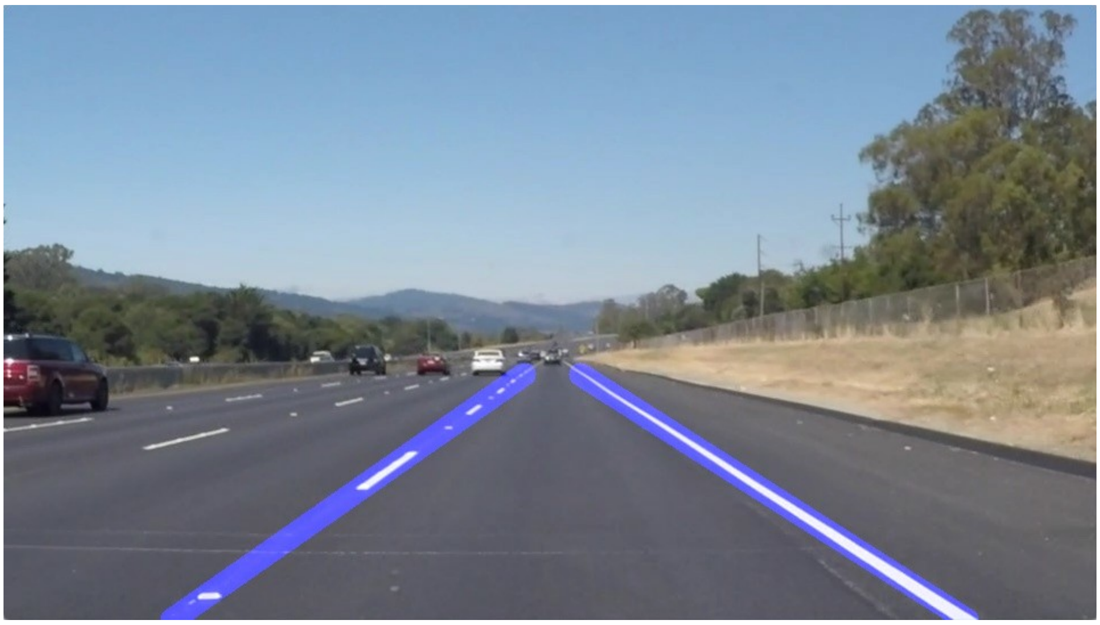
Finding Lane Lines on the Road



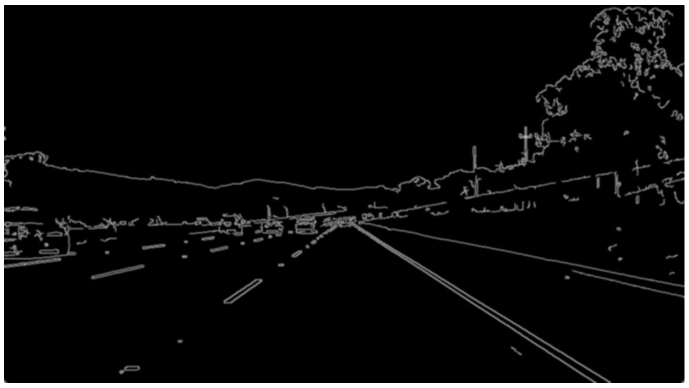
The goals of this project are the following:

- Make a pipeline that finds lane lines on the road
- Reflect on the work in a written report

Reflection

The pipeline consists on six different steps.

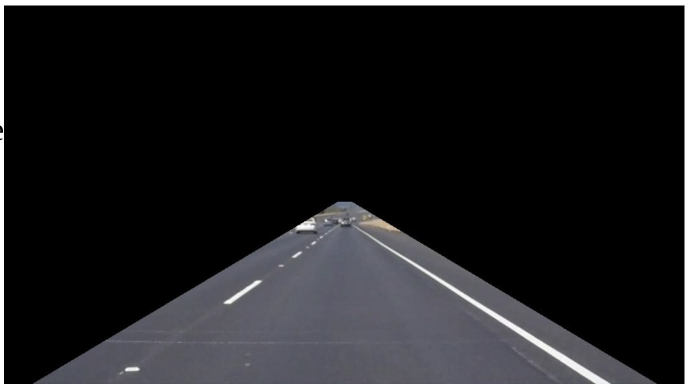
- **Grayscale:**
  Transforming the colored input image into grayscale by using the *cv2.cvtColor* method of the python library opencv.



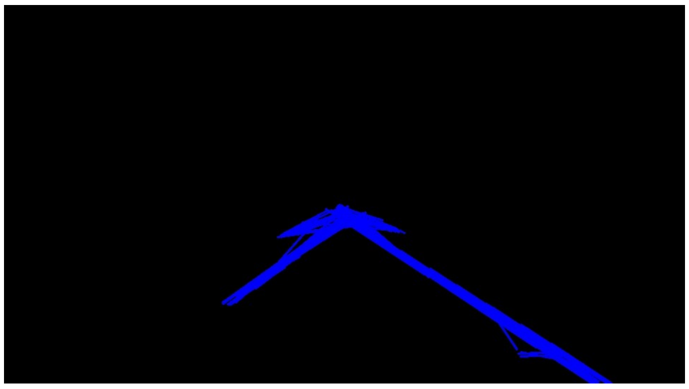
- **Canny edge detection:**
  Detecting the edges on the image by using the function *cv2.Canny* of opencv.

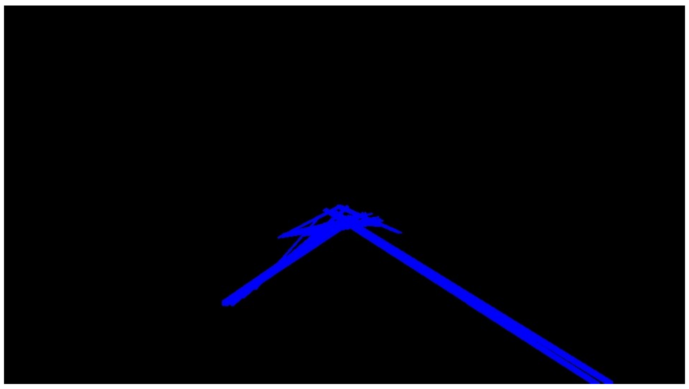
- **Gaussian blur:**
  Applying a Gaussian blur to reduce image noise. This method was also a function of the opencv library and is called *cv2.GaussianBlur.*

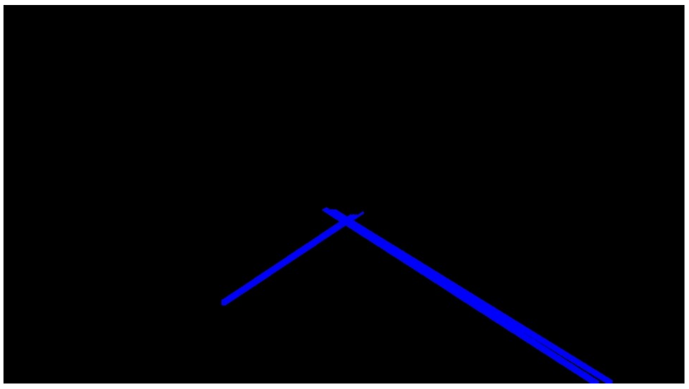
- **Masking the image:**
  The next step was to get rid of the unimportant area of the image. As the camera on the car stays steady, the region of interest stays mostly the same. Therefore a mask is applied to the image. It only keeps the region of the image defined by a polygon formed from vertices. The rest of the image is set to black. (On the sample image on the right the region of interest is still in color. This is just the case to show the masking. In the real case, the mask is applied to the blurred image.)



- **Hough transformation:**
  Following a hough transformation is applied on the masked image to find the lines using also a function of opencv, called *cv2.HoughLinesP.* Through adapting the parameters of the hough transformation the noise of (unimportant) detected lines decreases.







- **Averaging the detected lines:**
  Averaging the detected lines by grouping them based on the slope of the lines and drawing one averaged line on the right and one on the left for the mean of both groups.

After all six steps the lines were merged with the original input image by the function *cv2.addWeighted* of the library opencv.

Potential shortcomings/suggestions

- The lines shake on the video → improving the way of averaging
- In the video the lines disappear sometimes for some milliseconds → improving averaging or implementing a memory of the last x lines, so if no line on the right or left side is detected, instead of showing no line the average of the last x lines is therefor applied.
- The region of interest is steady. For straight roads optimal, but sharp curves offer issues. → dynamic polygon.
- Shadows and bright points outside the lines were taking into account
- Cars in front of the camera expose problems
- Strong gradients of the street expose problems