Project #4: Advanced Lane Detection

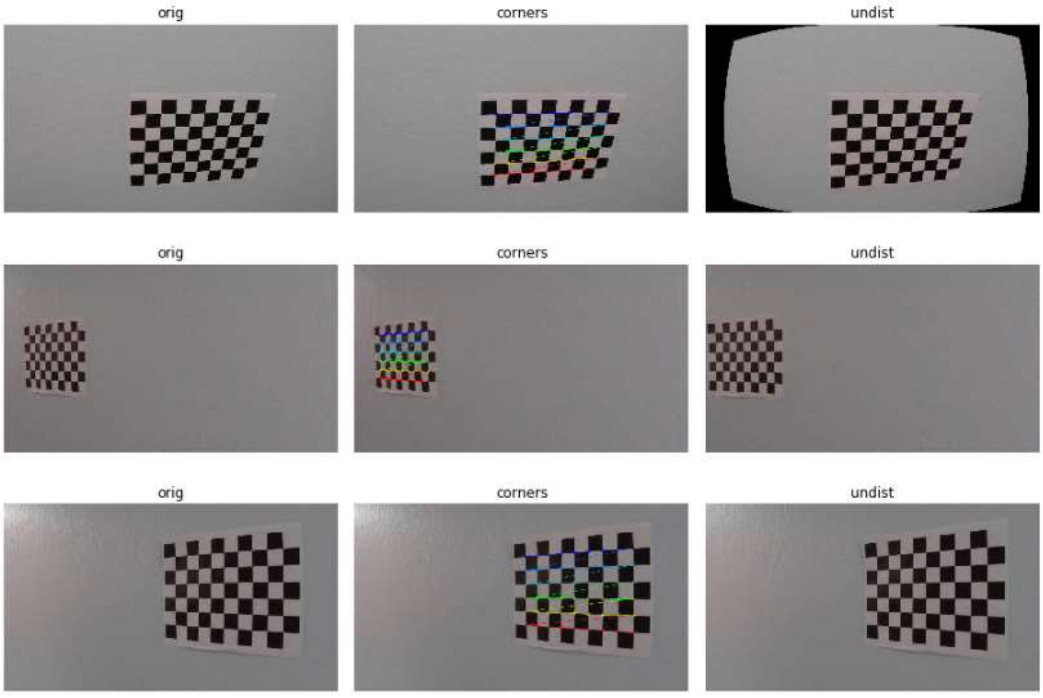
Self-Driving Car Nano Degree | Vikas Sharma

The goals / steps of this project are the following:

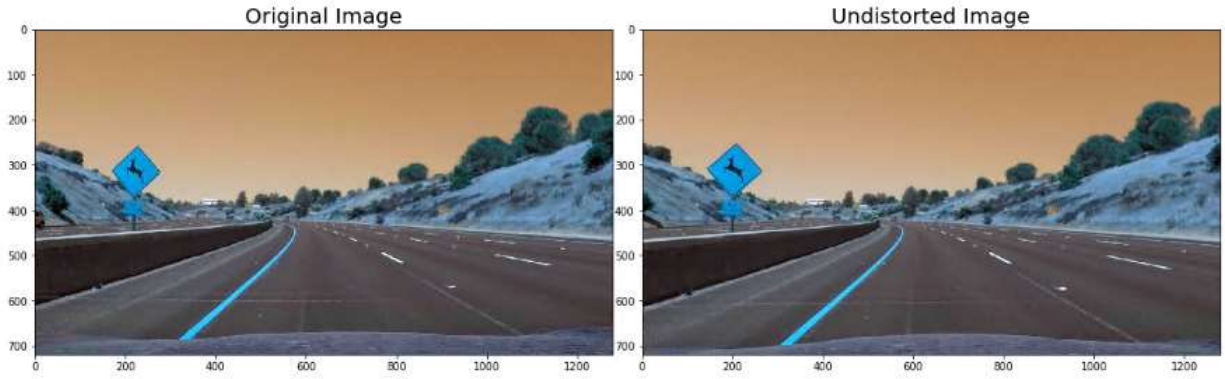
- Compute the camera calibration matrix and distortion coefficients given a set of chessboard images.
- Apply a distortion correction to raw images.
- Use color transforms, gradients, etc., to create a thresholded binary image.
- Apply a perspective transform to rectify binary image ("birds-eye view").
- Detect lane pixels and fit to find the lane boundary.
- Determine the curvature of the lane and vehicle position with respect to center.
- Warp the detected lane boundaries back onto the original image.
- Output visual display of the lane boundaries and numerical estimation of lane curvature and vehicle position.

Camera calibration

By using provided images, I started off with camera calibration. cv2.findChessboardCorners() function was able to find corners for 17/20 images. I then used cv2.calibrateCameral() function to undistort the images.

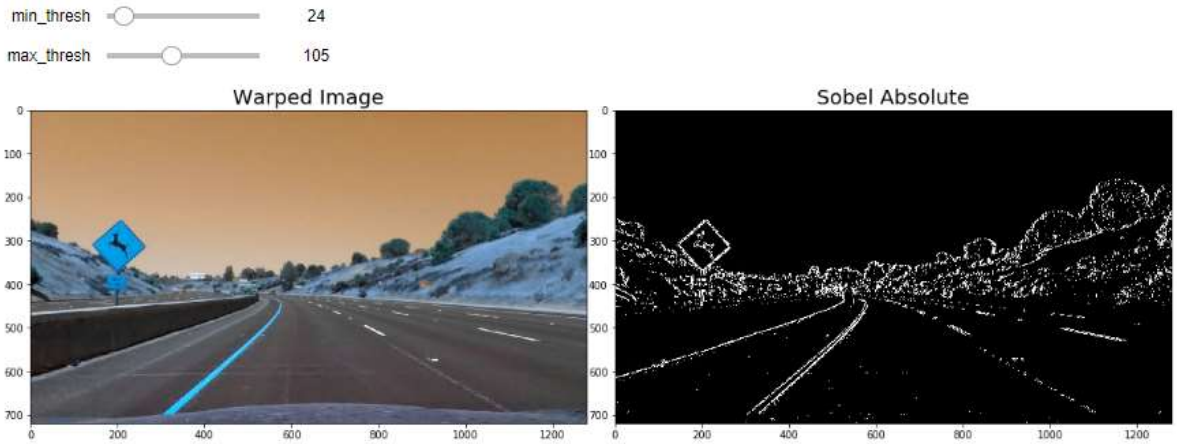


Calibration tested on actual camera image:

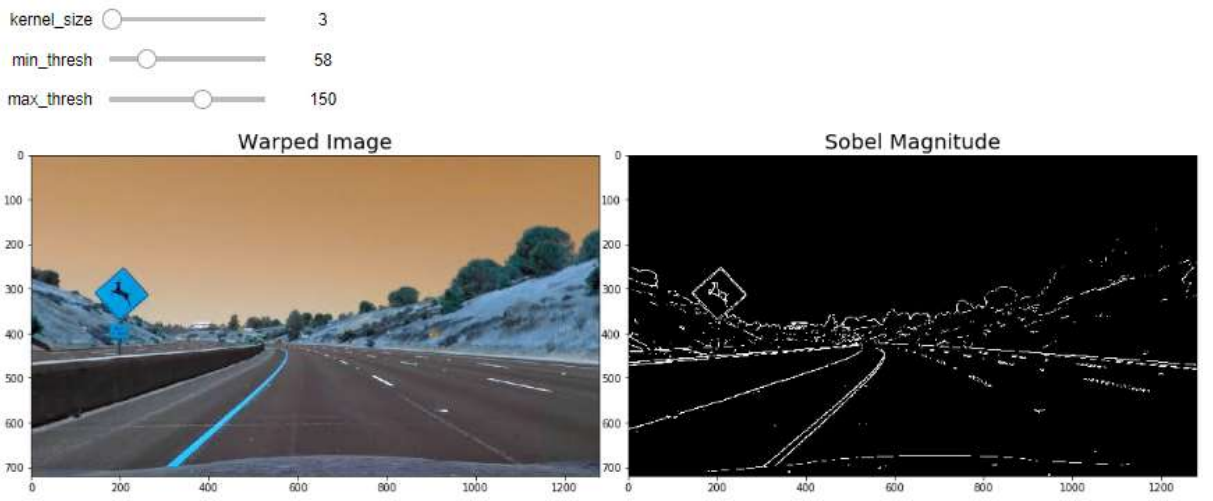


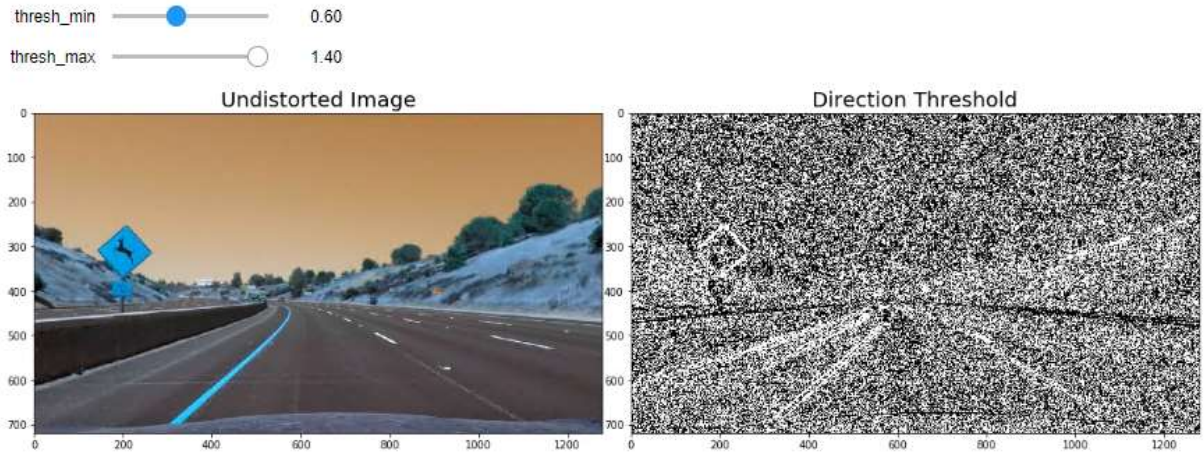
Convert to Binary Image using Sobel, Magnitude, Direction of gradient threshold

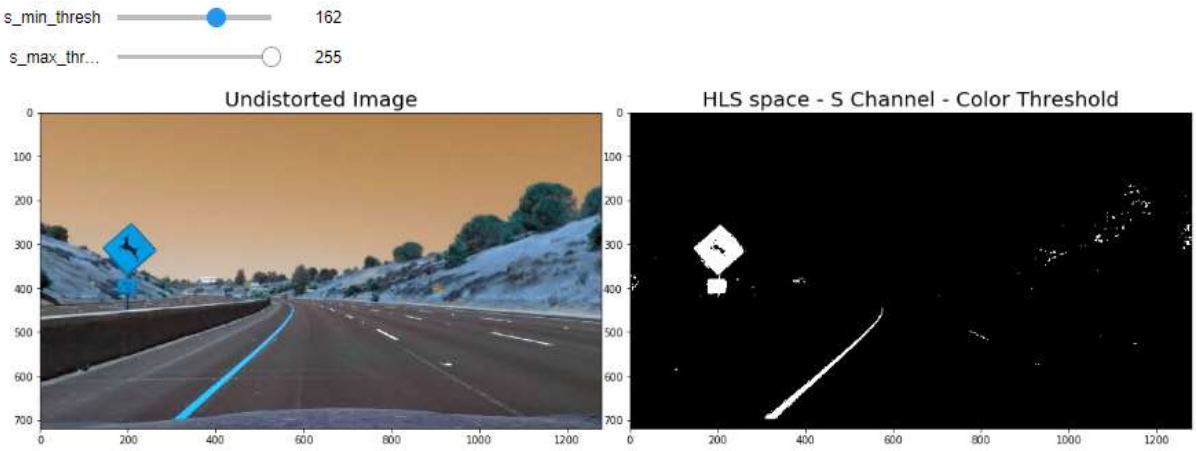
I used combination of Sobel, Magnitude, Direction of gradient threshold and color threshold to convert raw image to binary. I tried to play aroud with values a lot to have them work on all the images and found it quite tedious. I found out about interact() function of ipywidgets which makes findind the value for threshold faster, in an intreactive way.



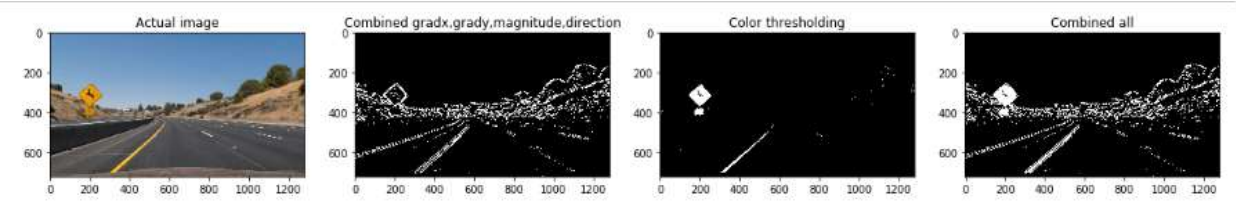
Self-Driving Car Nano Degree | Vikas Sharma



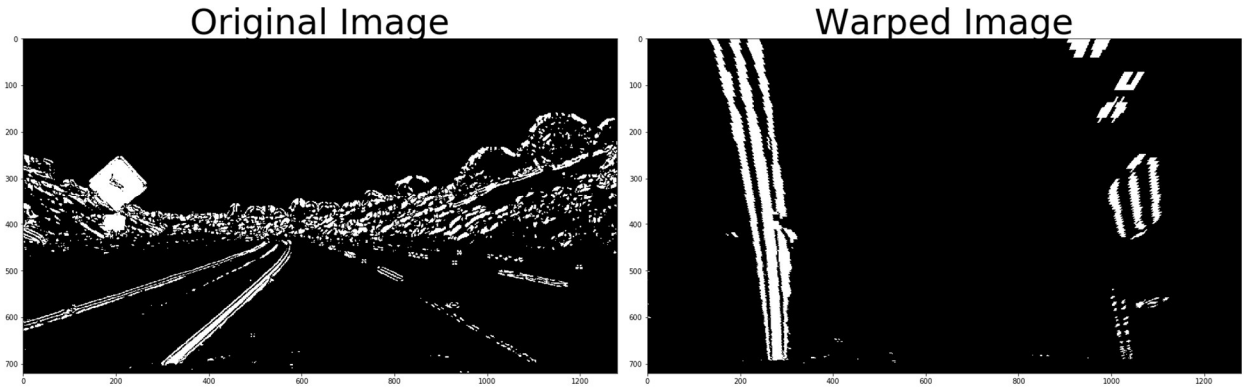




Combined



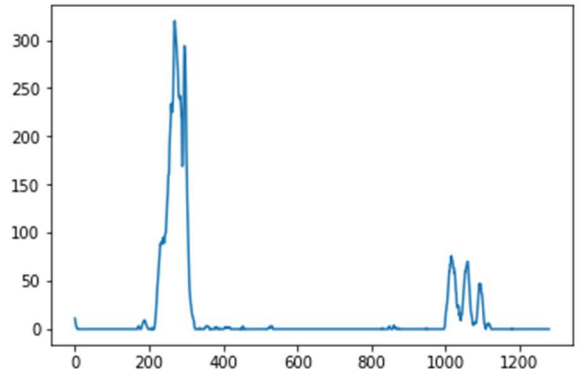
Perspective Transform of the Combined Binary Image

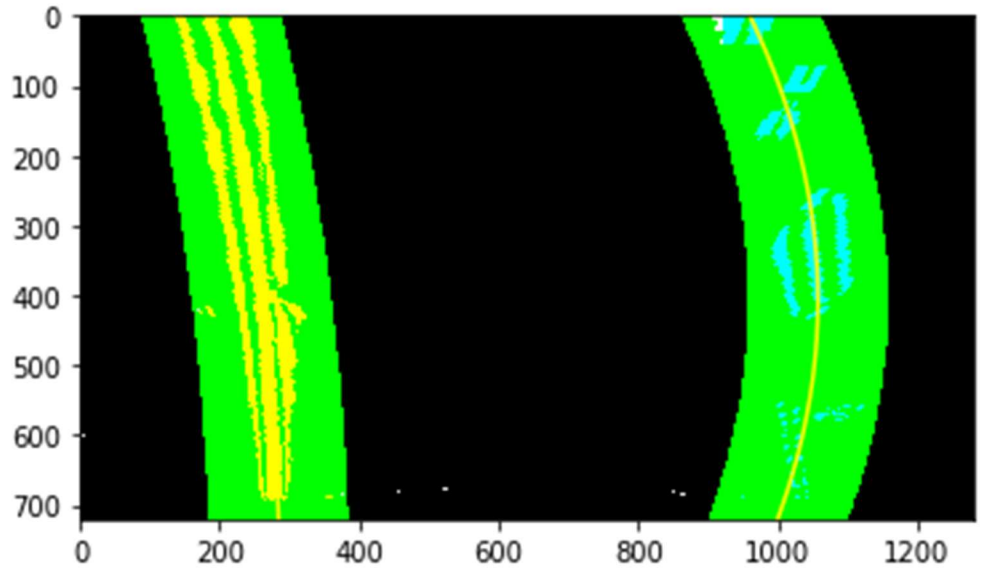


Sliding fit polynomial

I performed a sliding window search, starting with the base likely positions of the 2 lanes, calculated from the histogram. I used 9 windows.

Histogram





Finding Radius of Curvature

I used the below formula to calculate the radius of curvature and display it on the image. The sliding polynomial fit was done in pixels but we need to calculate radius of curvature in meters, we will need to do recomputing

Radius of Curvature

The radius of curvature (awesome tutorial here) at any point x of the function $x = f(y)$ is given as follows:

$$R_{curve} = \frac{[1+(\frac{dx}{dy})^2]^{3/2}}{|\frac{d^2x}{dy^2}|}$$

In the case of the second order polynomial above, the first and second derivatives are:

$$f'(y) = \frac{dx}{dy} = 2Ay + B$$

$$f''(y) = \frac{d^2x}{dy^2} = 2A$$

So, our equation for radius of curvature becomes:

$$R_{curve} = \frac{(1+(2Ay+B)^2)^{3/2}}{|2A|}$$

Pipeline to process videos

I created a lane class to define characteristics of a lane(ref).I undistorted, applied thresholds and convert to binary, applied warp and found radius of curvature and drew on images. Please see attached videos for results.

Discussion

Briefly discuss any problems / issues you faced in your implementation of this project. Where will your pipeline likely fail? What could you do to make it more robust?

- I had to do a lot of trial and error to find right values for sobel gradient and color thresholding. interact() function of ipywidgets made the job slightly easier and faster, in an intreactive way.
- My current pipeline doesn't work well on harder challenge video because it has sharp turns and lighting conditions change a lot(tree shadow, sunlight)
    - May need a better perspective transform with smaller window
    - Mix up color spaces to extract lane lines in every lighting condition