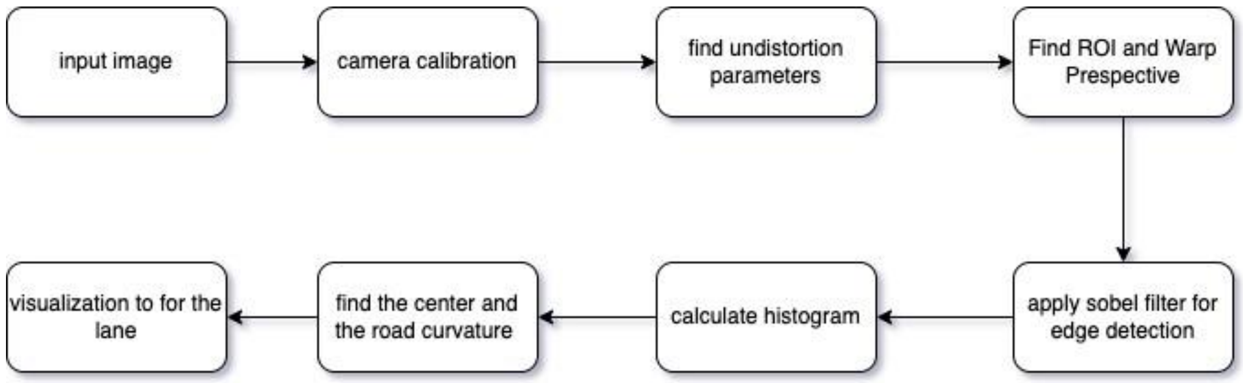
Computer vision term project –phase 1

Student Name: Ashraf Muhammed Sabry
Student ID: 12p2018

Under the supervision of:
Prof. Mahmoud Khalil
Teaching assistant. Mahmoud Ahmed Selim

Lane Detection Pipeline


input image
camera calibration
find undistortion parameters
Find ROI and Warp Prespective
apply sobel filter for edge detection
calculate histogram
find the center and the road curvature
visualization to for the lane

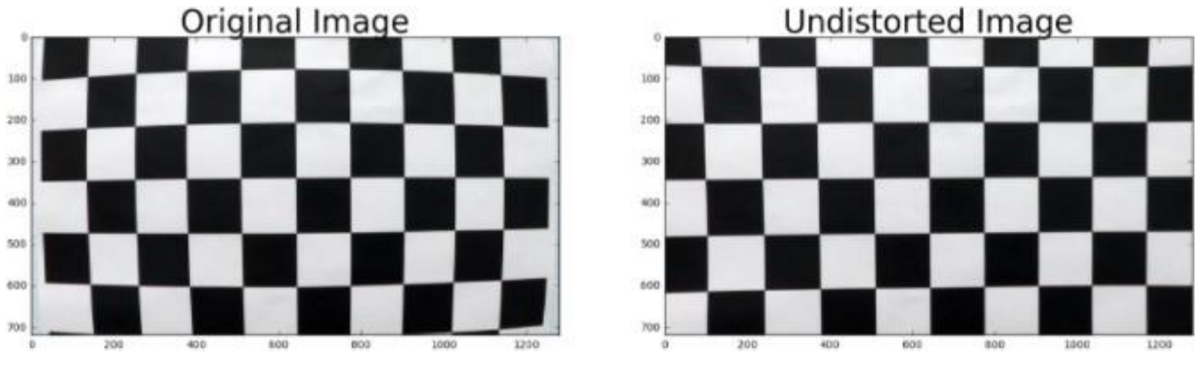
Camera Calibration

starts by Preparing Objpoints and ImagePoints. The following steps were followed-:

- Grayscale the image
- Find Chessboard Corners. It returns two values ret, corners. ret stores whether the corners were returned or not
- If the corners were found, append corners to image points.
- Also, there is a draw for the chessboard corners to visualize the corners

With this step, we will be able to get image points and object points which will be required to calculate the camera calibration and distortion coefficients.

We call the calibrated camera function which returns us a bunch of parameters, but the ones we are interested in are the camera matrix (mtx) and distortion coefficient (dist).



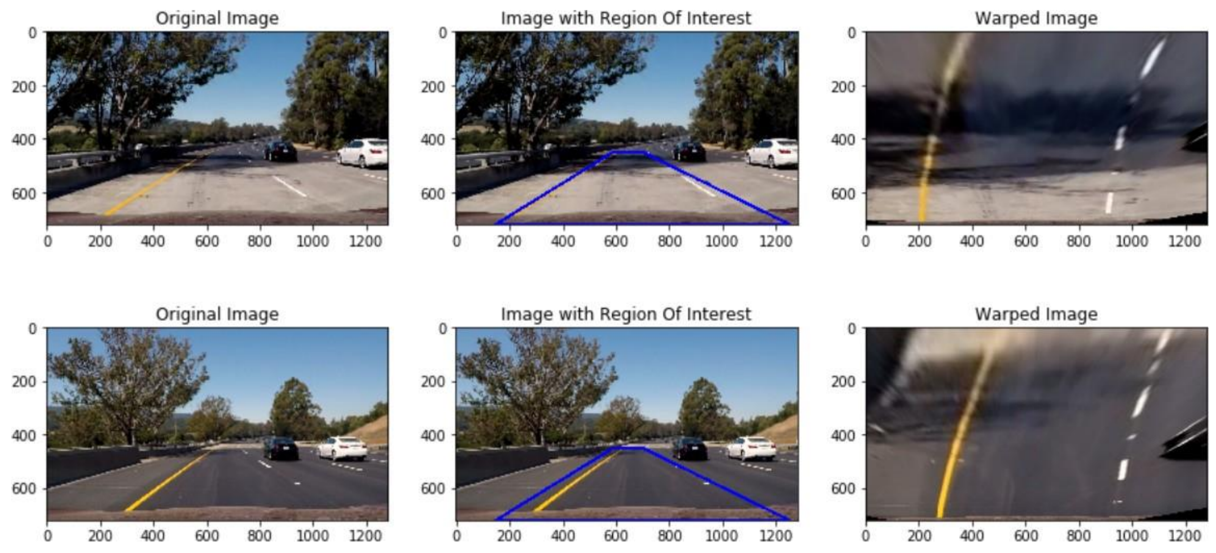
Perspective Transform with Example

In this step, I first defined a Region Of Interest (ROI) i.e. a Trapezoid with four points:

1. Left Bottom Corner is defined as "left"
2. Right Bottom Corner is defined as "right"
3. Left Upper Corner defined as "apex_left"
4. Right Upper Corner is defined as "apex_right"

After defining the ROI, the next step is to warp the image, to see the image from a bird's eye perspective. To do this we need to calculate a Matrix with the source and destination points. The destination points were selected appropriately so as to see a good bird's eye perspective. The selection of these points was based on hit a trial mechanism only.

Once we get the Matrix, we will do that along with the Image to CV2 warpPerspective function to get the final warped image.

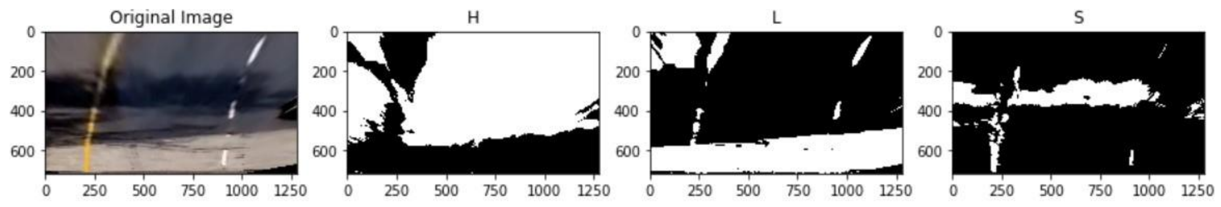


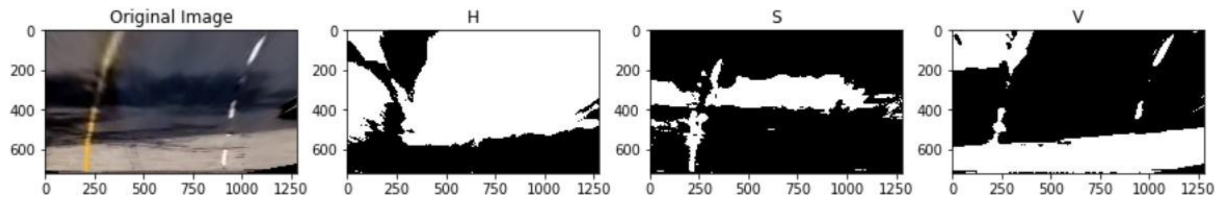
Color Transformations with example

I tried various color spaces to get a good binary image in all lighting conditions. I tried the following color Spaces-:

- HLS
- HSV
- LAB
- YUV
- YCrCb

defined a common function to extract a particular channel from a colorspace.





Sobel Filter for edge detection

I defined a common function to apply Sobel.

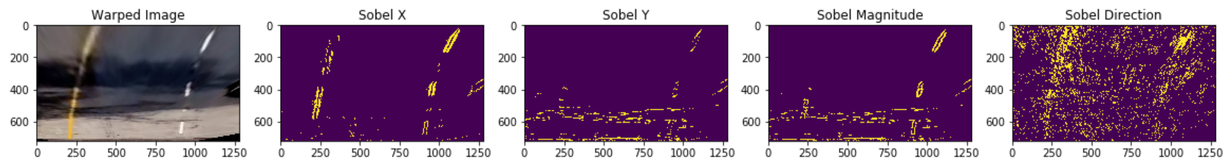
Function Name- Sobel

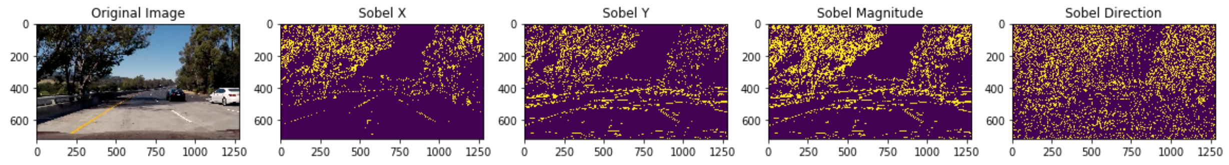
Inputs:

- **warpedimage**- the original warped image
- threshold- the threshold that is to be applied to select the pixel values
- **sobelType**- the direction where we need to take the gradient. values- x- for x gradient, y- for y gradient, XY for absolute, and dir for direction
- **kernelSize**- the size of the kernel

Output:

- Binary Image with the required thresholds, sober type and kernel size





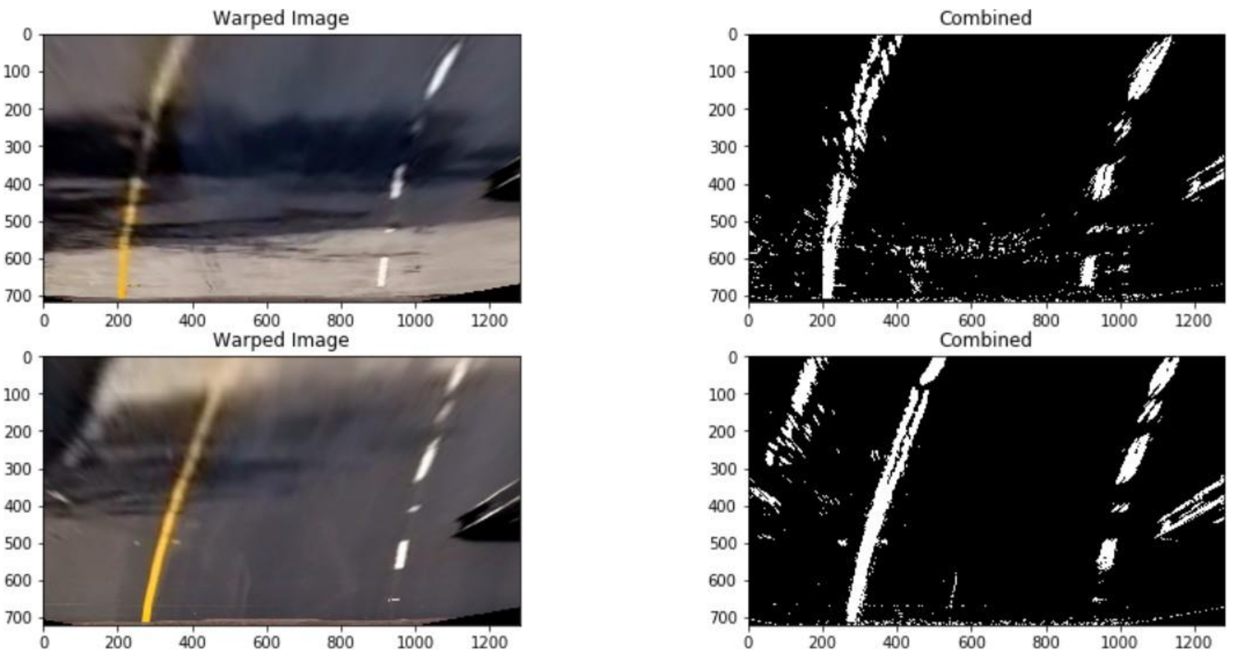
Combination of Color Transform and Gradients with example

The choice of the color spaces was random but with a purpose. I decided to use the Saturation channel of HLS because it works sort of well under all conditions. But that was not enough as it was not able to generate lines for dotted white lines. I observed that the Lightness channel HLS works well in all the conditions except the case when the image is too bright. I decided to use both Saturation and Lightness channels. But I was not even happy with that as some faint edges were still not detected so I decided to use another luminance channel, this time from YUV colorspace- the Y channel.

Once I was done with selecting the color space the next step was to select the Gradient I wanted to apply. As I could see clear vertical edges using the x gradient, I decided to use X gradient only.

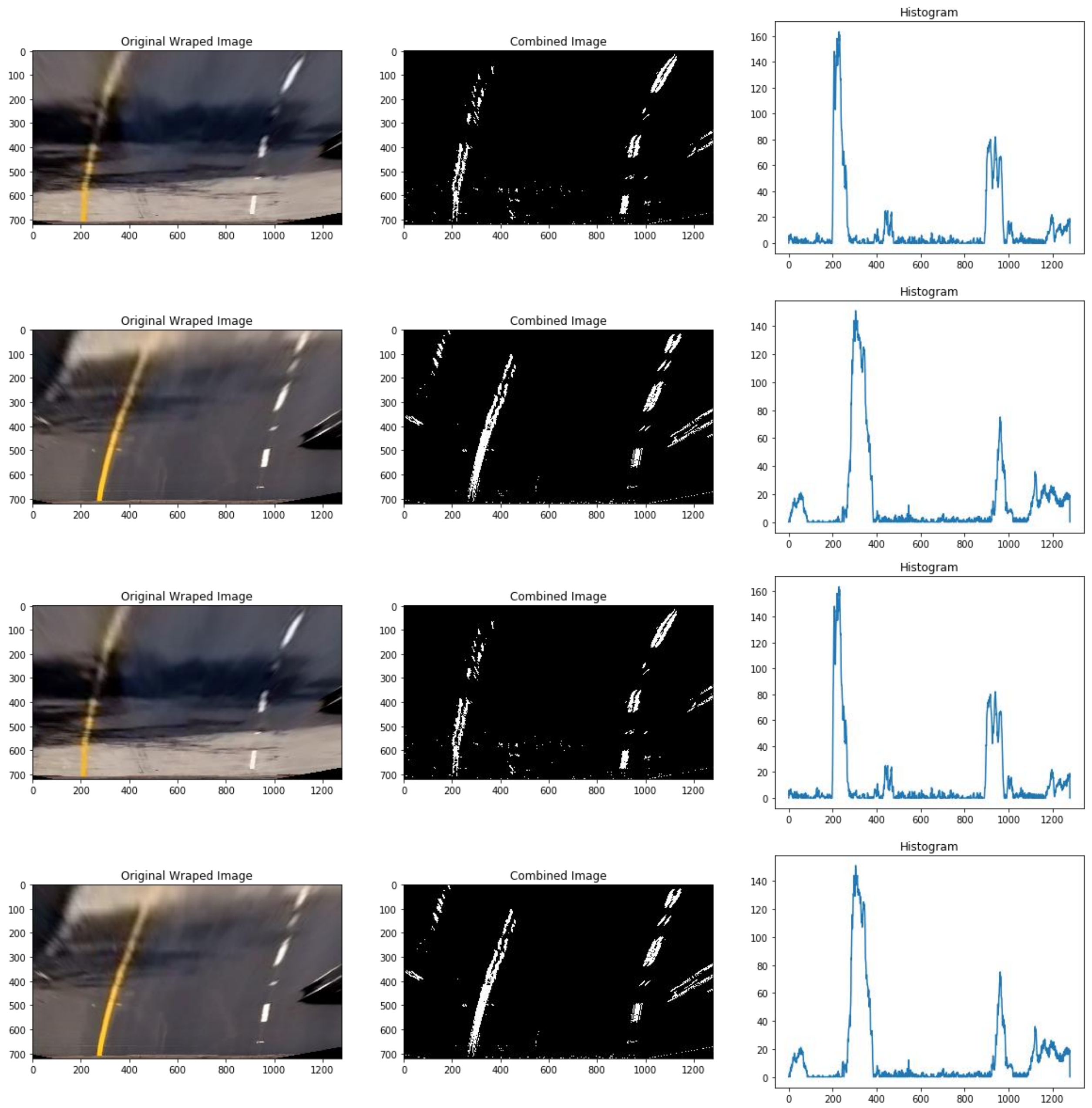
Final Combination

Mix Channel 1 = Saturation Channel and Lightness Channel from HLS
Mix Channel 2 = Mix Channel 1 and Y channel for YUV
Final Combination= Mix Channel 2 or Sobel Gradient in the X direction



Identifying lane-line pixels

The first step is to create a Histogram of the lower half of the image. In this way, we can find out a distinction between the left lane pixels and right lane pixels.

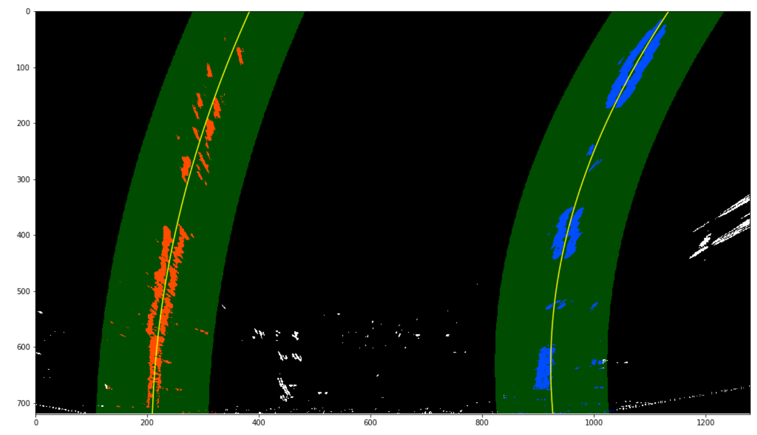


The next step is to initiate a Sliding Window Search in the left and right parts which we got from the histogram.

The sliding window is applied in the following steps:

The left and right base points are calculated from the histogram
We then calculate the position of all non-zero x and non-zero y pixels.
We then Start iterating over the windows where we start from points calculated in point 1.
We then identify the non-zero pixels in the window we just defined
We then collect all the indices in the list and decide the center of the next window using these points

Once we are done, we separate the points into the left and right positions
We then fit a second-degree polynomial using np.polyfit and point calculated in step 6.



The radius of Curvature and Distance from Center Calculation

To calculate Radius-:

- First, we define values to convert pixels to meters
- Plot the left and right lines
- Calculate the curvature from left and right lanes separately
- Return mean of values calculated in step 3.

For Distance, we know that the center of the image is the center of the car. To calculate the deviation from the center, we can observe the pixel positions in the left lane and the right lane. We take the mean of the left bottom most point of the left lane and the right bottom most point of the right lane and then subtract it from the center of the car to get the deviation from the center.

Image Plotting

Once we are done with all this the next step is to unwarp the image back to the original image. To do so the following steps were followed-:

1. Recast the x and y points to give as input in cv2.fillPoly. These are the same points we got from fitting the lines.
2. Calculate the Minv which is Inverse Matrix. This is done by passing the reverse points this time to `getPerspectiveTransform` function
3. Draw the sidelines from the points selected in step 1 onto a blank warped image
4. Unwarp the image using cv2.warpPerspective.
5. Combine the original image with the image we got from step 4 to plot the lane lines.