Advanced Lane Finding Project

GOALS

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 from the vehicle camera.
- 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 using sliding window or blindsearch mechanisms.
- 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.

RUBRIC POINTS

1. Camera Calibration

Briefly state how you computed the camera matrix and distortion coefficients. Provide an example of a distortion corrected calibration image.

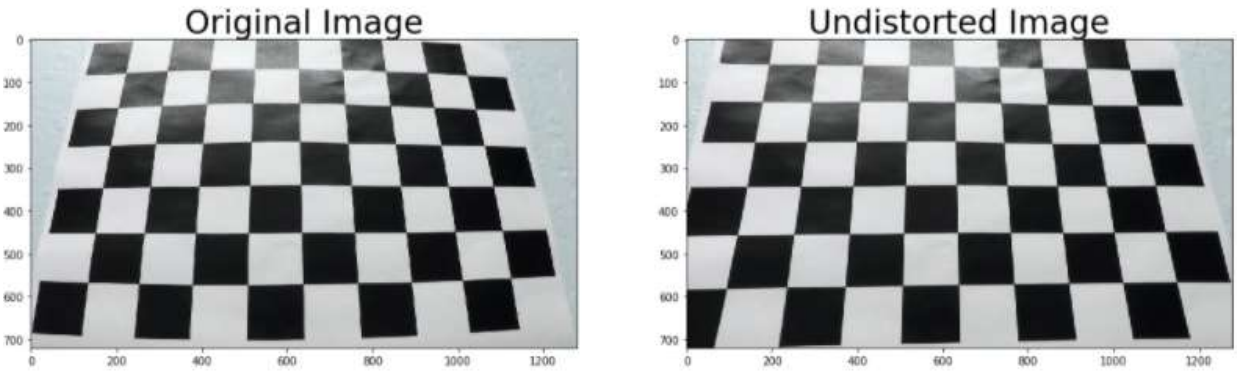
- Notebook to refer: “CarND-Advanced-Lane-Lines\Notebooks\NB1.ipynb”
- I used two arrays:

  objpoints = [] # to store 3d points in real world space, assuming the image has only x any y coordinates

  imgpoints = [] # to store 2d points in image plane.

- Steps:
  1. For each of the chessboard images, convert to grayscale
  2. Use cv2.findChessboardCorners function to find corners of the chessboard squares and store in imgpoints
  3. Map the same corners to the image representing this image without any distortion and store in objpoints
  4. Perform camera calibration using cv2.calibrateCamera
  5. Save the camera calibration coefficients “mtx” and “dist” for future use

6. It was saved under “CarND-Advanced-Lane-Lines/calibration_coefficients/wide_dist_pickle.p”

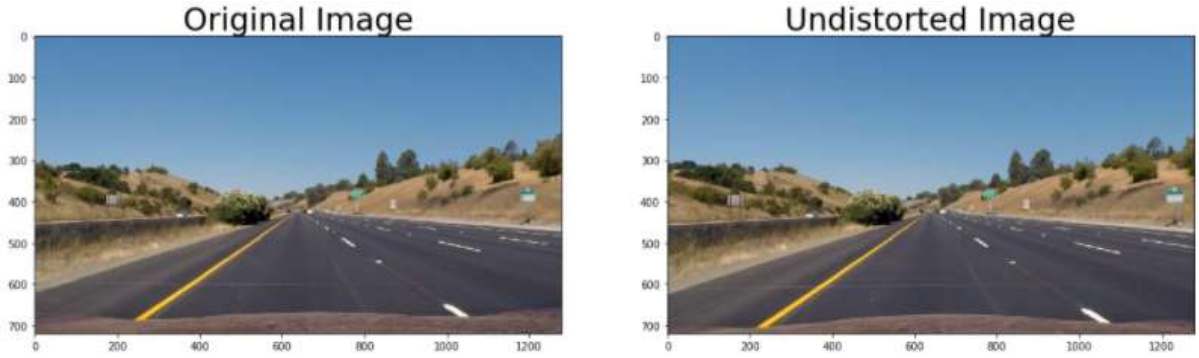


2. PIPELINE

The following are the steps in the pipeline for a single frame:

a. Provide an example of a distortion-corrected image.

Use the camera calibration coefficients to undistort the frame



b. Describe how (and identify where in your code) you used color transforms, gradients or other methods to create a thresholded binary image. Provide an example of a binary image result.

Convert the color image to binary image. I have used a combination two different threshold parameters for binary conversion. The first parameter is Sobel threshold and next one is the s component of the hls converted image

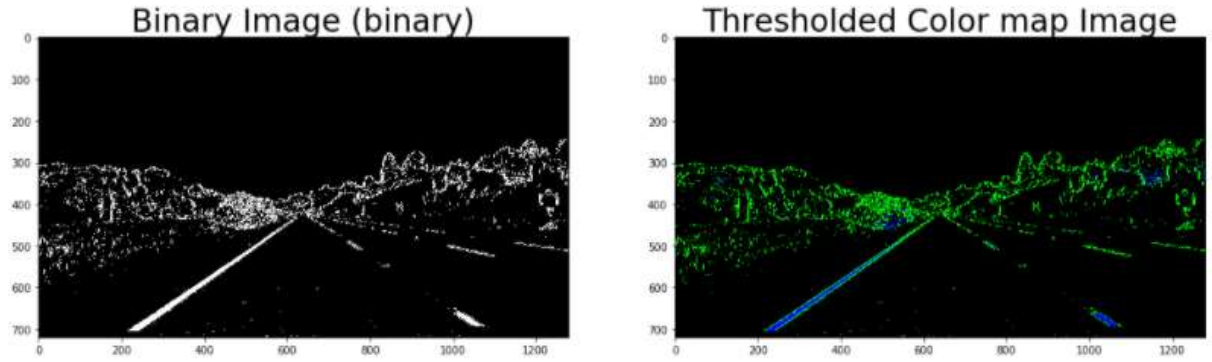
Sobel threshold: (20,100)

S threshold : (170,255)

THRESHOLDED BINARIES

In [4]:

```
def to_binary(img, s_thresh=(170, 255), sx_thresh=(20, 100)):
    img = np.copy(img)
    # Convert to HLS color space and separate the V channel
    hls = cv2.cvtColor(img, cv2.COLOR_RGB2HLS)
    l_channel = hls[:,:,1]
    s_channel = hls[:,:,2]
    # Sobel x
    sobelx = cv2.Sobel(l_channel, cv2.CV_64F, 1, 0) # Take the derivative in x
    abs_sobelx = np.absolute(sobelx) # Absolute x derivative to accentuate lines away from horizontal
    scaled_sobel = np.uint8(255*abs_sobelx/np.max(abs_sobelx))

    # Threshold x gradient
    sxbinary = np.zeros_like(scaled_sobel)
    sxbinary[(scaled_sobel >= sx_thresh[0]) & (scaled_sobel <= sx_thresh[1])] = 1

    # Threshold color channel
    s_binary = np.zeros_like(s_channel)
    s_binary[(s_channel >= s_thresh[0]) & (s_channel <= s_thresh[1])] = 1

    # Combine the two binary thresholds
    combined_binary = np.zeros_like(sxbinary)
    combined_binary[(s_binary == 1) | (sxbinary == 1)] = 1

    # Stack each channel
    color_binary = np.dstack(( np.zeros_like(sxbinary), sxbinary, s_binary)) * 255  #seting red channel to zero

    return combined_binary,color_binary
```



c. Describe how (and identify where in your code) you performed a perspective transform and provide an example of a transformed image.

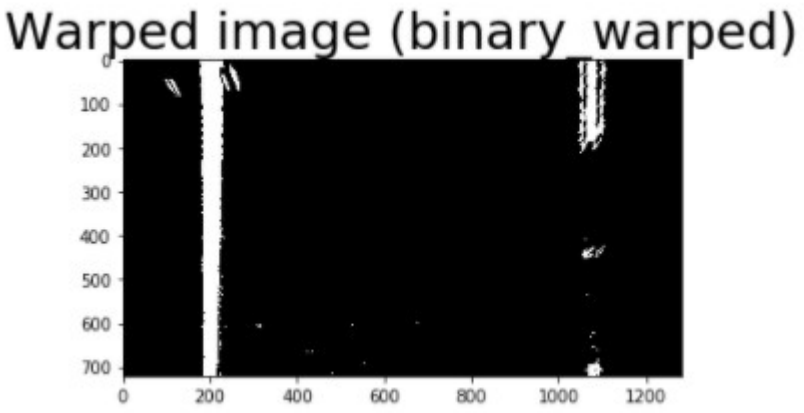
Warp the binary image to obtain the birds eye view. For this, a trapezoid is identified in the binary image representing the corners of two parallel lane endings and mapped to a conversion using “cv2.getPerspectiveTransform”. The conversion gives a transformation matrix “M”.Using M and “cv2.warpPerspective”, the binary image is converted to birds eye view.

PERSPECTIVE TRANSFORMATION: WRAP/ UNWARP

WARP: RETURNS THE BIRDS EYE VIEW

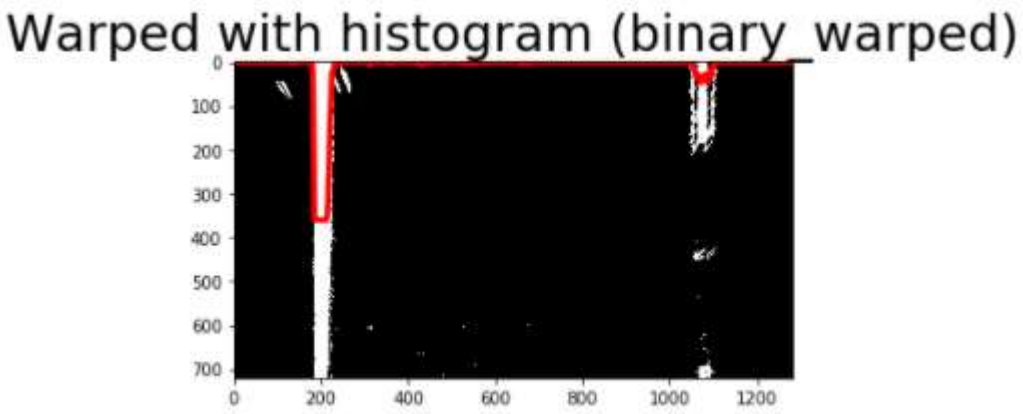
In [5]:

```
def warp(binary):
    img_size = (binary.shape[1], binary.shape[0])
    offset = 100 # offset for dst points
    src = np.float32([[525.744,499.092],[762.396,499.092],[1046.65,682.505],[261.147,682.505]])
    dst = np.float32([[2*offset, offset], [img_size[0]-2*offset, offset],
                                      [img_size[0]-2*offset, img_size[1]],
                                      [2*offset, img_size[1]]]) #2*offset= horizantal clearance on either sides
    ##        1              2
    ##
    ##
    ##        4              3  ## represent order of coordinates in the arguments


    M = cv2.getPerspectiveTransform(src, dst)
    Minv = cv2.getPerspectiveTransform(dst, src)
    # Warp the image using OpenCV warpPerspective()
    warped = cv2.warpPerspective(binary, M, img_size)


    return warped, M, Minv
```



d. Histogram

The histogram of the image is identified to understand the concentration of pixels representing the two lanes (as shown above)



e. *Describe how (and identify where in your code) you identified lane-line pixels and fit their positions with a polynomial?*

Sliding window

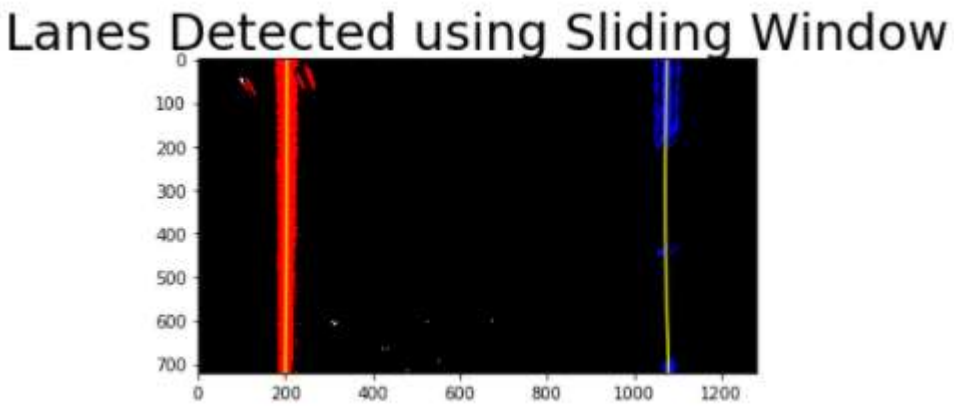
Next, a sliding window is implemented to identify the the cluster of lane pixels. The starting window corresponds to the cluster identified by the two histogram peaks. A total of 9 windows is used.

FINDING LANE PIXEL INDICES USING SLIDING WINDOW

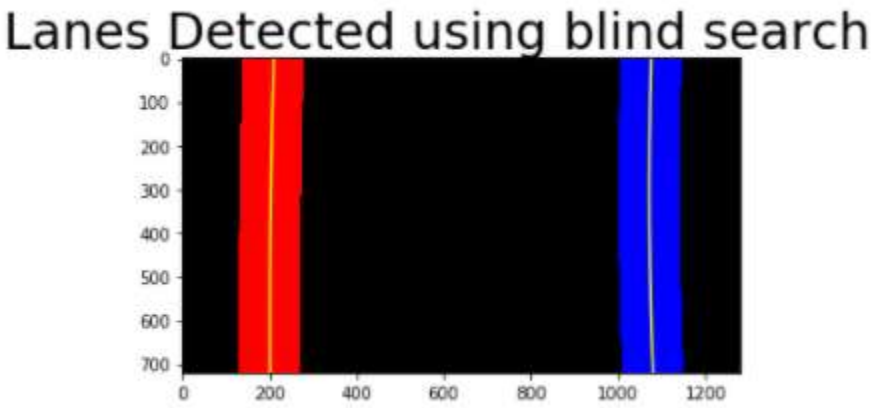
```
22]: def fit_polynomial_using_sliding_window(binary_warped):
    # Take a histogram of the bottom half of the image
    histogram = hist(binary_warped)    # Create an output image to draw on and visualize the result
    midpoint = np.int(histogram.shape[0]//2)
    leftx_base = np.argmax(histogram[:midpoint])
    rightx_base = np.argmax(histogram[midpoint:]) + midpoint

    # HYPERPARAMETERS
    nwindows = 9    # Choose the number of sliding windows
    margin = 100
    minpix = 50    # Set minimum number of pixels found to recenter window
```

After identifying the histogram peaks, the mean of the points represented by the histogram cluster is considered the starting point for the next window.
However a minimum of 50 pixels is required for the window to shift.

```
    # If you found > minpix pixels, recenter next window on their mean position
    if len(good_left_inds) > minpix:
        leftx_current = np.int(np.mean(nonzerox[good_left_inds]))
    if len(good_right_inds) > minpix:
        rightx_current = np.int(np.mean(nonzerox[good_right_inds]))
```



Blind search

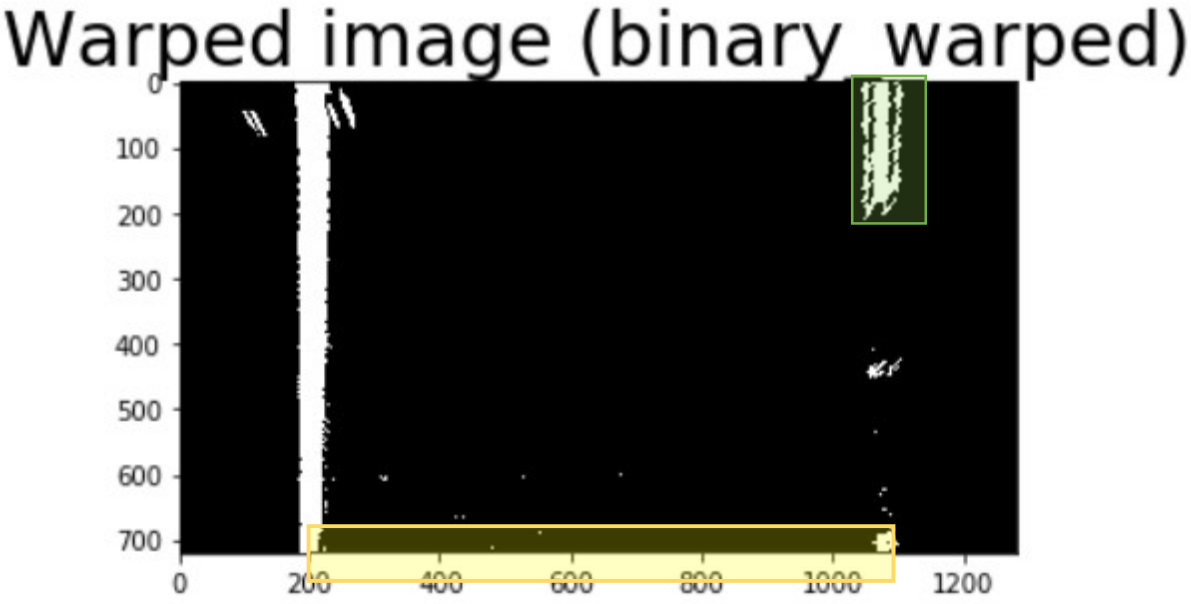


Alternately, if the lanes are identified using sliding windows with high degree of confidence, it can be assumed that the lane shape wont change for the next few frames. Hence it is only required to search in the vicinity of the pixels already identified by the sliding window. The margin chosen is 70 pixels to the either side of the polynomial fitted using sliding window. The picture above shows the margin window for searching and the polynomial that was eventually fitted in this margin.

f. Describe how (and identify where in your code) you calculated the radius of curvature of the lane and the position of the vehicle with respect to center.

RADIUS OF CURVATURE

To calculate the radius of curvature , we need to provide multiplication factors for both x and y axes, i.e how many pixels represent the height and width of the lane in real world. To do this, I closely inspected the warped image .



As seen in this image a vertical lane segment of length 3 metres is represented by 200 pixels. And the lane width of 3.7 m is represented by 900 pixels in the horizontal axis. The x and y pixels are scaled accordingly. After this, the polynomial fit function is used to fit the polynomial through the scaled coordinates.

MEASURE REAL CURVATURE

```
]:

def measure_real_curvature(binary_warped,leftx, lefty, rightx, righty):
    # Define conversions in x and y from pixels space to meters
    ploty = np.linspace(0, binary_warped.shape[0]-1, binary_warped.shape[0] )    # Generate x and y values for plottin
    ym_per_pix = 3/200 #30/720 # meters per pixel in y dimension
    xm_per_pix = 3.7/900 #3.7/700 # meters per pixel in x dimension

    left_fit_cr = np.polyfit(lefty*ym_per_pix, leftx*xm_per_pix, 2)
    right_fit_cr = np.polyfit(righty*ym_per_pix, rightx*xm_per_pix, 2)

    y_eval = np.max(ploty )

    # Calculation of R_curve (radius of curvature)
    left_curverad = ((1 + (2*left_fit_cr[0]*y_eval*ym_per_pix + left_fit_cr[1])**2)**1.5) / np.absolute(2*left_fit_cr[0])
    right_curverad = ((1 + (2*right_fit_cr[0]*y_eval*ym_per_pix + right_fit_cr[1])**2)**1.5) / np.absolute(2*right_fit_cr[0])

    return left_curverad, right_curverad
```

To get the position of the vehicle, the image centres and lane centres are calculated and the difference between the two is the offset and is scaled accordingly.

```
image_center=img.shape[1]//2
lane_center=(left_fit[0]*img.shape[0]**2 + left_fit[1]*img.shape[0] + left_fit[2]+
right_fit[0]*img.shape[0]**2 + right_fit[1]*img.shape[0] + right_fit[2])//2
offset=np.abs(image_center-lane_center)*3.7/900
```

g. Provide an example image of your result plotted back down onto the road such that the lane area is identified clearly.

3. PIPELINE (VIDEO)

In the video, the sanity checks have been implemented for checking lane curvature, lane width and difference in radius of curvatures of two lanes. If the sanity check fails, that frame is skipped and an average of previous 5 good frames is made. If consecutive frames fail, then the sliding window is implemented and calculations start from scratch., otherwise blind search is implemented for every normal frame. The averaging of frmaes helps smoothen the overlay.

The video can be found at : “CarND-Advanced-Lane-Lines\output_Videos” as project_video_output.mp4

DISCUSSION

The algorithm is not successful in the challenge video and this could be the improvement that can be made to this project. Also since histogram considers lanes are vertical, lanes which have high curvature will fail using histogram based sliding window.