Advanced Lane Finding Project

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.

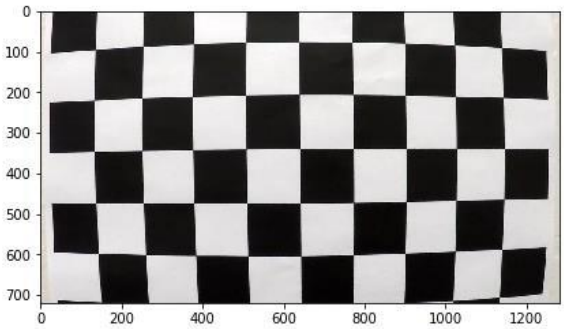
Rubric Points

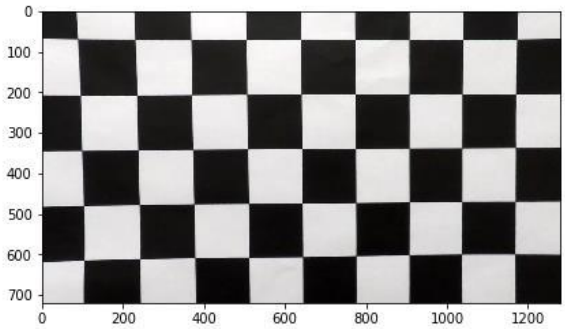
Camera Calibration

1. **Have the camera matrix and distortion coefficients been computed correctly and checked on one of the calibration images as a test?**

I start by preparing "object points", which will be the (x, y, z) coordinates of the chessboard corners in the world. Here I am assuming the chessboard is fixed on the (x, y) plane at z=0, such that the object points are the same for each calibration image. Thus, `objp` is just a replicated array of coordinates, and `objpoints` will be appended with a copy of it every time I successfully detect all chessboard corners in a test image. `imgpoints` will be appended with the (x, y) pixel position of each of the corners in the image plane with each successful chessboard detection.

I then used the output `objpoints` and `imgpoints` to compute the camera calibration and distortion coefficients using the `cv2.calibrateCamera()` function. I applied this distortion correction to the test image using the `cv2.undistort()` function and obtained this result:



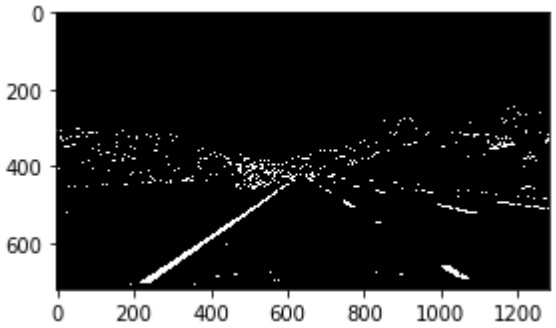


Pipeline (single images)

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

To demonstrate this step, I will describe how I apply the distortion correction to one of the test images like this one:

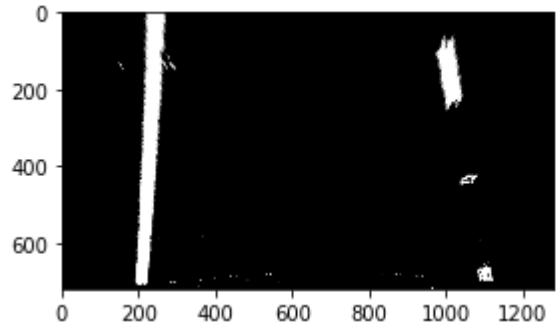
3. **Describe how (and identify where in your code) you used color transforms, gradients or other methods to create a thresholded binary image. Provide an example of a binary image result.**
   I used a combination of color and gradient thresholds to generate a binary image.
   I used thresholds on Saturation in HSL Color space.
   I used X and Y, magnitude and Direction thresholds as gradient thresholds to generate the following image



4. **Describe how (and identify where in your code) you performed a perspective transform and provide an example of a transformed image.**
   I have used cv2.perspective Transform to compute the perspective transform of the image.

Source	Destination
490,422	0,0
810,482	1280,0
1250,720	1250,720
40,720	40,720

```
plt.imshow(result,cmap='gray')
img_size = (result.shape[1],result.shape[0])
src = np.float32([[490, 482],[810, 482],
                  [1250, 720],[40, 720]])
dst = np.float32([[0, 0], [1280, 0],
                 [1250, 720],[40, 720]])
M = cv2.getPerspectiveTransform(src, dst)
warped = cv2.warpPerspective(result, M, img_size, flags=cv2.INTER_LINEAR)
plt.subplot(1, 3, 3)
plt.imshow(warped,cmap='gray')
```

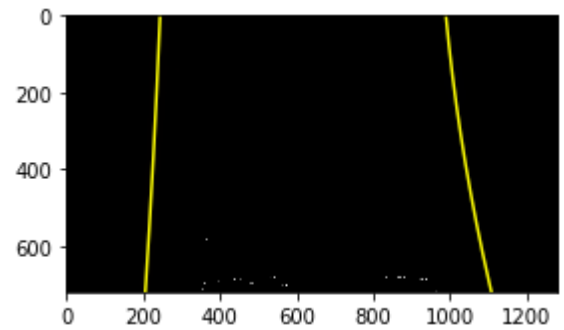


I verified that my perspective transform was working as expected by drawing the `src` and `dst` points onto a test image and its warped counterpart to verify that the lines appear parallel in the warped image.

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

I plotted a histogram of the warped Image. Then I identified peaks in the histogram. Then we take window of definite Size to implement sliding window protocol.

I use sliding window to find the left and right points and Then I used polyfit to fit those points into the polynomial of degree 2.



5. **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.**

The radius of curvature are calculated at a point in the ploynimial to identify the left and right curvature and then these curvature are averaged to get the radius of curvature as you can see in below image. AS we have pixels so we multiply them with constants to get radius of curvature in metres

```
    ym_per_pix = 30/720 # meters per pixel in y dimension
    xm_per_pix = 3.7/700 # meters per pixel in x dimension

# Fit new polynomials to x,y in world space
    left_fit_cr = np.polyfit(np.array(lefty,dtype=np.float32)*ym_per_pix,np.array(leftx,dtype=np.float32)*xm_per_pix, 2)
    right_fit_cr = np.polyfit(np.array(righty,dtype=np.float32)*ym_per_pix,np.array(rightx,dtype=np.float32)*xm_per_pix, 2)
# Calculate the new radii 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])
# Now our radius of curvature is in meters
```

To calculate centre offset we assume that camera is fitted in middle on the top of vehicle so we deference between the image centre and our car position

```
    lane_center = (leftx[719] + rightx[719])/2
    car_position = result.shape[1]/2
    # Define conversions in x and y from pixels space to meters


    xm_per_pix = 3.7/700 # meteres per pixel in x dimension

    font = cv2.FONT_HERSHEY_SIMPLEX
    cv2.putText(result,'Center Offset : '+str(np.round((np.absolute(car_position - lane_center))*xm_per_pix ,2)),(350,100), font,
    return result
```

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

Pipeline (video)

1. Does the pipeline established with the test images work to process the video?

Yes, it Did, Here is link.

Discussion

1. **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?**

   Challenges:

   - I had to experiment a lot with gradient and colour channel thresholding.
   - The challenge video has a section where the car goes underneath a tunnel and no lanes are detected

   Failure:

   - The pipeline is failing in the challenge section as it was not able to detect lines in the shadow portion.
   - In harder video challenge there is a sharp turn where pipelines seems to fail.

   Area of Improvement

   - Better perspective transform and to choose smaller section of road to take better perspective transform.
   - Use other colour channels such as luminous or hue.