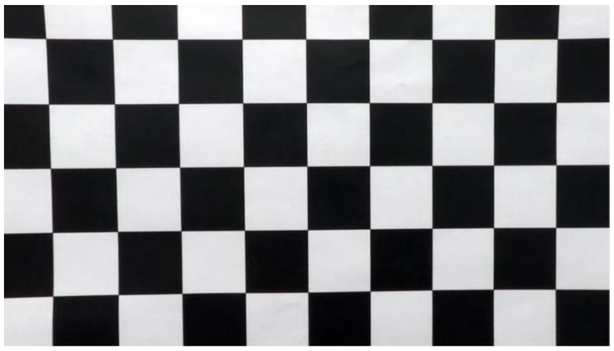
ADVANCED LANE FINDING PROJECT

The following steps describe advanced lane finding algorithm. This algorithm uses various tools by openCV, finally detects lanes, also computes the left and right lane marking's curvatures in real world co-ordinates.

CAMERA CALIBRATION

The code for this cell is contained in the beginning of the iPython notebook (after all the imports). This function calibrates the camera to return camera matrix and distortion coefficients. For each image in the folder camera_cal/ this function computes the image points. Also appends the already prepared object points. The function cv2.calibrateCamera() returns the camera matrix and distortion coefficients.

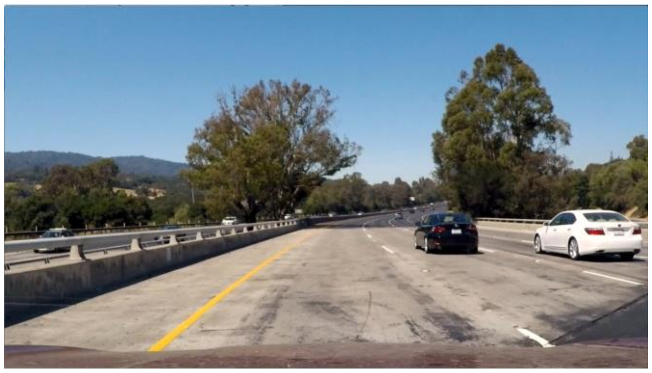
This data was further used in the cv2.undistort() function, to correct the distortion in the image. After applying the above two operations. I got this undistorted result:



PIPELINE

DISTORTION CORRECTION

I applied the distortion correction to one of the test images. The result looks like below:

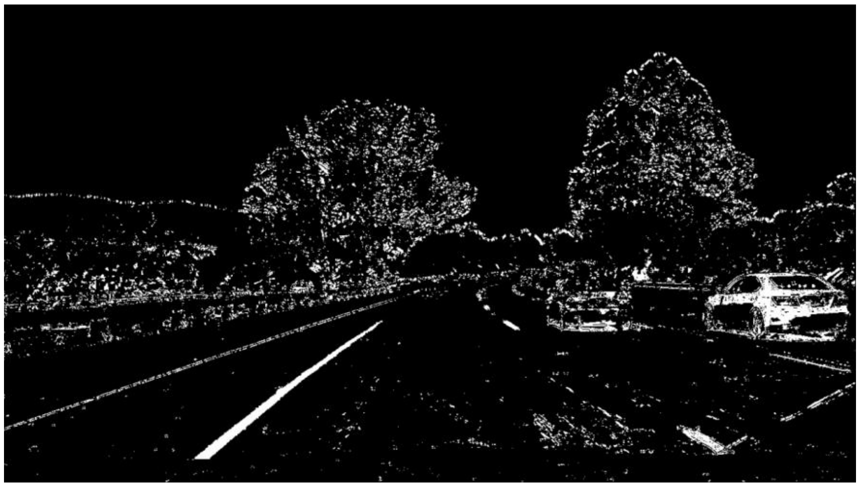


Thresholding

A combination of color and gradient thresholds were used to generate a binary image and discard unnecessary detail in the image. I used the following thresholds:

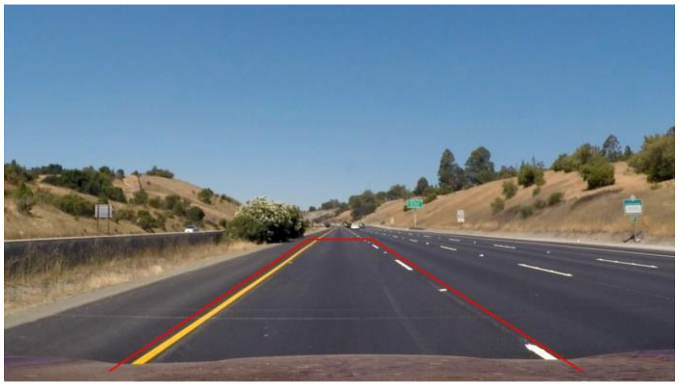
- SobelX
- SobelY
- Gradient Magnitude
- Gradient Direction
- H and S Channel Thresholds

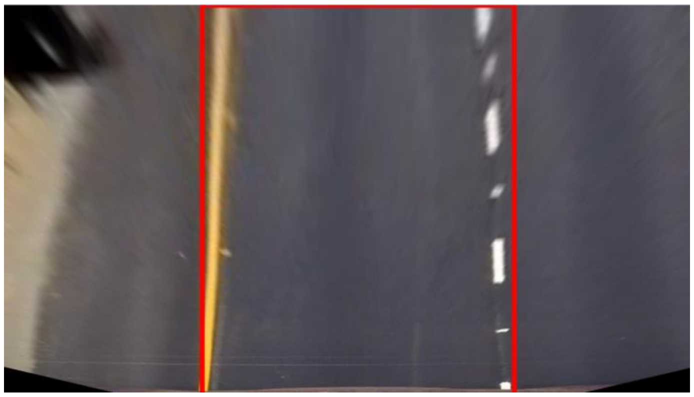
The above thresholds were combined to form a binary image. These images can be found in the output_images folder with suffix (unthresh) meaning undistorted thresholded image. One example below:



Perspective Transform

In the 6th cell of the python notebook, I have written a class called Process Image. It provides two functions, warp and unwarp. The src and dst matrices are hardcoded by trial and error. I verified the warping happens accurately by making a test on the straight_line images. Example of output:

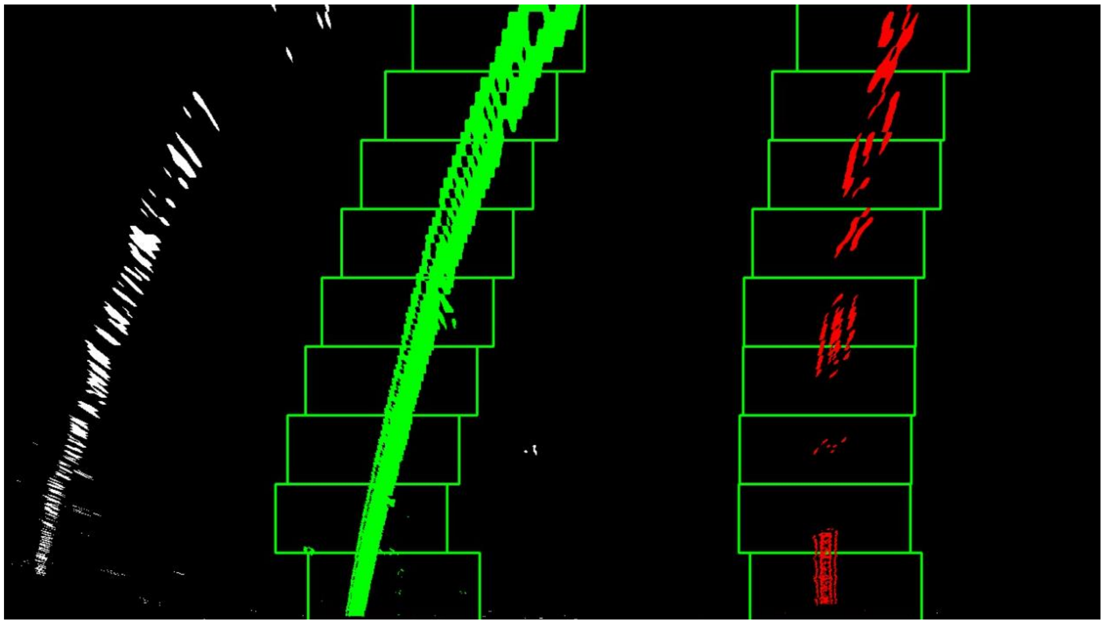




Identifying lane lines

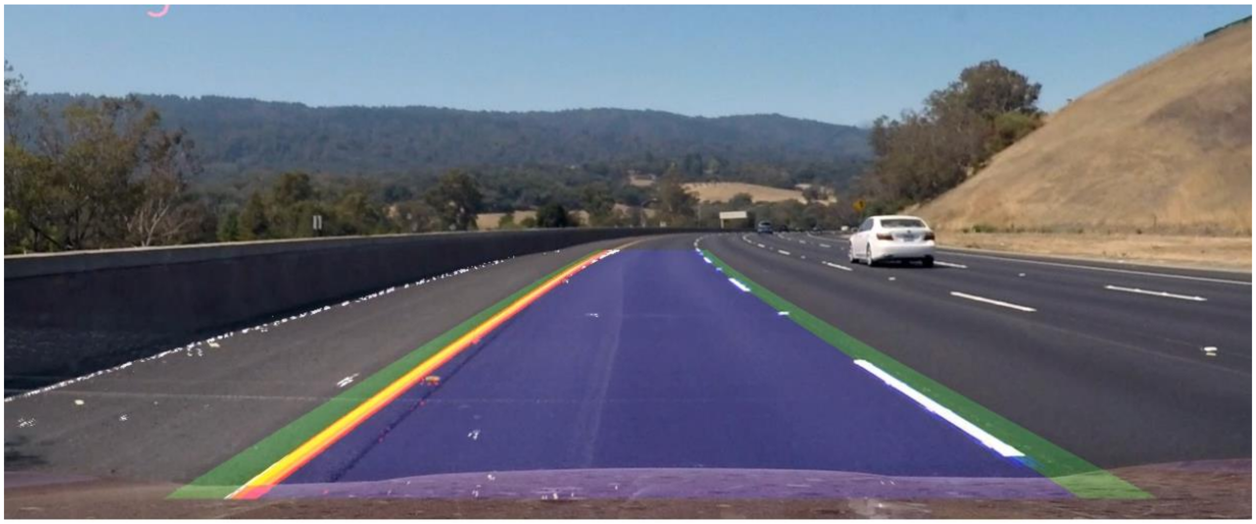
Perspective transform was then applied to undistorted, thresholded, warped images. Then, I plotted the histogram and identified the peaks. These peaks indicate the X positions of the left and the right lane marking. Then, a sliding window algorithm is used.

These windows moved up the image based on the mean position of the pixels. The output image colored all the identified pixels to indicate lane lines. In my 5th python cell, histogram function is written. The example output of lane lines detected by sliding window algorithm is below:



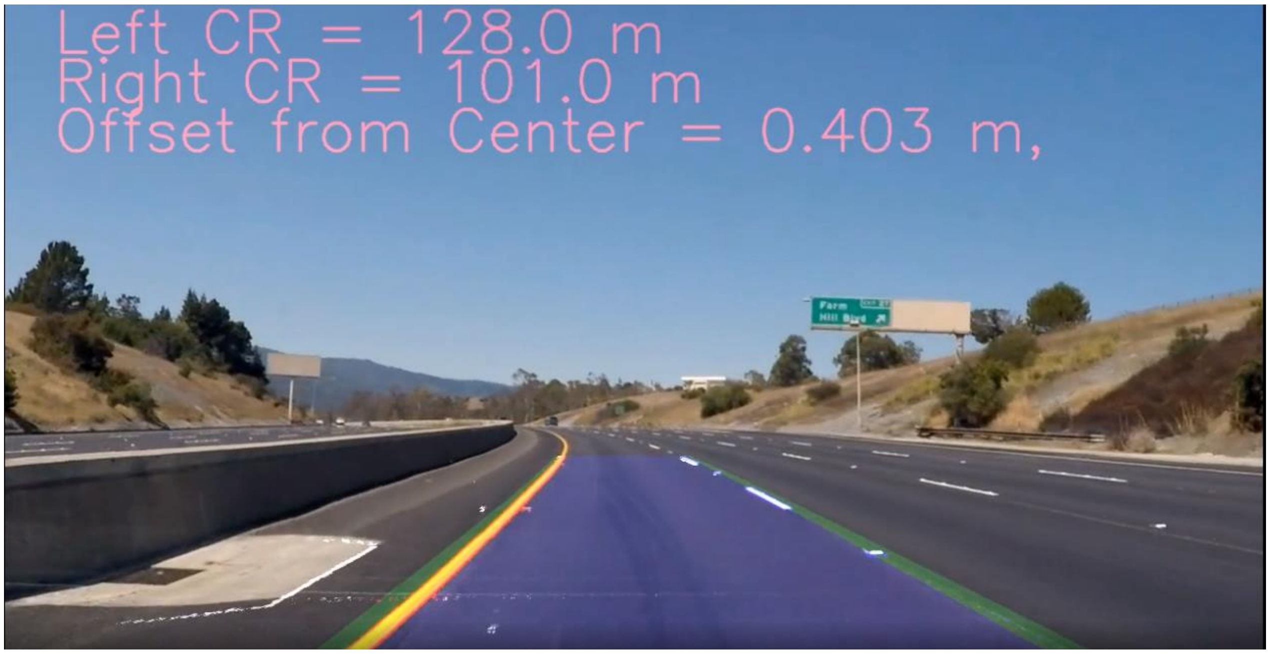
POLYNOMIAL FITTING

The pixel positions were identified, and the polynomial of the form ax*2 + bx + c was fit using numpy.folyfit() function. This function is in my ProcessImage class in the 6th cell of the python notebook. The search area for lanes was drawn around this polynomial curve. Finally, the lane itself was also drawn. Example of the output below:



CALCULATION OF RADIUS OF CURVATURE

I have done this calculation in the 7th Cell of Python notebook. With the fitted leftX points and the RightX points, a new polyline was fit. But this time, the XPoints and YPoints were multiplied with a factor (meters per pixel) to convert the measurement from Pixel Space to Real World space. The results were also appended to the image(hence, the video) is updated in real time. Offset from the center of the lane is computed. Example image:



```
Left CR = 128.0 m
Right CR = 101.0 m
Offset from Center = 0.403 m,
```

FINAL OUTPUT OF THE VIDEO

The pipeline seems to work successfully on the final project video. The link to the video is given below:

project_video.mp4 (udacity-student-workspaces.com)

DISCUSSION

In Summary, The pipeline was designed in the following manner:

1. The pipeline worked reasonably well for the project video. However, it did not work as precisely on the challenge video. The pipeline might fail if along with the lane markings, the cracks on the road can also resemble the lane markings.
2. Lanes are not checked to be valid. One way to check the validity of the lane would be that its width to be under a reasonable threshold. If such a lane is not found, Search again using the sliding window algorithm
3. Disappearing lane markings could be an issue
4. The design of the whole project could be improved. There is duplication of code in the notebook specially for saving the result images in different stages of the pipeline.
5. Selection of the points for perspective transform could be smarter, instead of hardcoding like I have done in the project.
6. Thresholding can be made better by using more parameters like Red and Gray channels, along with the existing thresholding parameters