Advanced Lane Finding Project

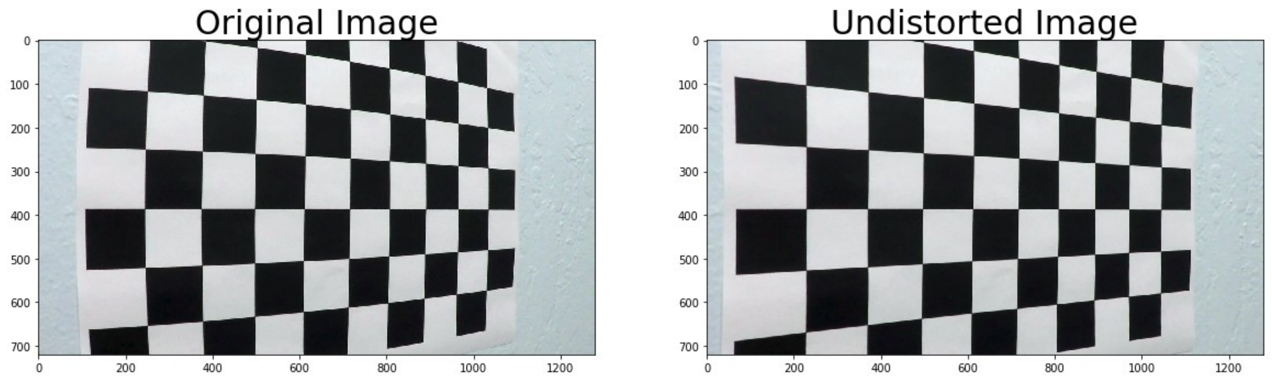
The goals / steps of this project are the following:

- 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

1. Camera Calibration

The code for this step is contained in the second code cell of the IPython notebook located in "./Project2.ipynb"
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:

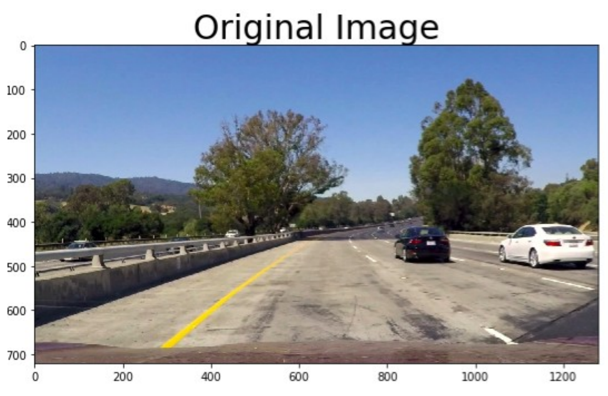


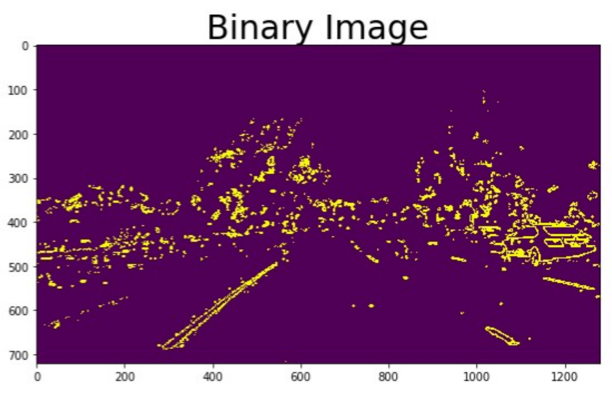
Pipeline

1. Distortion correction on image (single images)

I applied the distorion correction to the followig image and obtained the output

2. Filter Used
I used a combination of HUE (for yellow lanes) and lightness(for white lanes) thresholds to generate a binary image (thresholding through x,y and magnitude gradient). Here's an example of my output for this step. (note: this is not actually from one of the test images)
I tried using other feed(combining L from LUV and B from RGB), but L & H combination had the least noise.





3. perspective transform

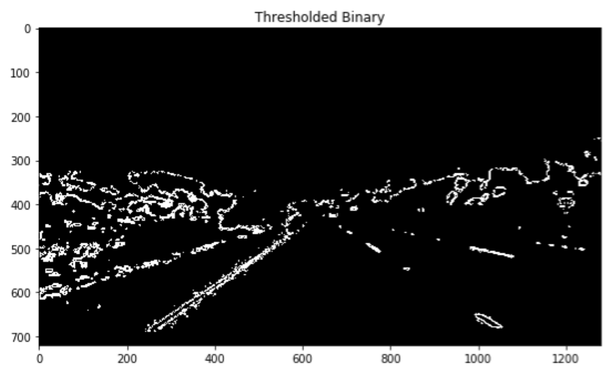
The code for my perspective transform appears In[17] cell of the IPython notebook. The code takes an image (img), as well as source (src) and destination (dst) points. I chose the hardcode the source and destination points in the following manner:

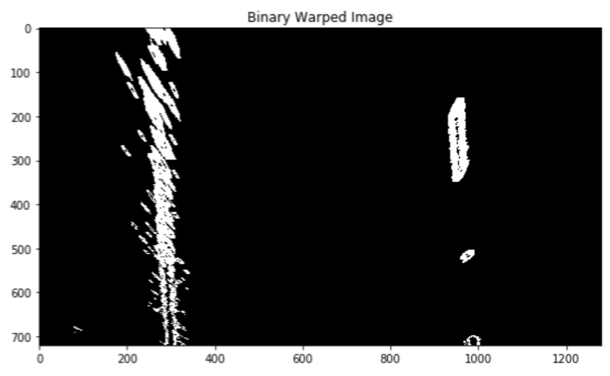
```
src = np.float32(
      [  548,   476 ],
      [  761,   476 ],
      [ 1136,   673 ],
      [  163 ,   673 ]])
dst = np.float32([[offset, 0], [img_size[0] - offset, 0], [img_size[0] - offset, img_size[1]], [offset,
img_size[1]]])
```

This resulted in the following source and destination points:

Source	Destination
548, 476	200, 0
761, 476	1150, 0
1136, 673	1150, 720
163, 673	200, 720

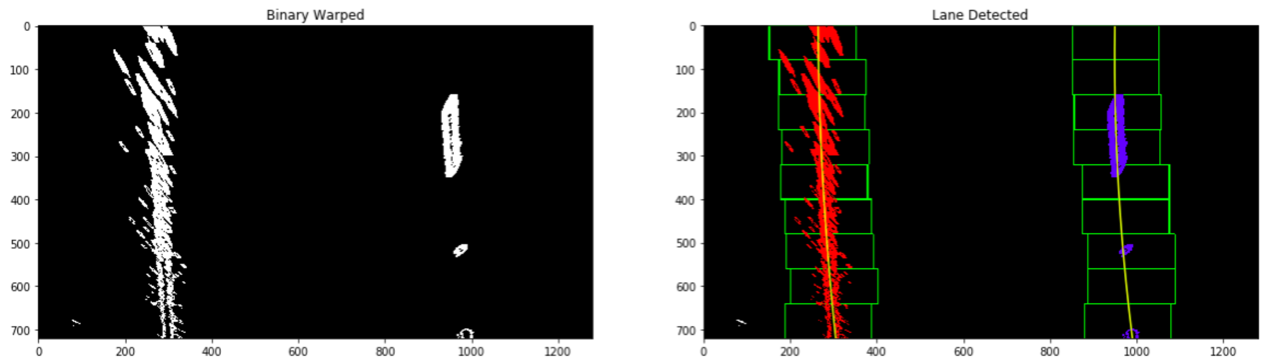
I verified that my perspective transform was working as expected by drawing the src and dst points onto a test image and its warped counterpart to verify that the lines appear parallel in the warped image.





4. Fitting Polynomial

Then I did some other stuff and fit my lane lines with a 2nd order polynomial with Sliding window and fitting polynomial which resulted in the following

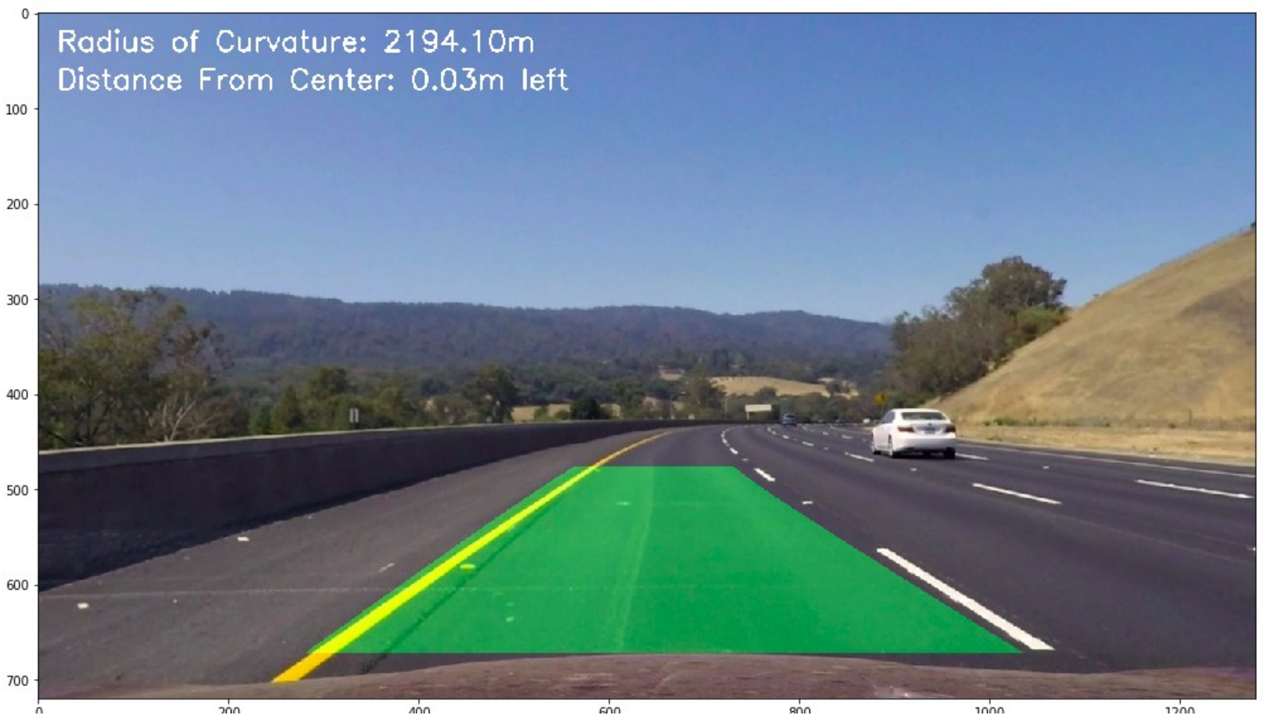


5. radius of curvature of the lane and the position of the vehicle with respect to center.

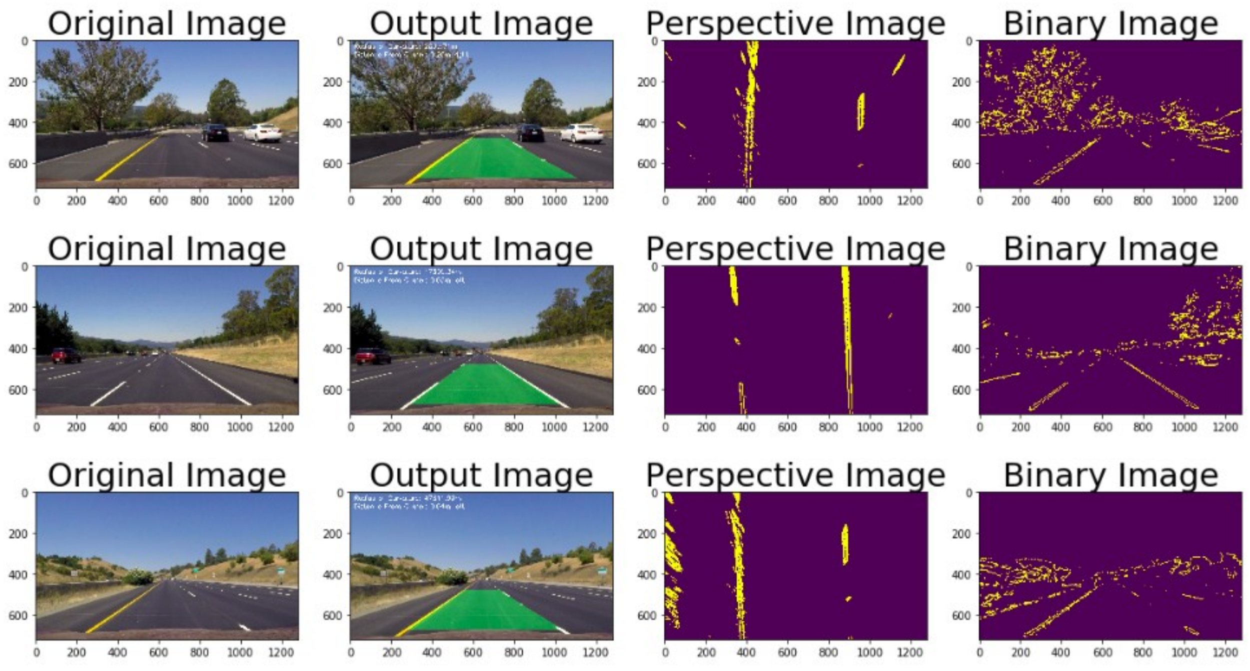
I did this in Cell In[10] in my notebook.

6. example image of result plotted back down onto the road

The code for generation of lane area identification in image is available in In[25] of notebook and yielded the following


Radius of Curvature: 2194.10m
Distance From Center: 0.03m left

7. End-to-end generation of lane identification on image (final pipeline)



Pipeline (video)

Here's a link to my video result

Discussion

Implementation

The approach i used in this project is to firast identify which feed of image gives maximum usable data and least noise. I identified a combination of HUE and Lightness can yield the yellow and white lanes. Following this i applied hough lines and on interested area and identified the lane.

Following the identification of Lanes, i transformed the binary image to birds eye view and fitted with a polynomial by choosing the histogram from the first frame(under the assumption that the lanes are parallel and the car is maintained at the center of the lane). Following this, i calculated the radius of curvature and the distance from center which is overlapped in the images

Issues/Failure Cases

There are minor distortion observed in the project video which is negligible as the area is within the bounds of lanes

For the challenge and harder challenge videos, the lane is detected but not in all framews and fails with increase in exposure

Improvements

Lane Identification (in challend and harder challenge) color selection may require fine tuning to improve quality of identification as the identification is affected by exposure.

Exploring other color spaces for advanced challenge might improve the lane detection.