UDACITY

PROJECT

Advanced Lane Finding

A part of the Self Driving Car Engineer Nanodegree Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Requires Changes

3 SPECIFICATIONS REQUIRE CHANGES

Awesome submission. It is clear a lot of work has gone into this pipeline and the results are generally great.

There are a few minor issues to be resolved in the next submission. I have provided suggestions to help you address them.

Keep doing an awesome job.

Writeup / README

The writeup / README should include a statement and supporting figures / images that explain how each rubric item was addressed, and specifically where in the code each step was handled.

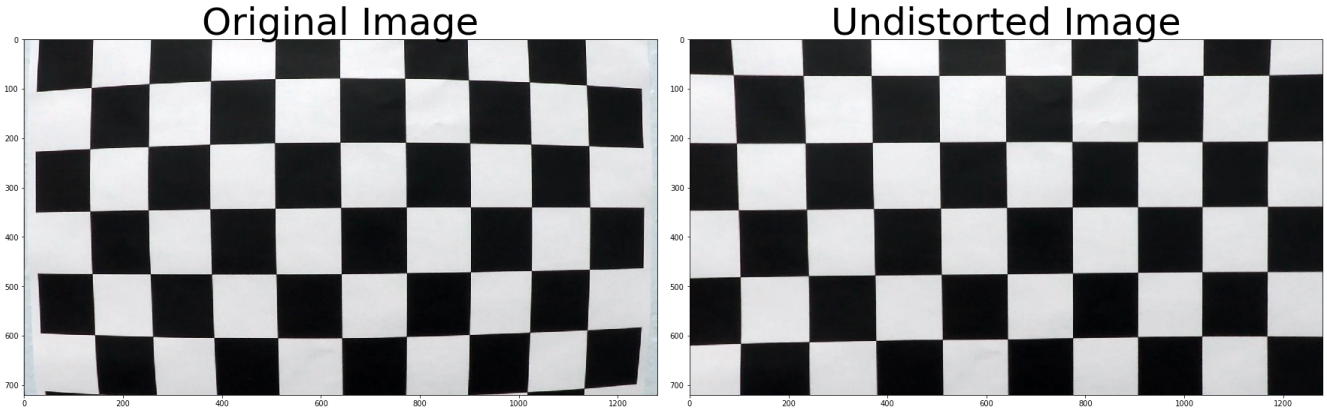
Awesome. `writeup.md` was an excellent reference for understanding the pipeline. Nice use of images to support the narrative.

Great to see you include a description for each rubric item explaining how you accomplished the task.

Camera Calibration

OpenCV functions or other methods were used to calculate the correct camera matrix and distortion coefficients using the calibration chessboard images provided in the repository (note these are 9x6 chessboard images, unlike the 8x6 images used in the lesson). The distortion matrix should be used to un-distort one of the calibration images provided as a demonstration that the calibration is correct. Example of undistorted calibration image is Included in the writeup (or saved to a folder).

Fantastic work.! OpenCV functions findChessboardCorners and calibrateCamera were used to calibrate the correct camera matrix and distortion coefficients. Un-distortion demonstration on a test checkerboard looks great. Good job.!



Pipeline (test images)

Distortion correction that was calculated via camera calibration has been correctly applied to each image. An example of a distortion corrected image should be included in the writeup (or saved to a folder) and submitted with the project.

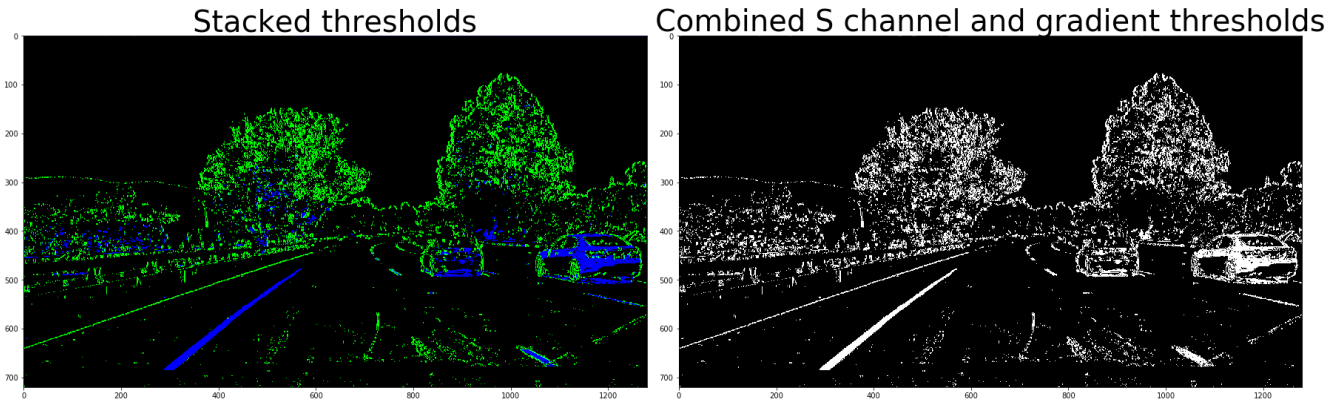
Awesome job applying distortion corrections to real world images. The results cannot be checked because the submission did not include a real world image that clearly shows the un-distortion effects.

However, the un-distort process is the first action performed on all binary images by the pipeline. Nice.!

A method or combination of methods (i.e., color transforms, gradients) has been used to create a binary image containing likely lane pixels. There is no "ground truth" here, just visual verification that the pixels identified as part of the lane lines are, in fact, part of the lines. Example binary images should be included in the writeup (or saved to a folder) and submitted with the project.

Well done. The Sobel operator was applied successfully to the L channel of HLS to find x gradients and then thresholds were applied to the absolute values vectors. A color threshold range was applied to the S channel of HLS. A combination of these results was used to produce binary images for finding likely lane pixels.

SUGGESTION:
If binary images are provided for all test images we are better equipped to evaluate the effectiveness of your strategy.



There are a significant number of pixels that do not belong to lane lines.
This rubric meets specifications because a binary image was produced.

recommendation

I encourage you to continue trying other color spaces to isolate the yellows and whites. Try thresholding in the RGB, HSV and HSL channels. The R and V channels are strong component of yellow and white, so try thresholding them and select multiple ranges. Also LUV and Lab are good possibilities. Try L of Luv for whites and `b` of Lab for yellows. By better identifying yellows and whites, the pipeline can rely less on the gradients. This is especially important and helpful for dealing with shadows and variations in the road lighting and appearance.

problem

Comments and variable names contradict actual code here

```
# Convert to HSV color space and separate the V channel
hsv = cv2.cvtColor(img, cv2.COLOR_RGB2HLS).astype(np.float)
```

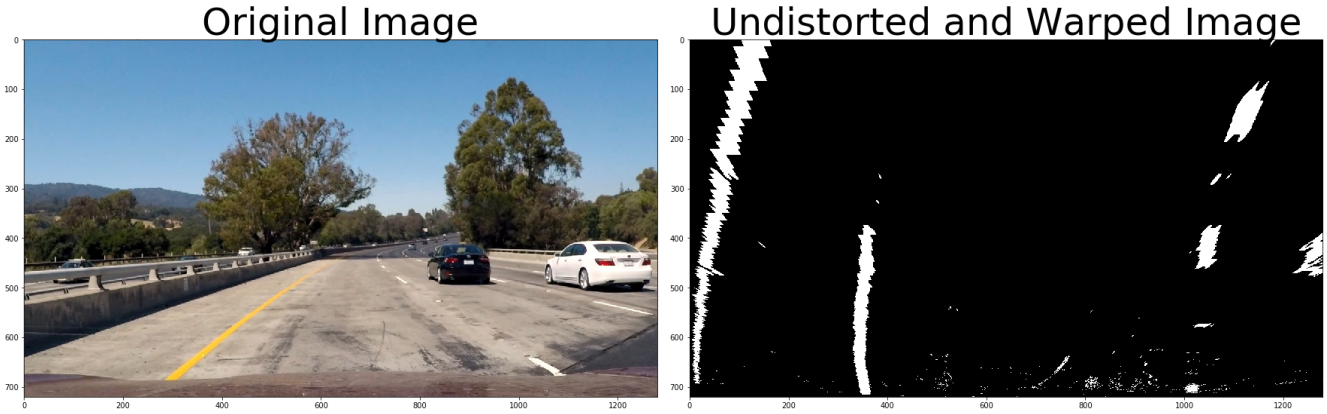
requirement

Please have this improved.

OpenCV function or other method has been used to correctly rectify each image to a "birds-eye view". Transformed images should be included in the writeup (or saved to a folder) and submitted with the project.

Nice work. Binary images are transformed to obtain a birds-eye perspective of the lane lines. The lane lines do not appear to be parallel but it is hard to be sure.

A good perspective transform is very important to obtaining good results. It is best to use an image with straight lane lines for this part of the project. If the lane lines are not transformed correctly the final polygon will have a curve when it should not.



When creating the binary image it may prove helpful to create the birds-eye perspective first. The reason is that error pixels in the binary get stretched out when the birds-eye transformation is performed second.

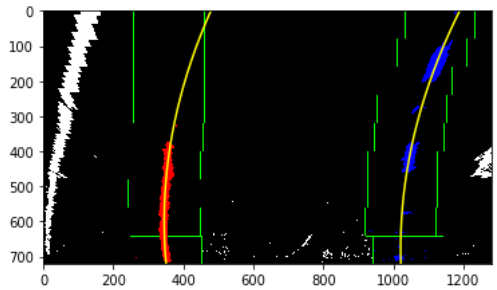
I noticed that you performed the operations in this order. Was this a conscious choice?

recommendation

Try to find source and destination points such that the lane lines appear to be parallel after transforming an image containing a straight section of road.

Methods have been used to identify lane line pixels in the rectified binary image. The left and right line have been identified and fit with a curved functional form (e.g., spine or polynomial). Example images with line pixels identified and a fit overplotted should be included in the writeup (or saved to a folder) and submitted with the project.

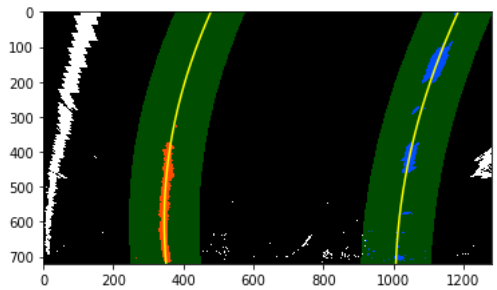
Nicely done. A histogram/sliding window method is used for identifying the left and right lane line pixels in a rectified binary image. The cv polyfit function is used to fit curves to the identified pixels and they are used to calculate lane curvature and vehicle lane position.



There are a significant number of pixels that do not belong to lane lines. Continue to explore better binary threshold selectors.

Once lines were found they were used to limit/focus the area of the search in the next frames. Terrific.!

Basic sanity checks were used to reject unusable (too few points) curves and replace them with results obtained from prior frames.



comment

The following code is executing the full blind search prematurely, and in the process is throwing away valuable information about the location of likely lane line pixels. It is necessary sometimes to perform a full blind histogram search, but it should be a last resort, only executed when you are certain that the lane lines will not reappear in the expected locations.

```
# Check if lines were last detected; if not, re-run first_lines
if left_line.detected == False | right_line.detected == False:
    draw_first_line(img, mtx, dist)
```

This code is essentially allowing random artifacts or even noise to replace the lane lines as soon as the thresholding strategy fails in a single frame, and worse, it does not even wait for failure on both lines. This is not good behavior.

recommendation

Have this improved.

suggestions

Some ideas to consider when choosing sanity checks are :

- Are the curve estimates close in their curvature, to each other and their prior frame estimates?
- Are the lines approximately parallel?
- Are the lines approximately the correct distance apart?
- Are they approximately in the same place at their closet position to the vehicle (as the last accepted curves).

Here the idea is to take the measurements of where the lane lines are and estimate how much the road is curving and where the vehicle is located with respect to the center of the lane. The radius of curvature may be given in meters assuming the curve of the road follows a circle. For the position of the vehicle, you may assume the camera is mounted at the center of the car and the deviation of the midpoint of the lane from the center of the image is the offset you're looking for. As with the polynomial fitting, convert from pixels to meters.

Excellent work in estimating the lane curvature and vehicle's lane placement position. Fantastic job displaying these estimates in the output video. The numbers look reasonable.

More visually pleasing results that are easier on the viewer can be obtained by averaging over a few frames and updating the values over the same number of frames.

problem

On the first frame of the output video, the annotations are overwritten. The correct values are not clearly discernible.

requirement

Please have this corrected.

The fit from the rectified image has been warped back onto the original image and plotted to identify the lane boundaries. This should demonstrate that the lane boundaries were correctly identified. An example image with lanes, curvature, and position from center should be included in the writeup (or saved to a folder) and submitted with the project.

The warped back results look good.!

Pipeline (video)

The image processing pipeline that was established to find the lane lines in images successfully processes the video. The output here should be a new video where the lanes are identified in every frame, and outputs are generated regarding the radius of curvature of the lane and vehicle position within the lane. The pipeline should correctly map out curved lines and not fail when shadows or pavement color changes are present. The output video should be linked to in the writeup and/or saved and submitted with the project.

Although there are brief imperfections this is an impressive submission. The input video is successfully processed by the pipeline! Every frame has annotations for the lane curvature and vehicle lane position, even when the road surface changes or when shading is encountered. The lane estimates are shown and for the most part are very good. Most of time the annotations are clearly displayed against the background.

requirement

Please have annotations clearly displayed on all frames, including the first (see comments above).

There were several instances in the video when one or both of the highlighted lane lines are significantly off from the true positions, typically when the car crossed a bridge and the road surface changed or when there was shadowing on the roadway. Here are three images showing what's happening. They are taken from the video supplied in the submission:

0.8 meters right of center
Radius of Curvature = 499.0(m)

0.04 meters left of center
Radius of Curvature = 1019.0(m)

To meet specifications the lane lines should reasonably track the actual lane lines and the pipeline should not break down on curves or on changes in lighting on the road surface.

To correct this deficiency I have some suggestions:

- Implement more/better sanity checks to reject unusable results and replace them with a result from prior frames. Some forms of averaging (of polynomial coefficients) over a few frames may be helpful. However don't over do it, because it is important to avoid reacting too slowly on curves or to changes in vehicle position within the lane.
- Continue to investigate color spaces to find a better thresholding solution. The goal is to rely more on color identification and less on gradients (which are not so useful in shadows or changing road conditions). The R and V color channels are strongly represented in yellows and whites. Try L of Luv and `b` of Lab also.
- Capture images of video frames where problems are occurring and run the pipeline on those images to avoid long processing times while trying to solve a localized problem.
- Exponential smoothing is an alternative to averaging over N frames. If you have a New frame and an Old frame, smooth by updating the New as follows: New = gamma * New + (1-gamma) * Old. 0 < gamma < 1.0
- Use an area around the lines fitted in prior images to search for lane pixels in a new image. But be prepared to do a full histogram search if valid data is not obtained for a few frames. The images above suggest this is needed; more precisely, it needs to be implemented in a better way.

Discussion

Discussion includes some consideration of problems/issues faced, what could be improved about their algorithm/pipeline, and what hypothetical cases would cause their pipeline to fail.

An excellent reflection on the pipeline, it's strengths, weaknesses and areas with potential for improvements.

You are correct: a good perspective transform is very important to obtaining good results.

You are correct: in order to do better on the challenge videos when difficult shading and lighting issues exist, it will be helpful to consider stronger thresholding for whites and yellows and relying less on using gradients and edge detections.

RESUBMIT

DOWNLOAD PROJECT