Writeup Template

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.

Computed the camera matrix and distortion coefficients.

Real cameras usually cause two kinds of distortion:

Radial distortion

Light rays often bend a little too much or too little at the edges of these lenses. This creates an effect that distorts the edges of images, so that lines or objects appear more or less curved than they actually are. There are three coefficients needed to correct for radial distortion: k1, k2, and k3. The distortion coefficient k3 is required to accurately reflect major radial distortion (like in wide angle lenses)

Tangential distortion

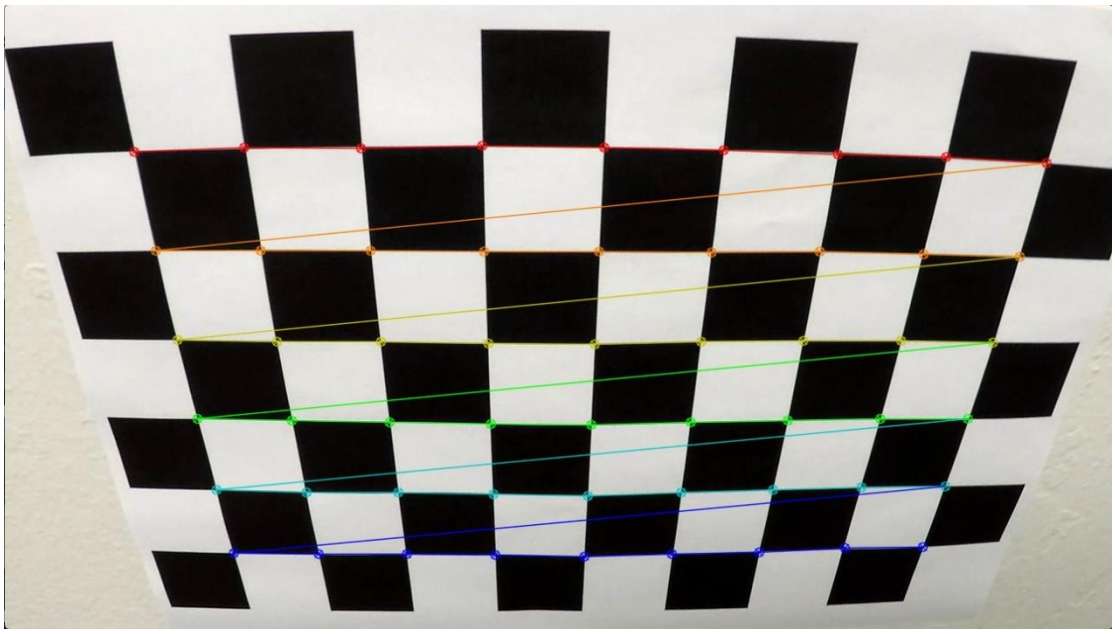
When a camera's lens is not aligned perfectly parallel to the imaging plane, where the camera film or sensor is, it will makes an image look tilted so that some objects appear farther away or closer than they actually are. We can take picture of known shapes, then we detect and correct the image distortion

How I do it

We can use the OpenCV functions findChessboardCorners() and drawChessboardCorners() to automatically find and draw corners in an image of a chessboard pattern.

- Count the number of inner corners per row and column
- Prepare a list of object points from these corners, which will be the same for all images in the set of calibration images
- Find the image chess corners using opencv function findChessboardCorners
- Append object points(same for all images) and image points to different lists
- The camera is calibrated using opencv function calibrateCamera using the object points and image points obtained above to obtain the distortion coefficients and camera coefficient

matrix.



Pipeline (single images)

1. Distortion-corrected image.

The camera matrix and distortion coefficients obtained in the previous steps were used to correct distortion in each frame of the video.

2. Combined binary image

To correctly detect the lane lines I am going to use gradient threshold and color threshold to create a binary image:

Gradient Threshold

Using canny edge detection algorithms to detect lines in images, identifying lane lines by the size of the slope. The slope of the lane line is close to the vertical, so the $Sobel_x$ Operator is efftive in finding lane.

Color Threshold

HLS space (hue, lightness, and saturation) is one of the most commonly used color spaces in image analysis.

We can create a binary combination of these two images to map out where either the color or gradient thresholds were met.

3 Warped Image

A perspective transform maps the points in a given image to different, desired, image points with a new perspective. Viewing a lane from above (bird's-eye view transform) is useful for calculating the lane curvature. Starting at the base of each line, a sliding window is used to find the pixels belonging to that lane line. This technique is used for both the left and the right lane lines.

Compute the perspective transform, M, given source and destination points:

M = cv2.getPerspectiveTransform(src,dst)

Compute the inverse perspective transform:

Minv = cv2.getPerspectiveTransform(dst,src)

Warp an image using the perspective transform, M:

warped = cv2.warpPerspective(img, M, img_size, flags=cv2.INTER_LINEAR)

4 Draw the region that the car will drive

A histogram is used over the lower half of the image to find the peaks and hence the base of the two lane lines. Starting at the base of each line, a sliding window is used to find the pixels belonging to the left or the right lane line.

The distorted lane lines are warped back onto the original image using the inverse tranformation matrix Minv. And the image is overlapped with the original image using the function

final = weighted_img(lane_lines, image).

Problem

The model behaves badly on the challenge video

- When there are parallel lines and longer lines on the road, the model cannot accurately judge which line is lane.

The lane line has a specified standard, each lane line has a fixed width, and the distance between the lanes is fixed, which may help to find the lane line more accurately.

- When the driving environment of vehicle is changing rapidly, the model can't find pixels belonging to the left or the right lane line accurately.

The model take the average value of the upper level as the middle point of the next stage. When the curvature is too large, the error is large. And when a certain lane line cannot be identified, the model default lane line is a vertical line, the generated lane line deviates from the actual lane line.