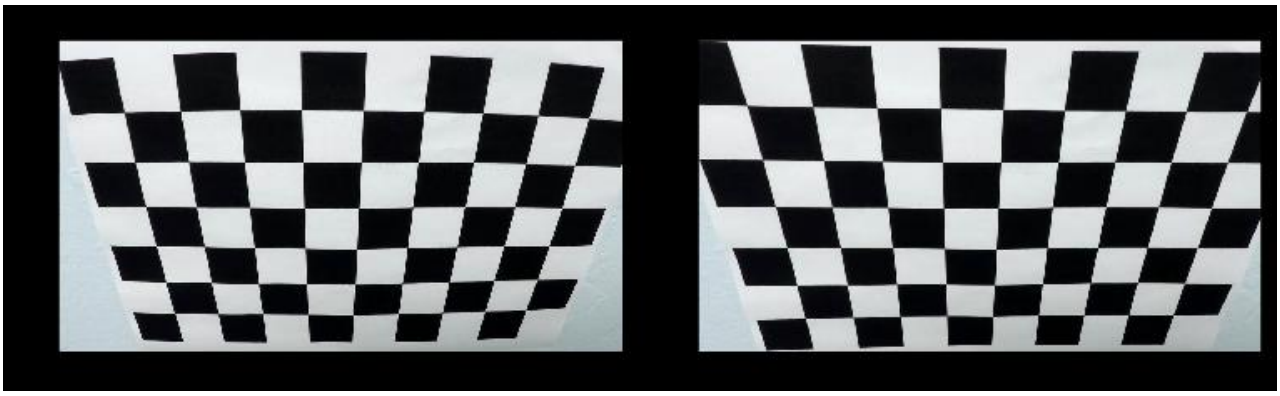
Advanced Lane Finding Project

1. Distortion corrected calibration image.

The code for this step is contained in the calibration.cpp.

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 `calibrateCamera()` function.



Pipeline.

At First, I resized the image and then I converted Frame as Bird view and then I used a combination of color and gradient thresholds to generate a binary image code and then. Here's an example of my output for this step.

Step 1: Undisort Image.

Step 2: Binary Image.

Step 3: I take a **histogram** along all the columns in the lower half of the image and split histogram for two sides for each lane.

Step 4: I used the two highest peaks from our histogram as a starting point for determining where the lane lines are, and then use sliding windows moving upward in the image to determine where the lane lines go.

2. Identified lane-line pixels and fit their positions with a polynomial.

Then I did fit my lane lines with a 2nd order polynomial kind of like this:

3. Radius of curvature of the lane and the position of the vehicle with respect to center.

I found the bottom left and right cordinates and calculated the midpoint of lanes and used the image center as reference to calculate distance away from center.

In the LaneProcessing.cpp

a- AlertSide: Will output 1 or 2 that represent left or right vehicle from center of image.
b- right_curveature: Will output right curvature in meters.

c- left_curvature: Will output left curvature in meters.
d- center_data: Will output the distance between center of lane and Vehicle in meters.