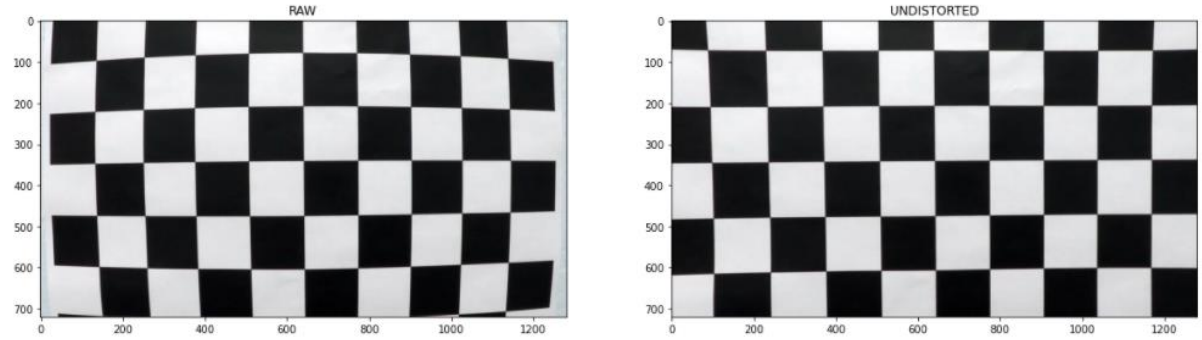
Advanced Lane detection Project Writeup: Rohit Ravikumar

Step 1: Detect the chess board corners by reading the calibration images 1 by 1. Calibrate the camera using objpoints and imgpoints

Step 2: Undistort the chessboard images that did not result in corners in the previous step, using the transformation matrix and distortion co-effecients received from the camera calibration function.

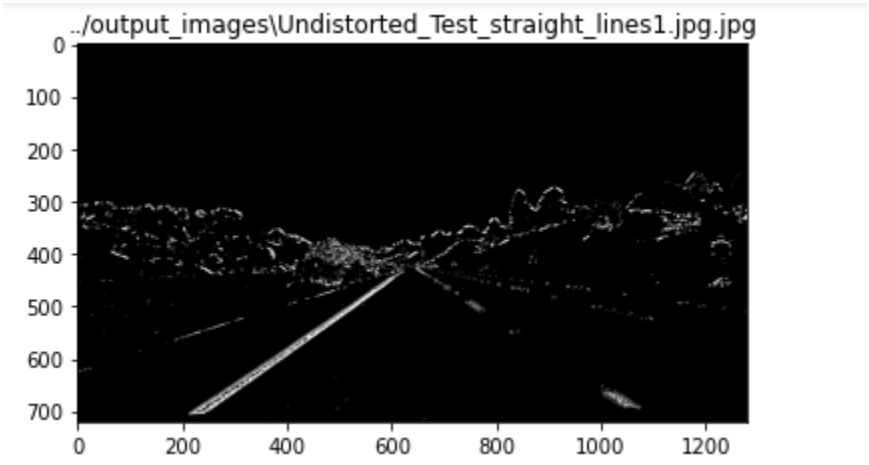
```
Distorted Image: ../camera_cal\calibration1.jpg
Distorted Image: ../camera_cal\calibration4.jpg
Distorted Image: ../camera_cal\calibration5.jpg
```



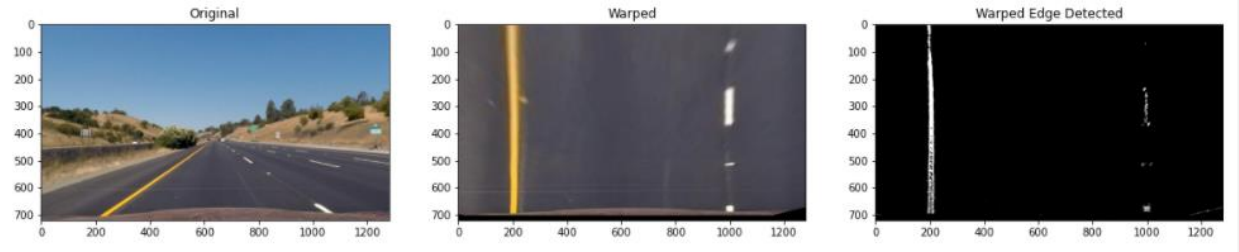
Step 3: Undistort the test images using the transformation matrix and distortion co-efficient received from the camera calibration function and save the undistorted images in the output folder

Step 4: Detect edges of the undistorted images using sobel x, sobel y, soble magnitude, sobel direction and s channel thresholds: return the combined-threshold edge detected grayscale image. Play around with different threshold values till we get a good edge detected image.

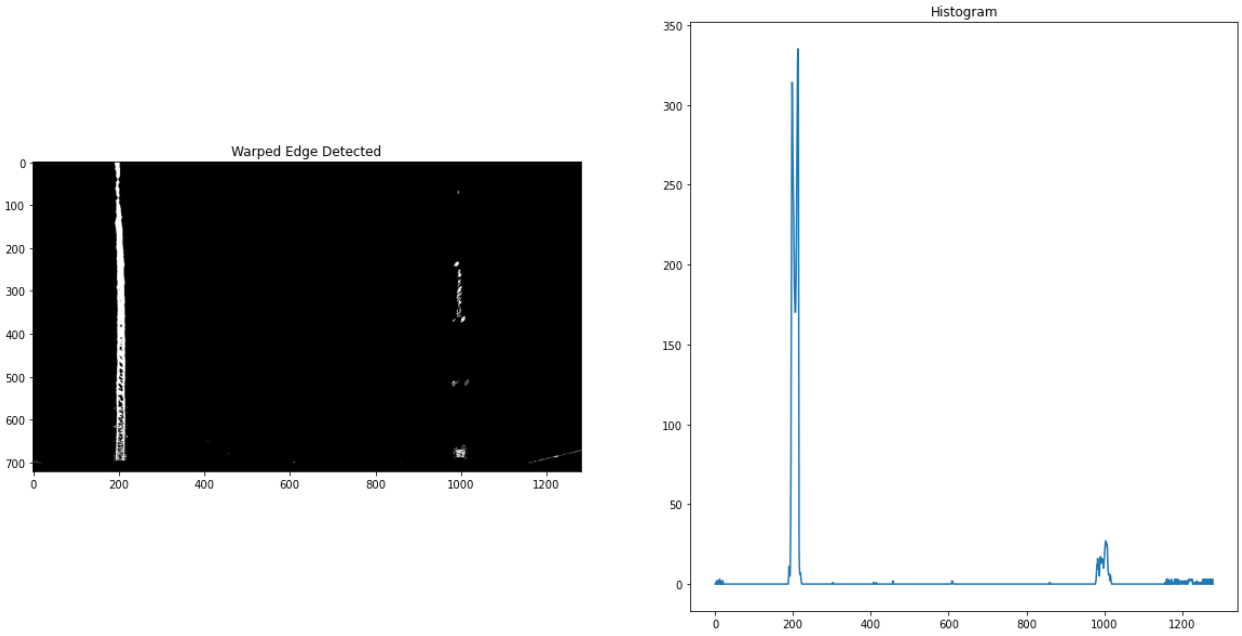
Selected Values: Kernel Size 15, s channel threshold 170-250, direction and magnitude thresholds 07,1.3



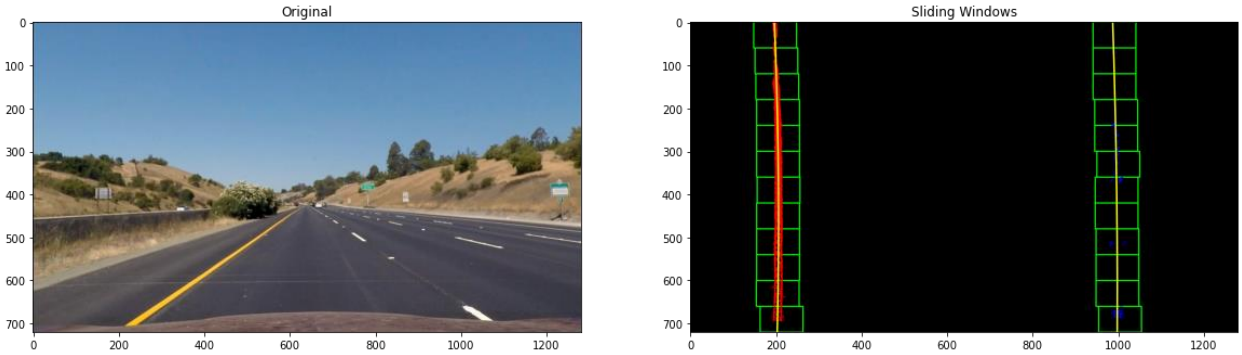
Step 5: Warp the edge detected image to make it an aerial view



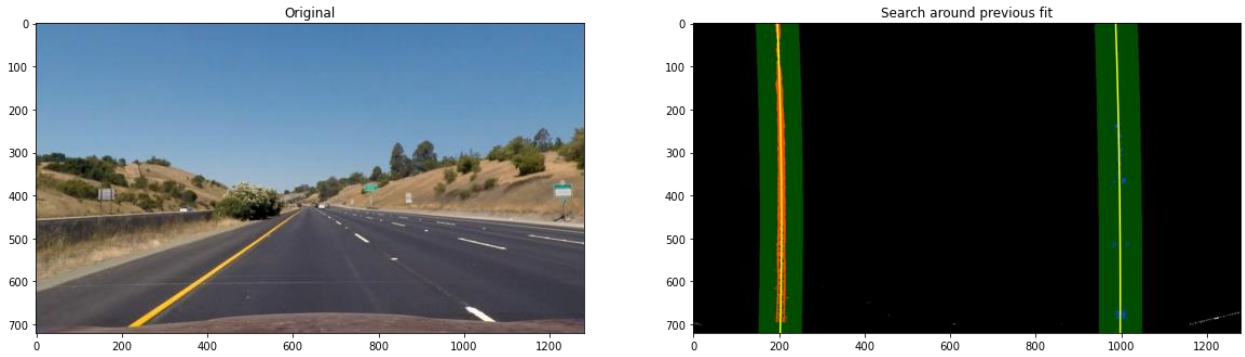
Step 6: Get a histogram of the lower half of the image to see if the starting points of left and right lanes can be detected successfully



Step 7: Use the warped image to determine fit by through the sliding windows function



Step 8: Use the fit as determined from the sliding windows function to find similar points around and improve the fit



Step 9: Measure curvature using the improved fitted lane lines. This function returns the left lane line curvature, right lane lane curvature and distance from the center of the lane

Step 10: Project the lane lines back on the original image and fill the whole region using polyfill function to show the region of interest. Enter the curve radius and offset from the center as text on the images

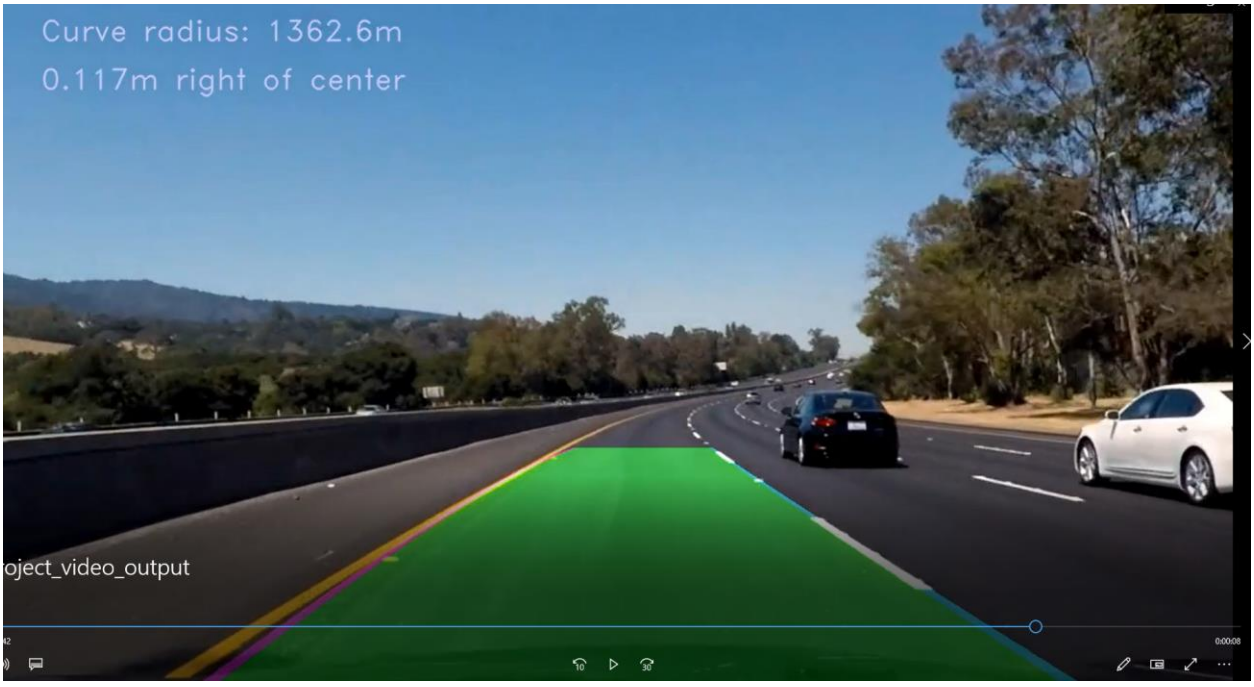
Step 11: Setup image pipeline to undistort ➔ edge detect ➔ warp ➔ sliding windows fit detect ➔ search around initial fit to improve the fit ➔ determine curvature and draw the lane lines back on the original image

Step 12: Run the Image pipeline on the undistorted images and save the images in the output images folder

Step 13: Use the same pipeline on the project video and save the output to the output_images folder

Step 14: Improve the pipeline for video processing undistort ➔ warp ➔ edge detect ➔ sliding windows fit detect ➔ only allow the fit from images based on left lane base position being between 300 and 500 pixels and right lane base position of at least 900 pixels and lane width between 3.4 and 4 m, else use the older known correct fit ➔ search around initial fit and improve the fit ➔ determine curvature and draw the lane lines back

Step 15: Run the video pipeline on the project video and save the output to the output_images folder



Step 16: Run the challenge video and harder challenge video and save output: **code failed to detect lanes completely** ☹

Challenges:

- Auto-selection of points for perspective
- Noise reduction from frame to frame

Improvements:

- Using additional color spaces
- Detecting and isolating shadows effectively
- Detecting different color lines
- Reducing noise when moving frame to frame and adding reliability per frame to discard them