Camera Calibration:

- Calibrate camera by taking a 9x6 chess board, taking 20 pictures around it, and using opencv functions (such as cv2.findChessBoardConers) to locate the chess board corners.

Then, pass the coordinates (the ground truth points np.mgrid[0:9,0:6], and the coordinates obtained by cv2.findChessBoardConers) to cv2.calibrateCamera(ground\_truth, camera\_points, ...) function, which outputs the camera matrix and distortion coefficients. Example of unwarped image can be found in output\_images

Pipeline (found in function ‘find\_lane lines’):  
- First undistort the image based on the distortion coeffs and camera matrix found above

- Warp a manually selected trapezoidal region of interest (found in transform\_roi) to a rectangle, to create the effect of looking at the road from a bird’s eye view.

- Then, apply sobel x gradient and extract the points between a certain thershold, and OR (|) it with values between a certain threshold for the ‘S’ channel after converting the image to HLS space. This creates a binary image, where relevant pixels lane finding are 1 and irrelevant pixels are 0 (found in get\_binary\_image)

- Sum up the binary values found in each column, and take the argmax of left half and right half of image, which represents the starting x coordinate for each lane (found in image\_col\_peaks)

- Starting from those x positions, and the binary image found above, use a sliding window (starting from the bottom, and sliding upwards) for each the l and r line to collect relevant pixel coordinates within the box. To capture curved lane lines, you must slide the window (the window’s x midpoint) horizontally to the avg coordinate position of (window-1)’s found pixels. Alternatively, if there is an already valid polynomial fitted to each line, collected the relevant pixels (=1) within a reasonable distance from the polynomial line.

- After gathering these pixels, run np.polyfit(y,x,2) for each line to determine a 2nd degree polynomial that takes a y coordinate (in the bird’s eye view image) and outputs the corresponding x coordinate for the lane line

-Generate the curve for the polynomial by running it for all y btwn 0 and img.shape[0]

-On a black image (np.zeros(dsize=img.shape)), draw the lines using the function cv2.polylines

-Then, using the source and dst points from the original warp (which was converting the car camera image to bird’s eye view image), warp your image (the image is black except for the lane lines) that you retrieved from last step, except flip source and dst points to retrieve the original car camera image.

- With the image above, which is black aside for the lane lines, and is now in the original perspective, overlay that image with the original camera car image, to get the final result.