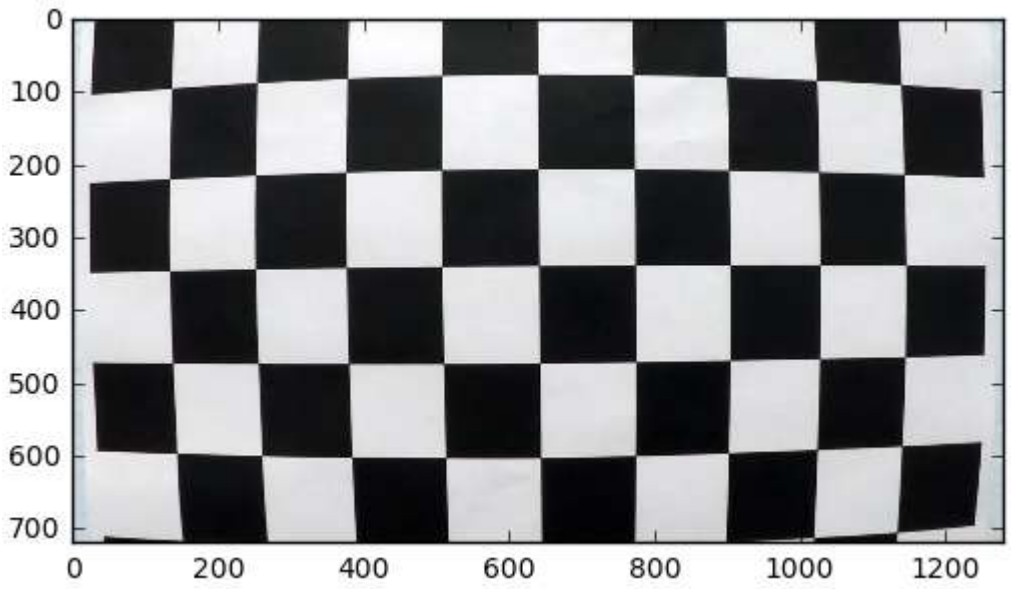
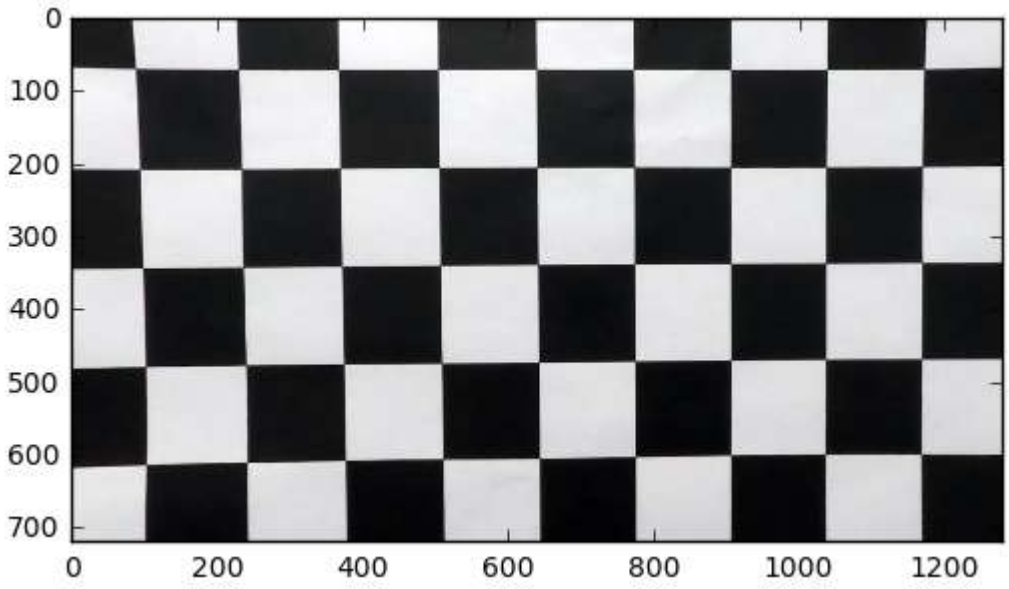
Camera Calibration

Image distortion occurs when a camera looks at 3D objects in the real world and transforms them into a 2D image. The camera calibration is required to correct this distortion.

The calibration process is carried out by comparing distorted corners of a chessboard image and its actual positions.



Distorted Image



Distortion Corrected Image

Pipeline

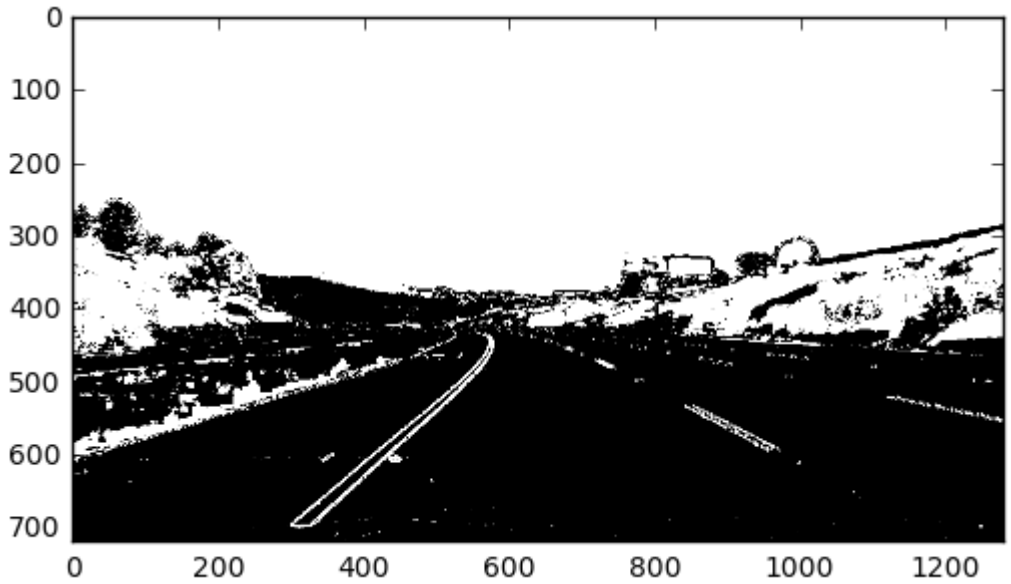
1. Undistort the image

Below is one of the undistorted images.

2 Threshoulded Binary image

Used combination of colour and gradient thresholds to generate a binary image

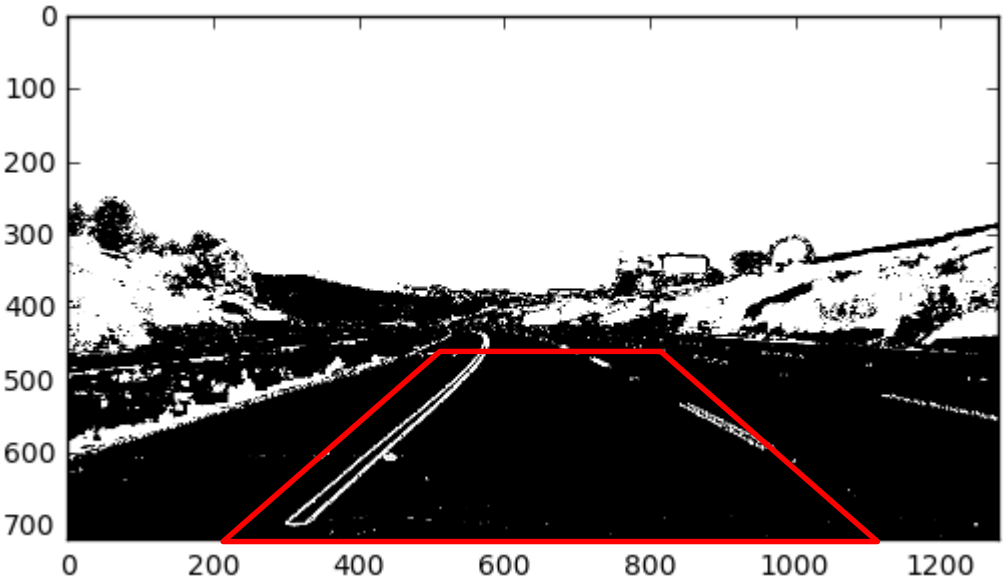
binary = binary_transform(un_deis_img, thresh_min=40, thresh_max=200, ksize=3, thresh=(0.7, 1.3), hls_thresh=(175, 255))



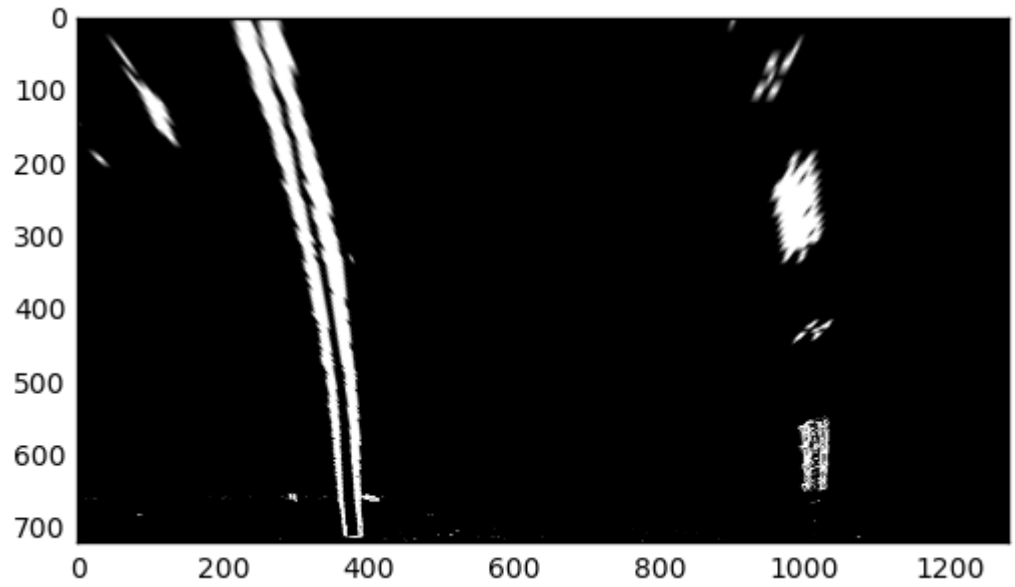
3 Perspective transform

Following Source and Destination points were used for perspective transformation

```
Source                                  Destination
[[  585.           460.        ]        [[ 320.    0.]
[  203.33332825   720.        ]         [ 320.  720.]
[ 1126.66662598   720.        ]         [ 960.  720.]
[  695.           460.        ]]        [ 960.    0.]]
```



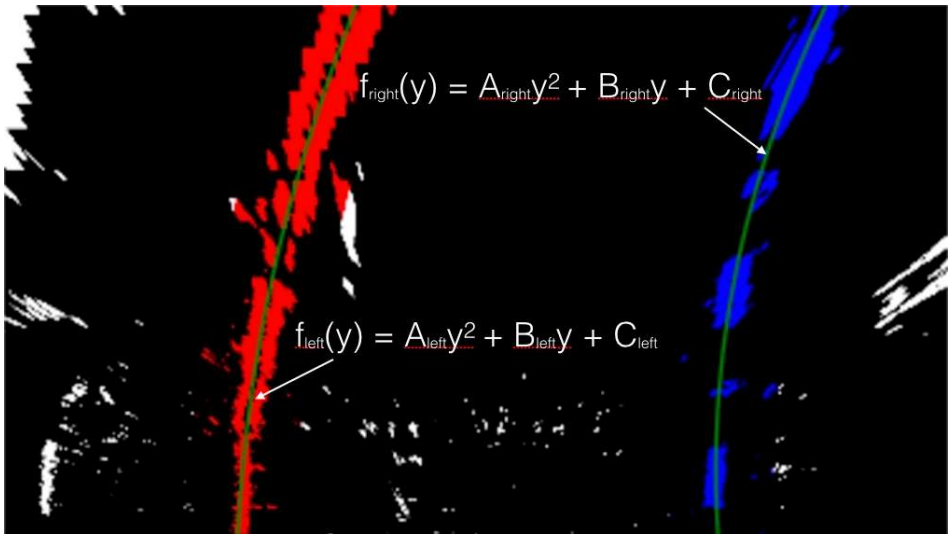
Perspective transformed binary image looks like this



The width of the lanes is about 600pixles

4 Polynomial fit

A second order polynomial was fit to the identified lane line




```
# Define conversions in x and y from pixels space to meters

ym_per_pix = 30 / 720 # meters per pixel in y dimension

xm_per_pix = 3.7 / 600 # meters per pixel in x dimension


    # Fit new polynomials to x,y in world space

left_fit_cr = np.polyfit(lefty * ym_per_pix, leftx * xm_per_pix, 2)

right_fit_cr = np.polyfit(righty * ym_per_pix, rightx * xm_per_pix, 2)
```

5 Calculate the radius of the lane

```
   # Calculate the new radii of curvature

  left_curverad = ((1 + (2*left_fit_cr[0]*y_eval*ym_per_pix + left_fit_cr[1])**2)**1.5)
  / np.absolute(2*left_fit_cr[0])

  right_curverad = ((1 + (2*right_fit_cr[0]*y_eval*ym_per_pix +
  right_fit_cr[1])**2)**1.5) / np.absolute(2*right_fit_cr[0])
```

6 Calculate the offset from the center

```
off_center_deviation = image_center - (rightx_base + leftx_base) / 2
```

Pipeline Video

With this I have attached “video_out_1.mp4” the output of the pipeline

Discussion

This pipeline can be improved by

1. Camera mounting angle and directions may get change from vehicle to vehicle. Therefore perspective transformation has to be done for every camera
2. When lane crossing there can be errors
3. If there is are vehicles in front, this line detection may not work