Advanced Lane Finding Project

Camera Calibration

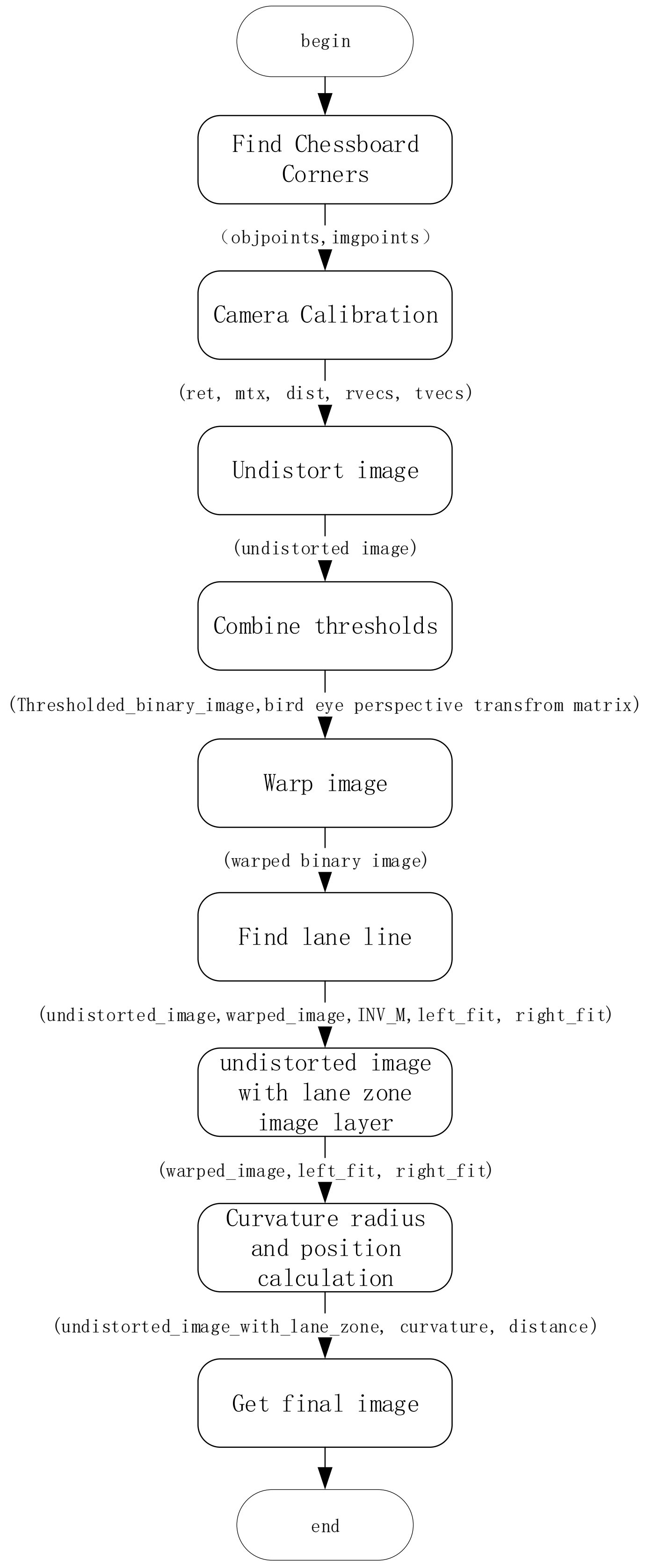
1. Chessboard images processing.

Call function GetObjImagePoints(images,chessboard_grid=(9,6)), get object points and image points.

2.Get camera calibration parameters.

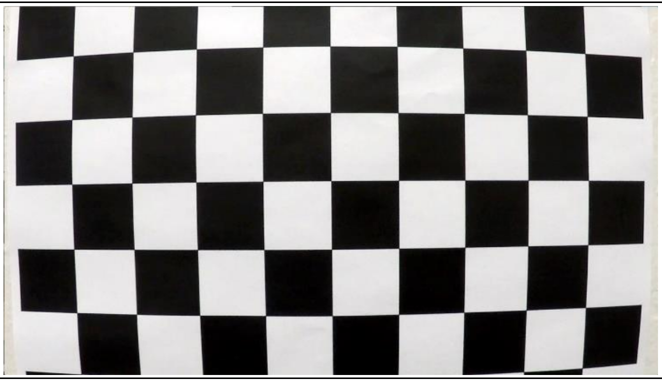
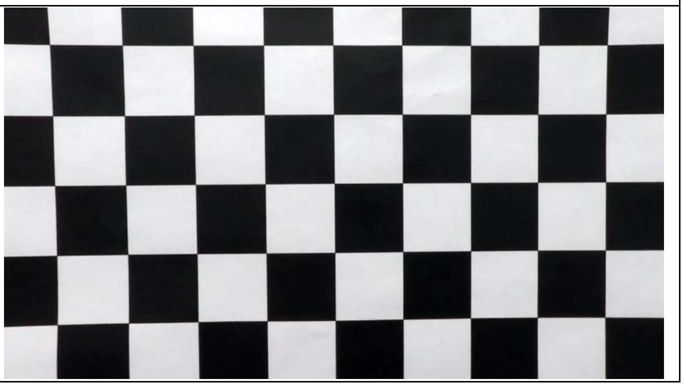
Call function cv2.calibrateCamera(objpoints, imgpoints, image.shape[1::-1], None, None) ,get (ret, mtx, dist, rvecs, tvecs).

Pipeline (single image)


begin
Find Chessboard Corners
（objpoints,imgpoints）
Camera Calibration
(ret, mtx, dist, rvecs, tvecs)
Undistort image
(undistorted image)
Combine thresholds
(Thresholded_binary_image,bird eye perspective transfrom matrix)
Warp image
(warped binary image)
Find lane line
(undistorted_image,warped_image,INV_M,left_fit, right_fit)
undistorted image with lane zone image layer
(warped_image,left_fit, right_fit)
Curvature radius and position calculation
(undistorted_image_with_lane_zone, curvature, distance)
Get final image
end

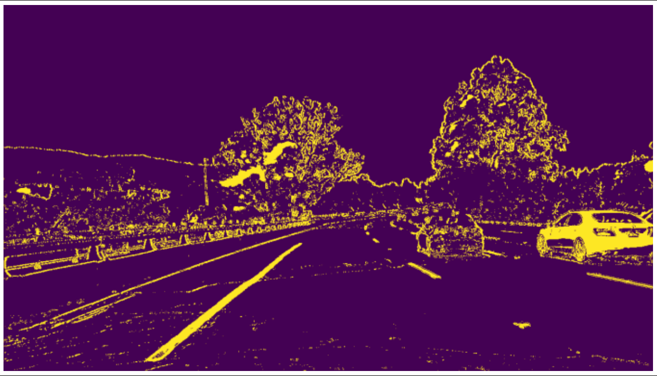
1. Undistort image.

Call function UndistortImage(image, objpoints, imgpoints) get undistorted image.

Original image	Undistorted image
	

2. Combine thresholds.

I use sobel,mag,dir,hls,lab,luv thresholds' combination.

Undistorted image	Thresholed binary image
	

3. Perspective transform.

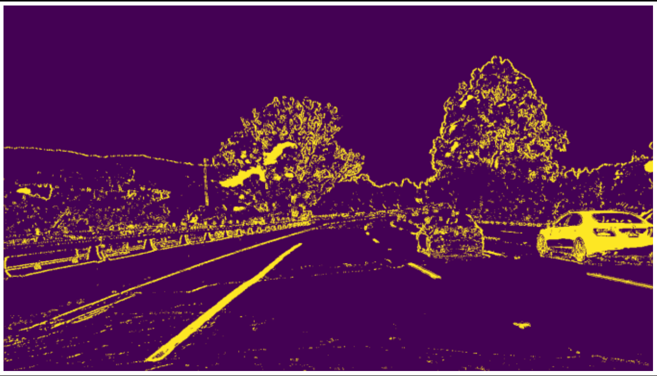
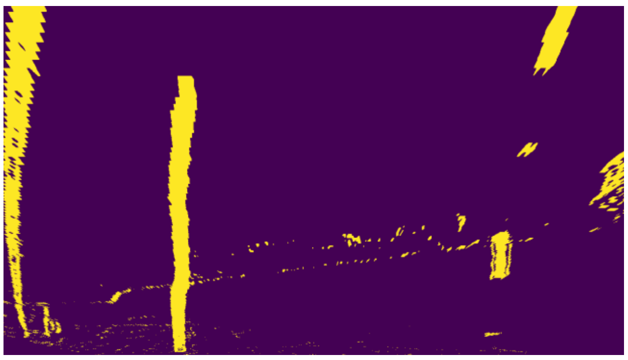
Call function GetPerspectiveTransformMatrix() get perspective transform matrix ,M(src->dst) and INV_M(dst->src).

```
src = np.float32(
    [[(img_size[0] / 2) - 55, img_size[1] / 2 + 100],
    [((img_size[0] / 6) - 10), img_size[1]],
    [(img_size[0] * 5 / 6) + 60, img_size[1]],
    [(img_size[0] / 2 + 55), img_size[1] / 2 + 100]])
dst = np.float32(
    [[(img_size[0] / 4), 0],
    [(img_size[0] / 4), img_size[1]],
    [(img_size[0] * 3 / 4), img_size[1]],
    [(img_size[0] * 3 / 4), 0]])
```

This resulted in the following source and destination points:

Source	Destination
585, 460	320, 0
203, 720	320, 720
1127, 720	960, 720
695, 460	960, 0

I verified that my perspective transform was working as expected by drawing the `src` and `dst` points onto a test image and its warped counterpart to verify that the lines appear parallel in the warped image.

Undistorted image	Thresholed binary image
	

4. Find lane lines.

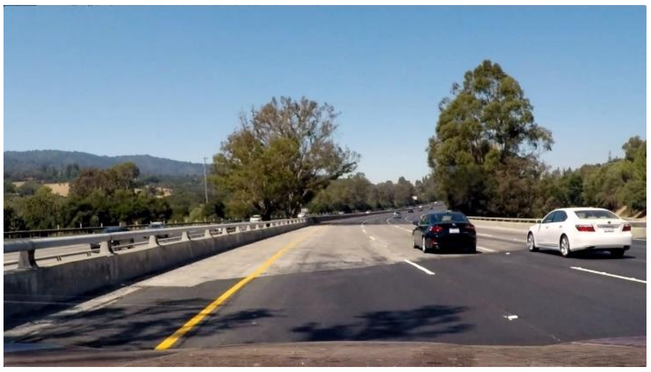
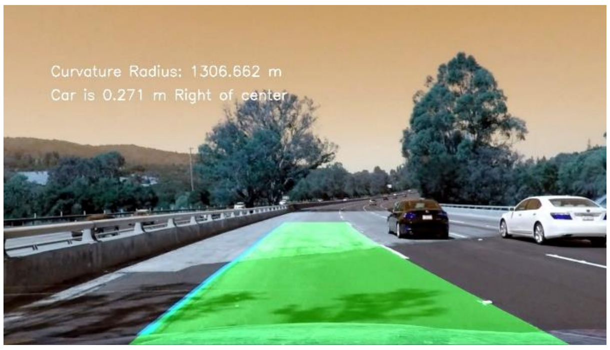
Call FindLaneLine(warped_binary_image) to find lane’s left and right lines,and call DrawLaneZone(undistorted_image,warped_binary_image,inverse_matrix,left_fit, right_fit) draw lane zone color onto undistorted original image.

5. Calculated the radius of curvature of the lane and the position of the vehicle with respect to center.

```
curvature,distance_in_car_and_road_center =
GetCurvatureAndPosition(warped_binary_image,left_fit, right_fit)
```

6. Draw lane zone color onto original undistorted image.

```
Call Pipline(img)
```

Original image	Pipline image
	 Curvature Radius: 1306.662 m Car is 0.271 m Right of center

Pipeline (video)

1.Final video output.

https://github.com/silodiq/self_driving_car_engineer/blob/master/CarND-Advanced-Lane-Lines/output_videos/project_video_output.mp4

Discussion

1. When process video ,conversion speed is so slowly, it can’t be used in real world self driving car.

2. When the light is more complex, the processing is not ideal