Writeup of Thierry Monteil project

1) Advanced Lane Finding Project

The goals / steps of this project are the following:

- Compute the camera calibration matrix and distortion coefficients given a set of chessboard images.
- Apply a distortion correction to raw images.
- Use color transforms, gradients, etc., to create a thresholded binary image.
- Apply a perspective transform to rectify binary image ("birds-eye view").
- Detect lane pixels and fit to find the lane boundary.
- Determine the curvature of the lane and vehicle position with respect to center.
- Warp the detected lane boundaries back onto the original image.
- Output visual display of the lane boundaries and numerical estimation of lane curvature and vehicle position.

2) Writeup

The written program is in the file `find_lines.py`

a) Camera Calibration

I have defined a function called: init_calibration_distorsion. This function uses all the images defined in the directory camera_cal to calculate the distorsion parameters.

I use the code provided during the lesson. As mentioned in the example, “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 `cv2.calibrateCamera()` function. I applied this distortion correction to the test image using the `cv2.undistort()` function and obtained this result (Figure 1)

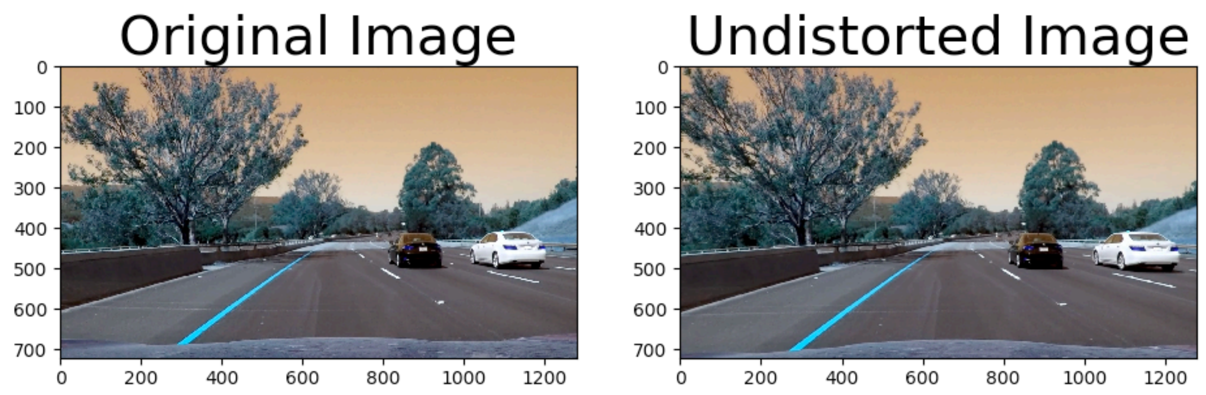

Figure 1: undistorbed image

b) Pipeline (single images)

The pipeline for image processing is made with the class Line. It contains all functions to apply the transformations and to test them.

1. Distortion-corrected image

The call of function previously described to undistorb an image is called by the constructor of the class Line. The parameters are saved in a file and in several attributes of the class (result on figure 1)

2. Thresholded binary image

As explained: “I used a combination of color and gradient thresholds to generate a binary image”. This is done in the method: colour_thresholding of the class Line. Here's an example of my output for this step (figure 2).

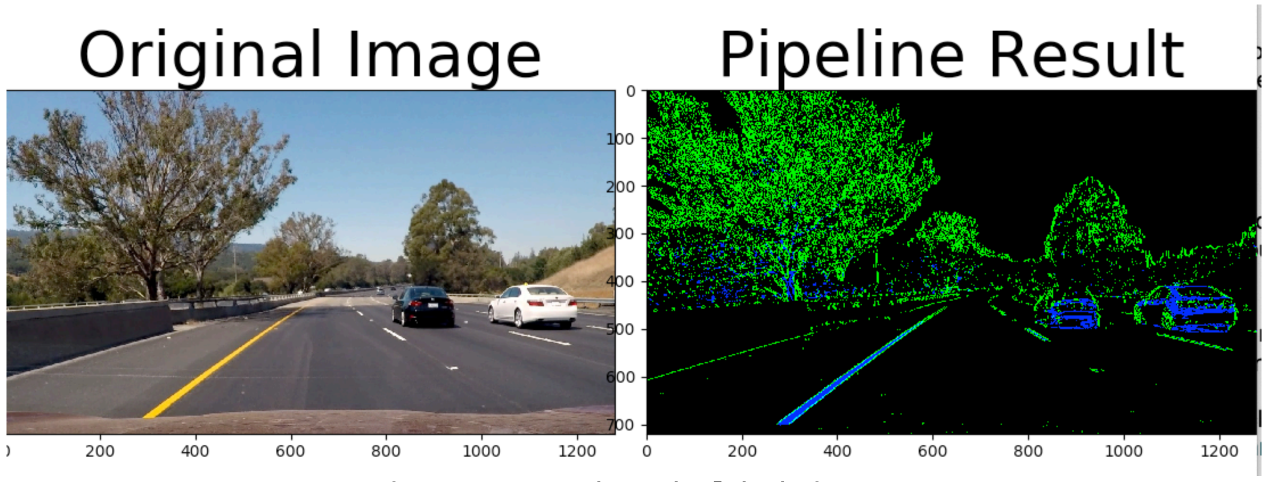

Figure 2 : Thresholded image

3. Perspective transformation

Similary to the exemple, I chose the hardcode the source and destination points in the image file straight_lines1.jpg which correspond to a part of straight lines.
The poins are:
src = np.float32([[268.762,676.967],[1037.94,676.967], [769.189, 505.01],[523.094, 505.01]])
  dst = np.float32([[268.762,676.967],[1037.94,676.967], [1037.94, 505.01],[268.762, 505.01]])

The matrix are calculated in the constructor of the class Lines. I verified that my perspective transform was working as expected on Figure 3.

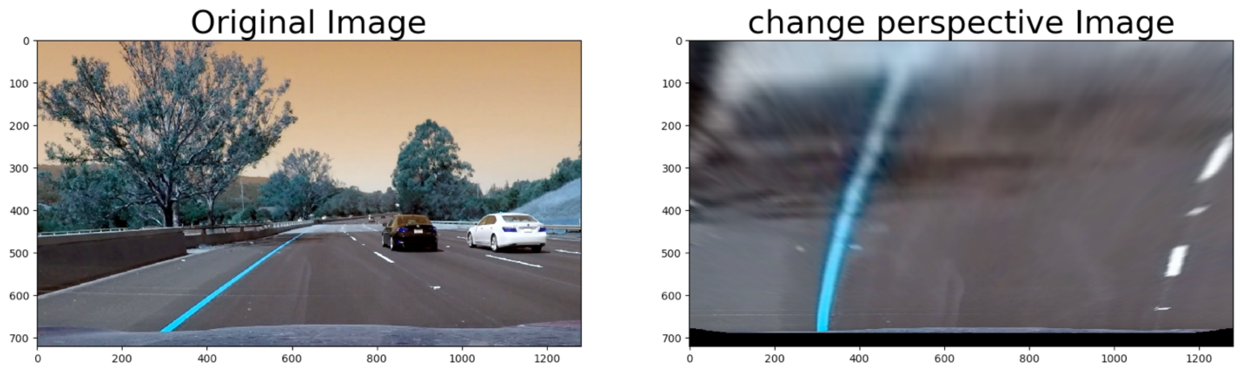

Figure 3: Perspective transformation

4. identification of lane-line pixels and fit their positions with a polynomial

Then I did some other stuff and fit my lane lines with a 2nd order polynomial with the method lines_from_scratch. The result is shown in Figure 4.

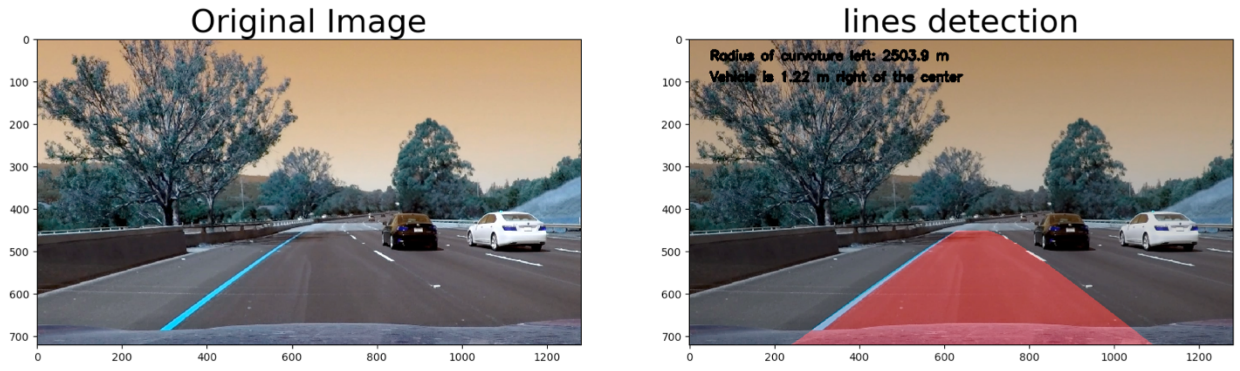
Original Image
lines detection
Radius of curvature left: 2503.9 m
Vehicle is 1.22 m right of the center

Figure 4 : lines and radius detection

5. Radius estimation

The estimation of radius is done in method: measure_curvature_real. We suppose that meters per pixel in y dimension is 30/720 and meters per pixel in x dimension is. Results are shown in Figure 4.

6. <u>Visualisation</u>

The result is plotted back down onto the road such that the lane area is identified clearly with the method: insert_region. I applied the full process for all images of test_images and write the result in the directory “output_images”

c) Pipeline (video)

I test the pipeline on video on the different video in the repository. The results of the processing are written in the directory “output_images”.

3) Discussion

Because, I learn python for this course, I passed a long time on python problems and management of tuple, list and find the good function in python.
On more challenge videos the algorithm fails very often. To make it more robust as suggested in the course, we should use rules like the lines don’t crossed and stay parallel. We should also average results of calculation lines parameter over several images because the difference between several images couldn’t be important. The interpolation with a simple polynome is to simplest when the form of the line is too complicated in a same image, two solutions: perhaps cut the image in different part and do estimation in each part or use a more sophisticated function for approximation.