Advance Lane Tracking

January 17, 2019

1 Advanced Lane Finding Project

The goals / steps of this project are the following:

1. Compute the camera calibration matrix and distortion coefficients given to a set of chessboard images
2. Apply a `distortion correction` to raw images.
3. Then apply `perspective transform` on undistorted image.
4. Then created `binary image` of warped image with a threshold between 200 and 255.
5. Then only considering the part of image which is containg lanes.
6. Detect lane pixels and fit to find the lane boundary.
7. Determine the `curvature` of the lane and vehicle `position` with respect to center.
8. `Warp` the detected lane boundaries `back onto the original image`.
9. Output visual display of the lane boundaries and numerical estimation of lane curvature and vehicle position.

2 1. Camera Calibration

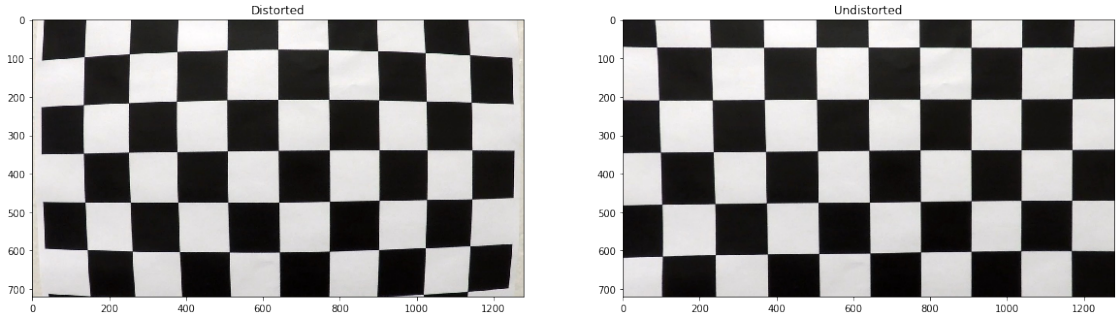
The code for this step is contained in the first code cell of the IPython notebook located in `Advance Lane Finding.ipynb`.

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, `obj_p` 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 in a list named `object_points` and `imgpoints` will be appended with the (x, y) pixel position of each of the corners in the image plane with each successful chessboard detection inside `calibrate_camera` function.
I am storing `object points` and `Image Points` in `obj_points` and `img_points` by calling calibrate_camera with folder named `camera_cal` as this folder contains all the images of chess board required for camera callibration. I then used the output `obj_points` and `img_points` 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:

```
In [47]: f = plt.figure(figsize = (20, 12))
         plt.subplot(1, 2, 1)
```

```
plt.title('Distorted')
plt.imshow(distorted)

plt.subplot(1, 2, 2)
plt.title('Undistorted')
plt.imshow(undistorted);
```



3 Pipeline (single images)

I will be using the below image for describing the pipeline.

```
In [56]: f = plt.figure(figsize=(12, 8))
         plt.imshow(plt.imread("test_images/test2.jpg"));
```

3.1 1. Undistorting an Image

To undistort an image just call `create_undistorted_image(image, obj_points, img_points)` where `obj_points` and `img_points` are the output which we have got from `calibrate_camera('camera_cal')` and here is the undistorted image of the image we are using in this pipeline.

```
In [60]: f = plt.figure(figsize=(12, 8))
         plt.title('undistorted_image')
         plt.imshow(plt.imread("test_images/undistorted_image1.jpg"));
```

3.2 2. Perspective Transform

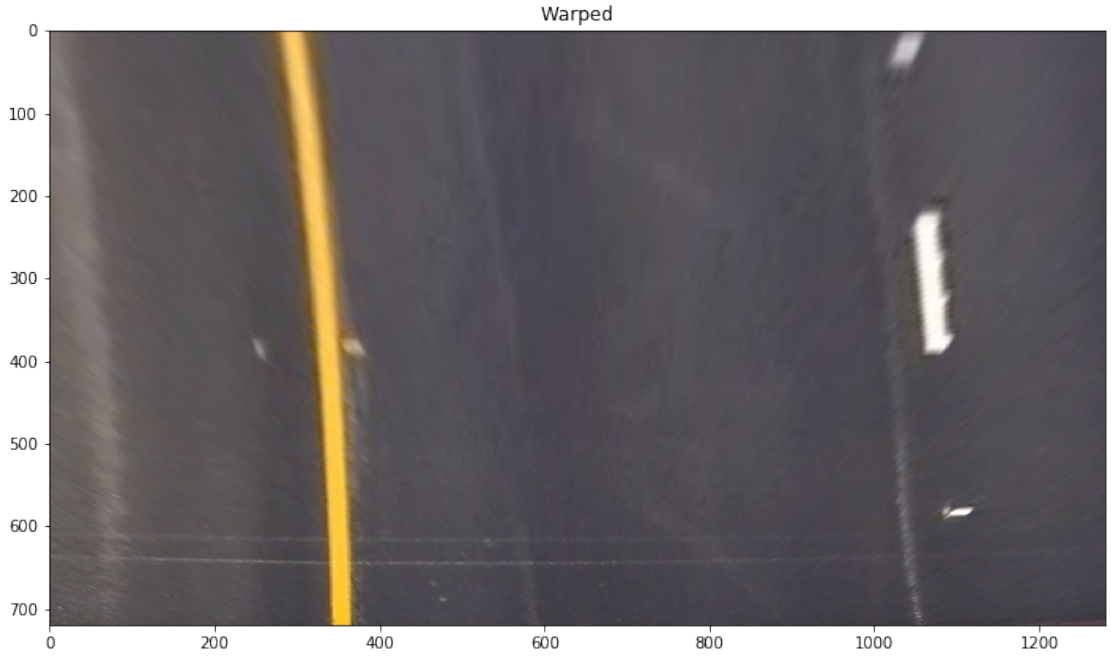
Now, We will apply Perpective transform to get the warped image by calling `warp(img, inv=False)` where `inv = True` will return normal image and `inv=True` will return warped image. Below is the warped image.

I am using the following as aource and destionation points

Source	Destination
384, 665	384, 720
896, 665	896, 720
699, 477	896, 0
581, 477	384, 0

```
In [62]: f=plt.figure(figsize=(12, 8))
```

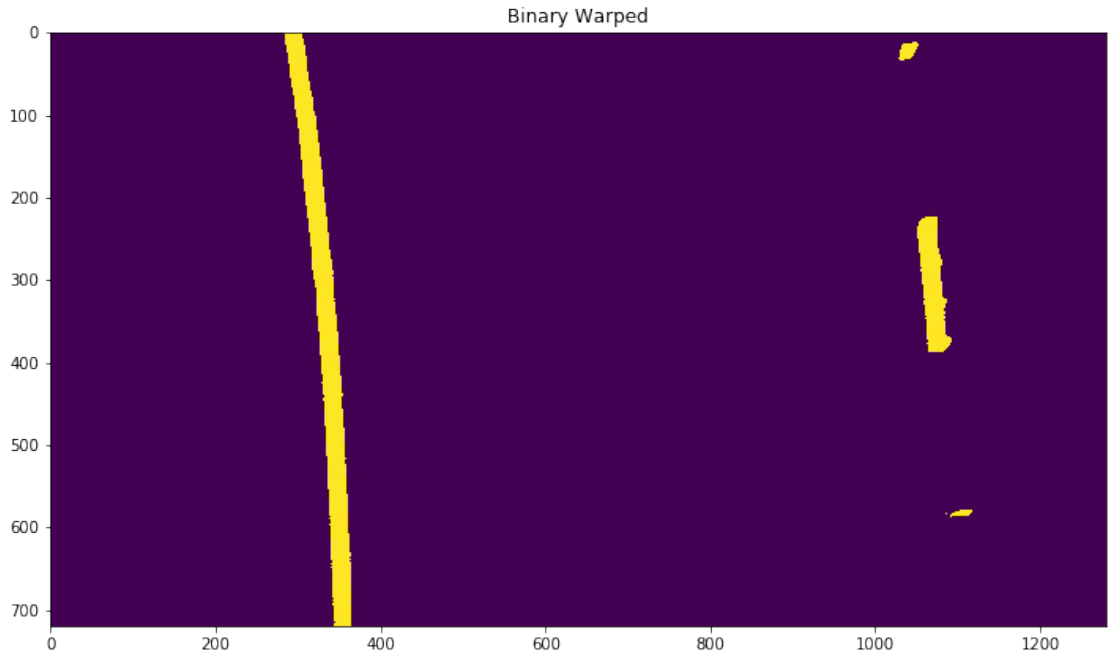
```
plt.title('Warped')
plt.imshow(plt.imread('test_images/warped.jpg'));
```



3.3 3. Create Binary Image

Create a binary Image by first by selecting R channel of RGB image and applying CLAHE algorithm to R channel of Image so it will work in every condition.

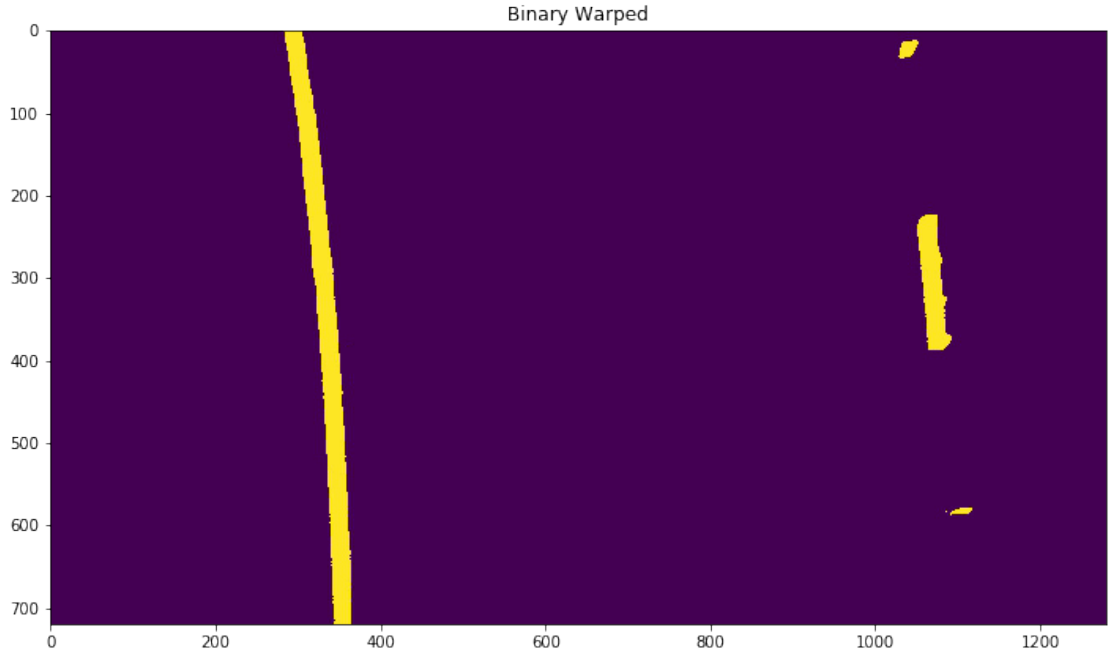
```
In [64]: f = plt.figure(figsize=(12, 8))
         plt.title('Binary Warped')
         plt.imshow(plt.imread("test_images/binary_warped.jpg"));
```



3.4 4. Region of Interest

We are removing the part of Image which is not required in Lane Detection.

```
In [65]: f = plt.figure(figsize=(12, 8))
         plt.title('Binary Warped')
         plt.imshow(plt.imread("test_images/reuired_region.jpg"));
```



3.5 5. Tracking Lane and Frame

We will be using class named `Frame` and `Line` to keep track of `Frames` and `Lines`.And also these will be used for processing the values that we will get while we will process the video and for further `improvement` in lane detection project.

3.5.1 5.1. Line class

In this class I am using three functions 1. best_x: Here I am taking averaged value of x values.

2. valid_coeffiecient: Here we are processing the coefficient based on `polyfit` function using the concept of `mean` and `standard deviation`.

3. Position_from_lane: To detect position of vehicle from centre of image, We are selecting the left_lane_position and right_lane_position from bottom of image and to calculate the center of lane I am using the formula `(right_lane_position - left_lane_position)/2 + left_lane_position` where I am subtracting `left_lane_position` from `right_lane_position` this much length will be common in right_lane_position as well so by subtracting them we got width of lane and dividing by 2 gives us centre of lane and added left_lane_position to get postion of vehicle from the left part of image as we are considering the center part of image as the cener of vehicle.

4. measure_curvature: Stadard Lane width is 3.7 metres so here we are taking and 1m=1/1000km and we are assuming camera is coverting around 30m, so ykm_per_pix = 30/(720*1000) and xkm_per_pix = 3.7/(700*1000). and using polyfit function and radius of curvature formuala to calculate the radius of curvature of particular line.

3.5.2 5.2. Frame Class

I have created this class to keep track of frames and for further improvemnet in this project.

3.6 6. Detecting Lane Lines

We are detecting lane lines In Two steps:

3.6.1 6.1 Using Slidow window approach

Currently we are using this approach for first frame only, In this case first we are using concept of `histogram` means suming all the pixel value column wise of half of image and the divide the image in two part column wise. And select `max` value of `both the side` which will be the `base position of lanes`. and for the right part add the midpoint value that is the the point at which we devided the array to get exact position in image.

And in window based approach we will detect the lanes based on number of windows means we will create a retangle for both side of lanes and we will detect lanes based on minimum pixel value means we will be moving the window based on margin value and select the coordinates in which we find the length pixel indices to be greater than minpix.

And based on the coordinate we got we will select non zero coordinate value for both lanes and then use polyfit function to find the coefficient.

3.6.2 6.2 Finding lines based on ayˆ2+ by + c

In this we will be using the coefficient which we got from `Sliding Window Approach` and detect the nonzero pixel value based on the formula `ayˆ2 + by + c` and caluculate `left_line x and y_value`, `right_line x and y value` and further process to get `left_x`, `right_x` and `ploty` values to fill the green color between the lanes.

3.6.3 7. After doing all above steps and using draw() function we get the following output value.

```
In [66]: f = plt.figure(figsize=(12, 8))
         plt.title('Pipeline Output')
         plt.imshow(plt.imread("test_images/test_image_output.jpg"));
```

4 Project Video Output

4.1 1. Check Out Video with name `project_video_output.mp4` in home folder.

4.2 2. Check Out Video with name `challenge_video_output.mp4` in home folder.

5 Discussion

5.1 Problem faced and how I solved it.

In this project I was able to detect lanes in most part of video but when there is shadow in video it was causing issues specially in challenge video, So to overcome it I used `CLAHE` algorithm and in while tracking line I was averaging the x values including last n frame.And also using concept of mean and standard deviation to select only coefficient which are falling in the particular range based on standard deviation which I defined in `Line` class or else use the mean of last `n` frame coeffiecient

5.2 Improvements required

I tried many ways but still I am having issues in `harder_video_challenge.mp4` and I went through a paper known as `Lane Tracking using B-snake` and I think we can use this concept to track lanes better but still we can face issues, So I think best way is using `deep learning` approach as I have read it is best way for tracking lanes as it overcome lot of manual problem which we are not able to detect.