## Advanced Lane Finding

[![Udacity - Self-Driving Car NanoDegree](https://s3.amazonaws.com/udacity-sdc/github/shield-carnd.svg)](http://www.udacity.com/drive)

In this project, your goal is to **write a software pipeline** to identify the lane boundaries in a video, but the **main output or product we want you to create is a detailed writeup of the project**. Check out the [writeup template](https://github.com/udacity/CarND-Advanced-Lane-Lines/blob/master/writeup\_template.md) for this project and use it as a starting point for creating your own writeup.

Creating a great writeup:

---

A great writeup should **include the rubric points** as well as your description of how you addressed each point. You should include **a detailed description of the code used in each step** (with line-number references and code snippets where necessary), **and links to other supporting documents** or external references. You should **include images** in your writeup to demonstrate how your code works with examples.

All that said, please be concise! We're not looking for you to write a book here, just a brief description of how you passed each rubric point, and references to the relevant code :).

You're not required to use markdown for your writeup. If you use another method please just submit a pdf of your writeup.

The 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.

The images for camera calibration are stored in the folder called `camera\_cal`. The images in `test\_images` are for testing your pipeline on single frames. If you want to extract more test images from the videos, you can simply use an image writing method like `cv2.imwrite()`, i.e., you can read the video in frame by frame as usual, and for frames you want to save for later you can write to an image file.

To help the reviewer examine your work, **please save examples of the output from each stage of your pipeline in the folder called `output\_images`, and include a description in your writeup for the project of what each image shows.**  The video called `project\_video.mp4` is the video your pipeline should work well on.

The `challenge\_video.mp4` video is an extra (and optional) challenge for you if you want to test your pipeline under somewhat trickier conditions. The `harder\_challenge.mp4` video is another optional challenge and is brutal!

If you're feeling ambitious (again, totally optional though), don't stop there! We encourage you to go out and take video of your own, calibrate your camera and show us how you would implement this project from scratch!

## How to write a README

A well written README file can enhance your project and portfolio. Develop your abilities to create professional README files by completing [this free course](https://www.udacity.com/course/writing-readmes--ud777).