# Finding Edges and Custom Kernels

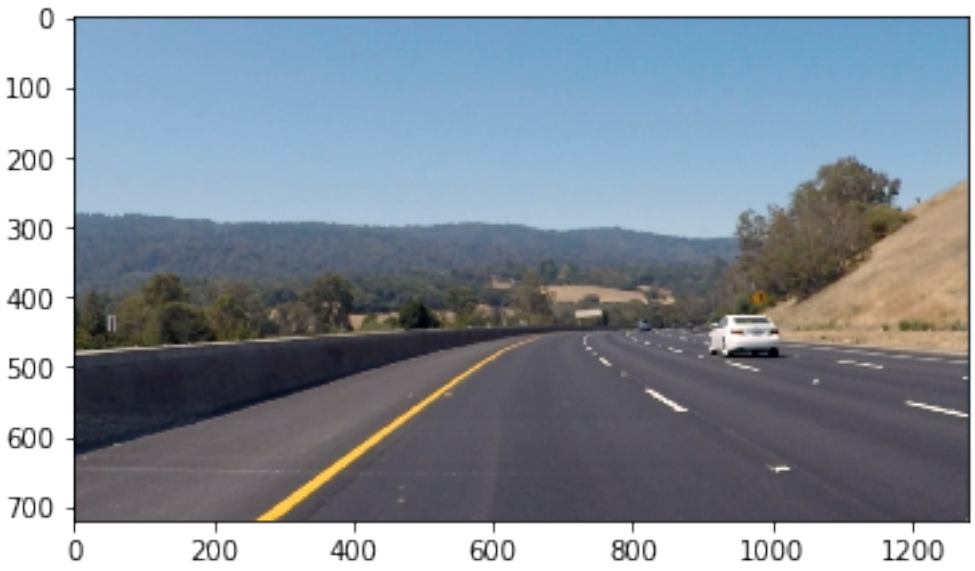
April 27, 2018

## 1 Creating a Filter, Edge Detection

### 1.0.1 Import resources and display image

In [1]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

import cv2
import numpy as np

%matplotlib inline

# Read in the image
image = mpimg.imread('images/curved_lane.jpg')

plt.imshow(image)
```

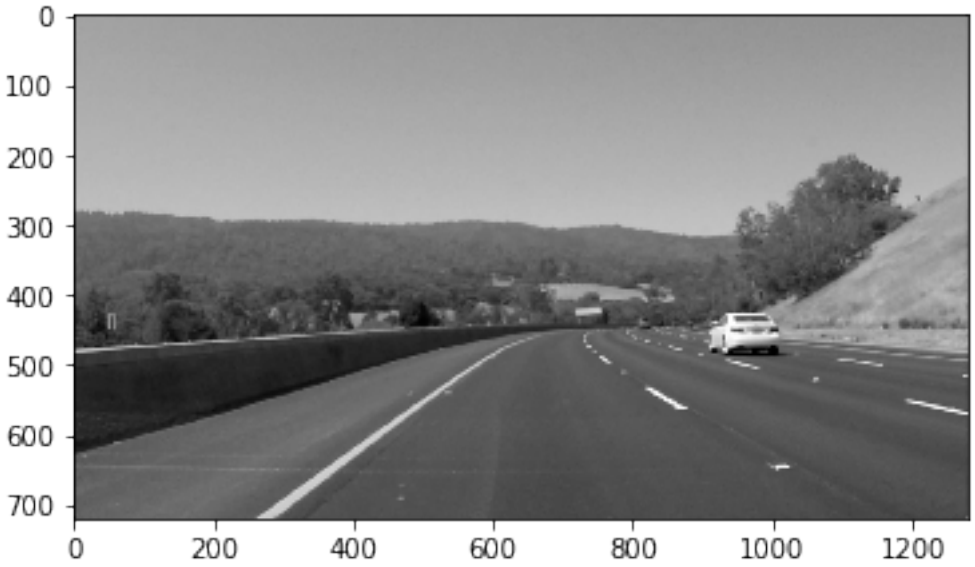
Out[1]: <matplotlib.image.AxesImage at 0x7fcc0b2afdd8>



### 1.0.2 Convert the image to grayscale

In [2]:
```
# Convert to grayscale for filtering
gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)

plt.imshow(gray, cmap='gray')
```

Out[2]: <matplotlib.image.AxesImage at 0x7fcc0b1dccc0>




### 1.0.3 TODO: Create a custom kernel

Below, you've been given one common type of edge detection filter: a Sobel operator.

The Sobel filter is very commonly used in edge detection and in finding patterns in intensity in an image. Applying a Sobel filter to an image is a way of **taking (an approximation) of the derivative of the image** in the x or y direction, separately. The operators look as follows.

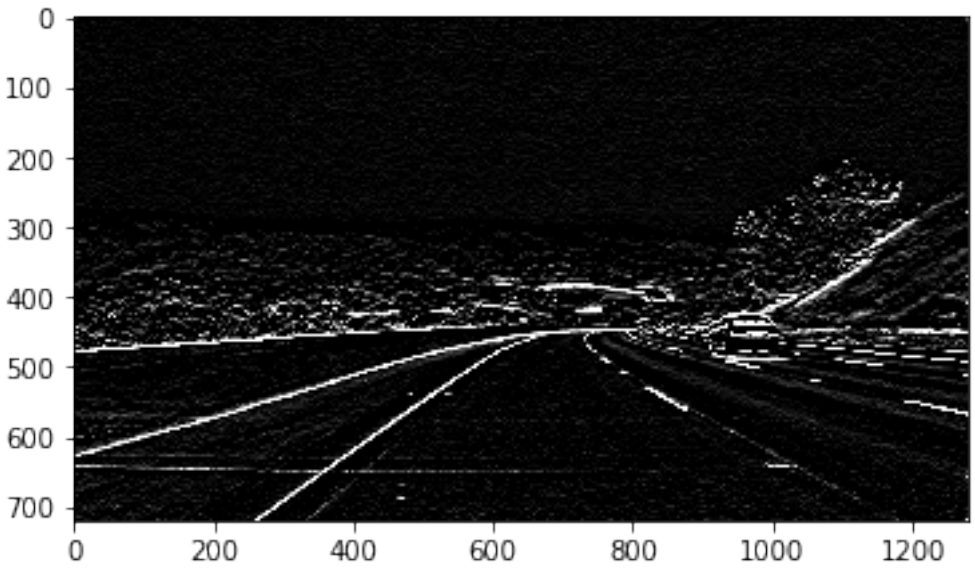
**It's up to you to create a Sobel x operator and apply it to the given image.**

For a challenge, see if you can put the image through a series of filters: first one that blurs the image (takes an average of pixels), and then one that detects the edges.

In [10]:
```
# Create a custom kernel

# 3x3 array for edge detection
sobel_y = np.array([[ -1, -2, -1],
                    [ 0, 0, 0],
                    [ 1, 2, 1]])

## TODO: Create and apply a Sobel x operator
sobel_x = np.array([[-1,-2,-1],[-2,-3,-2],[0,0,0],[2,3,2],[1,2,1]])
```

```
# Filter the image using filter2D, which has inputs: (grayscale image, bit-depth, kerne
filtered_image = cv2.filter2D(gray, -1, sobel_x)

plt.imshow(filtered_image, cmap='gray')
```

```
Out[10]: <matplotlib.image.AxesImage at 0x7fcc084295f8>
```




### 1.0.4 Test out other filters!

You're encouraged to create other kinds of filters and apply them to see what happens! As an **optional exercise**, try the following: * Create a filter with decimal value weights. * Create a 5x5 filter * Apply your filters to the other images in the `images` directory.