**Finding Lane Lines on the Road**

How it works:

My pipeline consider of 6 steps:

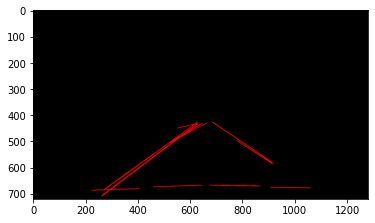
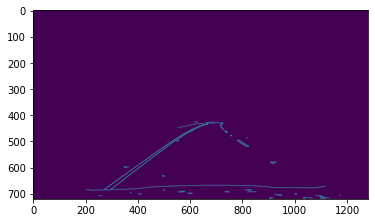
1. Convert to gray scale.
2. Filtering the image.
3. Detecting the edges.
4. Selecting interest zone from the edges.
5. Detecting lines.
6. Filtering and combining detected sub lines.

The step 6 has some configurations:

* min\_abs\_cof\_a # any detected line of slot < then this value will be eliminated.
* max\_cof\_a\_diff # max allowed slot difference to combine lines.
* max\_cof\_b\_diff # max allowed offset difference to combine lines.

The pictures bellow shows the results for the steps:





Potential shortcomings:

What could happen when image is too much noisy for example because of rain?

Possible improvements:

* Only processing image in zone of interest (From beginning copy just the needed zone to process from the image)🡺 This make execution faster and decrease needed memory.
* In gay scale street yellow lanes are too much close to street gray color so edge detection fails to detect them. 🡺 Detecting them in RGB image and highlighting them in gray scale image so they can be detected by canny detection.
* Step 6 can be improved by collecting lines that had a gap less then x value.
* Step 6 can be improved by only consider to showing 2 lines that are the longest if more than two lines had been produced.