Advanced Lane Finding

The objective of this project is to detect lanes and keep the car on the track. This video was taken from a camera mounted on the front of a car.

Steps/Theory

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.

Files in this repo

- `AdvancedLaneFinding_Prj4.ipynb`: Jupyter Notebook with final model
- `README.pdf`: This doc!
- 'project_output.mp4': Final Output
- `harder_challenge_video_output.mp4`: Final output for harder road

Lane finding Procedure

Steps for Lane finding

1. Get image with clear lane lines. (after calibration, thresholding, perspective transformation)
2. Divide image into half from middle, assuming car is in within lane. Left half image will have left lane and right half image will right lane.
3. Identify starting location of left lane in left half of image and right lane in right half of image.
4. After identifying starting points, we will use “sliding window” technique identify lane across the image, moving bottom to top.
5. We will use 2 order polynomial to draw a line across this predicted lane pixel points.
6. For error correction and removing outline pixels,

   a. For first frame, we will discard pixel as lane pixel, if pixel lane is more than 60 pixel away from previous identified pixel. Reason being we are assuming lane will be relatively straight so angle between current and previous prediction will be relatively less. Pixel is more than 60 we will use previous predicted location.

   b. For following frames for treatment of error propagation,

      i. Adjust coefficient. For that we use the standard deviations. If deviation is more than 0.0005
      we are using previous image frame coefficient.

      ii. After image thresholding (with binary image mask) with we are making sure pixel count is more than 5 (to remove noise) (after filtering pixels with strength less 0.5)

   c. For smoothing we used a moving average filter, it smooths data by replacing each data point with the average of the neighboring data points defined within the span. This process is equivalent to lowpass filtering with the response of the smoothing given by the difference equation

   https://www.mathworks.com/help/curvefit/smoothing-data.html (https://www.mathworks.com/help/curvefit/smoothing-data.html)

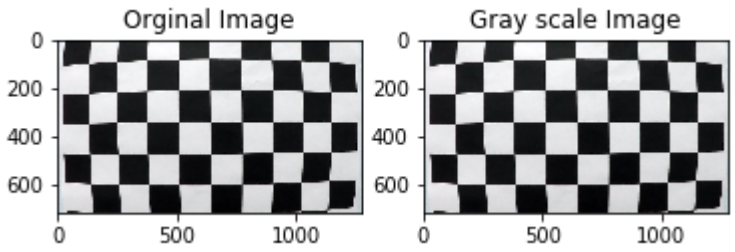
Image processing Steps

1. Correct Image
2. Corp region of interest
3. Do Perspective transformation
4. Apply color filter (white and yellow) (on Perspective transformation)
5. Apply Sobel filter (on Perspective transformation)
6. Combined filter (color and sobel filtered image)

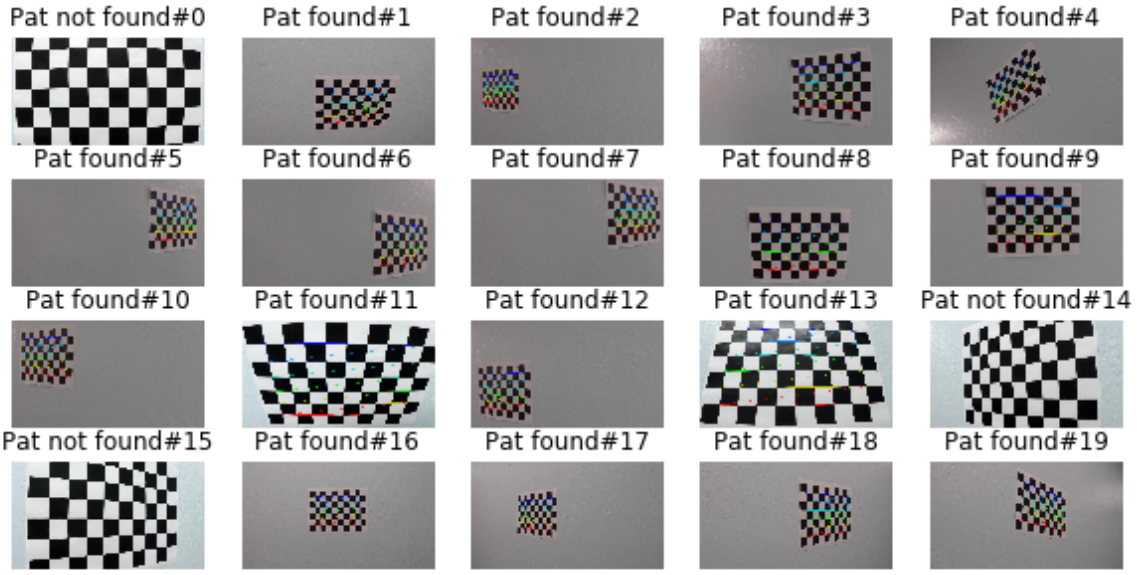
Import Images for calibration

In [2]:

```
Count of images  20
Shape of image  (720, 1280, 3)
```



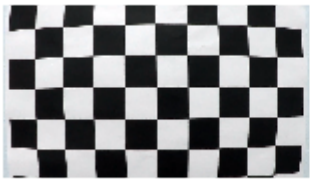
In [3]:



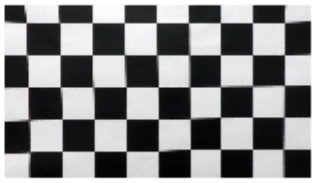
Distortion correction to the raw image

In [4]:

ORG



UNDIST



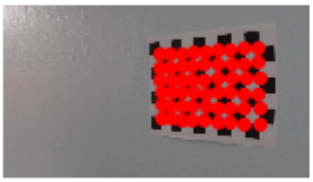
ORG

UNDIST

ORG

UNDIST

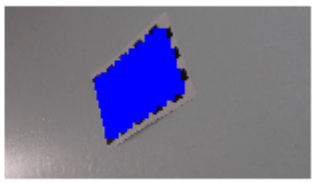
ORG



UNDIST

ORG

UNDIST



ORG

UNDIST

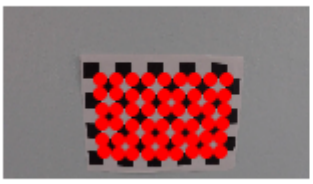
ORG

UNDIST

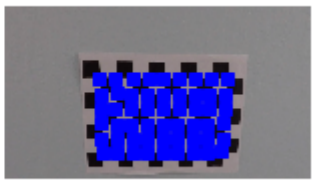
ORG

UNDIST

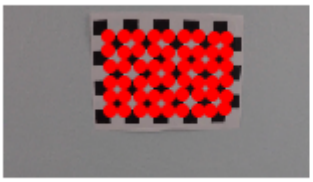
ORG



UNDIST



ORG



UNDIST

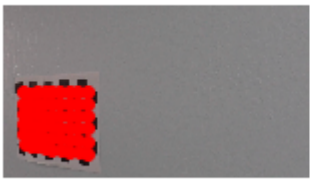
ORG

UNDIST

ORG

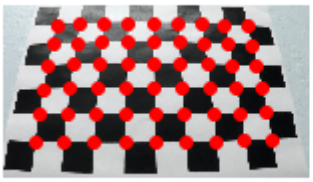
UNDIST

ORG



UNDIST

ORG



UNDIST

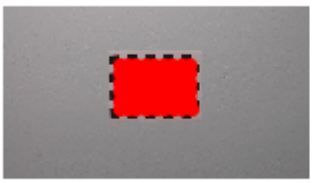
ORG

UNDIST

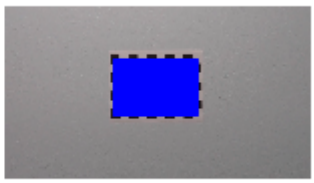
ORG

UNDIST

ORG

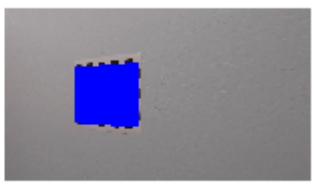


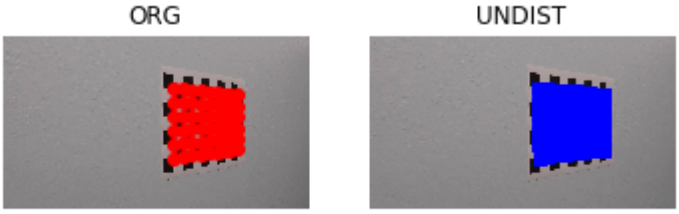
UNDIST

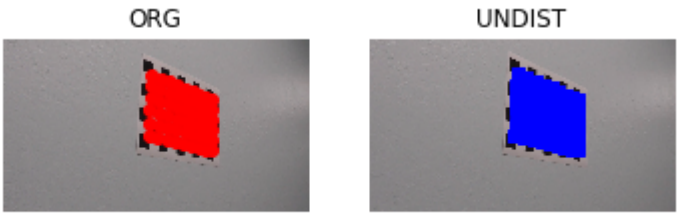


ORG

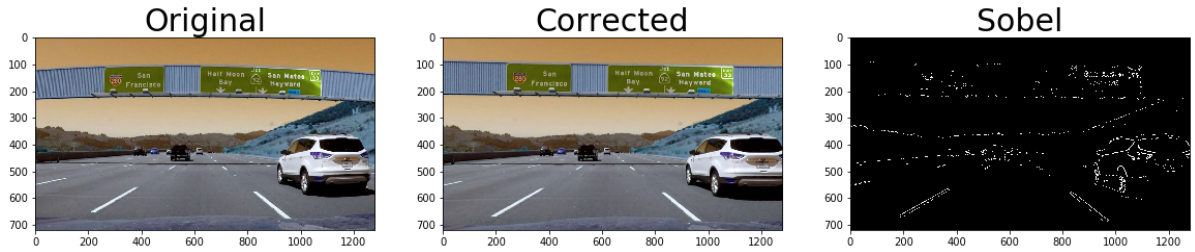
UNDIST



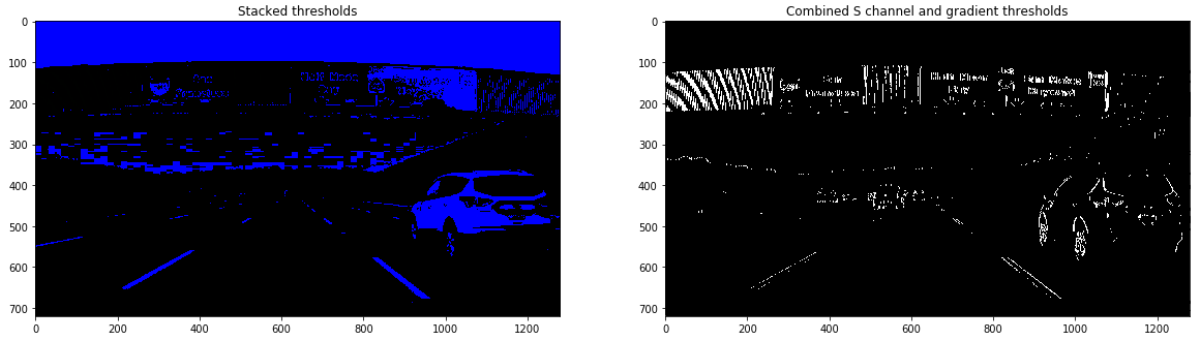




In [6]:

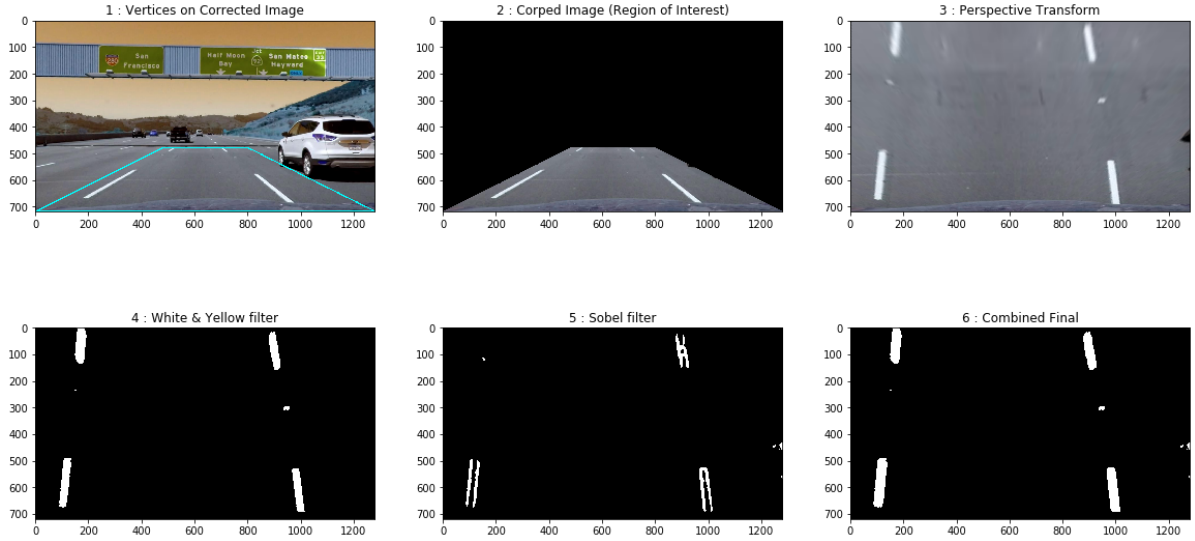


In [7]:

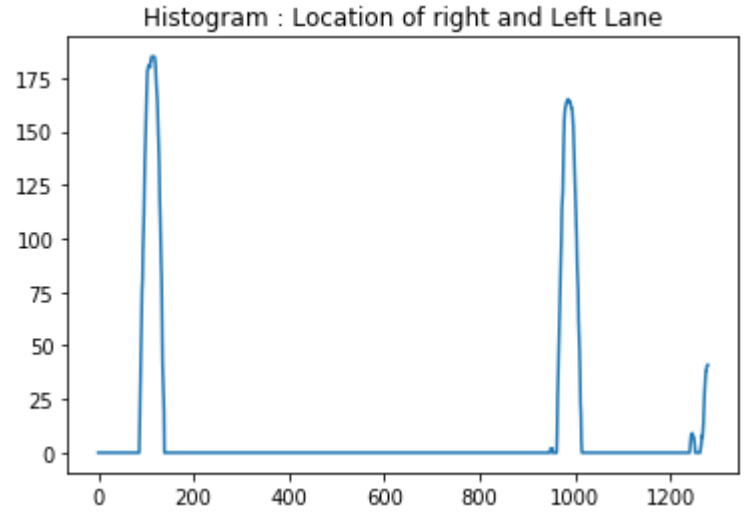


Perspective transform and (then Apply color and sobel filter)

In [8]:

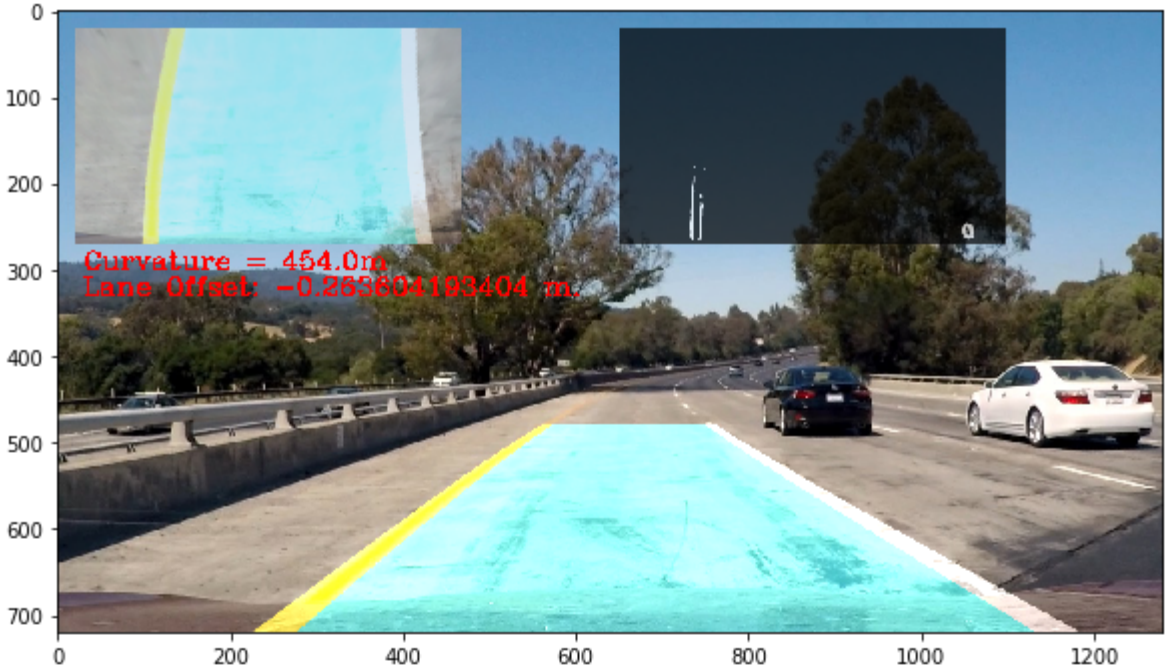


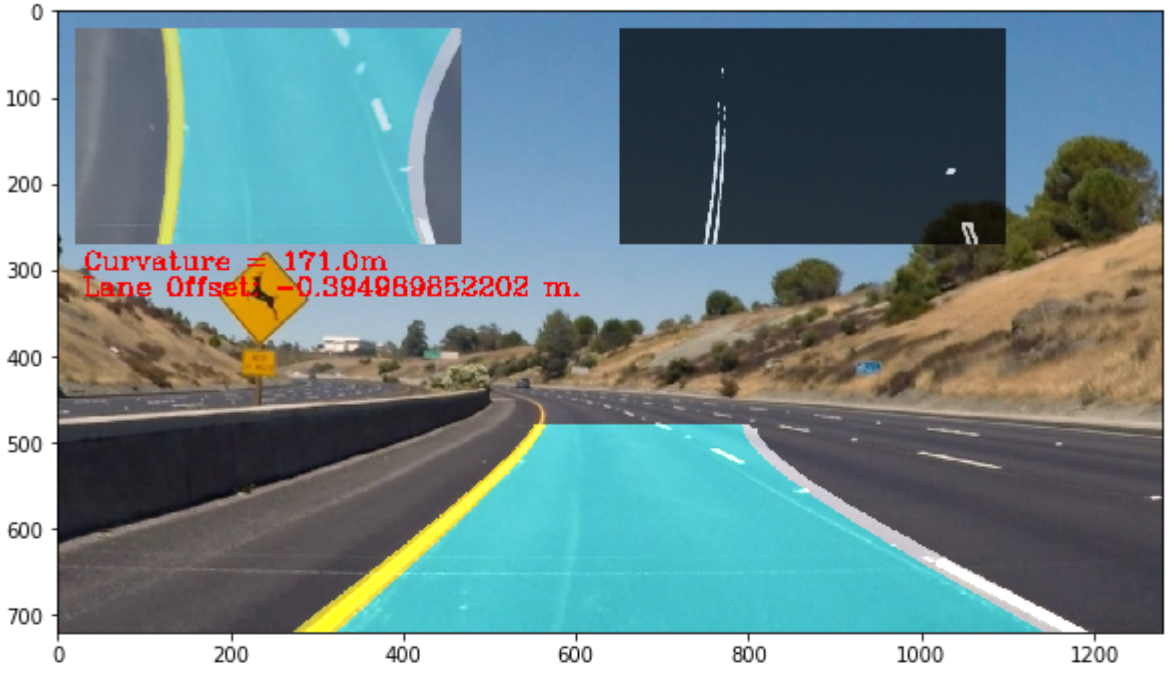
In [9]:



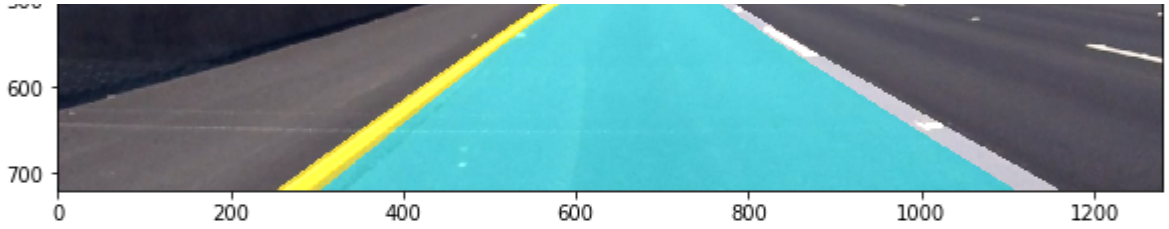
Test Image pipeline on single Image

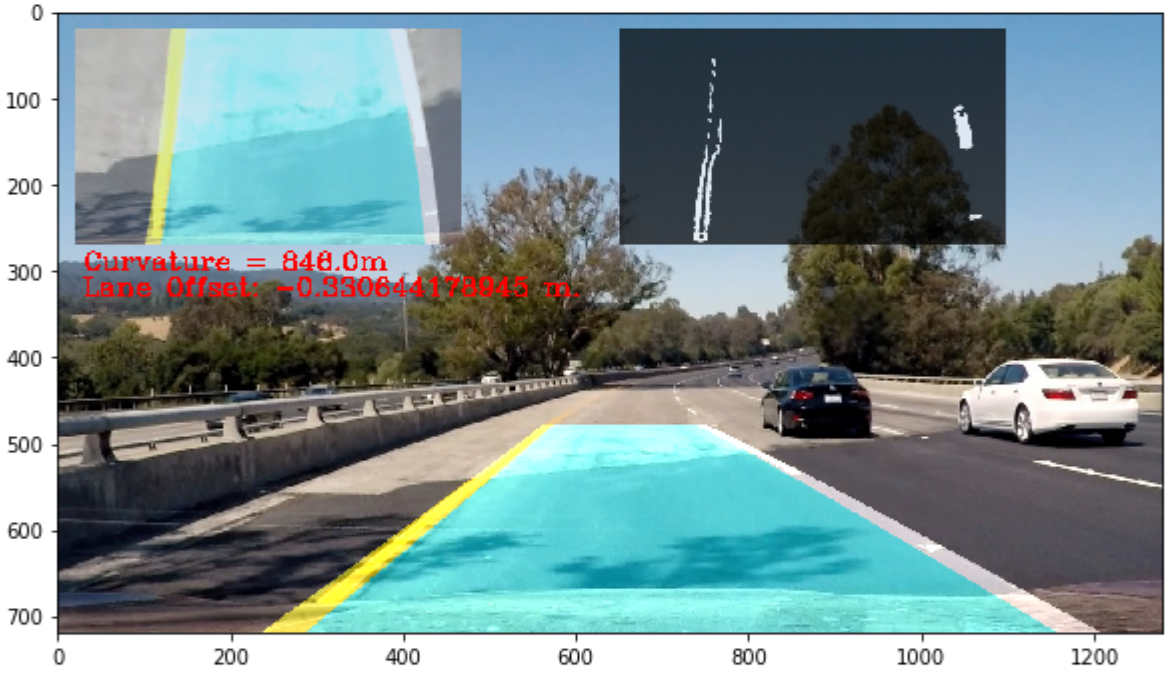
In [14]:

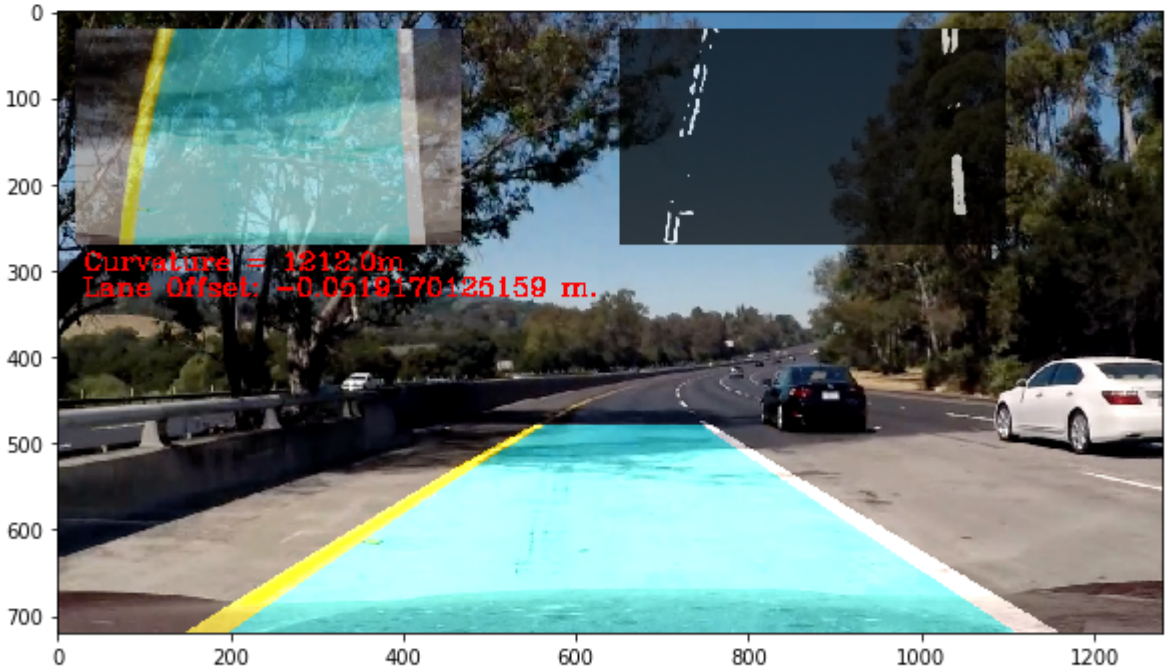

Curvature = 454.0m
Lane Offset: -0.263604183404 m.

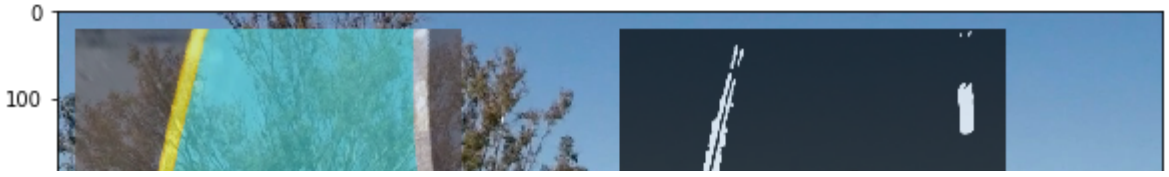

Curvature = 171.0m
Lane Offset: -0.394989852202 m.

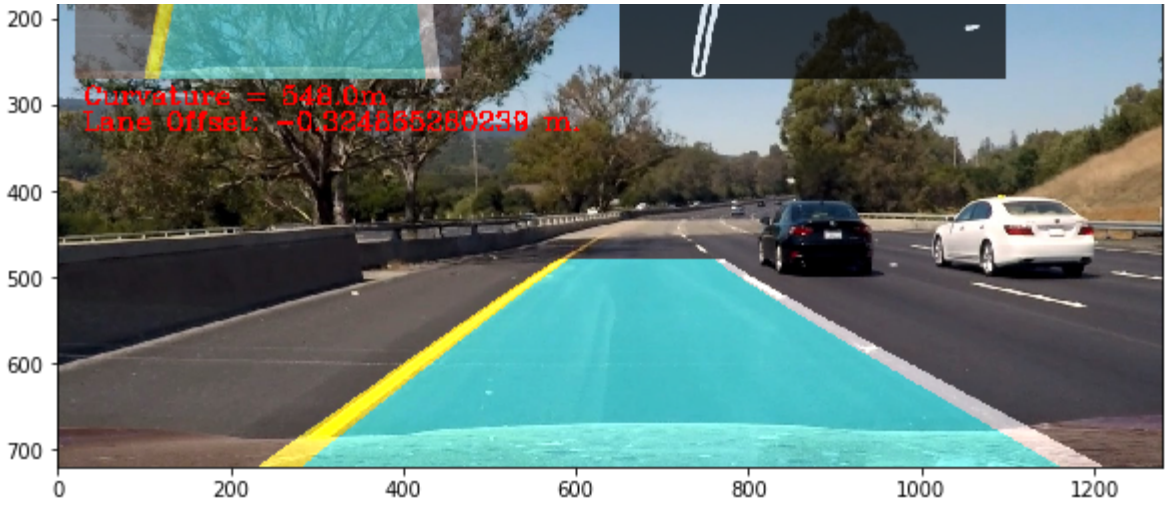

Curvature = 431.0m
Lane Offset: -0.288221271127 m.


600
700
0
200
400
600
800
1000
1200


Curvature = 848.0m
Lane Offset: -0.330644178945 m.
0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200


Curvature = 1212.0m
Lane Offset: -0.0519170125159 m.
0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200


0
100



Test Image pipeline on Video

In [15]:
```
set_prev=0
output1 = 'project_output.mp4'
clip1 = VideoFileClip("project_video.mp4")
output_clip = clip1.fl_image(process_image)
%time output_clip.write_videofile(output1, audio=False)
```

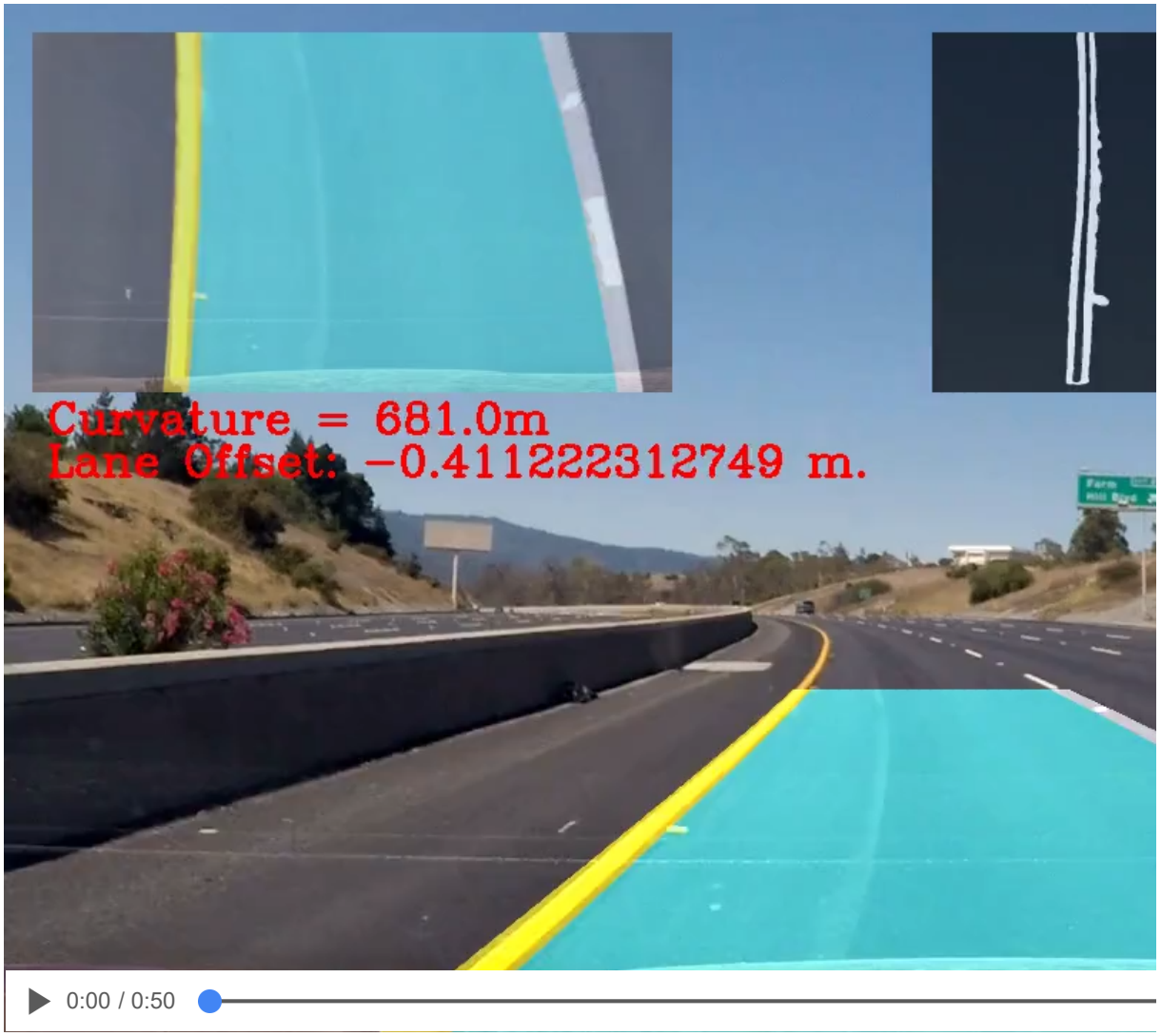
```
[MoviePy] >>>> Building video project_output.mp4
[MoviePy] Writing video project_output.mp4
```

```
100%|████████████████████████████████████████████████████████████████████████████▉| 1260/1261 [09:22<00:00,  2.24it/s]
```

```
[MoviePy] Done.
[MoviePy] >>>> Video ready: project_output.mp4

Wall time: 9min 23s
```

In [16]:
```
HTML("""
<video width="960" height="540" controls>
  <source src="{0}">
</video>
""".format(output1))
```

Out[16]:



Test Image pipeline on Video 2 (challenge_video)

In [17]:
```
set_prev=0
output2 = 'challenge_video_output.mp4'
clip1 = VideoFileClip("challenge_video.mp4")
output_clip = clip1.fl_image(process_image)
%time output_clip.write_videofile(output2, audio=False)
```

```
[MoviePy] >>>> Building video challenge_video_output.mp4
[MoviePy] Writing video challenge_video_output.mp4

100%|████████| 485/485 [03:31<00:00,  2.44it/s]

[MoviePy] Done.
[MoviePy] >>>> Video ready: challenge_video_output.mp4

Wall time: 3min 32s
```

In [18]:
```
HTML("""
<video width="960" height="540" controls>
  <source src="{0}">
</video>
""".format(output2))
```

Out[18]:

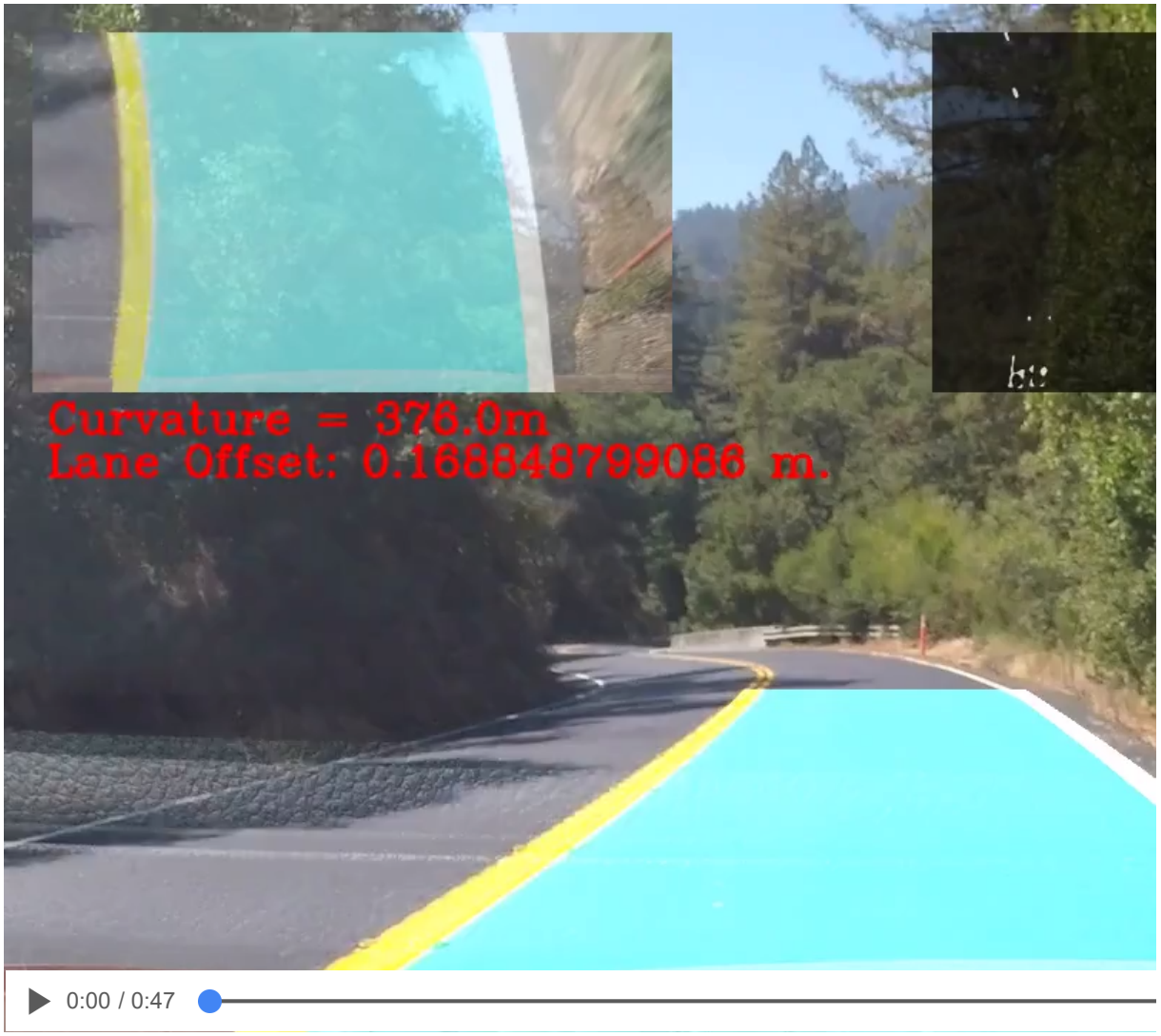
Test Image pipeline on Video 3 (harder_challenge_video)

In [19]:
```
set_prev=0
output = 'harder_challenge_video_output.mp4'
clip1 = VideoFileClip("harder_challenge_video.mp4")
output_clip = clip1.fl_image(process_image)
%time output_clip.write_videofile(output, audio=False)
```

```
[MoviePy] >>>> Building video harder_challenge_video_output.mp4
[MoviePy] Writing video harder_challenge_video_output.mp4

100%|████████████████████████████████████████████████████████████████
██████| 1199/1200 [08:45<00:00,  2.32it/s]

[MoviePy] Done.
[MoviePy] >>>> Video ready: harder_challenge_video_output.mp4

Wall time: 8min 45s
```

In [20]:
```
HTML("""
<video width="960" height="540" controls>
  <source src="{0}">
</video>
""".format(output))
```

Out[20]:



Discussion

Where its failing. Algorithm will fail if road is too much winding. It means in one direct when turning, you might have smaller road portion in front of you. Means you might be seeing road side edge. There is no point of drawing lane on it. see harder challenge video for this effect. One might thing road curving angling might help but this situation could even arise when you reach dead end of road or you have T intersection, in front of you is road edge . this is no point of doing image processing on it. What might help is that we have to keep track of road itself not only lane line on it. we could train our processing pipeline color of road both day and night. Accurate Detection of road edge might help a lot.

What happens when left or right line is not detected for long or its missing from start. I added/subtracted 850 based on detected lane (right or left, less or more than 500). This technique kept lane relatively smooth than before. See “challenge video” for this. see function get_polynomial_fit for implementation. 850 number came from histogram analysis (Step 4 in notebook) its average distance between left and right (see historgram in step 4). If no edge was detected we set 150 for left and 1000 for right again based on histogram.

Acknowledgement

1) https://www.mathworks.com/help/curvefit/smoothing-data.html (https://www.mathworks.com/help/curvefit/smoothing-data.html) 2) https://github.com/jeremy-shannon/CarND-Advanced-Lane-Lines (https://github.com/jeremy-shannon/CarND-Advanced-Lane-Lines) 3) https://github.com/sumitbinnani/CarND-Advanced-Lane-Lines (https://github.com/sumitbinnani/CarND-Advanced-Lane-Lines) 4) As usual my friend Siraj was a great help to understand the concept of this project 5) I would like to spend more time on Harder Challenge video whenever I would get some time.