Test Images

Build your pipeline to work on the images in the directory "test_images"
You should make sure your pipeline works well on these images before you try the videos.

In [13]:
```
import os
images=os.listdir("test_images/")
```

Build a Lane Finding Pipeline

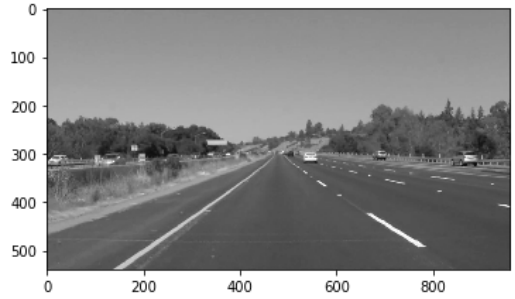
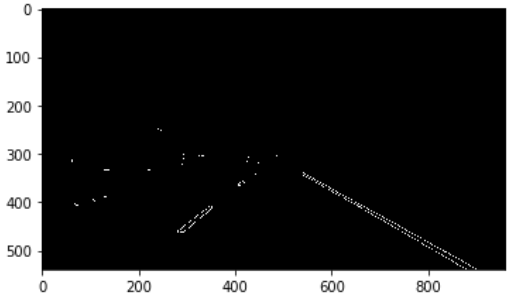
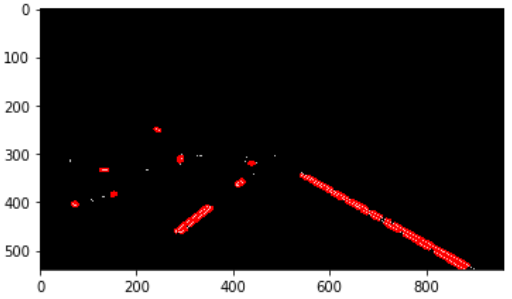
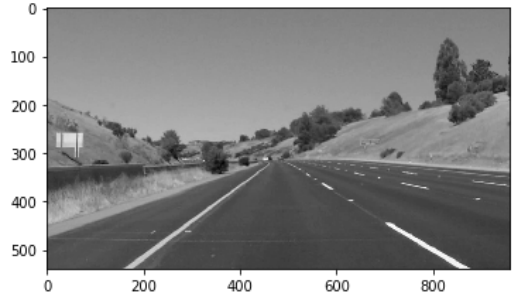
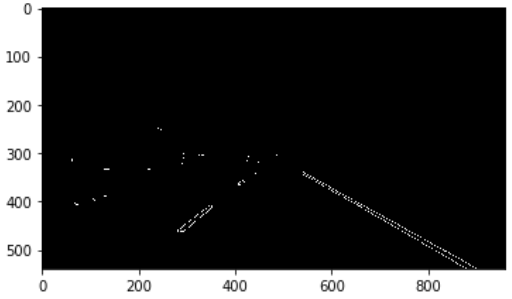
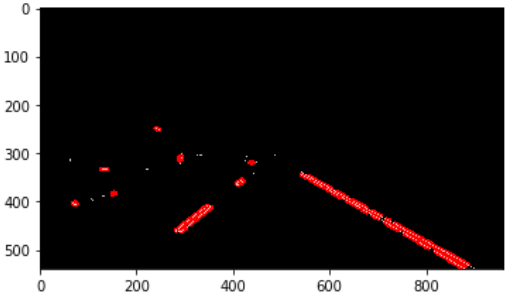
Build the pipeline and run your solution on all test_images. Make copies into the test_images directory, and you can use the images in your writeup report.

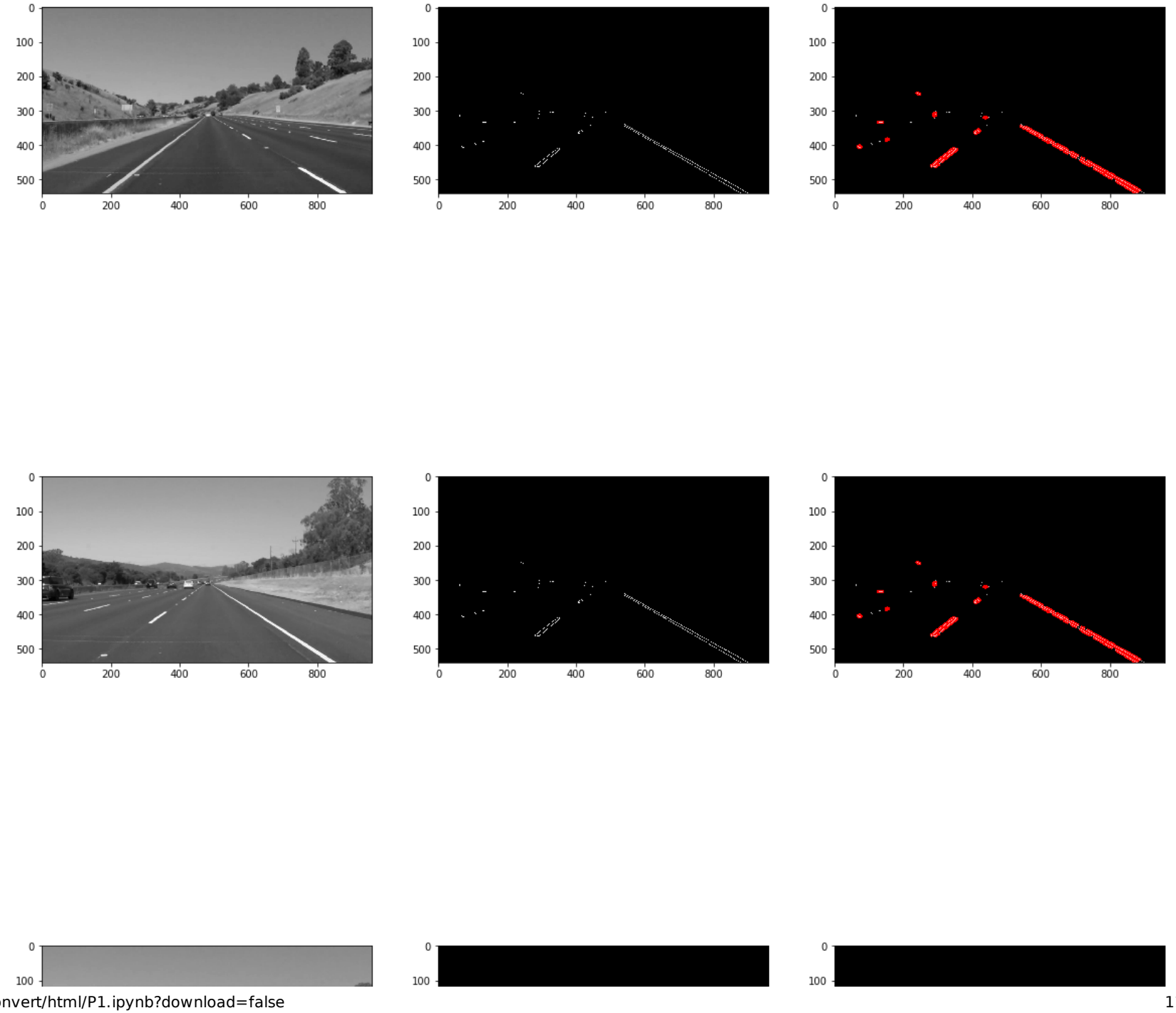
Try tuning the various parameters, especially the low and high Canny thresholds as well as the Hough lines parameters.

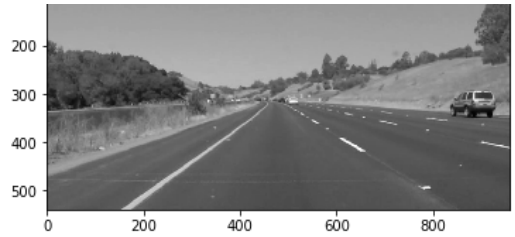
In [53]:
```
# TODO: Build your pipeline that will draw lane lines on the test_images
# then save them to the test_images directory.
# 1. Loop all images
plt.figure(figsize=(20,50))
image_id=1
for image_name in images:
    image_name = "test_images/" + image_name
    print("Process %s" % image_name)
    image = mpimg.imread(image_name)
    gray=grayscale(image)
    plt.subplot(6,3,image_id)
    plt.imshow(gray, cmap='gray')
    image_id+=1

    kernel_size = 7
    blur_gray = gaussian_blur(gray, kernel_size)
```

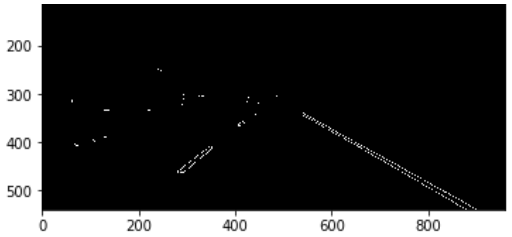
```
# Define parameters for Canny and run it
# NOTE: if you try running this code you might want to change these!
low_threshold = 220
high_threshold = 255
canny(blur_gray, low_threshold, high_threshold)

# Display the image
plt.subplot(6,3,image_id)
plt.imshow(edges, cmap='Greys_r')
image_id+=1

# Define the Hough transform parameters
# Make a blank the same size as our image to draw on
rho = 1
theta = 2*np.pi/180
threshold = 1
min_line_length = 6
max_line_gap = 1
line_image = np.copy(image)*0 #creating a blank to draw lines on

# Run Hough on edge detected image
lines = cv2.HoughLinesP(edges, rho, theta, threshold, np.array([]),
                        min_line_length, max_line_gap)

# Iterate over the output "lines" and draw lines on the blank
for line in lines:
    for x1,y1,x2,y2 in line:
        cv2.line(line_image,(x1,y1),(x2,y2),(255,0,0),10)

# Create a "color" binary image to combine with line image
color_edges = np.dstack((edges, edges, edges))

# Draw the lines on the edge image
combo = cv2.addWeighted(color_edges, 0.8, line_image, 1, 0)
plt.subplot(6,3,image_id)
plt.imshow(combo)
image_id+=1
```

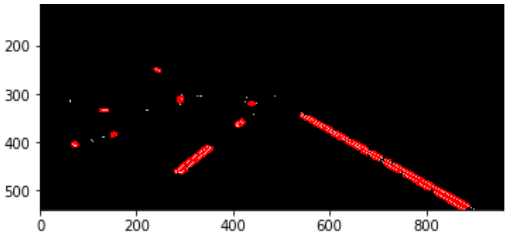
```
Process test_images/solidYellowLeft.jpg
Process test_images/solidYellowCurve2.jpg
Process test_images/whiteCarLaneSwitch.jpg
Process test_images/solidWhiteRight.jpg
Process test_images/solidYellowCurve.jpg
Process test_images/solidWhiteCurve.jpg
```

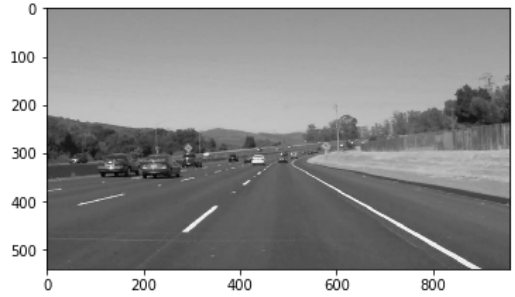







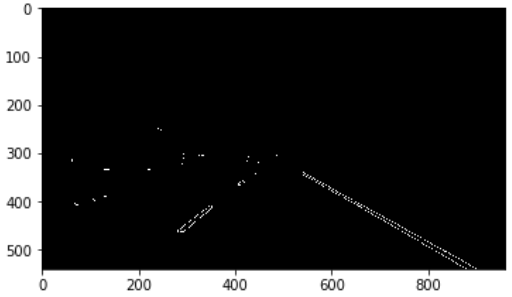


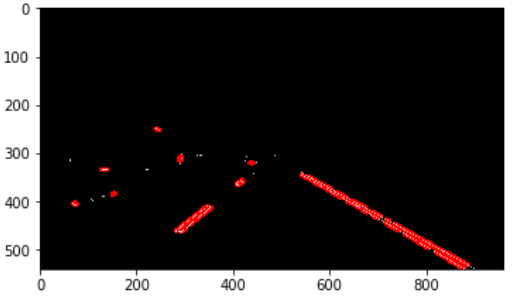

200
300
400
500
0
200
400
600
800


200
300
400
500
0
200
400
600
800


200
300
400
500
0
200
400
600
800


0
100
200
300
400
500
0
200
400
600
800


0
100
200
300
400
500
0
200
400
600
800


0
100
200
300
400
500
0
200
400
600
800