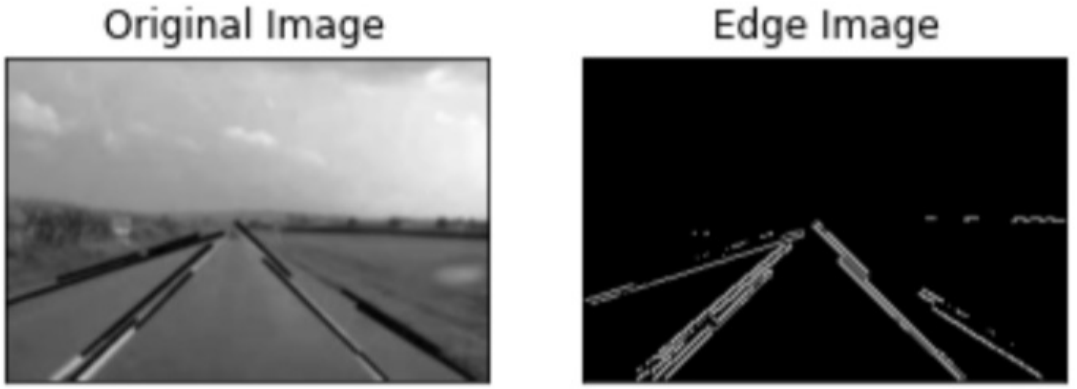
In [8]:

```
# canny edge detection

edges = cv.Canny(img,250,250)

plt.subplot(121),plt.imshow(im
plt.title('Original Image'), p
plt.subplot(122),plt.imshow(ed
plt.title('Edge Image'), plt.x
```



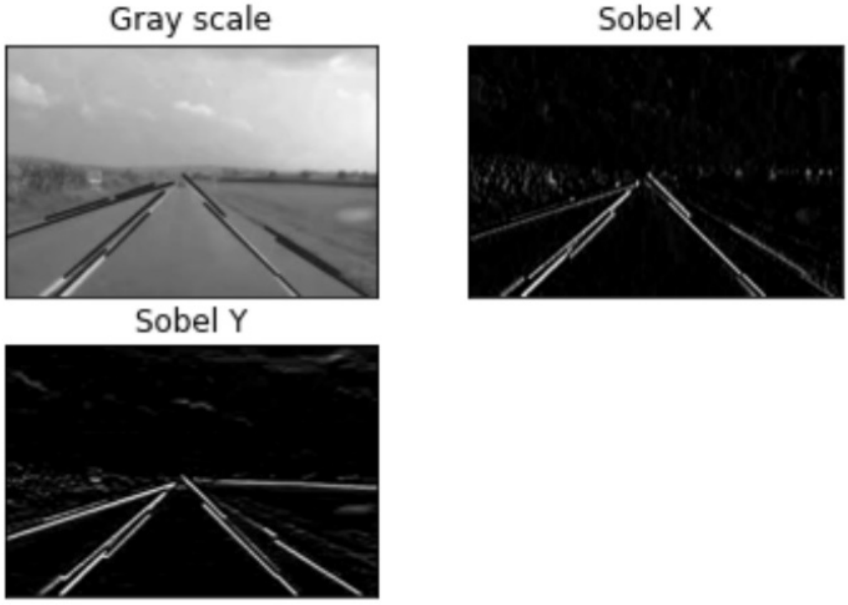
In [ ]:

In [6]:

```
# Create a custom kernel

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

## TODO: Create and apply a So
sobel_x = np.array([[-1,0,1],
                      [ -2, 0 , 
                      [ -1,0,1]]
```

In [7]:

```
# Filter the image using filte
filtered_image_y = cv.filter2D
filtered_image_x = cv.filter2D

plt.subplot(2,2,1),plt.imshow(
plt.title('Gray scale'), plt.x
plt.subplot(2,2,2),plt.imshow(
plt.title('Sobel X'), plt.xticl
plt.subplot(2,2,3),plt.imshow(
plt.title('Sobel Y'), plt.xticl
plt.show()
```



Edge detection

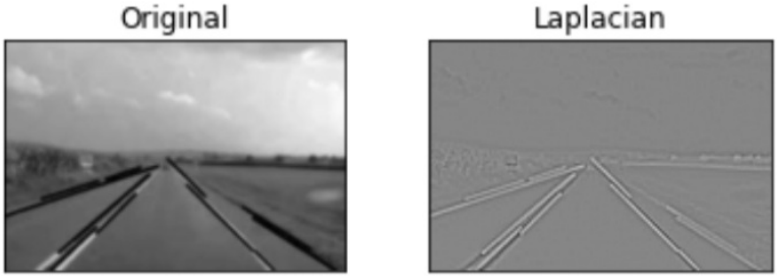
Different edge detection is applied:

- Laplacian edge detection
- Sobel method
- Canny edge detection

In [3]:
```
# Import necessary modules!
import cv2 as cv
import numpy as np
from matplotlib import pyplot a
```

In [4]:
```
# read the image
img1 = cv.imread('img/road_1.jp
```

In [5]:
```
# remove noise
img = cv.GaussianBlur(img1,(3,3

# convolute with proper kernel
laplacian = cv.Laplacian(img,cv

plt.subplot(2,2,1),plt.imshow(i
plt.title('Original'), plt.xtic
plt.subplot(2,2,2),plt.imshow(l
plt.title('Laplacian'), plt.xti

plt.show()
```



In [1]:

```
'''
Python Program for Road Detect
'''

# Import necessary modules!
import numpy as np
import matplotlib.pyplot as pl
import matplotlib.image as img
import cv2 as cv
```

In [2]:

```
image=cv.imread('img/road_1.jp
height=image.shape[0]
width=image.shape[1]
img=np.copy(image)
def region_of_interst(img,vert
    mask = np.zeros_like(img)
    #channel_count = img.shape
    match_mask_color = 255
    cv.fillPoly(mask, vertices
    masked_image = cv.bitwise_
    return masked_image
region_of_interest_vertices =
    (0, height),
    (width / 2, height / 2),
    (width, height),]
gray=cv.cvtColor(image,cv.COLO
canny=cv.Canny(gray,100,200)
cropped_image = region_of_inte
plt.imshow(cropped_image)
```

Out[2]: