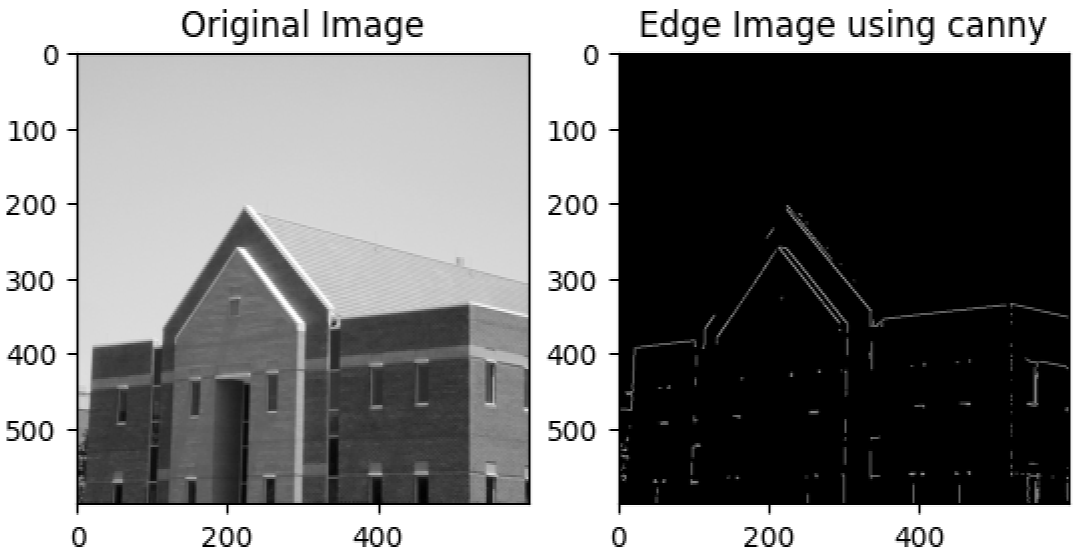
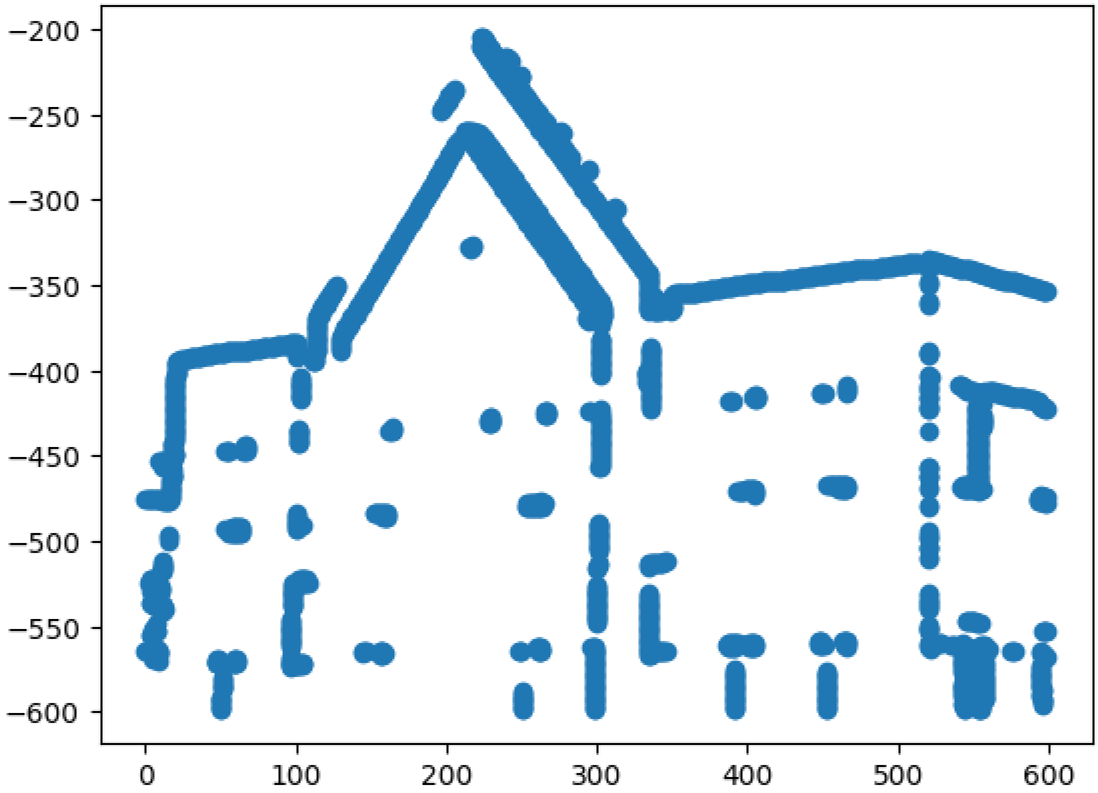
```
In [ ]: import cv2 as cv
        import numpy as np
        import matplotlib.pyplot as plt
        img = cv.imread('/examples/building.png')
        edges = cv.Canny(img, 350, 390)
        plt.subplot(121), plt.imshow(img)
        plt.title('Original Image'),
        plt.subplot(122)
        plt.imshow(edges, cmap='gray')
        plt.title('Edge Image using canny')
        plt.show()
```



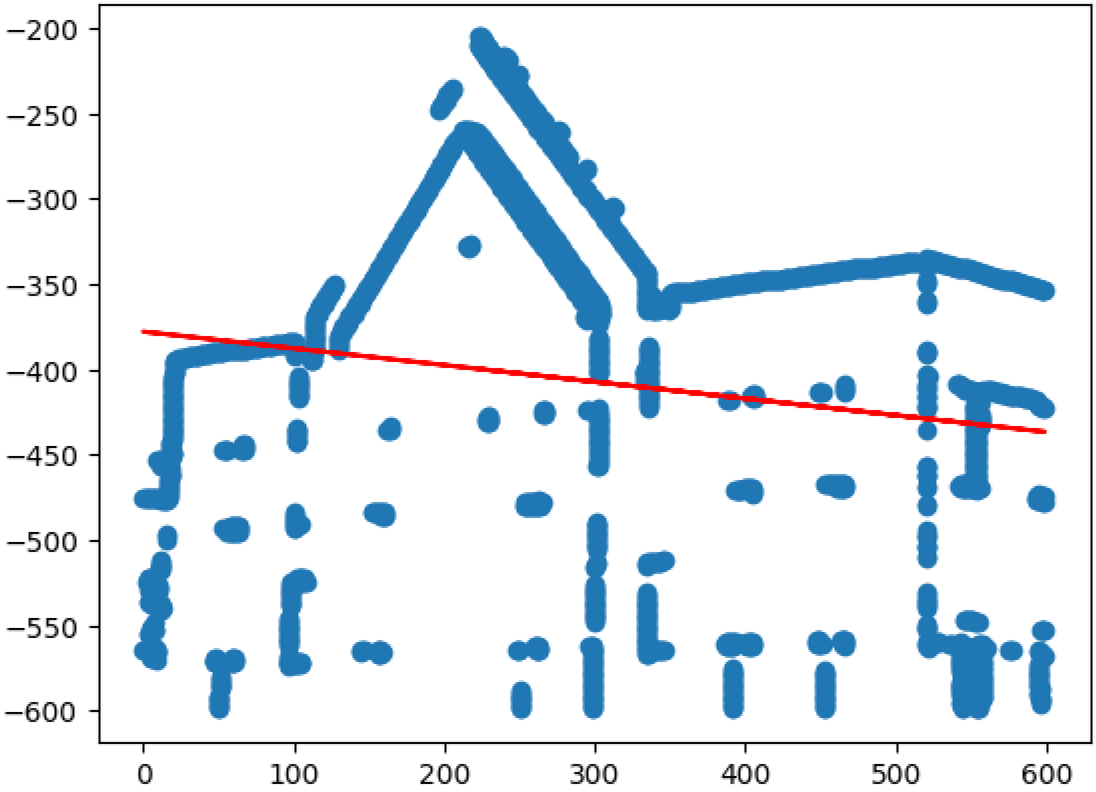
```
In [ ]: #2
        # Extracting the coordinates of the edges
        indices = np.where(edges != [0])
        x = indices[1]
        y = -indices[0]
```

```
In [ ]: # Plotting the coordinates
        plt.scatter(x, y)
        plt.show()
```



In [ ]:

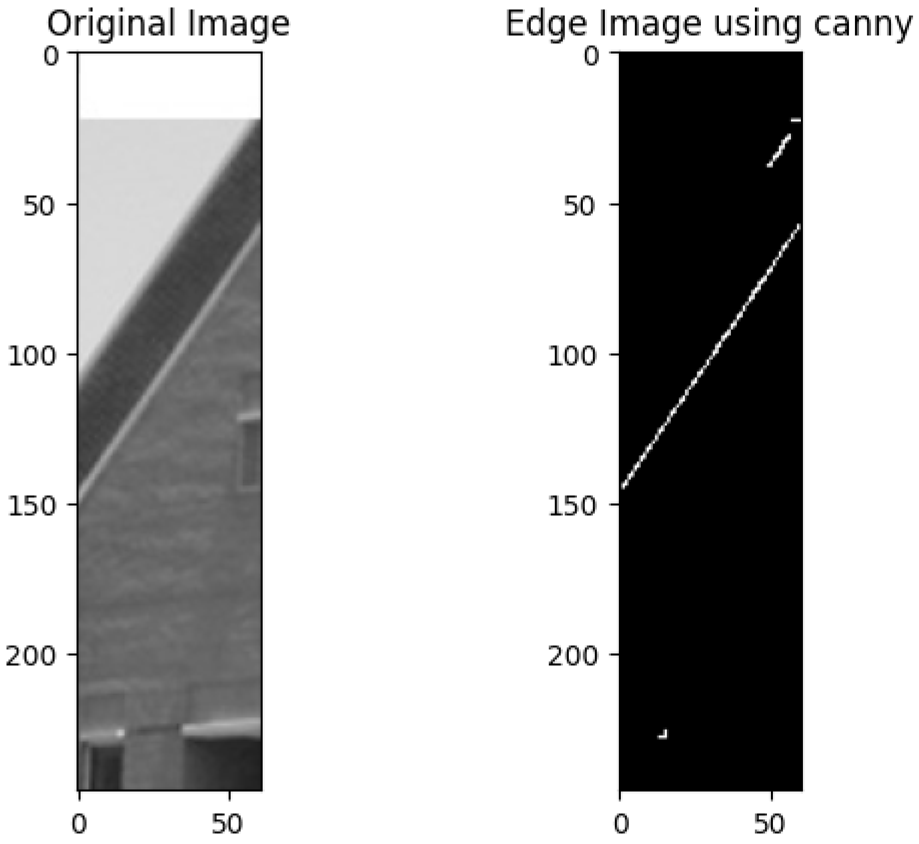
```
# Finding the total least-squares-fit line
x_mean = np.mean(x)
y_mean = np.mean(y)
Sxy = np.sum((x-x_mean)*(y-y_mean))
Sxx = np.sum((x-x_mean)**2)
Syy = np.sum((y-y_mean)**2)
b = Sxy/Sxx
a = y_mean - b*x_mean
e = np.sqrt(Syy-b*Sxy)/(len(x)-2)
# Plotting the line and coordinates
plt.scatter(x, y)
plt.plot(x, b*x + a, 'r')
plt.show()
# Calculating the angle
theta = np.arctan(b)*180/np.pi
print('Roof angle based on total least-squares-fit:', theta)
```



```
Roof angle based on total least-squares-fit: -5.602123856692337
```

In [ ]:
```
#5 Estimated value for the roof was 30 degrees
#6 Do you think the estimation is correct? if not explain reason for this error?
# No Since the LS fitting method is inacurate also  Canny edge detector algorithm m
#  have extracted some unnecessary edges or missed some important edges,
#  which can affect the accuracy of the line fitting.
# 7 Propose a better alogorithm
# We could always use RANSAC
```

In [ ]:
```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
img = cv.imread('/examples/building_crop.jpg')
edges = cv.Canny(img, 350, 390)
plt.subplot(121), plt.imshow(img)
plt.title('Original Image'),
plt.subplot(122)
plt.imshow(edges, cmap='gray')
plt.title('Edge Image using canny')
plt.show()
```

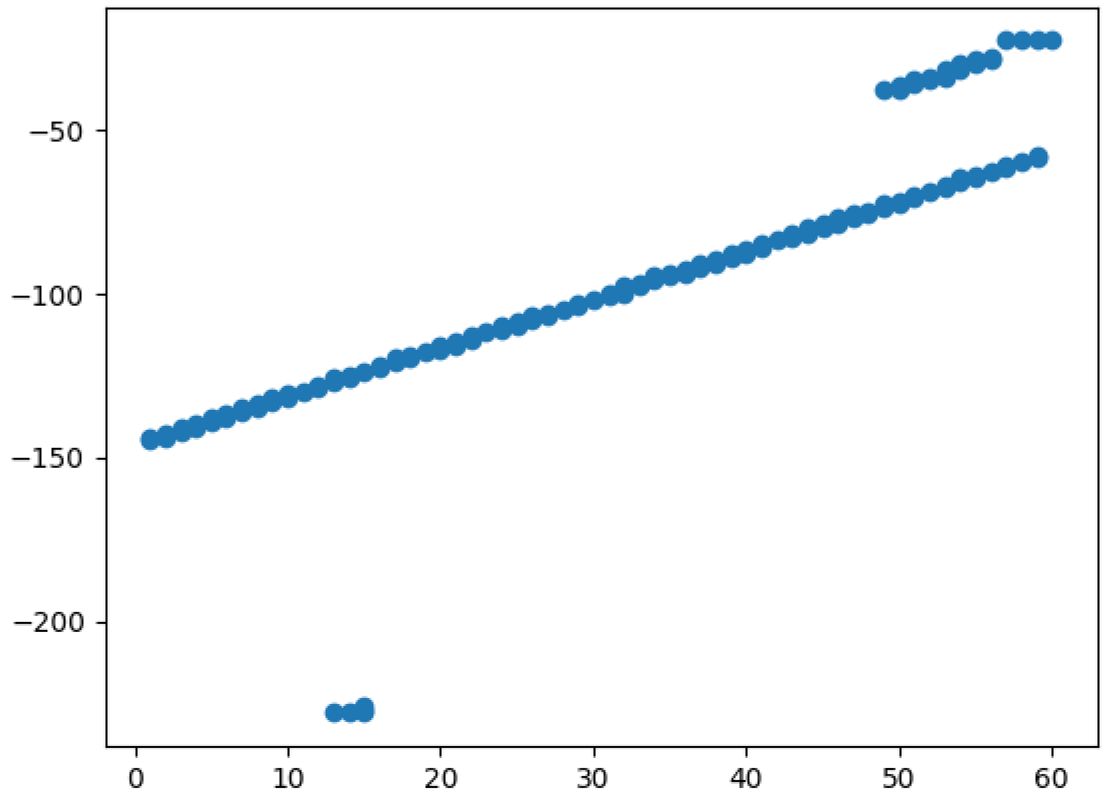


In [ ]:

```
indices = np.where(edges != [0])
x = indices[1]
y = -indices[0]
```

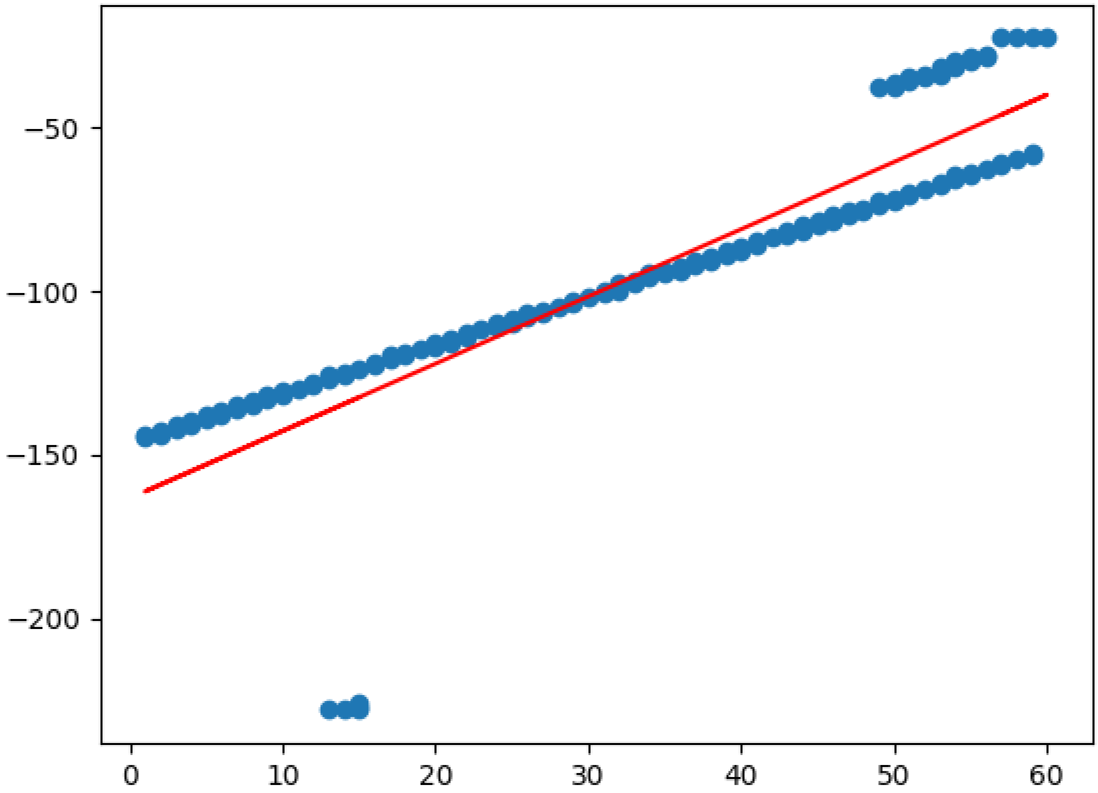
In [ ]:

```
plt.scatter(x, y)
plt.show()
```



In [ ]:

```
x_mean = np.mean(x)
y_mean = np.mean(y)
Sxy = np.sum((x-x_mean)*(y-y_mean))
Sxx = np.sum((x-x_mean)**2)
Syy = np.sum((y-y_mean)**2)
b = Sxy/Sxx
a = y_mean - b*x_mean
e = np.sqrt(Syy-b*Sxy)/(len(x)-2)
# Plotting the line and the coordinates
plt.scatter(x, y)
plt.plot(x, b*x + a, 'r')
plt.show()
# Calculating angle
theta = np.arctan(b)*180/np.pi
print('Roof angle based on total least-squares-fit:', theta)
```



```
Roof angle based on total least-squares-fit: 64.00898352832498
```

In [ ]:

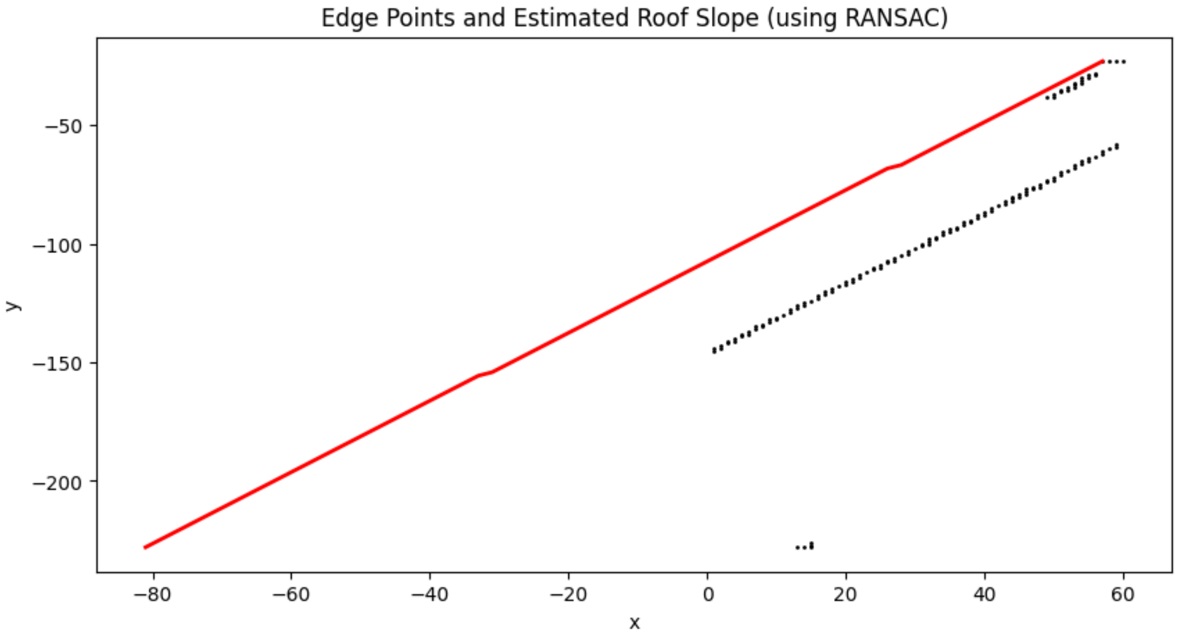
```
# Quetsion 09
## It may not be accurate since there are outliers in the above graph therefore
## due to the oitliers the LS result may not be accurate
## Since we used the edge only its more accurate than the earlier image

# Question 10
## A better performing algorithm would be RANSAC we can try and apply RANSAC method
```

In [ ]:

```
import cv2
import numpy as np
from sklearn.linear_model import RANSACRegressor
import matplotlib.pyplot as plt

img = cv2.imread('/examples/building_crop.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
edges = cv2.Canny(gray, 350, 390)
indices = np.where(edges != [0])
x = indices[1]
y = -indices[0]

model_ransac = RANSACRegressor(base_estimator=None, min_samples=2, residual_thresho
model_ransac.fit(x.reshape(-1, 1), y)
slope = model_ransac.estimator_.coef_[0]
angle = np.arctan(slope) * 180 / np.pi
print("The angle when we use this method is : " ,angle)
line_y = np.linspace(np.min(y), np.max(y), len(x))
line_x = np.round(x[0] + (line_y - y[0]) / slope).astype(int)

plt.figure(figsize=(10,5))
plt.scatter(x, y, s=1, c='black')
plt.plot(line_x, line_y, linewidth=2, c='red')
plt.title('Edge Points and Estimated Roof Slope (using RANSAC)')
plt.xlabel('x')
plt.ylabel('y')
plt.gca()
plt.show()

# The provided angle is almost accurate and there is a difference between LS and RA
```

```
The angle when we use this method is :  55.98734019204485
```

```
c:\Python39\cv\Lib\site-packages\sklearn\linear_model\_ransac.py:343: FutureWarnin
g: `base_estimator` was renamed to `estimator` in version 1.1 and will be removed
in 1.3.
  warnings.warn(
```



In [ ]:

```
# Question 13
## Explain, why your proposed approach is performing better than the least-squares-
## least-squares-fit.
#RANSAC method was  designed to handle outliers in the data,
#  by iteratively fitting a model to a random subset of the data and identifying
#  the remaining points as "outliers". The inliers are then used to calculate a rob
#  which is less affected by the presence of outliers. This process is repeated a f
#  and the best model found during the iterations is returned as the final result.
#  RANSAC is expected to provide a more accurate estimate of the line slope in the
#  compared to the least-squares and total least-squares methods.
```