Assignment7

March 14, 2021

1 Assignment 7 :

1.0.1 For image1 use Sobel operators to find the horizontal and vertical edges in the image. Repeat the steps, but with the original image smoothed using a 5 × 5 averaging kernel prior to edge detection. Show the results and write your observations.

1.0.2 Use Canny edge detector on image1. Use different thresholds and analyze your results.

2 1- Import Libraries

[1]:
```
import numpy as np # numpy
import matplotlib.pyplot as plt # matplotlib
import cv2 # opencv
```

3 2- Read the img and turn it into grayscale

[2]:
```
img = cv2.imread('1(2).tif') # read image
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # BGR ==> GRAY
```

4 3- Function to display image

[3]:
```
def show_image(img):
    """
    This Method to disply image in 10*10 scale
    """
    fig = plt.figure(figsize=(10,10))
    ax  = fig.add_subplot(1,1,1, xticks=[], yticks=[])
    ax.imshow(img, cmap='gray')
```

5 4- Display the image and image shape

[6]:
```
show_image(img)
```

[7]:
```
img.shape
```

[7]: (834, 1114, 3)

[8]:
```
gray.shape
```

[8]: (834, 1114)

[9]:
```
img.size
```

[9]: 2787228

[10]:
```
gray.size
```

[10]: 929076

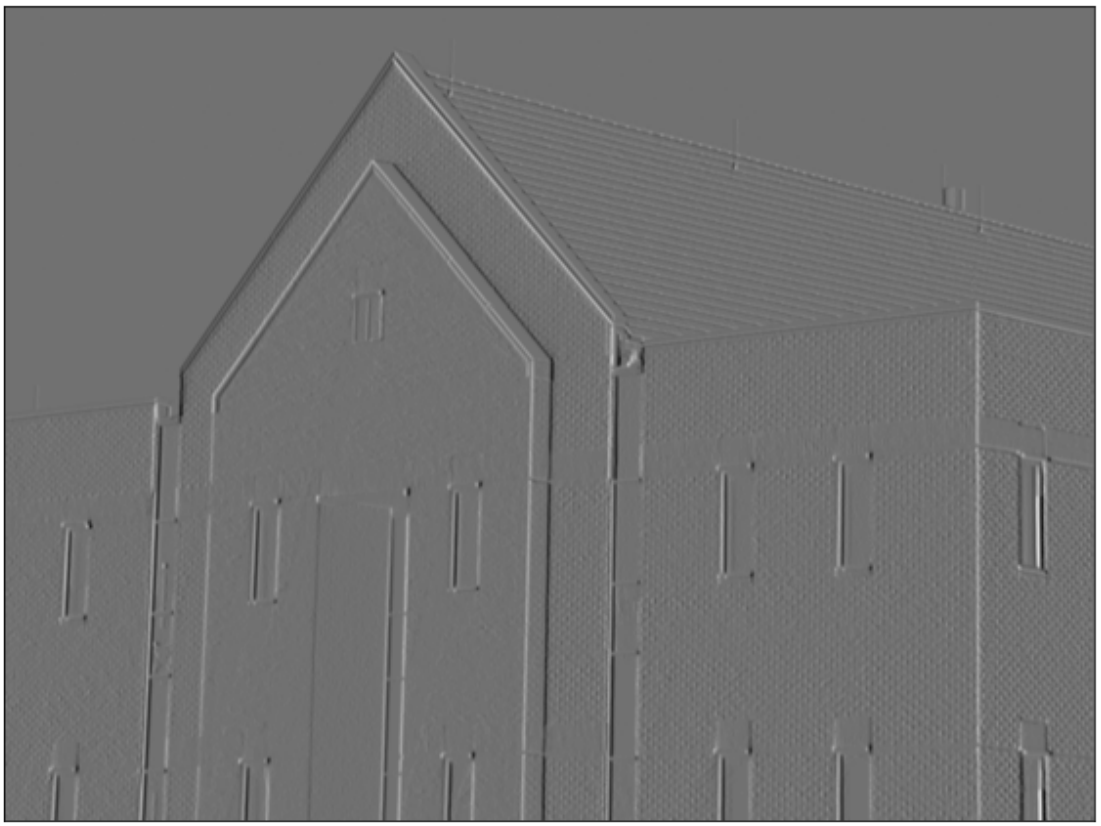
6 We notice the reducing in image size between the 3 channels image and one channel image

7 5- Sobel edge detection without Blur

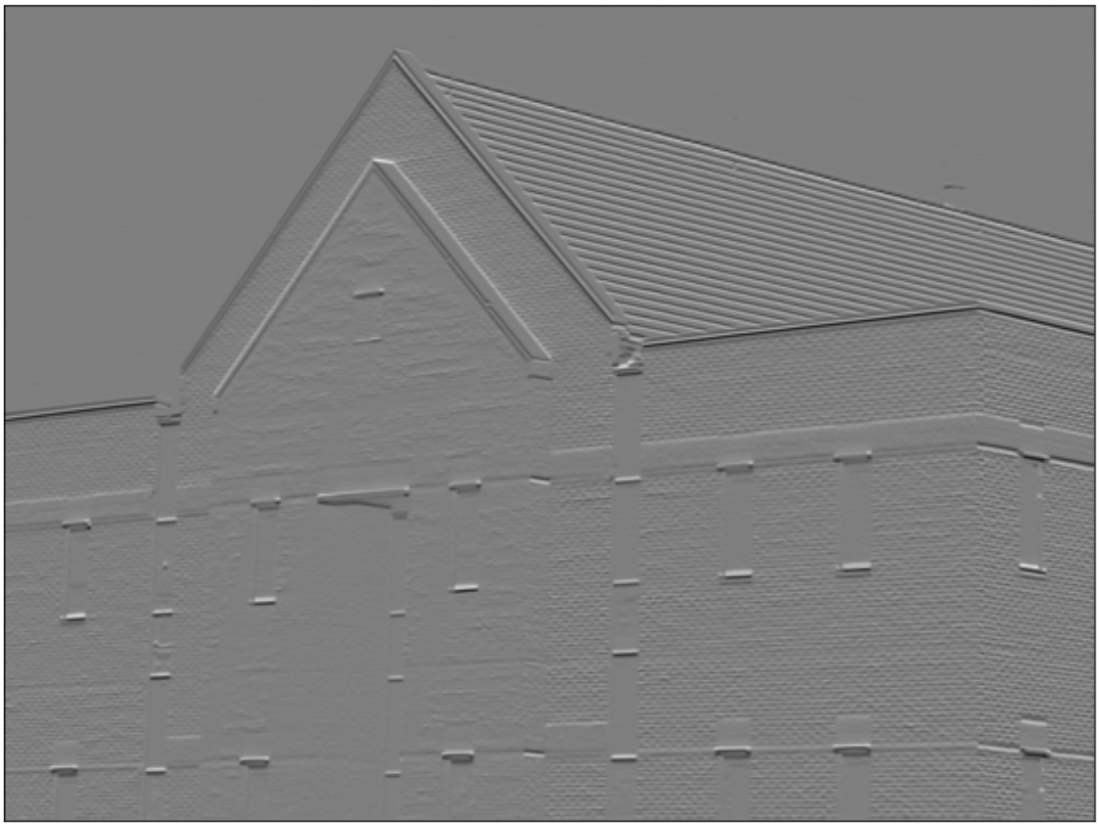
[11]:
```
sobelX = cv2.Sobel(src=gray, ddepth=cv2.CV_64F, dx=1, dy=0, ksize=5)  # X edge
↪detection

sobelY = cv2.Sobel(src=gray, ddepth=cv2.CV_64F, dx=0, dy=1, ksize=5)  # Y edge
↪detection
```

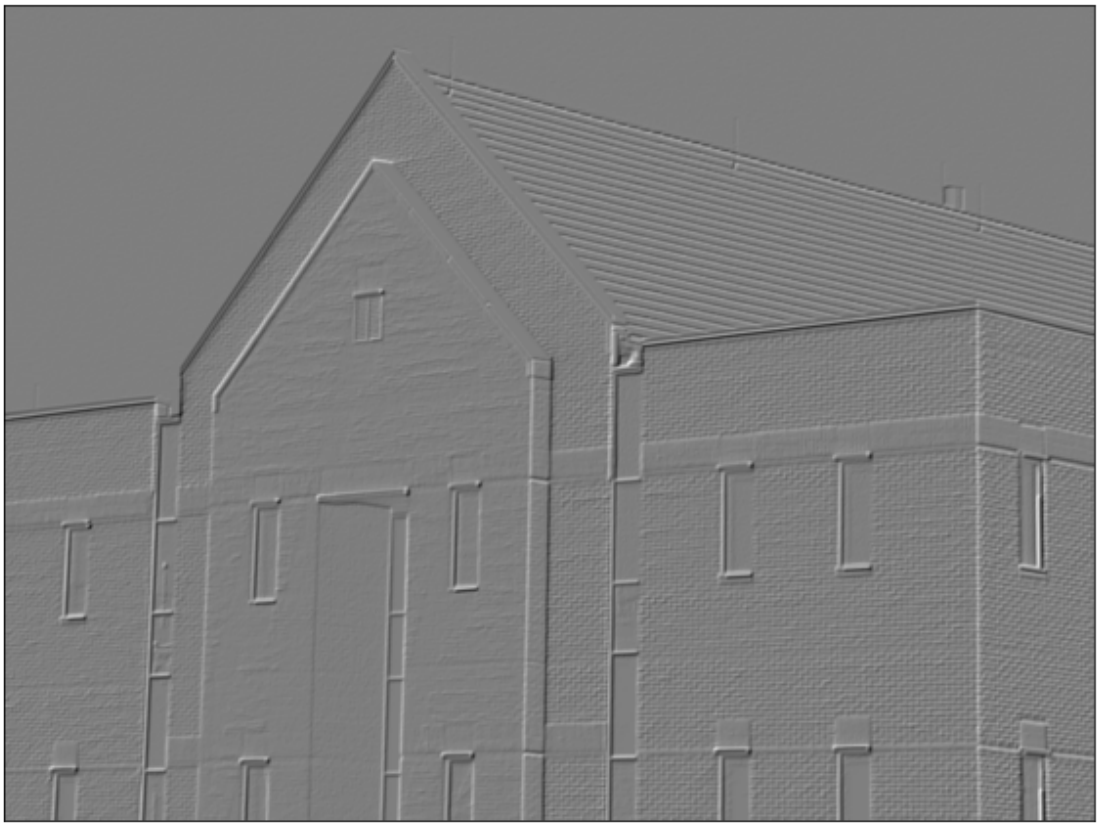
[13]:
```
show_image(sobelX)
```



[14]:
```
show_image(sobelY)
```



[15]:
```
sobel = sobelX +sobelY # combine them togather
show_image(sobel)
```



8 6- Sobel edge detection with Blur

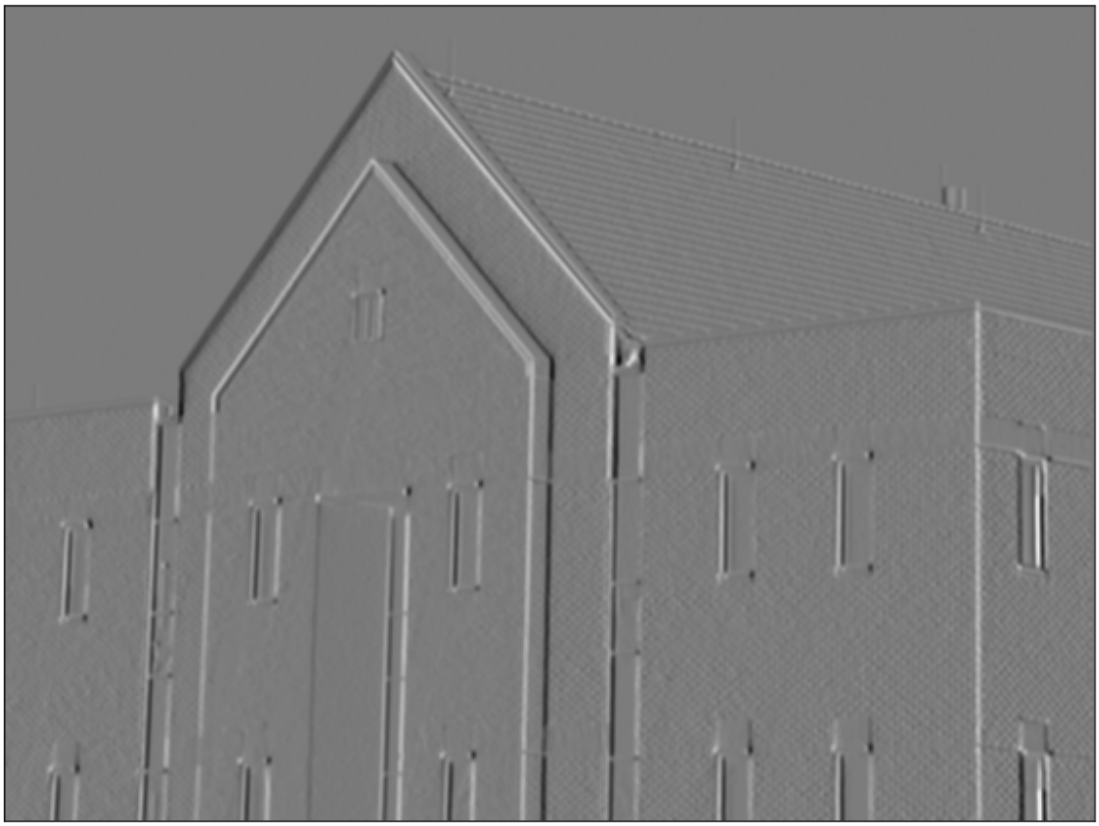
[16]:
```
blur = cv2.blur(gray, (5,5))
```

[18]:
```
show_image(blur)
```

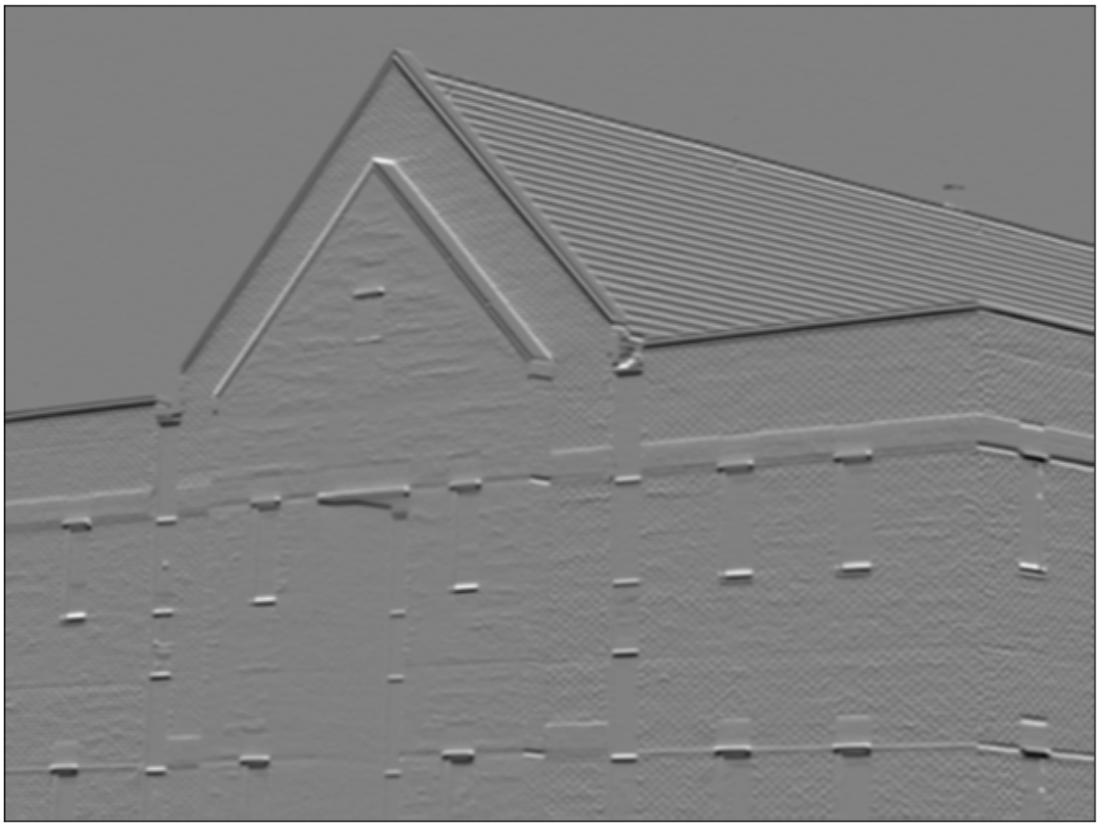
[17]:
```
sobelX = cv2.Sobel(src=blur, ddepth=cv2.CV_64F, dx=1, dy=0, ksize=5)  # X edge␣
↪detection

sobelY = cv2.Sobel(src=blur, ddepth=cv2.CV_64F, dx=0, dy=1, ksize=5)  # Y edge␣
↪detection
```

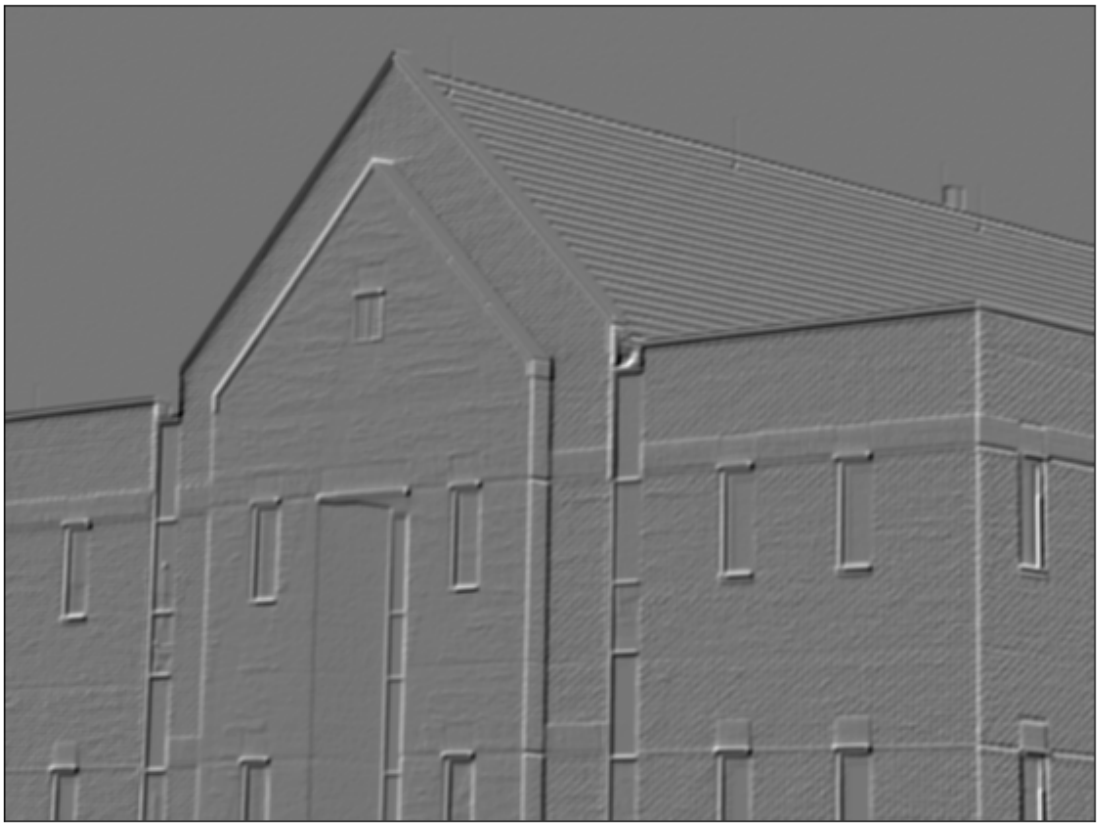
[19]:
```
show_image(sobelX)
```



[20]:
```
show_image(sobelY)
```



[21]:
```
sobel = sobelX +sobelY # combine them togather
show_image(sobel)
```



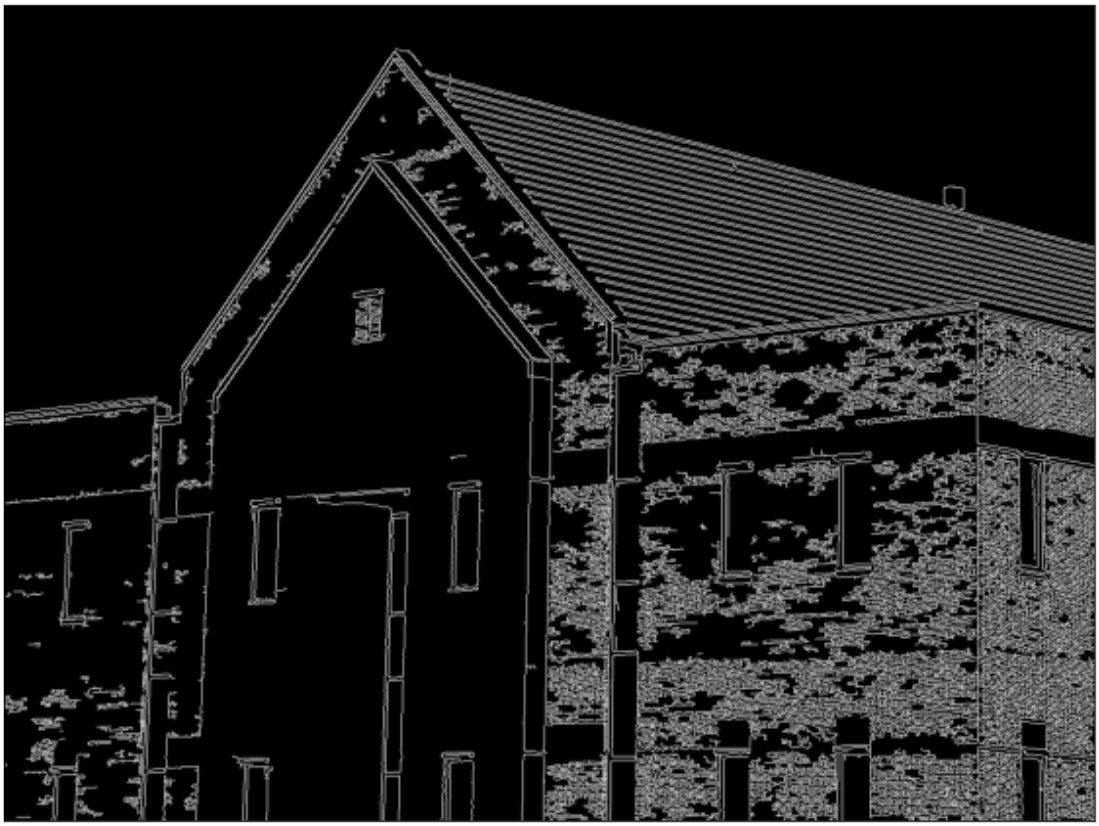
9 7- Write the observation of previous two experiment

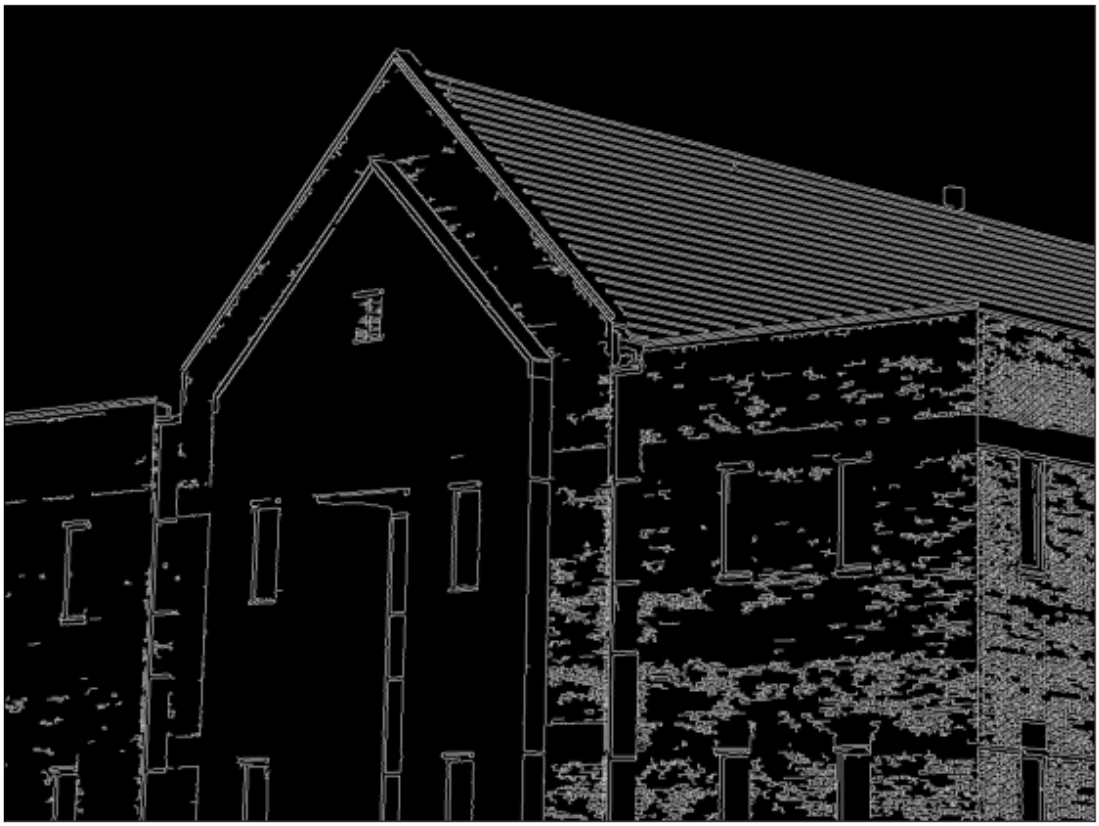
10 What I notice that the image become less sharp and it is ignore small details and focus on the shape of building in general

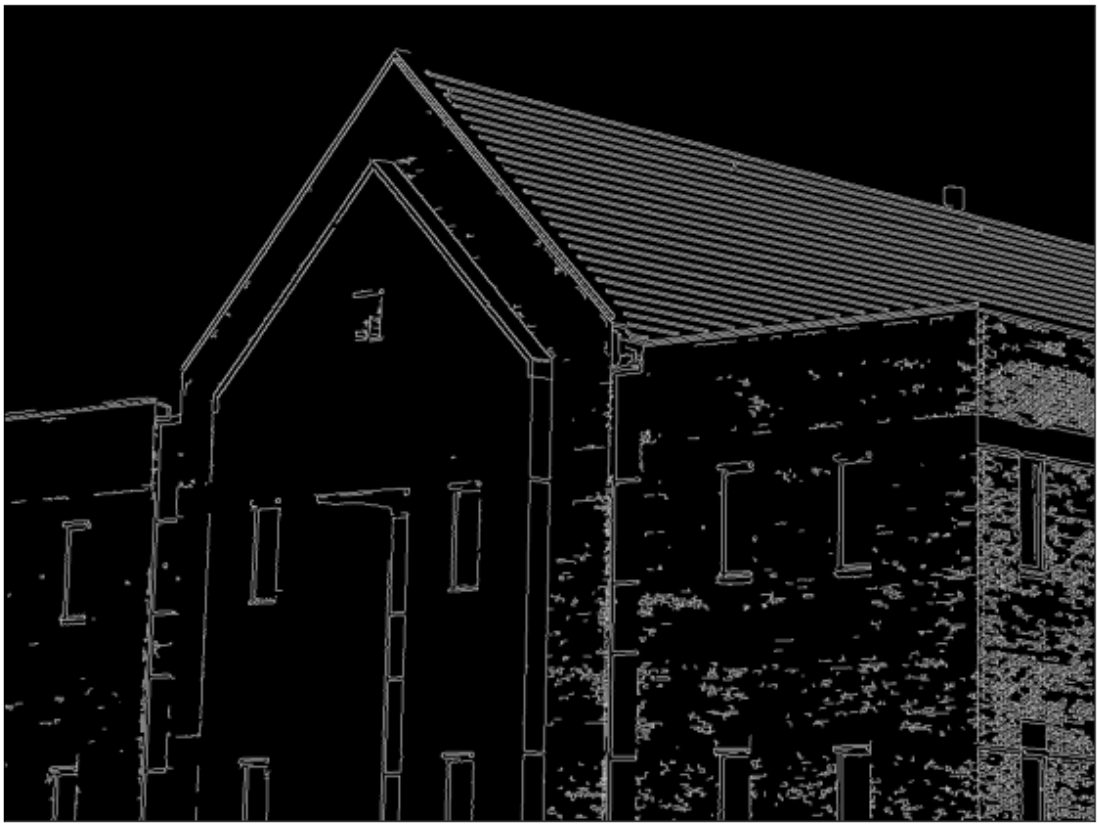
11 8- Canny edge detection without Blur

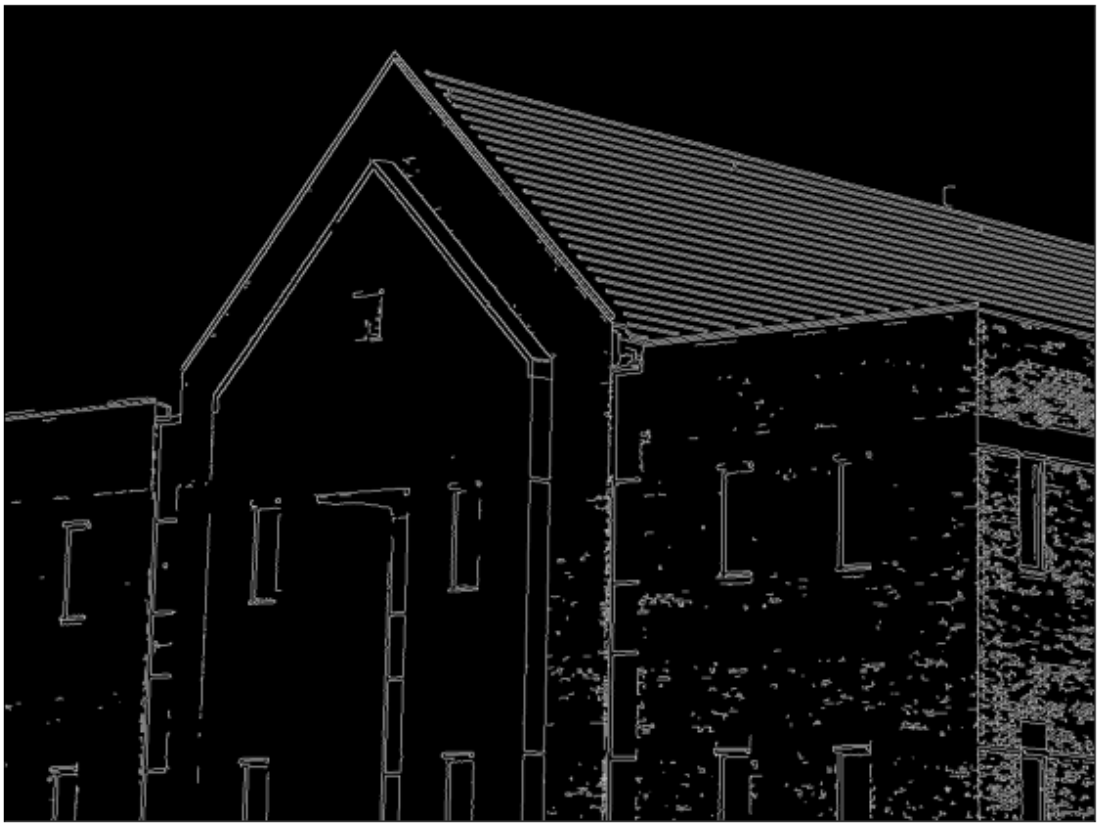
[22]:
```
min_threshs = [75,100, 128, 150, 175, 200] #list of min threshs
max_threshs = [200, 210,220,230,240,250]#list of max threshs
```

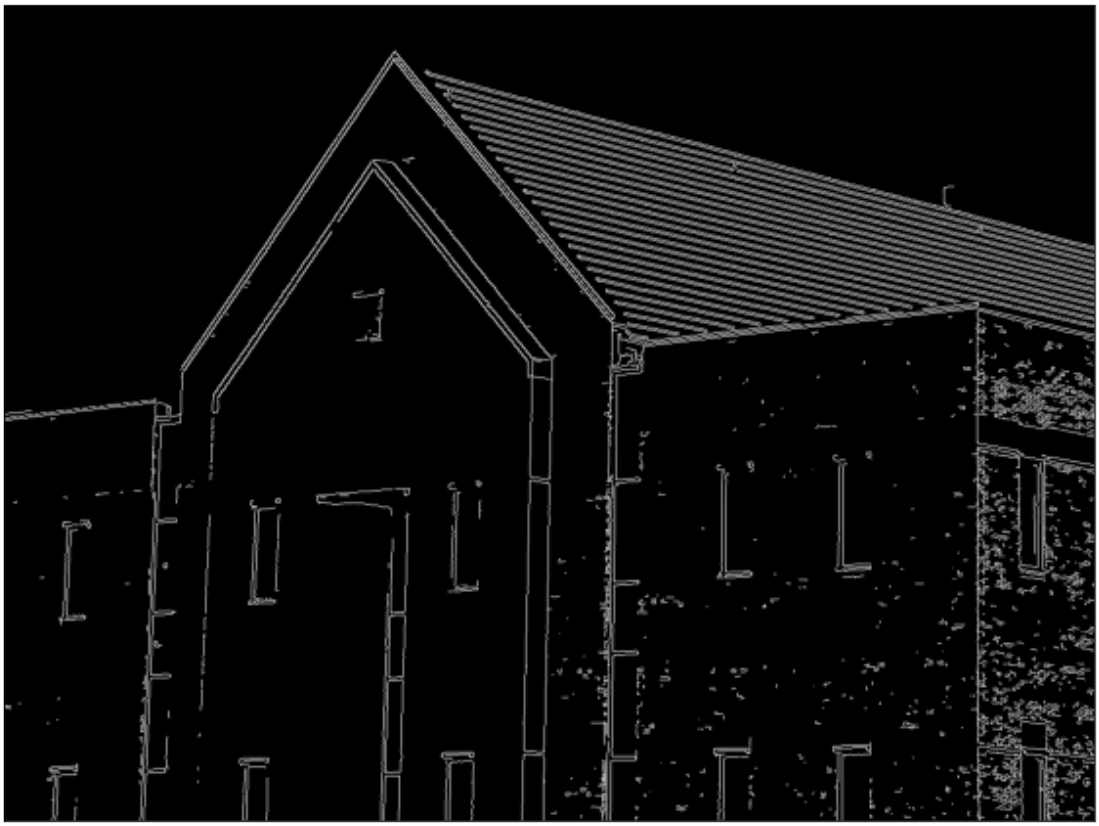
[25]:
```
for i in range(len(min_threshs)):
    Canny = cv2.Canny(gray, min_threshs[i],max_threshs[i])
    show_image(Canny)
```

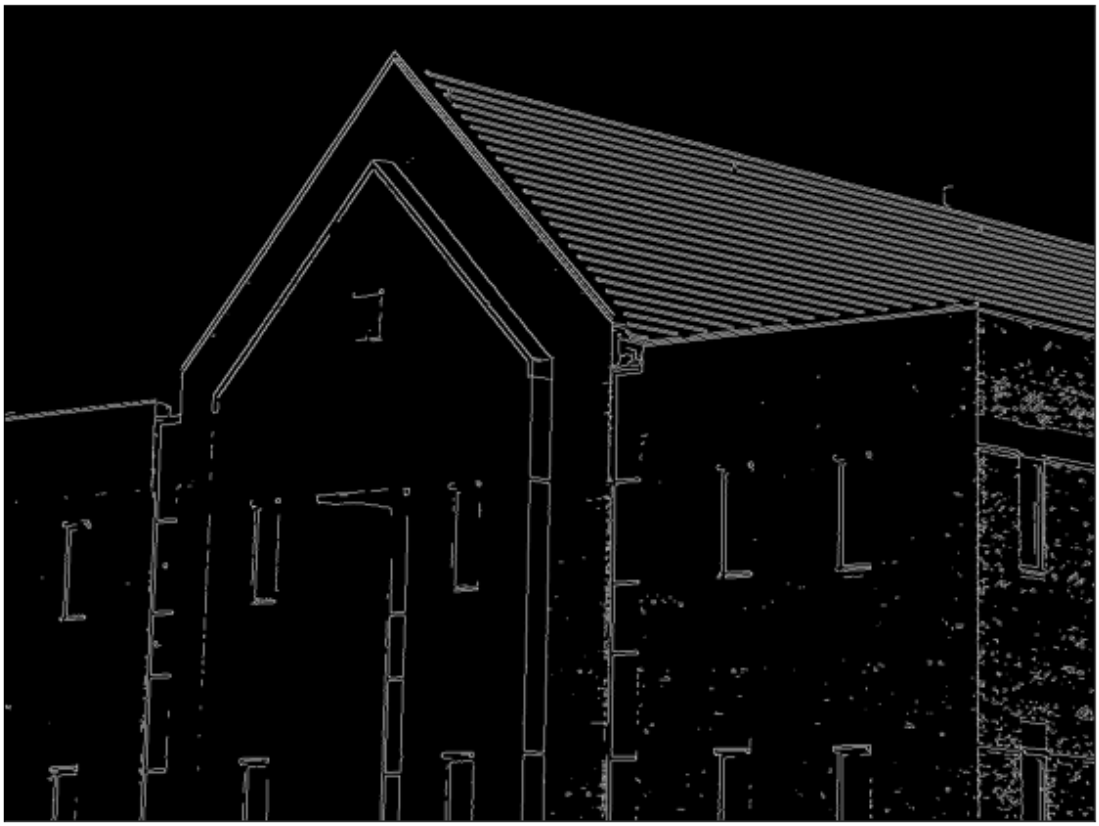








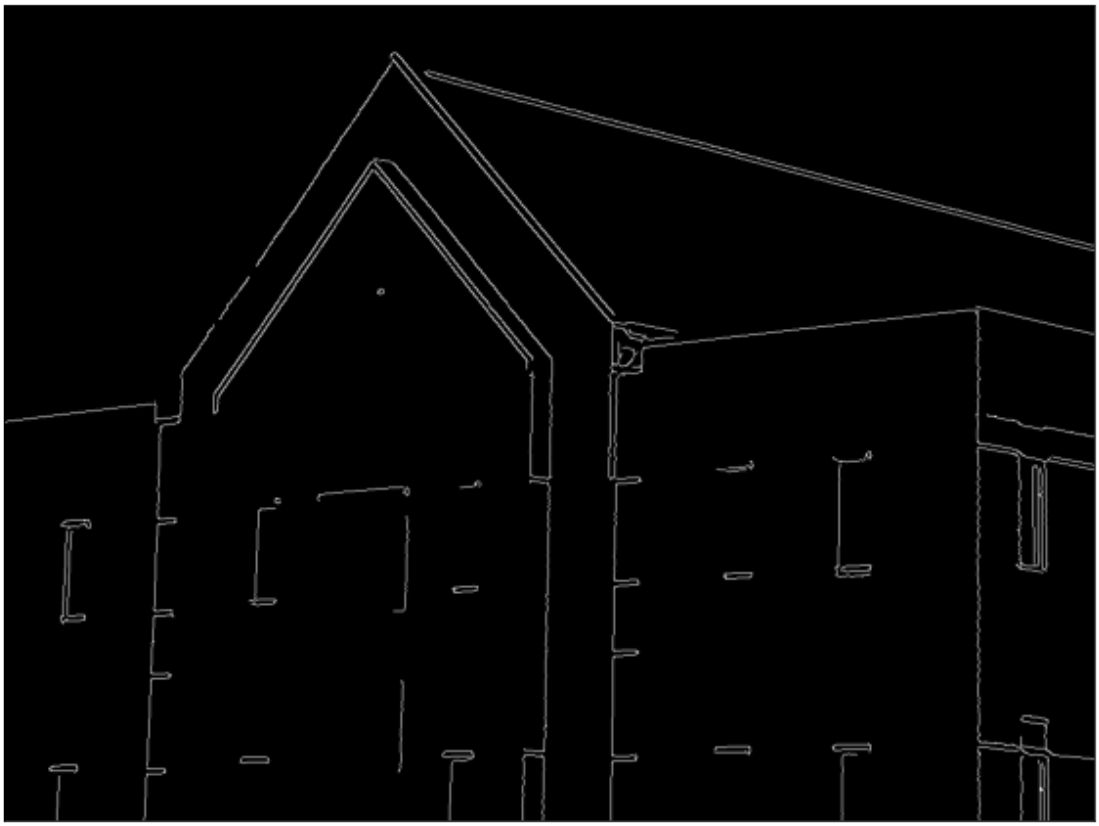


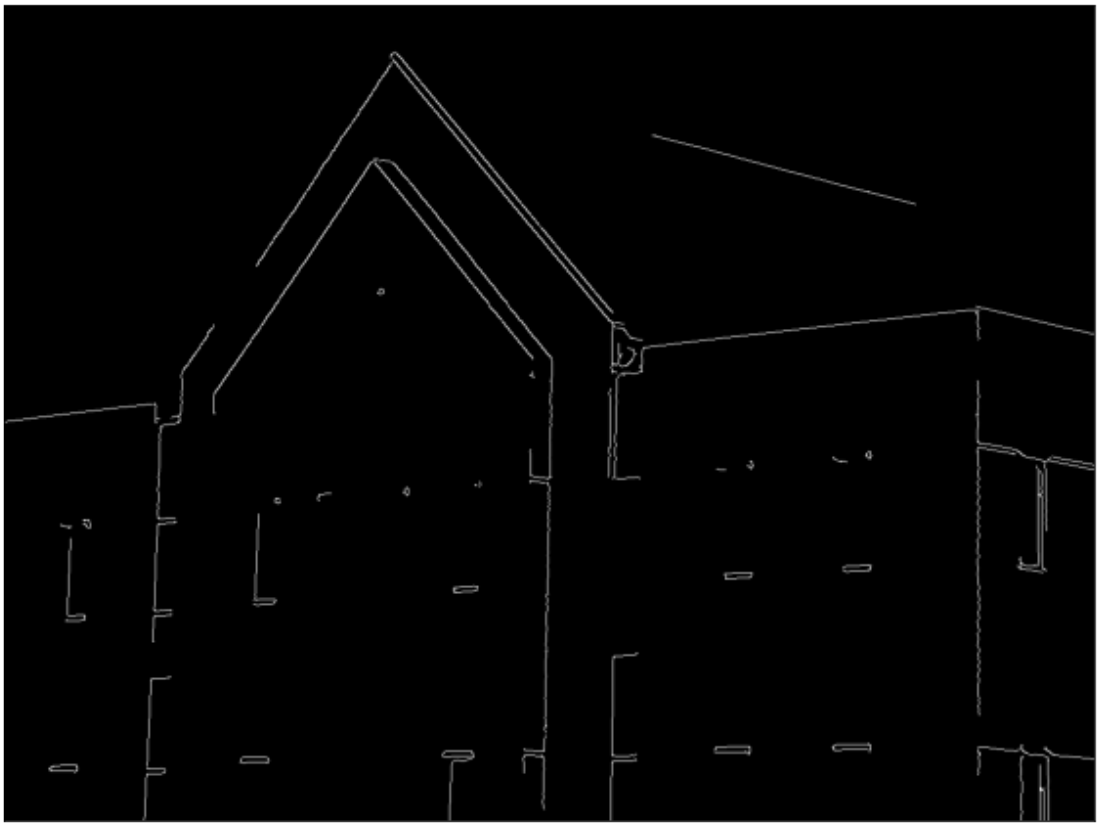


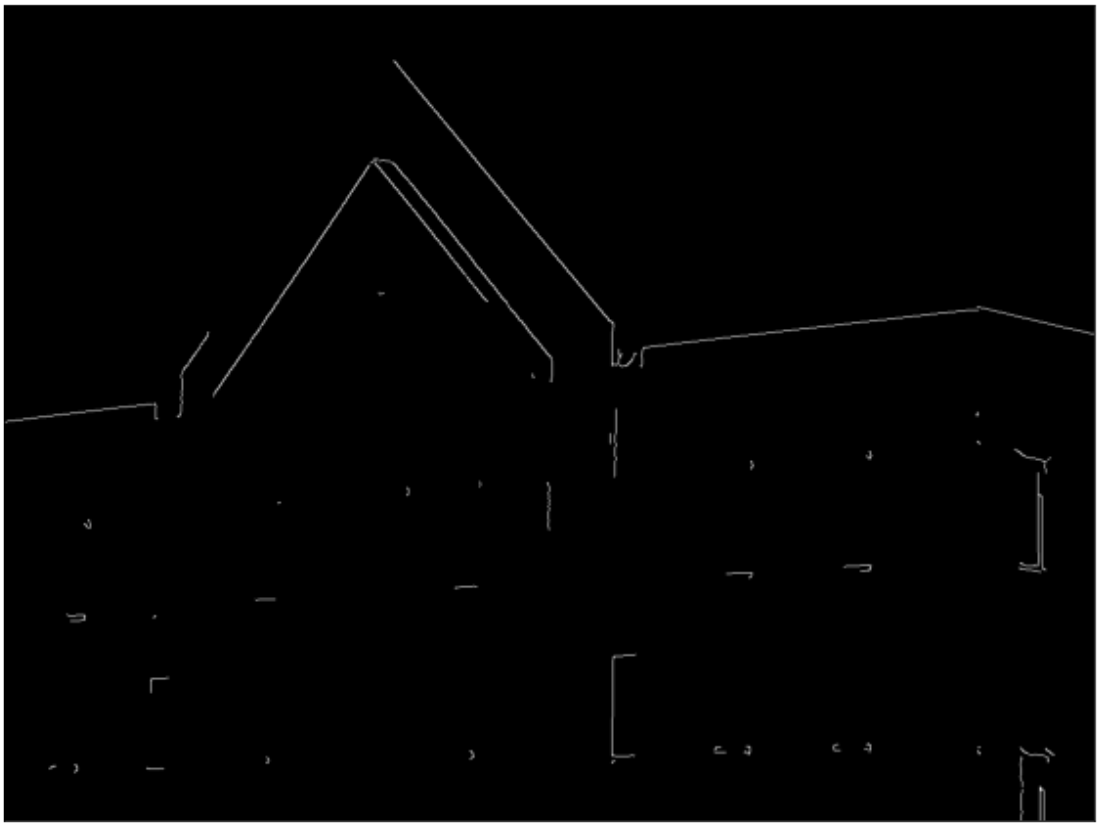
[ ]:

12 9- Canny edge detection with Blur

[26]:
```
for i in range(len(min_threshs)):
    Canny = cv2.Canny(blur, min_threshs[i],max_threshs[i])
    show_image(Canny)
```







10- Analysis of results

The Canny expermint we do two steps first define list of 6 min and max threshold and try it on the image one without blur and the second one with blur

The result as obvious the one without blur has more details as the second one with blur,

When you do not apply blur you should choose greater values of threshes, but that will lead to reduce the neccassoory observations in your image.

When you do apply blur make shur you choose small threshs values beacuse the best result with blur was the first one (75,200)

So make a blur and start with small threshs values.

[ ]: