**Department of Computer Science and Engineering**

**National Institute of Technology, Hamirpur**

**Name: Aryan Puri**

**Roll Number: 22BEC032**

**Digital Image Processing CS-325**

**Solution Laboratory Assignment -4**

**Topic: Image Segmentation (Edge Detection) using Canny Edge Detector**

**Question 1: Design a program to read an image (head.jpg and house.jpg) as given and apply Canny Edge Detector on one of the plane. Display the input image and processed (output) image. Consider low and high threshold as per the requirement or user’s choice. Write the conclusion based on the observation of the output image.**

**Solution:**

**Python Program:**

import cv2

import numpy as np

def normalize(image):

    rows,cols=image.shape

    max=0

    min=255

    for i in range(rows):

        for j in range(cols):

            if(image[i][j]>max):

                max=image[i][j]

            if(image[i][j]<min):

                min=image[i][j]

    newimage=np.zeros((rows,cols),dtype=np.uint8)

    for i in range(rows):

        for j in range(cols):

            newimage[i][j]=round(255\*((image[i][j]-min)/(max-min)))

    return newimage

sd=2

gxy=np.zeros((11,11),dtype=np.float64)

for i in range(11):

    for j in range(11):

        gxy[i][j]=np.exp(-((pow(i-6,2)+pow(j-6,2))/2\*sd\*sd))

img=cv2.imread('./Assignment 4/house.jpg',0)

img=cv2.resize(img,(512,512))

rows,cols=img.shape

f=np.zeros((rows,cols),dtype=np.float64)

maxv=0

minv=255

for i in range(rows):

    for j in range(cols):

        if(img[i][j]>maxv):

            maxv=img[i][j]

        if(img[i][j]<minv):

            minv=img[i][j]

for i in range(rows):

    for j in range(cols):

        f[i][j]=(img[i][j]-minv)/(maxv-minv)

fs=cv2.filter2D(f,-1,gxy)

gx=fs.copy()

gy=fs.copy()

for i in range(rows):

    for j in range(cols-1):

        gx[i][j]=fs[i][j+1]-fs[i][j]

for i in range(1,rows):

    for j in range(cols):

        gy[i][j]=fs[i-1][j]-fs[i][j]

cv2.imshow('The Gx is',normalize(gx))

cv2.imshow('The Gy is',normalize(gy))

Mxy=np.zeros((rows,cols),dtype=np.float64)

Axy=np.zeros((rows,cols),dtype=np.float64)

for i in range(rows):

    for j in range(cols):

        Mxy[i][j]=pow(gx[i][j]\*gx[i][j] +gy[i][j]\*gy[i][j],0.5)

        Axy[i][j]=np.arctan((gy[i][j]/(gx[i][j]+1e-10)))\* (180 / np.pi)

cv2.imshow('The Mxy is',normalize(Mxy))

cv2.imshow('The Axy Is:',normalize(Axy))

gNxy=np.zeros((rows,cols),dtype=np.float64)

def givedirection(angle):

    if angle<-157.5:

        return 'd1'

    elif angle<-112.5:

        return 'd2'

    elif angle<-67.5:

        return 'd3'

    elif angle<-22.5:

        return 'd4'

    elif angle<22.5:

        return 'd1'

    elif angle<67.5:

        return 'd2'

    elif angle<112.5:

        return 'd3'

    elif angle<157.5:

        return 'd4'

    else:

        return 'd1'

for i in range(1,rows-1):

    for j in range(1,cols-1):

        dir=givedirection(Axy[i][j])

        if(dir=='d1'):

            if(Mxy[i][j]>=max(Mxy[i][j-1],Mxy[i][j+1])):

                gNxy[i][j]=Mxy[i][j]

        elif(dir=='d2'):

            if(Mxy[i][j]>=max(Mxy[i-1][j+1],Mxy[i+1][j-1])):

                gNxy[i][j]=Mxy[i][j]

        elif(dir=='d3'):

            if(Mxy[i][j]>=max(Mxy[i-1][j],Mxy[i+1][j])):

                gNxy[i][j]=Mxy[i][j]

        else:

            if(Mxy[i][j]>=max(Mxy[i-1][j-1],Mxy[i+1][j+1])):

                gNxy[i][j]=Mxy[i][j]

Th=int(input('Enter The Value of Higher Threshold:'))

Tl=int(input('Enter The Value of Lower Threshold:'))

gNxy=normalize(gNxy)

cv2.imshow('The GNxy is:',gNxy)

gnl=np.zeros((rows,cols),dtype=np.uint8)

gnh=np.zeros((rows,cols),dtype=np.uint8)

for i in range(rows):

    for j in range(cols):

        if gNxy[i][j]>=Th:

            gnh[i][j]=255

        elif gNxy[i][j]>=Tl:

            gnl[i][j]=255

cv2.imshow('The gnh',gnh)

cv2.imshow('The gnl',gnl)

Edges=gnh.copy()

for i in range(1,rows-1):

    for j in range(1,cols-1):

        if(gnh[i][j]>0):

            Edges[i-1][j-1]=max(Edges[i-1][j-1],gnl[i-1][j-1])

            Edges[i-1][j]=max(Edges[i-1][j],gnl[i-1][j])

            Edges[i-1][j+1]=max(Edges[i-1][j+1],gnl[i-1][j+1])

            Edges[i][j-1]=max(Edges[i][j-1],gnl[i][j-1])

            Edges[i][j+1]=max(Edges[i][j+1],gnl[i][j+1])

            Edges[i+1][j-1]=max(Edges[i+1][j-1],gnl[i+1][j-1])

            Edges[i+1][j]=max(Edges[i+1][j],gnl[i+1][j])

            Edges[i+1][j+1]=max(Edges[i+1][j+1],gnl[i+1][j+1])

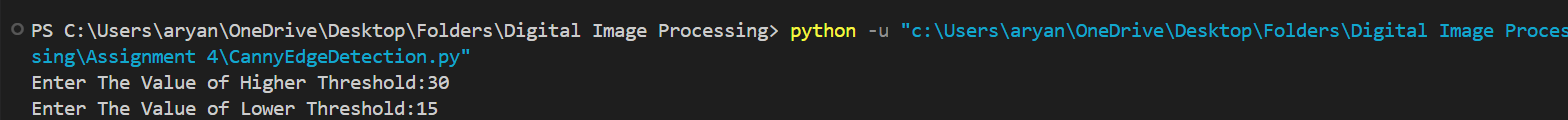
cv2.imshow('Input Image',img)

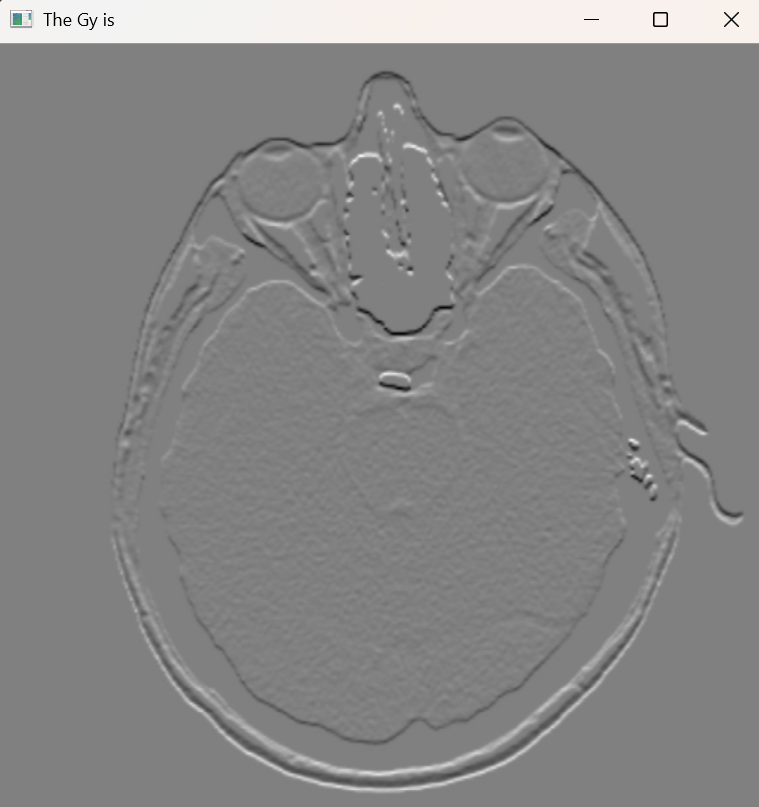
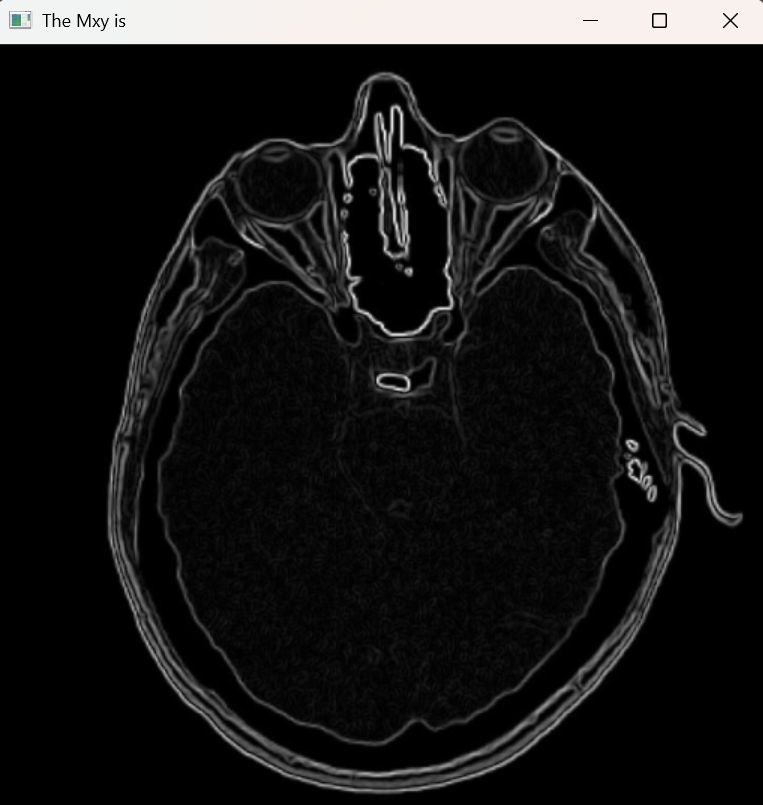
cv2.imshow('Edges',Edges)

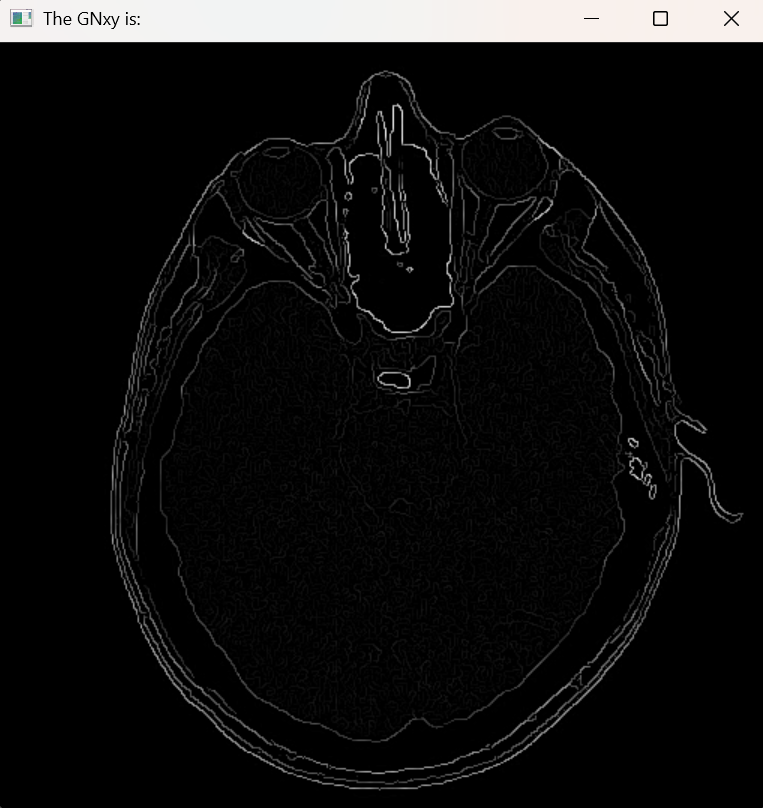
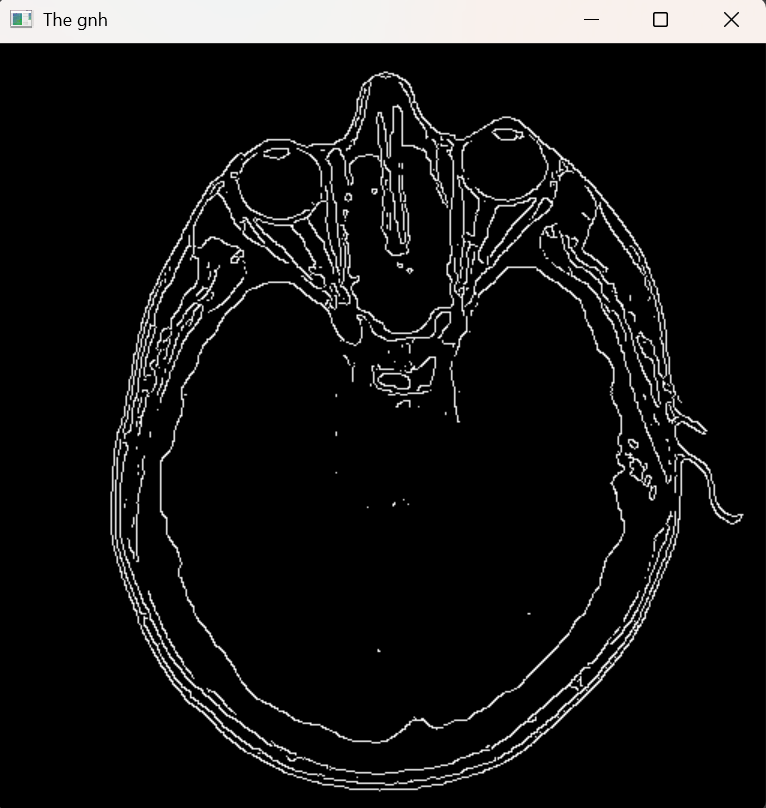
cv2.waitKey(0)

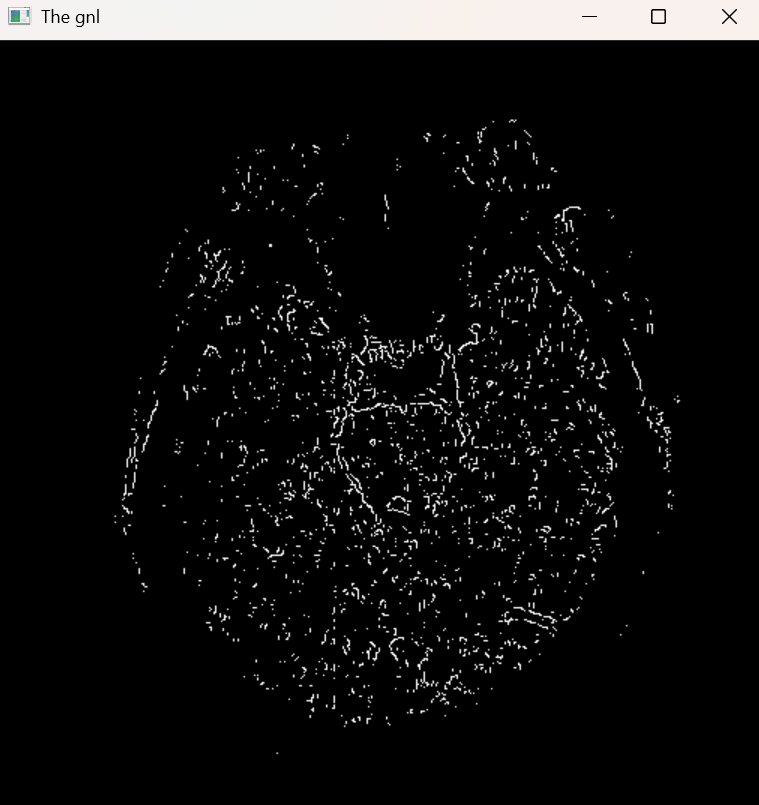
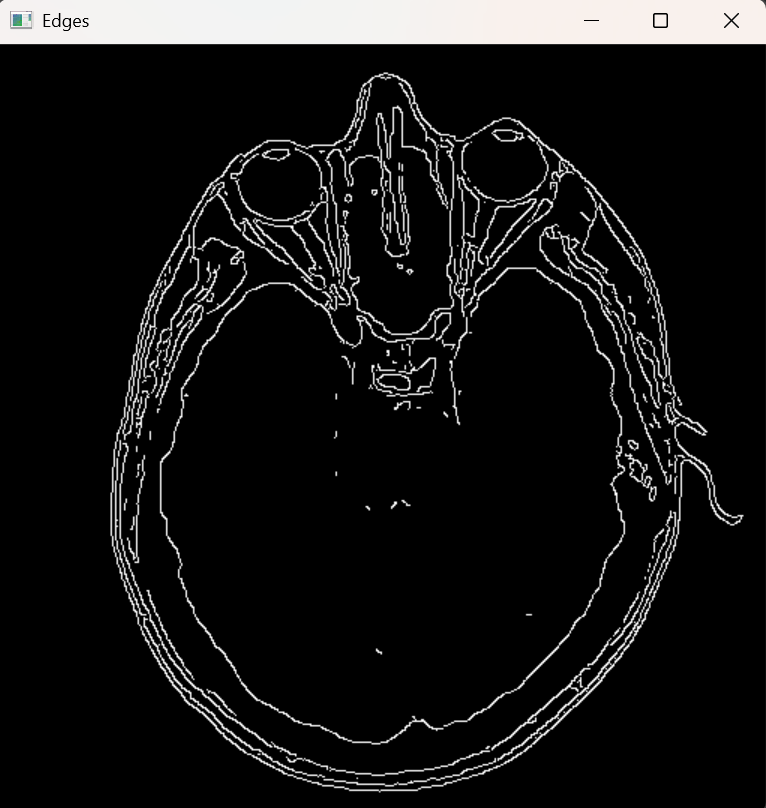
cv2.destroyAllWindows()

**Input Higher and Lower Thresholds:**

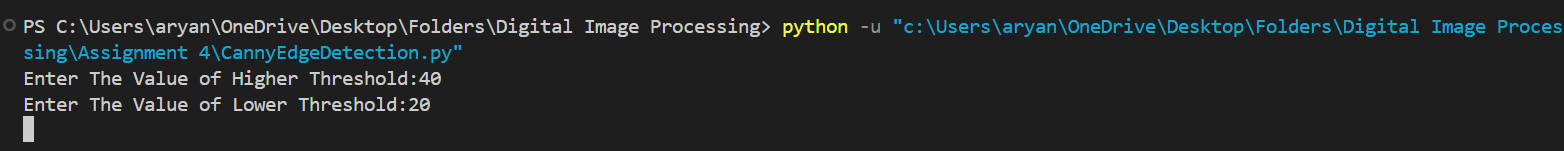


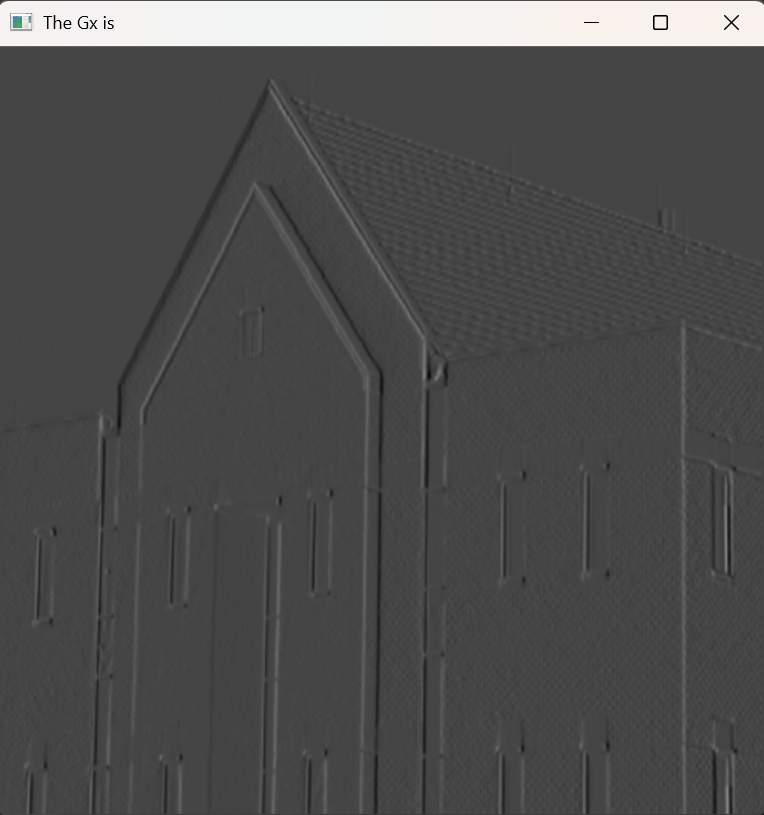
**Input Image Image of First derivative across X-axis** **Image of First derivative across X-axis Image of gradient magnitude**  

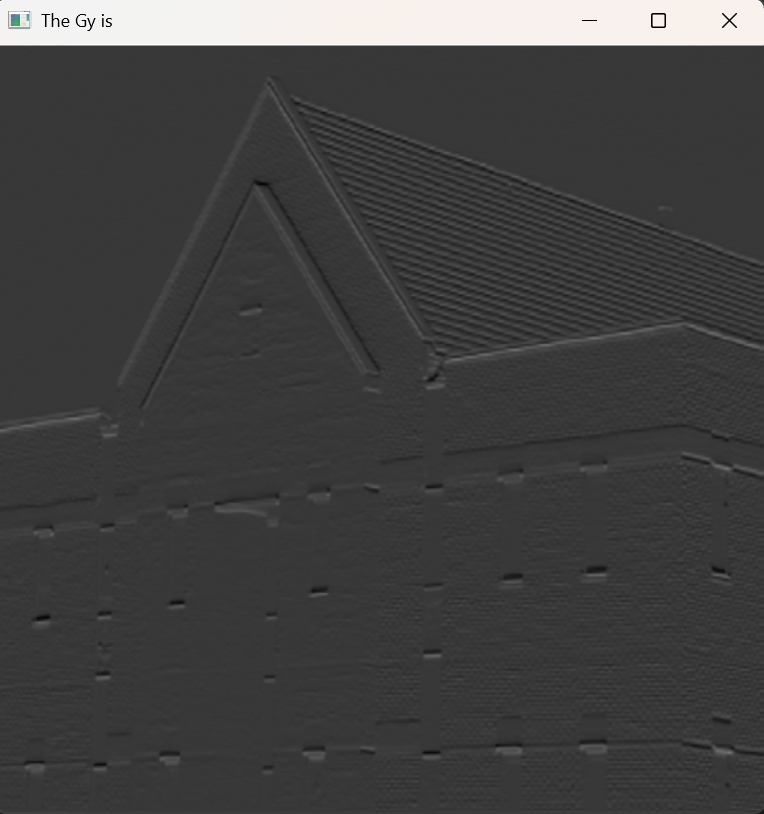
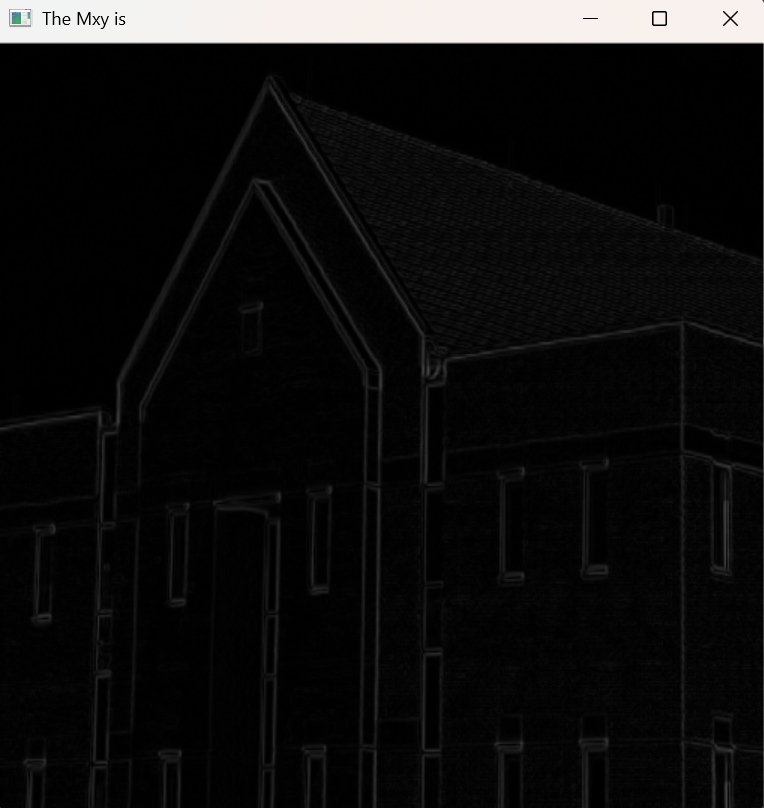
**Image after non maximal supression Image of all Strong points**  

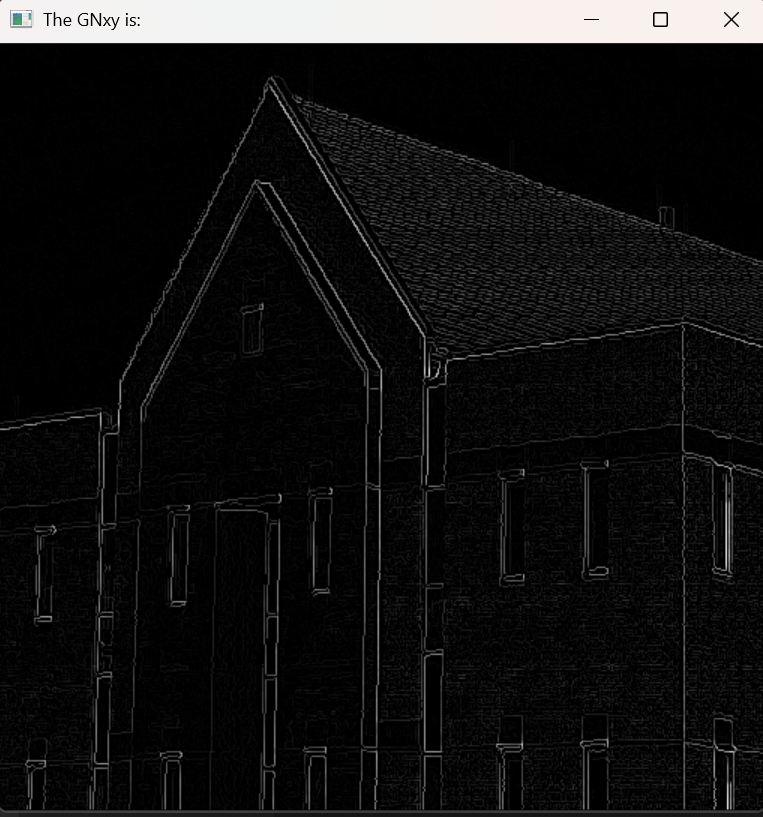
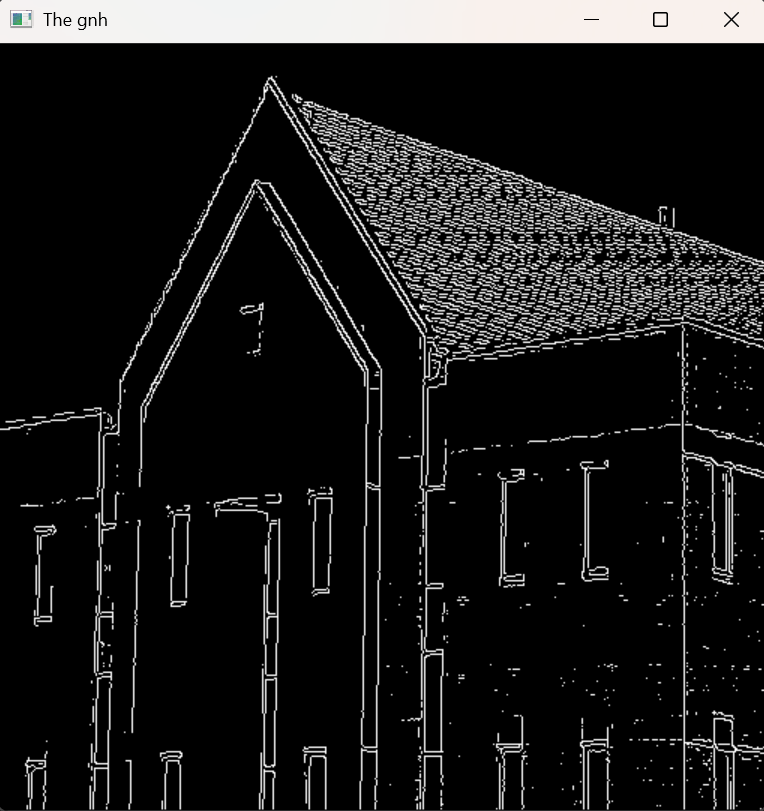
**Image of all weak points Final Edges detected by Canny Edge Detector** 

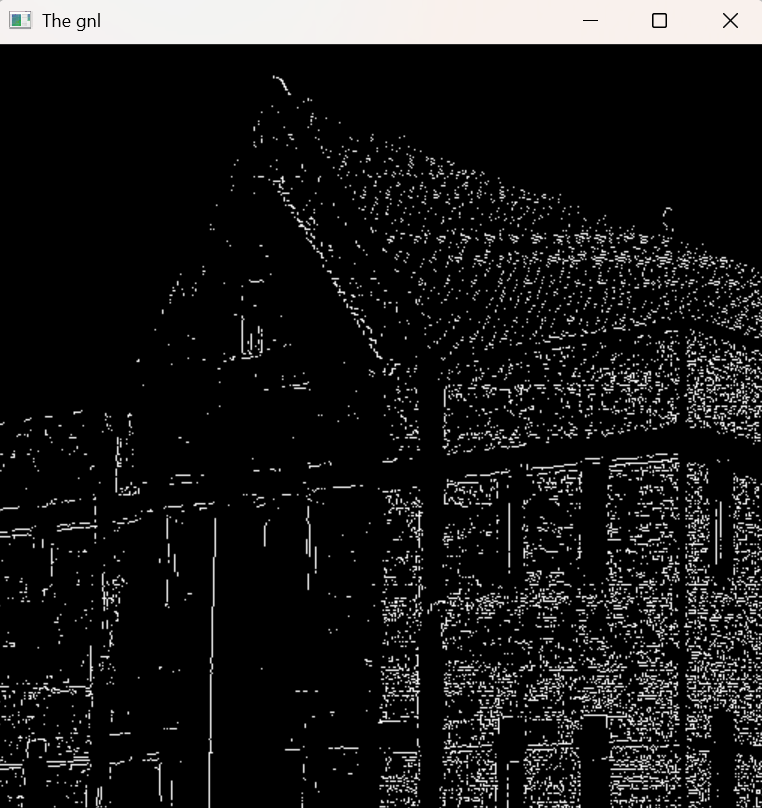
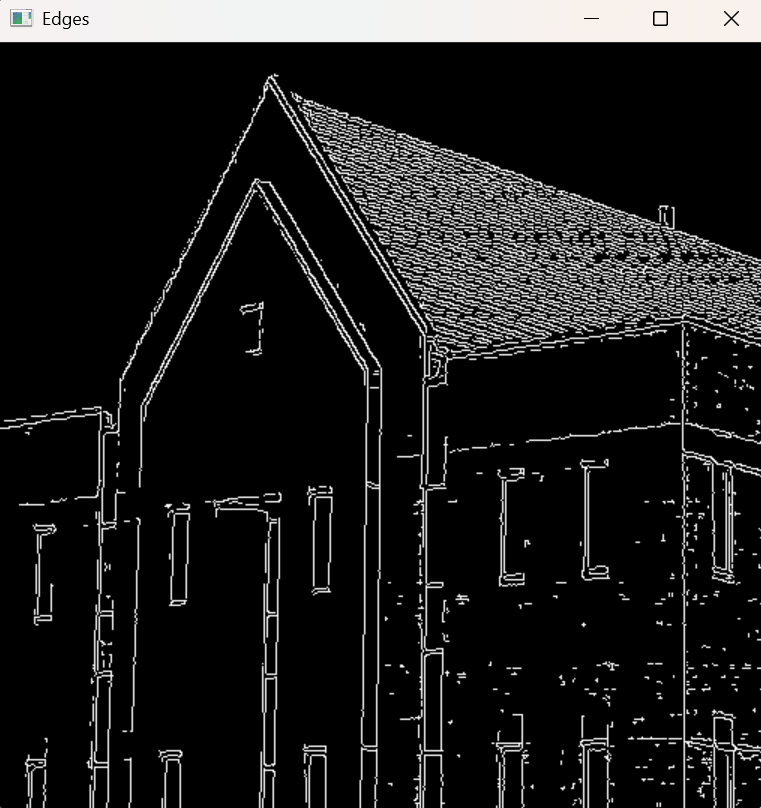
**Input Higher and Lower Thresholds:**



**Input Image Image of First derivative across X-axis ** 

**Image of First derivative across X-axis Image of gradient magnitude**   

**Image after non maximal supression Image of all Strong points**  

**Image of all weak points Final Edges detected by Canny Edge Detector** 

**Conclusion:** The Canny Edge Detector effectively enhances the edges in an image, making object boundaries clearer and more distinct. By adjusting the high and low thresholds, we can control the level of detail captured—higher thresholds focus on strong edges, while lower ones bring out finer details. The method efficiently reduces noise and preserves essential features, making it a reliable technique for edge detection in image processing.