Source Code:

```
from PIL import Image
import numpy as np
import math as mt
from cv2 import COLOR_BGR2GRAY, COLOR_RGB2GRAY, cvtColor, imread, imshow,
waitKey,imwrite


def resize_image(ip_im, filter_size):
    dx, dy = ip_im.shape
    fs = int((filter_size-1)/2)
    opimg1 = dx+2*(fs)
    opimg2 = dy+2*(fs)
    opImg = np.zeros((opimg1,opimg2))

    for i in range(dx):
        for j in range(dy):
            opImg[i+fs][j+fs] = ip_im[i][j]

    for i in range(fs):
        for j in range(fs):
            opImg[i][j] = opImg[fs][fs]

    for i in range(fs):
        for j in range(opimg2-fs, opimg2):
            opImg[i][j] = opImg[fs][opimg2-fs-1]

    for i in range(opimg1-fs, opimg1):
        for j in range(fs):
            opImg[i][j] = opImg[opimg1-fs-1][fs]

    for i in range(opimg1-fs, opimg1):
        for j in range(opimg2-fs, opimg2):
            opImg[i][j] = opImg[opimg1-fs-1][opimg2-fs-1]

    for i in range(fs):
        for j in range(fs, opimg2-fs):
            opImg[i][j] = opImg[fs][j]

    for i in range(opimg1-fs, opimg1):
        for j in range(fs, opimg2-fs):
            opImg[i][j] = opImg[opimg1-fs-1][j]

    for i in range(fs, opimg1-fs):
        for j in range(fs):
            opImg[i][j] = opImg[i][fs]

    for i in range(fs, opimg1-fs):
        for j in range(opimg2-fs, opimg2):
            opImg[i][j] = opImg[i][opimg2-fs-1]
```

```
    return opImg


def img_convolve(ip,filter, filter_size):
    fs = int((filter_size-1)/2)
    ip_r, ip_c = ip.shape
    dx = ip_r - 2*fs
    dy = ip_c - 2*fs
    opImg = np.zeros((dx, dy))
    for i in range(dx):
        for j in range(dy):
            for k in range(filter_size):
                for l in range(filter_size):
                    opImg[i][j] = opImg[i][j] + (filter[k][l] * ip[i+k][j+l])
    return opImg


def sobel_filter(im):
    kernel_x = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]])
    kernel_y = np.array([[1, 2, 1], [0, 0, 0], [-1, -2, -1]])

    og_im = np.array(im)
    dx, dy = og_im.shape
    m_im = resize_image(og_im, 3)
    filter_sizex = int(mt.sqrt(kernel_x.size))
    filter_sizey = int(mt.sqrt(kernel_y.size))
    gx_im = img_convolve(m_im,kernel_x, filter_sizex)
    gy_im = img_convolve(m_im,kernel_y, filter_sizey)

    theta = np.zeros((dx, dy))
    opImg = np.zeros((dx, dy))
    for i in range(dx):
        for j in range(dy):
            opImg[i][j] = mt.sqrt((gx_im[i][j]**2) + (gy_im[i][j]**2))
            if(opImg[i][j]<80):
                opImg[i][j]=0

            theta[i][j] = mt.degrees(mt.atan(gy_im[i][j]/gx_im[i][j]))
    opImg = Image.fromarray(opImg)
    return opImg, theta

"""
Implementation of gaussian filter algorithm
"""
def gaussian_filter(image, k_size, sigma):
    og_im = np.array(image)
    filter = np.zeros((k_size,k_size))
    fs = int((k_size-1)/2)
    dy, dx = np.ogrid[float(-fs):float(fs+1),float(-fs):float(fs+1)]
    sum = 0
    for i in range(k_size):
        for j in range(k_size):
            e = mt.exp((-((dx[0][j]**2)+(dy[i][0]**2))/(2*(sigma**2))))
            filter[i][j] = e*(1/(2*mt.pi*(sigma**2)))
            sum +=  filter[i][j]
```

```
    for i in range(k_size):
        for j in range(k_size):
            filter[i][j] = filter[i][j]/sum
    dx, dy = og_im.shape
    m_im = resize_image(og_im, k_size)
    m_r, m_c = m_im.shape
    filter_size = int(mt.sqrt(filter.size))
    opImg = img_convolve(m_im, filter, filter_size)
    opImg = Image.fromarray(opImg)
    return opImg

def non_max_supp(im, theta):
    og_im = np.array(im)
    dx, dy = og_im.shape
    opImg = np.zeros((dx, dy))
    m_im = resize_image(og_im, 3)
    for i in range(dx):
        for j in range(dy):
            if(float(-30)<=theta[i][j]<float(30) or float(150)<=theta[i][j]<float(-150)):
                if(m_im[i+1][j+1]==max(m_im[i+1][j],m_im[i+1][j+1],m_im[i+1][j+2])):
                    opImg[i][j] = m_im[i+1][j+1]
                else:
                    opImg[i][j] = 0
            if(float(30)<=theta[i][j]<float(60) or float(-150)<=theta[i][j]<float(-120)):
                if(m_im[i+1][j+1]==max(m_im[i+2][j],m_im[i+1][j+1],m_im[i][j+2])):
                    opImg[i][j] = m_im[i+1][j+1]
                else:
                    opImg[i][j] = 0
            if(float(60)<=theta[i][j]<float(120) or float(-120)<=theta[i][j]<float(-60)):
                if(m_im[i+1][j+1]==max(m_im[i][j+1],m_im[i+1][j+1],m_im[i+2][j+1])):
                    opImg[i][j] = m_im[i+1][j+1]
                else:
                    opImg[i][j] = 0
            if(float(120)<=theta[i][j]<float(150) or float(-60)<=theta[i][j]<float(-30)):
                if(m_im[i+1][j+1]==max(m_im[i][j],m_im[i+1][j+1],m_im[i+2][j+2])):
                    opImg[i][j] = m_im[i+1][j+1]
                else:
                    opImg[i][j] = 0
    opImg = opImg.astype(np.uint8)
    opImg = Image.fromarray(opImg)
    return opImg

if __name__ == "__main__":
    str = input('Please enter the image name with format and Path: ')
    sigma = input('Please enter the value of sigma: ')
    size = input('Please enter an odd filter size: ')
    im = Image.open(str)
    size = int(size)
    sigma = float(sigma)
    im = gaussian_filter(im, size, sigma)
    im, theta = sobel_filter(im)
    im = non_max_supp(im, theta)
    im.save('abc2.bmp')
    im.show()
```

Output Images:

Note: We have to wait a little after running the program. it took some time to process so we must wait for 5-10 second for getting these output image as popup.

Gaussian Filter: File=Red.pgm, Gaussian: Sigma = 3, Size = 7

Kunal Goyal – CS558 - Homework1

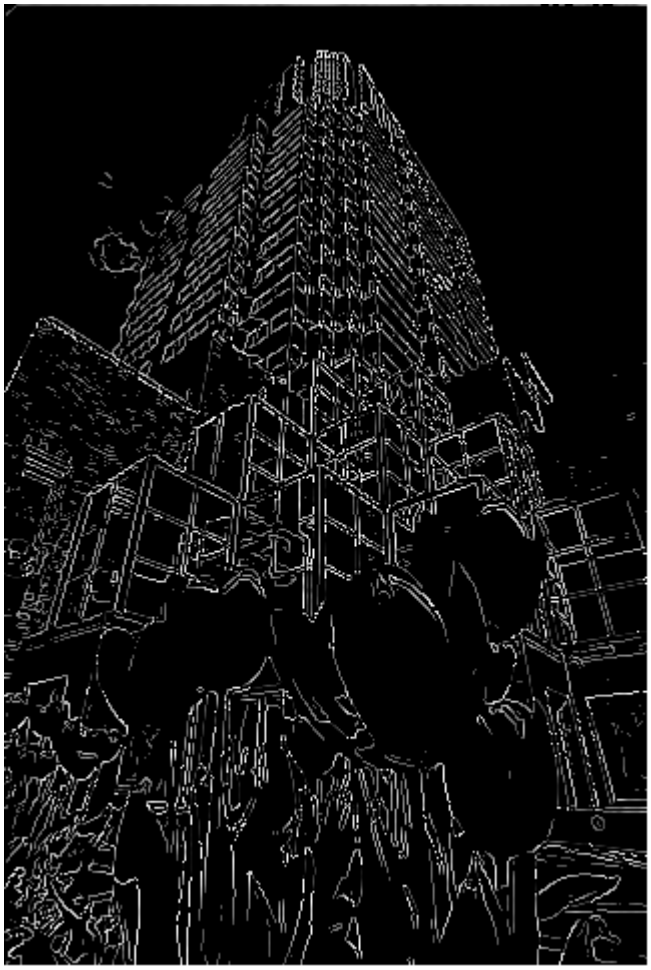
Gaussian Filter: File=plane.pgm, Gaussian: Sigma = 3, Size = 7

Sobel Filter: Kangaroo Gradient: Sigma = 2, Threshold = 75

Sobel Filter: Kangaroo Gradient: Sigma = 1, Threshold = 75

Sobel Filter: red Gradient: Sigma = 3, Threshold = 75

non_max_supp: File = Red Final: Sigma = 1.5, Threshold = 80



non_max_supp: File = Red Final: Sigma = 0.5, Threshold = 80

non_max_supp: File = plane.pgm Final: Sigma = 2.5, Threshold = 80

non_max_supp: File = Plane.pgm Final: Sigma = 3.5, Threshold = 80

non_max_supp: File = kangaroo Final: Sigma = 3.5, Threshold = 80