Project 5

ID：311512040　　name：林胤宏

(a) Source codes:

```
import cv2
import numpy as np
from PIL import Image
import math
from scipy import ndimage

def gaussian_kernel(size, sigma):
    size = int(size) // 2
    x, y = np.mgrid[-size:size+1, -size:size+1]
    normal = 1 / (2.0 * np.pi * sigma**2)
    g =  np.exp(-((x**2 + y**2) / (2.0*sigma**2))) * normal
    return g

def sobel_filters(img):
    Kx = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]], np.float32)
    Ky = np.array([[1, 2, 1], [0, 0, 0], [-1, -2, -1]], np.float32)

    Ix = ndimage.filters.convolve(img, Kx)
    Iy = ndimage.filters.convolve(img, Ky)

    G = np.hypot(Ix, Iy)   #hypot ()返回歐幾里德範數 sqrt(x*x + y*y)
    G = G / G.max()
    theta = np.arctan2(Iy, Ix+1e-8)  #arctan2 的值域是[ - π , π ]

    return (G, theta)

def non_max_suppression(img, D):
    M, N = img.shape
    Z = np.zeros((M,N), dtype=np.float32)
    angle = D * 180. / np.pi
    angle[angle < 0] += 180

    for i in range(1,M-1):
        for j in range(1,N-1):
            #angle 0
            if (0 <= angle[i,j] < 22.5) or (157.5 <= angle[i,j] <= 180):
                q = img[i, j+1]
                r = img[i, j-1]
            #angle 45
            elif (22.5 <= angle[i,j] < 67.5):
                q = img[i+1, j-1]
                r = img[i-1, j+1]
```

```
            #angle 90
            elif (67.5 <= angle[i,j] < 112.5):
                q = img[i+1, j]
                r = img[i-1, j]
            #angle 135
            elif (112.5 <= angle[i,j] < 157.5):
                q = img[i-1, j-1]
                r = img[i+1, j+1]

            if (img[i,j] >= q) and (img[i,j] >= r):
                Z[i,j] = img[i,j]
            else:
                Z[i,j] = 0

    return Z

def threshold(img,lowThreshold,highThreshold):

    M, N = img.shape
    back = np.zeros((M,N), dtype=np.float32)

    weak = np.float32(0.5)
    strong = np.float32(1)

    strong_i, strong_j = np.where(img >= highThreshold)
    zeros_i, zeros_j = np.where(img < lowThreshold)

    weak_i, weak_j = np.where((img <= highThreshold) & (img >= lowThreshold))

    gn_l = np.zeros((img.shape[0], img.shape[1]), dtype=np.float32)
    gn_h = np.zeros((img.shape[0], img.shape[1]), dtype=np.float32)
    gn_l[weak_i, weak_j] = weak
    gn_h[strong_i, strong_j] = strong

    back[strong_i, strong_j] = strong
    back [weak_i, weak_j] = weak

    return (back, gn_l, gn_h)

def hysteresis(img, weak, strong=1):
    M, N = img.shape
    for i in range(1, M-1):
        for j in range(1, N-1):
            if (img[i,j] == weak):
                try:
```

```
                    if ((img[i+1, j-1] == strong) or (img[i+1, j] == strong) or
(img[i+1, j+1] == strong)
                        or (img[i, j-1] == strong) or (img[i, j+1] == strong)
                        or (img[i-1, j-1] == strong) or (img[i-1, j] == strong) or
(img[i-1, j+1] == strong)):
                        img[i, j] = strong
                    else:
                        img[i, j] = 0
                except IndexError as e:
                    pass
    return img

img = cv2.imread('Kid at playground.tif',-1)
cv2.imshow("img",img)
img = img / 255

kernel = gaussian_kernel(30,sigma=4.8)
gaussian = ndimage.filters.convolve(img,kernel)
cv2.imshow("Gaussian",gaussian)

magnitude, theta = sobel_filters(gaussian)
cv2.imshow("magnitude",magnitude)
cv2.imshow("angle",theta)

non_max = non_max_suppression(magnitude, theta)
cv2.imshow("suppression", non_max)

gn, gn_weak, gn_strong = threshold(non_max,0.04,0.1)
# print(gn[:,1])
final = hysteresis(gn, 0.5, 1)   #除了 0.5 以外都濾不掉
# print(final[:,1])
cv2.imshow("gn",gn)
cv2.imshow("gNL",gn_weak)
cv2.imshow("gNH",gn_strong)
cv2.imshow("final",final)
cv2.waitKey(0)

magnitude = magnitude*255
magnitude_save = Image.fromarray(magnitude.astype(np.uint8))
magnitude_save.save("output/magnitude.png",dpi=(200,200))
theta = theta/np.pi*255
theta_save = Image.fromarray(theta.astype(np.uint8))
theta_save.save("output/angle.png",dpi=(200,200))
gn_weak = gn_weak*255
gn_weak_save = Image.fromarray(gn_weak.astype(np.uint8))
```

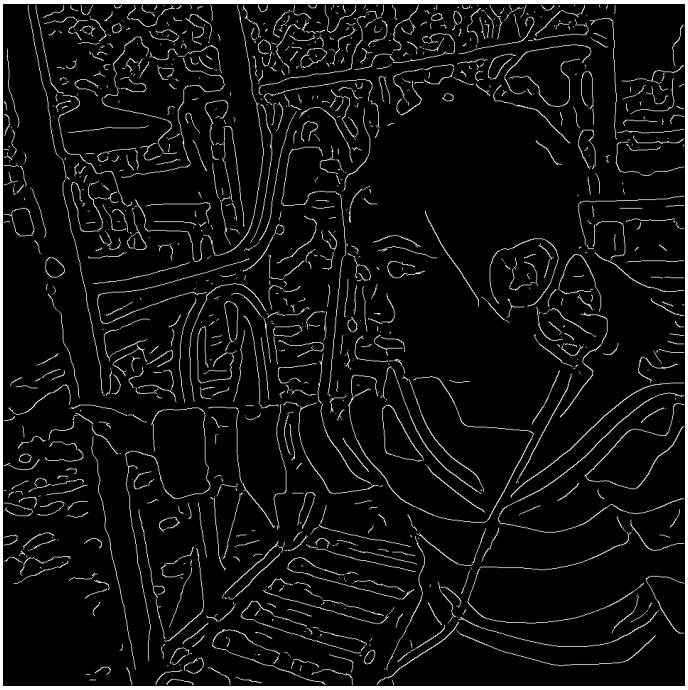
```
gn_weak_save.save("output/gNL.png",dpi=(200,200))
gn_strong = gn_strong*255
gn_strong_save = Image.fromarray(gn_strong.astype(np.uint8))
gn_strong_save.save("output/gNH.png",dpi=(200,200))
gn = gn*255
gn_save = Image.fromarray(gn.astype(np.uint8))
gn_save.save("output/gN.png",dpi=(200,200))
final = final*255
final_save = Image.fromarray(final.astype(np.uint8))
final_save.save("output/final.png",dpi=(200,200))
```

(b) Plot images of the gradient magnitude and gradient angle:

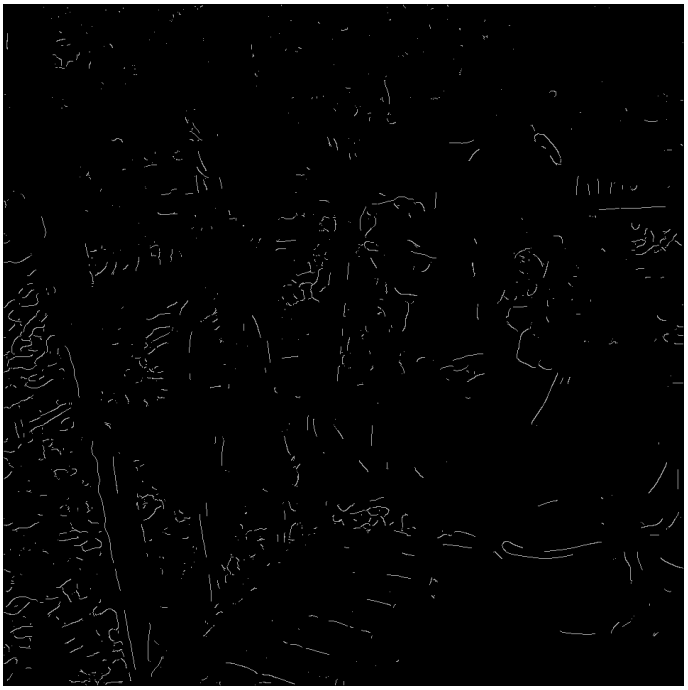
magnitude	angle
	

(c) Plot nonmaxima suppressed image g_N (x,y) as well as images of g_{NL}(x,y) and g_{NH}(x,y):

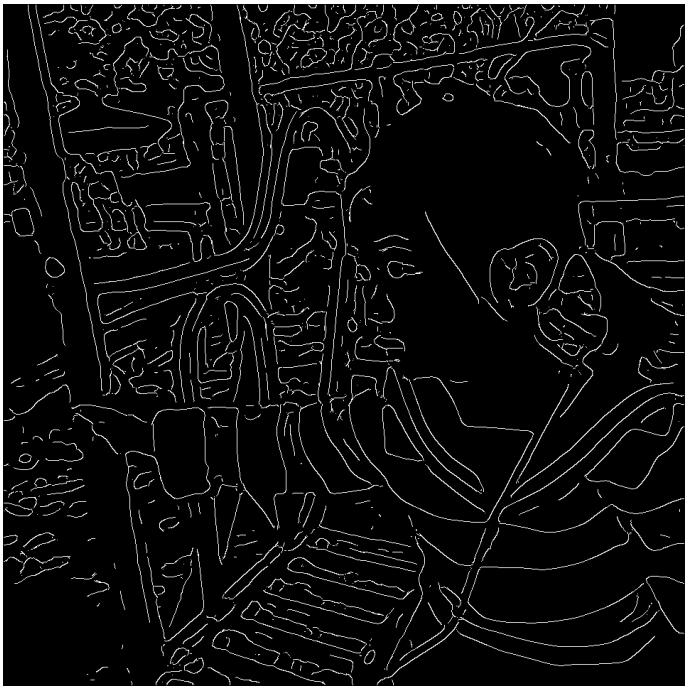
g_N (x,y)



g_{NL}(x,y)



g_{NH}(x,y)



(d) Plot final edge map e(x,y):