Image Processing

Project6 Report

電機碩乙 0850736 楊登宇

code

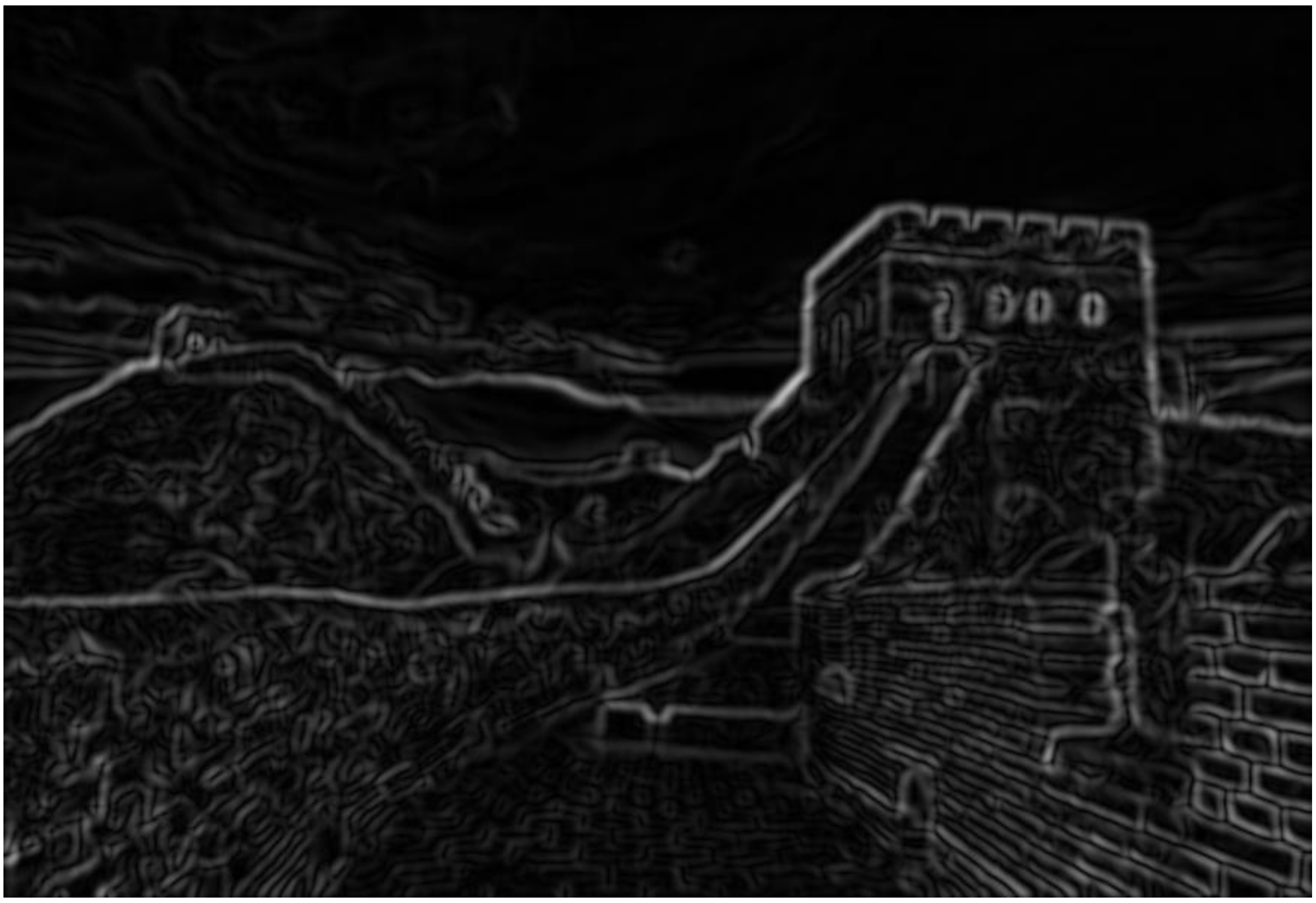
```
import numpy as np
import cv2
import matplotlib.pyplot as plt
import os
import math

def read_file():
   img = cv2.imread('image-pj6(Canny).tif',cv2.IMREAD_GRAYSCALE)
   print("Image size: {:}x{:}".format(img.shape[0],img.shape[1]))
   img = img/255
   sigma = int(min(img.shape[0],img.shape[1]) * (0.5 / 100))
   n = 6 * sigma
   if n % 2 == 0:
      n += 1
   print("Gaussian blur: sigma = {}, n = {}".format(sigma, n))
   blur_img = cv2.GaussianBlur(img,(n,n),sigma)
   show_img(blur_img*255,"blur image")
   return img, blur_img

output_dir = os.path.join('output')
if not os.path.exists(output_dir):
   os.makedirs(output_dir)

def show_img(img,figname):
   path = os.path.join(output_dir,figname+'.png')
   cv2.imwrite(path,img)
   return

def rescale(img):
   scale_img = img.copy()
   scale_img += -np.min(scale_img)
   scale_img /= np.max(scale_img)
   return scale_img
```

```
def sobel_filter(img):
  Gx = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=3)
  Gy = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize=3)
  mag_x = rescale(np.absolute(Gx))
  mag_y = rescale(np.absolute(Gy))
  show_img(mag_x*255,"Sobel Gx")
  show_img(mag_y*255,"Sobel Gy")
  magnitude = cv2.addWeighted(mag_x, 0.5, mag_y, 0.5, 0)
  magnitude_g = np.power((Gx*Gx)+(Gy*Gy), 0.5)
  show_img(magnitude*255, "Gradient Magnitude")
  show_img(magnitude_g*255, "Gradient Magnitude(from G)")

  angle = np.arctan2(Gy,Gx)
  ang_img = angle - np.min(angle)
  ang_img = ang_img / np.max(ang_img) * 255
  show_img(ang_img,"Gradient Angle")
  return magnitude, np.degrees(angle)

def nonmax_suppression(magnitude, angle):
  d1 = np.array([[-1, 0], [1, 0]])
  d2 = np.array([[-1, 1], [1,-1]])
  d3 = np.array([[ 0,-1], [0, 1]])
  d4 = np.array([[-1,-1], [1, 1]])
  d = [d1, d2, d3, d4]

  gradient_vec = np.zeros((angle.shape))
  for i in range(angle.shape[0]):
    for j in range(angle.shape[1]):
      deg = angle[i,j]
      if (-22.5 <= deg and deg <= 22.5) or (157.5 <= deg or deg <= -157.5):
        gradient_vec[i,j] = 0
      elif (-157.5 <= deg and deg <= -112.5) or (22.5 <= deg and deg <= 67.5):
        gradient_vec[i,j] = 1
      elif (-112.5 <= deg and deg <= -67.5) or (67.5 <= deg and deg <= 112.5):
        gradient_vec[i,j] = 2
      else:
        gradient_vec[i,j] = 3
```

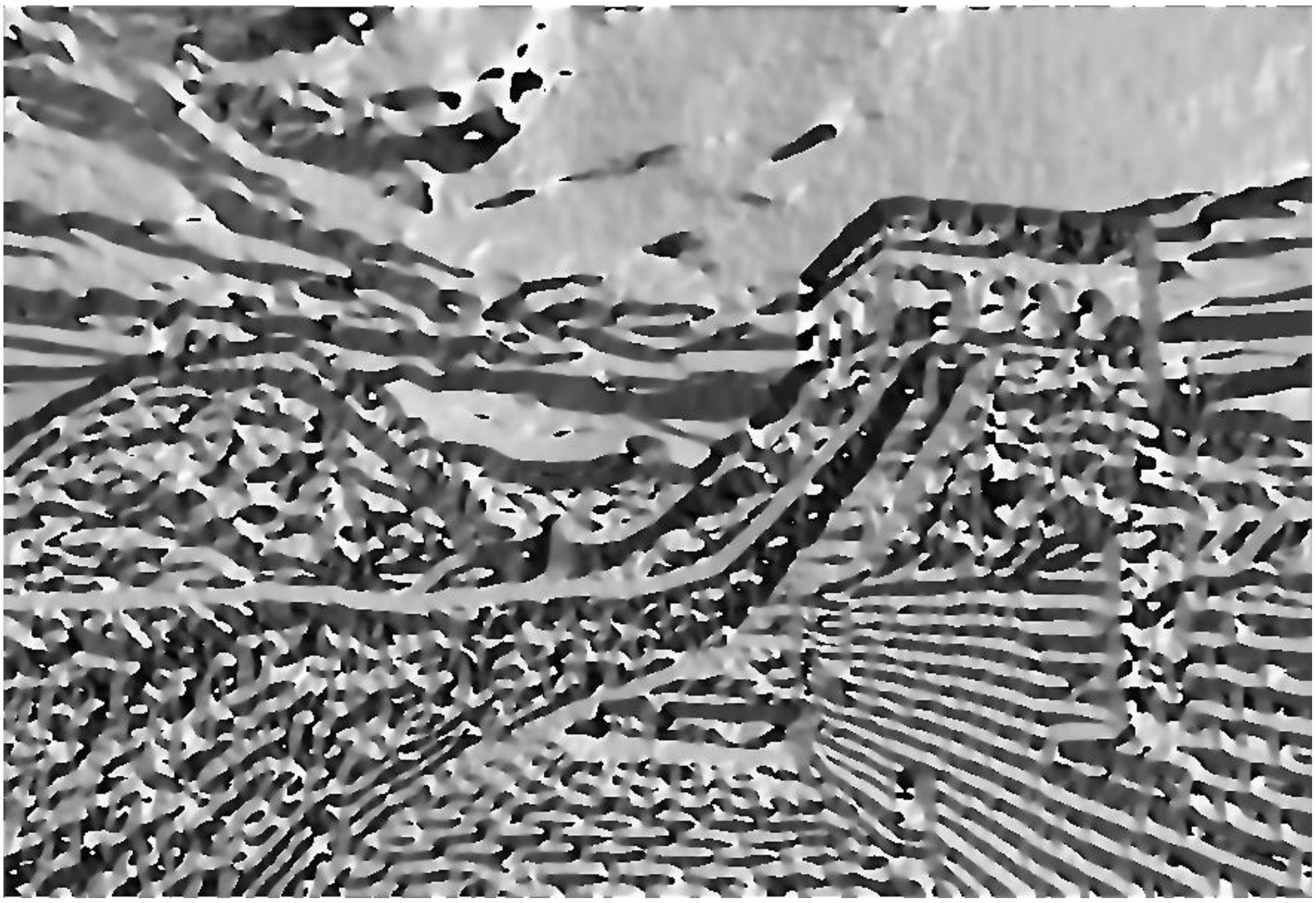
```
   g_n = np.zeros((angle.shape))
   for i in range(angle.shape[0]):
      for j in range(angle.shape[1]):
         a1 = i - d[int(gradient_vec[i,j])][0][1]
         b1 = j + d[int(gradient_vec[i,j])][0][0]
         a2 = i - d[int(gradient_vec[i,j])][1][1]
         b2 = j + d[int(gradient_vec[i,j])][1][0]
         fs1 = -1
         fs2 = -1
         if  (angle.shape[0] > a1) and (a1 >= 0) and (angle.shape[1] > b1) and( b1 >= 0):
            fs1 = magnitude[a1,b1]
         if (angle.shape[0] > a2) and (a2 >= 0) and (angle.shape[1] > b2) and (b2 >= 0):
            fs2 = magnitude[a2,b2]
         if (magnitude[i,j] < fs1) or (magnitude[i,j] < fs2):
            g_n[i,j] = 0
         else:
            g_n[i,j] = magnitude[i,j]

   return g_n

def Hysteresis_threshold(g_n):
   g_nh = np.zeros((g_n.shape))
   g_nl = np.zeros((g_n.shape))
   edge_map = np.zeros((g_n.shape))
   g_n = g_n / np.max(g_n)
   Th = 0.10
   Tl = 0.04
   for i in range(g_n.shape[0]):
      for j in range(g_n.shape[1]):
         if g_n[i,j] >= Th:
            g_nh[i,j] = g_n[i,j]
         elif g_n[i,j] >= Tl:
            g_nl[i,j] = g_n[i,j]

   for i in range(g_n.shape[0]):
      for j in range(g_n.shape[1]):
         if g_nh[i,j] != 0:
                edge_map[i,j] = 1
                for k in range(-1,2):
                    for l in range(-1,2):
                        try:
                            if g_nl[i+k,j+l] != 0:
                                edge_map[i+k,j+l] = 1
                        except:
                            continue

    show_img(rescale(edge_map)*255,"Edge map")
    show_img(rescale(g_nh)*255, "G nh")
    show_img(rescale(g_nl)*255, "G nl")
    return edge_map

if __name__ == "__main__":
    img, blur_img = read_file()
    magnitude, angle = sobel_filter(blur_img)
    g_n = nonmax_suppression(magnitude, angle)
    edge_map = Hysteresis_threshold(g_n)
```

Gradient image

1)Magnitude image

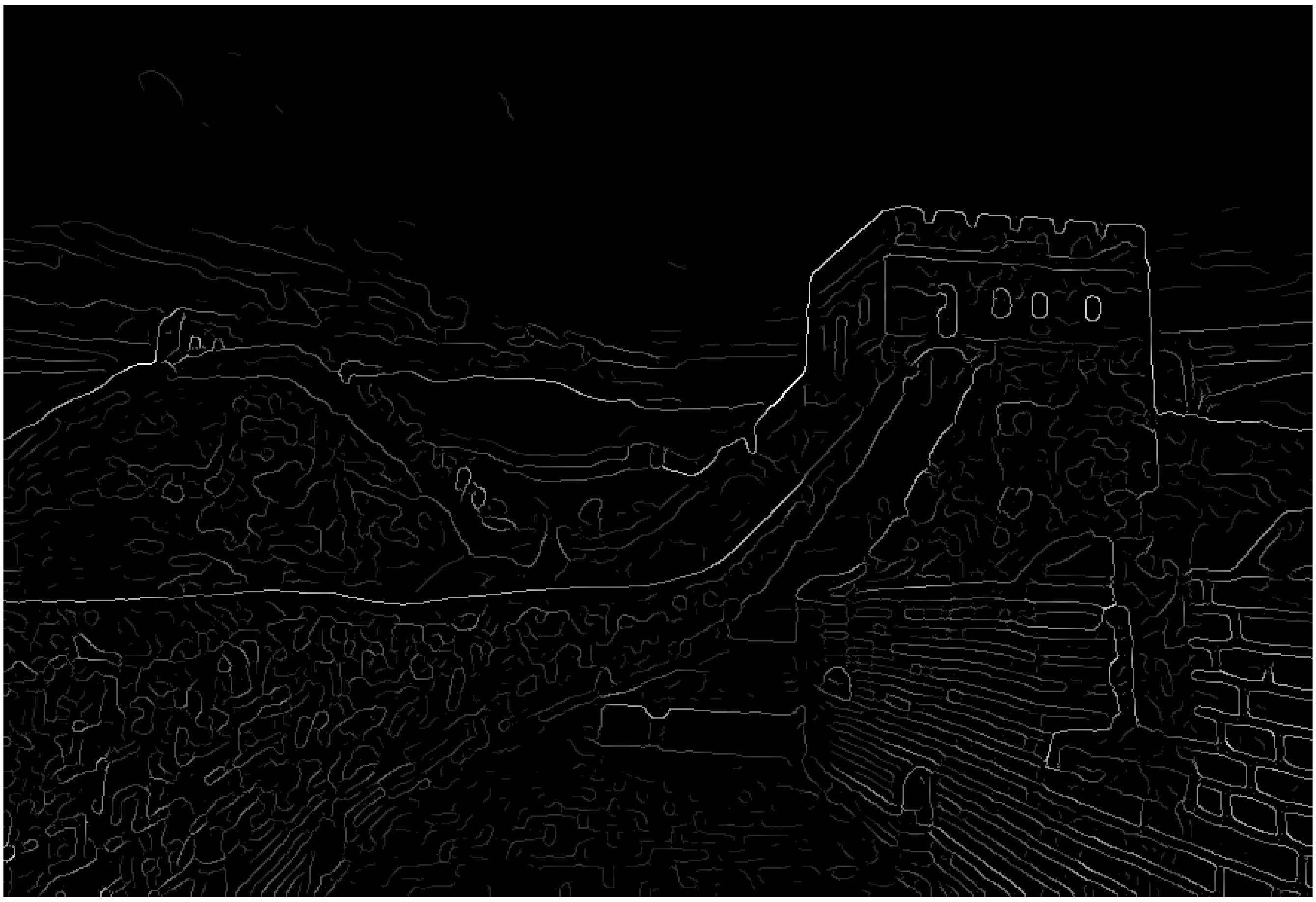


2)Angle image

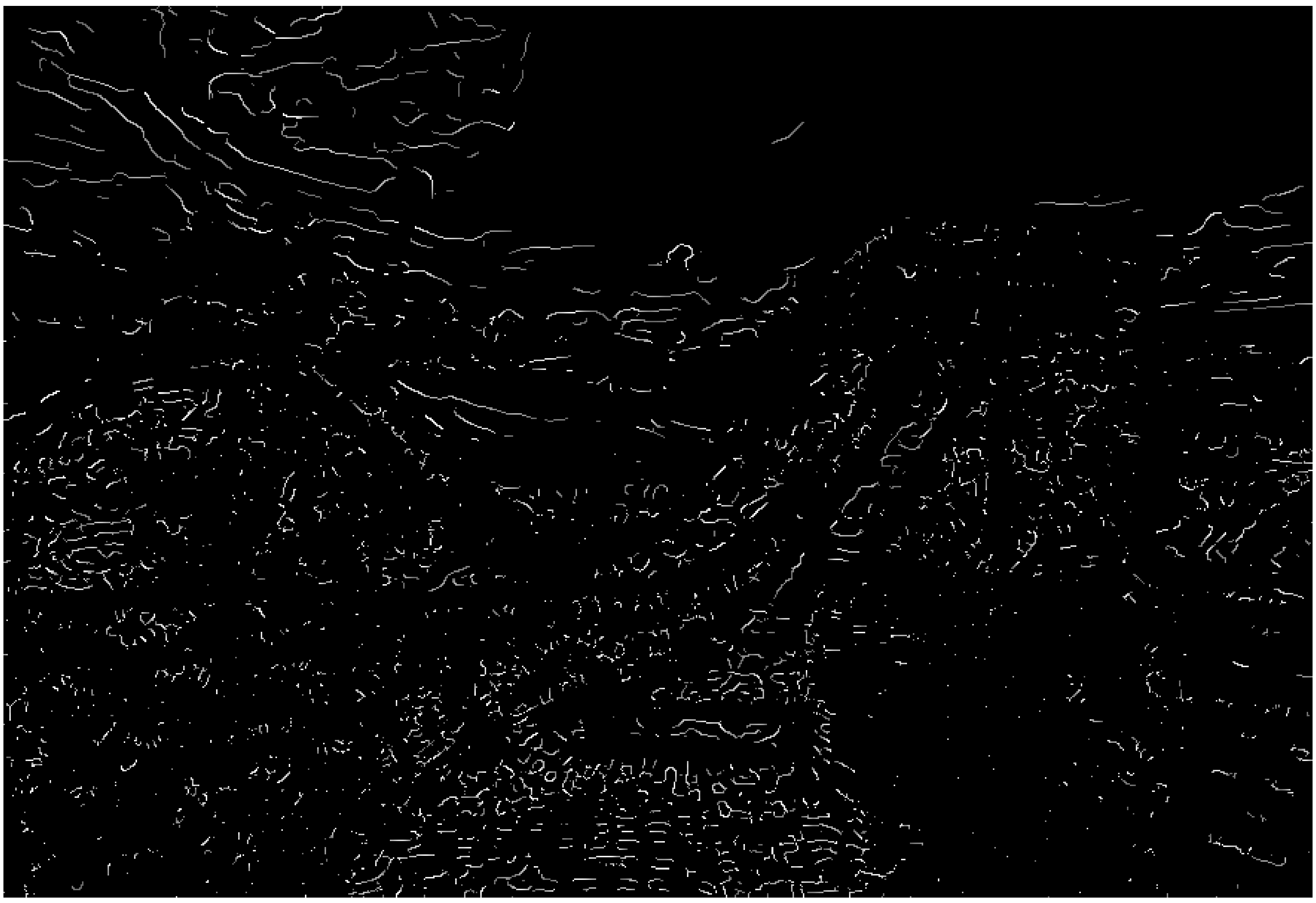


Hysteresis Image

1)g_{NH} image



2) g_{NL} image



Edge map