**台科大 111 學年度「彩色影像處理」 作業一：空間濾波(不得使用現成影像濾波函式)**

**M11125016 蕭強**

**Programming language: Python**

1. 第一題「影像邊緣偵測」：

import numpy as np

import cv2

def processingImg(imagePath):

    image = cv2.imread(imagePath, 0)

    # 4. 將影像轉換成 double 格式，數值範圍在[0 1]之間。

    image = image / 255  # float64

    return image

def edgeDetect(image, kernel):

    # 4. 用雙層迴圈由左而右，由上而下讀取以(x,y)為中心的 3×3 影像區域。

    # 5. 將 3×3 影像區域點對點乘上圖 1 Sobel 濾鏡數值矩陣後，將數值總和存入輸出影像的(x,y) 位置。

    img\_row, img\_col = image.shape

    ker\_row, ker\_col = kernel.shape

    pad = int((ker\_col - 1) / 2)

    image = np.pad(image, (pad, pad), 'symmetric')

    output = np.zeros((img\_row, img\_col), dtype=np.float64)

    for y in range(img\_row):

        for x in range(img\_col):

            cim = image[y:y + 3, x:x + 3]

            result = (cim \* kernel).sum()

            output[y, x] = result

    return output

def embossing(image):

    # 6. 將濾波後的影像加上 0.5，呈現近似圖 2(b)的浮雕影像。

    image = image + 0.5

    return image

def threshold(image, threshold):

    # 7. 分別將濾波後的影像開絕對值，再二值化(門檻值自訂)，用 bitor (bitwise or)或直接相加，產生近似圖 2(c)的輪廓影像。

    # image = abs(image)

    h, w = image.shape

    img\_thres = np.zeros((h, w))

    for y in range(0, h):

        for x in range(0, w):

            pixel = image[y, x]

            if pixel < threshold:

                np\_pix = 0

            else:

                np\_pix = 1

            img\_thres[y, x] = np\_pix

    return img\_thres

# filters

sobel\_hor = np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]])

sobel\_ver = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]])

# load image

raw\_img = processingImg('ntust\_gray.jpg')

# 4. 用雙層迴圈由左而右，由上而下讀取以(x,y)為中心的 3×3 影像區域。

# 5. 將 3×3 影像區域點對點乘上圖 1 Sobel 濾鏡數值矩陣後，將數值總和存入輸出影像的(x,y) 位置。

sobel\_hor\_img = edgeDetect(raw\_img, sobel\_hor)

sobel\_ver\_img = edgeDetect(raw\_img, sobel\_ver)

# 6. 將濾波後的影像加上 0.5，呈現近似圖 2(b)的浮雕影像。

sobel\_hor\_emb\_img = embossing(sobel\_hor\_img)

sobel\_ver\_emb\_img = embossing(sobel\_ver\_img)

# 7. 分別將濾波後的影像開絕對值，再二值化(門檻值自訂)，用 bitor (bitwise or)或直接相加，產生近似圖 2(c)的輪廓影像。

sobel\_hor\_thres\_img = threshold(sobel\_hor\_img, 0.3)

sobel\_ver\_thres\_img = threshold(sobel\_ver\_img, 0.3)

doubel\_sobel\_img = sobel\_hor\_thres\_img + sobel\_ver\_thres\_img

sobel\_hor\_emb\_img = sobel\_hor\_emb\_img \* 255

sobel\_ver\_emb\_img =sobel\_ver\_emb\_img \* 255

doubel\_sobel\_img = doubel\_sobel\_img \* 255

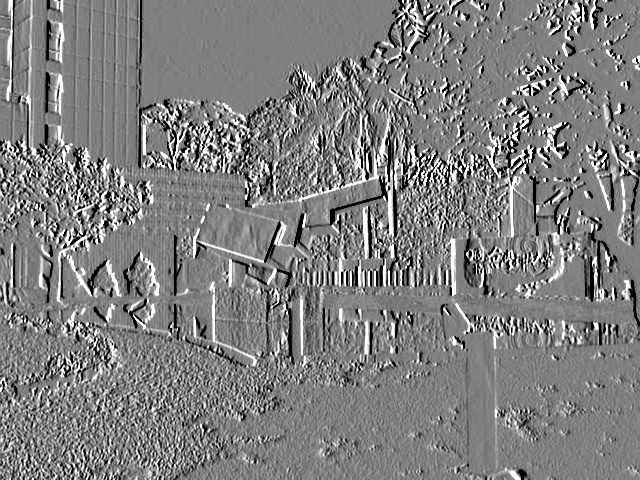
cv2.imwrite("sobel\_hor\_emb\_img.jpg", sobel\_hor\_emb\_img)

cv2.imwrite("sobel\_ver\_emb\_img.jpg", sobel\_ver\_emb\_img)

cv2.imwrite("doubel\_sobel\_img.jpg", doubel\_sobel\_img)

**A picture containing outdoor, water, nature, flock

Description automatically generated**

圖表 1水平濾波浮雕處理****

圖表 2垂直濾波浮雕處理

A picture containing text, crowd

Description automatically generated

圖表 3雙向濾波輪廓處理

1. 第二題「Unsharp Masking(USM)影像銳化」

import numpy as np

import cv2

def processingImg(imagePath):

    image = cv2.imread(imagePath, 0)

    # 4. 將影像轉換成 double 格式，數值範圍在[0 1]之間。

    image = image/255  # float64

    return image

def unsharpMask(image, kernel):

    # 5. 用雙層迴圈對 n x n 濾鏡(均值濾鏡或高斯濾鏡)做影像模糊化，獲得近似圖3(b)的結果。

    img\_row, img\_col = image.shape

    ker\_row, ker\_col = kernel.shape

    pad = int((ker\_col - 1) / 2)

    image = np.pad(image, (pad, pad), 'symmetric')

    output = np.zeros((img\_row, img\_col), dtype=np.float64)

    for y in range(img\_row):

        for x in range(img\_col):

            cim = image[y:y + 3, x:x + 3]

            result = (cim \* kernel).sum()

            output[y, x] = result

    return output

# filters

mean\_filter = np.ones((3,3))/9

gaussian\_Filter = np.array([[1, 2, 1], [2, 4, 2], [1, 2, 1]]) / 16

# load image

raw\_img = processingImg('ntust\_gray.jpg')

# 5. 用雙層迴圈對 n x n 濾鏡(均值濾鏡或高斯濾鏡)做影像模糊化，獲得近似圖3(b)的結果。

blur\_image = unsharpMask(raw\_img, gaussian\_Filter)

# 6. 利用原圖與模糊影像的差異，加上原圖，獲得類似圖 3(c)的銳利影像。

usm\_image = 0.8 \* (raw\_img - blur\_image) + raw\_img  # 0.8(raw - processed) + raw

raw\_img = raw\_img \* 255

blur\_image = blur\_image \* 255

usm\_image = usm\_image \* 255

cv2.imwrite("gaussian\_Filter\_image.jpg", blur\_image)

cv2.imwrite("gaussian\_usm\_image.jpg", usm\_image)

A picture containing outdoor, tree, white, black

Description automatically generated

圖表 4均值濾波處理A picture containing outdoor, tree, white, black

Description automatically generated

圖表 5高斯濾波處理A picture containing text, tree, outdoor, black

Description automatically generated

圖表 6均值濾波銳利化

A picture containing tree, outdoor, white, black

Description automatically generated

圖表 7高斯濾波銳利化