Homework6

R08525116 吳承哲

Part 1

Trapezoidal Transformation

```
def Trapezoidal_Transform(img):
    rows, cols = img.shape
    new_img = np.zeros(img.shape, dtype=img.dtype)

    for i in range(rows):
        for j in range(cols):
            new_x = int(np.round(3*i/4 + j*i/(cols*rows
)))
            new_y = int(np.round(j+i/4 - j*i/(2*cols)))
            new_img[new_x][new_y] = img[i][j]
    return new_img
```

Wavy Transformation

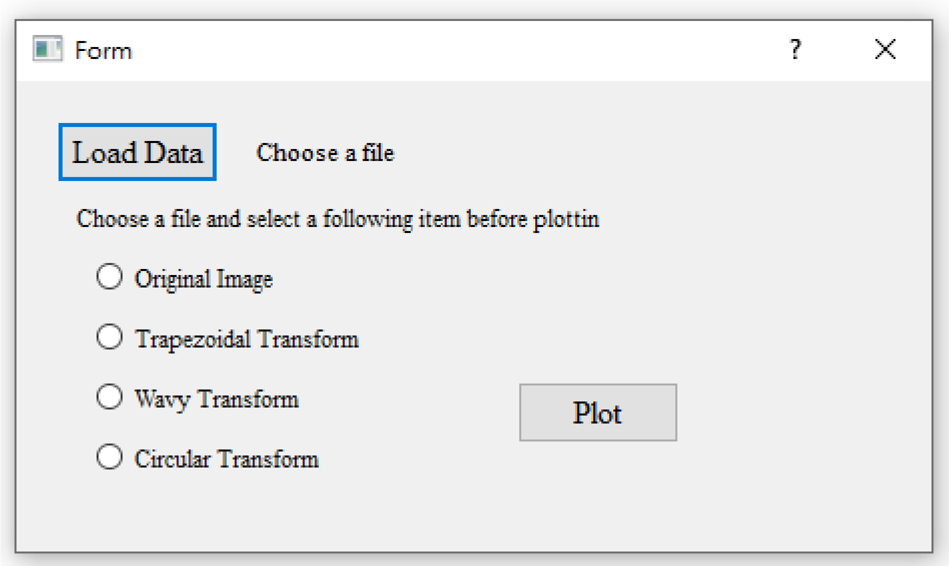
```
def Wavy_Transform(img):
    rows, cols = img.shape
    new_img = np.zeros(img.shape, dtype=img.dtype)

    for i in range(rows):
        for j in range(cols):
            new_x = int(np.round(j - 32*np.sin(i/32)))
            new_y = int(np.round(i - 32*np.sin(j/32)))
            if new_x >= 0 and new_x <= rows-
1 and new_y >= 0 and new_y <= cols-1:
                new_img[j][i] = img[new_x][new_y]
    return new_img
```

Circular Transformation

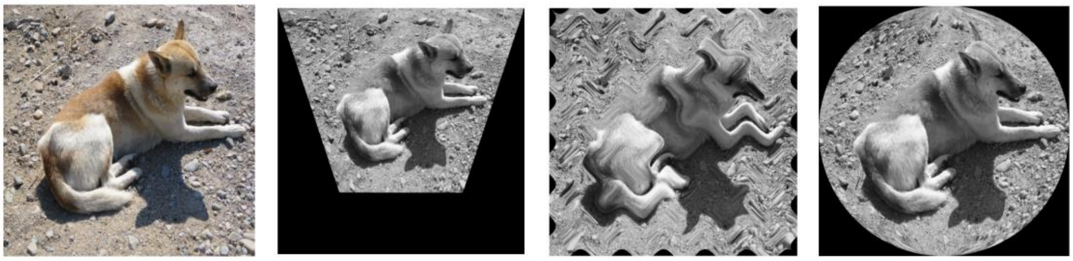
```
def Circular_Transform(img):
    rows, cols = img.shape
    new_img = np.zeros(img.shape, dtype=img.dtype)

    for i in range(cols):
        for j in range(rows):
            d = np.sqrt((rows/2)**2 - (rows/2 - i)**2)
            new_x = np.round((j - cols/2)*cols/(d*2) + cols/2)
            new_y = i
            if new_x >= 0 and new_x <= cols-1 and new_y >= 0 and new_y <= cols-1:
                new_img[i][j] = img[new_y][int(new_x)]
    return new_img
```

GUI

執行 part1.py



Original / Trapezoidal / Wavy / Circular



幾何轉換為利用座標轉換來使圖像產生變形。

Trapezoidal Transformation： 透過座標轉換轉為梯形的圖片。

Wavy Transformation： 利用 sin 函數使圖形轉為 sin 波的形狀。

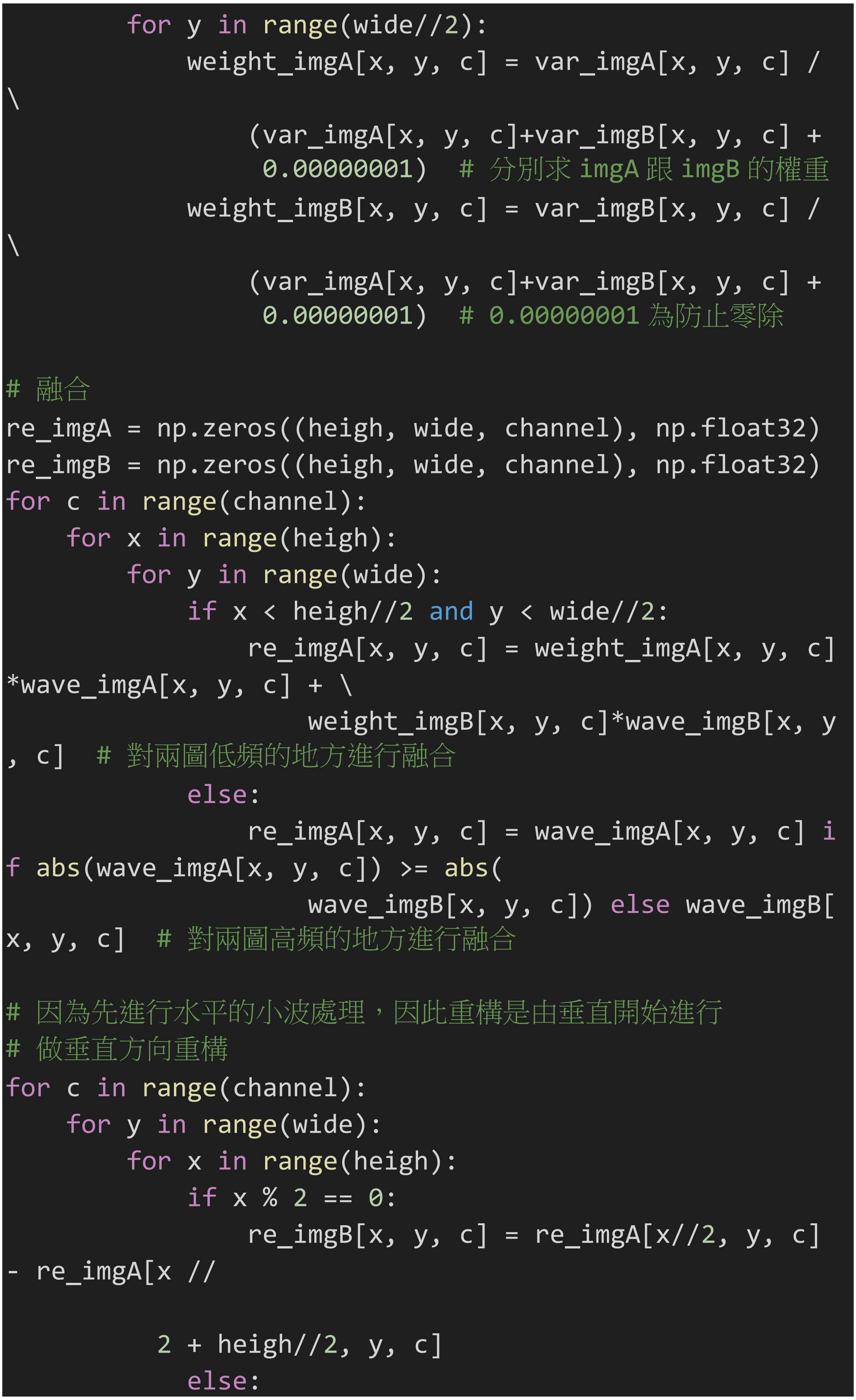
Circular Transformation： 計算每列的三角函數，將每列的像素壓成圓形的排列。

Part 2

```
imgA = cv2.imread("./Image Set 1/clock1.JPG")
imgB = cv2.imread("./Image Set 1/clock2.JPG")
imgA = cv2.cvtColor(imgA, cv2.COLOR_BGR2RGB)
```

```
imgB = cv2.cvtColor(imgB, cv2.COLOR_BGR2RGB)


heigh, wide, channel = imgA.shape

tmp1 = []
tmp2 = []
tmp3 = []
tmp4 = []

wave_imgA = np.zeros((heigh, wide, channel), np.float32)  # 儲存小波處理後的imgA
wave_imgB = np.zeros((heigh, wide, channel), np.float32)  # 儲存小波處理後的imgB
# 對圖片RGB通道做水平方向的小波處理
for c in range(channel):
    for x in range(heigh):
        for y in range(0, wide, 2):
            # 將imgA處理後的低頻存在tmp1
            tmp1.append((float(imgA[x, y, c]) + float(imgA[x, y+1, c]))/2)
            # 將imgA處理後的高頻存在tmp2
            tmp2.append(
                (float(imgA[x, y, c]) + float(imgA[x, y+1, c]))/2 - float(imgA[x, y, c]))
            # 將imgB處理後的低頻存在tmp3
            tmp3.append((float(imgB[x, y, c]) + float(imgB[x, y+1, c]))/2)
            # 將imgB處理後的高頻存在tmp4
            tmp4.append(
                (float(imgB[x, y, c]) + float(imgB[x, y+1, c]))/2 - float(imgB[x, y, c]))
        tmp1 = tmp1 + tmp2  # 將imgA處理後的數據全部存在tmp1
        tmp3 = tmp3 + tmp4  # 將imgB處理後的數據全部存在tmp3
        for i in range(len(tmp1)):
            wave_imgA[x, i, c] = tmp1[i]  # 前半段為低頻，後半為高頻
            wave_imgB[x, i, c] = tmp3[i]  # 前半段為低頻，後半為高頻
        tmp1 = []
        tmp2 = []
        tmp3 = []
        tmp4 = []

# 對圖片 RGB 通道做垂直方向的小波處理
for c in range(channel):
    for y in range(wide):
        for x in range(0, heigh-1, 2):
            tmp1.append(
                (float(wave_imgA[x, y, c]) + float(wave_imgA[x+1, y, c]))/2)
            tmp2.append(
                (float(wave_imgA[x, y, c]) + float(wave_imgA[x+1, y, c]))/2 - float(wave_imgA[x, y, c]))
            tmp3.append(
                (float(wave_imgB[x, y, c]) + float(wave_imgB[x+1, y, c]))/2)
            tmp4.append(
                (float(wave_imgB[x, y, c]) + float(wave_imgB[x+1, y, c]))/2 - float(wave_imgB[x, y, c]))
        tmp1 = tmp1 + tmp2
        tmp3 = tmp3 + tmp4
        for i in range(len(tmp1)):
            wave_imgA[i, y, c] = tmp1[i]
            wave_imgB[i, y, c] = tmp3[i]
        tmp1 = []
        tmp2 = []
        tmp3 = []
        tmp4 = []

# 求以 x,y 為中心的 5x5 矩陣的方差
var_imgA = np.zeros((heigh//2, wide//2, channel),
                    np.float32)
var_imgB = np.zeros((heigh//2, wide//2, channel),
                    np.float32)
```

```
for c in range(channel):
    for x in range(heigh//2):
        for y in range(wide//2):
            # 對圖片邊界做處理
            if x - 3 < 0:
                up = 0
            else:
                up = x - 3
            if x + 3 > heigh//2:
                down = heigh//2
            else:
                down = x + 3
            if y - 3 < 0:
                left = 0
            else:
                left = y - 3
            if y + 3 > wide//2:
                right = wide//2
            else:
                right = y + 3
            # 求 imgA 以 x,y 為中心的 5x5 矩陣的方差，mean 表示平均值，var 表示方差
            meanA, varA = cv2.meanStdDev(wave_imgA[up:down, left:right, c])
            meanB, varB = cv2.meanStdDev(
                wave_imgB[up:down, left:right, c])  # 求 imgB 以 x,y 為中心的 5x5 矩陣的方差，

            var_imgA[x, y, c] = varA
            var_imgB[x, y, c] = varB

# 求兩圖的權重
weight_imgA = np.zeros((heigh//2, wide//2, channel), np.float32)
weight_imgB = np.zeros((heigh//2, wide//2, channel), np.float32)
for c in range(channel):
    for x in range(heigh//2):
```

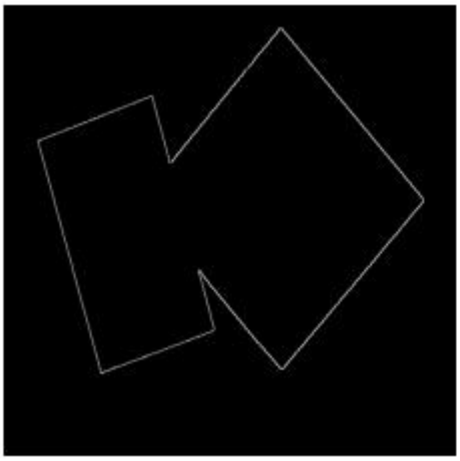


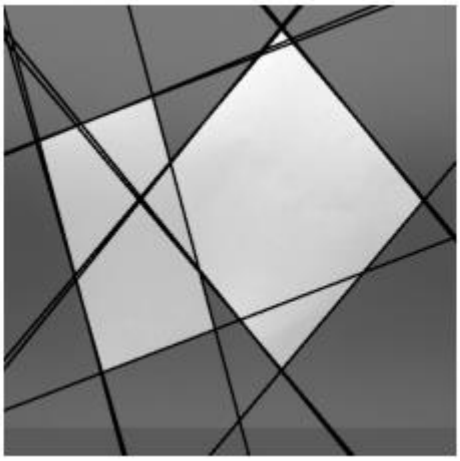
```
        for y in range(wide//2):
            weight_imgA[x, y, c] = var_imgA[x, y, c] /
\
                (var_imgA[x, y, c]+var_imgB[x, y, c] +
                 0.00000001)  # 分別求 imgA 跟 imgB 的權重
            weight_imgB[x, y, c] = var_imgB[x, y, c] /
\
                (var_imgA[x, y, c]+var_imgB[x, y, c] +
                 0.00000001)  # 0.00000001 為防止零除

# 融合
re_imgA = np.zeros((heigh, wide, channel), np.float32)
re_imgB = np.zeros((heigh, wide, channel), np.float32)
for c in range(channel):
    for x in range(heigh):
        for y in range(wide):
            if x < heigh//2 and y < wide//2:
                re_imgA[x, y, c] = weight_imgA[x, y, c]
*wave_imgA[x, y, c] + \
                    weight_imgB[x, y, c]*wave_imgB[x, y
, c]  # 對兩圖低頻的地方進行融合
            else:
                re_imgA[x, y, c] = wave_imgA[x, y, c] i
f abs(wave_imgA[x, y, c]) >= abs(
                    wave_imgB[x, y, c]) else wave_imgB[
x, y, c]  # 對兩圖高頻的地方進行融合

# 因為先進行水平的小波處理，因此重構是由垂直開始進行
# 做垂直方向重構
for c in range(channel):
    for y in range(wide):
        for x in range(heigh):
            if x % 2 == 0:
                re_imgB[x, y, c] = re_imgA[x//2, y, c]
- re_imgA[x //

          2 + heigh//2, y, c]
            else:
                re_imgB[x, y, c] = re_imgA[x//2, y, c]
+ re_imgA[x//2 +

          heigh//2, y, c]

# 做水平重構
for c in range(channel):
    for x in range(heigh):
        for y in range(wide):
            if y % 2 == 0:
                re_imgA[x, y, c] = re_imgB[x, y//2, c]
- \
                    re_imgB[x, y//2 + wide//2, c]
            else:
                re_imgA[x, y, c] = re_imgB[x, y//2, c]
+ \
                    re_imgB[x, y//2 + wide//2, c]

re_imgA[re_imgA[:, :, :] < 0] = 0
re_imgA[re_imgA[:, :, :] > 255] = 255

re_imgA = re_imgA.astype(np.uint8)

plt.subplot(131)
plt.axis("off")
plt.imshow(imgA)
plt.subplot(132)
plt.axis("off")
plt.imshow(imgB)
plt.subplot(133)
plt.axis("off")
plt.imshow(re_imgA)
plt.show()
```

執行 part2.py

Part3

```
img = cv2.imread("./rects.bmp")
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
row, col = img.shape

edges = cv2.Canny(img, 50, 150)

lines = cv2.HoughLines(edges, 1, np.pi/180, 80)
# for i in range(lines.shape[0]):
for rho, theta in lines[3]:
    a = np.cos(theta)
    b = np.sin(theta)
    x0 = a*rho
    y0 = b*rho
    x1 = int(x0 + 1000*(-b))
    y1 = int(y0 + 1000*a)
    x2 = int(x0 - 1000*(-b))
    y2 = int(y0 - 1000*a)

    cv2.line(img, (x1, y1), (x2, y2), (0, 0, 255), 2)


plt.subplot(121)
plt.imshow(rec1_img, cmap="gray")
plt.subplot(122)
plt.imshow(img, cmap="gray")
plt.show()
```





Area1 = 9330.5 mm^2

Perimeter1 = 411.5 mm

Area2 = 15580.5 mm^2

Perimeter2 = 502.9 mm

先使用 Canny 邊緣檢測出物件邊緣，然後再使用 Hough Transform 偵測出物體的每個邊。

再 Hough Transform 中，對 Accumulator 的 threshold 設定很重要，要超過 threshold 才會認定為是一條直線。