HW7 R11944009 何明倩

(a) thinning

- Description:
  針對 lena 轉換成 binary image 並 downsmaple 成 64x64。之後做 thinning。

- Algorithm:

  Step 1: preprocess

  利用 hw6 的 function 先做二值化再做 downsample，即得到 symbol image。

  Step 2: 計算 Yokoi connectivity number

  利用 HW6 Yokoi connectivity number 的 function 得到此圖的 Yokoi connectivity number

  Step 3: 計算 Pair Relationship

  先產生跟 step2 output 的矩陣一樣大的 Zero matrix，result。針對 step2 得到的 output，先做上下左右各 1 pixel 的 zero padding。接著從左到右從下到上拜訪每個 pixel，當遇到有值才做判斷，當該 pixel 值為 1 時且 4-connected 有至少一個 pixel 為 1，則 assign result 這個位置的 pixel 為 2（PPT 上的 p），反之，則 assign 為 1（PPT 上的 q）

  Step 4: Connected Shrink Operator

  針對 step3 的 result 先做上下左右各 1 pixel 的 zero padding。接著從左到右從下到上拜訪每個 pixel，當遇到該 pixel 值為 2 時，去計算這個 pixel 的 Yokoi connectivity number 若為 1，則將 symbol image 的該位置 pixel 值設為 0，反之，則保留原本的值。拜訪即可得到新的 symbol image。

  Step 5:

  一直重複 step2 到 step4，直到 symbol image 和前一輪的 symbol image 一致才停止，得到最終 thinning 的影像

- Principal code fragment:
  - 計算該 pixel 的 Yokoi connectivity number

```
def Yokoi_num(img_tmp):
    x = 1; y = 1
    a1 = h_f(img_tmp[x, y], img_tmp[x, y+1], img_tmp[x-1, y+1], img_tmp[x-1, y])
    a2 = h_f(img_tmp[x, y], img_tmp[x-1, y], img_tmp[x-1, y-1], img_tmp[x, y-1])
    a3 = h_f(img_tmp[x, y], img_tmp[x, y-1], img_tmp[x+1, y-1], img_tmp[x+1, y])
    a4 = h_f(img_tmp[x, y], img_tmp[x+1, y], img_tmp[x+1, y+1], img_tmp[x, y+1])
    out = f(a1, a2, a3, a4)
    return out
```

  - Pair Relationship Operator

```
def pair_relate_op(Yokoi):
    h, w = Yokoi.shape
    Yokoi_padding = np.zeros((h+2, w+2))
    Yokoi_padding[1:h+1, 1:w+1] = Yokoi
    result = np.zeros((h,w))
    # q: 1; p: 2
    for r in range(h):
        for c in range(w):
            x = c+1; y = r+1
            if (Yokoi[r, c] > 0):
                if (Yokoi[r, c] == 1):
                    nebor_4 = np.array([Yokoi_padding[y, x+1], Yokoi_padding[y-1, x], Yokoi_padding[y
                    edge_num = np.sum(nebor_4 == 1)
                    result[r, c] = 2 if edge_num > 0 else 1
                else:
                    result[r, c] = 1
    return result
```

  - shrink

```
def shrink(pair_relate_img, symbol_img):
    h, w = symbol_img.shape
    symbol_img_padding = np.zeros((h+2, w+2))
    symbol_img_padding[1:h+1, 1:w+1] = symbol_img
    for r in range(h):
        for c in range(w):
            x = c+1; y = r+1
            if(pair_relate_img[r, c] == 2):
                isedge = Yokoi_num(symbol_img_padding[y-1:y+2, x-1:x+2])
                symbol_img_padding[y, x] = 0 if isedge == 1 else 1
    return symbol_img_padding[1:h+1, 1:w+1]
```

  - thinning

```
def thinning(symbol_img):
    output = np.zeros(symbol_img.shape)
    while(np.sum(np.abs(symbol_img - output))):
        output = np.array(symbol_img)
        # Yokoi_4connected
        Yokoi = Yokoi_4connected(symbol_img)
        # Pair Relationship Operator
        pair_relate_img = pair_relate_op(Yokoi)
        # shrink
        symbol_img = shrink(pair_relate_img, symbol_img)
    return output
```

- Result: