Computer and Robot Vision Homework 7
Thining

資工碩一 張家源 r07922102

使用語言： **Python**

Downsample

將圖片Binarize後，透過8x8 filter downsample 成 64x64 image

```python
def Binary(img):
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            if img[i][j] >= 128 :
                img[i][j] = 255
            else:
                img[i][j] = 0
    return img


def Resize(img, shape):
    width = int(img.shape[1]/shape[1])
    height = int(img.shape[0]/shape[0])
    img_res = np.zeros( (height, width)).astype(int)
    for i in range(0,height):
        for j in range(0,width):
                img_res[i][j] = img[i*8][j*8]
    cv2.imwrite('resize.jpg',img_res)
    return img_res
```

Yokoi and Pair Relationship

先Yokoi，再按照Pair Relationship定義，如果此Pixel yokoi connectivity number=1，檢查Neighbor yokoi number=1的個數來更新每個Pixel。

- H function: (m="1", means "edge" in Yokoi)
  - $h(a,m) = \begin{cases} 1, if\ a = m \\ 0, otherwise \end{cases}$
- Output:
  - $y = \begin{cases} q, if\ \sum_{n=1}^{4} h(x_n, m) < 1\ or\ x_0 \neq m \\ p, if\ \sum_{n=1}^{4} h(x_n, m) \geq 1\ and\ x_0 = m \end{cases}$

以3代表p，4代表q

```python
def PairRelationship(img):
    move = [(1,0), (0,-1), (-1,0), (0,1)] #4 connectivity
    #move = [(1,0), (0,-1), (-1,0), (0,1), (1,1), (1,-1), (-1,-1), (-1,1)] #8 connectivity

    res = np.zeros(img.shape).astype(int)
    for r in range(img.shape[0]):
        for c in range(img.shape[1]):
            if img[r,c] == 1:
                interior = 0
                for m in move:
                    x_r = r+m[0]
                    x_c = c+m[1]
                    if not(x_r < 0 or x_r >= img.shape[0] or x_c < 0 or x_c >= img.shape[1]):
                        if img[x_r, x_c] == 1:
                            interior += 1
                if img[r, c] == 1 and interior >= 1: #border and next to interior
                    res[r, c] = 3 # 3 represent p
                elif img[r, c] != 1 or interior < 1:
                    res[r, c] = 4 # 4 represent q
```

Connected Shrink

經過Pair relationship之後，針對是p的pixel，計算出它的a1,a2,a3,a4，最後決定此pixel是否該被Remove。

```python
def ConnectedShrink(img, img_bin):
    move = [(0,0), (1,0), (0,-1), (-1,0), (0,1), (1,1), (1,-1), (-1,-1), (-1,1)] #move for getting x
    res = img_bin
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            if img[i][j] == 3: #if foreground is p
                x = []
                for m in move: #set x[i]
                    r = i+m[1]
                    c = j+m[0]
                    if r < 0 or r >= img.shape[0] or c < 0 or c >= img.shape[1]:
                        x.append(0)
                    else:
                        x.append(res[r][c])
                a1 = H(x[0], x[1], x[6], x[2])
                a2 = H(x[0], x[2], x[7], x[3])
                a3 = H(x[0], x[3], x[8], x[4])
                a4 = H(x[0], x[4], x[5], x[1])
                if not F(a1,a2,a3,a4): #delete
                    res[i, j] = 0
                    global remove
                    remove = True
```

Thinning

經過一次Mark Interior/Border operation、Pair Relationship、Connected Shrink operation之後，如果有Pixel被移除，則再重新做一次，直到沒有任何pixel被移除為止。

Result:

```python
def Thinning(img_resize):
    global remove
    while remove == True:
        remove = False
        img_ib = InteriorBorder(img_resize)
        img_pr = PairRelationship(img_ib)
        img_resize = ConnectedShrink(img_pr, img_resize)
```