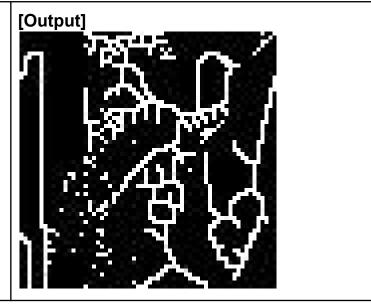
## Computer Vision - Assignment 7

R09922A04 資工所人工智慧組 黃品硯

## (a) Thinning

First, perform a yokoi connectivity algorithm on the downsampled image. Then get the marked image by the pair relationship operator. Next, use the connected shrink operator to check whether the pixel is removable and is also marked by the pair relationship operator. If yes, then remove the pixel. Repeat the process until the image has no more pixels to remove.



## [Code]

```
def h(b, c, d, e):
    if b == c and (b != d or b != e):
        return "q"
    elif b == c == d == e:
        return "r"
    elif b != c:
        return "s"
def f(a1, a2, a3, a4):
    if a1 == a2 == a3 == a4 == "r":
        return 5
    return len([a for a in [a1, a2, a3, a4] if a == "q"])
def get_neighbors_pixel(img, y, x):
    coords = [
        (x, y), (x + 1, y), (x, y + 1), (x - 1, y), (x, y - 1),
        (x + 1, y - 1), (x + 1, y + 1), (x - 1, y + 1), (x - 1, y - 1),
    1
    neighbors pixel = []
    for x, y in coords:
```

```
if x < width and x >= 0 and y < height and <math>y >= 0:
            neighbors pixel.append(img[y, x])
        else:
            neighbors_pixel.append(0)
    return neighbors_pixel
def yokoi connectivity(img):
    for y in range(height):
        for x in range(width):
            if img[y, x] == 255:
                x_i = get_neighbors_pixel(img, y, x)
                a1 = h(x_i[0], x_i[1], x_i[6], x_i[2])
                a2 = h(x_i[0], x_i[2], x_i[7], x_i[3])
                a3 = h(x_i[0], x_i[3], x_i[8], x_i[4])
                a4 = h(x_i[0], x_i[4], x_i[5], x_i[1])
                n = f(a1, a2, a3, a4)
                out img[y, x] = n
            else:
                out img[y, x] = 0
    return out_img
def pair relationship operator(img):
    for y in range(height):
        for x in range(width):
            if img[y, x] == 1:
                is p = False
                x_is = get_neighbors_pixel(img, y, x)
                for x i in x is[1:5]: \# x 1, x 2, x 3, x 4
                    if x i is not None and <math>x i == 1:
                         is p = True
                out_img[y, x] = "p" if is_p else "q"
            elif img[y, x] > 1:
                out img[y, x] = "q"
            else:
                out_img[y, x] = ""
    return out_img
```

```
def connected_shrink_operator(original_img, marked_img):
    for y in range(height):
        for x in range(width):
            if marked_img[y, x] == "p":
                x_i = get_neighbors_pixel(out_img, y, x)
                a1 = h(x_i[0], x_i[1], x_i[6], x_i[2])
                a2 = h(x_i[0], x_i[2], x_i[7], x_i[3])
                a3 = h(x_i[0], x_i[3], x_i[8], x_i[4])
                a4 = h(x_i[0], x_i[4], x_i[5], x_i[1])
                n = f(a1, a2, a3, a4)
                if n == 1:
                    out_img[y, x] = 0
    return out_img
if __name__ == "__main__":
    while True:
        yokoi_img = yokoi_connectivity(img)
        marked img = pair relationship operator(yokoi img)
        thinning_img = connected_shrink_operator(img, marked_img)
        if np.array equal(img, thinning img):
            break
        img = thinning_img
```