Computer Vision_HW7

- Command Line

  python main.py Thin

- Thinning

1. Binarize (threshold as 128)

```
    img_bin = np.zeros(img.shape, np.int)
    for y in range(img.shape[0]):
        for x in range(img.shape[1]):
            if img[y][x] >= 128:
                img_bin[y][x] = 255
```

2. Downsample to 64*64

```
img_ds = np.zeros((64,64), np.int)
for y in range(img_ds.shape[0]):
    for x in range(img_ds.shape[1]):
        img_ds[y][x] = img_bin[8*y][8*x]
```

3. Mark the interior/border pixels

```
def interior_border(img):
    def h(c, d):
        if c == d:
            return c
        else:
            return 'b'

    img_ib = np.zeros(img.shape, np.int)
    for y in range(img.shape[0]):
        for x in range(img.shape[1]):
            if img[y][x] > 0:
                x1, x2, x3, x4 = 0, 0, 0, 0
                if y == 0:
                    if x == 0:
                        x1, x4 = img[y][x+1], img[y+1][x]
                    elif x == img.shape[1]-1:
                        x3, x4 = img[y][x-1], img[y+1][x]
                    else:
                        x1, x3, x4 = img[y][x+1], img[y][x-1], img[y+1][x]
                elif y == img.shape[0]-1:
                    if x == 0:
                        x1, x2 = img[y][x+1], img[y-1][x]
                    elif x == img.shape[1]-1:
                        x2 ,x3 = img[y-1][x], img[y][x-1]
                    else:
                        x1, x2, x3 = img[y][x+1], img[y-1][x], img[y][x-1]
                else:
                    if x == 0:
                        x1, x2, x4 = img[y][x+1], img[y-1][x], img[y+1][x]
```

4. Pair relationship operator

```
def pair_relationship_operator(img):
    def h(a, i):
        if a == i:
            return 1
        else:
            return 0

    img_marked = np.zeros(img.shape, np.int)
    for y in range(img.shape[0]):
        for x in range(img.shape[1]):
            if img[y][x] > 0:
                x1, x2, x3, x4 = 0, 0, 0, 0
                if y == 0:
                    if x == 0:
                        x1, x4 = img[y][x+1], img[y+1][x]
                    elif x == img.shape[1]-1:
                        x3, x4 = img[y][x-1], img[y+1][x]
                    else:
                        x1, x3, x4 = img[y][x+1], img[y][x-1], img[y+1][x]
                elif y == img.shape[0]-1:
                    if x == 0:
                        x1, x2 = img[y][x+1], img[y-1][x]
                    elif x == img.shape[1]-1:
                        x2 ,x3 = img[y-1][x], img[y][x-1]
                    else:
                        x1, x2, x3 = img[y][x+1], img[y-1][x], img[y][x-1]
                else:
                    if x == 0:
                        x1, x2, x4 = img[y][x+1], img[y-1][x], img[y+1][x]
```

5. Marked-pixel connected shrink operator(using yokoi number and marked image on Step4)

```
img_yokoi = yokoi(img_thin)
for y in range(img_thin.shape[0]):
    for x in range(img_thin.shape[1]):
        if img_yokoi[y][x] == 1 and img_marked[y][x] == 1:
            img_thin[y][x] = 0
```

6. Repeat Step3,4,5 until the last output never changed

```
while True:
    img_thinned_original = copy.deepcopy(img_thin)

    #Step1: mark the interior/border pixels
    #input: original symbolic image
    #output: interior/border image
    img_ib = interior_border(img_thin)

    #Step2: pair relationship operator
    #input: interior/border image
    #output: marked image
    img_marked = pair_relationship_operator(img_ib)

    #Step3: marked-pixel connected shrink operator
    #input: original symbolic image + marked image
    #output: thinned image
    img_yokoi = yokoi(img_thin)
    for y in range(img_thin.shape[0]):
        for x in range(img_thin.shape[1]):
            if img_yokoi[y][x] == 1 and img_marked[y][x] == 1:
                img_thin[y][x] = 0

    #Use thinned image as next original symbolic image
    #Repeat Step1,2,3 until the last output never changed
    if np.sum(img_thin == img_thinned_original) == img_thin.shape[0] * img_thin.shape[1]:
        break
```

- Result