Computer-vision-Homework 7

Thinning

Due date：30 Nov 2021

Programming language: python 3.9.7

Import lib:

- Opencv: to read and write the image file
- Numpy: to work with the arrays

Original image: lena.bmp

[512(width),512(height),1channel(cv2.IMREAD_GRAYSCALE)]

Code explanation:

```
lena = cv2.imread('lena.bmp', cv2.IMREAD_GRAYSCALE)
```

- imread to load a file with cv2.IMREAD_GRAYSCALE(or using 0 as the parameter of the function)

Q1：**image of thinning on lena**

```
def DownSample(image, DownSampleSize):
    DownSampleImage = np.zeros((DownSampleSize, DownSampleSize))
    # return DownSampleImage
    DownSamplelength = int(len(image) / DownSampleSize)
    for i in range(len(DownSampleImage)):
        for j in range(len(DownSampleImage[i])):
            DownSampleImage[i][j] = image[DownSamplelength *
i][DownSamplelength * j]
    return DownSampleImage


def Binarize(image):
    BinarizeImage = np.zeros(image.shape)
    for i in range(len(image)):
        for j in range(len(image)):
            BinarizeImage[i][j] = 0 if image[i][j] < 128 else 255
    return BinarizeImage
```

First I Binarized and Downsampling the lena.bmp from 512 to 64 with threshold value 128.

```
def Yokoi(image):
    YokoiImage = np.zeros(image.shape)
    for i in range(len(image)):
        for j in range(len(image[i])):
            if image[i][j] == 0:
                YokoiImage[i][j] = 0
                continue
            r, q = 0, 0
            # a1
            if j + 1 < len(image[i]) and image[i][j] == image[i][j +
1]:
                q += 1
            if i - 1 >= 0 and j + 1 < len(image[i]) and image[i][j] ==
image[i][j + 1] == image[i - 1][j] == image[i - 1][j + 1]:
                q -= 1
                r += 1
            # a2
            if i - 1 >= 0 and image[i][j] == image[i - 1][j]:
```

```
            q += 1
        if j - 1 >= 0 and i - 1 >= 0 and image[i][j] == image[i -
1][j] == image[i - 1][j - 1] == image[i][j - 1]:
            q -= 1
            r += 1
        # a3
        if j - 1 >= 0 and image[i][j] == image[i][j - 1]:
            q += 1
        if i + 1 < len(image) and j - 1 >= 0 and image[i][j] ==
image[i][j - 1] == image[i + 1][j] == image[i + 1][
            j - 1]:
            q -= 1
            r += 1
        # a4
        if i + 1 < len(image) and image[i][j] == image[i + 1][j]:
            q += 1
        if j + 1 < len(image[i]) and i + 1 < len(image) and
image[i][j] == image[i + 1][j] == image[i][j + 1] == image[i + 1][j +
1]:
            q -= 1
            r += 1
        if r == 4:
            YokoiImage[i][j] = 5
        else:
            YokoiImage[i][j] = q
    return YokoiImage
```

According to the definition of Yokoi connectivity number and details in the last homework6.

```
def PairRelationship(image):
    PairImage = np.zeros(image.shape)  # 這裡拿 q=1 , p=2
    list = [[1, 0], [-1, 0], [0, 1], [0, -1]]
    for i in range(len(image)):
        for j in range(len(image[i])):
            if image[i][j] == 0:
                continue
            PairImage[i][j] = 1
            if image[i][j] == 1:
```

```
            for position in list:
                r, c = position
                if 0 <= i + r < len(image) and 0 <= j + c <
len(image[i]) and image[i][j] == image[i + r][j + c]:
                    PairImage[i][j] = 2
                    break
    return PairImage
```

Mark the image with pair relationship operator according to the following functions.

Pair Relationship Operator

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

```
def thinning(image, thinningImage):
    for i in range(len(image)):
        for j in range(len(image[i])):
            if image[i][j] <= 1:  # q=1, p=2 我只處理p
                continue
            r, q = 0, 0
            # a1
            if j + 1 < len(image[i]) and thinningImage[i][j]
== thinningImage[i][j + 1]:
                q += 1
            if i - 1 >= 0 and j + 1 < len(image[i]) and
thinningImage[i][j] == thinningImage[i][j + 1] == \
                    thinningImage[i - 1][j] == \
                    thinningImage[i - 1][j + 1]:
```

```
                q -= 1
                r += 1
            # a2
            if i - 1 >= 0 and thinningImage[i][j] ==
thinningImage[i - 1][j]:
                q += 1
            if j - 1 >= 0 and i - 1 >= 0 and
thinningImage[i][j] == thinningImage[i - 1][j] ==
thinningImage[i - 1][
                j - 1] == thinningImage[i][
                j - 1]:
                q -= 1
                r += 1
            # a3
            if j - 1 >= 0 and thinningImage[i][j] ==
thinningImage[i][j - 1]:
                q += 1
            if i + 1 < len(image) and j - 1 >= 0 and
thinningImage[i][j] == thinningImage[i][j - 1] == \
                    thinningImage[i + 1][j] == \
                    thinningImage[i + 1][j - 1]:
                q -= 1
                r += 1
            # a4
            if i + 1 < len(image) and thinningImage[i][j] ==
thinningImage[i + 1][j]:
                q += 1
            if j + 1 < len(image[i]) and i + 1 < len(image)
and thinningImage[i][j] == thinningImage[i + 1][j] == \
                    thinningImage[i][j + 1] == \
                    thinningImage[i + 1][j + 1]:
                q -= 1
                r += 1
            if q == 1:
                thinningImage[i][j] = 0
                image[i][j] = 0
    return thinningImage
```

Applied connected shrink operator to each pixels and with the following functions.

Connected Shrink Operator

- H function: (yokoi corner => “q”)
  - $h(b,c,d,e) = \begin{cases} 1, if\ b = c\ and\ (d \neq b\ or\ e \neq b) \\ 0, otherwise \end{cases}$
- Output:
  - $f(a_1, a_2, a_3, a_4, x) = \begin{cases} g, if\ exactly\ one\ of\ a_n = 1, n = 1 \sim 4 \\ x, otherwise \end{cases}$

Repeat several times with the above function and we’ll get the final image.

```
if __name__ == '__main__':

    lena = cv2.imread("lena.bmp", cv2.IMREAD_GRAYSCALE)
    # 先 down sample 成 64*64 再 binarize
    finalImage = Binarize(DownSample(lena, 64))
    for i in range(7):
        # 做 Yokoi 標出每個點的關係
        shrinkImage = Yokoi(finalImage)
        # pair relation
        shrinkImage = PairRelationship(shrinkImage)
        # thinning
        finalImage = thinning(shrinkImage, finalImage)
    finalImage = ScaledUp(finalImage, len(lena))

    cv2.imwrite('thinning_image.bmp', finalImage)
    cv2.imshow('thinnig' , finalImage)
    cv2.waitKey()
    cv2.destroyAllWindows()
```