R07922141
張緣彩
10/19/2018

Computer Vision Hw7 Report

I. Implementation

First down sampling and binarizing lena same as homework 6.

```
# Downsampling and binarize
downsampled_lena = [[0 for j in range(64)] for i in range(64)]

for i in range(64):
    for j in range(64):
        pixel = lena_img[i*8][j*8]
        if pixel < 128:
            downsampled_lena[i][j] = 0
        else:
            downsampled_lena[i][j] = 255

cv2.imwrite('downsampled_lena.png', np.array(downsampled_lena))
```

Then keep repeating step 1, 2 and 3 until nothing change.

```
while True:
    notChange = True

    cv2.imwrite('lena_thinning_'+str(count)+'.png', np.array(thinned))
    count += 1

    borderImage = getInteriorBorder(thinned, 8)
    pairImage = getPairRelationship(borderImage, 8)

    for i in range(len(thinned)):
        for j in range(len(thinned[0])):
            neighbors = getNeighbors(thinned, i, j, 8)

            a1 = h(neighbors[0], neighbors[1], neighbors[6], neighbors[2], cc)
            a2 = h(neighbors[0], neighbors[2], neighbors[7], neighbors[3], cc)
            a3 = h(neighbors[0], neighbors[3], neighbors[8], neighbors[4], cc)
            a4 = h(neighbors[0], neighbors[4], neighbors[5], neighbors[1], cc)
            if f(a1, a2, a3, a4, thinned[i][j]) == 'g' and pairImage[i][j] == 'p' and thinned[i][j] != 0:
                thinned[i][j] = 0
                notChange = False

    if notChange:
        break
```

For the shrink operator, we need to find the ‘h’ and ‘f’ function according to lecture note.

for 4-connectivity

$$h(b,c,d,e) = \begin{cases} 1 & if\ b = c \wedge (b \neq d \vee b \neq e) \\ 0 & otherwise \end{cases}$$

for 8-connectivity

$$h(b,c,d,e) = \begin{cases} 1 & if\ b \neq c \wedge (b = d \vee b = e) \\ 0 & otherwise \end{cases}$$

```
def h(b, c, d, e, cc=4):
    if cc == 4:
        return 1 if b == c and (b != d or b != e) else 0
    else:
        return 1 if b != c and (b == d or b == e) else 0
```

$$output = f(a_1, a_2, a_3, a_4, x_0) = \begin{cases} g & if\ exactly\ one\ of\ a_1, a_2, a_3, a_4 = 1 \\ x_0 & otherwise \end{cases}$$

```
def f(a1, a2, a3, a4, x0):
    return 'g' if (a1 + a2 + a3 + a4) == 1 else x0
```

For the Interior-Border function, iterate every pixel and get its neighbor points, then find its ‘h’ and ‘f’ function according to lecture note.

$$h(c,d) = \begin{cases} c & if\ c = d \\ b & if\ c \neq d \end{cases}$$

$$f(c) = \begin{cases} b & if\ c = b \\ i & if\ c \neq b \end{cases}$$

```
def getInteriorBorder(image, cc=4):
    def _h(c, d):
        return c if c == d else 'b'

    def _f(c):
        return 'b' if c == 'b' else 'i'

    toReturn = copy.deepcopy(image)

    for i in range(64):
        for j in range(64):
            neighbors = getNeighbors(image, i, j, 8)
            a0 = neighbors[0]
            if cc == 4:
                for k in range(4):
                    a0 = _h(a0, neighbors[k+1])
            else:
                for k in range(8):
                    a0 = _h(a0, neighbors[k+1])
            toReturn[i][j] = _f(a0)

    return toReturn
```

For the pair relationship function, iterate every pixel and get its neighbor points, then find ‘h’ and output p and q note according to lecture note.

$$h(a,i) = \begin{cases} 1 & if\ a = i \\ 0 & otherwise \end{cases}$$

$$output = \begin{cases} q & if \sum_{n=1}^{4} h(x_n, i) < 1 \vee x_0 \neq b \\ p & if \sum_{n=1}^{4} h(x_n, i) \geq 1 \wedge x_0 = b \end{cases}$$

```
def getPairRelationship(image, cc=4):
    def _h(a, m='i'):
        return 1 if a == m else 0

    toReturn = copy.deepcopy(image)

    for i in range(64):
        for j in range(64):
            neighbors = getNeighbors(image, i, j, 8)
            delta = 1
            x0 = neighbors[0]
            c = 0

            if cc == 4:
                for k in range(1, 5):
                    c += _h(neighbors[k])
            else:
                for k in range(1, 9):
                    c += _h(neighbors[k])

            if c < delta or x0 != 'b':
                toReturn[i][j] = 'q'
            elif c >= delta and x0 == 'b':
                toReturn[i][j] = 'p'

    return toReturn
```

II. Result

Iteration	1st	2nd	3rd	4th	5th (Final)
Result					

Final Result (border/interior 8, pair relation 8, yokoi 4)