R09945020 朱彥如

Homework 7

Write a program which does thinning on a downsampled image (lena.bmp).

Downsampling Lena from 512x512 to 64x64:
- Binarize the benchmark image lena as in HW2, then using 8x8 blocks as a unit, * take the topmost-left pixel as the downsampled data.
- You have to use 4-connected neighborhood detection.
- You can use any programing language to implement homework, however, you'll get zero point if you just call existing library.

The requirements above are accomplished by Python with the help of matplotlib package, cv2 package, numpy package.
In the beginning, I use 128 as a threshold to binarize the image.

Downsampling

For the requirement, I wrote the downsampling function to transform the 512x512 image into 64x64. The input of this function is 3d image array and the size set of reduction kernel. In the function, the reduction kernel will cover the image and take the topmost-left pixel in each kernel corresponded to the input image as the downsampled image. The return is the downsampled 3d image.

Yokoi connectivity number labeling

For labeling the Yokoi connectivity number, I wrote the QRS and yokoi function.
The QRS function has four inputs, and the first input is the center pixel while other three inputs are the angle neighbor pixels. The expression below is the formula of QRS.

$$QRS(b,c,d,e) = \begin{cases} return\ Q\ if b = c\ and\ (d \neq b\ \vee\ b \neq e) \\ return\ S,\ if b \neq c \\ return\ R,\ if b = c\ and\ (d = b\ \wedge\ b = e) \end{cases}$$

$$yokoi(image, r, c) = \begin{cases} return\ 0,\ if\ count(neighbors\ of\ image[r,c] = S) = 4 \\ return\ 5,\ if\ count(neighbors\ of\ image[r,c] = R) = 4 \\ return\ count(neighbors\ of\ image[r,c] = Q),\ otherwise \end{cases}$$

In the yokoi function, its inputs are image array and the coordinates of the target pixel. It will count the number of S labels, R labels and Q labels in the neighbor pixels of the target pixel with the help of QRS function and according to the counts, it will return the yokoi label for the target pixel.

To get the yokoi label image, I wrote yokoi_operation function, whose input is the 2D image array, and the output is the 2D yokoi label image. It uses yokoi function mentioned above to calculate each pixel's yokoi label. During detecting the neighbor pixels of boundary pixels, there are some out-range problems, so I used np.pad() to pad the image with 0.

R09945020 朱彥如

Pair Relationship Operator

To mark the pair relation of the image, I wrote PairRelationship(im,ll,pp) and h2_fun(a,m), two functions. The formula of functions are shown below.

$$h2\_fun(a,m) = \begin{cases} return\ 1,\ if\ a = m \\ return\ 0,\ otherwise \end{cases}$$

$$PairRelationship(img,l,p) = \begin{cases} return\ p,\ if\ count(neighbors\ of\ img[l,p] = 1) \geq 0 \\ \qquad\qquad and\ img[l,p] = 1 \\ return\ q,\ if\ count(neighbors\ of\ img[l,p] = 1) = 0 \\ \qquad\qquad or\ img[l,p] \neq 1 \\ return\ 0,\ otherwise \end{cases}$$

The inputs of PairRelationship() function are image array and coordinates of target pixel. It uses h2_fun() function to determine whether the neighbor pixels equal 1, representing edge in yokoi labels. If the count of neighbor pixels which equal to 1 is greater than 0, this function returns p. The pixel marked with label p will be checked whether turns into background pixel in Shrinking_fun() below.

Main shrinking function

The Shrinking_fun() function is the main function to create the thinning 2D image. Its input is the 2D image array. It has variables to record iteration number and whether there is a change in image array.

In the first loops, it labels pair-relationship for each pixel to find the pixels which are candidates become background and needed to detect in the follow-up codes. And in the second loop, I use yokoi function to find the pixels needed to be turned into background from the pixels labeled in p.

Also, for the boundary pixels problem, I used np.pad() to pad the image with 0. At last the function will be executed iteratively until no pixel changing. The function will return the thinning image and the number of iteration.

Results

It takes 7 iterations to complete the thinning of the lina image. After the main shrinking function, I used np.stack() function to transform 2D thinning image array into 3D (RGB) thinning image array, and used cv2 to export the thinning image as Fig. 1 shown.

R09945020 朱彥如

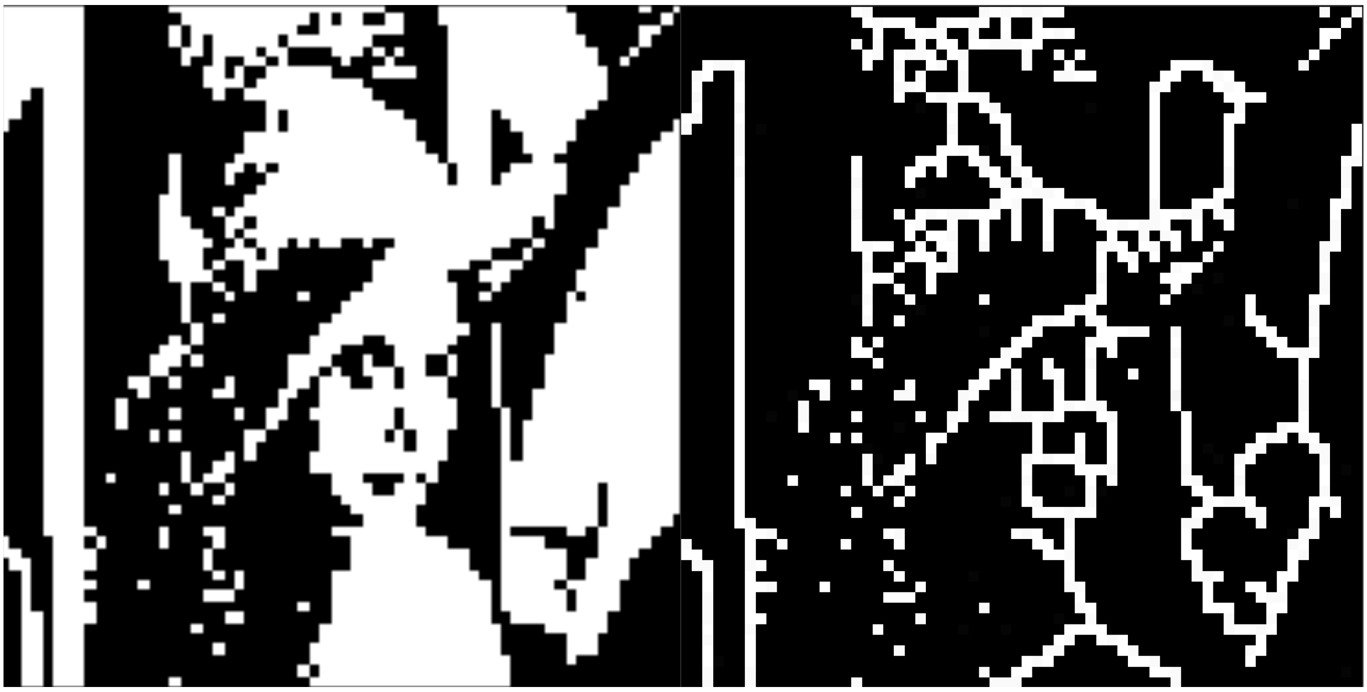

Figure 1. The left one is the downsized binary image. The right one is the thinning image.