CMPEN 555: Digital Image Processing II

Computer Project # 2:

Homotopic Skeletonization and Shape Analysis

Naichao Yin, Wenrui Wang, Zekai Liu

Date:2/21/2020

A. Objectives

1. Learn how to use MATLAB for general image processing.
2. do homotopic skeletonization of the target images
2. Study how to computer size distribution, pattern spectrum and complexity.
3. Use pattern recognition to match corresponding objects.

B. Methods

1) Question one

Top-level Process Flow:

1. Input images X: binary images “penn256.gif” and “bear.gif”,
2. Input structuring element pairs B_i: 8 different 3*3 kernels B_1-B_8 (Figure 1)
3. Do thinning iteration through the complete set of B_1 - B_8 burns off a layer around X
4. Repeat the step 3) twice/five times/ten times to get images X_2 / X_5 / X_{10}
5. Repeat the step 3) n times until the image can no longer be changed, get the final skeletonized image X_n.
6. Output images: X_2, X_5, X_{10} and X_n of corresponding binary images superposed on the original.

Thinning is a morphological operation that is used to remove selected foreground pixels from binary images, somewhat like erosion or opening. It can be used for several applications, but is particularly useful for homotopic skeletonization which is the process of stripping off layers of points from the original image, but maintaining the original shape until we get the skeleton of the image. It is subject to the target center and refines the target. Generally, the target after refining is

the width of single-layer pixel. And the skeleton can be understood as the central axis of the image.

Our approach is to do "thinning" on the binary images "penn256.gif" and "bear.gif", with the idea of moving from the periphery of the target to the center of the target, using the 8 different 3*3 structuring elements (Figure 1) to do skeletonize to the target until the operation reaches the point where the image can no longer be changed (the width of the single-layer pixel), to obtain the homotopic skeletonization of the image. The thinning operation is related to the hit-and-miss transform. The following is a detailed description of our homotopic skeletonization algorithm.

Structuring Element Pairs:

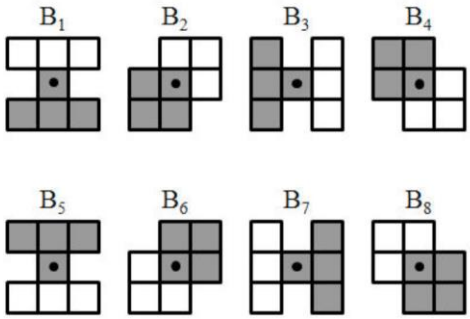

Figure 1. Set of B_i

Like other morphological operators, the behavior of the thinning operation is determined by a structuring element. For each B_i ($B_i = \{B_i^f, B_i^b\}$) in Figure 1, the "•' pixel corresponds to the origin. The 4 shaded pixels correspond to the "1' values for B_i^f and the 3 white pixels correspond to the "1" values for B_i^b. We use function "diskB" to get every B_i^f and B_i^b for each B_i.

In our function "output = diskB (num, flag)", input parameter "num" is the index "i" of B_i. For the parameter "flag", we set 1 to get the foreground B_i^f of B_i; set -1 to get the background B_i^b of B_i. Then we set value 0 for every negative pixel but leave the rest unchanged. The output A is the structuring element which we need to use in next operation(thinning). For example, we can call "output = diskB (1, -1)" to get B_1^b in Figure 1.

We use each structuring element B_i in turn to strip a layer of the target image in a specific direction: B_1 strips off north pixels, B_2 strips off northeast pixels, B_3 strips off east pixels, etc.

Because structuring elements are symmetric about the origin, such as B_1, B_2, B_3, B_4 are the reflection of B_5, B_6, B_7, B_8, we don't need to do symmetric operation "$(.)^s$" for the structuring elements in the thinning transform.

Image Thinning Process Flow:

1. Do erosion between binary image X and the foreground of the structuring element B_i^f.

$$\left(X \ominus B_i^f\right) \tag{1.1}$$

2. Do dilation between binary image X and the background of the same structuring element B_i^b.

$$\left(X \oplus B_i^b\right) \tag{1.2}$$

3. Design a hit-or-miss transform between the result of 1) and 2)

$$\left(X \ominus B_i^f\right) - \left(X \oplus B_i^b\right) \tag{1.3}$$

4. Do thinning of X by B_i, where the second term on the right is a hit-or-miss transform

$$X\ O\ B_i = X - \left(\left(X \ominus B_i^f\right) - \left(X \oplus B_i^b\right)\right) \tag{1.4}$$

The dilation operation (1.2) uses structure element B_i^b to scan each pixel of the image and to do OR operation on the covered binary. If one pixel of the scanned area is 255, the rest of the scanned area are is 255. Otherwise it is 0. In our MATLAB code, we first calculate the outer border in order to start operation at good point, then use sum () to accumulate the dilated pixels in corresponding area for each operation. As for the erosion (1.1), we do a little trick in this operation, $\left(X \ominus B_i^f\right) = \left(X^C \oplus B_i^f\right)^C$, use structure element B_i^f to scan each pixel of background and to do OR operation with the binary image it covers, then do complement image.

Use structuring element B_i^f for erosion in function "ErosionOwn" and structuring element B_i^b for dilatation in function "DilationOwn", next we compute a hit-or-miss operation (1.3) of subtracting the dilated image from the eroded image in function "compare_lma". The thinning operation is related to the hit-and-miss transform and can be expressed quite simply in terms of it, if the foreground and background pixels in the structuring element exactly match foreground and background pixels in the image, then the image pixel underneath the origin of the structuring element is set to background(zero), otherwise it is left unchanged. We do thinning operation (1.4) by subtracting the image after hit-or-miss from the original image X, to get the one-layer-burned

skeleton.

Iteration Process Flow:

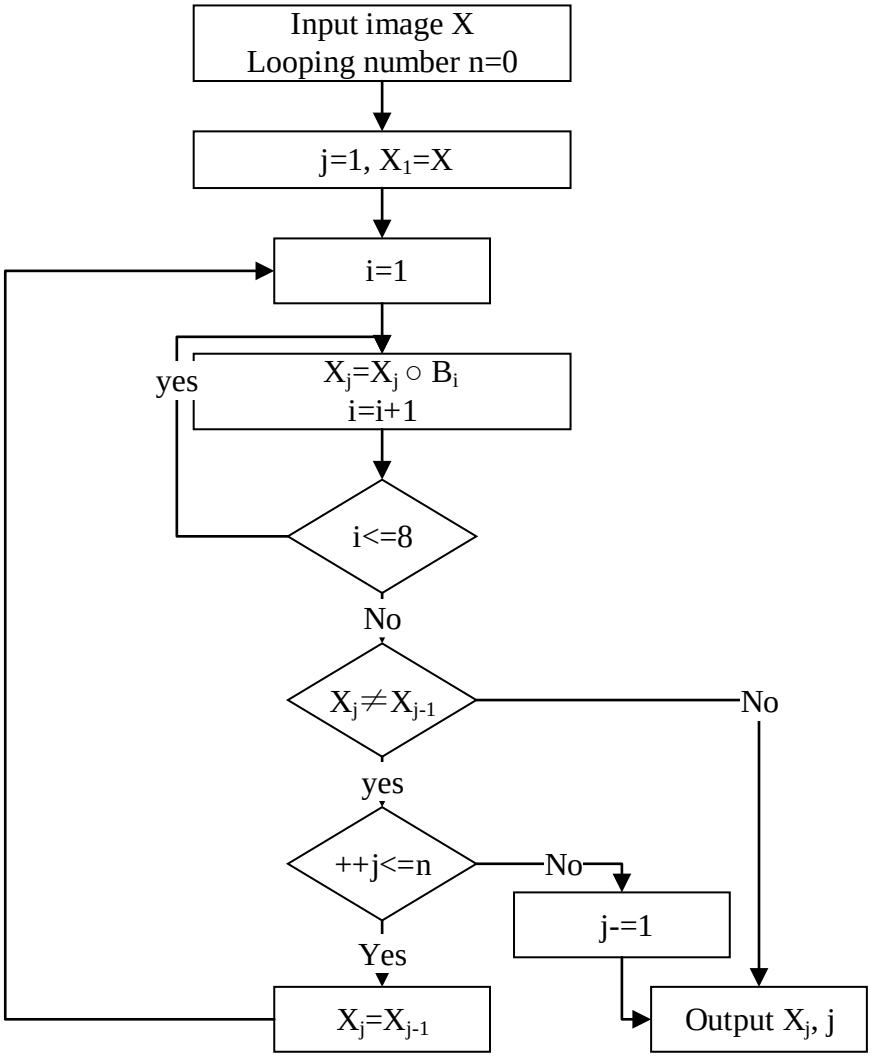

Figure 2. flowchart of iteration

In our outer loop(J), N can set to 2/5/10/999(to get the final skeletonized image) in order to do thinning iteration 2/5/10/999 times, but J will never reach 999, because the loop will stop once the image cannot be changed anymore (Figure 14 and Figure 16). In our inner loop(I), we do 8 times iteration, "I" start from 1 to 8 give the index of the structuring elements B_i which use to do thinning of X in next operation. Before the end of the outer loop, we compare the output image

X_i of this iteration to the last iteration Y (X_{i-1}), and if nothing changes, the whole iteration will stop, otherwise set Y to the current output image (X_i) and continue to do next iteration.

MATLAB

There are 3 code files in this question: "proj2q1.m", "diskB.m", "hitormiss.m". We create structuring elements by using file "diskB.m", get the required structuring elements by entering the appropriate input parameters. "hisormiss.m" is used for doing hit-or-miss transform to target image, we create three functions in this file: "ErosionOwn", "DilationOwn" and "compare_lma". Function "ErosionOwn" is used for doing erosion in our tricky method. Function "DilationOwn" is used for doing dilation. Function "compare_lma" is used for subtracting between the eroded image and dilated image to get the result after hit-or-miss transform. "proj2q1.m" is the main code of question 1. Run "proj2q1.m" directly to get the result of the thinning X by B_i by selecting the required number of iterations "n" and the input image "x".

2) Question two

1. Theory and Algorithms

A. Theory

(a) Size Distribution

Size distribution can be used to express distribution of shapes in an image. Opening operations with structuring elements of increasing size are used to get shapes with different sizes. The areas of these shapes can be seemed as size distribution. In discrete domain, the area is the amount of X's shape covered by rB. This can be represented in formula (2.1).

$$U(r) = m(X_{rB}), r \geq 0 \tag{2.1}$$

(b) Pattern Spectrum

Pattern spectrum of shape X is the amount of area in X per component. We can get the pattern spectrum through formula (2.2).

$$f(r) = \frac{-\frac{dU(r)}{dr}}{m(X)} \tag{2.2}$$

Where m(X) is the area of X and u(r) is the size distribution. In discrete domain, we can use difference instead of derivation.

(d) Shape Complexity

Shape complexity can express the complexity of X. Shape complexity can be computed through formula (2.3).

$$H(X|B) = -\sum_{i=0}^{N} f(i) \log f(i) \tag{2.3}$$

Where f(i) is the pattern Spectrum of X.

(c) Pattern Recognition

We can use the pecstrum to do pattern recognition. Suppose $f_{Ri}(n)$ is the pecstrum for reference object Ri, f(n) is pecstrum for test object. The test object can be determined by formula (2.4).

$$\begin{aligned} d_i &= \left[\sum_{n=0}^{N-1} C_n (f(n) - f_{Ri}(n))^2 \right]^{\frac{1}{2}} \\ \arg\{\min_i d_i\} &= i_{\min} \end{aligned} \tag{2.4}$$

Where Cn are the weights to emphasize various components. The Ri minimizing di is probably owns same pattern with test object. Besides, the weights C_n can be chosen to be equal to unity or chose some special values to emphasize specific parts. If the large differences from the reference pecstrum are to be emphasized, the weights can be chosen as follows:

$$c_n = \exp\left(a \left[f(n) - f_R(n) \right]^2 \right) \; a > 0 \tag{2.5}$$

If the small differences from the reference pecstrum are to be emphasized, the value of a is negative.

B. Algorithms

Our goal is to firstly compute size distribution, pecstrum and complexity, then use pecstral to complete pattern recognition. We need to firstly isolate distinct objects and find the minimum bounding rectangle. The process of MBR can be expressed as figure 3.

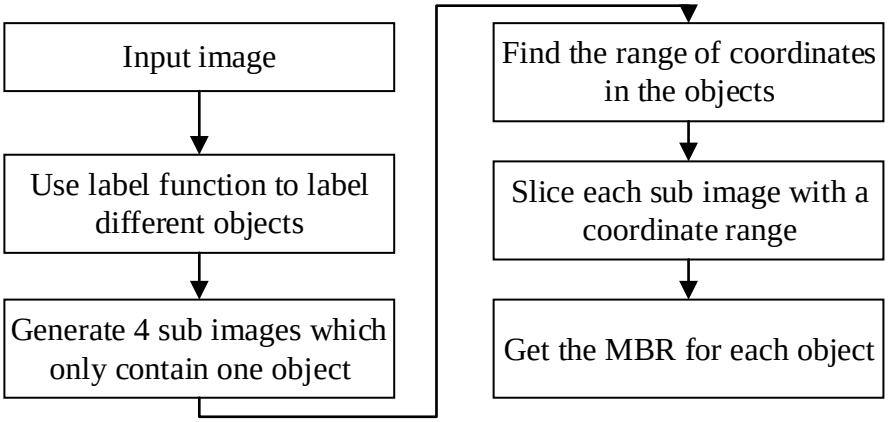

Figure 3. flowchart of MBR

Then the size distribution of each object is computed by formula (2.1). The flowchart to complete it is shown as figure 4.

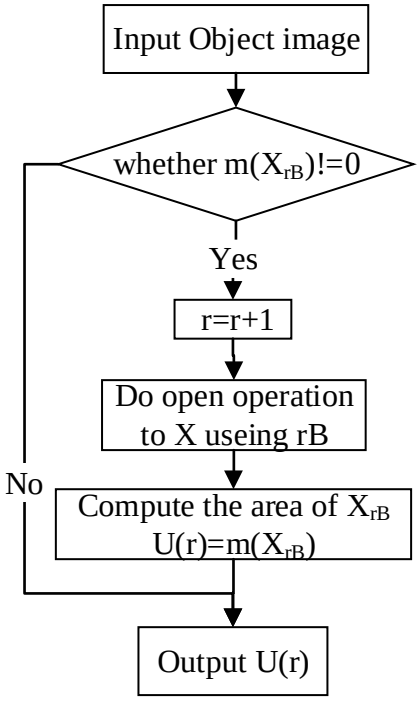

Figure 4. flowchart of computing size distribution

The pattern spectrum of each object is computed by formula (2.2). Pattern spectrum is

computed based on the result of size distribution. The computing flowchart is shown as figure 5.

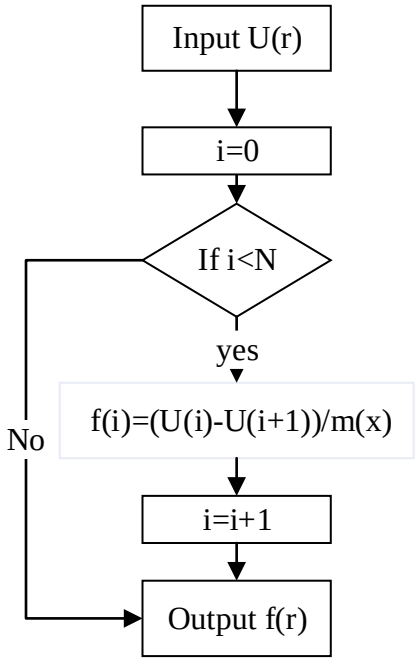

Figure 5. flowchart of computing pattern spectrum

The shape complexity of each object is computed by formula (2.3). Shape complexity is computed based on the result of pattern spectrum. The computing flowchart is shown as figure 6.

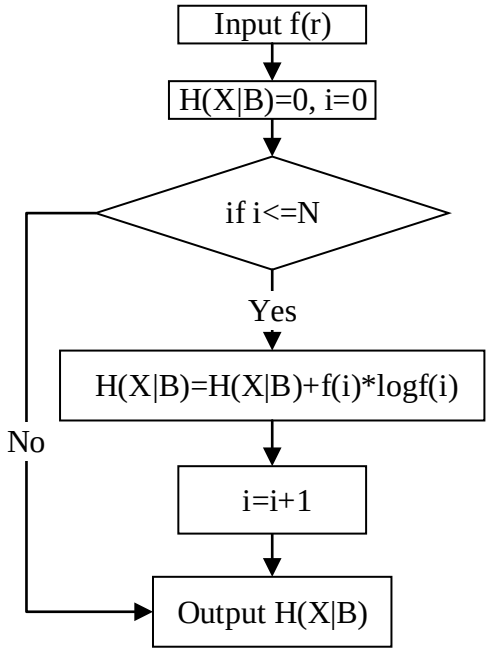

Figure 6. flowchart of computing shape complexity

Pecstral analysis can be utilized to find best matches of objects. Matching methods is the minimum-distance classifier. Pattern spectrum distance can be computed by formula (2.4) and the flowchart is shown as figure 5. The pattern spectrums of reference and test objects can be computed using methods above. The reference object in which the pattern spectrum distance is minimum can be the matching object.

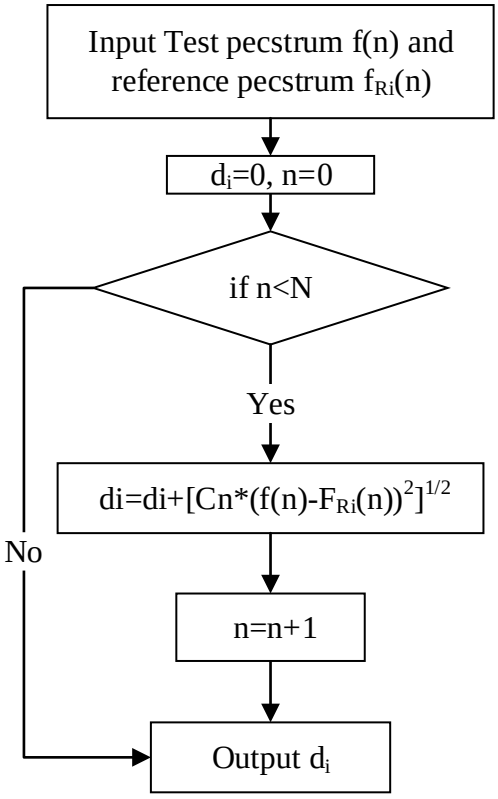

Figure 7. flowchart of computing pattern spectrum distance

Besides, the weights C_n can be chosen to be equal to unity or chose some special values to emphasize specific parts.

2. Matlab

There are 5 code files in our project: DiationOwn.m, erosion.m, size_distribution.m, q2_1_1.m, q2_1_2.m, q2_2_2.m. “DiationOwn.m” is used to do dilation operation to image and “erosion“ is used to do erosion operation to image. “size_distribution.m” is used for computing

image's size distribution and it calls the erosion and dilation function to complete the open operation. "q2_1_1.m" is the main code of question 2.1. Directly running "q2_1_1.m" can get the result of question 2_1_1. It calls label function in Matlab to label different objects then it calls "size_distribution.m" to compute the size distribution of "match1". Q2_2_1 calls the "size_distribution .m" to compute the size distribution. Then the result of size distribution is used to compute pattern spectrum of "match1" and "match 2". The complexity is computed based on pattern spectrum. Minimum distance of pattern spectrum is computed to match corresponding objects in "match1" and "match3". "Q2_2.m" is the main function of question 2_2. Directly running it can get the result of pattern spectrum of image" shadow1" and "shadow1rotated". It firstly calls 'size_distribution.m' to compute the size distribution of image'shadow1' solid parts, which is used for pattern spectrum. Then it also calls 'size_distribution.m' to compute the size distribution of image'shadow1rotated' solid parts. Finally use pattern spectrum to match every part in the two images.

C. Result

1) Question one

Figure 8. X_2 of "penn256"

Figure 9. X_5 of "penn256"

Figure 10. X_{10} of "penn256"

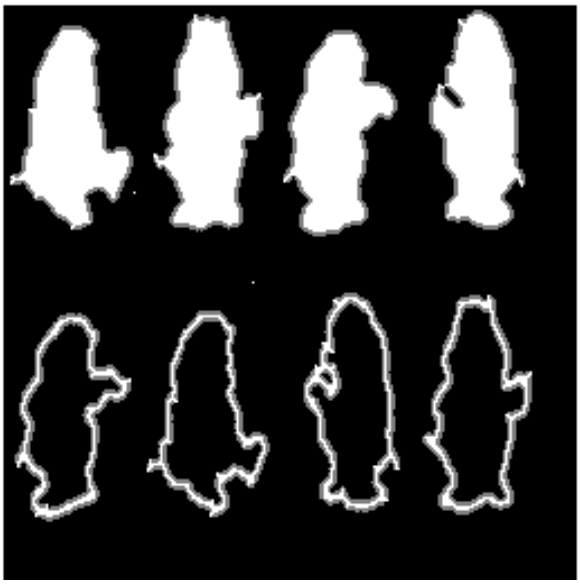

Figure 11. X_2 of "bear"

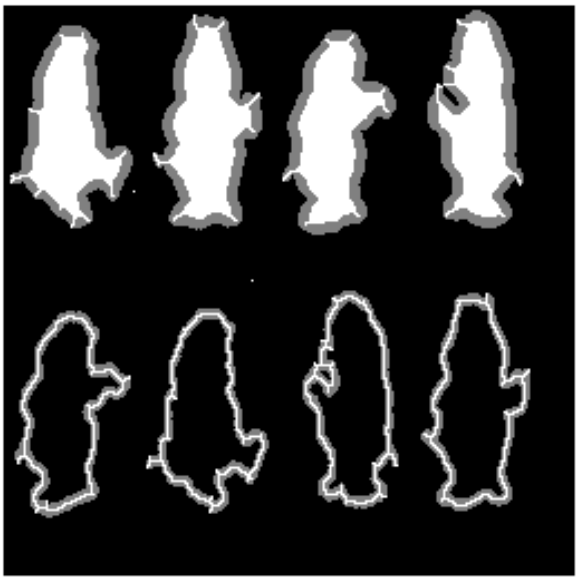

Figure 12. X_5 of "bear"

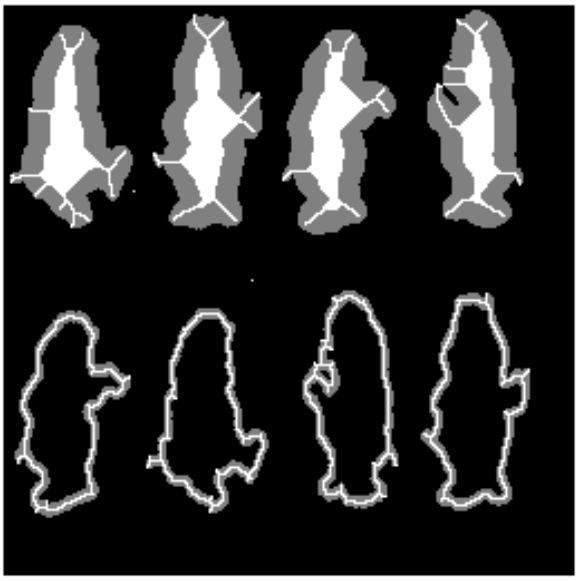

Figure 13. X_{10} of "bear"

After thinning the image "penn256.gif" and the image "bear.gif" 2/5/10 times, we show the results as a superposition of X_2, X_5, X_{10} on the original image X in figure 8/9/10 and figure 11/12/13. We can see that X_2 have burned off two layers, and the foreground area of X_2 have shrink a little in all directions. After five times thinning iterations, we roughly obtained the skeleton in "penn256.gif" and the skeleton of the bears in the second row in "bear.gif". Because the first-row bears have more foreground pixels than the second-row bears in "bear.gif", so the uppers require more thinning iterations than the lowers to get the final homotopic skeletonization.

Figure 14. The final skeletonized image of "penn256"

Workspace

Name	Value
i	8
j	8
n	999
x	*256x256 uint8*
x_b	*256x256 uint8*
y	*256x256 uint8*
z	*256x256 uint8*

Figure 15. The Workspace about final skeletonized image of "penn256"

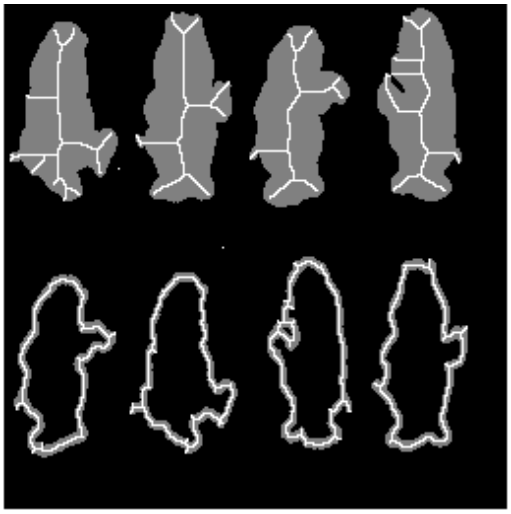

Figure 16. The final skeletonized image of "bear"

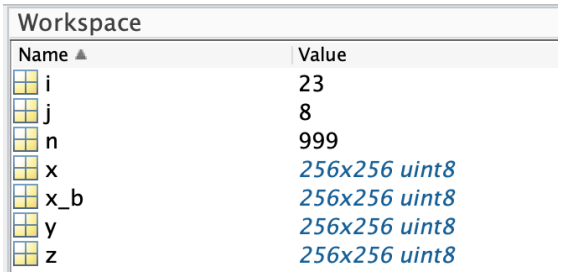

Figure 17. The Workspace about final skeletonized image of "bear"

The operation applied repeatedly until it causes no further changes to the image (Figure 14 and Figure 16). All detected characters have all been reduced to a single pixel width, but the skeletons produced by this method often contain undesirable short spurs produced by small irregularities in the boundary of the object. In addition, the index j displayed in the MATLAB workspace represents the actual number of iterations to obtain the final skeletonized image which converges at 8 and 23 (Figure 15 and Figure 17).

2) Question two

The size distribution of objects in "match1" is plot in table 1.

sdTable_match1 =

16×4 table

	object1	object2	object3	object4
U[0]	2125	1511	1945	1003
U[1]	2082	1488	1930	990
U[2]	2075	1332	1900	850
U[3]	2025	1215	1852	647
U[4]	1995	1155	1773	346
U[5]	1900	1127	1746	225
U[6]	1864	1081	1734	0
U[7]	1773	1015	1667	0
U[8]	1697	924	1590	0
U[9]	746	873	1072	0
U[10]	651	663	821	0
U[11]	0	621	0	0
U[12]	0	0	0	0
U[13]	0	0	0	0
U[14]	0	0	0	0
U[15]	0	0	0	0

Table 1. size distribution of "match1"

From the table we can see that object 2 has more size distribution and object 4 has less size distribution. All objects have no size distribution when r is more than 12.

The patter spectrum of objects in "match1" is plot in table 2.

psTable_match1 =

15×4 table

	object1	object2	object3	object4
f[0]	0.0202	0.0152	0.0077	0.013
f[1]	0.0033	0.1032	0.0154	0.1396
f[2]	0.0235	0.0774	0.0247	0.2024
f[3]	0.0141	0.0397	0.0406	0.3001
f[4]	0.0447	0.0185	0.0139	0.1206
f[5]	0.0169	0.0304	0.0062	0.2243
f[6]	0.0428	0.0437	0.0344	0
f[7]	0.0358	0.0602	0.0396	0
f[8]	0.4475	0.0338	0.2663	0
f[9]	0.0447	0.139	0.129	0
f[10]	0.3064	0.0278	0.4221	0
f[11]	0	0.411	0	0
f[12]	0	0	0	0
f[13]	0	0	0	0
f[14]	0	0	0	0

Table 2. pattern spectrum of “match1”

From the table 2, we can conclude that objects in “match1” have more small differences because small patterns account for a large proportion. This property provides with us reason for choosing weights of pattern recognition.

The size distribution of objects in “match1” is plot in table 3. “match3” has similar size distribution as “match1”.

sdTable_match3 =

16×4 table

	object1	object2	object3	object4
U[0]	2053	1397	1921	871
U[1]	2021	1362	1902	852
U[2]	2018	1269	1876	691
U[3]	1989	1117	1854	336
U[4]	1934	1060	1784	314
U[5]	1866	1051	1737	224
U[6]	1750	1028	1690	0
U[7]	1627	948	1649	0
U[8]	854	838	1578	0
U[9]	572	719	1470	0
U[10]	572	662	821	0
U[11]	0	575	799	0
U[12]	0	0	775	0
U[13]	0	0	0	0
U[14]	0	0	0	0
U[15]	0	0	0	0

Table 3. size distribution of "match3"

The patter spectrum of objects in "match1" is plot in table 4, which is similar as table 2.

psTable_match3 =

15×4 table

	object1	object2	object3	object4
f[0]	0.0156	0.0251	0.0099	0.0218
f[1]	0.0015	0.0666	0.0135	0.1848
f[2]	0.0141	0.1088	0.0115	0.4076
f[3]	0.0268	0.0408	0.0364	0.0253
f[4]	0.0331	0.0064	0.0245	0.1033
f[5]	0.0565	0.0165	0.0245	0.2572
f[6]	0.0599	0.0573	0.0213	0
f[7]	0.3765	0.0787	0.037	0
f[8]	0.1374	0.0852	0.0562	0
f[9]	0	0.0408	0.3378	0
f[10]	0.2786	0.0623	0.0115	0
f[11]	0	0.4116	0.0125	0
f[12]	0	0	0.4034	0
f[13]	0	0	0	0
f[14]	0	0	0	0

Table 4. pattern spectrum of "match3"

The spectrum distance between each object in “match1” and “match3” is shown in table 5.

distable =

4×4 table

	object1	object2	object3	object4
object1	0.0303	0.0724	0.0532	0.1109
object2	0.0475	0.0149	0.0502	0.0838
object3	0.0461	0.0436	0.0495	0.0951
object4	0.0692	0.069	0.0859	0.0382

Table 5. spectrum distance of “match1” and “match3”

The result that which two images matches are shown in Figure 18 to Figure 21.

Figure 18. First two images matching

Figure 19. Second two images matching

Figure 20. Third two images matching

Figure 21. Forth two images matching

We use weights C_n which can emphasize small difference to complete pattern recognition better. The C_n is set as formula (1.6)

$$c_n = \exp\left(-20\left[f(n) - f_R(n)\right]^2\right) \tag{2.6}$$

From table 5, we can conclude that object1 and object1, object2 and object2, object3 and object3, object4 and object4 in "match1" and "match3" have the minimum distance in column, so they are in the same pattern (the objects are ordered from left to right and top to bottom in images). The result is correct.

The patter spectrum of objects in "shadow1" and "shadow1rotated" is shown as table 7 and table 8. Based on our observation, the second object is different from the three other objects. The second object owns more large difference and the others owns more small difference. Therefore, we set weight a as 30 in formula (1.7) for pattern recognition of the second objects and set weight a as 1 in formula (1.8) for pattern recognition of the others.

$$c_n = \exp\left(30\left[f(n) - f_R(n)\right]^2\right) \tag{2.7}$$

$$c_n = \exp\left(1*\left[f(n) - f_R(n)\right]^2\right) \tag{2.8}$$

psTable_shadow1 =

15×4 table

	object1	object2	object3	object4
f[0]	0.0102	0.0118	0.0209	0.0046
f[1]	0.0102	0.0124	0.0262	0.0123
f[2]	0.0085	0.0399	0.022	0.0046
f[3]	0.0529	0.0169	0.0377	0.0331
f[4]	0.0845	0.0281	0.0502	0.0293
f[5]	0.1826	0.0326	0.1444	0.0547
f[6]	0.1894	0.0837	0.3431	0.0847
f[7]	0.0256	0.0258	0	0.0601
f[8]	0.029	0.1612	0.3556	0.3151
f[9]	0.407	0.2787	0	0.0616
f[10]	0	0.309	0	0.3398
f[11]	0	0	0	0
f[12]	0	0	0	0
f[13]	0	0	0	0
f[14]	0	0	0	0

Table 7. pattern spectrum of "shadow1"

psTable_shadow1rotated =

15×4 table

	object1	object2	object3	object4
f[0]	0.0214	0.0067	0.0076	0.011
f[1]	0.0398	0.0333	0.003	0.0245
f[2]	0.0183	0.0338	0.0122	0.0178
f[3]	0.053	0.0283	0.0327	0.044
f[4]	0.0571	0.0255	0.0213	0.0821
f[5]	0.211	0.0255	0.0708	0.0863
f[6]	0.2569	0.0449	0.0838	0.2453
f[7]	0.3425	0.1592	0.0586	0.0135
f[8]	0	0.2546	0.3123	0.0465
f[9]	0	0.0211	0.0305	0.4289
f[10]	0	0.0483	0.3671	0
f[11]	0	0.3189	0	0
f[12]	0	0	0	0
f[13]	0	0	0	0
f[14]	0	0	0	0

Table 8. pattern spectrum of "shadow1rotated"

The pattern spectrum distance of all objects is shown in table 9.

distable =

4×4 table

	object1	object2	object3	object4
object1	0.3146	15.499	0.43402	0.013807
object2	0.39371	3.202	0.095946	0.17563
object3	0.28789	3.5239	0.23949	0.34067
object4	0.38735	3.2603	0.0022077	0.39439

Table 9. pattern spectrum distance of "shadow1" and "shadow1rotated"

The result that which two images matches are shown in Figure 22 to Figure 25.

Figure 22. First two images matching

Figure 23. Second two images matching

Figure 24. Third two images matching

Figure 25. Forth two images matching

We can see that object1 and object 3, object2 and object 2, object3 and object4 and object4 and object 1 have minimum distance. It can be concluded that they have the same pattern.

D. Conclusion

In question one, the image and a set of eight different structural elements are selected to be processed for thinning transform. In each iteration, one layer of the target area (foreground) on the image will be burned off. After all iterations, homotopic skeletonization with a width of one pixel will be gained. This related operation has been popular in digital image processing, especially in OCR.

Size distribution can represent the number of patterns and be used to compute the pattern spectrum. Pattern spectrum represents the weight of each pattern and can be used to compute the complexity. Pattern spectrum distance can also be utilized to complete pattern recognition. The objects with the same patterns will have a relatively small distance. However, due to different

patterns with different proportion, major difference should be concentrated more. We should adjust the weight to compute distance in order to recognize perfectly.