

# DIP Homework #3

ID #: B04902028

Department & grade: CSIE 3

Name: 洪浩翔

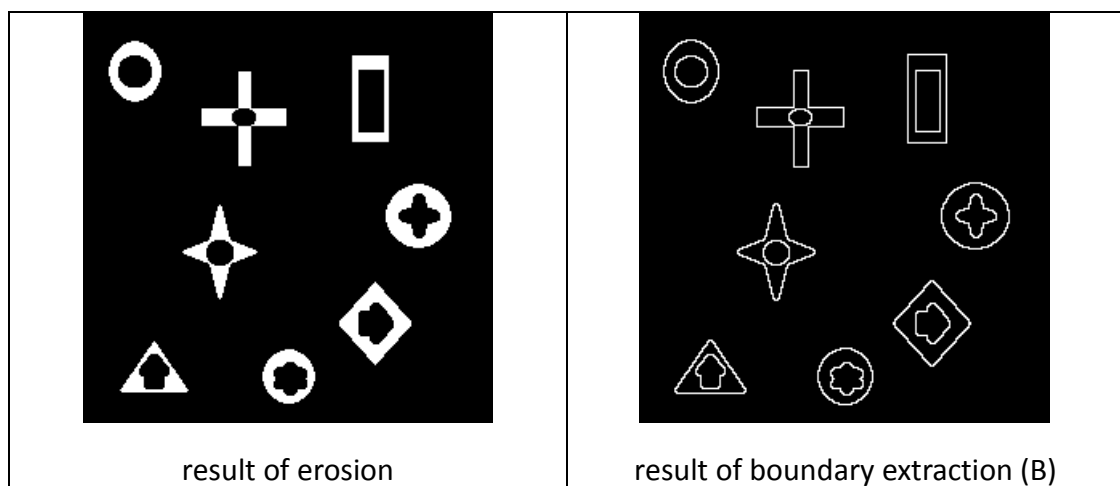
Email: b04902028@ntu.edu.tw

Submit time: 05/02/2018

## PROBLEM 1: Morphological Processing

Given a binary image I1 as shown in Fig. 1. White pixels represent the objects and black pixels represent the background. Please follow the instructions below to create several new images and describe the method in detail for each case.

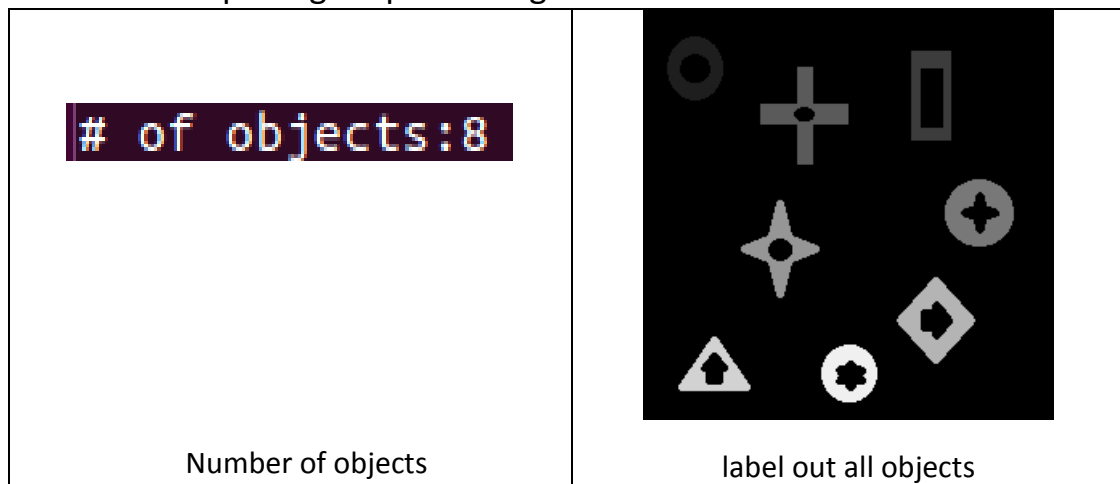
(a) Perform boundary extraction on I1 to extract the objects' boundaries and output the result as image B. Please provide some discussions about image B.



To extract the boundary, I erode I1 to get the erosion result in left figure. Then, I subtract I1 with the erosion result to get the boundary extraction result.

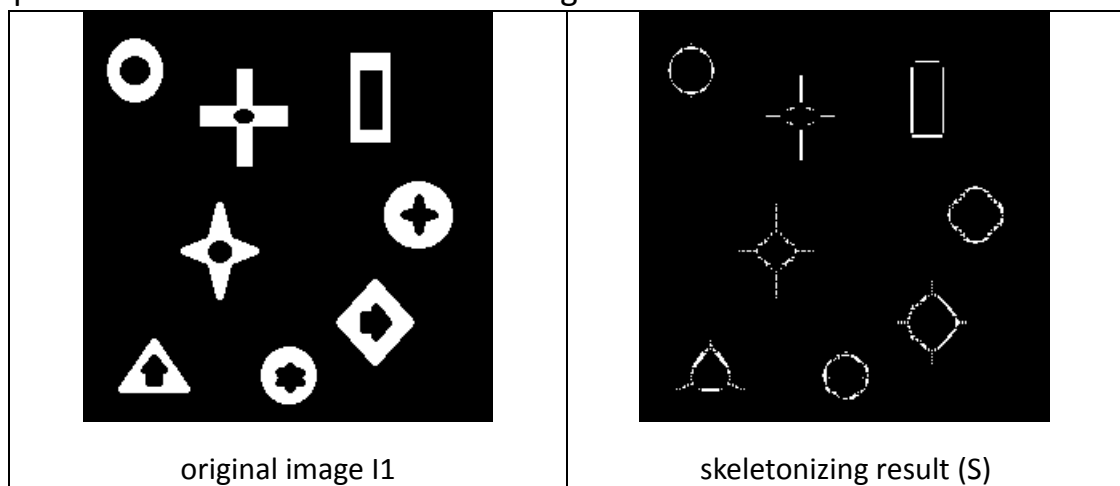
In B, the boundary is perfectly extracted with no error. It seems that the method is indeed useful for binary image. The explanation for this result is also easy. While doing the erosion operation, the outer sides of objects will be eroded. Therefore, after I2 subtracts the erosion result, only the outer sides eroded in erosion result will be remained, which is in fact the boundary.

(b) Please design an algorithm to count the number of objects in I1 based on morphological processing.



I use the connected components here to deal with the objects counting. The method is very straight forward. Just run connected components with 8-connected neighbor and record the label value which starts from 1. After calculating, the number of objects is just the label value minus 1. Therefore, in the end the number 8 is gotten after counting the objects in I1.

(c) Perform skeletonizing on I1 and output the result as image S. Please provide some discussions about image S.



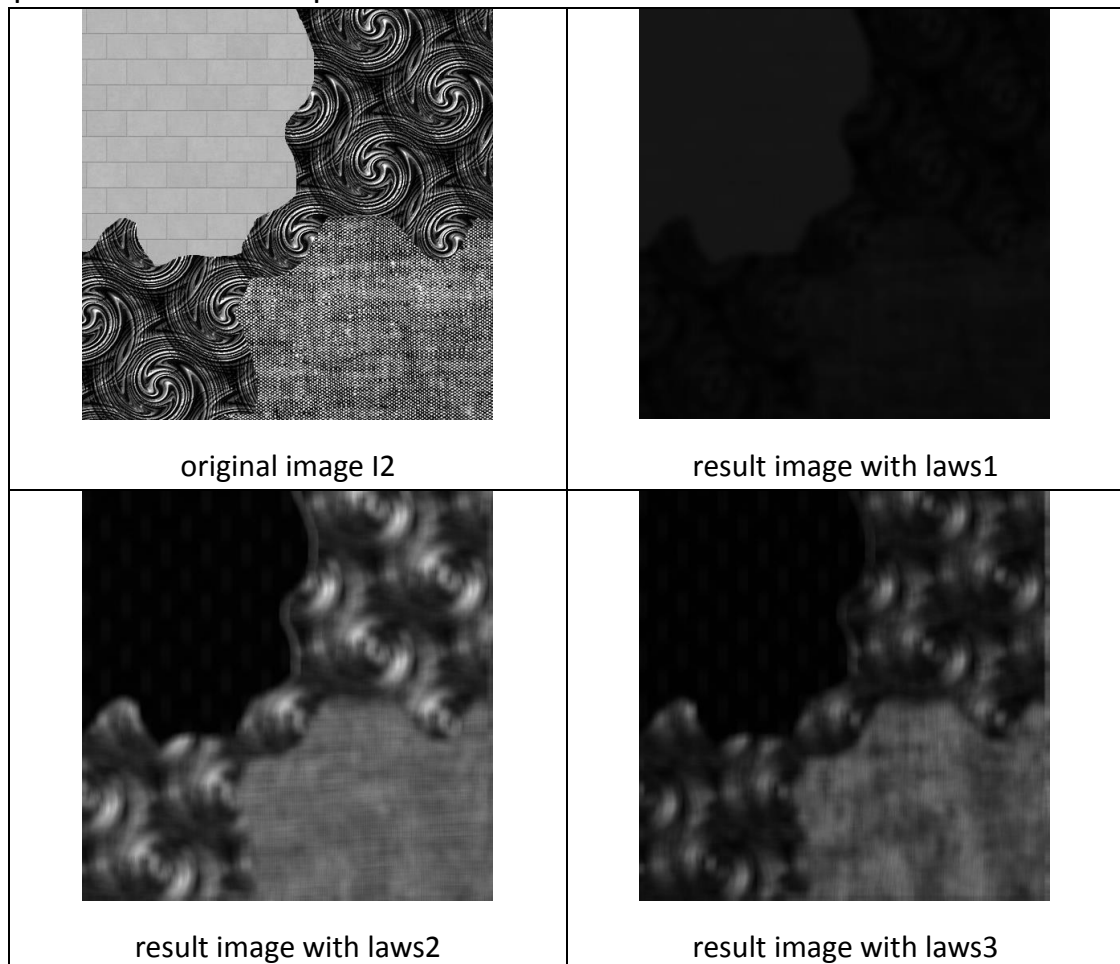
To perform skeletonizing, I use a method similar with what skeletonizing tool in MATLAB does, but I implement it in c++. First, I erode I1 to get A0, and do the opening operation on A0 to get B0. Second, make A0 subtract B0 to get G0. Then, erode A0 to get A1, open A1 to get B1, and A1 minus B1 to get G1. Repeat these steps to get A0...An, B0...Bn, and G0...Gn with  $n \geq 1$ . In last step, do union on G0...Gn to get the skeletonizing result. The most important part in this method is n should be taken carefully, because n should be large enough to ensure the result converging, but too large n will waste too much time and resources. In my homework, I use n =

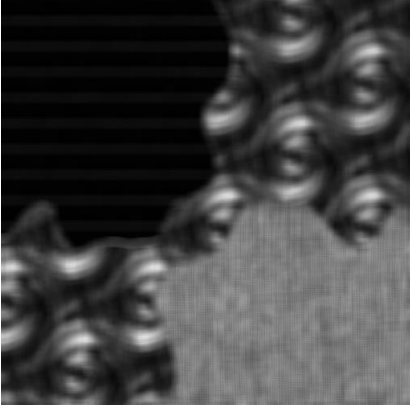
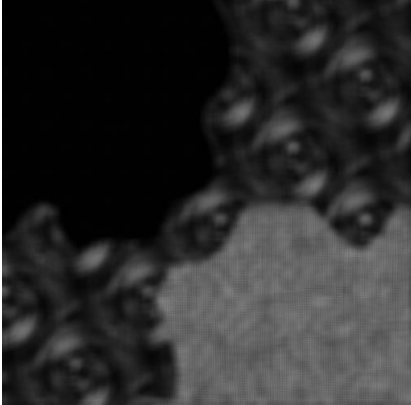
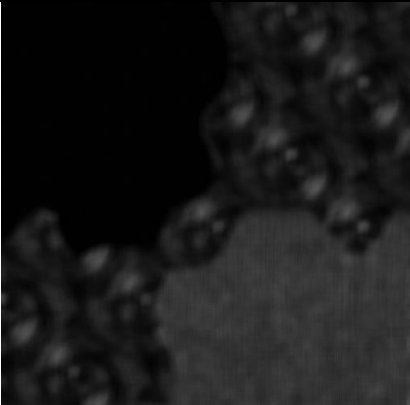
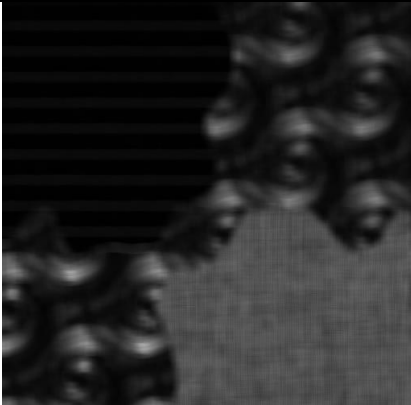
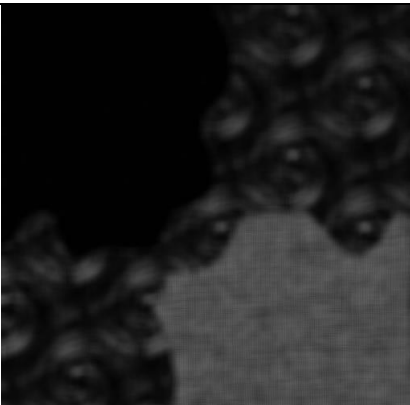
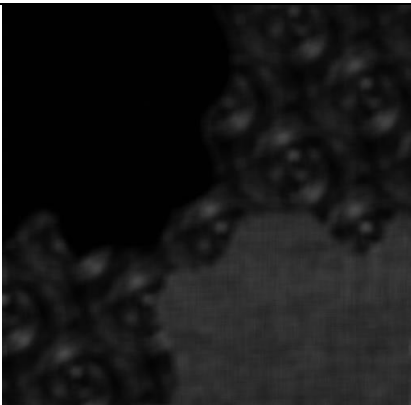
20 to do the skeletonizing and get the result S. In the result S, it also points out that  $n = 20$  is enough to produce the skeleton. Although this method is easy to implement, its result may not as good quality as using hit and miss. If zoom in S, it is not hard to find that the skeleton is not continuous. There are many apertures in the skeleton. Thus, this is indeed a trade-off between complexity and quality.

## PROBLEM 2: Texture Analysis

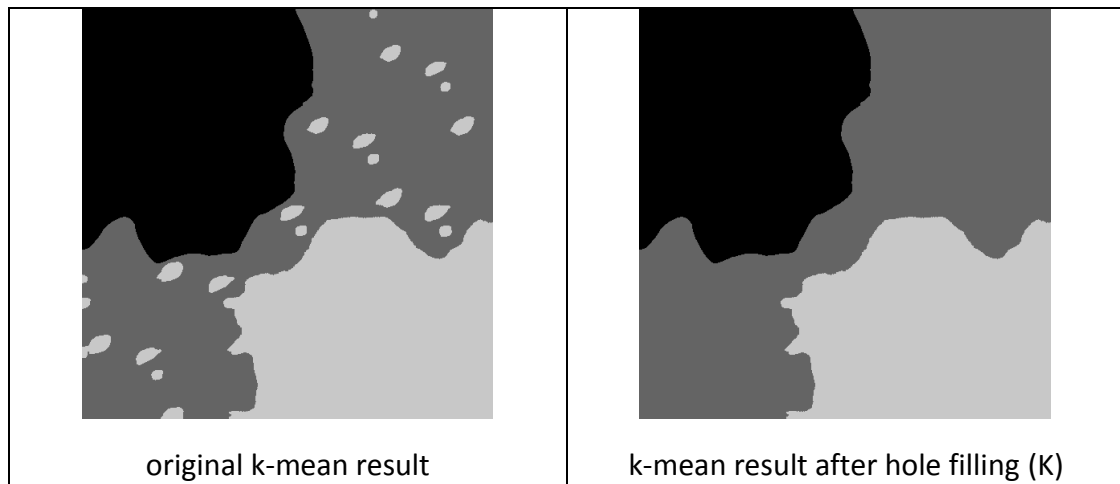
An image I2 which is composed of several different textures is given in Fig. 2.

(a) Perform Law's method on I2 to segment the image into 3 different texture groups. Label the pixels of the same texture group with same intensity values. Please detail the method you choose, specify all the parameters and output the result as K.



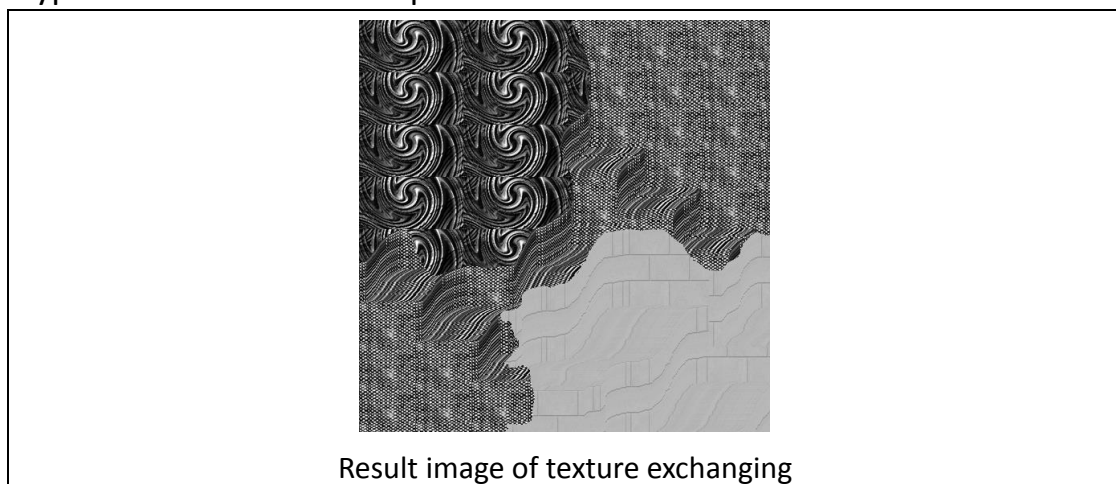
 <p>result image with laws4</p>	 <p>result image with laws5</p>
 <p>result image with laws6</p>	 <p>result image with laws7</p>
 <p>result image with laws8</p>	 <p>result image with laws9</p>

First, conduct step one of law's method to get 9 different convolution results. Each result is post above, and it is not hard to find that different convolution filters can produce different frequent results. In step two, I use  $\sum \sum |M_i(j + m, k + n)|^2$  with window size = 13 x 13 to compute the energy and get T1...T9. Then, use these 9 results to run k-mean algorithm to get the grouping result.



In k-mean algorithm, I set the initial mean points first and compute the distance of every pixel to these points. Depending on these distance value, classify each pixel to form 3 groups. Then, compute mean points again and so does distance, and group all pixels depending on the new mean points. Repeat this step until no points are changed. In the end, the result is the image in the left. It is obvious that there are many errors in the middle group. To improve the result, I use holes filling to fix the error label and get the result image K in the right, which separates different textures perfectly.

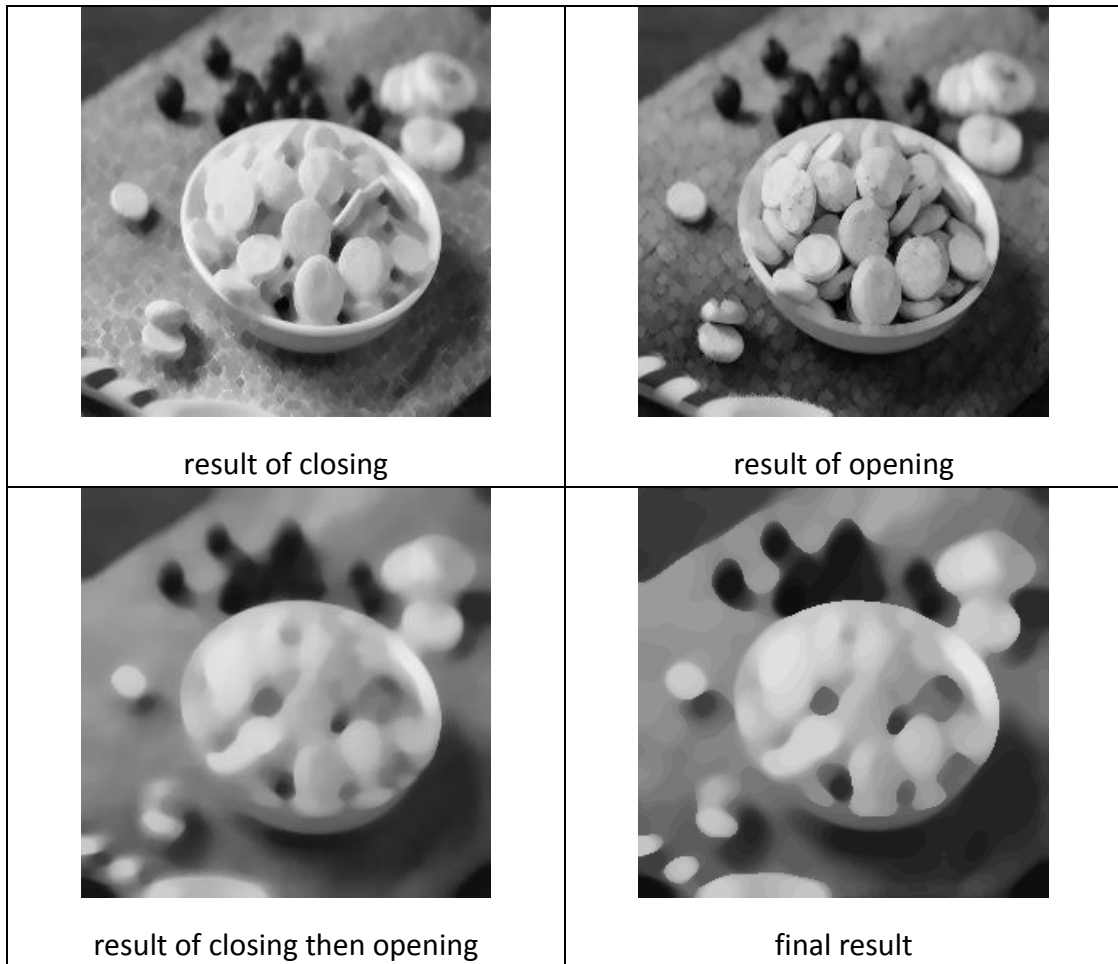
(b) Based on K, try to generate another texture image by exchange the types of different texture patterns.



I sample different textures manually and paste it onto other textures. For example, I sample the texture of the top-left of the original image, and paste it on the right-down part with the sample texture. This method is also easy to implement but the result may not be perfect. This can be found in the down-right part of the result image. The line should be straight line if the texture should be the same as the original texture. However, the line becomes curve here because of the pasting method. This condition causes the result becoming unreal, or saying, fake.

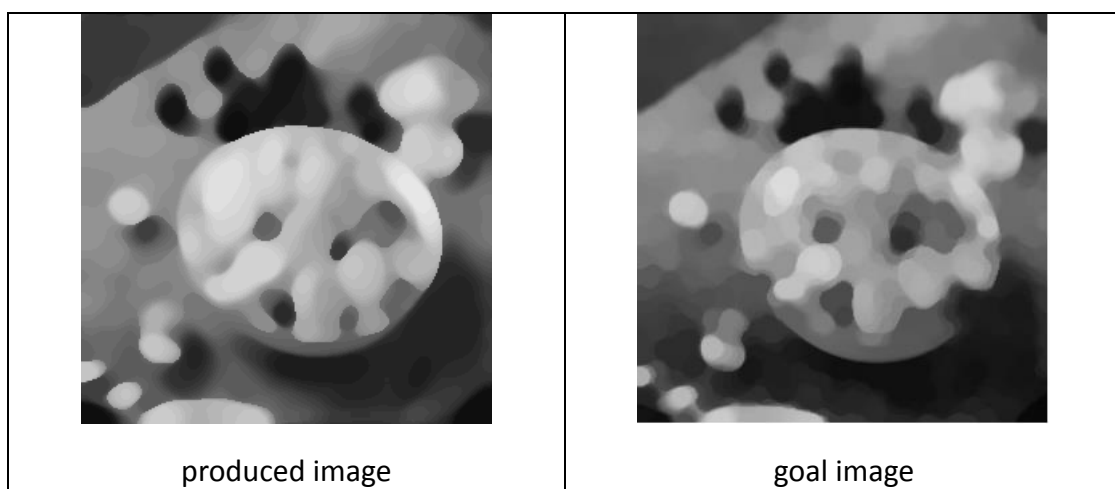
### [Bonus]

Given an image I3 shown in Fig. 3, please try to produce an image as illustrated in Fig. 4 by adopting appropriate morphological processing. Please describe the designed algorithm in detail and provide some discussions.



First, I implement closing and opening with octagon kernel and test the result to get the images above. It is obvious that the opening and closing operation will spread out the bright and dark value respectively. Therefore, I try closing then opening combined with median filter and get the result image at down left above. It is similar with the goal image but still not enough. To improve, I try to combine many different functions to reach the goal. In the end, I decide to use a structure “median filter – opening – closing – requantization with 5 – bound extraction – separate dilation or erosion – median filter – requantization with 7” to produce the final image. I will explain each step one by one. First, I use a median filter with 13 x 13 sizes to blur the original image. Second, do the opening then closing operation after median filter. Then, do the first time requantization. I divide every pixel value with 5 and store the result in integer, and multiply 5 back. This step is designed to produce the step-like

texture. Forth, extract the boundary of the result of previous step and try to smooth the boundary. This step is designed to smooth the boundary because the result of previous step still has an obvious boundary. If a pixel is not zero, it means that it is determined as boundary, and replaces it with the average value of its 4-connected neighbor. Fifth, set a threshold = 127, and if a pixel value is larger than the threshold, conduct the dilation operation on it, erosion operation otherwise. This step is designed to mix the both bright side and dark side pixel value, producing the effect of step-like overlapping. Sixth, do median filter again with size = 9 x 9. This step is designed to blur again, so as to smooth the image. The last step is to requantize again using 7, in order to emphasize the step-like effect. The final result is shown above.



Comparing the resultant image with the goal image, resultant image doesn't have an obvious step-like texture. In addition, goal image has a very large region of dark pixels at the down-right part. I still can't find a way to produce the effect like that. In the end, I just can produce an image similar to the goal image, and still can't find a good method to improve it. If I have an opportunity to know the process of the goal image, please tell me the detailed algorithm.

## **README**

```
# DIP Homework Assignment #3
# 05/02/2018
# Name: 洪浩翔
# ID: B04902028
# email: b04902028@ntu.edu.tw
# compiled on ubuntu 16.04.3 LTS with g++11
# output name and file is decided, so no need to change
# image path should be "../raw/*.raw" and output image will be here as well
# type make -f README to compile and run
# below is for makefile
```

```
.PHONY: all
```

```
CC=g++
```

```
LN=g++
```

```
All: prob1 prob2 prob3
```

```
prob1 :
```

```
    @echo "hw3-1"
    @echo "compiling and linking the code"
    $(CC) -std=c++11 -c hw3.cpp main.cpp
    $(LN) -std=c++11 -o hw3-1 hw3.o main.o
    @echo "running the program"
    ./hw3-1
```

```
prob2 :
```

```
    @echo "hw3-2"
    @echo "compiling and linking the code"
    $(CC) -std=c++11 -c hw3-2.cpp main2.cpp
    $(LN) -std=c++11 -o hw3-2 hw3-2.o main2.o
    @echo "running the program"
    ./hw3-2
```

```
prob3 :
```

```
    @echo "hw3-3"
    @echo "compiling and linking the code"
    $(CC) -std=c++11 -c hw3-3.cpp main3.cpp
    $(LN) -std=c++11 -o hw3-3 hw3-3.o main3.o
    @echo "running the program"
    ./hw3-3
```