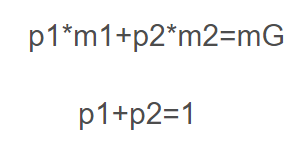
**Class Project 10**

Use two images for each operation to do the following operations and write down their advantages and disadvantages and explain your results:

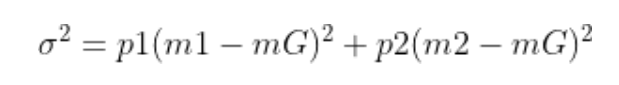
1. **Otus (large\_septagon\_gaussian\_noise\_mean\_0\_std\_50\_added, it’s 5\*5 smooth):**

**Algorithm:**

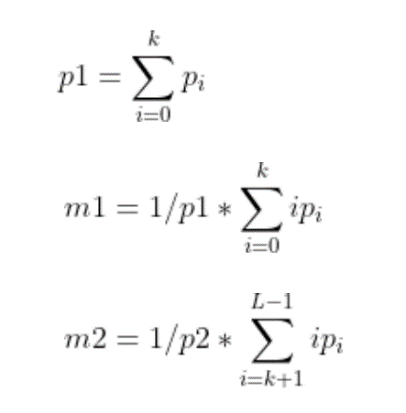
The assumption of the OTSU algorithm is that there is a threshold TH that decomposes all images into two categories C1 (less than TH) and C2 (greater than TH), then these two categories have their respective mean values of m1, m2, and the mean image size mG. The probabilities of being classified into C1 and C2 at the same time are p1 and p2, respectively. Therefore:



According to the concept of variance, the between-class variance expression is:



The gray level k that can maximize the above formula is the OTSU threshold.



According to the formula, it is ok to traverse 0~255 gray levels and find the largest k.

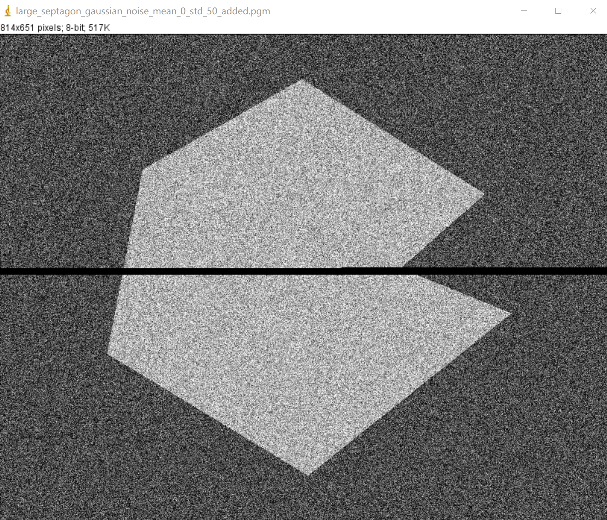
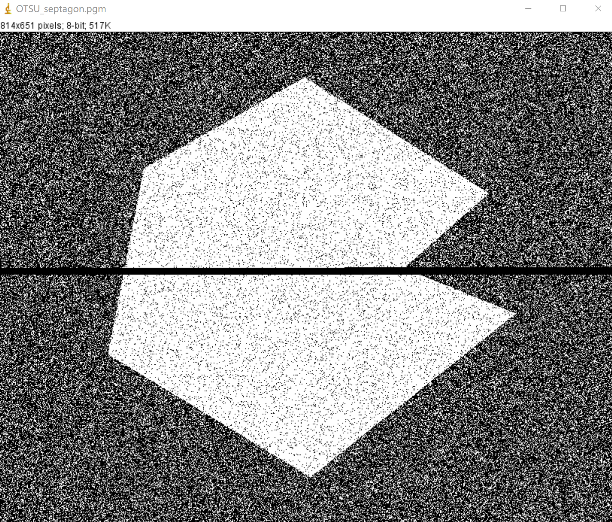
**Results (including pictures):**

Result of processing “large\_septagon

\_gaussian\_noise\_mean

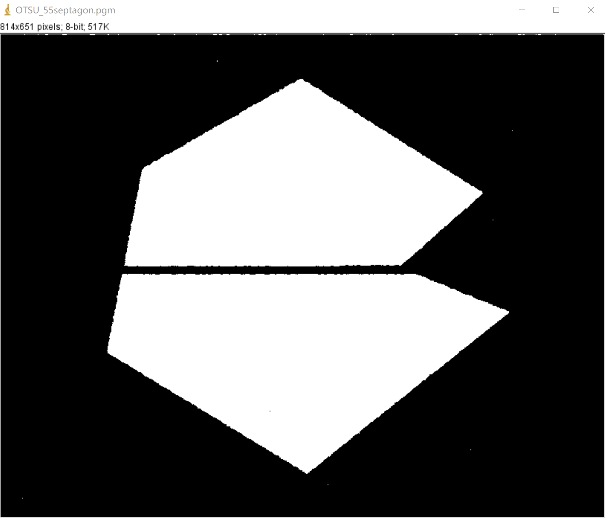
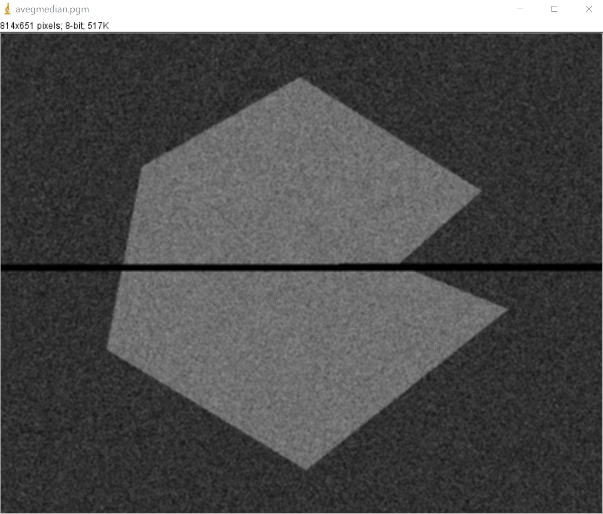
\_0\_std\_50\_added.pgm”:

Source Image: Result after Otus:



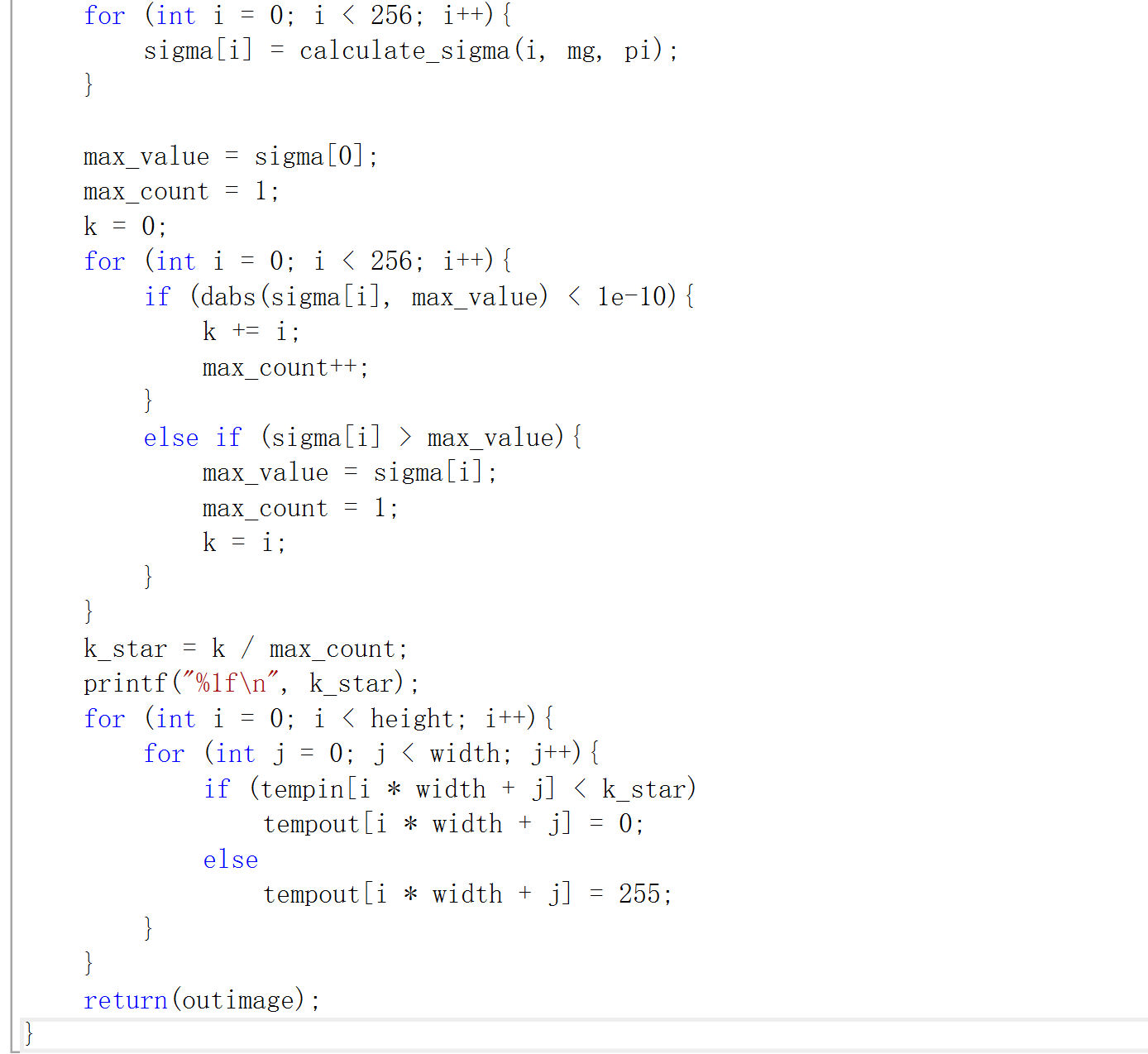
Result of processing “avegmedian”:

Source Image: Result after Otus:

****

**Discussion:**

OTSU is the best method to find the global threshold of an image. It goes without saying that it is suitable for most occasions where the global threshold of an image is required. The advantage is that the calculation is simple and fast, and is not affected by image brightness and contrast. The disadvantage is that it is sensitive to image noise; it can only be segmented for a single target; when the size ratio of the target and the background is large, and the variance function between classes may show double peaks or multiple peaks, the effect is not good at this time.

**Codes:** 

1. **Partition (septagon\_noisy\_shaded):**

**Algorithm:**

Otus’s method is told in 1.

Partition:

One of the simplest methods of variable threshold processing is to divide an image into non-overlapping rectangles. This method is used to compensate for the unevenness of illumination and or reflection. The selected rectangle should be small enough so that the illumination of each rectangle is approximately uniform.

Divide a picture into the same or different number of rectangles along the horizontal and vertical directions. For each rectangle:

For p from 0 to rectangle width

For q from 0 to rectangle height

For i from 0 to imagewidth / rectangle width

For j from 0 to imageheight / rectangle height

Temp[imagewidth / rectangle width \* i + j] = inimage[(i + q \* imageheight / rectangle height) \* imagewidth + p \* imagewidth / rectangle width + j]

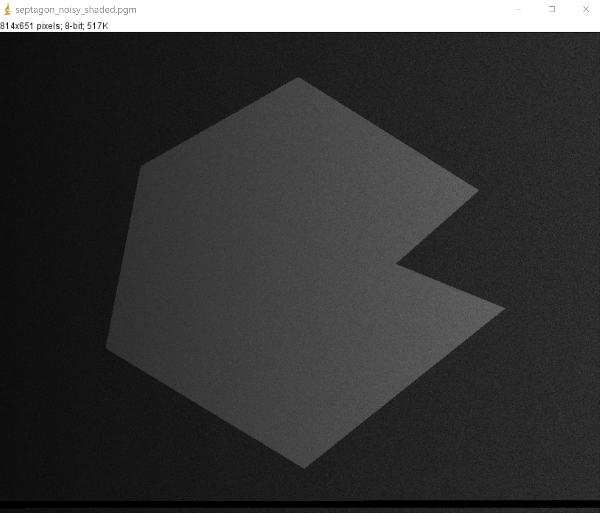
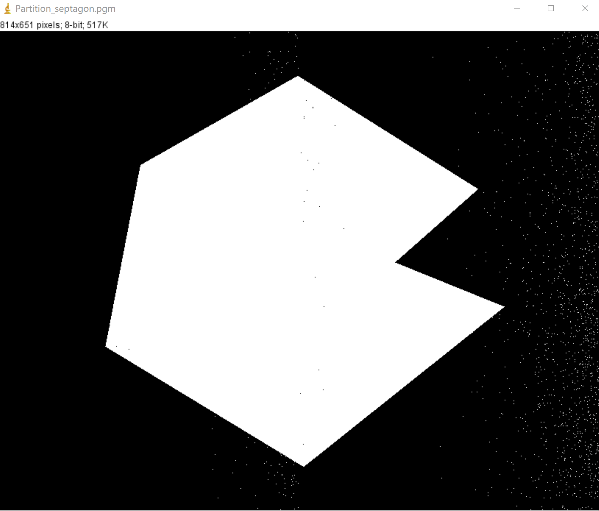
For temp use Otus

Then do the change to out image

**Results (including pictures):**

Result of processing “septagon\_noisy\_shaded.pgm”:

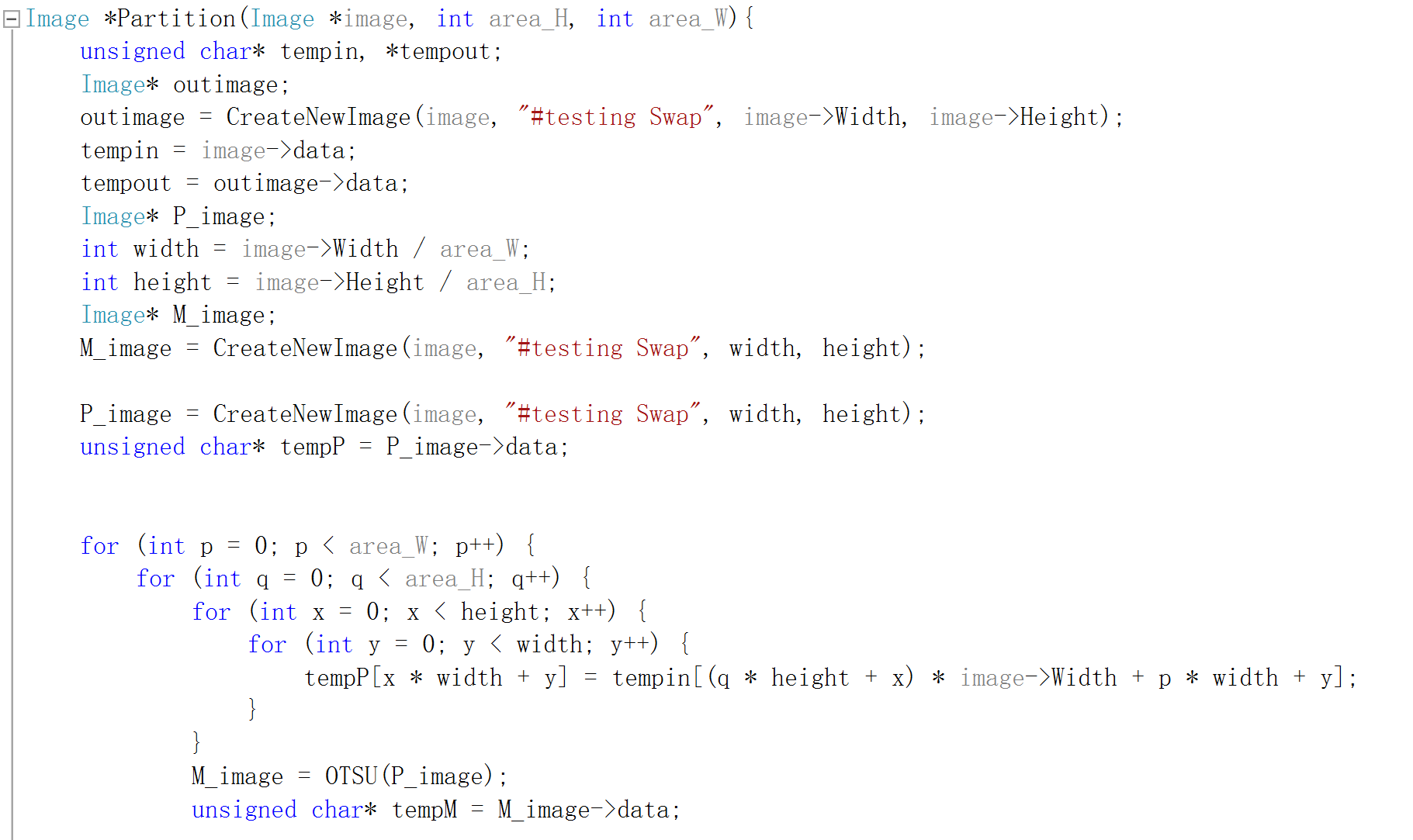
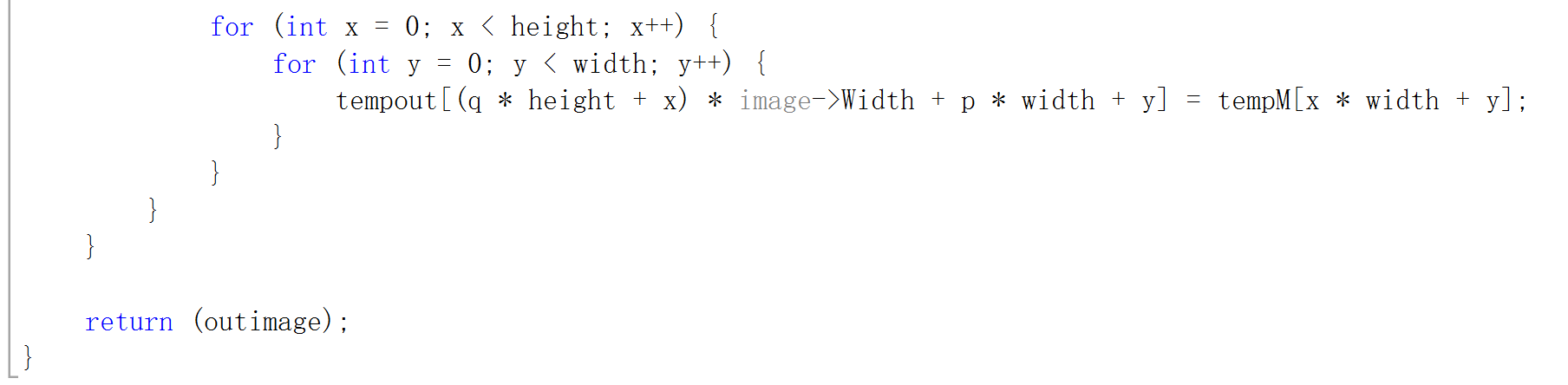
Source Image: Result after Partition:



**Discussion:**

One of the simplest methods of variable threshold processing is to divide an image into non-overlapping rectangles. This method is used to compensate for the unevenness of illumination and or reflection. The selected rectangle should be small enough so that the illumination of each rectangle is approximately uniform. Due to a series of problems such as noise or uneven illumination, it is impossible to directly perform threshold segmentation. Image smoothing and edge information are beneficial to threshold processing. However, in more frequent cases, the above two methods have no obvious effect and can only be used. Change the threshold to solve it.

**Code:**

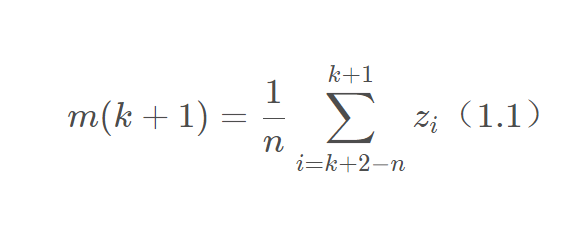
**** ****

1. **Moving average thresholding (****spot\_shaded\_text\_image):**

**Algorithm:**

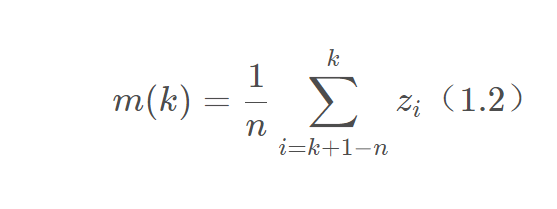
Compared with taking a fixed threshold for the entire image, due to uneven illumination and other reasons, the general method is to calculate the threshold for each pixel in an image, which can change the local threshold.

The algorithm is expressed as follows:

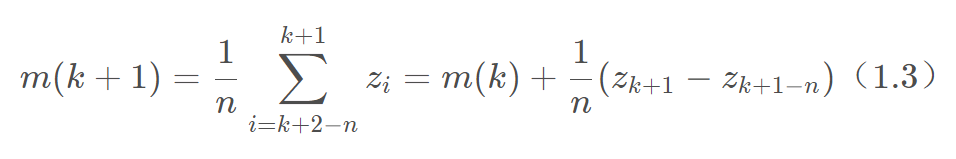


Where represents the gray value of the point encountered in the k + 1 step of the scan sequence, m (k) is the pixel value of the k point of the input image, and n represents the number of points used to calculate the average.

According to (1.1), there is the following formula:



According to (1.1) and (1.2), it is not difficult to get the following formula:

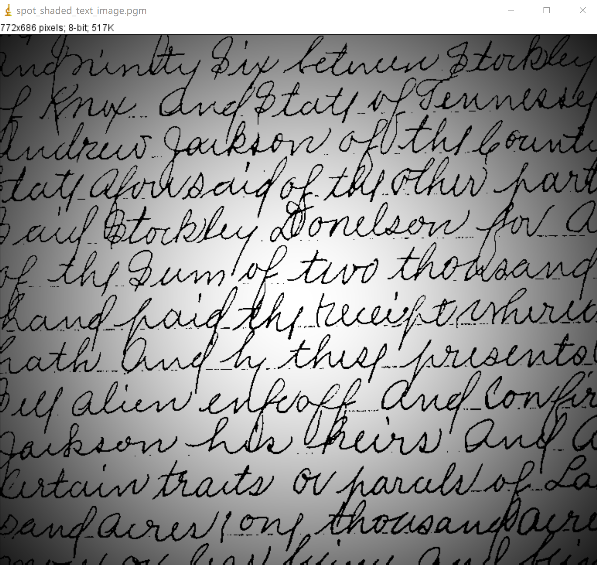
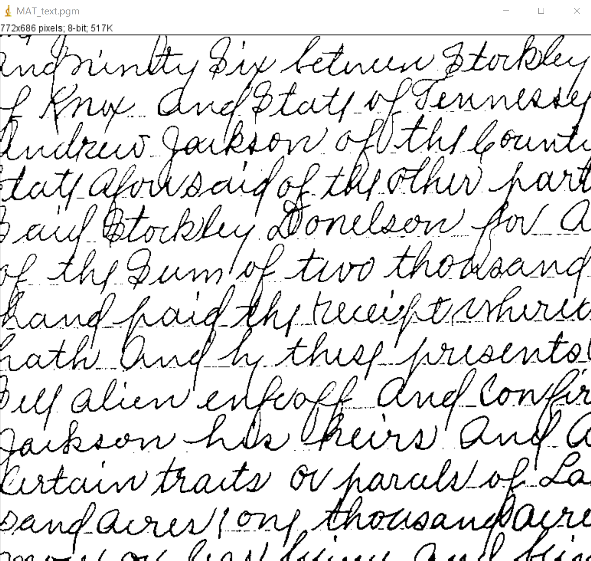


Next, perform the binarization operation according to the threshold of each point.

**Results (including pictures):**

Result of processing “spot\_shaded\_text\_image.pgm”:

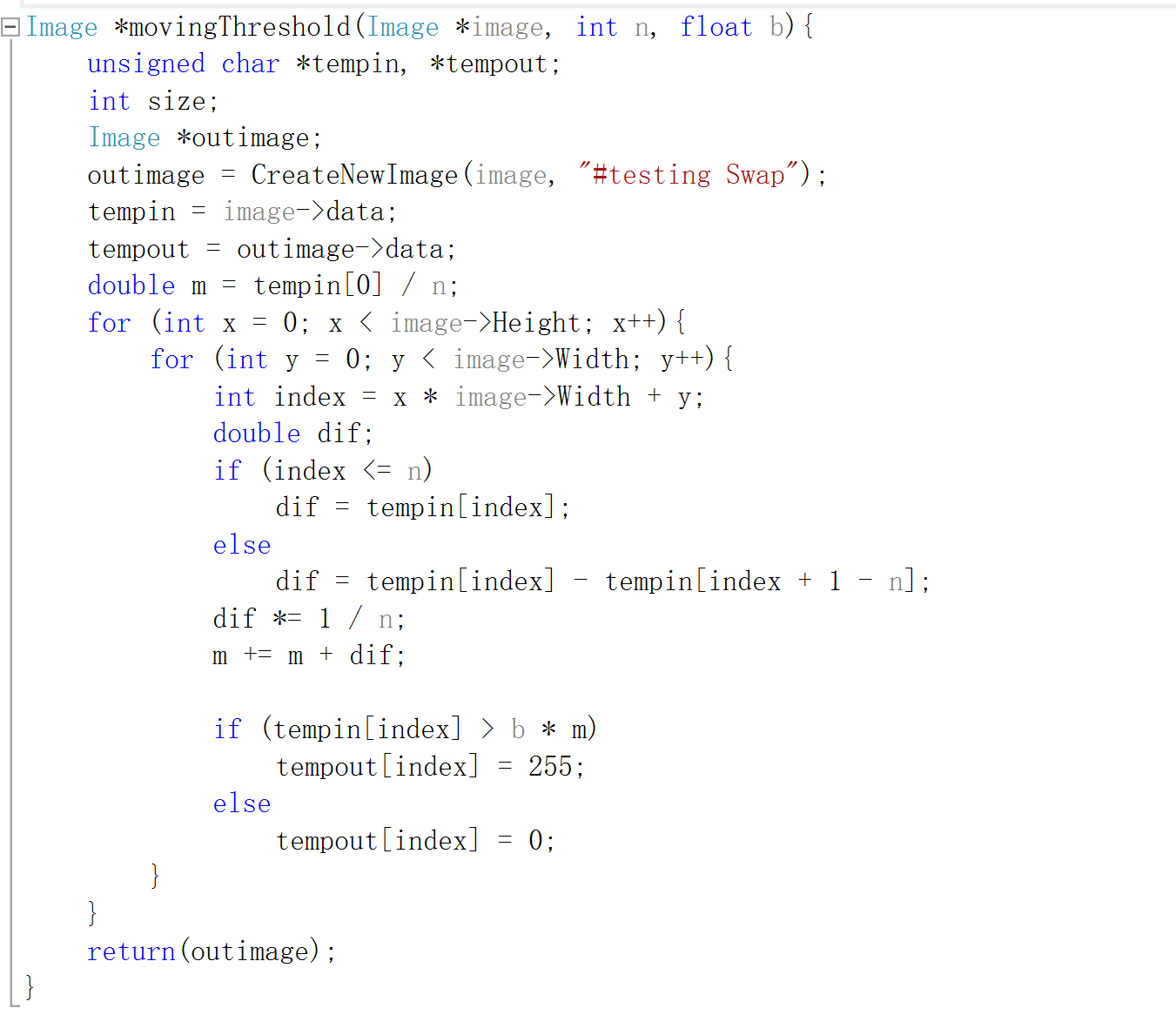
Source Image: Result after Moving average:



**Discussion:**

The moving average method is a kind of variable threshold processing. The variable threshold is relative to the global threshold processing. The global threshold processing refers to calculating a fixed threshold based on the entire picture. If each pixel in the picture is larger than this The value is considered the foreground, otherwise it is the background. The variable threshold means that the pixel or pixel block at each position in the picture has a different threshold. If the pixel is greater than its corresponding threshold, it is considered as the foreground. The moving average method scans the entire picture in a linear zigzag shape, and a threshold is generated at each point, and the gray value at that point is compared with the threshold calculated at that point to segment the picture.

**Code:**



1. **Region growing (****defective\_weld, noisy\_region):**

**Algorism:**

The basic idea of the region growing algorithm is to merge pixels with similar properties. For each area, a seed point must be designated as the starting point for growth, and then the pixels in the surrounding field of the seed point and the seed point are compared, and points with similar properties are combined to continue growing outwards until no pixels that meet the conditions are Include until it comes in. The growth of such an area is complete.

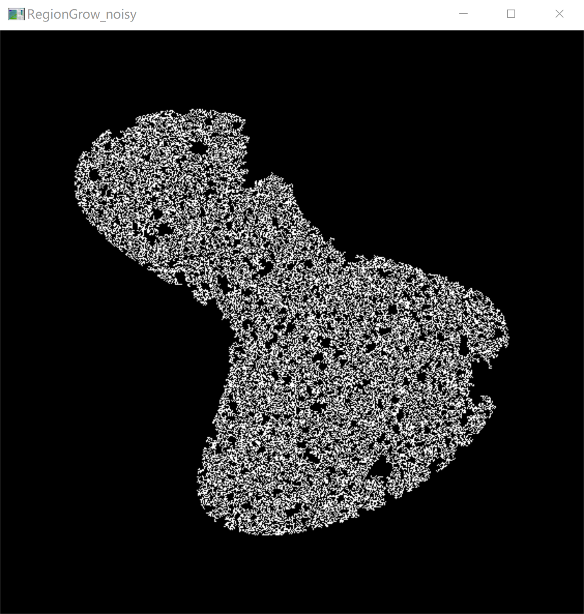
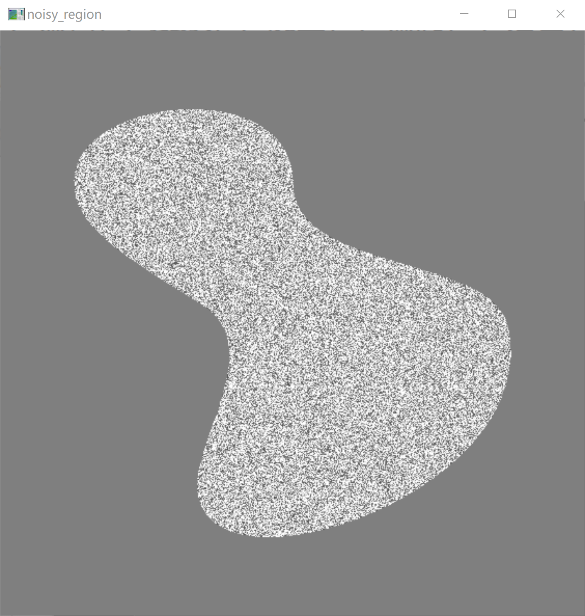
The steps to achieve regional growth are as follows:

1. Scan the image sequentially! Find the first pixel that has not yet belonged, and set this pixel as (x0, y0);
2. Taking (x0, y0) as the center, consider the 8 neighborhood pixels (x, y) of (x0, y0), if (x, y) meets the growth criterion, combine (x, y) and (x0, y0) Merge (in the same area), and push (x, y) onto the stack at the same time;
3. Take a pixel from the stack, treat it as (x0, y0) and return to step 2;
4. When the stack is empty! Return to step 1;
5. Repeat steps 1-4 until each point in the image has an attribution. The growth is over.

**Results (including pictures):**

Result of processing “defective\_weld.pgm”:

Source Image: Result after region growing:

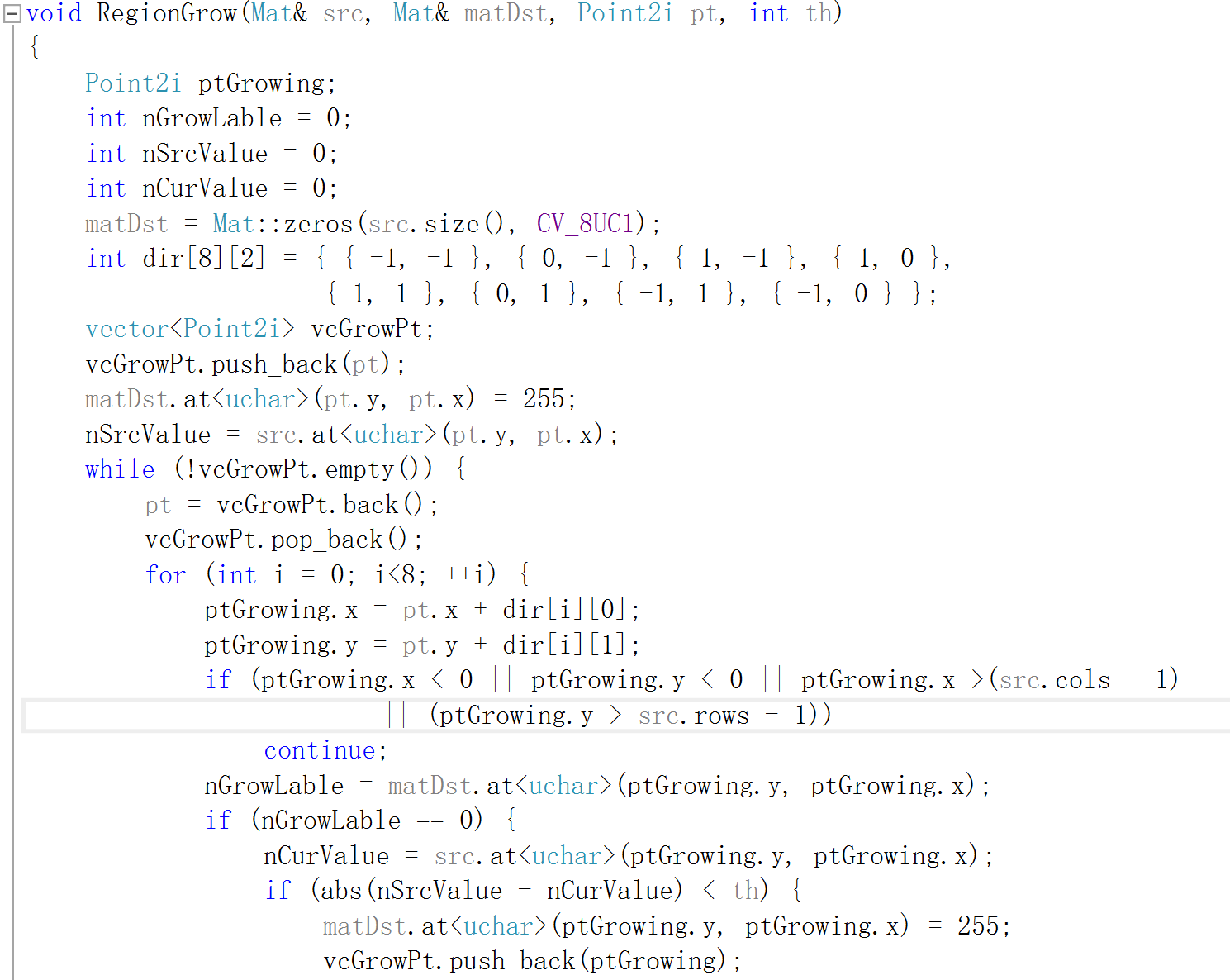
Result of processing “noisy\_region.pgm”:

Source Image: Result after region growing:

**Discussion:**

Region growing is an image segmentation method of serial region segmentation. Its advantage is that the basic idea is relatively simple. It can usually segment the connected regions with the same characteristics and provide good boundary information and segmentation results. When there is no prior knowledge available, the best performance can be obtained, and it can be used to segment more complex images, such as natural scenery. However, the region growing method is an iterative method, and the space and time costs are relatively large. Noise and gray-level unevenness may cause holes and over-segmentation, and it is often not very good in processing the shadow effect in the image.

**Codes:** ****