Chapter 3

Error Correction of Secret Images by Search-Order Coding

In this chapter, an image recovery method is presented. By using the method of search-order coding (SOC) which was proposed in [30], an SOC-image of the secret image can be generated. If the secret image is damaged or modified during recovery, by referring to the SOC-image and repairing the damaged image, people can still obtain a version of the secret image that looks not bad. Besides, the information of the secret image cannot be retrieved from the SOC-image alone. In other words, the SOC-image is useful only when it is together with the secret image (no matter damaged or not). In addition to the simple version of the SOC-image, a modified version of the SOC-image is also proposed. The modified version is a more flexible method and can be applied during repairing. Then, the secret image can still be reconstructed even when it is *seriously* damaged. The structure of this chapter is described below. Section 3.1 is the introduction. Sections 3.2 and 3.3 present the simple and modified versions, respectively, of the image recovery method using SOC. The experiment results are shown in Section 3.4 and the summary is made in Section 3.5.

3.1 Introduction

Computer network nowadays is convenient so that images can be transmitted via network. However, since network is an open environment, a secret image may be intercepted or damaged during its transmission. Since the original image is secret, to back up using a duplicated secret image or double transmission is not smart (this increases the chance that the secret get pirated). Thus, before transmission, people may transform the secret image into several shadow images by some sharing methods. Although this increases the security, the situation that a secret image is damaged is still very likely to happen; the reason is that if sufficient number of the shadow images is damaged during transmission, then the information about the secret image will go wrong during the recovery. To reduce the risk, and to achieve the goal of correcting the possibly damaged image (and huggermuggering the information about the secret image), an error correction method by SOC is proposed. Besides transmitting the shadow images, people may transmit an extra image called SOC-image. If the secret image is damaged during the recovery phase, a version of not-bad quality for the secret image may still be reconstructed with the help of the SOC-image. Of course, as stated earlier, before combining the SOC-image with the secret image (damaged or not), it is required that no information about the secret image can be found using the SOC-image alone.

Notably, if the recovered secret image is damaged too much, then the SOC-image cannot repair the broken image. Thus we also propose a modified version. In this modified version, the damaged image can then still be recovered when it is serious broken. The anti-theft principle is not changed, namely, the information about the secret image cannot be gotten from the SOC-image alone. In this new version, a small hash table is needed when we construct the SOC-image.

3.2 Error Correction by SOC (the Basic Version)

The Phase of Constructing SOC-image:

The method proposed here to produce SOC-images is somewhat like the one in [30]. However, the coding values used here are the gray value of pixels rather than the code indices. For every pixel, its searching range is limited to its "*previous*" points. The *previous* points of a pixel *p*=(x, y) are the collection of all pixels whose rows are above Row x, together with those pixels not only in Row x but also staying at the left of *p*. For example, in the 7 x 5 image shown in Fig. 3.1, the previous points of the pixel *P*=(4, 5) are those 3x7+4=25 points in the shaded region.

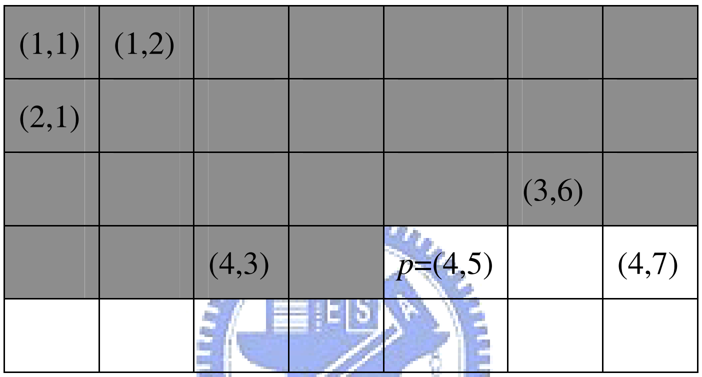

Fig. 3.1 The previous points of (4, 5) (the shaded region).

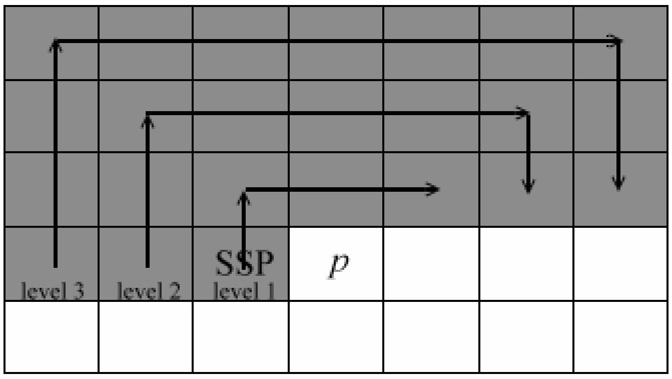

Fig. 3.2 The search order of p.

In the beginning of the search, the starting searching point (SSP) must be specified. After the specification of the SSP, the search order is then fixed. For example, in Fig. 3.2, if the SSP is the left point of the pixel *p*, then the search order is as shown in Fig. 3.2.

The search starts at SSP, and is level 1. At the end of each level, the search goes to next level and continues finding the pixel. In the very beginning of the search, the searching count is set to 1. If the gray value of the SSP equals the gray value of *p*, then the gray value of the pixel in the SOC-image and with identical position as *p* is set to 1. Otherwise, we find the next search point according to the order shown in Fig. 3.2, and add 1 to the searching count. Notably, since the gray values of neighborhood pixels are often similar, each gray value of the searching area may repeat frequently. Thus the searching count need not be added up if the gray value of the candidate point currently inspected has appeared before during this search. The search continues until a point whose gray value is the same as *p* is found, and the "searching count" is written to the SOC-image at position *p*.

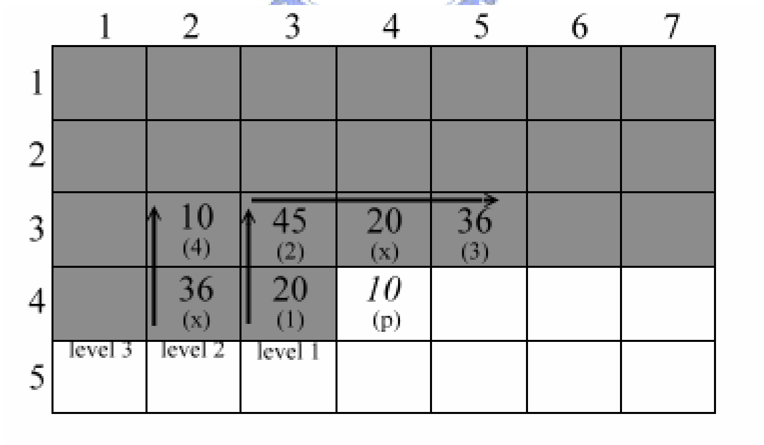

Fig. 3.3 The example of finding searching count.

For example, if the gray value of *p* (4, 4) shown in Fig. 3.3 is 10, and if the SSP of *p* is (4, 3), whose gray value is 20. These two gray values are not the same, thus 1 is added to the searching count. Then, the search continues according to the searching path. The gray value of the next point (3, 3) is 45, which is not equal to 10, so the search must proceed. When the search comes to the point (3, 4), whose gray value is 20. Since the gray value 20 already appeared earlier during this search (in fact, it is the gray value of the point (3, 3)), there is no need to increase the searching count. Finally, the gray value (10) been looked for can be found at the position (3, 2), and the searching count "4" at that moment is record to the SOC-image at the position of *p*. When all the previous points are searched and there is no point whose gray value is the same as *p*, then the value at position *p* in the SOC-image is set to 0. Since there are 8 bits can be used to record the search-order, the order's range is from 1 to 255 (as for 0, it is reserved to indicate the case when none of the points in the searching area has same gray value as *p*). There is only one condition in which the point whose gray value is the same as *p* in *p*'s previous points does exist, but the point's order cannot be recorded. The condition is that there are 256 different gray values in the original image and there are 255 different gray values occur in the searching process before finding the point with the same gray value as *p*'s. The search is repeated for every pixel of the original image, and finally an image which is encoded by SOC is constructed.

The Phase of Error Correction by SOC-image:

When we receive an image that may be damaged, the damaged image can be recovered partially and becomes a better quality image with the help of the SOC-image. In the first stage, all the pixels in the original (non-damaged) image are divided to 256 groups using the SOC-image. The grouping starts from the first pixel of the SOC-image. And the next grouped pixel is decided by a raster scan order, i.e. from left to right and top to bottom. If a pixel "*p*" is recorded as 0 in the SOC-image, it is allocated to a new group. Else the previous point is searched and found according to the value in the SOC-image. The only previous point can always be found as long as the SOC-image is correct. Thus *p* is merged to the group of the previous point being found. After grouping, the number of groups should be the same as the number of different gray values contained in the original image. Then the number of different gray values occurred in every group is calculated for the damaged image. If the image was not destructed, there must be only one gray value in each group. Thus the only gray value is assigned to the group. However, if the image is damaged, there will be different gray values in one group. And the possible value is decided by the gray value which appears most often. For example, in a group, if there are three different gray values 10, 73, and 240, respectively, and their appearing frequencies are 1547, 1036, and 4810, correspondingly. The most possible gray value of this group is set to 240 since the value appears most often. As a result, the damaged image can be recovered to a better quality image by checking the most possible gray value in every pixel's group.

3.3 Advanced Version of Error Correction Using SOC

The Phase of Constructing SOC-image (the Advanced Version):

Besides the version shown above, another convenient error correction version by SOC-image is proposed here. In the pre-processing stage, the 256 different gray values are divided to 128 groups. For example, Group 1 consists of the gray values 0 and 1, Group 2 consists of the gray values 2 and 3, and so on. After that, an invertible hash table which is one to one and onto is produced. The hash table scrambles each of the 128 groups to a new number ranged from 0 ~ 127. Thus the pre-processing is finished. Then, the SOC-image is calculated as in the above version (Sec. 3.2). However, the search of the pixel *p* stops when it meets a point whose group (*not the* *gray value*) is the same as *p*'s. The searching count then is recorded to the SOC-image at position *p*, and the first bit is set to 0 which denotes that the pixel value is the searching count (since there are 128 groups, the range of the searching count is from 0 to 127. Thus only 7 bits are needed to record the searching count). If there is no point whose group is the same as *p*; then, instead of 0, the value at the position *p* in the SOC-image is set to the hashed value of *p*'s group, and the first bit is set to 1, which denotes that a pixel value is the hashed value. After processing the search for each pixel of the original image, the SOC-image of the original image is constructed.

The Phase of Error Correction by SOC-image (the Advanced Version):

In the recovery phase, two different recovered ways are possible according to the availability of the hash table. Using an error correction system similar to the one proposed above in Sec. 3.2, for the basic version, the damaged image can be repaired roughly by the (advanced) SOC-image without the hash table. For example, if the first bit of position *p* in the SOC-image is 1, it means that the value of the last 7 bits is a hashed value. Since there is no hash table, the real gray value of pixel *p* is unknown. Thus the pixel *p* is allocated to a new group. On the contrary, if the first bit of position *p* in the SOC-image is 0, it means the value of the last 7 bits is a searching count. Then the previous point is searched (and found), and *p* is merged to the group of its previous point. After the grouping and the calculation of the most possible value of each group, the damaged image may become better. On the other hand, if the hash table is available, the hashed value can be returned to its original value. By the SOC-image and the hash table, a very good quality of image can be recovered from the damaged image. However, since the grouping of the original image merged two different gray values to one group (e.g.: the gray values 2 and 3 are in the same group), an adjustment must be made to reduce error. By the hash table, the two gray values of a group can be known (e.g.: 70 or 71). In the damaged image, if the gray value of an already-known pixel in this group is 71, the gray value of this new pixel is set to 71 in the recovered image instead of being set to 70. However, if the gray value of the already-know pixel is not 71, the new gray value is set to 70 in the recovered damaged image. With the adjustment, the quality of the recovered image will be better than the one without the adjustment. Of course, in the SOC-image producing phase, if the SOC-image takes 9 bits per pixel, then the original image can be divided to 256 groups, that is, every gray value has its own group instead of two gray values being shared in one group. Then, the lossless image can be recovered using the SOC-image and the hash table.

3.4 Experimental Results

Fig. 3.4 is the original image "Pepper" whose size is 512 x 512. The basic SOC-image of Fig. 3.4 is shown in Fig. 3.5. Fig. 3.6 presents a damaged image of Fig. 3.4 (contaminated by some noise). The PSNR of Fig. 3.6 is about 24.78 dB. After the repairing using the basic SOC-image shown in Fig. 3.5, an image of a quality better than that of Fig. 3.6 can be constructed, which is shown in Fig. 3.7. The PSNR of Fig. 3.7 is around 43.61 dB.

In the second example, Figures 3.8 and 3.9 show the original image "Lena" and its basic SOC-image, respectively. Figures 3.10 and 3.11 are, respectively, the damaged image and the recovered image using the basic SOC-image. The PSNRs of these two images are 19.95 dB and 53.95 dB, correspondingly.

The advanced version is also tested. The secret image used is the "Pepper" shown in Fig. 3.4. Fig. 3.12 is the SOC-image generated by the advanced version. A more severely damaged image is shown in Fig. 3.13, whose PSNR is 21.90 dB. If the damaged image is repaired by the advanced SOC-image (Fig. 3.12) alone, the user can get Fig. 3.14. Finally, a repaired image of much better quality which is Fig. 3.15, can be reconstructed by using the advanced SOC-image and the hash table. The PSNRs of Figures 3.14 and 3.15 are 28.35 dB and 51.15 dB, respectively.

3.5 Summary

In this chapter, we introduced the secret image recovery method by search-order coding (SOC). With the auxiliary image generated by SOC, the damaged image can be repaired to a better quality. Although the user can see some contour from the SOC-image, this can be solved by scrambling the image by a key. Thus, the SOC-image does not divulgate the information about the secret image. It provides a better method for safe-keeping than just backing up the original image by duplicating. By the advanced version described in Section 3.4, a flexible way of SOC-image can be used. If the hash table is not available, then the user can repair the damaged image by the SOC-image (advanced version) like the basic version. However, if the hash table is received, a good quality of secret image can be reconstructed using both the SOC-image (advanced version) and hash table.

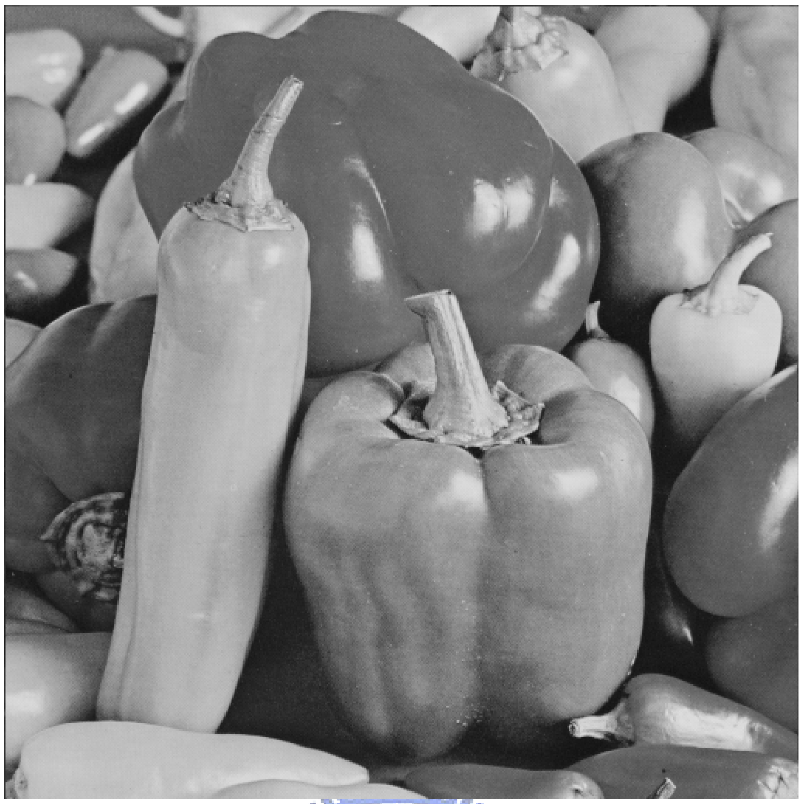
Fig. 3.4 The original image “Pepper”.

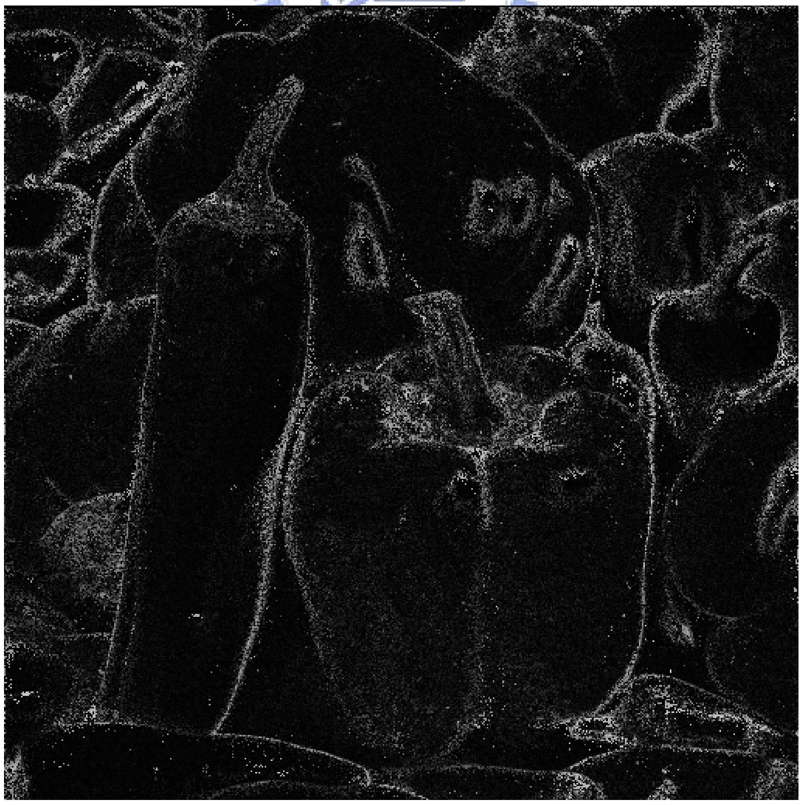
Fig. 3.5 The SOC-image (basic version) of Fig. 3.4.

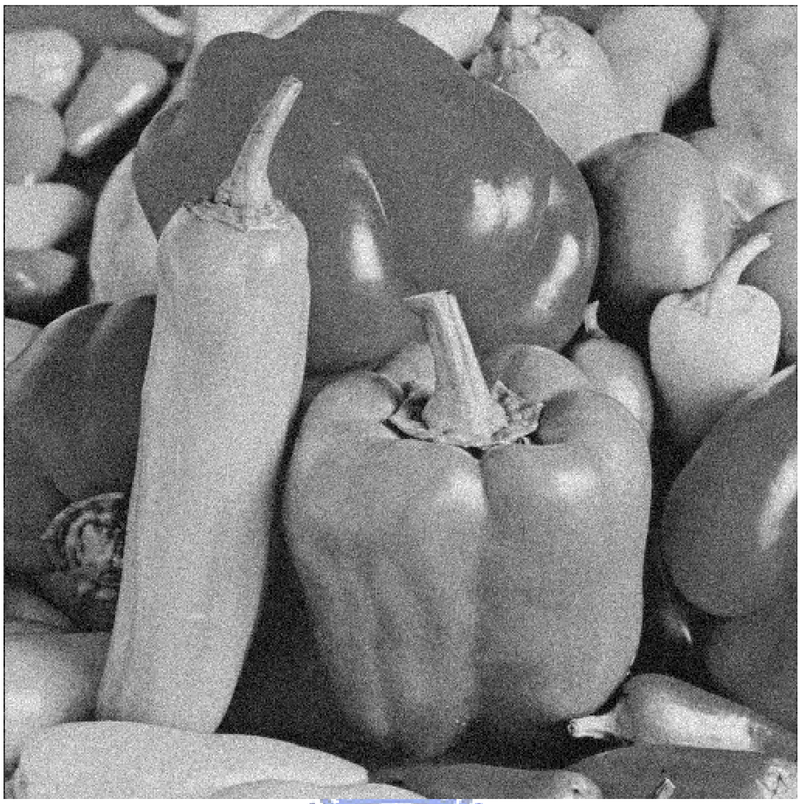

Fig. 3.6 The damaged image of Fig. 3.4. The PSNR is about 24.78 dB.

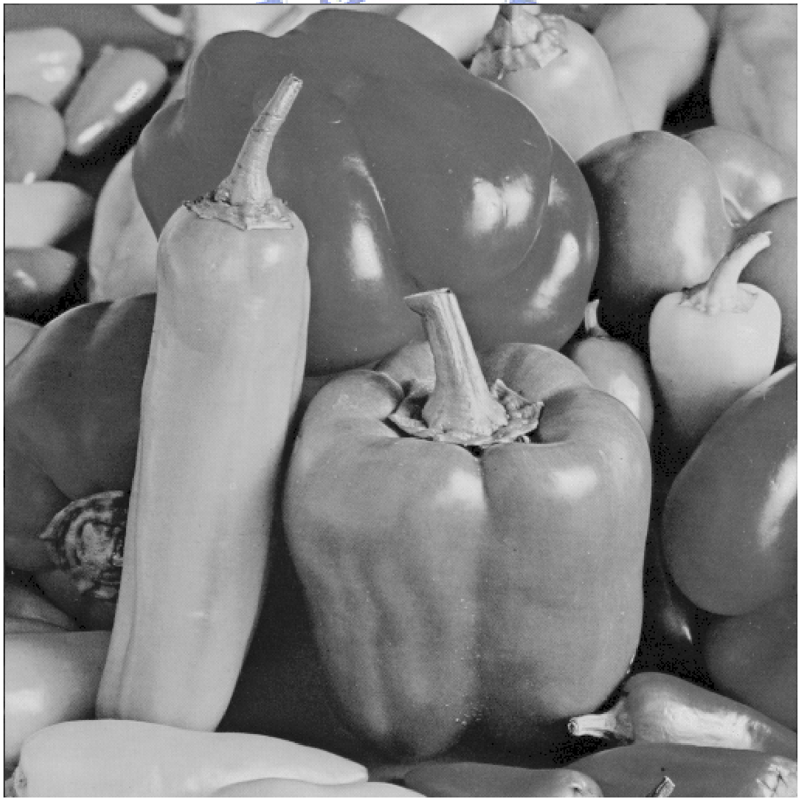

Fig. 3.7 The repaired image of Fig. 3.6 using Fig. 3.5. The PSNR is 43.61 dB.

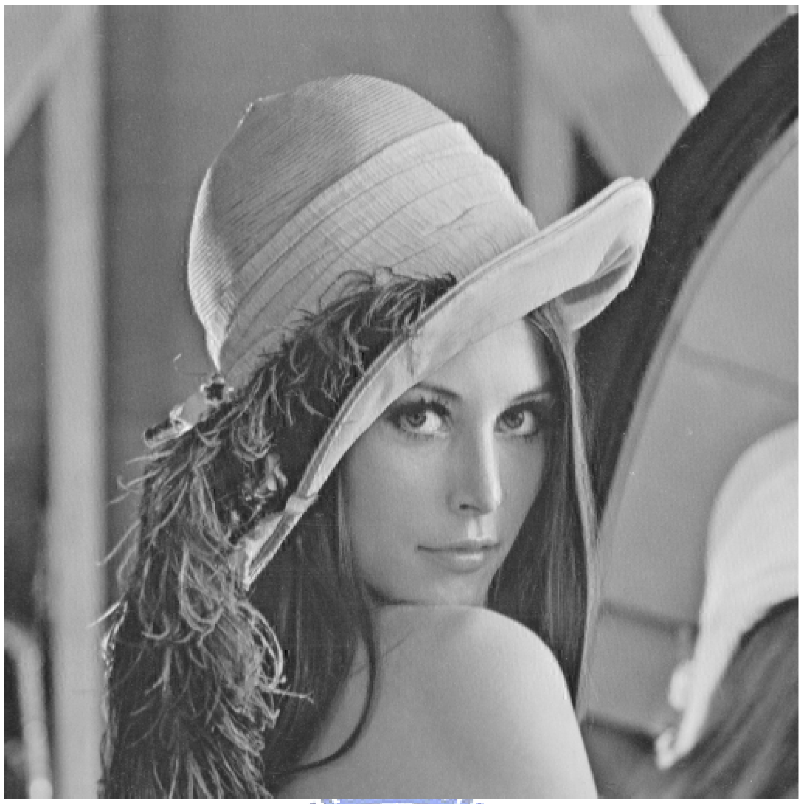

Fig. 3.8 The original image “Lena”.

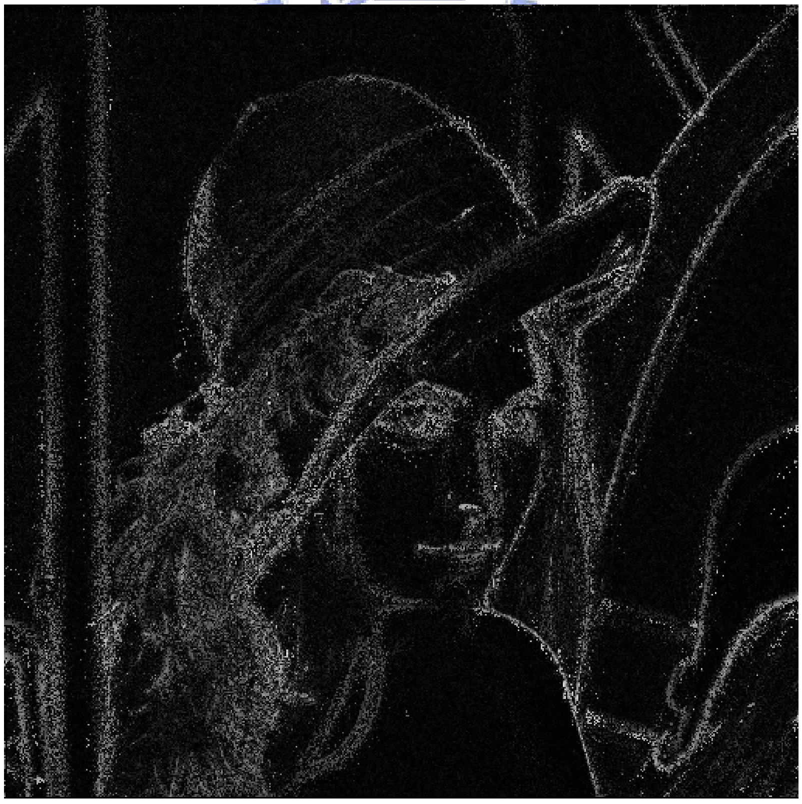

Fig. 3.9 The SOC-image (basic version) of Fig. 3.8.

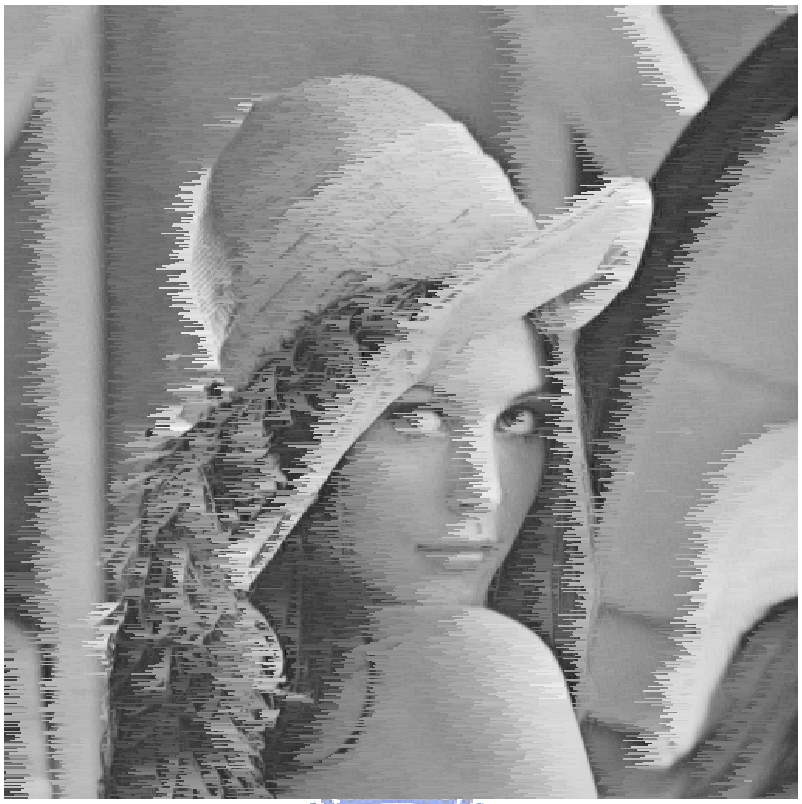
Fig. 3.10 The damaged image of Fig. 3.8. The PSNR is about 19.95 dB.

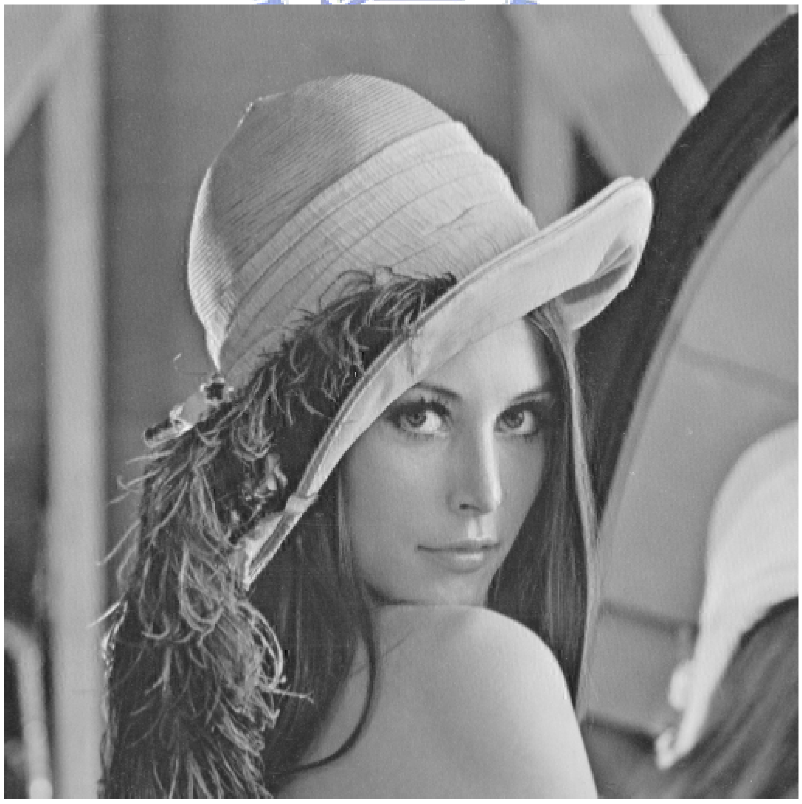
Fig. 3.11 The repaired image of Fig. 3.10 using Fig. 3.9. The PSNR is 53.95dB.

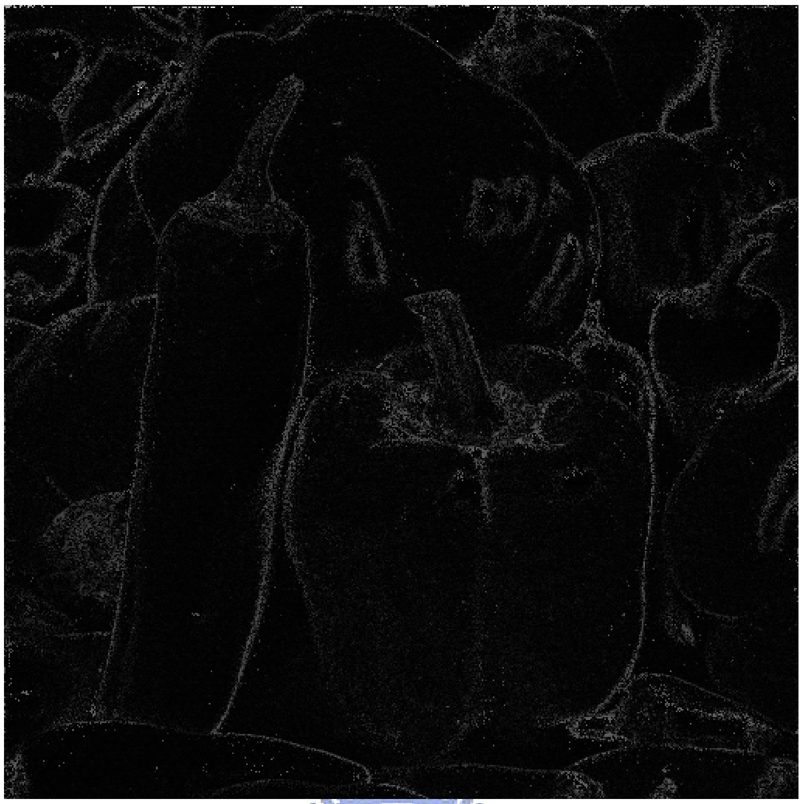

Fig. 3.12 The SOC-image of Fig. 3.4 (This image is constructed by the advanced version proposed in Section 3.3).

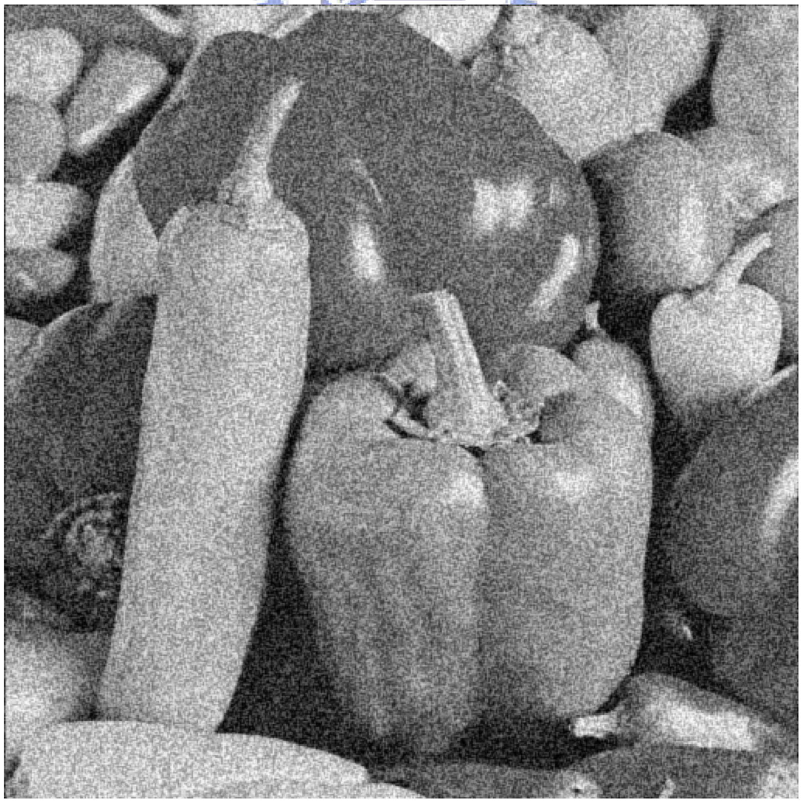

Fig. 3.13 The damaged image of Fig. 3.4. The PSNR is about 21.90 dB.

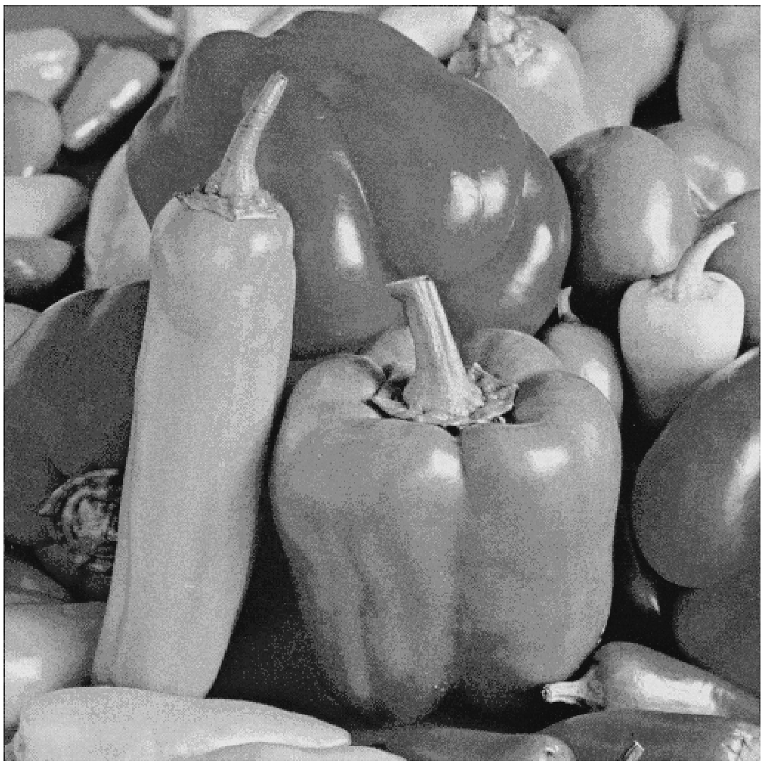

Fig. 3.14 The repaired image of Fig. 3.13 using Fig. 3.12. The PSNR is about 28.35 dB.

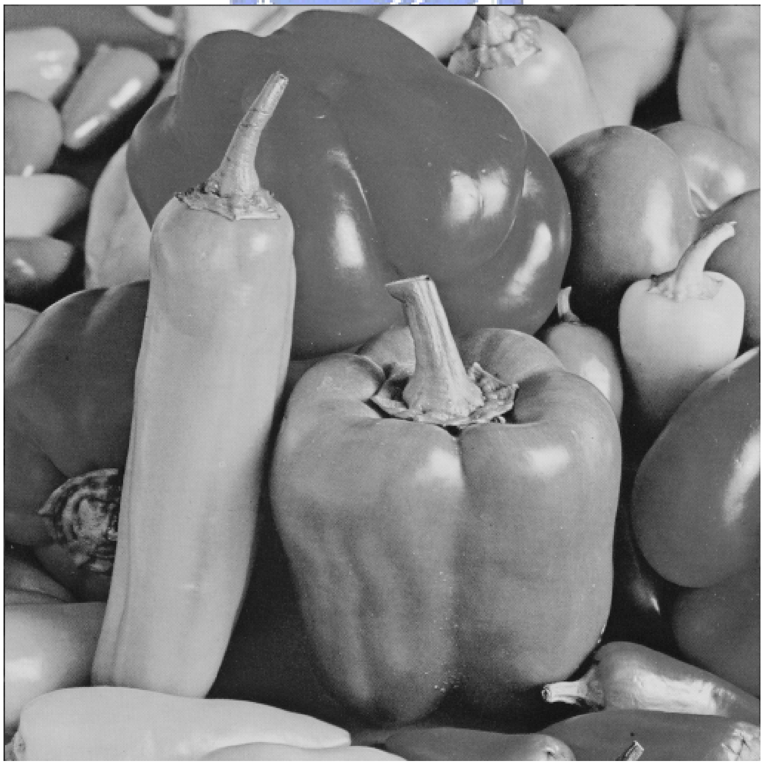

Fig. 3.15 The repaired image of Fig. 3.13 using Fig. 3.12 and the hash table. The PSNR is about 51.15 dB.