

Chapter 3

Data Hiding in Image Mosaics by Visible Boundary Regions and Its Copyright Protection Application Against Print-And-Scan Attacks

In this chapter, the proposed method for embedding data by visible boundary regions in an image mosaic is described. The idea of the method comes from image coding in cryptography [12]. Visible boundary regions are added to each tile image of the image mosaic and can be regarded as visible features. These features both can be extracted from the mosaic with the digital form and can be detected after the mosaic goes through a print-and-scan process. The print-and-scan process includes two jobs: printing a digital mosaic and scanning the mosaic of a paper format. These processes cause several types of attacks to the watermark, including image scaling, image rotation, and color distortion.

The remainder of this chapter is organized as follows. An introduction is given in Section 3.1. Section 3.2 describes the proposed data hiding method by the use of visible boundary regions of tile images. In Section 3.3, the application of the method to copyright protection against print-and-scan attacks is described. And finally, in Section 3.4, some discussions are made.

3.1 Introduction

Image mosaics are different from other digital images in their abundant contents and their ways of display. They may be shown in various media, such as displayed by a computer, or printed as posters or artistic productions. To prevent them from illegal personal or commercial uses, copyright protection of image mosaics becomes an important issue.

Recently, many data hiding techniques have been proposed and becomes one chief method to protect the copyrights of digital images. Invisible or visible watermarks can be embedded in and extracted from digital images with data hiding techniques to prove image ownerships. Although some of them are robust enough against many kinds of attacks, such as image rotation and scaling, most of them are weak when applied to deal with print-and-scan attacks.

3.1.1 Properties of Image Mosaics

Actually, image mosaics are frequently used as posters, covers of magazines, and billboards. It shows that they quite often do not exist in digital forms but in real copies. Therefore, for the purpose of copyright protection of image mosaics, the embedded watermark must strong to survive print-and-scan attacks.

The image mosaic has an important property in its impression on the human visual system. It combines colors in a small region such that an observer will only see an overall average color for that region at a distance. This important property gives us hints to deal with data hiding methods.

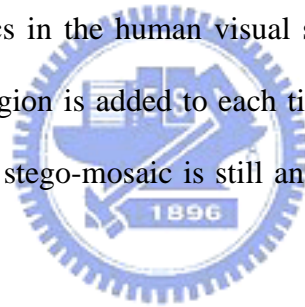
3.1.2 Problem Definition

In this study, the data hiding technique by adding visible boundary regions is proposed. The issues include firstly how to reduce the noticeable changes of the tile image while adding the visible boundary regions, and secondly how to detect the side

of an added boundary region correctly to extract embedded data. Because the data hiding method is applied to each tile images of an image mosaic, how to determine the tile size of a mosaic for tile image segmentation becomes another critical topic for study.

3.2 Proposed Data Hiding Technique by Visible Boundary Regions

In this section, the proposed method for hiding data in image mosaics and extracting data from image mosaics will be described. The method takes advantage of the property of image mosaics in the human visual system that we have mentioned before. A visible boundary region is added to each tile image of an image mosaic to generate a stego-mosaic. The stego-mosaic is still an image mosaic but each tile has changed slightly.



3.2.1 Properties of Visible Boundary Regions

A visible boundary region means observers can be aware of the existence of the region if he/she looks at the tile image very carefully. The size of a visible boundary region is eighth of the tile image and all pixels in the region have the same color. The regions obvious have a great effect on the impressions of image mosaics. However, if the colors of these visible boundary regions are filled to fit the corresponding target images and then become the part of the given tile images, the effect on image mosaics will be reduced.


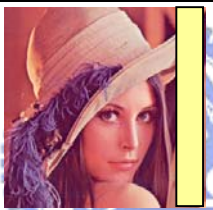
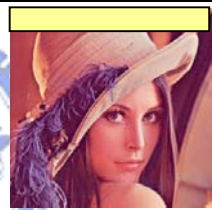
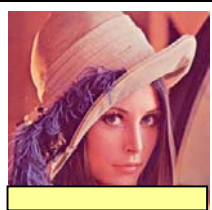
Besides, a property of visible boundary regions in statistics is that the variances of added regions are extremely small. The variance here represents the variation of the

RGB value of each pixel in the region. This property will by far be used in data extraction process.

3.2.2 Proposed Data Embedding Process

Without doubts, all rectangular digital images have four sides. In our embedding scheme, it is assumed that different sides represent different meanings. Figure 1 illustrates this simple rule. The left, right, upper, and lower sides in an image are regarded in this study to represent two bits “00”, ”01”, “10”, and “11,” respectively, as shown in Table 3.1.

Table 3.1 Types of boundary regions and their meanings.

Region Types				
Bits	00	01	10	11

In the data embedding process, an input data stream D with L characters is converted into a binary form in advance, which is denoted by $d_1d_2d_3d_4d_5\dots d_{8 \times L}$. The characters are grouped into many consecutive bit pairs, i.e. $\{\{d_1d_2\}, \{d_3d_4\}, \{d_5d_6\} \dots \{d_{8 \times L - 1}d_{8 \times L}\}\}$. The embedding process runs in the image mosaic creation stage when placing the selected tile images to the target image. Visible regions will be added to the tile images according to the current two bits and Table 3.1. The sizes of the added regions are all taken to be one eighth of the tile images size and the colors are the average color of that region of the tile image. Tile images are resized to guarantee the integrity of the image content. For the purpose of facilitating the hidden data extraction process, “variance check” and “noise generation” processes are applied to

three sides of boundary regions after the visible boundary region is added to the fourth side. The detail of the data embedding process can be expressed as an algorithm as follows, and Figure 2 illustrates the flowchart of the process.

Algorithm 3.1: Data embedding by visible boundary regions.

Input: an original image I , an image database D , an embedding stream S , a secret key K , and a variance threshold T .

Output: a stego-image mosaic M .

Steps.

Step1. Find tile image sequences from the given D that fit best to I by Step 1 through Step 3 of Algorithm 2.1.

Step2. Compute the hiding capacity C according to the size of I .

Step3. Generate a stego-stream S' in a binary form by the operation of K and S and repeat S' many times until the length of the stego-stream reach to the hiding capacity C .

Step4. Partition S' into many character groups S'_i with each group having only two bits.

Step5. Add visible boundary region into each tile image by the following steps:

- A. Resize the tile image according to S'_i .
- B. Add a visible region according to the S'_i .
- C. Compute the variances of the other three boundary regions except the added one; if the value is smaller than T , then gaussian noise is added (with zero mean and variance=30).
- D. Repeat Step 5.C until the variances of the other three boundary regions are all larger than T .
- E. Repeat Step 5 until the process has been done for each tile image in T .

Step6. Produce an image mosaic by Step 4 of Algorithm 2.2.

3.2.3 Proposed Data Extraction Process

The extraction process includes two parts. The first part is to detect the tile size of the image mosaic. The second part is to extract the embedded data.

A. Tile size detection

Because an image mosaic is made up of many tile images, so it contains several horizontal and vertical edges between two adjacent tiles obviously. We take advantage of this property in dealing with tile size detection.

An edge detection process is first used to find the horizontal and vertical edges of an image mosaic. Then statistical techniques are applied to estimate the distances between two adjacent edges. The tile size will be derived exactly with the two ways for verifications of the estimation results. Before describing the tile size detection algorithm, two definitions of terms are made as follows.

1. *Projection in Y-axis*: A projection in the Y-axis is the summation of all pixel values in one row of an image. The number of the projections in the Y-axis are equal to the number of columns in an image.
2. *Projection in X-axis*: A projection in the X-axis as well as the projection in the Y-axis is the summation of all pixel values in one column of an image. The total numbers of the projections in the X-axis are equal to the number of rows in an image.

The following algorithm shows the detail about tile size detection algorithm.

Algorithm 3.2: *Tile size detection.*

Input: an image mosaic M .

Output: a tile of the height H , and a width of the tile W .

Steps.

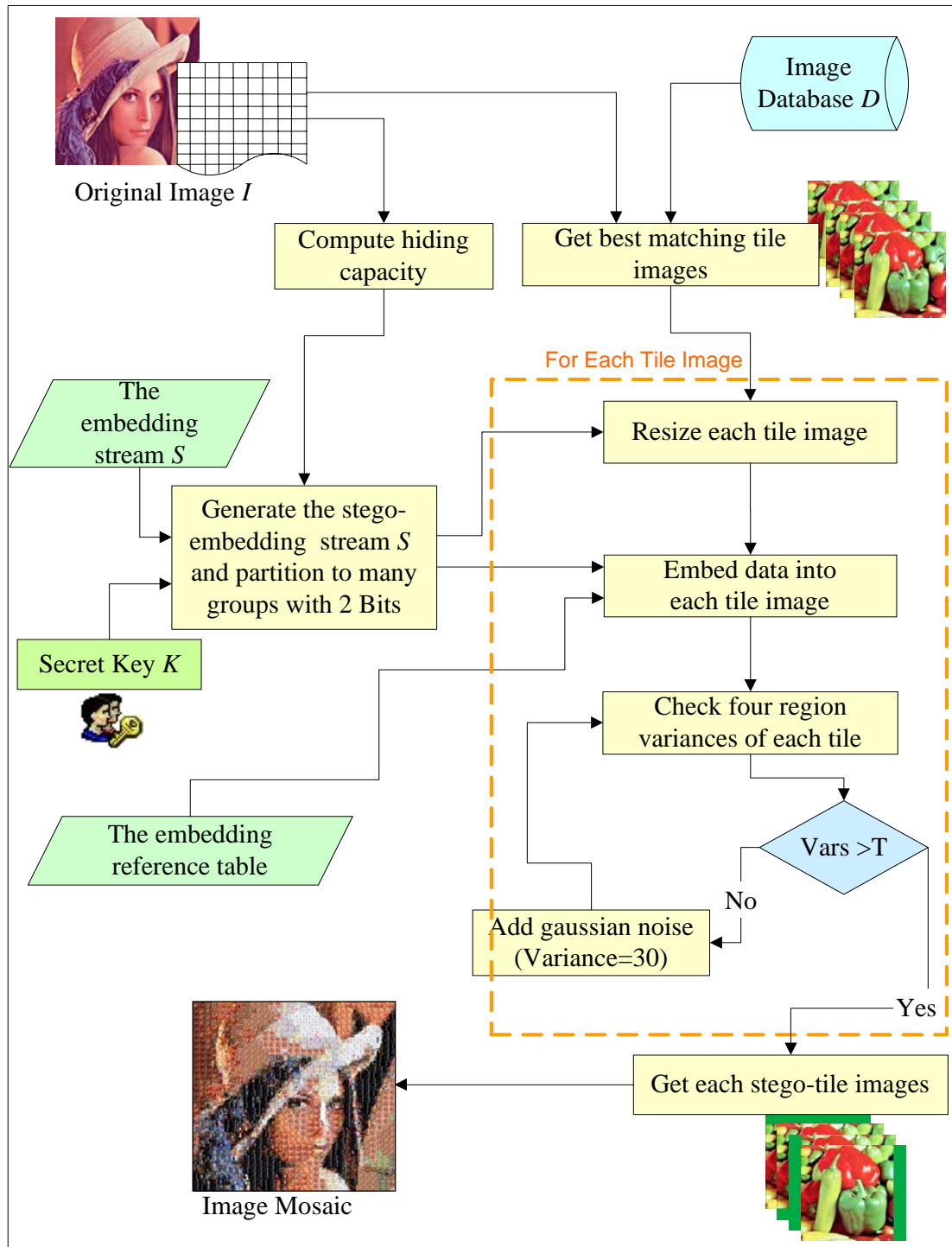


Figure 3.1 Flowchart of visible boundary embedding process.

Step1. Detect the edges of M by applying two 3×3 sobel mask as shown in Figure 3.2 and get a white and black Sobel image S .

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

Figure 3.2. 3x3 Sobel mask.

Step2. Derive two histograms X and Y that are the histograms of the projections in the X-axis and in the Y-axis respectively.

A. Let S_{ij} denote the gray value on location (i, j) of image S , where w is the width of S and h is the height of S , $S_{ij} \in \{0, 255\}$, and $i = 0, 1, 2, \dots, h - 1, j = 0, 1, \dots, w - 1$.

B. Let x_i denote the summation of the i th row and y_j denote the summation of the j th column. Define f_x and f_y to be two mapping functions from gray value indices to histogram values defined as follows:

$$f_x(i) = x_i = \sum_{j=0}^{w-1} S_{ij} \quad f_y(j) = y_j = \sum_{i=0}^{h-1} S_{ij}$$

C. Let X denote the histogram of x_i and Y denote the histogram of y_j :

$$X = \{ x_i \}, i = 0, 1, \dots, h - 1.$$

$$Y = \{ y_j \}, j = 0, 1, \dots, w - 1.$$

Step3. Get two sets PX and PY of the peaks in X and Y by applying a Laplacian mask as shown in Figure 3.3 and a thresholding technique with two predefined thresholds T_x, T_y .

A. Apply a one-dimension Laplacian mask.

-1	2	-1
----	---	----

Figure 3.3 Laplacian mask.

B. Calculate two thresholds T_x and T_y as follows:

$$T_x = c_1 \times \underset{i=0 \sim h-1}{\text{Max}}(x_i), T_y = c_2 \times \underset{i=0 \sim w-1}{\text{Max}}(y_i)$$

where c_1, c_2 are two pre-defined coefficients. Also, define *Max* to be the function that will return the maximum value of the input set.

- C. Find the sets of the peaks PX and PY according to the following rules:
- if $x_i \in X$ and $x_i > T_x$, then let $i \in \{PX\}$;*
 - if $y_j \in Y$ and $y_j > T_y$, then let $j \in \{PY\}$.*

Step4. Derive the center of each peak group PX^c and PY^c of the PX and PY with the operations of a clustering method and a pre-defined radius R in the following way.

- A. Let PX_i denotes the i th elements of the PX , and let PY_j denotes the j th elements of the PY .
- B. Define the center of each group of the PX and PY according the following rules:

$$\text{if } f_x(PX_i) = \underset{i=(i-R) \sim (i+R)}{\text{Max}}(f_x(PX_i)), \text{ then } PX_i \in \{PX^c\};$$

$$\text{if } f_y(PY_j) = \underset{j=(j-R) \sim (j+R)}{\text{Max}}(f_y(PY_j)), \text{ then } PY_j \in \{PY^c\}.$$

Step5. Derive the histogram Sx and Sy of the differences between two adjacent peaks in PX^c or PY^c by the following way.

- A. Let Sx_i denote the i th difference of the i th from the $(i+1)$ th peaks of PX^c , and let Sy_j denote the j th difference of the j th from the $(j+1)$ th peaks of PY^c . Compute Sx_i and Sy_i according to the following formula:

$$Sx_i = PX^c_{(i+1)} - PX^c_i \text{ where } i=0 \sim |PX^c| - 2;$$

$$Sy_i = PY^c_{(j+1)} - PY^c_j \text{ where } j=0 \sim |PY^c| - 2.$$

- B. Collect all the Sx_i and Sy_j to compose the histograms of Sx and Sy .

Step6. Get the temporary tile sizes W_t and H_t according the following rule:

$$W_t = k, \text{ if } Sx_k = \underset{i=0 \sim |Sx|-1}{\text{Max}}(Sx_i);$$

$$H_t = h, \text{ if } Sy_h = \underset{j=0 \sim |Sy|-2}{\text{Max}} (Sy_j).$$

Step7. Re-compute the W_t and H_t by using the projection histograms X and Y .

- A. Get the exact sets PX_t of the local maximums in X by comparing the histogram values around each histogram index which is a multiple of W_t .
- B. Apply Step 5 and Step 6 to the PX_t and reassign the W_t .
- C. Apply the same process A and B to Y to get H_t .

Step8. Correct the W_t and H_t by the division of the width and height of the image mosaic M . If the remainder is not zero, a number "1" will be added or subtracted until the remainder is zero.

Step9. Assign W_t and H_t to be the output values for W and H .

In fact, the described algorithm is not complicated. The main idea of the algorithm is to get the average distance between two adjacent edges. But the problems are both that the tile number of a row or a column in the mosaic is unknown and that it is also difficult to find local maximums in the histogram without the numbers of tiles. As a result, this method is a kind of non-supervised learning which tries to find out the tile number and size by analyzing the statistics of the peaks. Step 4 is applied because if the peaks set, which is derived in Step 3, are used for the subtraction process in Step 5, the results will probably be outliers and noise. Besides, in order to get a correct tile size, the temporary results are verified twice both in Step 7 and in Step 8. A Sobel image obtained in Step 1 is shown in Figure 3.4. Figures 3.5 illustrates the projections in the X-axis and in the Y-axis of the Sobel image, respectively. A flowchart of the process is shown in Figure 3.6.

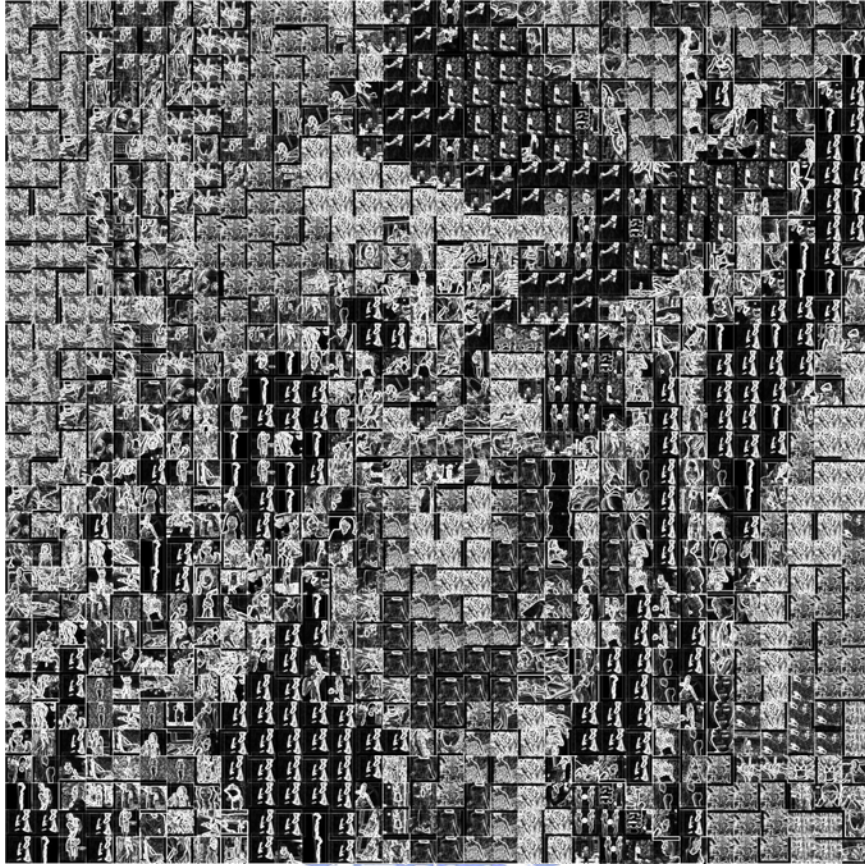


Figure 3.4 The Sobel image of an image mosaic.

B. Data Extraction Process

The data extraction process will be applied after the tile size is detected in Part A. The essence of the data extraction process is based on analyzing the variances of four boundary regions in a tile image. The boundary region with the smallest variance is determined to be the added region. According to Table 3.1, we can extract the embedded data.

Algorithm 3.3: Data Extraction by Visible Boundary Regions.

Input: an Image Mosaic M , a height of tile image H , a width of tile image W , and a secret key K .

Output: the extracted data E .

Step.

Step1. Divide M into tile images sequentially according to the input H and W .

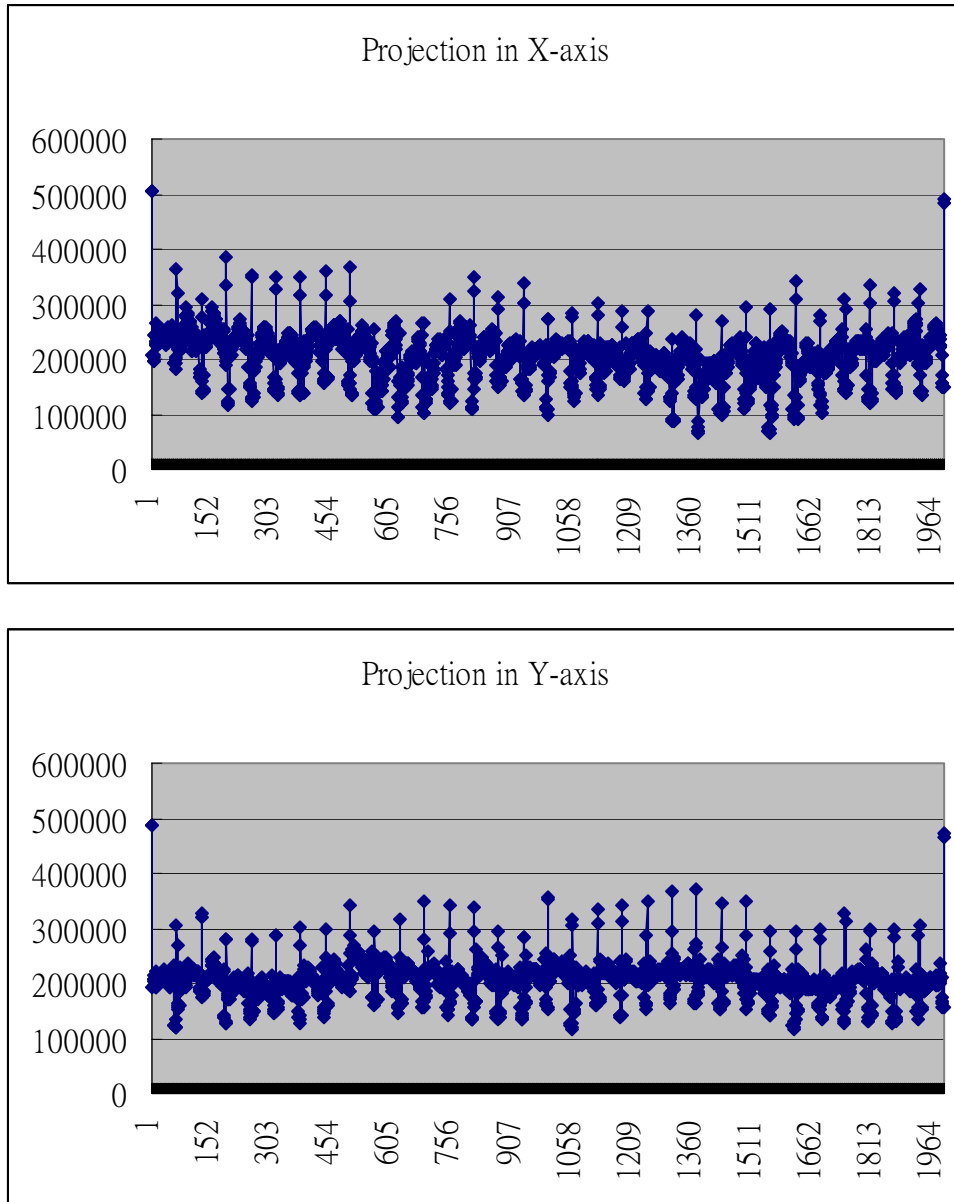


Figure 3.5 Histogram of (a) Projection of X-axis, (b) Projection of Y-axis.

- Step2. Calculate the variances of the four boundary regions of a tile image.
- Step3. Compare the four variances to get the side with the minimum variance.
- Step4. Derive and save the two bits of one character group according to Table 3.1.
- Step5. Repeat Step 2 through Step 4 until completing the process for all tile images and return the record E_1 .
- Step6. Get the extracted data by the procedure of E_1 and K .

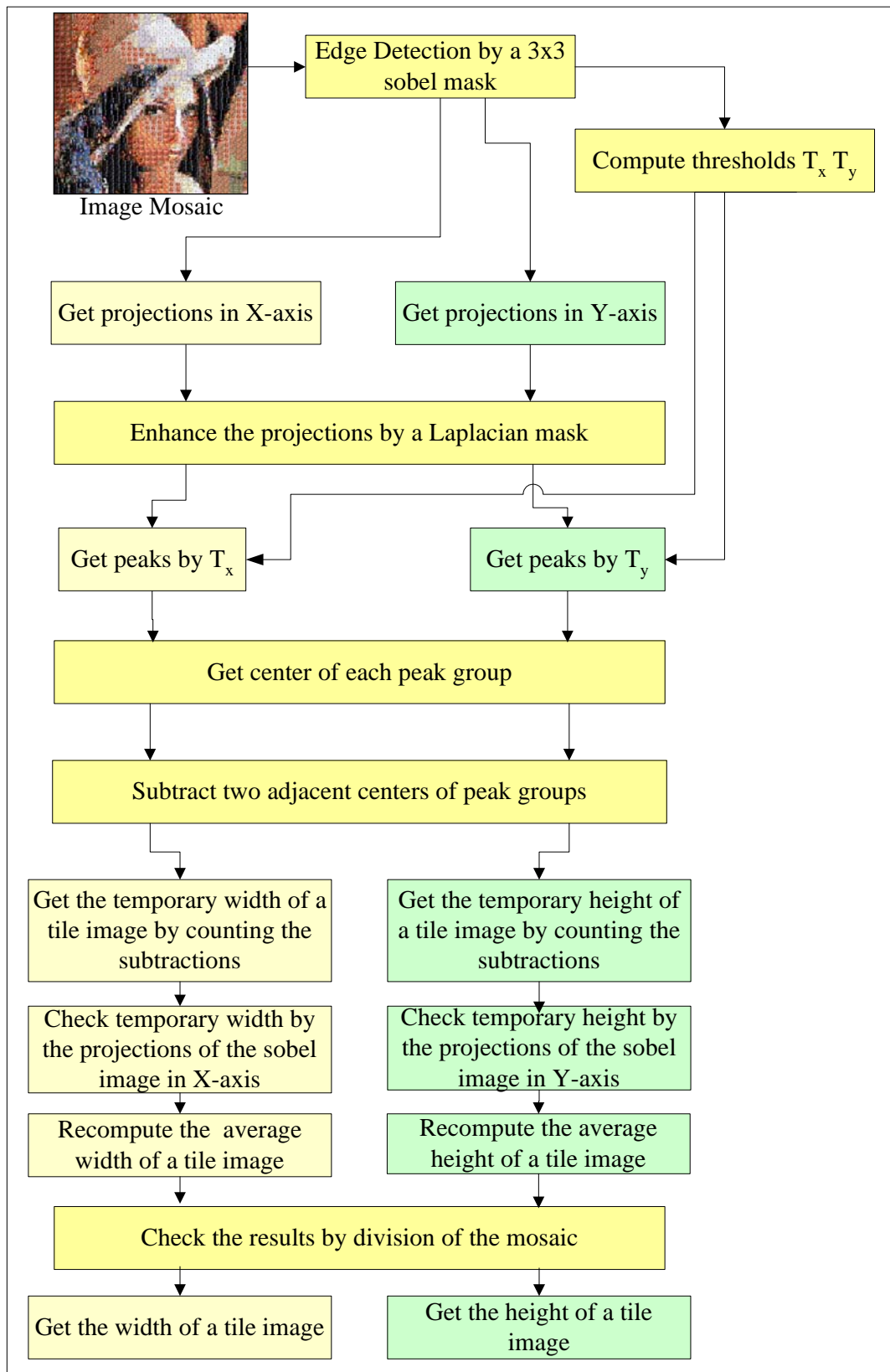


Figure 3.6 Flowchart of tile size detection process.

3.3 Copyright Protection against Print-and-Scan Attack

In this section, an application for copyright protection against print-and-scan attacks is described. Here we are focusing on dealing with the image after the print-and-scan process instead of the original digital form. The use of a semi-visible watermark is also proposed to protect the copyright of the image mosaic.

3.3.1 Description of Print-and-Scan Attack

Printing and scanning are two common processes for using digital documents in our daily life. In this study, the printing process means to publish a digital image in print and the scanning process means to digitize the image in paper form. Both printing and scanning processes are regarded as attacks to the digital or non-digital image according to the following descriptions. As far as the printing process is concerned, the problem of color distortion will happen no matter how great a printer is. The color distortion comes from the translation of the virtual RGB values into real colors. Besides, the image size will also be scaled due to the printer setups or paper sizes. In the scanning process, an image is digitized into the form of digital data. Attacks at least include color distortions and changes of image sizes. The reasons seem the same as those for the printing process. Furthermore, the problem of paper tilting in scanning is also concerned about because the image derived after the scanning process may not be straight. The tilt problem can be considered as a combination of several attacks because it may result in both color distortion and image scaling.

In this study, we only try to reorient a tilted image rather than to correct the other

two attacks, namely, color distortion and image scaling. We control the experiments of the print-and-scan process by some parameters, such as the number of points per inch for printing and the size of print paper. The related details about the setups of the printer and the scanner used in this study are described in Section 3.3.5.

3.3.2 Definition of Semi-visible Watermark

Many data hiding techniques are in wide use for copyright protection of digital images by embedding invisible watermarks. Invisible watermarks are definitely little sensitive to observers and can be extracted through appropriate algorithms. Although the invisible watermark is efficient in the protection of image copyright, it cannot stop the uses of digital images in private by illegal users. Some researchers presented methods to embed the removable visible watermarks instead of invisible watermarks to prevent digital images from illegal access directly. Only legal users can remove the embedded visible watermark and recover the original image. However, it is possible that observers, due to the embedded visible watermark, do not like to see the cover image. This result is a limit on the usage of embedding removable visible watermarks in practice.

In this study, the use of an irremovable semi-visible watermark is proposed. The irremovable semi-visible watermark is by far a watermark that can be seen by observers for it is also visible. Particularly, it can be insensitive to the human beings as well just like an invisible watermark because the embedded data are similar to the background of images. The functions of the semi-visible watermark are not only to achieve copyright protection but also to prevent them from illegal access. For the copyright protection purpose, the semi-visible watermark can be extracted and verified to prove the ownership of the image. And the influence on image contents that disturbs observers is also reduced by the property of the semi-visible watermark.

3.3.3 Proposed Semi-visible Watermark Embedding Process

The method of semi-visible watermark embedding is similar to that described in Section 3.2.1. The main difference between Section 3.2.1 and this section is that in this section the semi-visible watermark embedding process is emphasized.

Algorithm 3.4: *Semi-visible Watermark Embedding by use of Visible Boundary Regions.*

Input: an original image I , an image database D , a watermark W for embedding, a secret key K , and a variance threshold T .

Output: a stego-image mosaic M .

Steps.

Step1. Read W to get the embedded stream S .

Step2. Apply Step 2 of Algorithm 3.1 using the given S and K to get a stego-stream S_1 .

Step3. Apply Step 3 through Step 5 of Algorithm 3.1 by taking the given S_1 as an input to get an output M .

3.3.4 Proposed Semi-visible Watermark Extraction Process

The proposed semi-visible watermark extraction method against print-and-scan attacks includes four parts. Above all, the tilt adjustment algorithm is first used to detect the angle of a tilted image and re-orient it to solve the tilt problem after the scanning process. Then the tile size detection and data extraction methods that are described in Algorithm 3.2 and Algorithm 3.3 are applied. Finally, the watermark is

recovered by a voting scheme.

A. Tilt adjustment

The proposed tilt adjustment algorithm is based on edge detection and projection methods. Before scanning a printed mosaic picture with a table scanner, the picture is placed on a flat surface and a window is selected to specify the scanning scope. Here we assume that the picture is rectangular in shape and is placed carefully enough with its boundaries at a very small angle with respect to the corresponding scanning window boundaries. We also assume that the scanning scope is larger than the size of the picture to be scanned. In the scanning result which is an image, those non-image regions, each called a border of the image, are shown with a monotone color (i.e., black or white). In the proposed tilt adjustment algorithm, the image is rotated with a small angle many times to find out the tilt angle of the image by the projection method. A tilt angle is obtained after the edge detection by comparing the maximum value of the projections of all the rotated images. Finally the image is re-oriented with the detected tilt angle. Figure 3.7 shows the concept of tilt detection and the tilt adjustment algorithm is expressed as follows.

Algorithm 3.5: Tilt adjustment.

Input: an image mosaic M .

Output: a straight image mosaic S .

Step.

Step1. Apply Step1 and Step2 of Algorithm 3.2 to the given image M to get the projection histogram X .

Step2. Save the maximum value of X by taking a predefined range.

Step3. Obtain a new image M' by rotating M with a predefined small angle.

Step4. Repeat Step1 through Step 3 by taking the new M' as an input until

reaching a predefined number of times.

Step5. Compare the values saved in Step2 after finishing the looping in Step4 to get the overall maximum value P .

Step6. Decide the rotating angle A by mapping P to the rotated angle.

Step7. Rotate M with angle A to get straight image mosaic S .

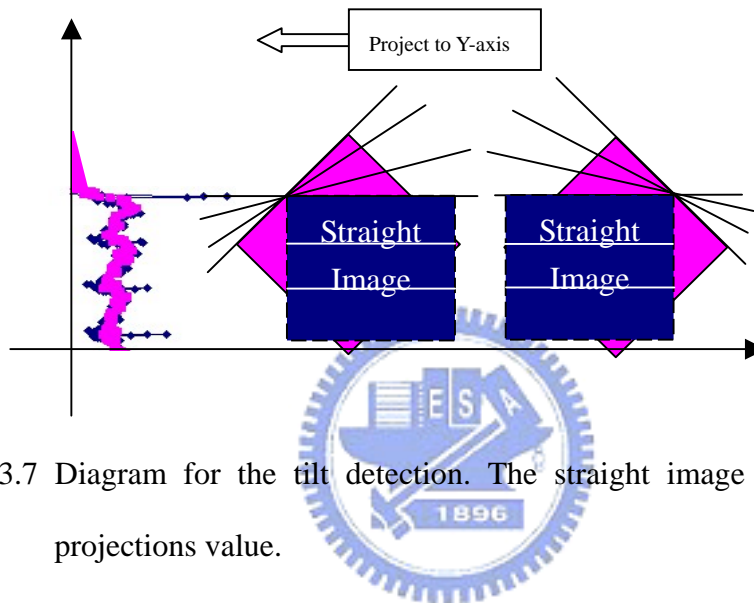


Figure 3.7 Diagram for the tilt detection. The straight image has the maximum projections value.

B. Border detection

The border detection algorithm is proposed by the use of the edge detection and the projection methods. A border need be detected because the projection value of the border area is smaller than those of the areas without borders. Figure 3.8 shows the concept of the border detection method. The border will be removed after it has been detected.

Algorithm 3.6: Border detection and removal.

Input: an image mosaic M .

Output: an image mosaic M' without border.

Steps.

Step1. Apply Step1 of Algorithm 3.2 to get a Sobel image S .

Step2. Compute the projection values of S in the X-axis and in the Y-axis with a predefined number of intervals.

Step3. Compare the projection values to each other to get the maximum value.

Step4. Define the border area by the four corner points of S .

Step5. Derive M' by removing the border area.

C. Tile size detection

The tile size detection method is just the same as Section 3.2.3.A. The details are omitted here.



Figure 3.8 Diagram for border detection. (1) The picture of an image with a border. (2)

A part of the Sobel image. (3) Partial projection histogram in the Y-axis of the Sobel image.

D. Data extraction process and watermark recovery

The data extraction process is almost the same as Section 3.2.3 B. However, here a process for watermark recovery by a voting scheme is applied additionally.

The voting scheme will be used when the hiding capacity is larger than the embedded data by three times. It improves the ability for the tolerance of extraction errors. Moreover, the data extraction process can still work well regardless of the voting scheme. Figure 3.9 illustrates the flowchart of the process.

Algorithm 3.6: Data extraction process and watermark recovery.

Input: an image mosaic M , and a secret key K .

Output: a recover watermark W .

Steps.

Step1. Reorient the image mosaic M to be upright by applying Algorithm 3.5.

Step2. Remove the border of an image mosaic by the principle of the projection method in Algorithm 3.5.

Step3. Get the height and width of tile images by Algorithm 3.2.

Step4. Divide M into tile images using the PX and PY obtained from Algorithm 3.2.

Step5. Get the stego--stream S' by applying Step 2 through Step 5 of Algorithm 3.3.

Step6. Get the embedded steam S by the randomization operation using the given secret key K and S' .

Step7. If the length of S is larger than the predefined size three times, then go to Step 8.

Step8. Get the originally embedded stream by a voting scheme.

Step9. Recover the watermark from the results in Step 8.

3.3.5 Experimental Results

Some experimental results of applying the above-mentioned methods are shown in this section. The related setups of the experiments are shown in Table 3.2 and the results are shown in Figure 3.10. We also conducted experiments on the tilt factor in the watermark extraction process, as shown in Figure 3.11 and Table 3.3. More experimental results of watermark extraction through print-and-scan are shown in Figure 3.12 and Table 3.4.

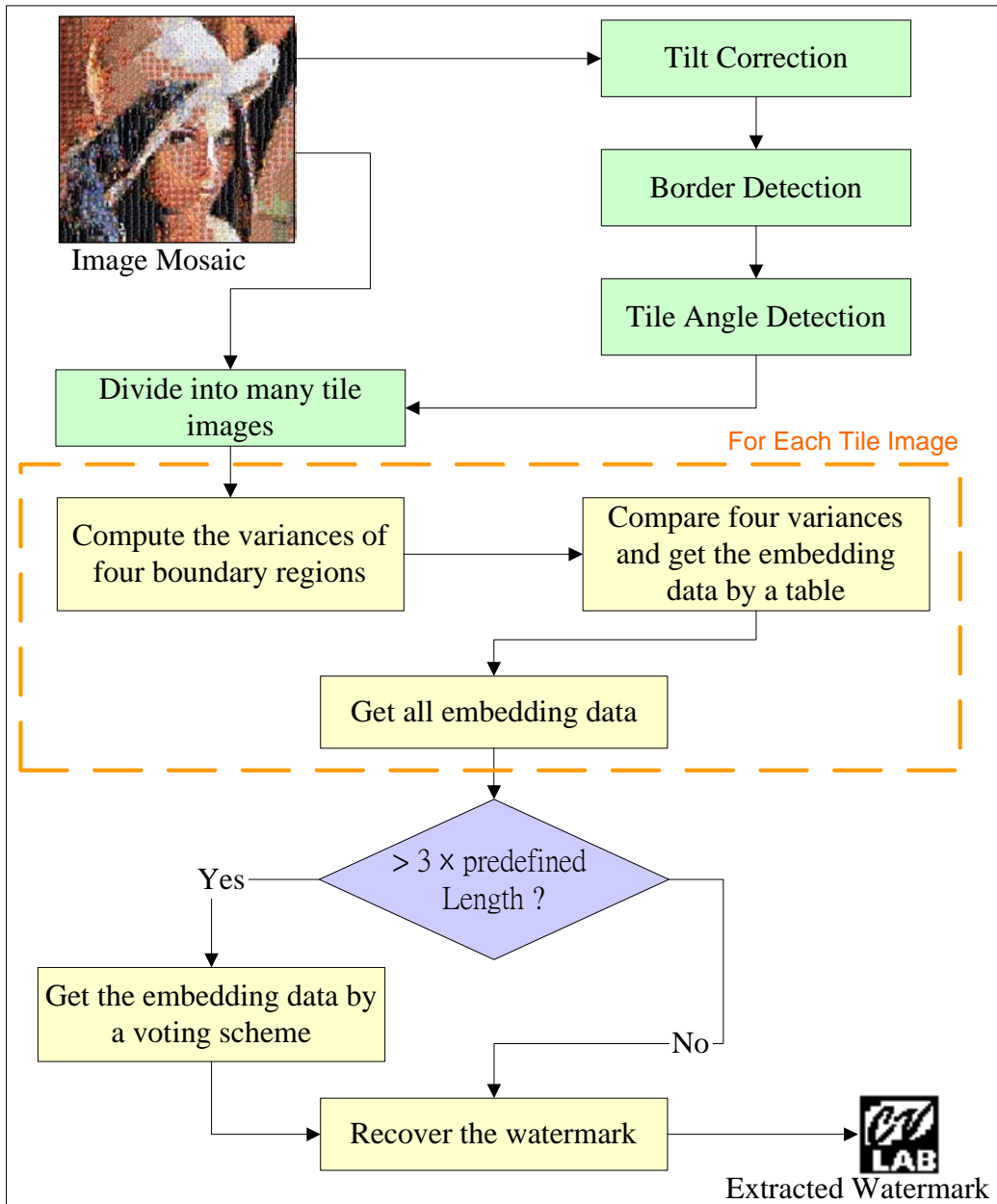
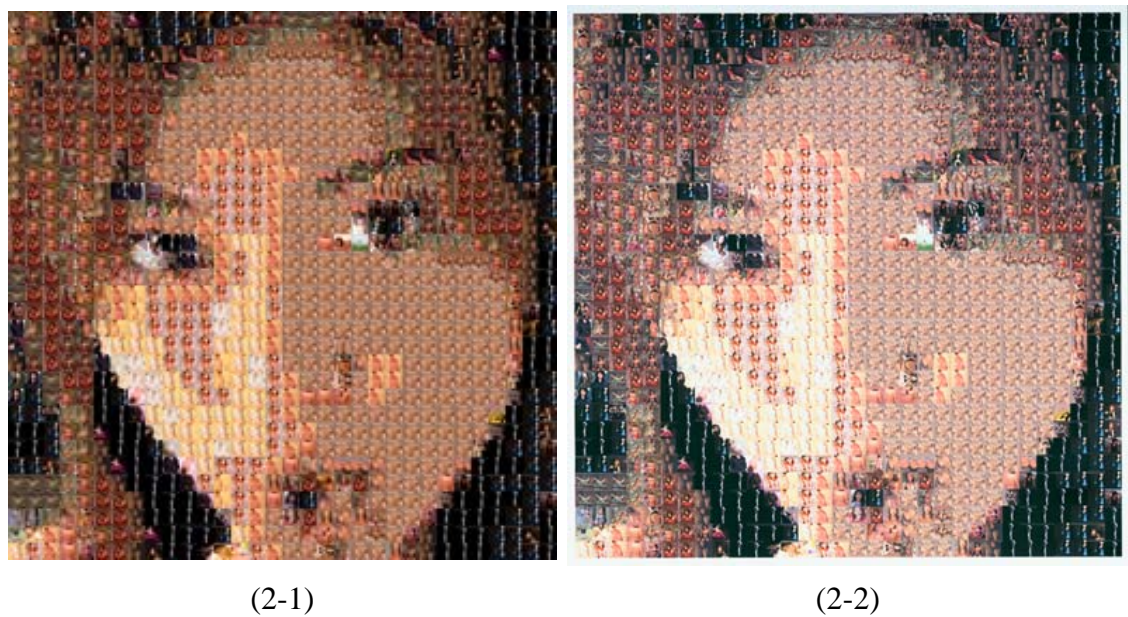
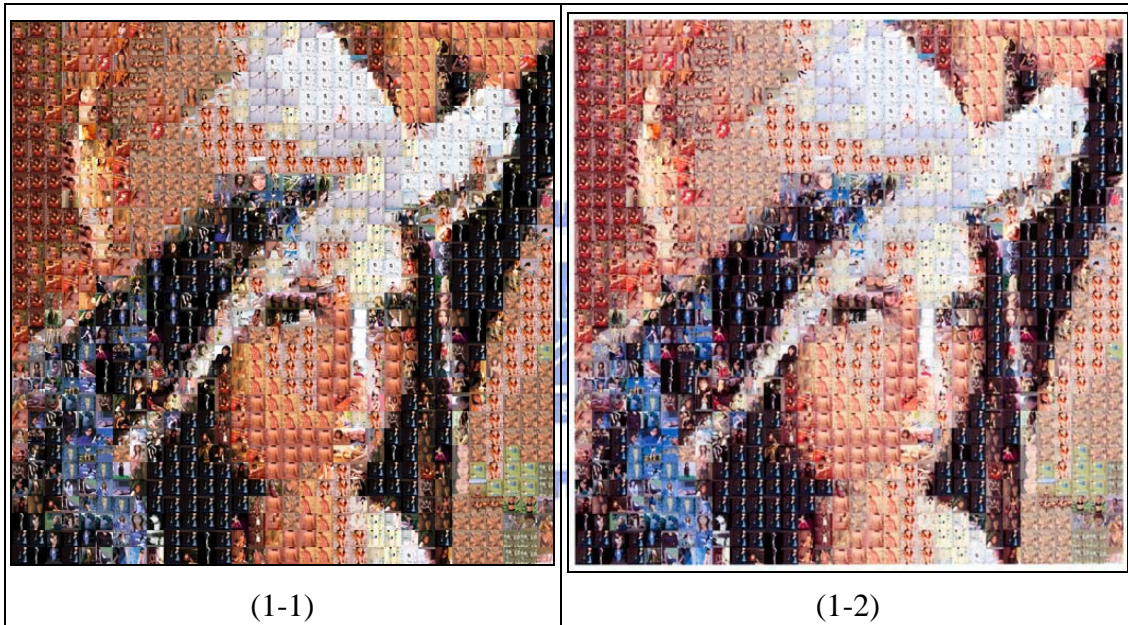


Figure 3.9 Flowchart of the data extraction process and watermark recovery.

Table 3.2 Related setup of the experiment.

Watermark	32×32 pixels, black and white image
Image Mosaic	1024×1024 pixels, 24bits compressed color image
Tile Image	32×32 pixels, 24bits color image
Printer Setup	Color laser printer with A4 size and r-correction $r=1.2$
Scanner Setup	250dpi with 100% aspect ratio
Scanned Image	Uncompressed image





(3-1)



(3-2)



(4-1)



(4-2)



(5-1)



(5-2)

Figure 3.10 Watermark extraction process against the print-and-scan attack. It shows the color distortions of scanned image mosaics are quite obvious. The image mosaics in left side are the original mosaics and in right side are the image mosaics obtained by the print-and-scan process.

Table 3.3 Watermarks and the error rates extracted from the image mosaics shown in Figure 3.10. (Error rate = numbers of the correct pixels / total number of the pixels)



Original watermark

	Watermark	Error rate		Watermark	Error rate
(1-1)		0.995	(1-2)		0.955
(2-1)		0.995	(2-2)		0.846
(3-1)		0.994	(2-2)		0.872
(4-1)		0.999	(4-2)		0.932
(5-1)		0.997	(5-2)		0.815

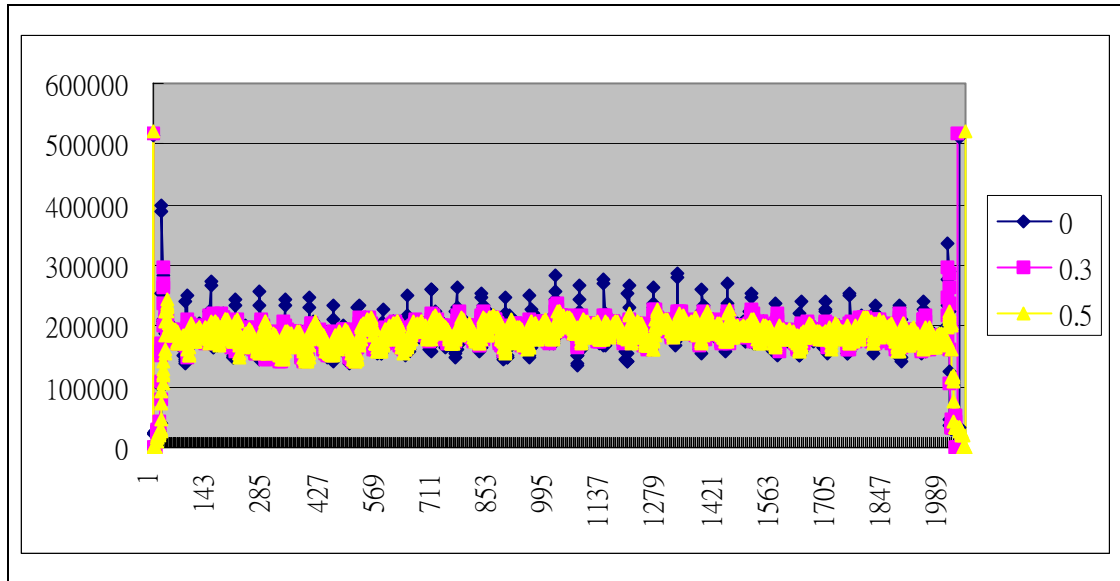





Figure 3.11 The distribution of the projection of Figure 3.10 (1-2) in the Y-axis used for the tilt detection. It shows that the distribution of the projections trend to be flat. As a result, the tilt image is harder for applying the peak finding algorithm.

Table 3.4 Watermark extracted from different tilt mosaics.

Angle	+0°	+0.1°	+0.2°	+0.3°	+0.4°	+0.5°
Extracted Watermark				Cannot detect the	Cannot detect the	Cannot detect the
Error rate	0.955	0.937	0.905	tile angle	tile angle	tile angle

3.4 Discussions and Summary

In this study, we proposed a data hiding method against print-and-scan attacks for copyright protection by embedding semi-visible watermarks. That is, the embedded watermark can survive certain attacks, such as image scaling, image rotation, and color distortion. Moreover, three methods have been proposed for the purposes of facilitating the watermark extraction process. The experimental results

show that the methods are practical and the semi-visible watermarks can be extracted correctly. It seems the error rates of the watermark extraction shown in Table 3.8 are image-dependent. Many factors influence the results of watermark extraction, such as the scanner setups and the quality of printers. Besides, the number of different colors in the tile image may be an important factor to the watermark extraction process. We may face the problems that if there is almost only one color in the tile image, the watermark extraction strategy will possibly fail in the determination of the tile image side with the smallest variance. It is the reason why the error rate of the watermark extraction through the print-and-scan process became worse than that for the original image mosaic.



Chapter 4

Data Hiding in Image Mosaics by Histogram Modification and Its Application in Image Authentication

In this chapter, a novel method for embedding data in an image mosaic is described. Recalling the techniques of mosaic modification described in Chapter 2, we have figured out a way to take advantage of the modifications for data hiding. The idea of the proposed method is based on histogram data hiding [11]. That is, data are embedded by altering the pixel values of each tile image in accordance with the color histogram of the target image. The data hiding method is by far employed in the spatial domain because it deals with the RGB values of the pixels. As a result, images with the BMP format are used in this study. We also propose an application based on this data hiding method for authentication of image mosaics.

The remainder of this chapter is organized as follows. An introduction is given in Section 4, including a review of related works and the problem definition. Section 4.2 describes the proposed data hiding method and its use for image authentication. In Section 4.3, some experimental results are shown and the chapter is summarized with some discussions.

4.1 Introduction

An image mosaic is composed of many tile images. However the color variety of a certain tile image sometimes might cause a prominent bad impression of the mosaic.

We have proposed a method for adaptively modifying the colors within the mosaic in Section 2.5 to reduce the distortion effect of this type. In this section, the method is improved for the data hiding purpose with a new technique.

4.1.1 Review of Data Hiding in Histograms

Ni, et al. [11] presented a novel reversible data hiding algorithm in the spatial domain. Data are embedded in the maximum point of the histogram by modifying the pixel values slightly. The method shifts the histogram in a way to empty out the bin of the maximum point of the histogram. If the gray value of a pixel belongs to the bin of the maximum point of the histogram, it is considered as an embeddable pixel. Then a bit “1” or “0” is embedded by changing the gray values of these embeddable pixels. In the extraction process, a *signature* is required to provide the information of where the bin of the maximum point is located in the histogram. The data can be extracted by comparing the gray value with that of the maximum point. And the original image is recovered by an inverse shifting of the histogram. The hiding capacity of the method is highly dependent on the distribution of the colors in the image.

4.1.2 Problem Definition

Based on the technique proposed in Section 2.5, the data hiding technique may be applied during the modification of a pixel’s color. As a result, the issues include both how to embed a bit during modification and what pixel is to be selected for modification. No matter what issues we are concerned about, the goals of hiding data in the mosaic include both improving the quality of the mosaic and increasing the hiding capacity. The above survey of related works inspired us to figure out a method

for choosing embeddable pixels. The proposed method is described in the following section.

4.2 Proposed Image Authentication Technique for Image Mosaics by Histogram Modification in Hue Channel

Recalling the previously-mentioned techniques of color modification, we see that the HSI color model is an ideal tool for the modification because it is natural and intuitive to humans. In this section, a method proposed to take advantage of the hue property of tile and target images is described. We also use the proposed data hiding method to achieve the purpose of image authentication.



4.2.1 Properties of Hue Channel in Image Mosaics

In the HSI color model, intensity is the key factor in describing color sensation and is sensitive to the human visual system. Hue describes a pure color, whereas saturation measures the degree of how a pure color is diluted by white light. Besides, the hue component is circular, which means that the hue value for 0 is identical to that for 360.

The distribution of the hue component in a small region of a mosaic is usually centralized near a specific value. However, the distribution of the hue components of the tile images is normally distributed because a tile image is usually rich in details and colors. Because the pixels with their color distribution far away from that of the

target image are considered disharmonious on the impressions, the idea of the proposed method is to adjust the color distribution of the tile image to approximate that of the target image. That is, the hue component is used in this study for the purpose of the modification of pixel values to hide data in the mosaic.

4.2.2 Authentication Signal Generation and Embedding Process

In this section, we describe the proposed data hiding method by use of the histogram of the hue component. Then the proposed application of it to image authentication is described, including the authentication signal generation and embedding processes.

A. Proposed Data Hiding Technique by Histogram Modification

The concept of hiding data in the hue component is based on the description of Section 2.6. The adaptive modification method is modified slightly to fit the purpose of data hiding here. The algorithm of the proposed data hiding technique is described briefly as follows.

We use the “Lena” image and a partial region of an original image as shown in Figure 4.2 as an example to illustrate our algorithm. In the example we assume that the “Lena” image (128×128) shown in Figure 4.2(1) is the best matching tile image with respect to a partial region of an original image as the target image (128×128) shown in Figure 4.2(3). Figure 4.2 also shows the histograms of the hue components of the tile and the target images. Figure 4.3 illustrates how to embed data by changing the histogram of the hue component. In this study, the hue component is quantized by

units of 30 to become 12 bins.

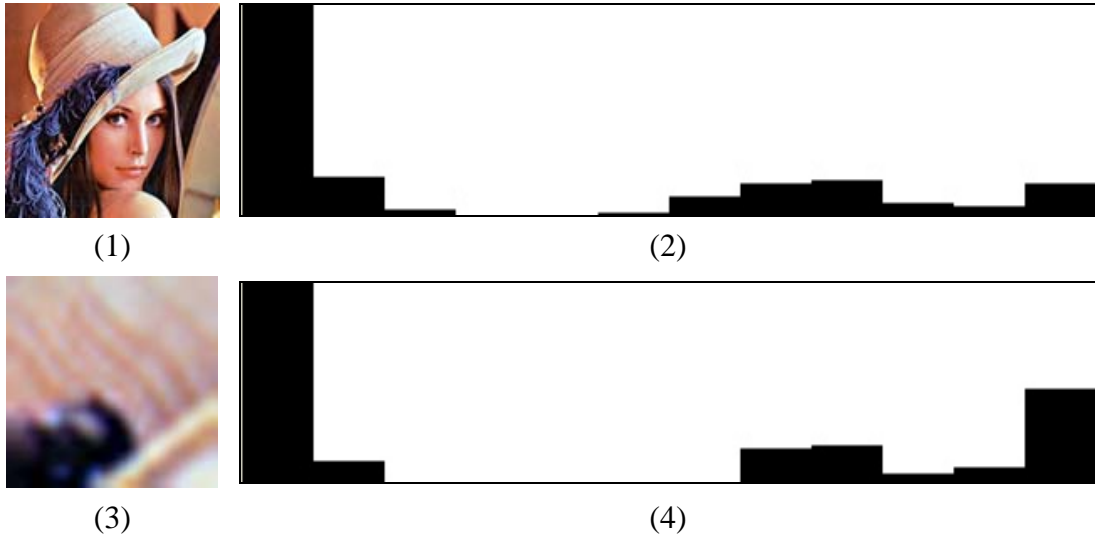


Figure 4.1 The tile and target image and their histogram of the hue components with 12 bins, respectively. (1) The tile image. (2) The histogram of the hue component of (1). (3) The target image. (4) The histogram of the hue component of (2).

Algorithm 4.1: Data Hiding by Histogram Modification in Hue Channel.

Input: a target image *Target*, a tile image *Tile*, and data bit *S* to be embedded.

Output: a modified tile image *Tile'*.

Step.

Step 1. Convert the colors of *Target* and *Tile* from the RGB model to the HSI one.

Step 2. Calculate the histograms, H^{tar} and H^{tile} , of the hue components of *Target* and *Tile* and quantized them into 12 bins.

Step 3. Find the peak bin of H^{tar} , denoted as H_{max}^{tar} .

Step 4. Derive embeddable pixels by scanning the entire image according to the following rules.

A. Let $H(P_i)$ denote the hue value of a pixel P_i .

B. Decide embeddable pixels according to the following rule:

Step 5. Quantize H_{\max}^{tar} into 3 bins, denoted as $H_{\max 1}^{tar}$, $H_{\max 2}^{tar}$, and $H_{\max 3}^{tar}$.

Reassign the hue value of the pixels that belong to H_{\max}^{tar} into $H_{\max 2}^{tar}$

and empty out $H_{\max 1}^{tar}$ and $H_{\max 3}^{tar}$ as shown in Figure 4.2.



Figure 4.2 The max bin is quantized into three parts.

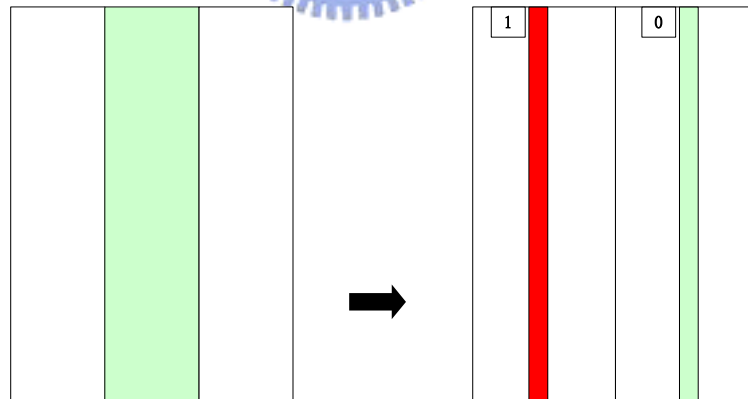


Figure 4.3 The max bin is used to embed data. The hue components of the pixels in the bin are reassigned according to the data for embedding. A pixel belonging to the red region represent bit “1” and a pixel belonging to the green region represent bit “0”.

Step 6. Embed data by scanning the entire image *Tile* according to the following

rules.

A. Decide the embeddable pixels by the rule:

$$\begin{cases} \text{if } P_i \in \text{Tile and } P_i \in \{\text{embeddable pixels}\}, \text{ then go to B;} \\ \text{if } P_i \in \text{Tile and } P_i \notin \{\text{embeddable pixels}\}, \text{ then end Step 6.} \end{cases}$$

B. Perform the following embedding rules:

$$\begin{cases} \text{if } S \text{ is odd, then let } H(P_i) \in \{H_{\max 3}^{tar}\} \text{ and set } H(P_i) = H_{\max}^{tar} \times 30 + 22.5; \\ \text{if } S \text{ is even, then let } H(P_i) \in \{H_{\max 1}^{tar}\} \text{ and set } H(P_i) = H_{\max}^{tar} \times 30 + 7.5. \end{cases}$$

Step 7. Convert the colors from the HSI model into the RGB one.

B. Authentication Signals Generation and Embedding Process

The authentication signals are generated by a randomization procedure using a secret key as the input seed. Then the 8×8 blocks of the tile image are processed to carry out the above-mentioned histogram modification in the hue channel. The authentication signals are embedded pixel by pixel, block by block, and tile by tile. Figure 4.4 shows the flowchart of the embedding process and the embedding algorithm is described as follows.

Algorithm 4.2: Authentication Signals Embedding Process.

Input: a sequence of the best matching images I , an authentication signal S , and a threshold of hiding capacity T .

Output: an image mosaic M .

Steps.

Step 1. Duplicate S with a pre-defined number n of times to be S' .

Step 2. Let I_i denote one image of the sequence I and divide I_i into 8×8 blocks.

Step 3. Derive the H^{tile} , H^{tar} , H_{\max}^{tar} according to Steps 1, 2, 3 of Algorithm 4.1.

Step 4. Derive the embeddable pixels of each 8×8 block according to Step 4 of

Algorithm 4.1 and increase the number of embeddable pixels by the given T according to the following rules:

A. Count the number N of embeddable pixels as follows:

$$\begin{cases} \text{if } P_i \in \text{Tile and } H(P_i) \in \{H_{\max}^{\text{tar}}\}, \text{ then } P_i \in \text{embeddable pixels;} \\ \text{if } P_i \in \text{Tile and } H(P_i) \notin \{H_{\max}^{\text{tar}}\}, \text{ then } P_i \notin \text{embeddable pixels.} \end{cases}$$

B. Enlarge the range of H_{\max}^{tar} until N is larger than T .

Step 5. Embed S' according to Step 6 of Algorithm 4.1.

Step 6. Repeat Step1 through Step 4 until all the blocks and tile images are finish the embedding process.

4.2.3 Image Authentication Process

The data extraction process as well as the image authentication process is described in this section. The method of data extraction is mainly to compare the hue value with the maximum bin peak of each 8×8 block of each tile image. The data extraction algorithm is described as follows.

Algorithm 4.3: Data Extraction Process.

Input: an image mosaic M and a secret key K .

Output: the signals S embedded in M .

Steps.

Step 1. Get the height h and the width w of the tile images by *Algorithm 3.2*.

Step 2. Divide M into multiple tile images using h and w .

Step 3. Calculate the histogram H of the hue component of each 8×8 block of each tile image and quantized it into 12 bins.

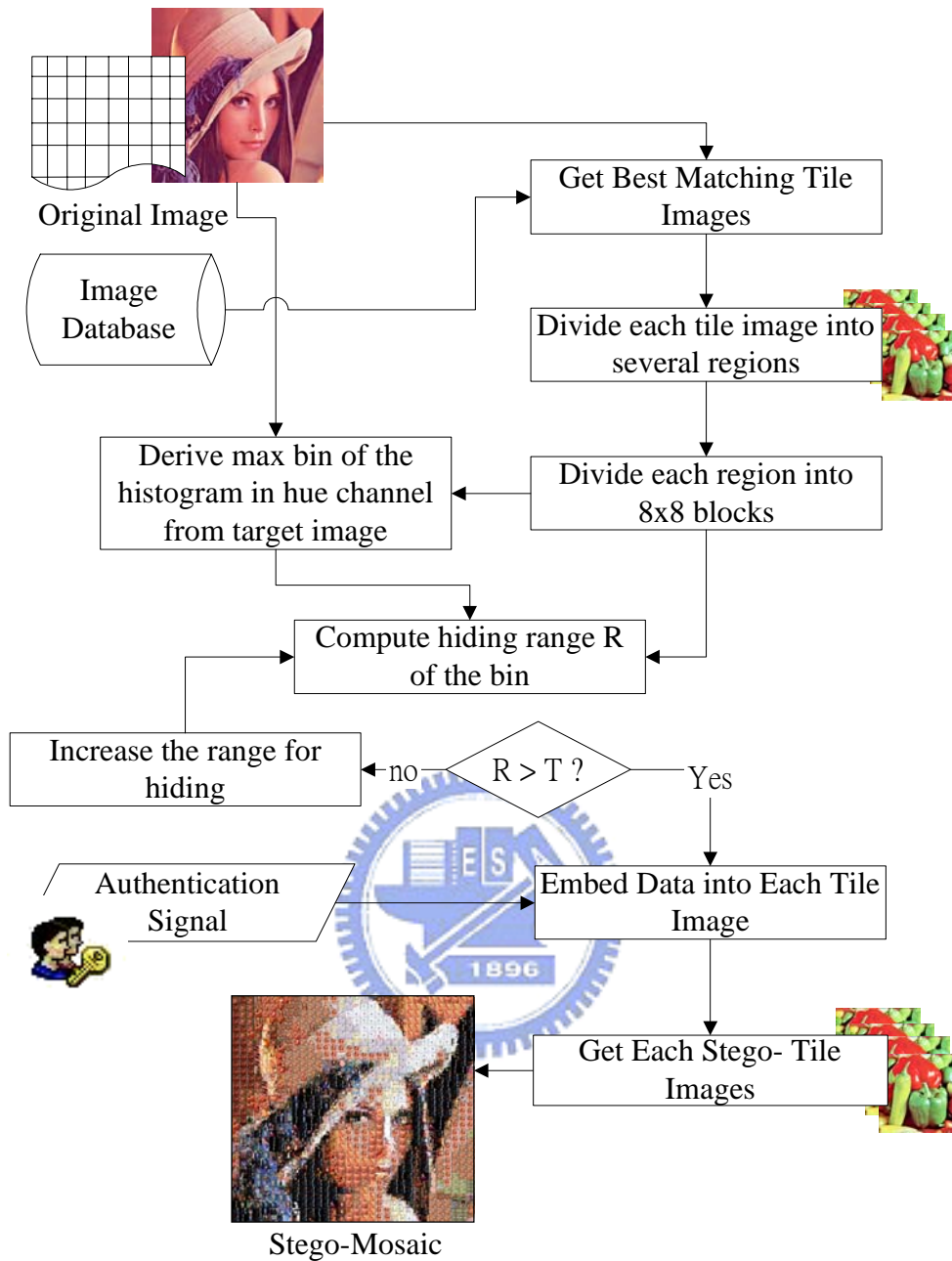


Figure 4.4 Flowchart of proposed embedding process.

Step 4. Derive the maximum peak bin Max of the 8×8 block.

Step 5. Extract data by scanning the entire block according the following rule,

assuming that $H(P_i)$ denotes the hue value of pixel P_i :

$$\begin{cases} \text{if } |H(x_i) - (Max * 30 + 7.5)| \geq 7.5, \text{ extract bit "1"}; \\ \text{if } |H(x_i) - (Max * 30 + 22.5)| \geq 7.5, \text{ extract bit "0"} \end{cases}$$

Step 6. Compose the extracted bits to be S .

The image authentication process is pixel by pixel, block by block, and tile by tile. In this study, two thresholds are given for the purpose of authenticating each 8×8 block in each tile image. The process for image authentication is described as follows.

Algorithm 4.4: Image Authentication Process.

Input: an image mosaic M , a secret key K , and two thresholds T_1 and T_2 .

Output: an authentication report R .

Steps.

Step 1. Derive the maximum peak bin Max by applying Step 1 through Step 4 of Algorithm 4.3.

Step 2. Generate the embedded steam S by taking K as a seed.

Step 3. Derive the extracted data S' of each 8×8 block by applying a voting scheme with T_1 as the voting threshold in the following way.

- A. Let S^1 denotes the extracted data of the 8×8 block.
- B. According to a predefined data-embedding unit, divide S^1 into multiple parts, $S_1, S_2, S_3, \dots, S_n$ with each part containing at least one bit, where n is a number as described in Step 1 of Algorithm 4.2. In the voting scheme, the bits of $S_1, S_2, S_3, \dots, S_n$ are taken sequentially to vote the corresponding bit of S' by the following rule:

if the votes of bit “1” are bigger than T_1 , then set the corresponding bit of S' to be bit “1”; otherwise, if the votes of bit “0” are bigger than T_1 , then set the corresponding bit of S' to bit “0”.

Step 4. Authenticate each 8×8 block by comparing S and S' . If S is identical to S' , then the block is regarded authentic; otherwise, unauthentic.

Step 5. Authenticate each tile image by counting the number of authentic blocks with the threshold T_2 according the following rule:

$\left\{ \begin{array}{l} \text{if the number of authentic blocks} \geq T_2, \text{ then the tile image is authentic;} \\ \text{otherwise, the tile image is unauthentic.} \end{array} \right.$

Step 6. Generate the authentication report.

4.3 Experimental Results and Summary

4.3.1 Experimental results

Some experimental results are given in this section. Figure 4.5 shows an image mosaic with no hidden data while Figure 4.6 shows an image mosaic resulting from the proposed method of histogram modification. Figure 4.7 shows two experimental results of image authentication.

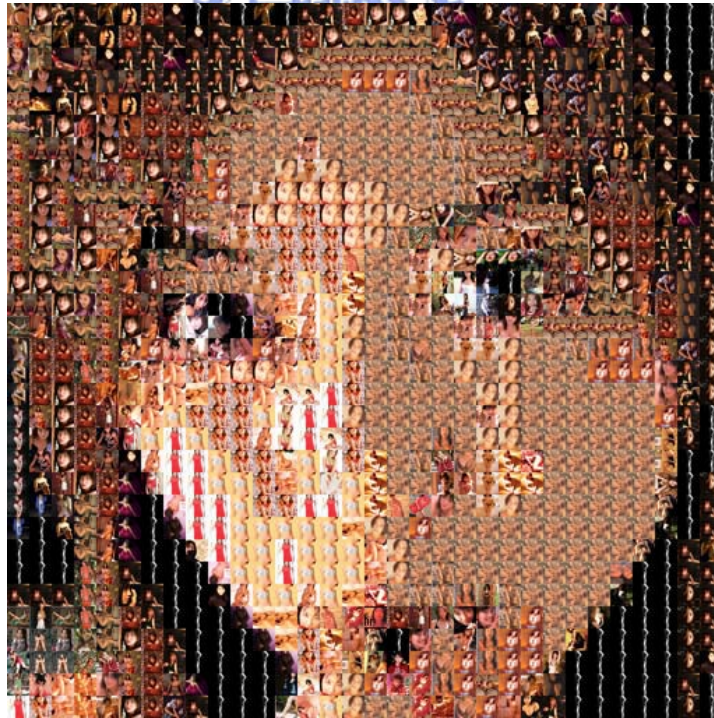


Figure 4.5 Image mosaic without modification

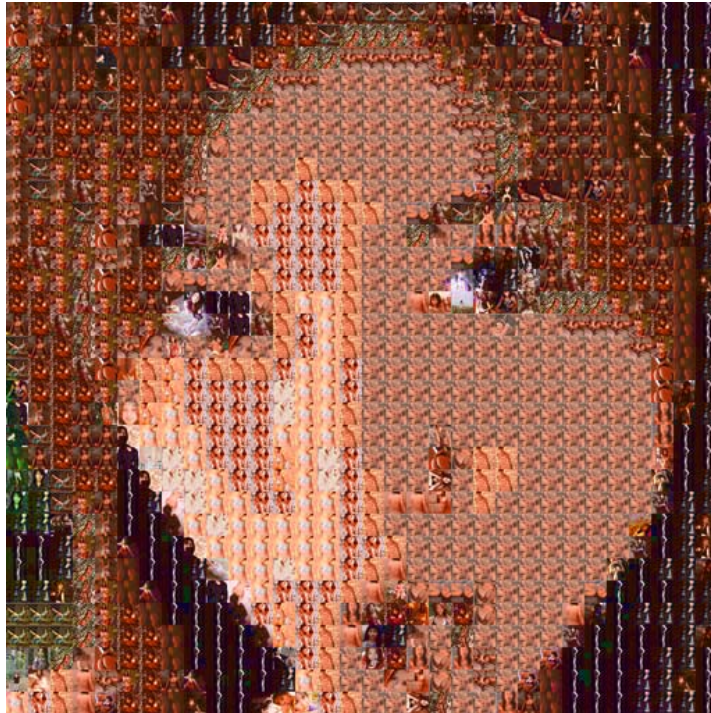


Figure 4.6 The image mosaic of Figure 4.4 with modification

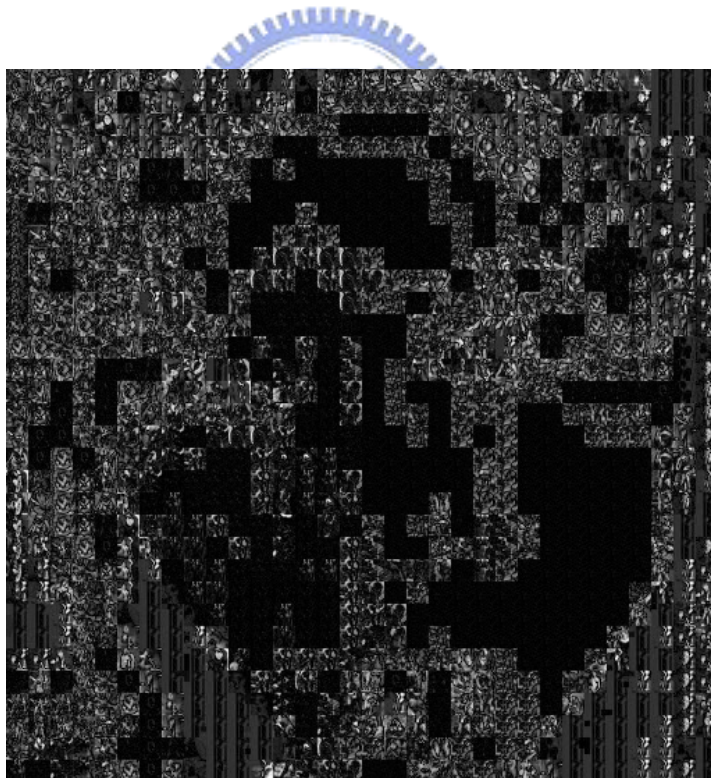


Figure 4.7 The difference image between Figure 4.5 and Figure 4.4

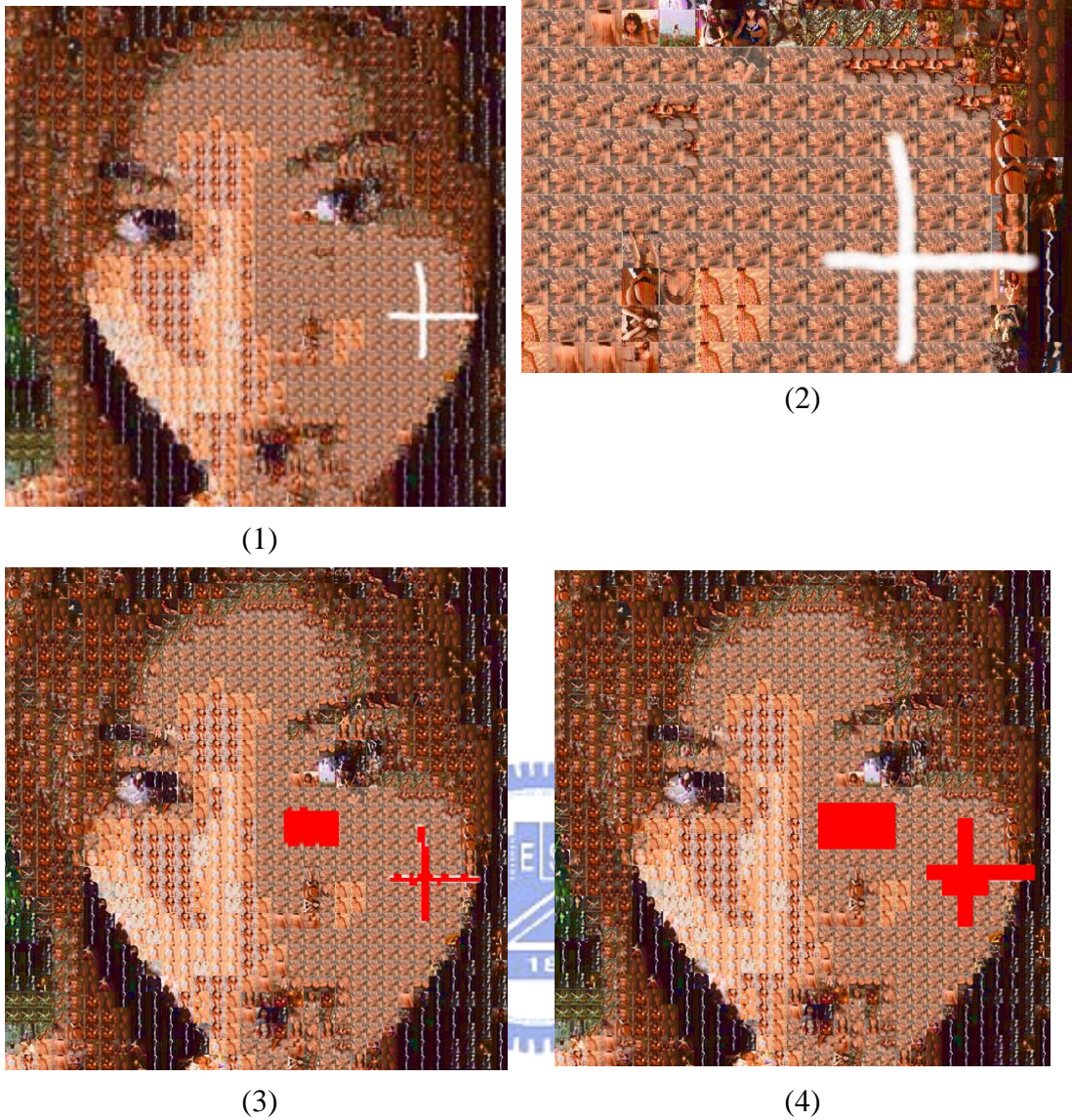


Figure 4.8 Tamper detection of the cover-mosaic. (1) A tampered mosaic. (2) The enlarged tampered region. (3) The result of block-based image authentication. (4) The result of tile-based image authentication.

4.3.2 Summary and Discussions

A data hiding method by histogram modification based on the use of 8×8 blocks has been proposed in this chapter. Generally speaking, the histogram modification method will sometimes cause the distortion of the tile images. They are highly depended on the size of the image database that used for mosaic creation. However, if

the image database is large, then the selected tile image will be very possible similar to the target image and the image distortions result from the proposed histogram modification method will be reduced. The image quality could also be controlled by the threshold T_1 which determines the data hiding capacity. On the other hand, data hiding in the spatial domain is more fragile than that in the frequency domain, so it is easier to achieve the tampering detection to carry out image authentication.

