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

Wei-Liang Lin[1] and Wen-Hsiang Tsai[1, 2]
[1]*Department of Computer and Information Science*
National Chiao Tung University Hsinchu, Taiwan 300
[2]*Department of Computer Science and Information Science*
Taichung Healthcare and Management University, Taichung 413
gis91523, whtsai @cis.nctu.edu.tw

Abstract *- A data hiding method in image mosaics for copyright protection against print-and-scan attacks is proposed. By adding visible boundary regions to the four sides of tile images to represent bit pairs, a binary data stream can be embedded into an input image mosaic. By detecting the visible boundary regions of the resulting stego-image mosaic, the embedded data can be extracted for copyright proof, even after the stego-image mosaic is attacked by print-and-scan operations. Experimental results showing the effectiveness of the proposed method are also included.*

Keywords: data hiding, image mosaics, visible boundary regions, tile images, watermarking, copyright protection, print-and-scan attacks.

1. Introduction

An *image mosaic* is an image obtained by arranging a large number of small *tile images* in a certain way so that each tile image resembles a small block of a given *target image*, and all the tile images together suggest a larger image when seen from a distance. It takes advantage of a property of the human visual system, namely, an observer will only see an average color in a multiple-colored region at a sufficient distance. So far as an image mosaic is concerned, each tile image is an element, which represents the average color of the corresponding region in the target image. The idea of building image mosaics automatically by computers comes from Silvers [1]. And several researches [2-4] investigated this problem with different goals. Zhang, Nascimento and Zaiane [5] gave an automatic measure to assess the quality of the resulting images.

Being regarded as artworks, image mosaics are surprising to most first-time observers and are frequently used as posters, billboards, magazine covers, etc. It is thus desired to protect their copyright. One way is to use digital watermarking techniques. Since image mosaics usually exist in the form of paper copies, the embedded watermark must be strong enough to survive print-and-scan attacks.

In this study, a method for embedding watermarks in image mosaics by the use of visible boundary regions in tile images is proposed. The idea comes from image coding by cryptography. Visible boundary regions are added to each tile image and can be regarded as visible features. These features not only can be extracted from the mosaic in the digital form, but also can be detected after the mosaic goes through a print-and-scan process. The print-and-scan process includes two steps: printing a digital mosaic as a paper copy and scanning the mosaic on the paper. These two steps cause several types of attacks to the watermark, including image scaling and rotation, and color distortion.

In Section 2, the proposed watermark embedding method is described. In Section 3, the application of the method to copyright protection against print-and-scan attacks is described. And in Section 4 a summary is given.

2. Proposed Watermarking Using Visible Boundary Regions in Tile Images

2.1. Properties of Visible Boundary Regions

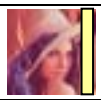
The term *visible boundary region* used in this paper means a strip of a tile image at its left, right, upper, or lower side, whose existence is obvious to an observer. The size of a visible boundary region is one eighth of the tile image and all pixels in the region have the same color. Such regions obviously will have great effects on the appearances of image mosaics. However, if these visible boundary regions are filled with appropriate colors to match those of the corresponding target images and so become parts of the given tile images, then their influences on image mosaics will be reduced. Besides, a statistical property of the specially-designed visible boundary regions is that the RGB color variances of each region are extremely small. This property facilitates designing an effective data extraction process, as described later in this paper.

2.2. Proposed Data Embedding Process

The boundary of a rectangular digital image has four sides, and in the proposed data embedding scheme, it is assumed that different image sides represent different meanings. The left, right, upper, and lower sides in an image are regarded in this

study to represent two bits of data “00,” ”01,” “10,” and “11,” respectively, as shown in Table 1.

Table 1 Types of boundary regions and their meanings.

Region Types		
Bits	00	01
Region Types		
Bits	10	11

In the data embedding process, an input data stream D with L characters is converted into binary form in advance, resulting in $d_1d_2\ldots d_{8\times L}$, and then grouped into consecutive bit pairs, d_1d_2, d_3d_4, …, $d_{8\times L-1}d_{8\times L}$. The process is conducted in the image mosaic creation stage in which tile images are selected and pasted onto the target image. A visible boundary region is added to an appropriate side of each tile image according to the currently-dealt bit pair in the input data stream and Table 1. The color of each added visible region is taken to be the average color of the corresponding region of the tile image. The tile image is resized in advance in such a way that after the region is added, the resulting shape is identical to that of the original tile image, to guarantee the integrity of the image content. For the purpose of facilitating extraction of the hidden data from the resulting image, called *stego-image mosaic*, “variance check” and “noise generation” processes are applied to the other three sides of the boundary after the visible boundary region is added. The detail of the proposed data embedding process is described as an algorithm as follows.

Algorithm 1: ***Data embedding by boundary regions***.
Input: an original image I, a tile image database D, an input stream S to be embedded, a secret key K, and a variance threshold T.
Output: a stego-image mosaic M.
Steps.

1. Divide I into tiles.
2. Extract the color features of each tile.
3. Get from D the best-matching tile image for each tile according to a similarity measure.
4. Compute the hiding capacity C according to the size of I.
5. Generate a stego-stream S' in binary form by encrypting S using K in a certain way, and repeat S' as many times as possible until the length of the stego-stream reaches the hiding capacity C.
6. Partition S' into a group S_i' of bit pairs.
7. Add a visible boundary region into each tile image I by the following steps:
   A. Resize I and add to it a visible region, according to S_i'.
   B. Compute the variances of the other three boundary regions except the added one; and if the computed variance value of any boundary region is smaller than T, then add to it gaussian noise with its mean equal to zero and variance equal to 30.
   C. Repeat Step 7B until the variances of the other three boundary regions are all larger than T.
8. Compose all tiles to produce a stego-image mosaic as the desired output.

2.3. Proposed Data Extraction Process

By assuming that the stego-image is in its digital form without being printed and rescanned, the proposed data extraction process includes two stages: detecting the tile size and extracting the embedded data.

A. Tile size detection

Because an image mosaic is made of many tile images, it contains many obvious horizontal and vertical inter-tile edges. 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 is derived precisely with two ways of verifications of the estimation results. Before describing the tile size detection algorithm, two terms are defined in advance here. First, a *projection in the Y-axis direction* is defined to be the summation of all pixel values in one row of an image. The number of the projections in the Y-axis direction is equal to the number of columns in an image. And a *projection in the X-axis direction* is defined in a similar way. The following algorithm shows the detail about the proposed tile size detection method.

Algorithm 2: ***Tile size detection***.
Input: an image mosaic M.
Output: a tile height H and a tile width W.
Steps.

1. Detect the edges of M by 3×3 Sobel mask as shown in Figure 1 and get a black and white Sobel edge value image S.

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

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

Figure 1. 3×3 Sobel mask.

2. Derive the projections in the X-axis and the Y-axis directions, respectively.
   A. Let S_{ij} denote the pixel gray value at image coordinates (i, j) in image S whose width is w and height is h, where $i = 0, 1, \ldots, h - 1$,

and $j = 0, 1, \ldots, w-1$.

B. Let x_i denote the projection value of the ith row and y_j denote the projection value of the jth column. Define f_x and f_y to be two mapping functions from image coordinates to projection values as follows:

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

C. Let X denote the histogram of x_i and Y denote that of y_j: $X = \{x_i\}$, $i = 0, 1, \ldots, h-1$. $Y = \{y_j\}$, $j = 0, 1, \ldots, w-1$.

3. Get two sets PX and PY of the peaks in X and Y by applying a Laplacian mask as shown in Figure 2 and a thresholding technique with two predefined thresholds T_x and T_y.

A. Apply a one-dimension Laplacian mask.

-1	2	-1

Figure 2 Laplacian mask.

B. Calculate two thresholds T_x and T_y as follows:

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

where c_1 and c_2 are two pre-defined coefficients, and *Max* is a function that returns the maximum value of the input set.

C. Find two sets PX and PY of peak values of x_i and y_i according to the following rules:

$$\text{if } x_i \in X \text{ and } x_i > T_x, \text{ then let } i \in \{PX\};$$
$$\text{if } y_j \in Y \text{ and } y_j > T_y, \text{ then let } j \in \{PY\}.$$

4. Derive the centers PX^c and PY^c of the two peak sets PX and PY, respectively, using a clustering method and a pre-defined radius R in the following way.

A. Let PX_i denotes the ith elements of PX, and let PY_j denotes the jth elements of PY.

B. Define the centers PX^c and PY^c according the following rules:

$$\text{if } f_x(PX_i) = \underset{i=(i-R),\ldots,(i+R)}{Max}(f_x(PX_i)),$$
$$\text{then } PX_i \in \{PX^c\};$$
$$\text{if } f_y(PX_j) = \underset{j=(j-R),\ldots,(j+R)}{Max}(f_y(PX_j)),$$
$$\text{then } PY_j \in \{PY^c\}.$$

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

A. Let Sx_i denote the ith difference between the ith and the $(i+1)$th peaks of PX^c, and let Sy_j be defined similarly. Compute Sx_i and Sy_i according to the following formula:

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

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

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

6. Get temporary tile length and width W_t and H_t according the following rule:

$$W_t = k, \text{ if } Sx_k = \underset{i=0,1,\ldots,|Sx|-1}{Max}(Sx_i);$$
$$H_t = h, \text{ if } Sy_h = \underset{j=0,1,\ldots,|Sy|-2}{Max}(Sy_j).$$

7. Re-compute the values W_t and H_t using the projection values in the X- and Y-axis directions in the following way.

A. Get the exact sets PX_t of the local maximums of the projection values in the X-axis direction by comparing the projections values around each x-coordinate which is a multiple of W_t.

B. Apply Steps 5 and 6 to PX_t and re-compute the value of W_t.

C. Apply the same process described by Steps 7.A and 7.B above to the projections in the Y-axis direction to get a new value of H_t.

8. Correct the values of W_t and H_t by dividing the width and height of the image mosaic M by them, respectively. If the remainder is not zero, W_t and H_t will be incremented or decremented by 1 until the remainder is zero.

9. Take the final W_t and H_t to be the desired values for W and H.

The main idea of the above algorithm is to get the average distance between two adjacent edges. But there are two problems. The first is that the number of tiles in a row or a column in the mosaic is unknown, and the second is that it is difficult to find local maxima in the projection values without the numbers of tiles. As a result, this method may be regarded as a kind of non-supervised learning which finds out the number of tiles in a row and in a column, as well as the tile size by analyzing the statistics of the peaks. Besides, in order to get a correct tile size, the temporary results are verified twice both in Step 7 and in Step 8. An example of Sobel edge value images obtained in Step 1 is shown in Figure 3. Figures 4 illustrate the projections in the X- and Y-axis directions of the Sobel image, respectively.

B. Data Extraction Process

The data extraction process is applied after the tile size is detected as described in the previous section 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 1, we can then extract the embedded data from the location (left, right, upper, or lower) of the added region.

Algorithm 3: ***Data Extraction.***

Input: an Image Mosaic *M*, a height *H* of tile image, a width *W* of tile image, and a secret key *K*.
Output: the extracted data *E*.
Step.

1. Divide *M* into tile images sequentially according to the input parameters *H* and *W*.
2. Calculate the variances of the four boundary regions of a tile image.
3. Compare the four variances to get the side with the minimum variance.
4. Derive and save the two bits of one character group according to Table 3.1 and the detected side location.
5. Repeat Step 2 through Step 4 until the process for all tile images is completed, and return the record E_1.
6. Get the extracted data by a process of decrypting E_1 using *K*.

3. Copyright Protection against Print-and-Scan Attacks

The proposed method described previously does not deal with print-and-scan attacks. That is, data extraction is conducted directly on stego-image mosaics in digital form. It is found from this study that even though a stego-image mosaic is printed to become a paper copy which is then rescanned to yield a second but quality-degraded digital version (called the *print-and-scan version* of the stego-image mosaic), the hidden data represented by the visible boundary regions of the tile images in the print-and-scan version can still be extracted after some additional image processing works are performed. Therefore, we can say that the added boundary regions in the tile images are robust against print-and-scan attacks to a certain degree. The boundary regions may also be regarded as a kind of *semi-visible watermark* because they are visible when observed carefully but they compose a coded watermark unknown to the observer.

The previously-mentioned additional image processing works include at least the task of reorienting the print-and-scan version of the stego-image mosaic because a paper copy of the original stego-image, when rescanned, might be slanted more or less. Color distortion and image scale changes might also occur in the print-and-scan process and so corrections of these effects are also necessary, but we assume in this study that these kinds of changes are minimal or ignorable and leave the correction works for dealing with them as topics for future studies.

In the following we describe how we reorient a print-and-scan version of a stego-image mosaic before the data extraction process.

3.1. Reorientation of Print-and-Scan Version

The proposed image reorientation algorithm is based on the use of edge detection and image projection techniques. Before scanning a printed mosaic picture with a table scanner, the picture is placed on a flat surface and a window is usually selected to specify the scanning scope. Here we assume that the picture is rectangular in shape and is placed carefully enough with a very small slant angle with respect to the scan window boundary. In the proposed image reorientation algorithm, the print-and-scan image version is rotated many times, each time with a small angle, to find out the slant angle of the image by an image projection method. A slant angle is obtained by edge detection and detection of the maximum projection value in those of all the rotated images. Finally the image is re-oriented through the detected slant angle. Figure 5 illustrates the previously-mentioned reorientation process and a corresponding algorithm is described as follows.

Algorithm 4: ***Image reorientation.***
Input: an image mosaic *M* possibly slanted.
Output: a reoriented image mosaic *S*.
Steps.

1. Apply Steps 1 and 2 of Algorithm 2 to *M* to get the projections in the *X*-axis direction.
2. Detect and save the maximum projection value in a set R.
3. Obtain a new image M' by rotating *M* with a predefined small angle.
4. Repeat Steps 1 through 3 with M' as input for a predefined number of times.
5. Compare the maximum values saved in R to get a global maximum value *P*.
6. Take the slant angle *A* corresponding to *P* and rotate *M* accordingly to get a reoriented image mosaic S as output.

3.2. Experimental Results

Some experimental results of applying the above-mentioned methods are shown here. The related setups of the experiments are shown in Table 2 and some images showing print and scan effects are shown in Figure 6. We conducted experiments on image reorientation and watermark extraction using Figure 6, and the result is shown in Table 3 in which the error rate is defined as the ratio of the number of correct pixels in the extracted watermark to that of the pixels in the original watermark. From the data shown in Table 3, we see that the proposed method is effective to yield recognizable watermarks after print-and-scan attacks.

4. Summary

A method of data hiding in image mosaics for copyright protection has been proposed, which is robust against print-and-scan attacks by embedding semi-visible watermarks in the form of visible boundary regions in tile images. A technique for reorienting slanted images after the print-and-scan process has also been proposed. The experimental results show the feasibility of the method. The error rates of watermark extraction shown in Table 3 seem

image-dependent. Many factors influence the results of watermark extraction, such as scanner setups and printer quality. Besides, the number of different colors in the tile image may be an important factor to the watermark extraction process

Acknowledgement

This work was supported partially by the NSC Program for Advanced Technologies and Applications for Next Generation Information Networks (II) – Sub-project 5: Network Security, Project No. NSC93-2752-E-009-006-PAE.

References

[1] R. Silvers, and M. Hawley, *Photomosaics*, Henry Holt and Co., 1997.

[2] R. Silvers. "Digital Composition of a Mosaic Image," US Patent No. 957833, October 2000.

[3] A. Finkelstein, and M. Range. "Image Mosaics," *Technical Report: TR-574-98*, Computer Science Department, Princeton University, Princeton, U.S.A., 1998.

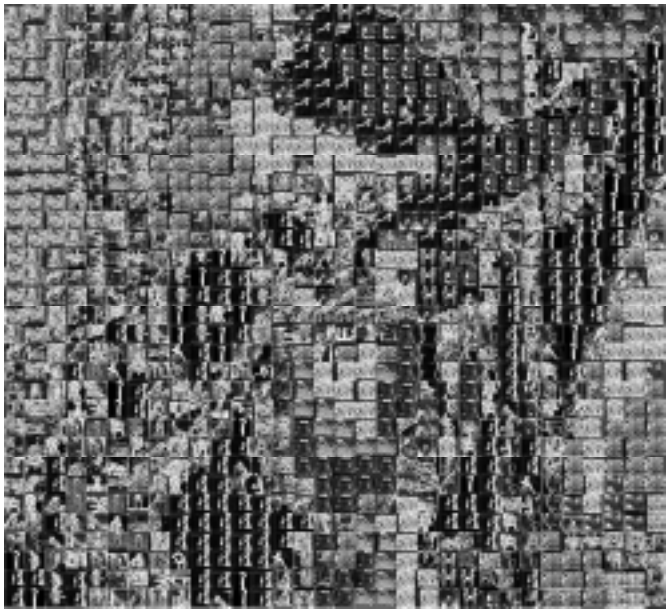
[4] N. Tran. "Generating Photomosaics: An Empirical Study," *Proc. 1999 ACM Symposium on Applied computing* (*SAC '99*), San Antonio, Texas, U. S. A., 1999, pp. 105-109.

[5] Y. Zhang, M. A. Nascimento, and O. R. Zaiane, "Building Image Mosaics: An Application of Content-Based Image Retrieval," *Proc. IEEE Int. Conf. on Multimedia and Expo* (*ICME' 03*), Baltimore, U.S.A., July, 2003.

[6] C. Blundo, and C. Galdi, "Hiding Information in Image Mosaics," *The Computer Journal*, vol.46, issue 2, Feb. 2003, pp. 202-212.

[7] Y. J. Cheng, "Copyright and Integrity Protection for Images by Removable Visible Watermarking Techniques," *M. S. Thesis*, Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan, June 2002.

(a) Original image.



(b) Sobel edge value image of (a).

Figure 3 Sobel edge value image of an image mosaic

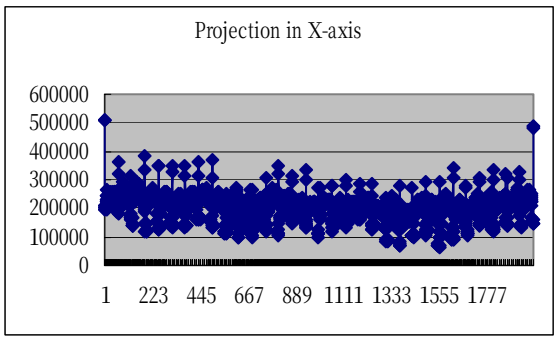


Figure 4 Image projections in X direction.

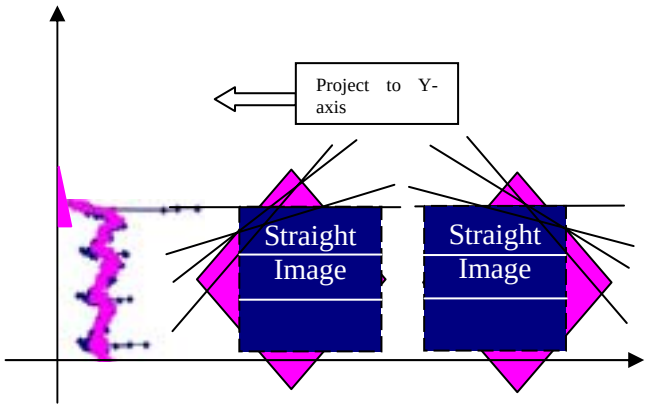
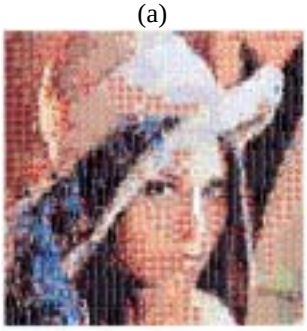


Figure 5 Slant angle detection where the straight image has the maximum projections value.

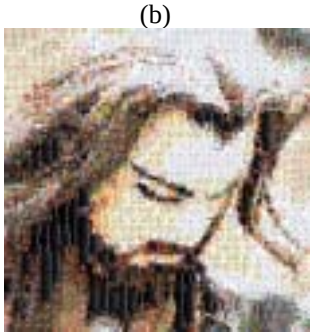
Table 2 Related setup of experiments.

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

(a)



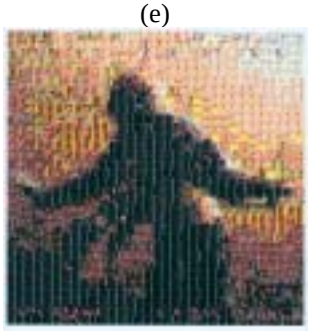
(b)



(c)

(d)

(e)



(f)

Figure 6 Effects of print-and-scan attacks. Images mosaics (a)(c)(e) are the original ones and (b)(d)(f) are the versions yielded by print-and-scan attacks.

Table 3 Watermarks extracted from image mosaics shown in Figure 6 and error rates of extracted data.

Original watermark					
	Extracted Watermark	Error rate		Extracted Watermark	Error rate
(a)		0.5%	(b)		4.5%
(c)		0.1%	(d)		6.8%
(e)		0.3%	(f)		18.5%