國立交通大學

多媒體工程研究所

碩 士 論 文

基於俄羅斯方塊的馬賽克畫：一種新的藝術畫與
其在資訊隱藏上的應用

Tetromino-based Mosaics — A New Type of Art Image and Its
Applications in Information Hiding

研 究 生：張均培

指導教授：蔡文祥　教授

中 華 民 國 九 十 八 年 六 月

基於俄羅斯方塊的馬賽克畫：一種新的藝術畫與其在資訊隱藏上的應用

Tetromino-based Mosaics — A New Type of Art Image and Its Applications in Information Hiding

研　究　生：張均培　　　　Student: Chun-Pei Chang

指導教授：蔡文祥　　　　Advisor: Prof. Wen-Hsiang Tsai

國　立　交　通　大　學

多　媒　體　工　程　研　究　所

碩　士　論　文

A Thesis

Submitted to Institute of Multimedia Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

June 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年六月

基於俄羅斯方塊的馬賽克畫：一種新的藝術畫與其在資訊隱藏上的應用

研究生: 張均培　　　　　　指導教授: 蔡文祥 博士

國立交通大學多媒體工程研究所

摘要

在本論文中，我們以俄羅斯方塊(Tetris)為基本元件，創造了一種新的藝術畫，稱之為俄羅斯方塊馬賽克畫(tetromino-based mosaic images)，並研究自動產生這種影像與在其中做資訊隱藏的技術。在俄羅斯方塊馬賽克畫中，我們找到了兩種可供隱藏資訊的特性，分別是俄羅斯方塊的排列組合和其顏色的變化。首先，我們利用樹狀搜尋演算法來找出所有俄羅斯方塊的排列組合，藉由選擇不同的組合，將秘密資訊或浮水印隱藏於俄羅斯方塊馬賽克畫之中。同時，利用偵測邊界所得到的資訊，我們也提出了一種強化邊緣的方法來改善上述資訊隱藏技術所造成的雜訊。另一方面，藉由調整鄰近俄羅斯方塊的顏色，我們可以將秘密資訊隱藏於其中，達到秘密訊息傳輸與版權保護之目的。此外，我們也提出一種可視浮水印技術來保護俄羅斯方塊馬賽克畫的版權，而且所嵌入的可視浮水印可以無失真地移除之。基於這種浮水印技術，最後我們利用調色盤的對應關係，將秘密影像隱藏於浮水印區域，來達到影像秘密傳輸之目的。透過良好的實驗結果，我們證明了所提出方法的實用性。

Tetromino-based Mosaics — A New Type of Art Image and Its Applications in Information Hiding

Student: Chun-Pei Chang Advisor: Prof. Wen-Hsiang Tsai, Ph. D.

Institute of Multimedia Engineering, College of Computer Science

National Chiao Tung University

ABSTRACT

A new type of art image is created in this study, namely, tetromino-based mosaic image. Methods for automatic creation of images of this type and data hiding in them are proposed. For the creation of tetromino-based mosaic images, we find all the possible combinations of tetrominoes by a tree enumeration algorithm. A tetromino database is constructed with different colors and tetromino combinations for image creation. We also propose a border enhancement process for improving visual effects of tetromino edges. For data hiding in tetromino-based mosaic images, we utilize two types of features of mosaic images, tetromino combination and color, to embed data. We can hide data by using distinct combinations of tetrominoes. An edge fitting method is used to enhance edge effects by adjusting combinations of tetrominoes using the information obtained by edge detection. In addition, the colors of tetrominoes are also used to hide data by shifting color values of tetrominoes slightly. For watermarking in tetromino-based mosaic images, we propose a removable lossless watermarking method by replacing the colors according to a mapping between two color palettes. Based on this invertible watermarking method, an image steganographic method is also proposed by replacing the colors of the watermark area according to the proposed mapping between the two color palettes. Experimental results show the feasibility of the proposed methods and systems.

ACKNOWLEDGEMENTS

I am in hearty appreciation of the continuous guidance, discussions, support, and encouragement received from my advisor, Dr. Wen-Hsiang Tsai, not only in the development of this thesis, but also in every aspect of my personal growth.

Thanks are due to Mr. Tsung-Yuan Liu, Mr. Chih-Jen Wu, Mr. Che-Wei Lee, Mr. Guo-Feng Yang, Mr. Jian-Yuan Wang, Mr. Yi-Chen Lai, Miss Mei-Fen Chen, Miss Chin-Ting Yang, Miss Chiao-Chun Huang and Miss Shu-Hung Hung for their valuable discussions, suggestions, and encouragement. Appreciation is also given to the colleagues of the Computer Vision Laboratory in the Institute of Multimedia Engineering at National Chiao Tung University for their suggestions and help during my thesis study.

Finally, I also extend my profound thanks to my family for their lasting love, care, and encouragement. I dedicate this dissertation to my parents.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1
Introduction

1.1 Motivation and Background

1.1.1 Motivation of Study

With the progress of computer networks, interpersonal communication is more and more popular than before. Exchanges of digital files have become convenient and fast through the Internet. For example, many people have the experience of sharing their photographs of traveling with families and friends on instant messaging services such as MSN Messenger and Skype. However, this trend brings many drawbacks in communication security. For example, hackers on the Internet can easily duplicate or tamper with the contents of digital files, causing many legal problems of privacy and copyright. In addition, important information, such as bank account passwords or intelligence data, is extremely secret and should be protected carefully. This demand has generated wide interests in enhancing the security involved in digital file transmission. Therefore, the study of covert communication and copyright protection becomes an important issue.

Different from the traditional information hiding techniques, this study tries to combine approaches to information hiding and art image creation. Using art images as camouflages of secret data brings some benefits. Someone who intends to steal the data may consider that these art images are only artistic productions and ignore them easily. In addition, a synthesis of information hiding techniques and art image creation can also be applied to enhance copyright protection and authentication for the art image which we create. These properties assure that approaches to information hiding in art images are safer than traditional ones.

Due to geometric characters, Tetris and tetrominoes, which will be introduced in detail in the following section, are chosen as materials to create mosaic images for use as the art image mentioned above. Tetrominoes can be used to fill a plane in certain arrangement without overlapping. This property makes tetrominoes become good components of a new type of mosaic image, called *tetromino-based mosaic image* in this study. It is desired to propose a method for creation of tetromino-based mosaic images and design new information hiding techniques for covert communication and copyright protection applications by taking advantages of the characteristics of this new type of mosaic image.

1.1.2 Introduction to Tetris and Tetrominoes

Tetris is a popular computer game invented by a Russian mathematician, named Alexey Pajitnov, in 1985. He derived the game’s name from the Greek numerical prefix "tetra-" and tennis, his favorite sport. "Tetra-" means that all of the game's pieces, called *tetrominoes*, contain four segments. A tetromino is a geometric shape composed of four squares, connected orthogonally. There are five basic tetrominoes (shown in Figure 1.1), and nineteen oriented tetrominoes obtained by rotating and reflecting the basic tetrominoes.

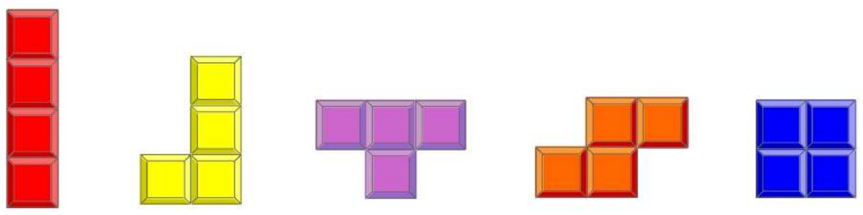


Figure 1.1 Five basic tetrominoes.

The game rule of Tetris is as follows: An initial game board, a rectangular grid with all empty cells, is given. A random sequence of tetrominoes is generated, and the player is allowed to rotate and horizontally move the falling piece before it lands on filled cells. If an entire row of the game board is filled with tetrominoes, all cells of this row are cleared and the game score is increased gradually. A player loses when a new piece is blocked by filled cells and cannot enter the game board anymore. Breukelaar et al. [1] showed that the problem of deciding whether a given Tetris configuration can be cleared using a given sequence of pieces is NP-hard.

In this study, we will not only use tetrominoes as materials for creating a novel type of mosaic image which is called *tetromino-based mosaic image* in this study, but also develop information hiding techniques in tetromino-based mosaic images for covert communication, copyright protection, and image steganography applications.

1.1.3 Introduction to Mosaic Images

Mosaic is the art of surface decorations composed of numerous small pieces of colored glass, stone, or other materials. It is used for cultural or religious significance in Greek and Roman times over 2000 years ago, and is still widely used today in many ways, such as decorations of house interiors, commercials and church windows, etc.

Creating different kinds of mosaic images by computing is a new research topic in recent years. A traditional mosaic image is obtained by arranging a large number of small images, called *tile images*, in a certain way so that each tile image presents a small piece of the *source image*, named *target image*. The overall effect is that the *mosaic image* seems like the source image when seen from a distance. This effect utilizes a property of the human visual system that an observer standing far away will only see the average color of a region even though the region is actually full of different colors. As a result, the key point of mosaic image creation is to make the average colors of tiles to be as similar to those of the target images as possible. Battiato et al. [2] gave the mathematical problem definition of mosaic creation and optimization issue as follows:

> *Given a rectangular region I^2 in the plane R^2, a tile dataset and a set of constraints, find N sites $P_i(x_i, y_i)$ in I^2 and place N tiles, one at each P_i , such that all tiles are disjoint, the area they cover is maximized and the constraints are verified as much as possible.*

In the 1970's, American artist Close [3] began producing gridded paintings by using some characteristics of the human visual system. Some of his paintings are shown in Figure 1.2. Harmon [4] created the first result of mosaic images through the computer and designed an automatic human-face recognition system for mosaic creation in 1973. A resulting image of his method is revealed in Figure 1.3 (a). Dali [5] imitated Harmon's achievement to paint a famous portrait of Abraham Lincoln, named *"Lincoln in Dalivision"* as seen in Figure 1.3 (b). He painted this masterpiece by putting many small images together which include a picture of his wife.

In this research, we are not only inspired by Close's and Dali's paintings to raise the idea of establishing a scheme for a new type of mosaic image creation, but also paid attention to its applications in information hiding.

1.2 Overview of Proposed Methods

In this thesis study, we concentrate our attention on full-color images. First of all, a typical scheme for creation of tetromino-based mosaic images and the related process are given. Then two data hiding techniques and applications of covert communication and watermarking are proposed based on the above-mentioned scheme. Furthermore, a removable lossless visual watermarking technique for tetromino-based mosaic images is proposed. Finally, we propose an image steganographic scheme for tetromino-based mosaic images based on the use of the watermarking technique.

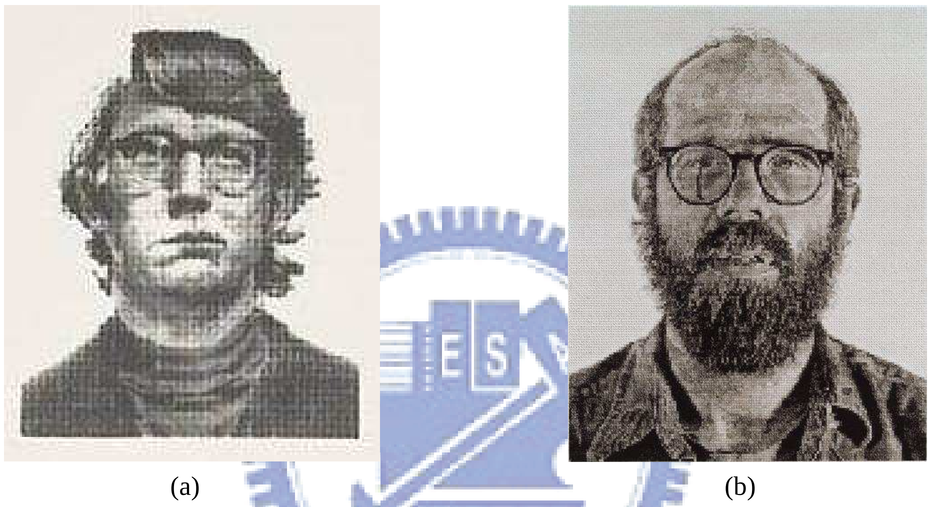


Figure 1.2 Close's paintings. (a) "Keith III, State I," 1975. (b) "Self-Portrait," 1977.

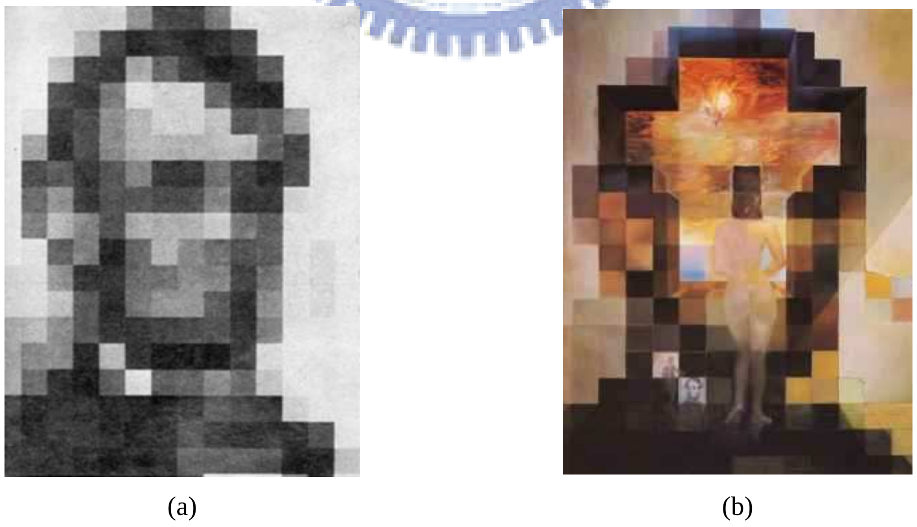


Figure 1.3 (a) "Lincoln Portrait" by Harmon. (b) "Lincoln in Dalivision" by Dali.

1.2.1 Definitions of Terms

Before describing the proposed methods, some definitions of terms are given to facilitate the understanding of the remainder of this thesis.

1. *Source image*: A source image is an image chosen to produce a mosaic image.
2. *Tile image*: A tile image is a small image, similar to a small specific block of the source image.
3. *Target image*: A target image is a sub-image of the source image obtained by dividing the source image into tiles. For each target image, there is a corresponding tile image for substitution in the mosaic creation process.
4. *Mosaic image*: A mosaic image is obtained by arranging a large number of tile images in a certain way, yielding an effect like the source image when seen from a distance.
5. *Cover image:* A cover image is a medium to be embedded with a watermark for copyright protection or some information for covert communication.
6. *Stego-image*: A stego-image is produced by embedding a watermark or some information into a cover image.
7. *Creation process*: A creation process generates a mosaic image from source image.
8. *Embedding process*: An embedding process is a process to embed data in an image.
9. *Extraction process*: An extraction process is a process to extract data from an image.
10. *Recovered image*: a recovered image is an image that is produced by removing an embedded visible watermark from a stego-image.
11. *Lossless recovery process*: A lossless recovery process is a process to

remove the embedded visible watermark from a stego-image and obtain exactly the original cover image without distortion.

12. *Image steganography*: Image steganography is the scheme to embed secret information into images for covert communication such that the secret transmitted in the communication is imperceptible.

1.2.2 Brief Description of Proposed Method for Creation of Tetromino-based Mosaic Images

The proposed creation process of tetromino-based mosaic images is depicted in Figure 1.4. According to the flowchart, first of all, we construct a mosaic database with tetromino combinations and colors. Then we divide a given image into many tile images to find the best matching combination and colors of tetrominoes from the mosaic database for each tile image. The tetromino-based mosaic image for the given image is generated by composing all tile images in a certain arrangement. Moreover, some related processes of such mosaic creation are proposed to improve the visual effects of the generated tetromino-based mosaic image. The detailed creation process will be described in Chapter 3.

1.2.3 Brief Description of Proposed Method for Data Hiding in Tetromino-based Mosaic Images by Distinct Combinations of Tetrominoes

Based on the scheme of above-mentioned tetromino-based mosaic image creation, we propose a data hiding technique in tetromino-based mosaic images by using distinct combinations of tetrominoes. The data embedding process of such data hiding is illustrated in Figure 1.5. At the beginning, we transform given data which will be hidden into a bit sequence. The given data might be a watermark, a piece of secret message, or any digital data which you want to transmit by covert communication. A user key is used to strengthen the security of data hiding by disarranging the bit sequence of the given data. Second, an edge fitting process is performed by edge detection thresholding before the data embedding process. Finally, a tetromino-based mosaic image with embedded data is created by the data embedding process using distinct combinations of tetrominoes. The detailed algorithm of the embedding method will be described in Section 4.3.2.

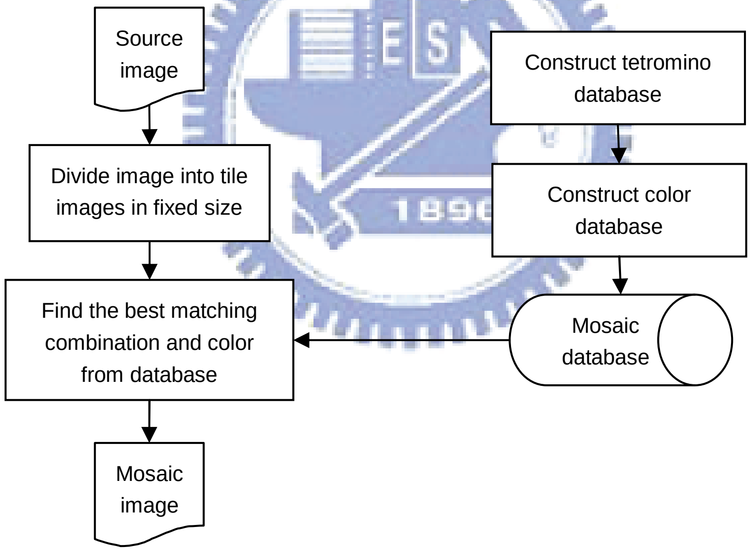
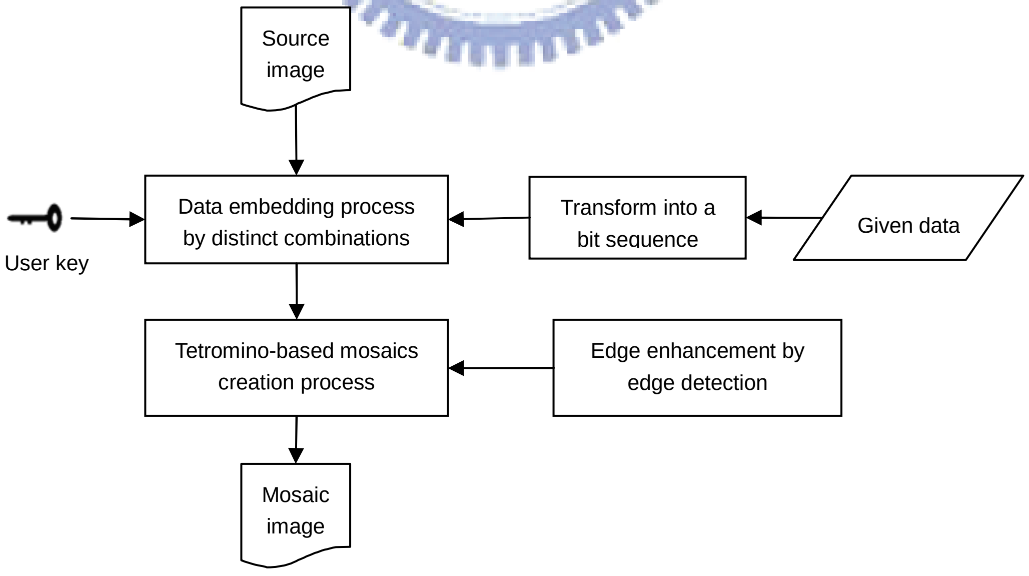

Source image
Construct tetromino database
Divide image into tile images in fixed size
Construct color database
Find the best matching combination and color from database
Mosaic database
Mosaic image

Figure 1.4 Creation process of tetromino-based mosaic images.

The data extraction process is an inverse version of the above-mentioned data embedding process. After the tetromino shape detection process, the extraction process is performed to obtain the embedded data by recognizing the combinations of tetrominoes. Then the user key is utilized to recover the given data in its original order. The description of the extraction process will be described in detail in Section 4.3.3.

1.2.4 Brief Description of Proposed Method for Data Hiding in Tetromino-based Mosaic Images by Small Tetromino Color Shiftings

A tetromino is composed by four squares with the same color. However, human visual system cannot be aware of small changes of colors. According to this property, colors of each tetromino can be shifted in small amounts to embed data. The embedding process of data hiding is illustrated in Figure 1.5. At first a tetromino shape detection process is performed on an input tetromino-based mosaic image. Then, the data embedding process by small color shiftings in each tetromino is performed and a user key is used to disarrange the given data for safety. Finally, a tetromino-based mosaic image with embedded data is generated through an imperceptible data embedding process.



Source image
User key
Data embedding process by distinct combinations
Transform into a bit sequence
Given data
Tetromino-based mosaics creation process
Edge enhancement by edge detection
Mosaic image

Figure 1.5 Embedding process of data hiding by distinct combinations of tetrominoes.

The data extraction process is a reversed procedure of the data embedding process. After the execution of a tetromino shape detection process, the extraction process by tetromino color detection is performed to obtain the embedded data. The detailed algorithm of the data embedding and extraction processes will be stated in Section 5.2.2 and 5.2.3, respectively.

1.2.5 Brief Description of Proposed Method for Removable Lossless Visible Watermarking in Tetromino-based Mosaic Images

Tetromino-based mosaic images are composed by limited colors which integrally are similar to a palette in concept. As a result, we can employ this characteristic to apply some palette-like watermarking technique to tetromino-based mosaic images. The proposed process of removable lossless watermarking in tetromino-based mosaic images is illustrated in Figure 1.7. In the first place, two palettes called *source palette* and s*ecret palette* are created by counting the colors of the source image and the secret image respectively. Second, a corresponding table of these two palettes for watermarking is constructed by the use of a measure of Euclidean distance. At last, the colors of the source image in the watermarked area are replaced with the corresponding colors in the table to form a watermarked stego-image. The watermark removing process is an inverse process of the watermark embedding process. The detailed algorithm will be described in Chapter 6.

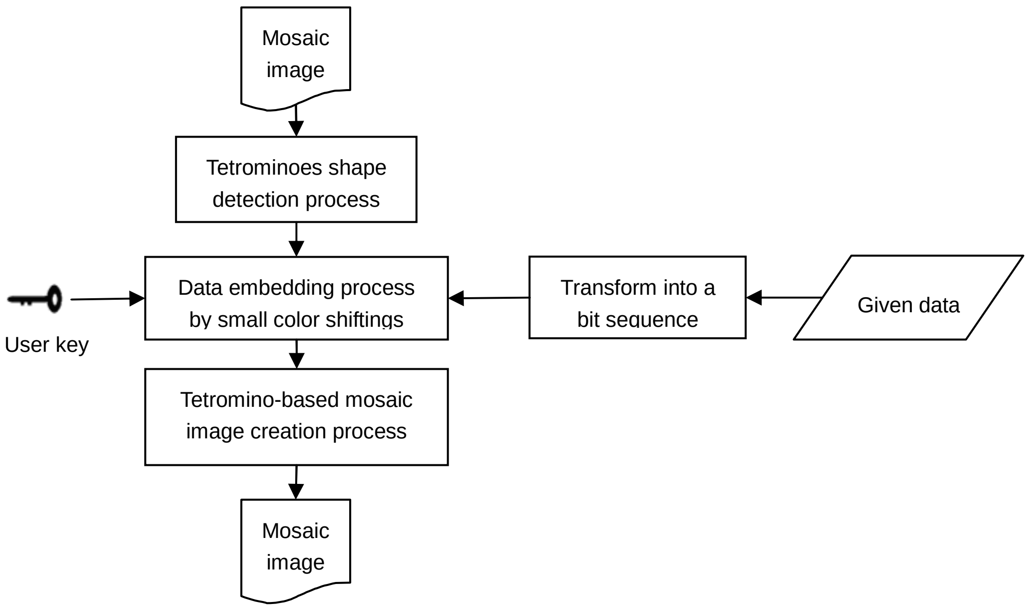

Mosaic image
Tetrominoes shape detection process
Data embedding process by small color shiftings
Transform into a bit sequence
Given data
User key
Tetromino-based mosaic image creation process
Mosaic image

Figure 1.6 Embedding process of data hiding by small color shiftings

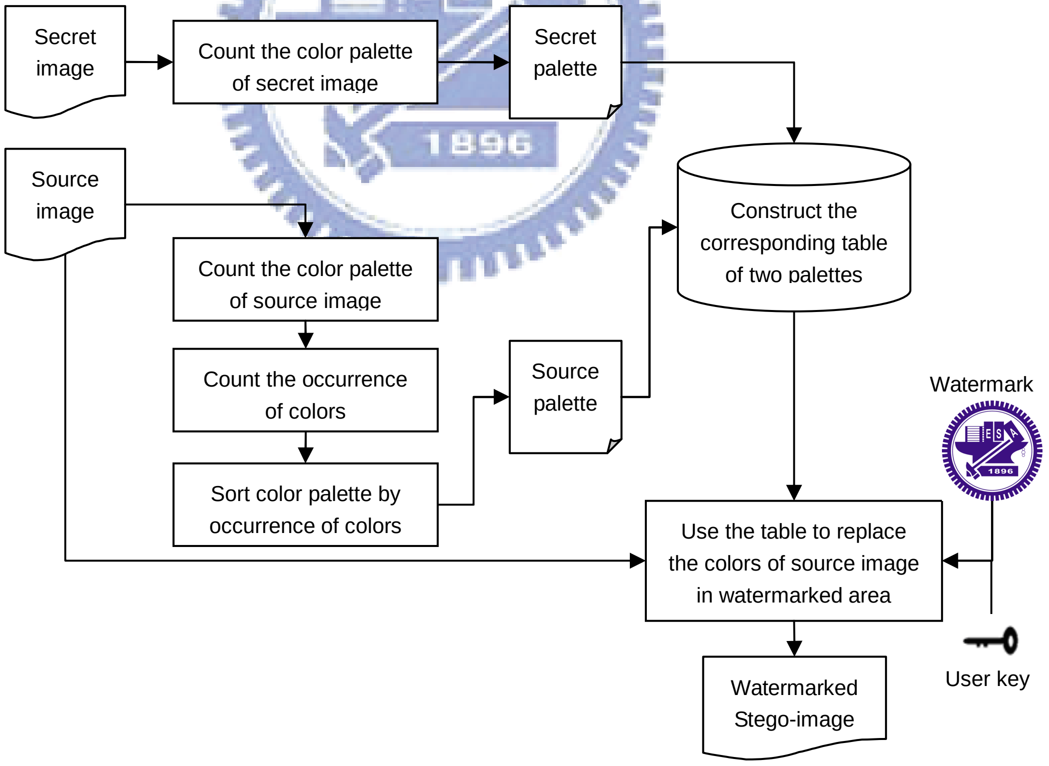

Secret image
Count the color palette of secret image
Secret palette
Source image
Construct the corresponding table of two palettes
Count the color palette of source image
Count the occurrence of colors
Source palette
Watermark
Sort color palette by occurrence of colors
Use the table to replace the colors of source image in watermarked area
User key
Watermarked Stego-image

Figure 1.7 Removable lossless watermarking process in tetromino-based mosaic images.

1.2.6 Brief Description of Proposed Method for Image Steganography by Watermarking in Tetromino-based Mosaic Images

The proposed method for image steganography is achieved by the concept of the removable watermarking technique. A watermarked tetromino-based mosaic image is created as a cover image to embed a secret image by the palette-like watermarking technique. In other words, the watermarked area is used to embed a secret image for image steganography. The detailed algorithm will be described in Chapter 7.

1.3 Contributions

Some major contributions of this study are listed as follows.

1. A method to find all arrangements of tetrominoes to fill a plane and construct a tetromino database for mosaic creation.
2. A method to create a new type of mosaic images composed of tetrominoes is proposed.
3. A method to recognize the tetromino combinations in tetromino-based mosaic images is proposed.
4. A method for data hiding in tetromino-based mosaic images by distinct combinations of tetrominoes is proposed.
5. A method for data hiding in tetromino-based mosaic images by small tetromino color shiftings is proposed.
6. A method to enhance the visual effects of edges by a border enhancement and an edge fitting process is proposed for tetromino-based mosaic

images.

7. A method for removable lossless watermarking technique for tetromino-based mosaic images is proposed.
8. A method for image steganography by watermarking in tetromino-based mosaic images is proposed.

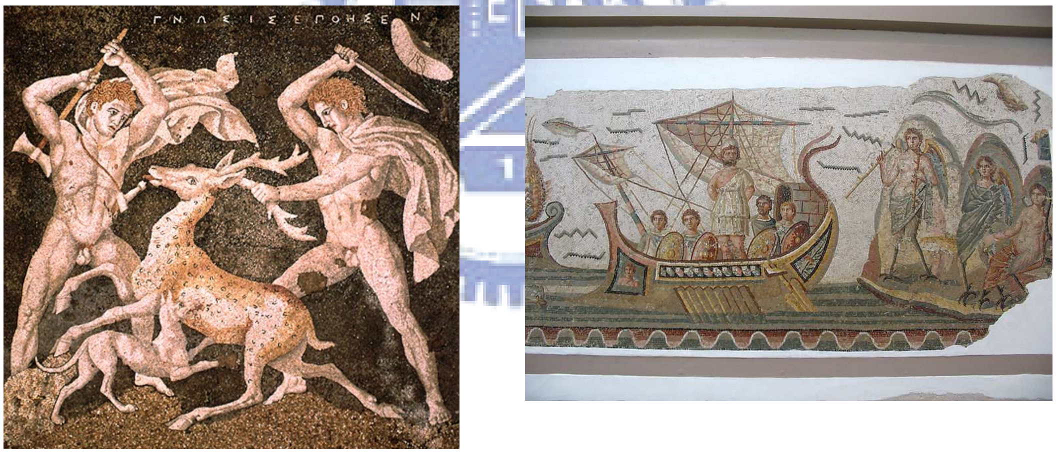
1.4 Thesis Organization

The remainder of this thesis is organized as follows. In Chapter 2, we review the related works of this study. In Chapter 3, the proposed method for creation of tetromino-based mosaic images is described. In Chapters 4, the data hiding method by distinct combinations of tetrominoes in tetromino-based mosaic images is presented. The edge fitting method by edge detection thresholding is also described. Then, the proposed data hiding method by small tetromino color shiftings is described in Chapter 5. We present the proposed removable lossless watermarking method for tetromino-based mosaic images in Chapter 6. In Chapter 7, the proposed method for image steganography by watermarking in tetromino-based mosaic images will be presented. Conclusions of our works and discussions on future works are included in Chapter 8.

Chapter 2
Review of Related Works

2.1 Previous Studies on Mosaic Images

Mosaic image is a common type of decorations composed of numerous small tiles arranged in a certain way. Tile mosaics appeared in Greek and Roman times over 2000 years ago. Two examples are shown in Figure 2.1. They are still widely used today. Creation of mosaic images is a new research topic in recent years.



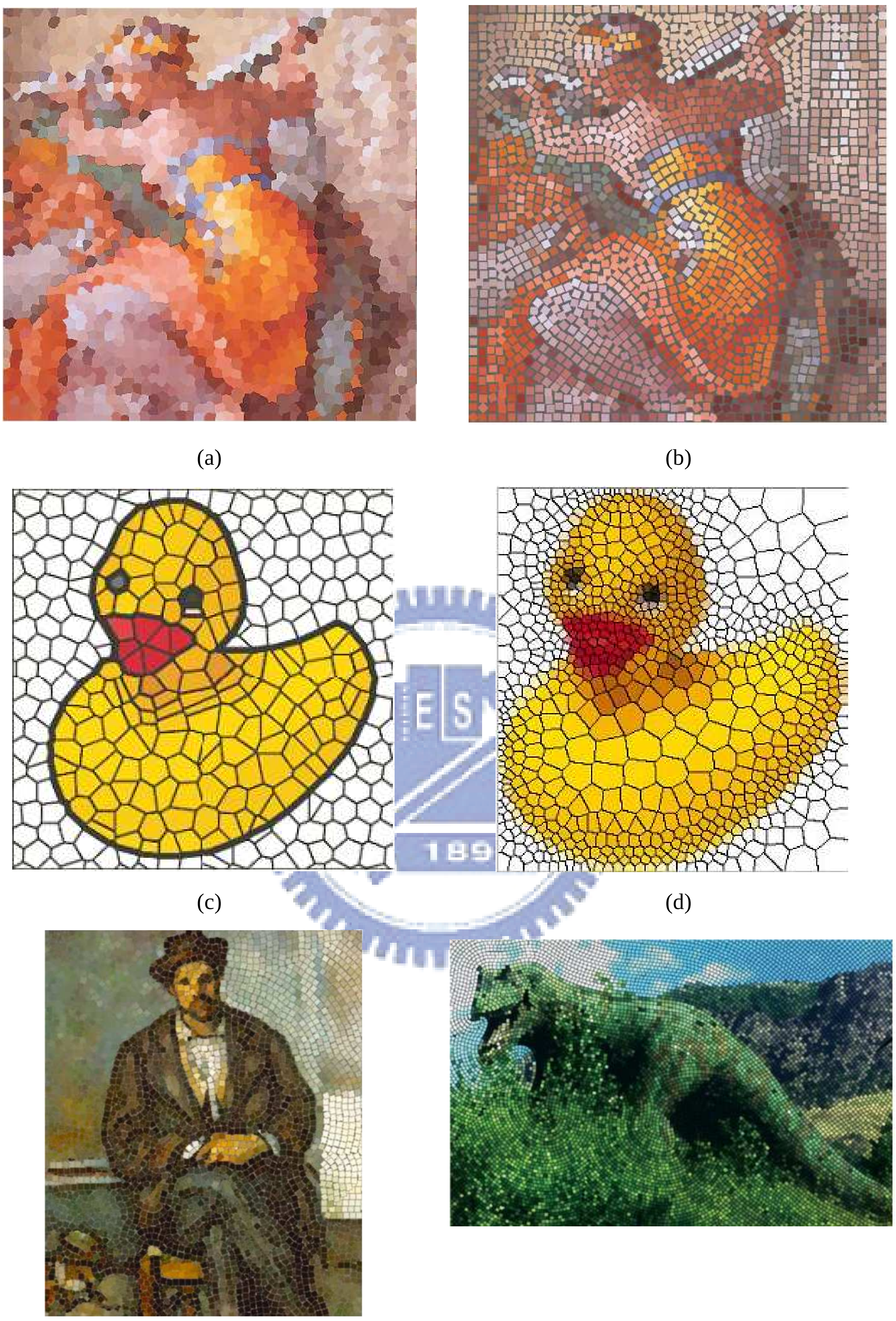
(a) (b)

Figure 2.1 Ancient mosaic images. (a) Detail from the mosaic floor signed Gnosis in the “House of the Abduction of Helen” at Pella, Greek, late 4th century BC, Pella, Archaeological Museum. (b) Roman mosaic of Ulysses at Carthage, Bardo Museum, Tunisia.

Haeberli [6] used Voronoi diagrams to place the Voronoi sites at random and fill each tile with a color which is sampled from the image. However, Haeberli’s method does not attempt to follow the edge features of the image and the tiles all have different shapes. Hausner [7] proposed a method to create tile mosaic images by utilizing centroidal Voronoi diagrams. In Hausner’s method, the method proposed by Hoff [8] was extended to draw a Voronoi diagram efficiently, and Lloyd’s algorithm [9] was utilized to produce centroidal Voronoi diagrams by moving each seed to the centroid of its Voronoi region.

Extending the original idea of Haeberli’s method, Dobashi et al. [10] optimized Voronoi diagrams by using the sites and edges of the Voronoi diagrams to reduce the error between the original image and the resulting image. Faustino and Figueiredo [11] presented a technique similar to Dobashi’s method. The main difference is that Faustino and Figueiredo created tiles whose sizes are adapted to the edge features of the image. This method is achieved by using a centroidal Voronoi diagram with a density function that emphasizes edge features. Di Blasi and Gallo [12] created another type of mosaic images by emphasizing boundaries by placing tiles along edge directions. Elber and Wolberg [13] proposed an advanced approach to rendering traditional mosaic images by recovering free-form feature curves from the image and laying rows of tiles along edge curves. Figure 2.2 shows some examples of the results of the above-mentioned methods.

Photomosaics created by Silvers and Hawley [14] is a different kind of mosaic where a collection of small images is arranged in a rectangular grid. When viewing the mosaic composed by small images from a distance, the small images combine to yield an impressive integrated painting. Kim and Pellacini [15] introduced a new kind of mosaic, called jigsaw image mosaic (JIM), as shown in Figure 2.3, where image tiles of arbitrary shape are used to compose the final picture. Gi et al. [16] proposed a technique of generating and arranging fragmentary piece of colored tiles to simulate colored paper mosaic as shown in Figure 2.4.



(a) (b)

(c) (d)

(e) (f)

Figure 2.2 Mosaic images created by a number of authors. (a) By Haeberli. (b) By Hausner. (c) By Dobashi et al. (d) By Faustino and Figueiredo. (e) By Di Blasi and Gallo. (f) By Elber and Wolberg.

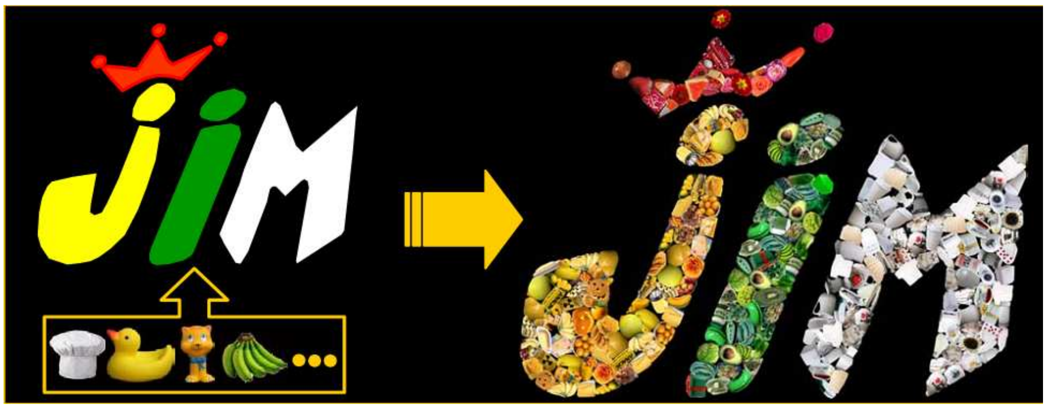


Figure 2.3 Jigsaw image mosaic (JIM) created by Kim and Pellacini.

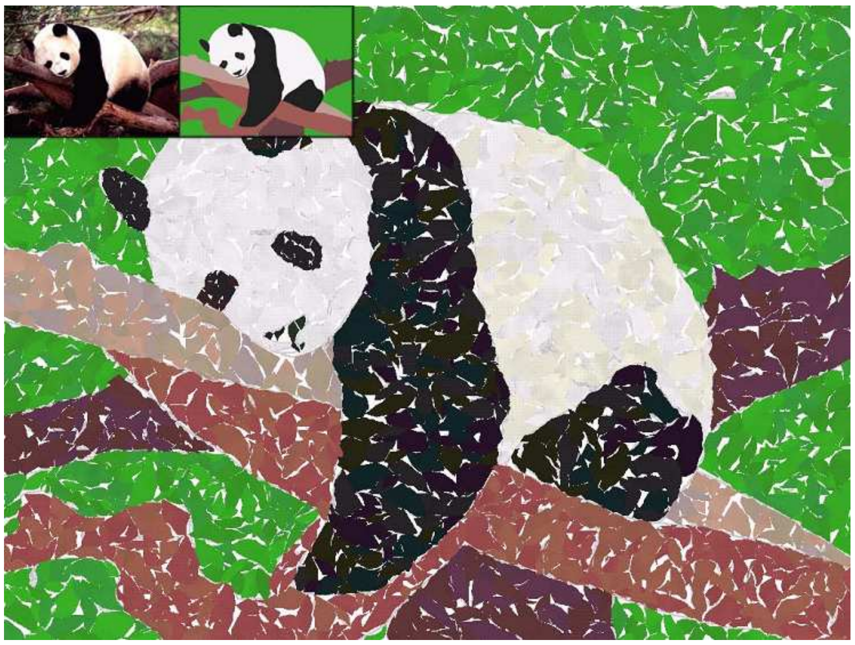


Figure 2.4 Colored paper mosaic created by Gi et al..

2.2 Previous Studies on Information Hiding Techniques

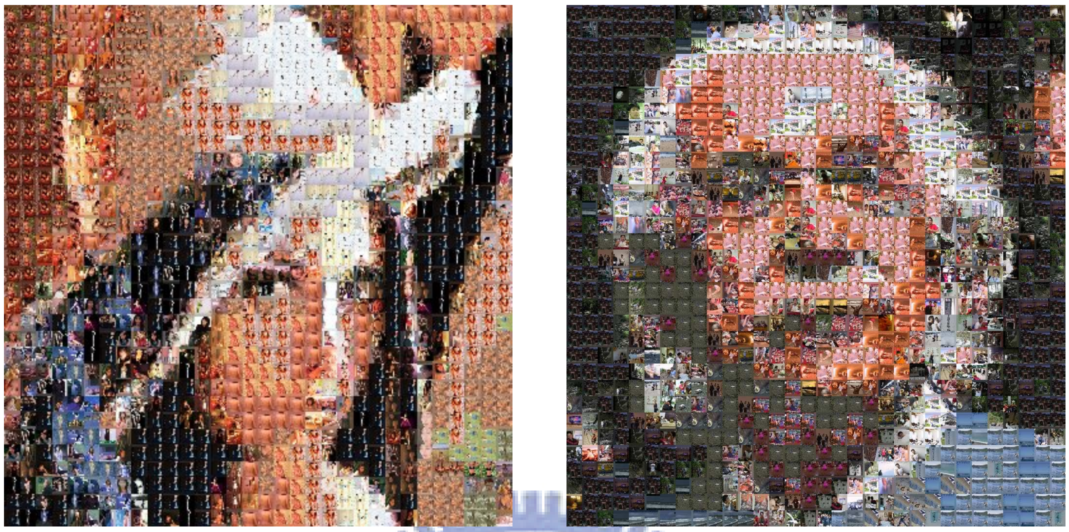
Many information hiding techniques have been proposed to embed data into given media for various purposes, such as covert communication or copyright protection. Most information hiding techniques in images utilize the weaknesses of the human visual system, for example, by changing the least significant bits of the pixels of a cover image to embed information [17]. The information embedded in an image can be used to protect the copyright of the image, verify the authenticity of the image, convey a secret message, and so on.

Information hiding in images is a popular research topic in recent years. Researches on this topic can be classified into three approaches, namely, the spatial-domain approach, the frequency-domain approach, and the combination of them [18]. Ni, et al. [19] presented a novel reversible data hiding algorithm in the spatial domain by utilizing the zero or the minimum point of the histogram and slightly modifying the pixel values to embed data. Cheng and Tsai [20] proposed a DCT-based method for embedding an invisible watermark by adjusting the magnitude relation of certain DCT coefficient pairs in the frequency domain. No matter what domains they belong to, most of these researches are based on pixel-wise or block-wise operations. Generally speaking, information hiding in the frequency domain is more robust than that in the spatial domain, but produces more distortion sometimes.

2.3 Previous Studies on Information Hiding in Art Images

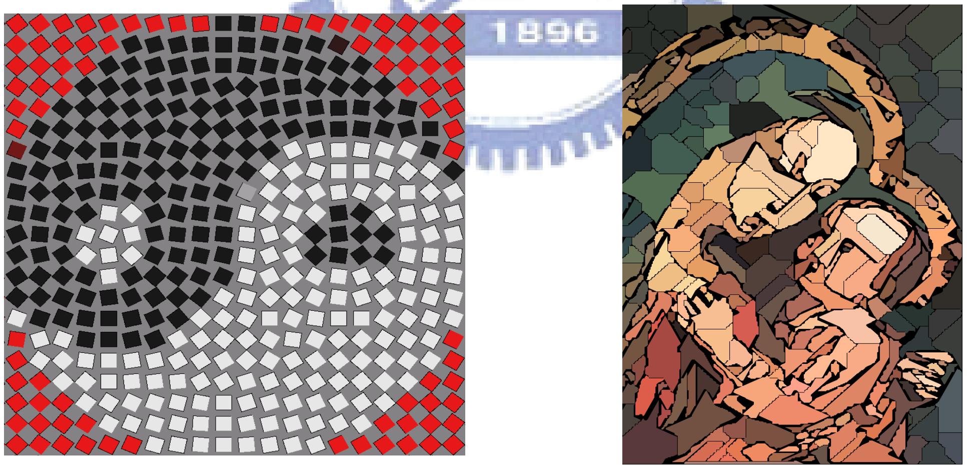
Lin and Tsai [21] proposed two methods to hide information in image mosaics. One is to manipulate the four borders of tile images. The other is to modify the histogram of tile images. Hung et al. [22] proposed two methods to hide information in art images. One is to embed data in the tile mosaic image by modifying the orientations, sizes, and textures of tile mosaic images. The other is to embed data in the stained glass image by cracking glasses slightly. The resulting images created by

Lin and Tsai [21] are shown in Figure 2.5. Two images generated by Hung et al. [22] are shown in Figure 2.6.



(a) (b)

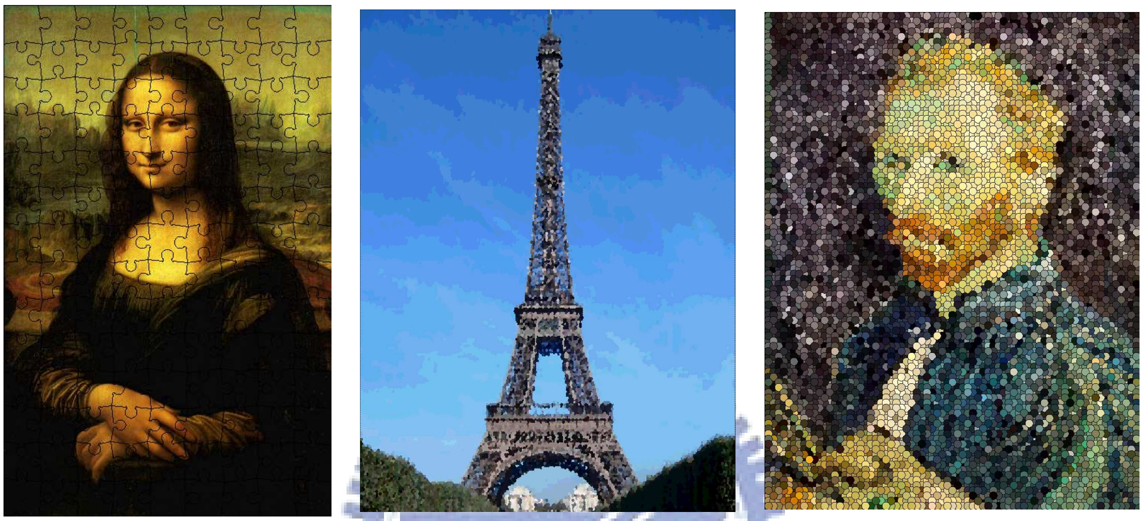
Figure 2.5 Image mosaics created by Lin and Tsai [21]. (a) A mosaic image of Lena. (b) A mosaic image of Albert Einstein.



(a) (b)

Figure 2.6 Art images created by Hung et al. [22]. (a) A tile mosaic image. (b) A stained glass image.

Hsu and Tsai [23] proposed three methods to hide information in art images. One is to embed data in digital puzzle images by modifying the orientations, sizes, and angles of the puzzle image. Another is to embed data in the pointillistic image by varying the RGB values of each color dot of the pointillistic image. The other is to embed data in the circular-dotted image by modifying the drawing order of the circular dots of the circular-dotted image. Some examples of the art images created by Hsu and Tsai [23] are shown in Figure 2.7.



(a) (b) (c)

Figure 2.7 Art images created by Hsu and Tsai [23]. (a) A digital puzzle image. (b) A digital pointillistic image. (c) A digital circular-dotted image.

Wang and Tsai [24] proposed the creations of three types of art images and their applications for information hiding. First, information is hidden in irregular hexagonal-tiled images by adjusting the locations of the two specific vertices and modifying the inner colors of hexagons. For tile-overlapping mosaic images, the information hiding work was implemented by changing the overlapping degrees of adjacent tile images. Last, an information hiding method in variable-sized mosaic images was proposed by changing the sizes of them without creating holes and overlapping areas in the resulting stego-image. Some examples of the art images created by Wang and Tsai [24] are shown in Figure 2.8.

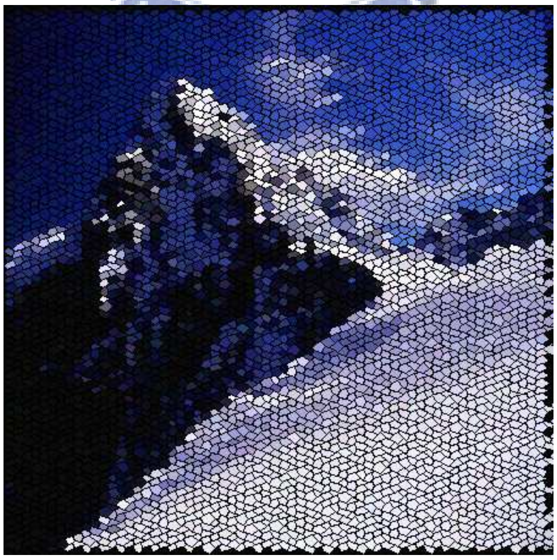
In this study, different from traditional methods of information hiding in images, the combinations of mosaic image creation and information hiding techniques will be proposed. We will focus on tetromino-based mosaic images, a novel type of mosaic image. A method for automatic creation of tetromino-based mosaic images and two data hiding methods for such images are proposed. Furthermore, we propose a technique of removable lossless visible watermarking in tetromino-based mosaic image. Using this watermarking technique, an image steganography method for tetromino-based mosaic images is also proposed in this study.

2.4 Previous Studies on Image Steganography

Steganography, which means "covered writing" in Greek, is the science of communicating a message by embedding it into data files of certain forms. This technique differs from cryptography in that communication is evident but the content of the communication is camouflaged. Unlike data hiding and digital watermarking, the main goal of steganography is to create complete imperceptibility. Many different file formats can be used as cover media for data hiding, but digital image are the most popular file type because they are used everywhere on the Internet.

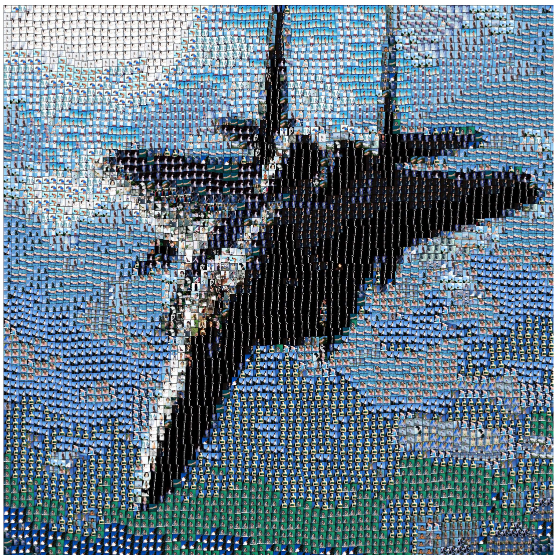
Least significant bit (LSB) insertion is the most common approach to embedding information in a cover image [17]. In this approach, the least significant bit of each byte inside an image is changed to hide at least a bit of the secret message. Lee and Chen [25] proposed an image steganographic model based on variable-sized LSB insertion to maximize the embedding capacity while maintaining image fidelity. Currie and Irvine [26] suggested a modified LSB insertion technique for use in JPEG images by using the compression properties of the frequency domain. Fridrich and Du [27] presented a steganographic technique for palette images by hiding message bits into the parity bits of colors which are close to one another.

Marvel et al. [28] presented a new type of digital image steganography, called *spread spectrum image steganography* (SSIS). The SSIS system proposed by them combines spread spectrum communication, error control coding, and image processing to hide information in images. Fridrich and Goljan [29] presented a new steganographic method for digital images in raster formats. Message bits are embedded in the cover image by adding a weak noise signal with a specified but arbitrary probabilistic distribution. A steganographic technique for embedding secret messages into a gray-valued cover image was proposed by Wu and Tsai [30]. Information is embedded into a cover image by replacing the difference values of the two-pixel blocks of the cover image with similar ones in which bits of embedded data are included.

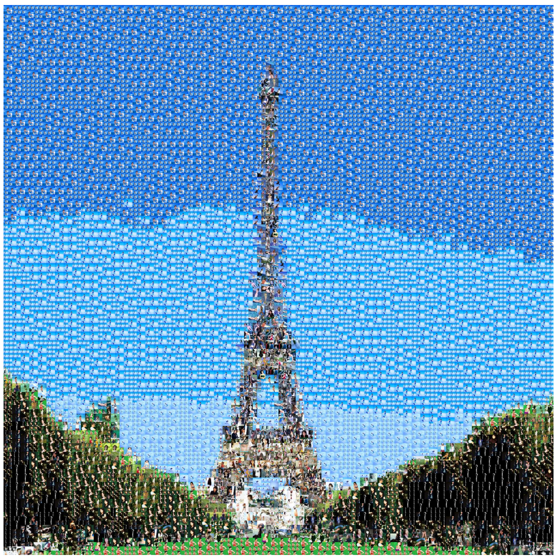


(a)

Figure 2.8 Art images created by Wang and Tsai [24]. (a) An irregular hexagonal-tiled images. (b) A tile-overlapping mosaic images. (c) A variable-sized mosaic images.



(b)



(c)

Figure 2.8 Art images created by Wang and Tsai [24]. (a) An irregular hexagonal-tiled images. (b) A tile-overlapping mosaic images. (c) A variable-sized mosaic images (continued).

Chapter 3
Creation of Tetromino-based Mosaic Images

3.1 Idea of Proposed Method

Lin and Tsai [21] proposed a method to create an image mosaic by arranging a large number of small tile images with each tile image resembling a similar-sized block of a given image. The image mosaic composed of all the tile images looks integrally like the source image when seen from a distance. The idea of the proposed art image creation method in this study was inspired by Lin and Tsai's method to create a new type of mosaic images, called *tetromino-based mosaic image*, as mentioned previously. The idea of tetromino-based mosaic image creation was developed by two main concepts. The first is an integration of art and computer technology. Users can create their own images which look more artistically with special visual effects for decorations. The second concept is that a mosaic image can be used as a fine medium for covert communication. As a result, the proposed method also deals with the problem of copyright protection.

In geometry, tetrominoes can be repeated to fill a plane and combined closely to one another to form a square lock. This is the reason why tetrominoes are also good units for mosaic image creation as proposed in this study. However, tiling a plane with tetrominoes is more difficult than traditional mosaic creation because tetrominoes have more complicated shapes. Consequently, how to combine different kinds of tetrominoes to create a variety of distinct mosaic image is a critical issue. In addition, using tetrominoes as components to generate mosaic images can provide a new style of visual effects for viewers. The tetromino-based mosaic image creation and other related processes are stated in the following sections.

3.2 Review of Traditional Image Mosaic Creation Process

3.2.1 Mosaic Image Creation Process

Two issues, mosaic database construction and similarity measure selection, play important roles in mosaic image creation. The former is the first step of the mosaic creation process to generate tiles images. Tile images are not only the basic components of mosaic images but also the key components determining how mosaic images look like. The latter issue is the design of a good measure for use in the process of choosing the best matching tile image from a mosaic database. In Lin and Tsai's study [21], mosaic images were created according to the following procedures. First, a mosaic database was constructed by selecting a set of tile images and then extracting from their relevant features which depend on what types of tile images are used as the input to the construction process. Next, the image mosaic creation process starts with a source image as an input. The process divides the source image into many small target images based on a given size. A similarity measure is then used to search the best matching tile images of the given size from the mosaic database. Finally, after putting the best-matching tile images together, a mosaic image is generated. A step-by-step process is described in Algorithm 3.1.

Algorithm 3.1. Traditional mosaic creation process.

Input: a source image *I*, a tile image database *DB*, and a given tile size *S*.

Output: a mosaic image *I*′.

Steps:

Step 1. Divide *I* into blocks of target images based on the given size *S*.

Step 2. Extract features from each target image.

Step 3. For each target image, search the best matching tile image from tile image database *DB* according to a similarity measure.

Step 4. Compose all tile images to generate a mosaic image *I*′.

The architecture of traditional mosaic creation is shown in Figure 3.1. The experimental result is show in Figure 3.2.

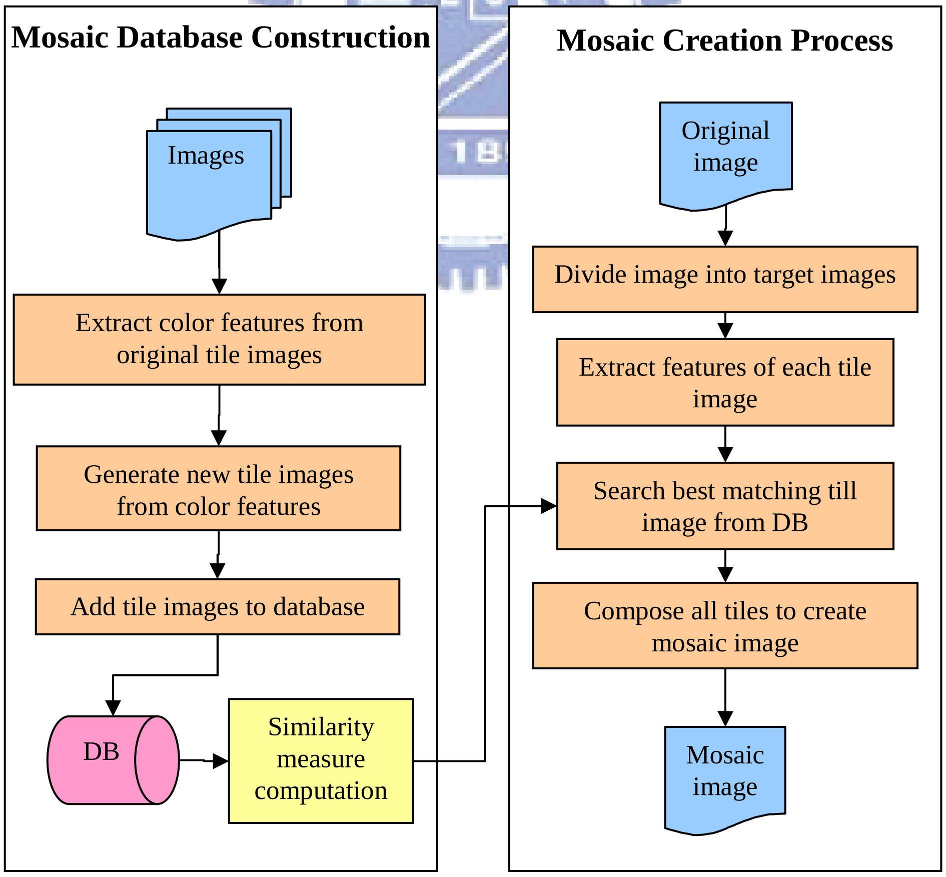

Mosaic Database Construction
Images
Extract color features from original tile images
Generate new tile images from color features
Add tile images to database
DB
Similarity measure computation
Mosaic Creation Process
Original image
Divide image into target images
Extract features of each tile image
Search best matching till image from DB
Compose all tiles to create mosaic image
Mosaic image

Figure 3.1 The system architecture of an image mosaic creation system.

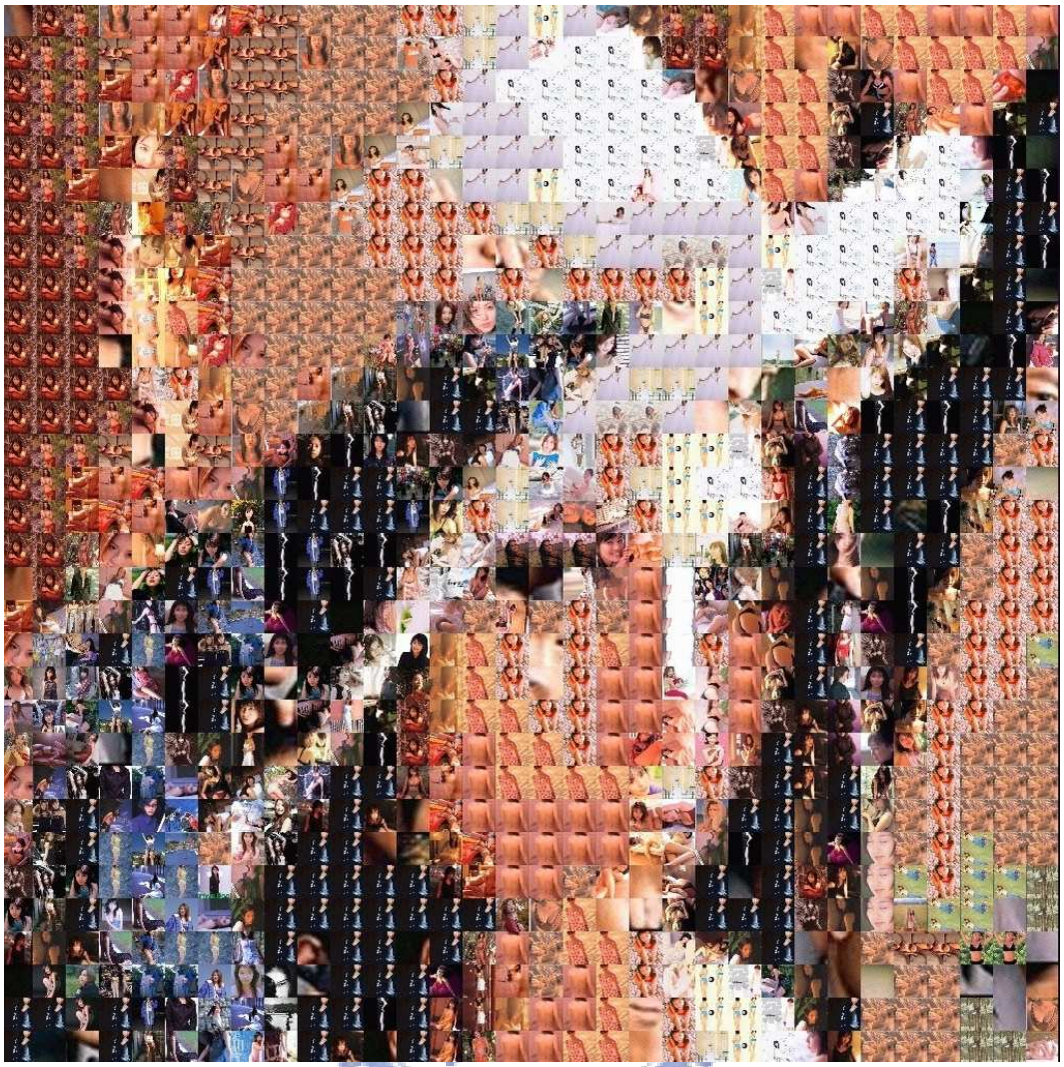


Figure 3.2 An experimental result of image mosaic creation by Lin and Tsai [21].

3.2.2 Tile Image Mosaic Database Construction

For mosaic image creation, we have to construct a database to accelerate the computation of searching the best matching tile image. We accomplish tile image database construction using the following algorithm.

Algorithm 3.2. Tile image database construction.

Input: a set of original tile images, $M = \{I_1, I_2, \ldots, I_n\}$.

Output: a set of new tile images, $N = \{T_1, T_2, .., T_n\}$ and a set of feature vectors $V = \{V_1, V_2, .., V_n\}$, for use as the tile image database.

Steps:

Step 1 Calculate the average color of each original tile image I_i in each of the RGB channels.

Step 2 Represent the average colors as a one-dimension feature vector V_i with three elements R_i, G_i, and B_i for I_i.

Step 3 Generate a new tile image T_i by resizing and cropping each original image I_i to a predefined tile size.

Step 4 Add all T_i and associated V_i to *DB*.

3.2.3 Similarity Measure Computation

In Algorithm 3.1, computation of the similarity measure between a target image and a tile image is mentioned. An input source image is first divided into blocks of small target images and the feature vector of each target image is obtained by calculating three average color values of the target image in the RGB channels. The vector extracted from a target image is taken as an input to the similarity measure computation with another input being the feature vector extracted from a tile image in the mosaic database. A tile image from the mosaic database is considered to be similar to the target image if the resulting value of the similarity measure, say a Euclidean distance between the two feature vectors, is the smallest among all the tile images in the database. The detailed algorithm of similarity measure computation is described as follows.

Algorithm 3.3. Similarity measure computation.

Input: a target image *T*, and a tile image *L* from the database *DB*.

Output: a similarity measure value *D* between *T* and *L*.

Steps:

Step 1. Divide T and L into N parts, where N is a predefined number.

Step 2. Extract color features from each part of T to form a vector V_T with $N\times 3$ elements.

Step 3. Extract color features from each part of L to form a vector V_L with $N\times 3$ elements.

Step 4. Calculate the Euclidean distance between the color vectors V_T and V_L according to the following similarity measure:

$$D = \sqrt{\|V_T - V_L\|^2}\ .$$

3.3 Proposed Tetromino-based Mosaic Image Creation Process

3.3.1 Scheme of Creation Process

In this section, based on above-mentioned traditional mosaic image creation process we show how to create a tetromino-based mosaic image as an algorithm as follows.

Algorithm 3.4. Creation process of tetromino-based mosaic images.

Input: a source image I, a given size s of one square in a tetromino, and a tile image database DB.

Output: a tetromino-based mosaic image I'.

Steps:

Step 1. Construct a tetromino database DB by generating numerous tile images with different tetromino combinations and colors.

Step 2. Divide the source image *I* into many target images with the given size *s*.

Step 3. For each target image *T*, perform the following operations.

3.1 Calculate the distance between *T* and each tile image *L* in the tetromino database *DB* by Algorithm 3.3.

3.2 Choose the tile image L_o in *DB* with the smallest distance as the best matching tile image for *T*.

3.3 Perform a border enhancement process to enhance the visual effect of each tetromino in *T*.

Step 4. For each target image *T* in *I*, replace it with the best matching tile image to create a tetromino-based mosaic image *I′* as output.

In the above steps, a best matching tile image, which is the most similar to the corresponding target image, is found by the similarity measure computation as stated in Algorithm 3.3. The construction of the tetromino database *DB* mentioned in the above algorithm is described in the next section.

3.3.2 Tetromino Database Construction

The tetromino database is used to find the best matching tile image for each target image, as described in the previous algorithm. It is desired to control the number of colors and combinations of the tetrominoes in the database. Before the beginning of tetromino database construction, we have to design an algorithm to enumerate all the possible tetromino combinations in a fixed-sized region. Nineteen types of tetrominoes, as shown in Figure 3.3, are used as inputs for the tetromino database construction algorithm. In this study, the fixed-sized region is a plane of 4×4 grids as shown in Figure 3.4 (a). For example, Figure 3.4 (b) shows one of tetromino combinations in a 4×4 grid. The detail of the enumerating process of tetromino combinations is described in Algorithm 3.4 below. The tetromino database is constructed by Algorithm 3.5 below.

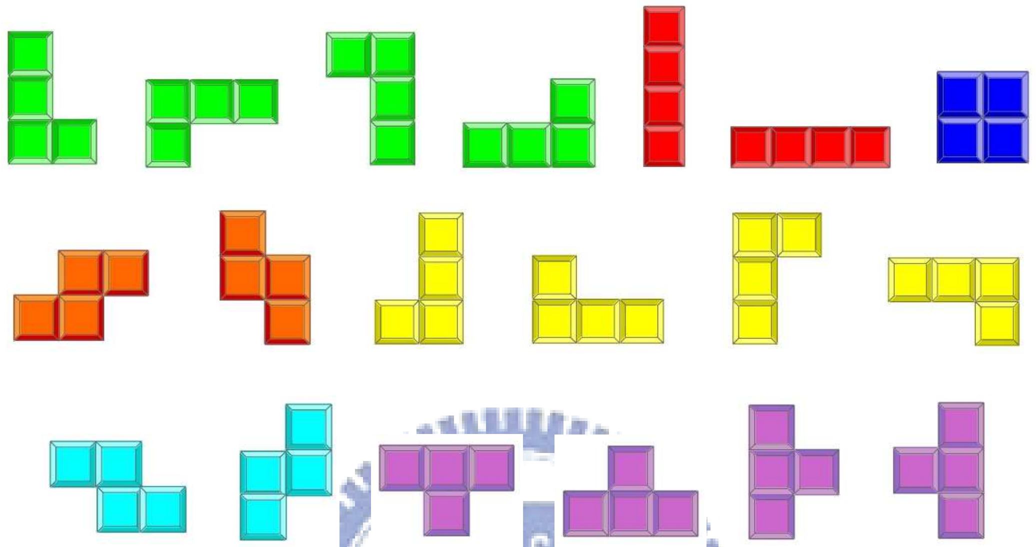


Figure 3.3 Nineteen types of oriented tetrominoes.

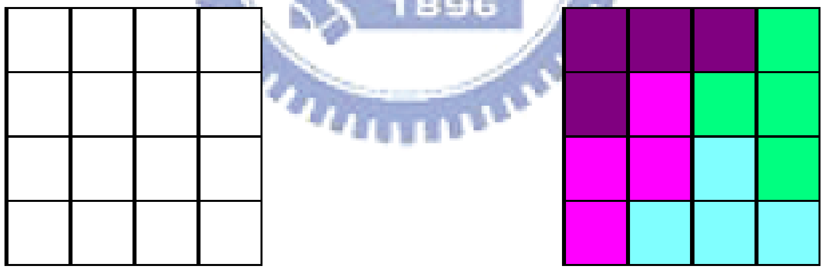


Figure 3.4 Some illustrations of inputs in Algorithm 3.5. (a) A plane of 4×4 grids. (b) A tetromino combination in a plane of 4×4 grids.

Algorithm 3.5. Process for enumerating tetromino combinations.

Input: 19 types of tetrominoes E and a plane P of 4×4 grids.

Output: a set of all possible tetromino combinations $T = \{T_1, T_2, .., T_n\}$ in P.

Steps:

Step 1. Create a four-level tree R with a root r.

```
Step 2. In first level, for each tetromino type $X$ in $E$, perform the following operations.
    2.1 For each position which can be used to place $X$, perform the following operations.
        2.1.1 Generate a child node $C_r$ of root $r$.
        2.1.2 If the filled tetromino crosses the boundary of $P$, delete $C_r$ and go back to 2.1.
        2.1.3 Record the tetromino type $X$ and the position of $X$ in $C_k$
Step 3. For each node of level $L_i$ , where $2 \geqq i \geqq 4$, perform the following operations.
    3.1 For each node $D$ in level $i - 1$, perform the following operations.
        3.1.1 Choose one type from $E$ and denote it by $Y$.
        3.1.2 For each position which can be used to place $Y$, perform the following operations.
            3.1.2.1 Generate a child node $C_k$ of node $D$.
            3.1.2.2 If the filled tetromino $Y$ crosses the boundary of $P$, delete the child node $C_k$ and go back to Step 3.1.2.
            3.1.2.3 If the filled tetromino $Y$ overlaps with any existing tetromino, delete child node $C_k$ and go back to Step 3.1.2.
            3.1.2.4 If the filled tetromino combination $Y$ already exists in the tree, delete child node $C_k$ and go back to Step 3.1.2.
            3.1.2.5 Record the tetromino type and the position of $Y$ in $C_k$.
Step 4. Through a tree traversal, search all paths from all leaves in level 4 to the root.
Step 5. For each path $P_i$ in the tree, form a tetromino combination $T_i$ by extracting data from each node of $P_i$.
```

Algorithm 3.6. Tetromino database construction.

Input: 19 types of tetrominoes, and a fixed number N of colors.

Output: a tetromino database DB with all possible tile images.

Steps:

Step 1. Generate a set of all possible tetromino combinations $T = \{T_1, T_2, .., T_n\}$ in a plane of 4×4 grids by Algorithm 3.5.

Step 2. Generate a set of N distinct colors, $C = \{C_1, C_2, .., C_N\}$, in the RGB color space uniformly.

Step 3. Create all possible tile images by generating all possible combinations of T_i in T and C_i in C.

3.3.3 Visual Effect Improvement by Border Enhancement

Through the above-proposed algorithms, a basic tetromino-based mosaic image is created. However, a problem occurs in our creation process. If two tetrominoes which are close to one another have the same or similar color, the edge between these two tetrominoes may be hard for the viewers to recognize. It is desire to enhance the boundary of tetrominoes in tetromino-based mosaic creation. Furthermore, we also hope to make each piece of tetrominoes look more three-dimensional. As a result, two visual effects, lightening and shading borders, are created as a post-processing step in this study after the tetromino-based mosaic image creation process, which we call border enhancement. An algorithm for this purpose is described as follows.

Algorithm 3.7. Border enhancement.

Input: a plane Y of 4×4 grids with a tetromino combination P.

Output: Y with lightening and shading effects P'.

Steps:

Step 1. For each tetrominoes *t* in *P*, transform the color of the tetromino from the RGB model to the HSL model

Step 2. For each tetrominoes *t* in *P*, check its edge type according to the follow steps.

2.1 If it is a top or left edge in *t*, then lighten the edge by increasing the L channel value of the color at the borders.

2.2 If it is a bottom or right edge in *t*, then shade the edge by decreasing the L channel value of the color at the borders.

3.4 Experimental Results

Some tetromino-based mosaic images generated by the proposed creation process (Algorithm 3.4) are shown in this section. All the tetromino-based mosaic images in our experimental results were created by the use of a tetromino database with 125 kinds of colors and 117 types of tetromino combinations constructed in this study. Figures 3.6, 3.7 and 3.8 show the results of using different border enhancements. Figures 3.9, 3.10 and 3.11 show some experimental results of using different kinds of original images obtained from the proposed mosaic image creation process.

3.5 Discussions and Summary

In this chapter, we reviewed a traditional mosaic image creation process and proposed accordingly an idea of creating a novel type of mosaic image composed by tetrominoes. Then, a creation process of tetromino-based mosaic images is proposed.

In addition, an algorithm has been proposed to generate a tetromino database with different colors and tiling styles. Finally, we also proposed a border enhancement process for improving edge effects in tetromino-based mosaic images. Some experimental results are presented to show the effects of the proposed algorithms. Colors and tetromino combination are the most important characteristics in tetromino-based mosaic image creation. With these two features, we will propose two data hiding methods in the two following chapters.

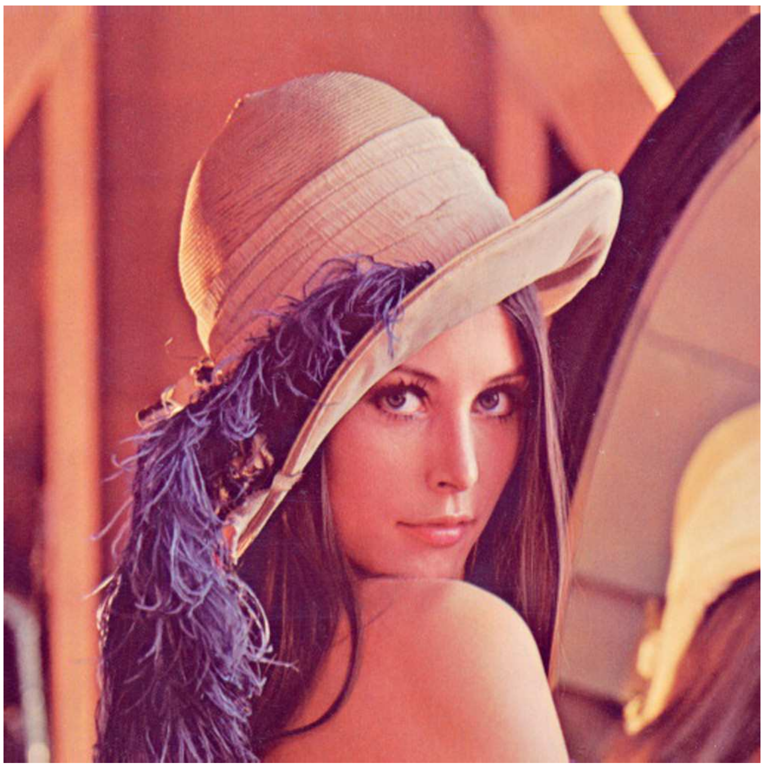
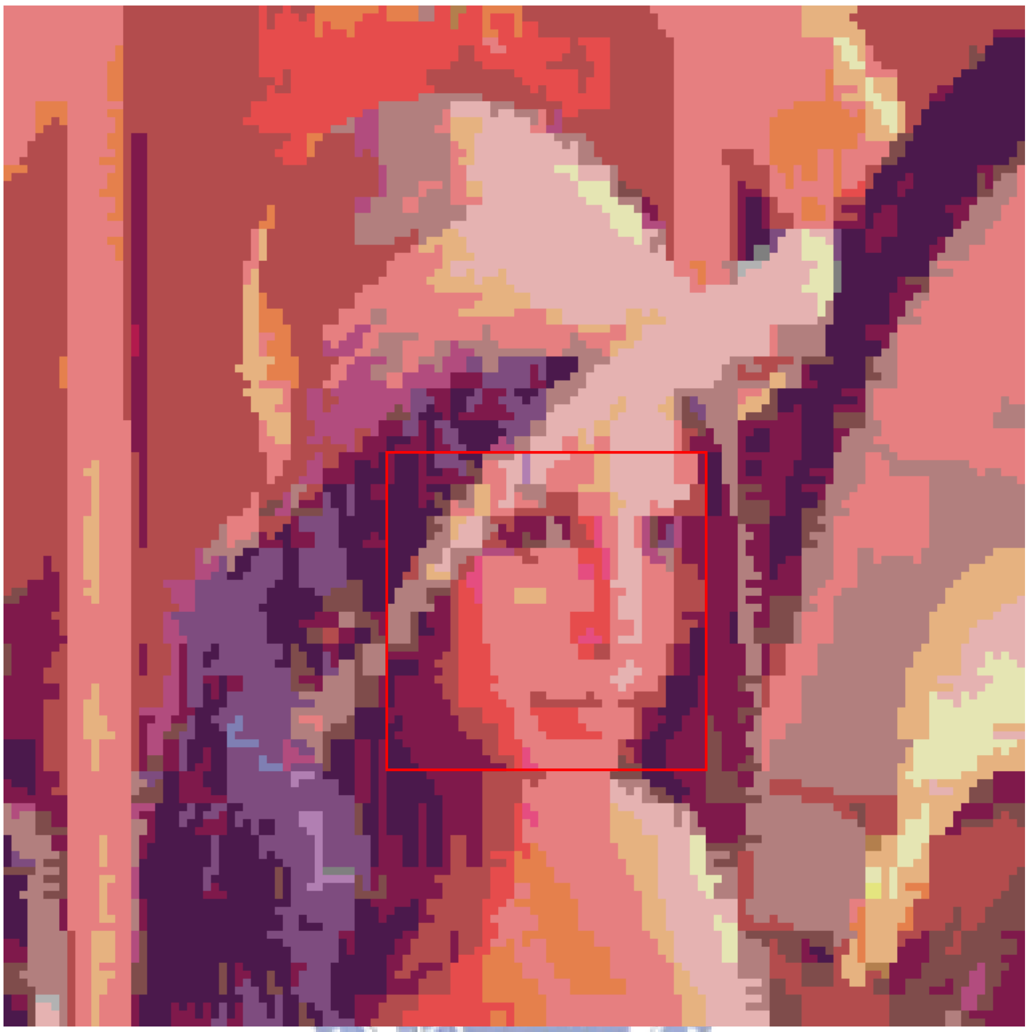
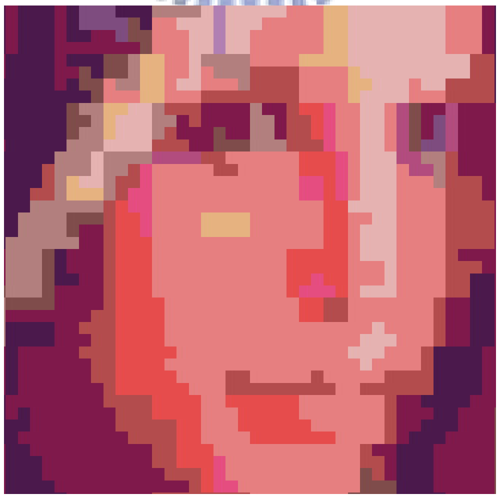


Figure 3.5 An original image.

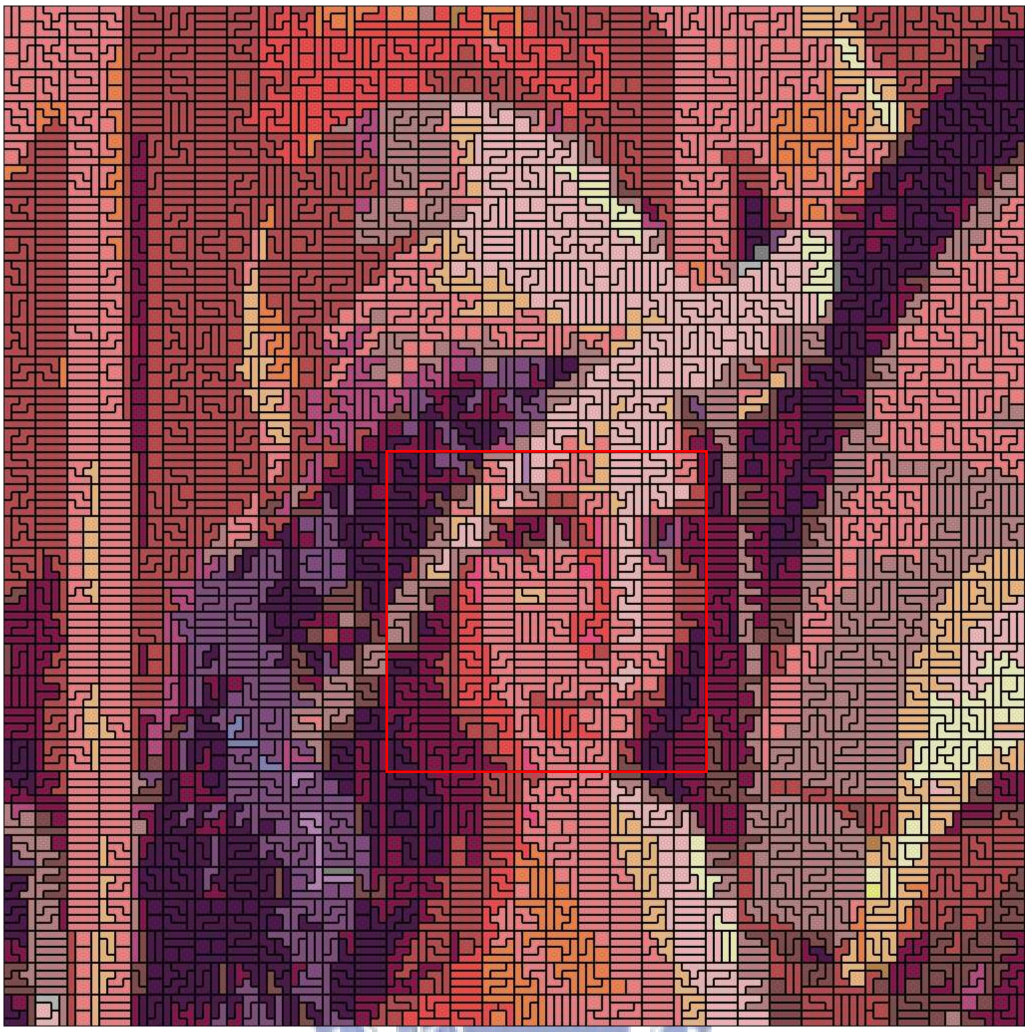


(a)

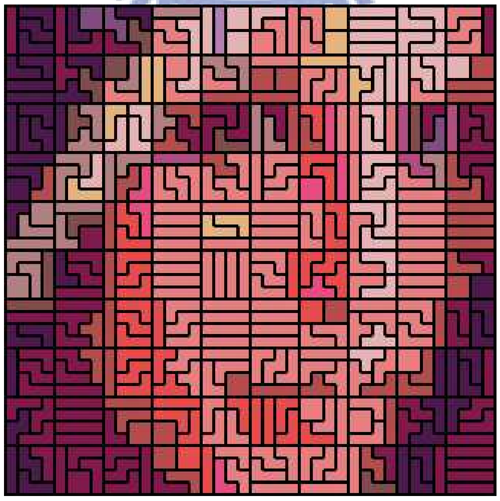


(b)

Figure 3.6 Some of experimental results of the proposed tetromino-based mosaic creation. (a) A tetromino-based mosaic image without border effects created from Figure 3.5. (b) A partial image of the red region in (a).

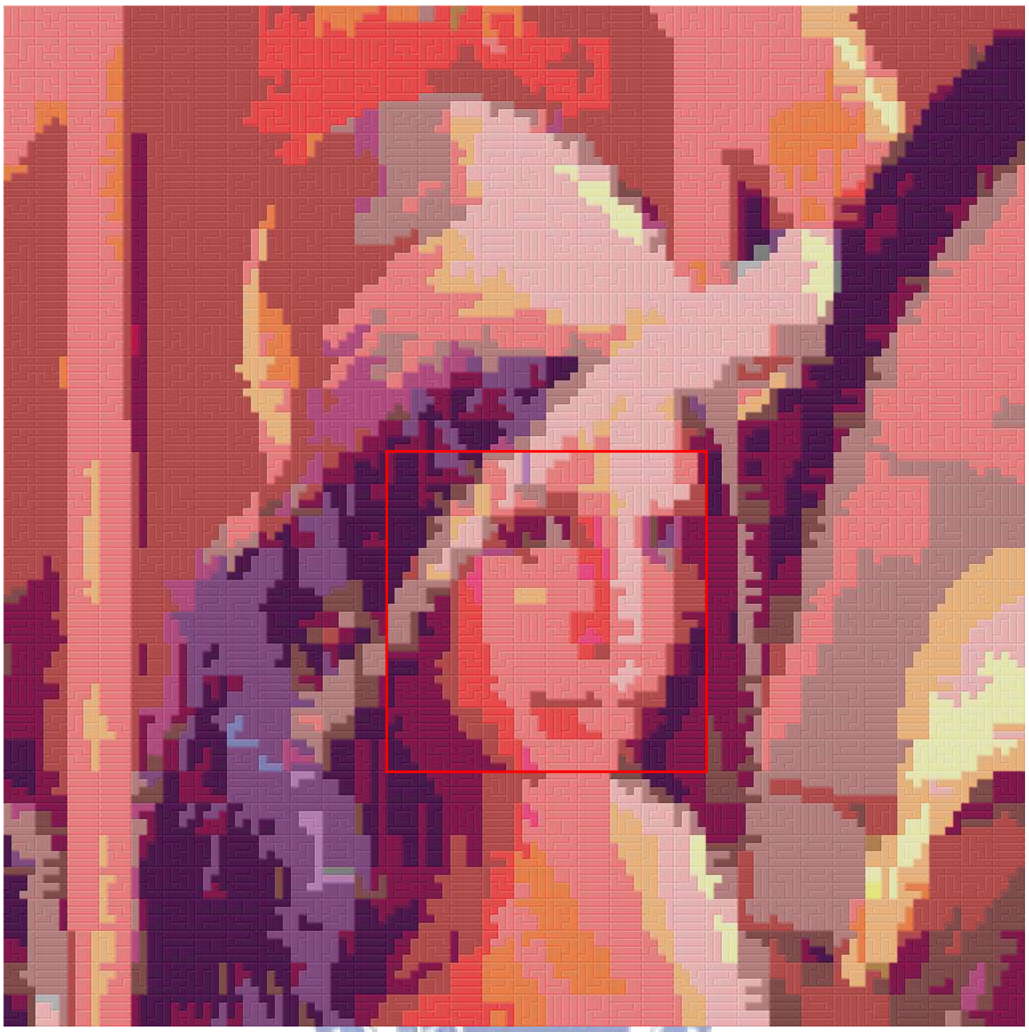


(a)



(b)

Figure 3.7 Some of experimental results of the proposed tetromino-based mosaic creation. (a) A tetromino-based mosaic image with dark border effects created from Figure 3.5. (b) A partial image of the red region in (a).


(a)

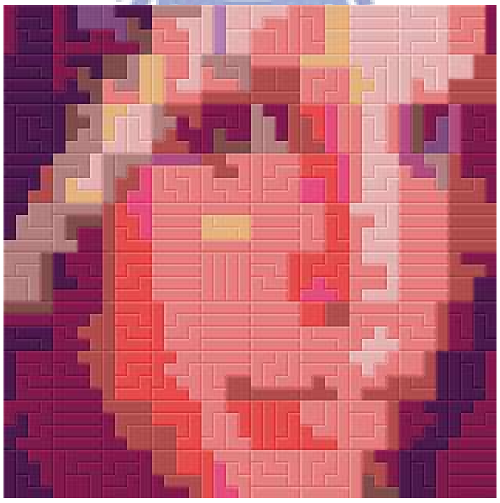
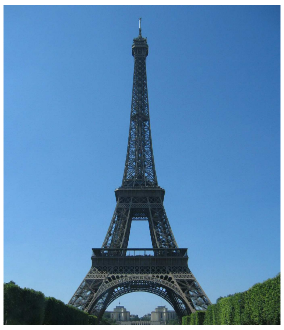
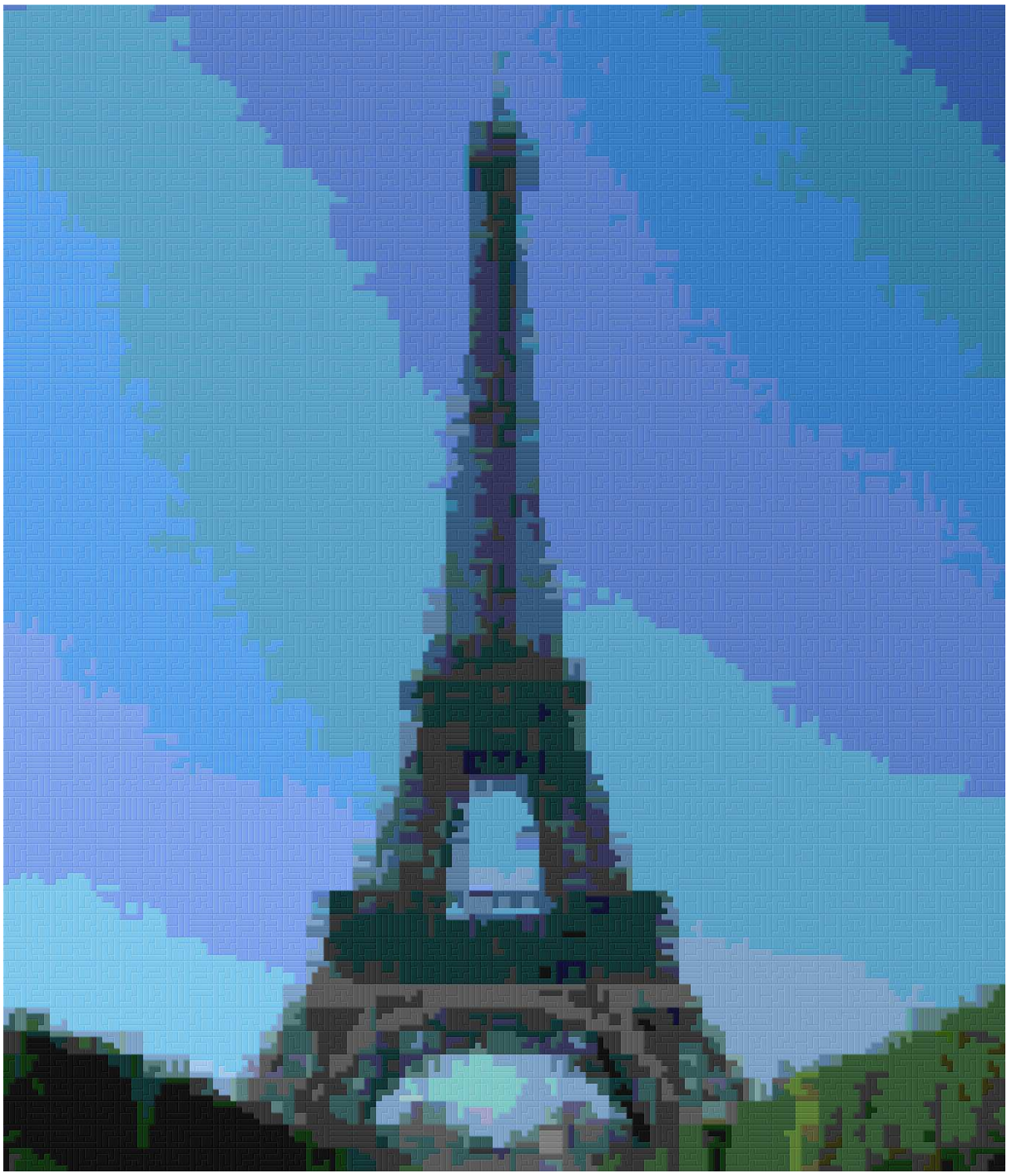

(b)

Figure 3.8 Some of experimental results of the proposed tetromino-based mosaic creation. (a) A tetromino-based mosaic image with lightening and shading border effects created from Figure 3.5. (b) A partial image of the red region in (a).

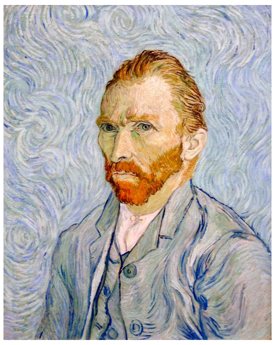


(a)

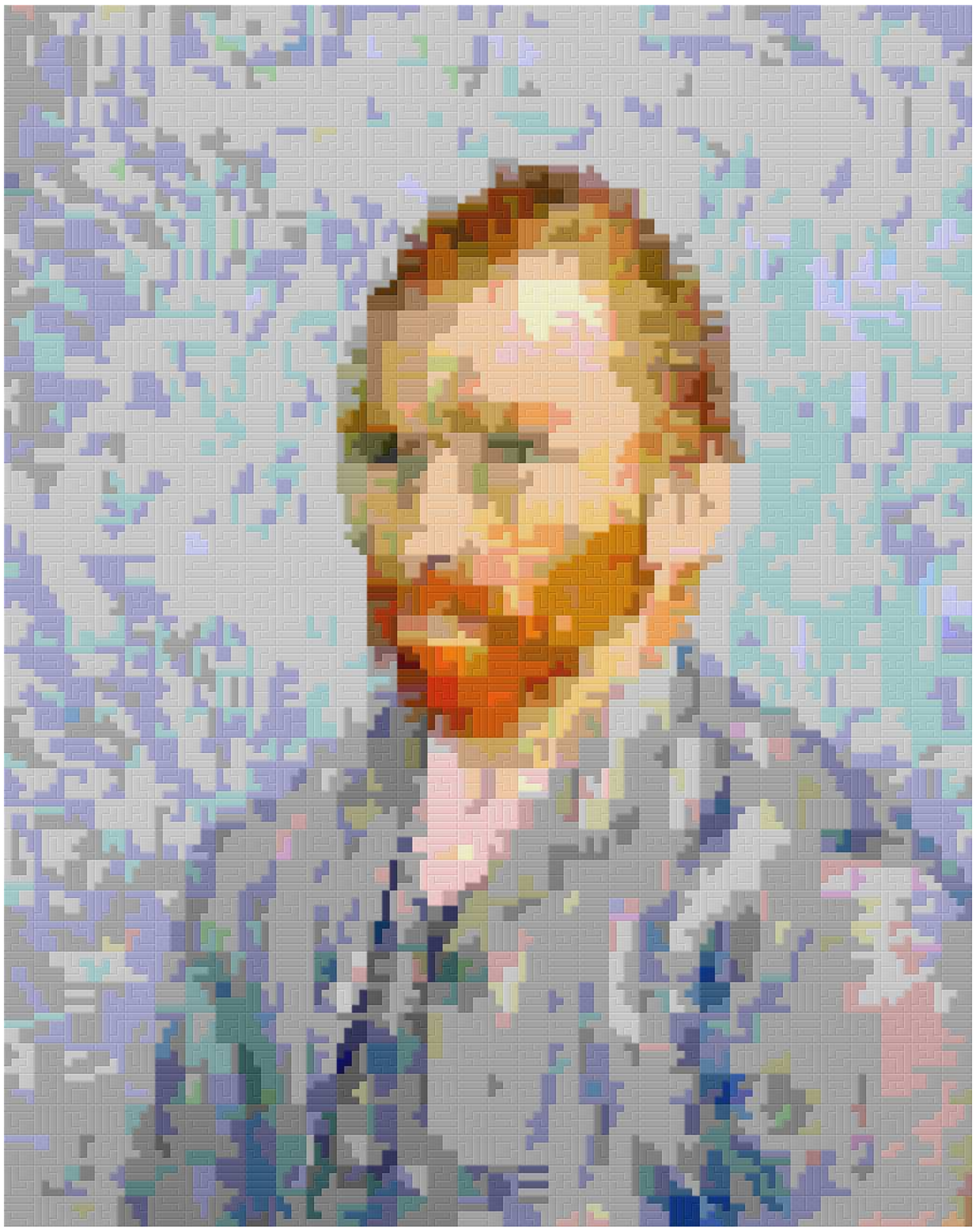


(b)

Figure 3.9 Some of experimental results of the proposed tetromino-based mosaic creation. (a) An image of Eiffel Tower. (b) A tetromino-based mosaic image created from (a)

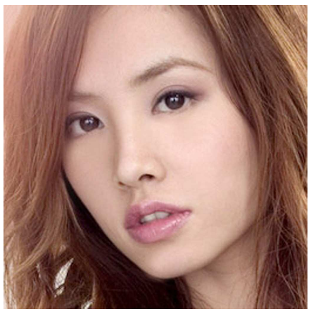


(a)

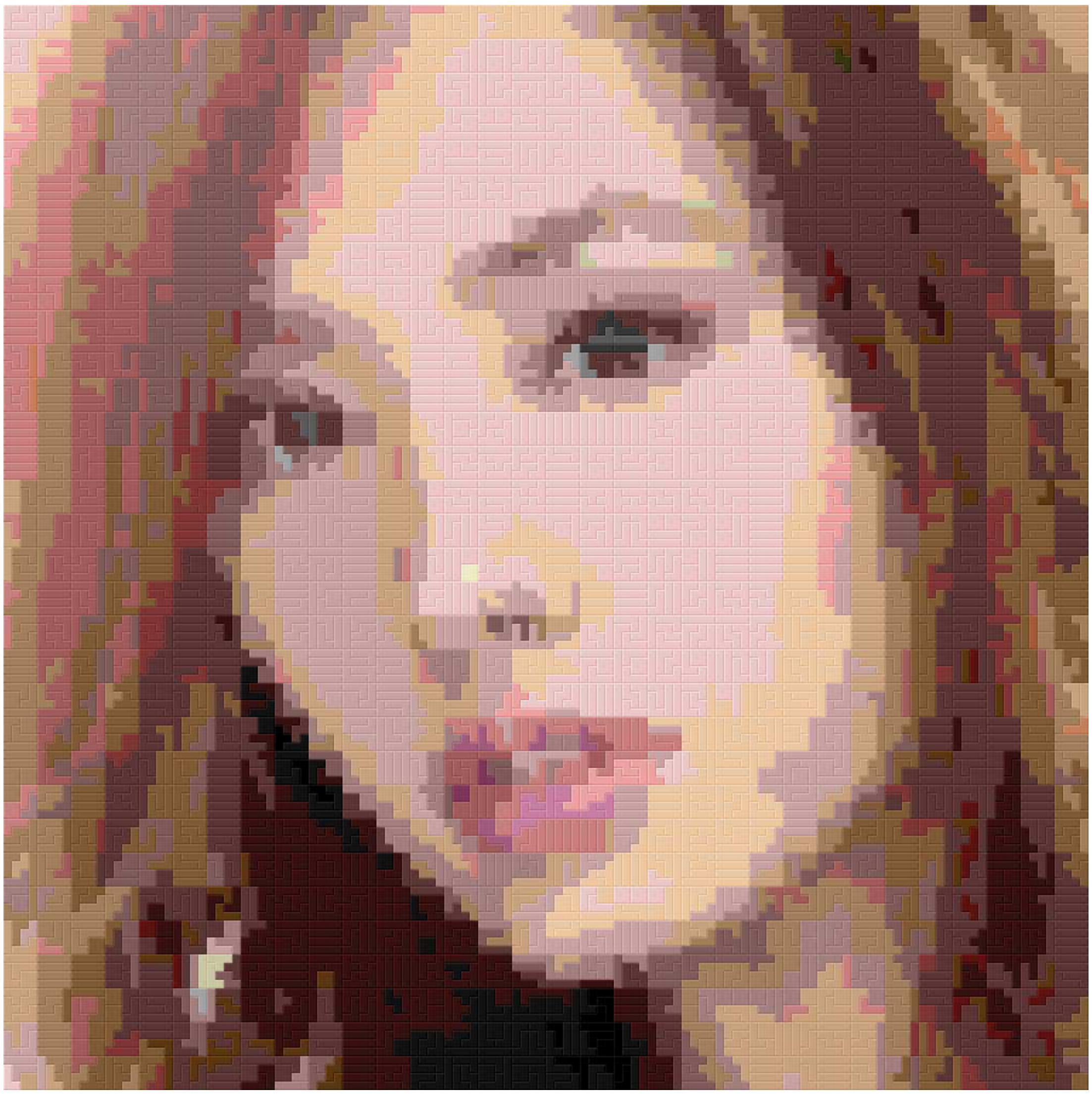


(b)

Figure 3.10 Some of experimental results of the proposed tetromino-based mosaic creation. (a) An self-portrait of van Gogh. (b) A tetromino-based mosaic image created from (a).



(a)



(b)

Figure 3.11 Some of experimental results of the proposed tetromino-based mosaic creation. (a) An image of Jolin, a singer. (b) A tetromino-based mosaic image created from (a).

Chapter 4
Data Hiding in Tetromino-based Mosaic Images by Distinct Combinations of Tetrominoes

4.1 Idea of Proposed Method

As mentioned in Chapter 3, we create a new type of mosaic images in this study, named tetromino-based mosaic images. This type of mosaic image is generated by the use of distinct tetromino combinations. Combinations of tetrominoes can be encoded to hide data. However, using this way to hide data may damage the tetromino-based mosaic images which we create. Therefore, we propose to use edge fitting to maintain visual qualities of tetromino-based mosaic images by sacrificing a portion of capacity of data hiding. In this chapter, we will describe the proposed data hiding method using distinct combinations of tetrominoes and an edge fitting technique based on edge detection.

4.2 Proposed Data Hiding Method by Distinct Combinations of Tetrominoes

4.2.1 Scheme of Propose Method

The main concept of the proposed method for data hiding using tetromino-based mosaic images is to encode tetromino combinations during tetromino-based mosaic image creation. The creation can be regarded as a two-level hierarchical operation. The lower level of the operation is to fill four tetrominoes into a block of 4×4 grids for tile image generation. And the upper level is to put all tile images together to create a tetromino-based mosaic image. As a result, we can use the lower level of this two-level hierarchical image creation to hide data. Through the tetromino database construction described in Chapter 3, 117 kinds of tetromino combinations can be found by an enumerating process, as mentioned previously. In theory, we can only embed six bits in a block of 4×4 grids by fundamental binary encoding because the largest power of 2 smaller than 117 is 2^6. To increase the capacity of our data hiding technique, a transformation process between a binary digit system and a decimal digit one are proposed. At first, the given data are transformed into a binary digit sequence. Subsequently, we transform the binary number sequence into a decimal digit sequence. Then, we encode 100 tetromino combinations for every two digits of the decimal digit sequence for data hiding. Because the capacity of the maximum integer data type in C/C++ is 64 binary bits long, which equals 20 decimal digits, we can embed 64×(2/20)=6.4 bits in a block of 4×4 grids by the above-mentioned process. The detail of the process is described as algorithm below.

Algorithm 4.1. Data transformation from binary to decimal.

Input: message data *D* and a secret key *K*.

Output: a randomized decimal digit sequence S_r to be used in the data hiding process.

Steps:

Step 1. Transform *D* into a binary digit sequence $S = \{S_1, S_2, ..., S_n\}$.

Step 2. Create an integer set *A* of integers in a form of the maximum integer data type.

Step 3. For every 20-bit segment S_t of S, transform S_t into a decimal number A_i and store it in A in order.

Step 4. Extract each decimal digit of every A_i in A to form a decimal digit array S_d.

Step 5. Use K to randomize S_d to get a randomized decimal digit sequence S_r.

4.2.2 Data Hiding Process

In this section, we combine a tetromino-based mosaic image creation and a data embedding technique to carry out the function of data hiding. Algorithm 4.2 shows the details of our method.

Algorithm 4.2. Data hiding in a tetromino-based mosaic image.

Input: a source image I, a tetromino database DB, a secret message M, and a secret key K.

Output: a stego-tetromino-based mosaic image I'.

Steps:

Step 1. Divide I into a set of 4×4 blocks, $B = \{B_1, B_2, .., B_n\}$.

Step 2. Use the secret key K to transform M into a randomized decimal digit sequence S_r by Algorithm 4.1.

Step 3. Divide S_r into several substrings $\{S_1, S_2, .., S_n\}$, each having a length of two digits.

Step 4. For each B_i in B and each S_i in S, perform the following operations.

4.1. Transform S_i to a decimal number d.

4.2. Choose a tile image L from DB according to d.

4.3. Perform a border enhancement process (Algorithm 3.6) on L to generate L'.

Step 5. For each block B_i in I, replace it with corresponding L' generated in the last step to create a stego-tetromino-based mosaic image I' as output.

4.2.3 Data Extraction Process

Before a data extraction process, we have to figure out the tetromino combination of each 4×4 block in a stego-tetromino-based mosaic image. Because a stego-image is created by a two-level hierarchical mosaic creation process, we also propose a scheme for recognizing two-level tetromino combinations in stego-tetromino-based mosaic images. In the lower level of recognition, we label each 4×4 grid with an index to identify its edge type at first. Fourteen edge types and their relative indexes are shown in Figure 4.1 for tetromino combination recognition. In the upper level of recognition, we represent indexes of edge types as a 4×4 integer array in each block of 4×4 grids. Each array entry records an index according to the edge type of the corresponding position in a block of 4×4 grids. For instance, Figure 4.2(a) shows a combination of tetrominoes. A corresponding edge type array is show in Figure 4.2(b). The detailed algorithm of tetromino combination recognition is stated in Algorithm 4.3. Using this algorithm, we propose a data extraction process, as described in Algorithm 4.4, for a stego-tetromino-based mosaic images.

Algorithm 4.3. Tetromino combination recognition.

Input: a block *B* of 4×4 grids, a corresponding table *T* of edge types and indexes (like Figure 4.1), and a tetromino database *DB.*

Output: a number *m* to recognize one of tetromino combinations from *DB.*

Steps:

Step 1. Create an array *A* to record the edge types of *B*.

Step 2. For each grid *G* in *B*, perform the following operations.

2.1.Creation a one-dimensional array *R*.

2.2.For each edge *e* of *G*, perform the following steps.

2.2.1. If the pixel value P_e of *e* is equal to the pixel value P_c of the center of *G*, record *false* in *R*.

2.2.2. If P_e is equal to P_c, record *true* in *R*.

2.3.Use *R* to search the edge type of *G* from *T* and obtain an index *D*.

2.4.Record *D* in the corresponding position of *A*.

Step 3. Search *A* from *DB* to acquire the corresponding number *m* for recognizing the combination of tetrominoes.

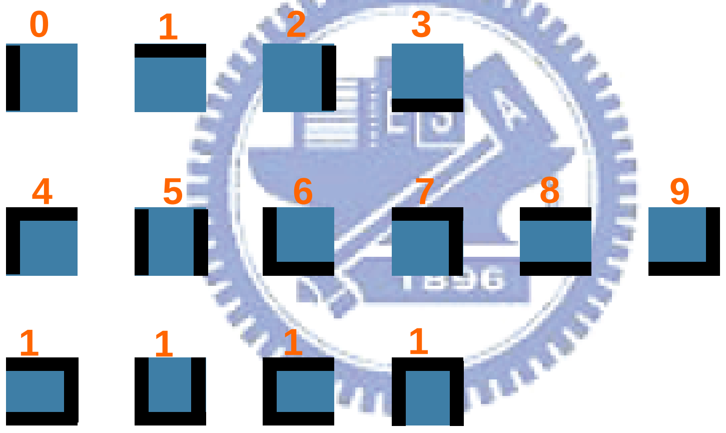
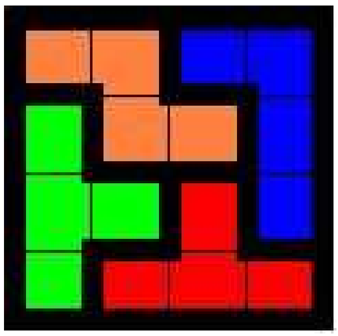
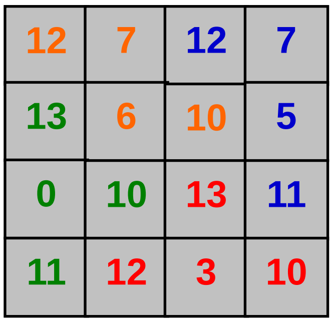


Figure 4.1 Fourteen edge types in a grid of 4×4 grids





12	7	12	7
13	6	10	5
0	10	13	11
11	12	3	10

(a) (b)

Figure 4.2 Some illustrations of tetromino combination recognition. (a) A tetromino combination of 4×4 grids. (b) A corresponding edge type array of (a).

Algorithm 4.4. Data extraction from a tetromino-based mosaic image.

Input: a stego-tetromino-based mosaic image *I*, a tetromino database *DB*, and a secret key *K*.

Output: a secret message *M*.

Steps:

Step 1. Create a decimal digit sequence $R = \{R_1, R_2, .., R_n\}$.

Step 2. Divide *I* into several blocks of 4×4 grids, $B = \{B_1, B_2, .., B_n\}$.

Step 3. For each block B_i in *B*, perform the following operations.

3.1.Obtain a number *m* by recognizing the tetromino combination B_i from *DB* by Algorithm 4.3.

3.2.Transform *m* to two decimal digits *b*.

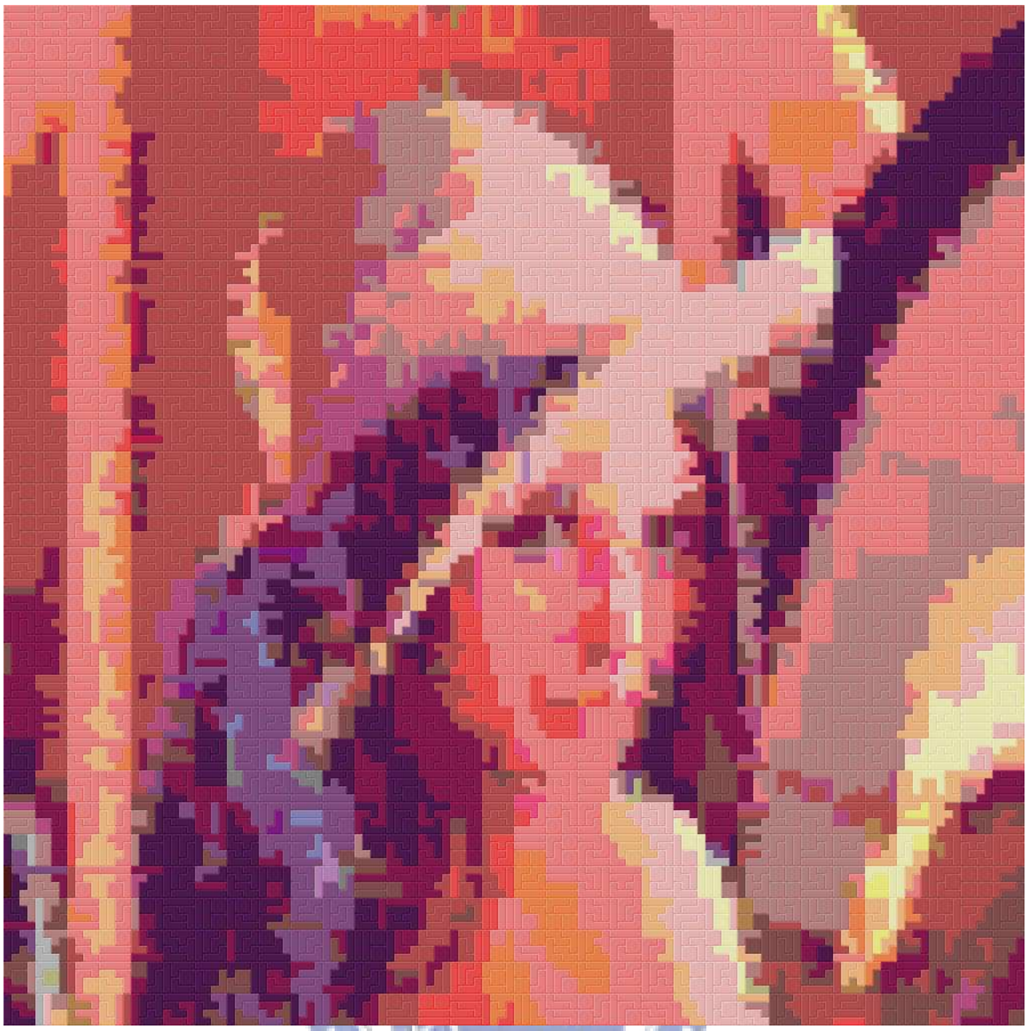
3.3.Record *b* in *R*.

Step 4. Use *K* to generate a random number sequence *r* and use *r* to recover *R* into a decimal digit sequence R' in its original order.

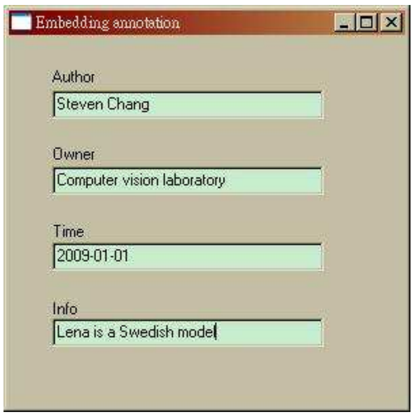
Step 5. Transform R' into a binary digit sequence to form a secret message *M* as output.

4.2.4 Experimental Results

Figure 3.4 shows some experimental results of applying the proposed data hiding method in a tetromino-based mosaic image. Figure 4.3(a) is a tetromino-based mosaic image with a secret message embedded with the proposed data embedding process - Algorithm 4.2. Figure 4.3(b) is the secret message which is extracted from 3.4(a) by the proposed data extraction process – Algorithm 4.4.



(a)



Author

Steven Chang

Owner

Computer vision laboratory

Time

2009-01-01

Info

Lena is a Swedish model

(b)

Figure 4.3 An experimental result. (a) A stego-tetromino-based mosaic image with embedded data. (b) The message extracted from (a).

4.3 Proposed Edge Fitting Method by Adjusting Combinations of Tetrominoes Using Information Obtained by Edge Detection

4.3.1 Scheme of Creation Process

Using tetromino combinations as codes of the proposed data hiding method may cause a main disadvantage. Because of utilizing distinct combinations of tetrominoes to hide data, we can not find the best matching tile image for each target image during tetromino-based mosaic image creation. This fact can cause some noise in tetromino-based mosaic images. Figure 4.4 shows a comparison between an image created by a mosaic creation process in Chapter 3 and an image created by our data hiding method in Section 4.2. The blue and green regions show the differences caused by noise. To solve this problem, an edge fitting method by adjusting combinations of tetrominoes using the information obtained by edge detection is proposed in this section. To improve the visual quality of stego-tetromino-based mosaic images, we sacrifice a part of data capacity of our data hiding technique to enhance edge effects. For each block of 4×4 grids, we perform an edge detection operation which was proposed by Canny [31] to obtain edge values and directions at the beginning. According this edge information, an edge fitting method is proposed to enhance edge effects. Two detailed algorithms of edge fitting method are described in Section 4.3.2.

4.3.2 Detailed Algorithms of Edge Fitting Method

A modified version of the Canny edge detection operation used in our edge fitting method is described in Algorithm 4.5. Then a combinational algorithm of edge fitting and data hiding is described in Algorithm 4.6.

Algorithm 4.5. Canny edge detection.

Input: a block of 4×4 grids, *B*.

Output: a set of four average edge values, $E=\{E_h, E_v, E_e, E_w\}$, of *B* for the four types of edge directions, respectively.

Steps:

Step 1. .Smooth *B* to generate a smooth block B_s by a 5×5 Gaussian filter matrix M_G which is shown in Figure 3.6 according to the following formula:

$$B_s = \frac{1}{159} M_G \times B .$$

Step 2. For each pixel P_i in B_s, perform the following steps.

2.1. Use two 3×3 Sobel masks (Figure 4.6) to compute the first derivatives in the horizontal and vertical directions of P_i to obtain two edge gradients G_x and G_y, respectively.

2.2. Compute the edge gradient *G* and the direction θ according to the following formula:

$$G = \sqrt{{G_x}^2 + {G_y}^2} ;$$

$$\theta = \tan^{-1}(\frac{G_y}{G_x}) .$$

2.3. Classify the edge direction θ into four directions according the following steps.

2.3.1. If θ is a west and east direction, denote it by a horizontal direction *h*.

2.3.2. If θ is a north and south direction, denote it by a vertical direction *v*.

2.3.3. If θ is a north east and south west direction, denote it by a north-east-and-south-west diagonal direction *e*.

2.3.4. If θ is a north west and south east direction, denote it also by a north-west-and-south-east diagonal direction *w*.

Step 3. Add up all edge gradients G_h with direction *h*, and then divide the sum by the number of G_h to obtain an average value E_h.

Step 4. Add up all edge gradients G_v with direction *v*, and then divide the sum by the number of G_v to obtain an average value E_v.

Step 5. Add up all each edge gradients G_e with direction *v*, and then divide the sum by the number of G_e to obtain an average value E_e.

Step 6. Add up all edge gradients G_w with direction *v*, and then divide the sum by the number of G_w to obtain an average value E_w.

Step 7. Assemble E_h, E_v, E_e,and E_w to form *E*.

Algorithm 4.6. Combining edge fitting and data hiding.

Input: a source image *I*, a tetromino database *DB*, a secret message *M*, a secret key *K*, and an edge threshold T_e.

Output: a stego-tetromino-based mosaic image *I′* with enhanced-edge visual effects.

Steps:

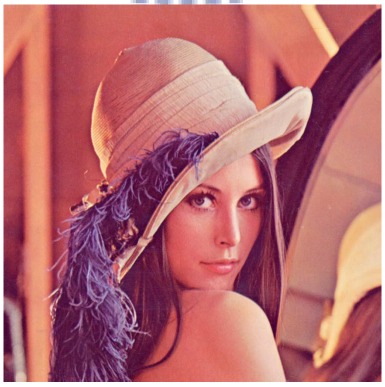
Step 1. Use the secret key *K* to transform *M* into a randomized decimal digit sequence S_r by Algorithm 4.1.

Step 2. Divide *I* into a set of 4×4 blocks, $B = \{B_1, B_2, .., B_n\}$.

Step 3. Divide S_r into several substrings $\{S_1, S_2, .., S_n\}$, each having a length of two digits.

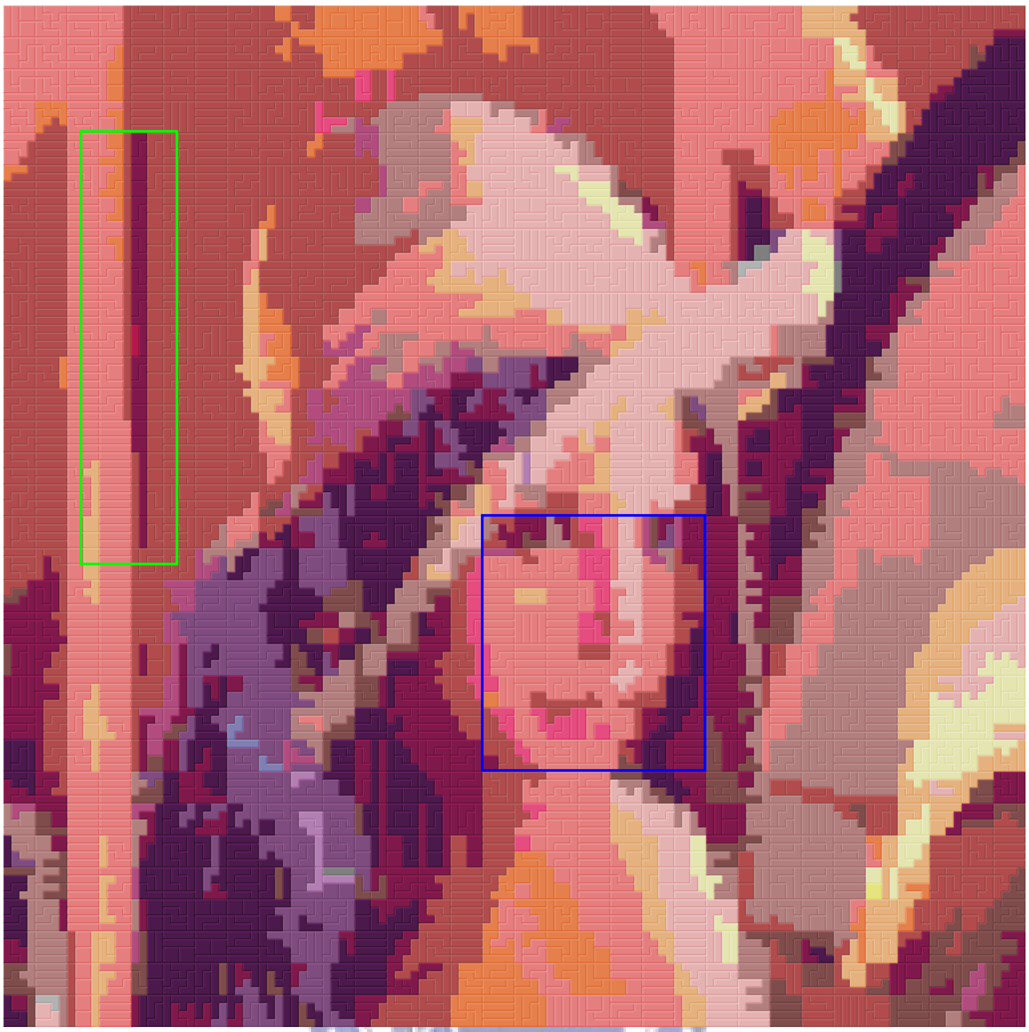
Step 4. For each block B_i in *B* and each S_i in *S*, perform the following steps.

```
    4.1. Perform the Canny edge detection on $B_i$ by Algorithm 4.5 to obtain four edge values $E$ of the four directions.
    4.2. Choose the maximum value of $E$ and denote it by $E_m$.
    4.3. Check $E_m$ according to the follow steps.
        4.3.1. If $E_m < T_e$, perform the following data hiding steps on $B_i$.
            4.3.1.1 Transform $S_i$ into a decimal number $d$.
            4.3.1.2 Choose a tile image $L$ from $DB$ according to $d$.
        4.3.2. If $E_m \geqq T_e$, perform the following edge fitting steps on $B_i$.
            4.3.2.1 Check the direction of $E_m$ and denote it by $R$.
            4.3.2.2 Choose a tile image $L$ from Figure 4.7 according to $R$.
        4.3.3. Replace $B_i$ with $L$.
        4.3.4. Perform a border enhancement process (Algorithm 3.6) on $L$ to generate $L'$.
Step 5. For each block $B_i$ in $I$, replace it with corresponding $L'$ to create a stego-tetromino-based mosaic image $I'$ as output.
```



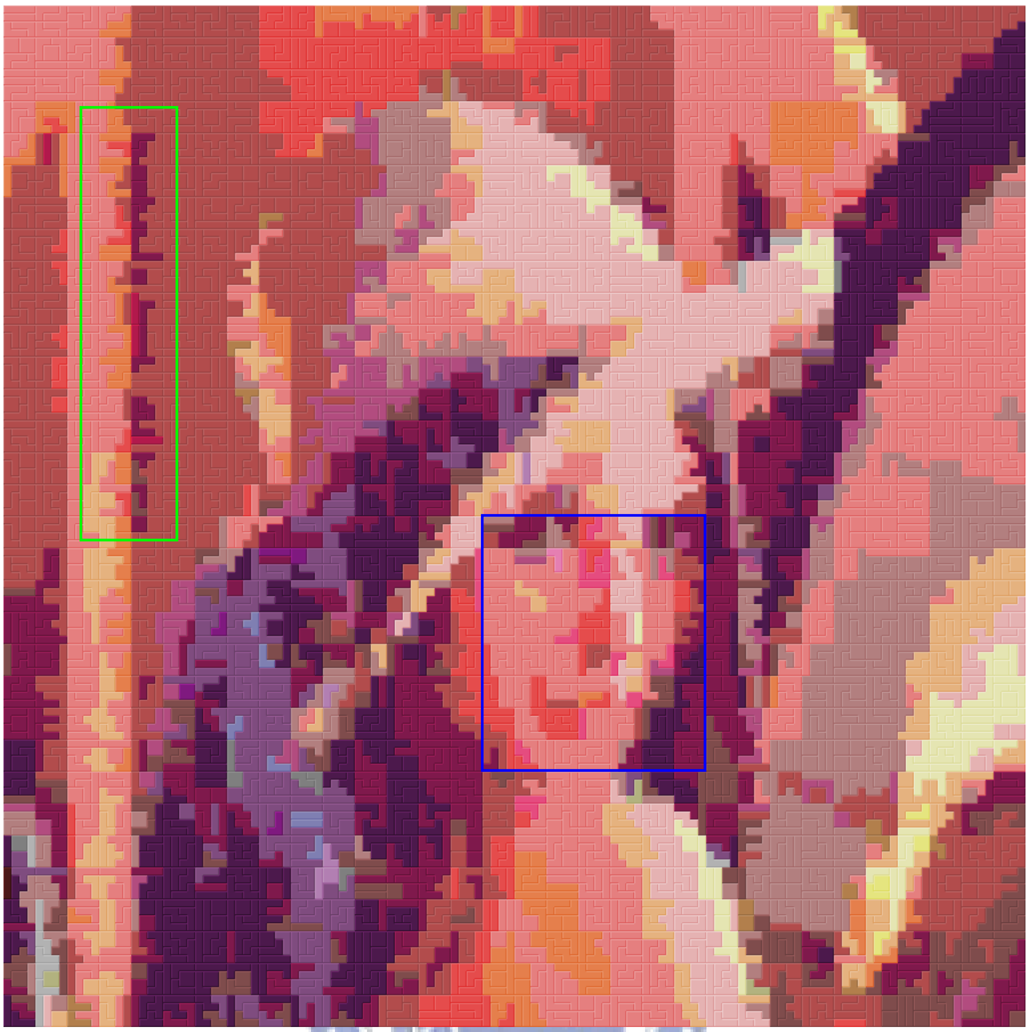
(a)

Figure 4.4 A comparison between two tetromino-based mosaic images. The blue and green regions show the differences caused by noise. (a) An original image. (b) An image created from (a) by a mosaic creation in Chapter 3. (c) An image created from (a) by our data hiding method in Section 4.2.



(b)

Figure 4.4 A comparison between two tetromino-based mosaic images. The blue and green regions show the differences caused by noise. (a) An original image. (b) An image created from (a) by a mosaic creation in Chapter 3. (c) An image created from (a) by our data hiding method in Section 4.2 (continued).



(b)

Figure 4.4 A comparison between two tetromino-based mosaic images. The blue and green regions show the differences caused by noise. (a) An original image. (b) An image created from (a) by a mosaic creation in Chapter 3. (c) An image created from (a) by our data hiding method in Section 4.2 (continued).

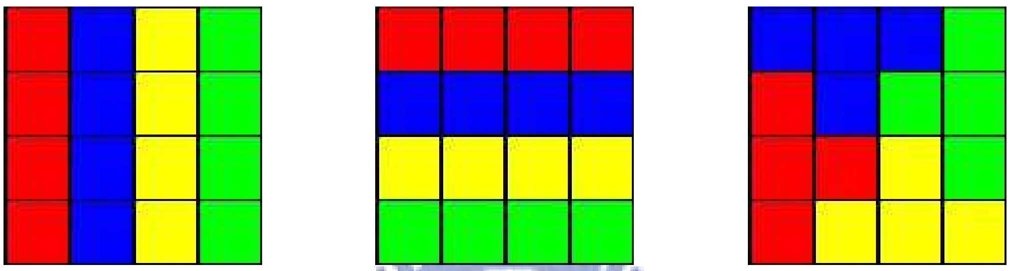
2	4	5	4	2
4	9	12	9	4
5	12	15	12	5
4	9	12	9	4
2	4	5	4	2

Figure 4.5 A Gaussian filter matrix.

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

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

Figure 4.6 3×3 Sobel masks



(a) (b) (c)

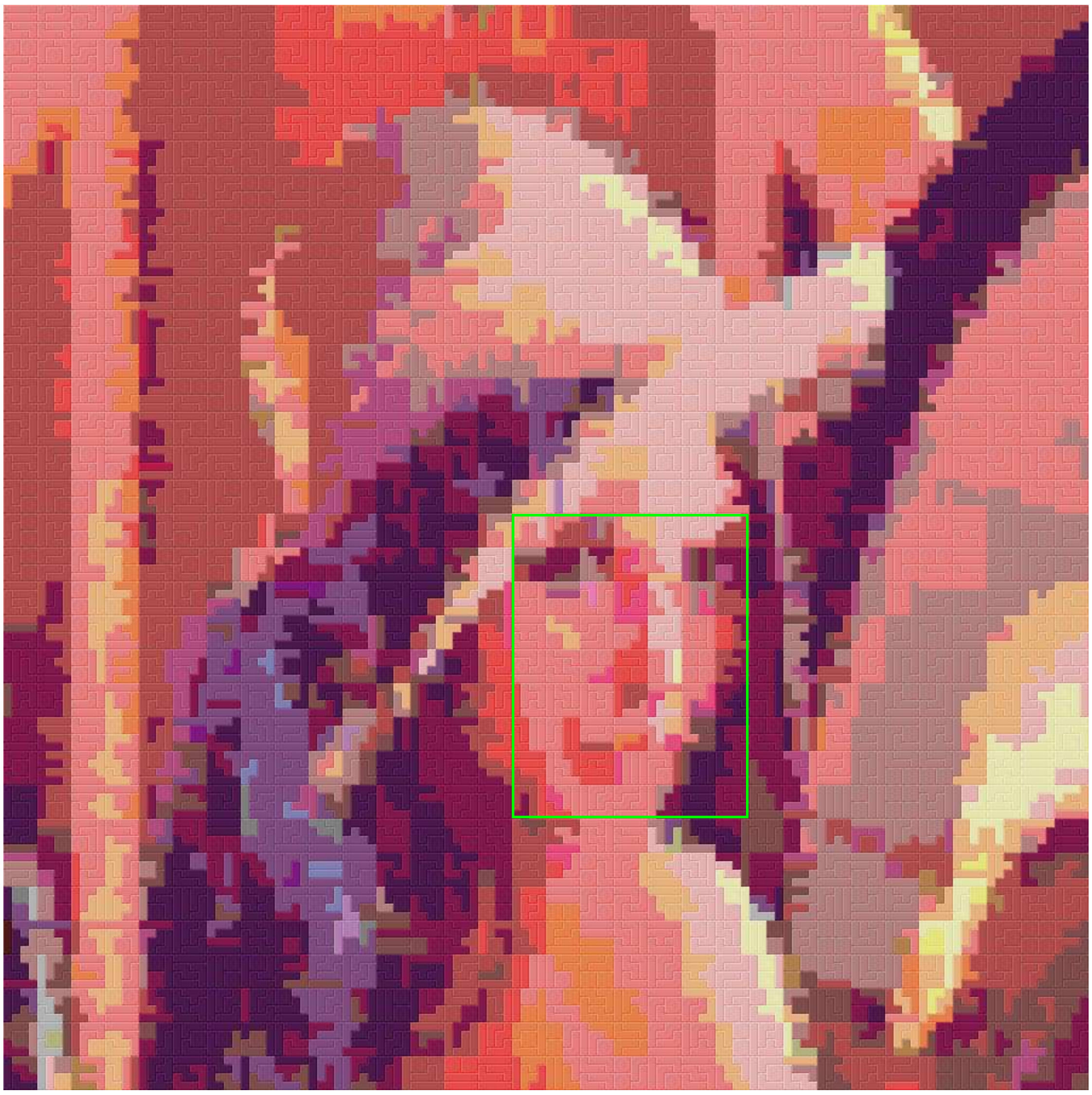
Figure 4.7 Some illustrations of tile images for different edge directions. (a) Vertical direction. (b) Horizontal direction. (c) Two diagonal directions.

4.3.3 Experimental Results

Some experimental results generated by the proposed edge fitting method (Algorithm 4.6) are shown in this section. Figures 4.8(a) and 4.8(b) show a source image and the result of embedding data process created from the source image without applying an edge fitting method. Figure 4.8(c) shows an improved result created by using the edge fitting method.

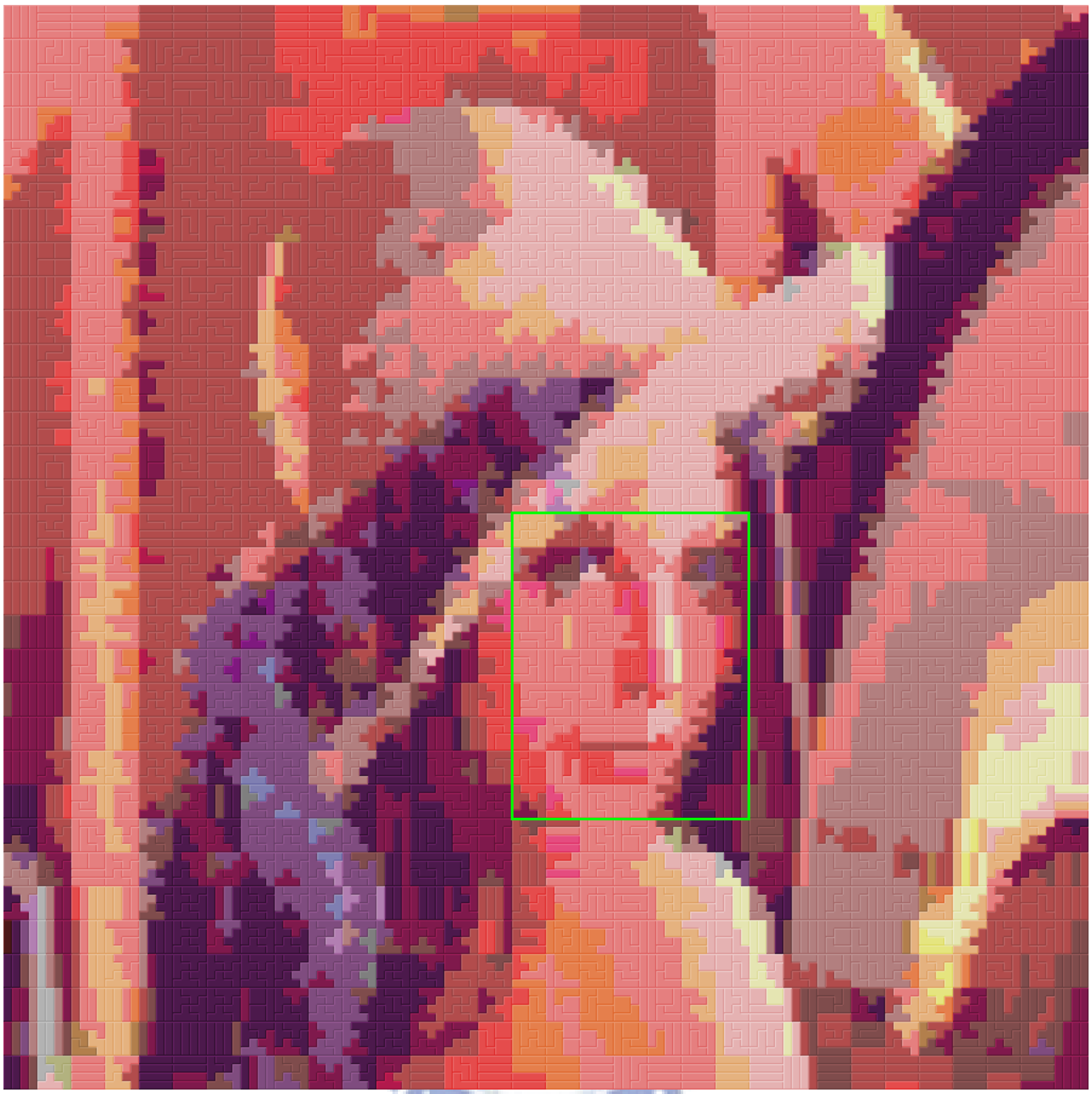
4.4 Discussions and Summary

In this chapter, we have proposed a data hiding technique by using distinct combinations of tetrominoes. Then, a process for extracting data from stego-tetromino-based mosaic images is proposed. In addition, an edge fitting method has been proposed to enhance edge effects by adjusting combinations of tetrominoes using the information obtained by edge detection. Some experimental results were presented to show the effects of the proposed algorithms.



(b)

Figure 4.8 Some experimental results. The green regions show the noise improvement achieved by an edge fitting method. (a) A stego-tetromino-based mosaic image created from 4.4(a) with embedded data. (b) A stego-tetromino-based mosaic image created from 4.4(a) with embedded data by the proposed edge fitting method.



(c)

Figure 4.8 Some experimental results. The green regions show the noise improvement achieved by an edge fitting method. (a) A stego-tetromino-based mosaic image created from 4.4(a) with embedded data. (b) A stego-tetromino-based mosaic image created from 4.4(a) with embedded data by the proposed edge fitting method (continued).

Chapter 5
Data Hiding in Tetromino-based Mosaic Images by Small Tetromino Color Shiftings

5.1 Idea of Proposed Method

Color is an important factor in the human visual system. In general, human eyes are perceptive to luminances of colors but insensitive to chrominances of colors. Figure 5.1 shows a comparison of four colors which are different from one another. It seems to have no difference among the four colors which are mutually shifted slightly in the RGB color model. Therefore, we can utilize this characteristic to hide data in a tetromino-based mosaic image, as done in the proposed method of hiding data in tetromino-based mosaic images. For each tetromino of a tetromino-based mosaic image, we perform a small color shifting to hide data by adjusting the colors of neighboring grids in a tetromino. In addition, the original tetromino-based mosaic image can be losslessly recovered from a stego-image which is created by this data hiding method.

5.2 Proposed Data Hiding by Small Tetromino Color Shiftings

5.2.1 Scheme of Propose Method

In this section, the proposed data hiding method by small tetromino color shiftings is described. We choose one grid from a tetromino and name it *pivot grid* at first. An illustration of pivot grids of the nineteen tetromino types is shown is Figure 5.2. A color of the tetromino is stored in the pivot grid for the use in data extraction with a lossless recovery of the original image from the stego-image. Next, the other three grids of the tetromino are used to embed data by changing slightly the colors of these grids. Proven by our experiments, it is possible to increase or decrease each color channel value of a tetromino grid pixel by one or two without incurring perceptual difference. This means that we can encode five color shift values which range from -2 to 2 in each color channel for data hiding. Because full-color images which are used as our inputs have three channels in the RGB color model, totally 5^3 types of color shiftings can be encoded for each grid of the tetromino. Also, according to the definition of a tetromino, it is composed of four grids which are connected orthogonally. Except the pivot grid, three grids can be used to embed data in a tetromino. And there are four tetrominoes in a block. As a result, $5^{3\times3\times4}$ variations can be encoded for hiding data in a block. That is to say, each block can embed at least 83 bits because the largest value of a power of 2 smaller than $5^{3\times3\times4}$ is 2^{83}. Then, a stego-image generated from a tetromino-based mosaic image is created by the above-mentioned data hiding process. A data extraction process of the proposed data hiding method is also described later in this chapter.

5.2.2 Data Hiding Process

In this section, we describe the proposed data hiding technique using a tetromino-based mosaic image as an input. Each color channel of a tetromino in this mosaic image is utilized to embed data by increasing or decreasing the values of the color channels slightly. Table 5.1 shows a coding table for encoding secret message bits by color shiftings for use in data hiding. Algorithm 5.1 shows the details of our method.

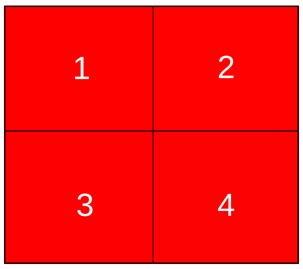


Figure 5.1 A comparison of four close colors in a block. (1) Color values of RGB channel = (255, 0, 0). (2) Color values of RGB channel = (255, 2, 0). (3) Color values of RGB channel = (253, 0, 2). (4) Color values of RGB channel = (253, 2, 2).

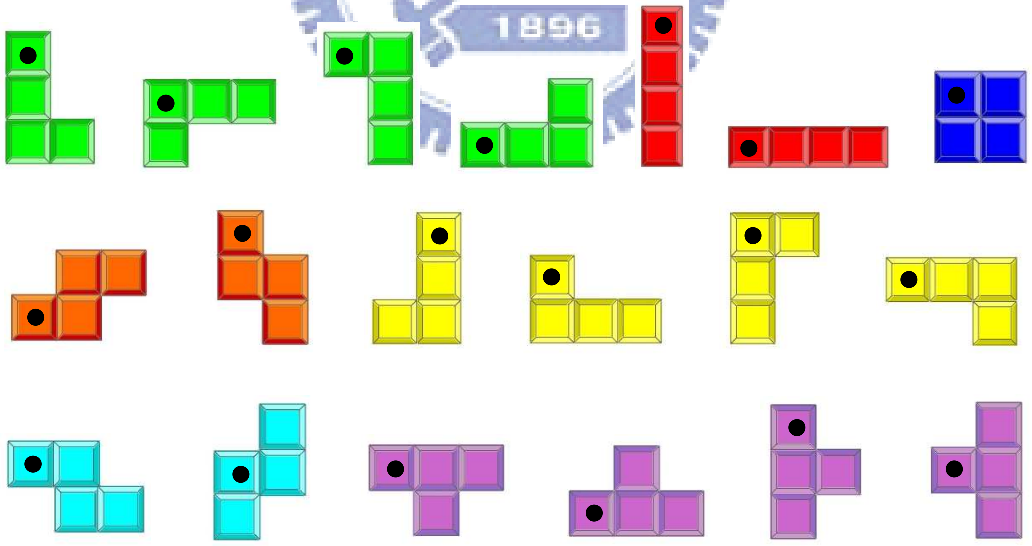


Figure 5.2 An illustration of nineteen tetromino types and their pivot grids. The black spot of each tetromino depicts its pivot grid.

Algorithm 5.1. Data hiding by small tetromino color shifting.

```
Input: a tetromino-based mosaic image $I$, a secret message $M$, a tetromino database
       $DB$, and a secret key $K$.
Output: a stego-tetromino-based mosaic image $I'$.
Steps:
Step 1. Divide $I$ into several blocks of 4×4 grids, $B = \{B_1, B_2, ..., B_n\}$.
Step 2. Transform $M$ into a quinary digit sequence $Q = \{Q_1, Q_2, ..., Q_n\}$ and use $K$ to
        randomize $Q$.
Step 3. For each block $B_i$ in $B$, perform the following steps.
    3.1. Obtain a number $m$ by recognizing the tetromino combination $B_i$ from $DB$
         by Algorithm 4.3.
    3.2. According to $m$, obtain the positions $P = \{P_1, P_2, P_3, P_4\}$ of four
         tetrominoes $T = \{T_1, T_2, T_3, T_4\}$ from $DB$.
    3.3. For each tetromino $T_i$ in $T$ and its position $P_i$ in $P$, perform the following
         steps.
        3.3.1. Find a pivot grid $G$ according to $P_i$.
        3.3.2. For each grid $g$ of the other three grids in $T_i$, perform the following
               operations.
            3.3.2.1 Obtain a color $C_g$ of the center pixel of $g$.
            3.3.2.2 Create an array $E$ to store the border information of $g$.
            3.3.2.3 For each border pixel $b$ of $g$, obtain a color $C_b$ of $b$ and check $C_b$
                    according to the follow steps.
                A   If $C_b$ is not equal to $C_g$, record *true* in $E$.
                B   If $C_b$ is equal to $C_g$, record *false* in $E$.
            3.3.2.4 For each color channel $c$ of $C_g$ and each $Q_i$ in $Q$, perform the
                    following operations.
                A.  Acquire a shifting parameter $d$ from Table 5.1 according to $Q_i$.
```

B. Use c and d to obtain c' by the following formula to embed Q_i:

$$c' = c + d\ .$$

C. For each pixel x in g, replace its color channel value c by c', except when x is a border pixel and the corresponding value in E is *true*.

3.4. Perform the above steps to obtain a stego-tetromino-based mosaic image I'.

In Step 3.3.2.4.C of Algorithm 5.1 above, we preserve the color values of the pixels in the region of the border enhancement which have been described in Chapter 3. This is because we hope to maintain the visual effects created by the border enhancement and it does not cause any effect on the proposed data hiding method.

Table 5.1 A coding table between a message digit Q_i and a shifting parameter d.

Message digit Q_i	Shifting parameter d
0	-2
1	-1
2	0
3	1
4	2

5.2.3 Data Extraction Process

The proposed data extraction process is an inverse version of the data hiding process. Besides extracting data, the process also restores the original cover image from the stego-image in the mean time. We so call it a lossless data recovery process, as mentioned previously, where “data” mean both the message data and the cover image data. In this section, we describe this process, which utilizes a color stored in a pivot grid of each tetromino in a stego-image to achieve its lossless recovery function. The detail of the process is described as algorithm below.

Algorithm 5.2. Lossless data recovery process.

Input: a stego-tetromino-based mosaic image I, a tetromino database DB, and a secret key K.

Output: a secret message M, and a losslessly recovered cover tetromino-based mosaic image I'.

Steps:

```
Step 1. Create an *empty* quinary digit sequence $Q$ initially.
Step 2. Divide $I$ into several blocks of 4×4 grids, $B = \{B_1, B_2, ..., B_n\}$.
Step 3. For each block $B_i$ in $B$, generate a tetromino-based mosaic image $I'$ by the
        following steps.
    3.1. Obtain a number $m$ by recognizing the tetromino combination $B_i$ from $DB$
         by Algorithm 4.3.
    3.2. According to $m$, obtain the positions $P = \{P_1, P_2, P_3, P_4\}$ of four
         tetrominoes $T = \{T_1, T_2, T_3, T_4\}$ from $DB$.
    3.3. For each tetromino $T_i$ in $T$ and its position $P_i$ in $P$, perform the following
         operations.
        3.3.1. Find a pivot grid $G$ according to $P_i$, and obtain a color $C_G$ of $G$.
        3.3.2. For each grid $g$ of the other three grids in $T_i$, perform the following
               operations.
            3.3.2.1. Obtain a color $C_g$ in $g$.
```

3.3.2.2. Create an array E to store the border information of g.

3.3.2.3. For each border pixel b of g, obtain the color C_b of b and check C_b according to the follow steps.

A. If C_b is not equal to C_g, record *true* in E.

B. If C_b is equal to C_g, record *false* in E.

3.3.2.4. For each color channel value c of C_g, perform the following operations.

A. Obtain the corresponding color channel value of C_G and denote it by c'.

B. Use c and c' to obtain d by the following formula:

$$d = c - c'.$$

C. Acquire a quinary digit Q_i from Table 5.1 according to d and store Q_i in Q in order.

D. For each pixel x in g, replace its color channel value c by c', except when x is a border pixel and the corresponding value in E is *true*.

3.4. Use K to generate a random number sequence r and use r to recover Q into a quinary digit sequence Q' in its original order.

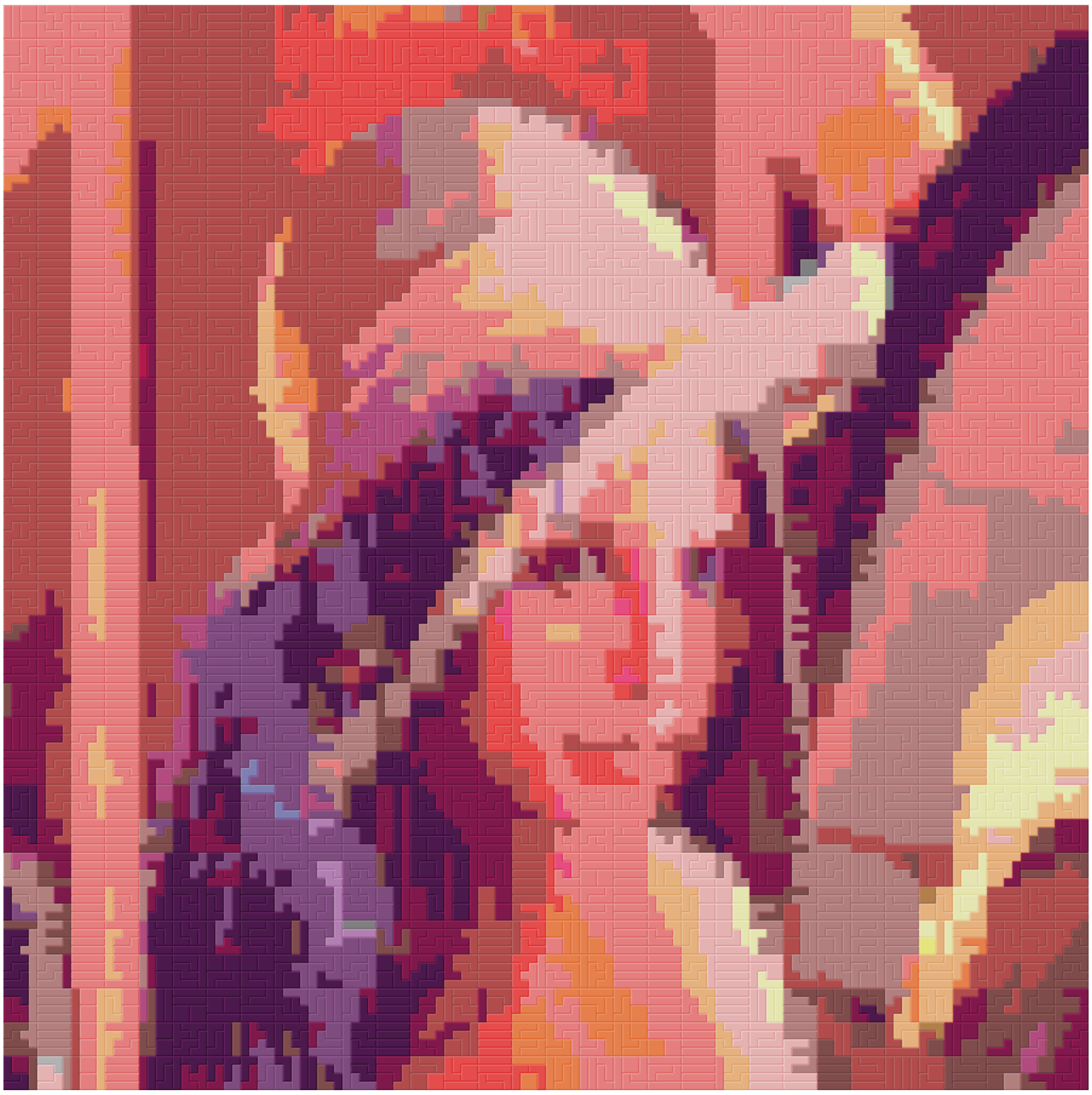
3.5. Transform Q' into a binary digit sequence to form a secret message M as output.

5.3 Experimental Results

Figure 5.3 and 5.4 show some experimental results of applying the proposed data hiding method in a tetromino-based mosaic image. Figures 5.3(b) and 5.4(b) are two tetromino-based mosaic images with watermarks embedded by the proposed data embedding process (Algorithm 5.1). Figures 5.3(d) and 5.4(d) are two watermarks extracted from Figures 5.3(b) and 5.4(b), respectively, with correct keys by the proposed data extraction process (Algorithm 5.2). Figures 5.3(e) and 5.4(e) show the two watermarks extracted from Figures 5.3(b) and 5.4(b), respectively, with wrong keys.

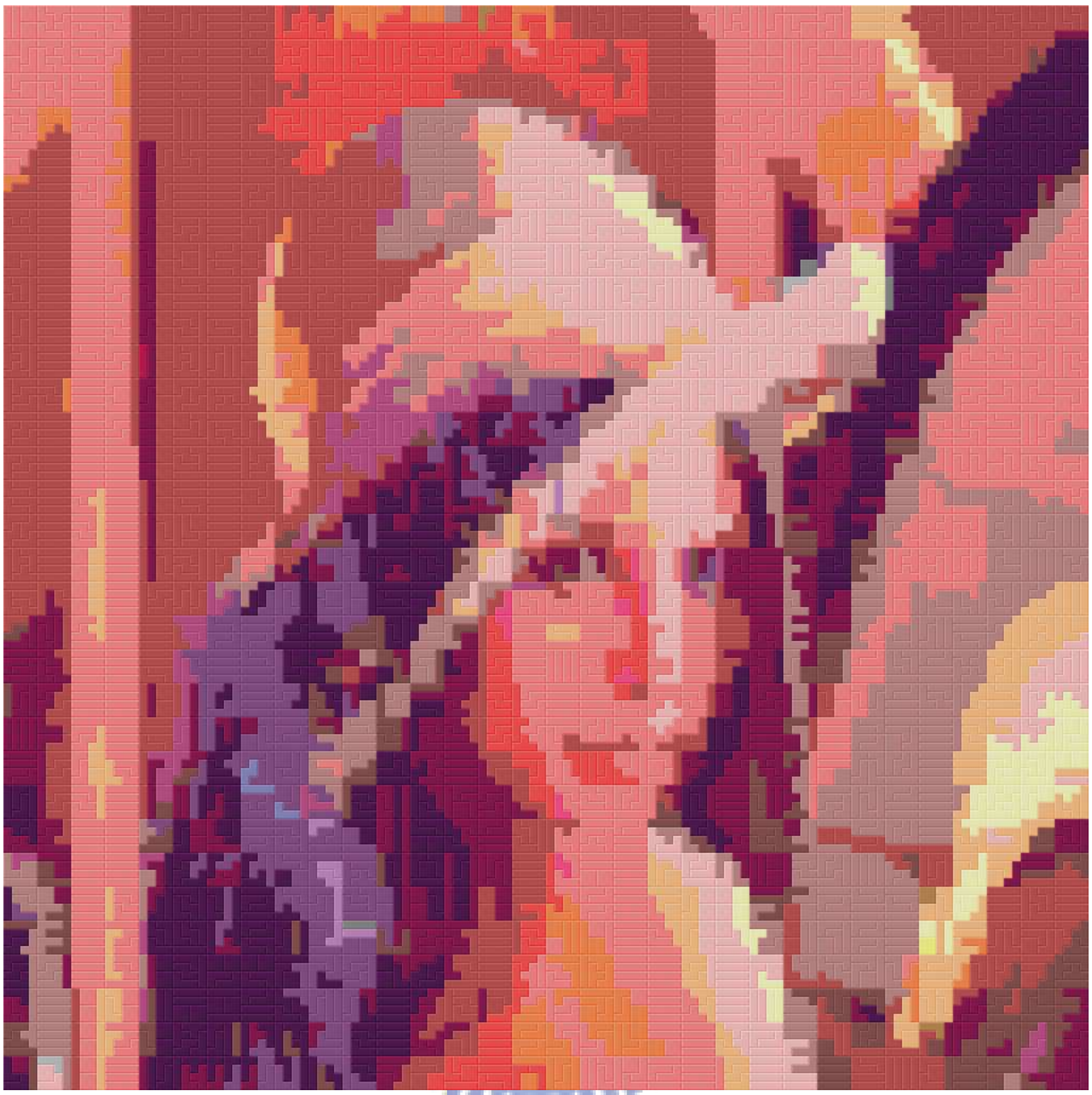
5.4 Discussions and Summary

In this chapter, a data hiding method based on increasing or decreasing values of color channels slightly has been proposed. Using the properties of tetromino-based mosaic images, we enlarge the capacity of data hiding by adjusting the colors of each tetromino. Due to its imperceptible characteristic and high data payload for data hiding, this method becomes a fine way for covert communication or copyright protection of tetromino-based mosaic images. Then a lossless data recovery process which combines the functions of data extraction and lossless cover image recovery has been described. By this process, we can extract data from a stego-image and recover the original tetromino-based mosaic image without any distortion. Some experimental results were generated by the proposed algorithms to prove the feasibility of the proposed method.

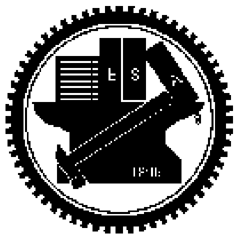


(a)

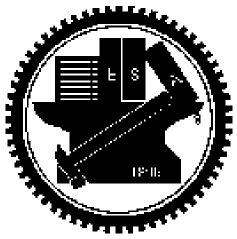
Figure 5.3 An experimental result. (a) A tetromino-based mosaic image. (b) A stego-tetromino-based mosaic image created from (a) with the watermark shown in (c) embedded. (c) The watermark. (d) A watermark extracted from (b) with a correct key. (e) A watermark extracted from (b) with a wrong key.



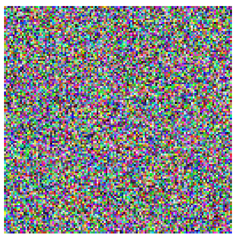
(b)



(c)

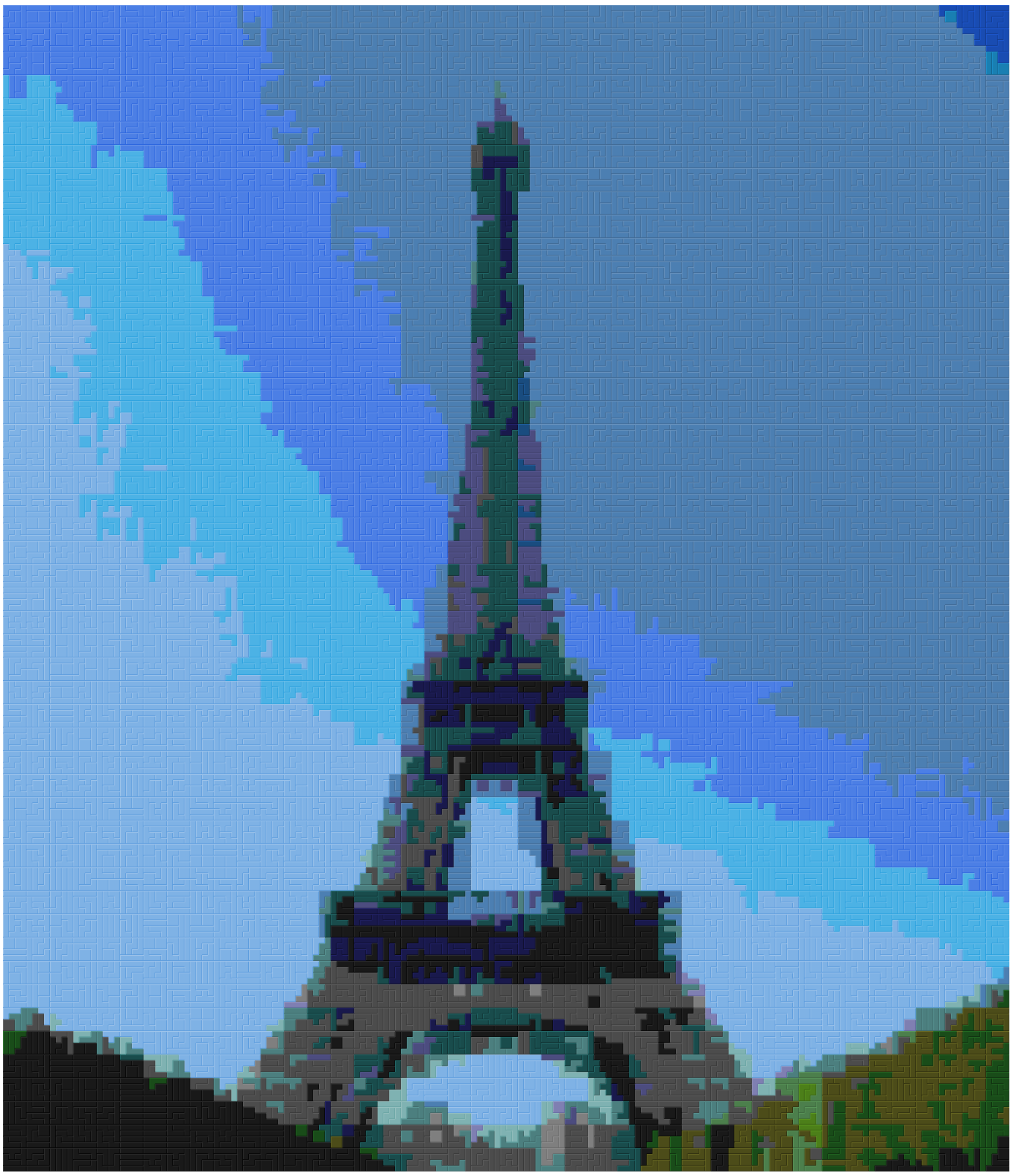


(d)



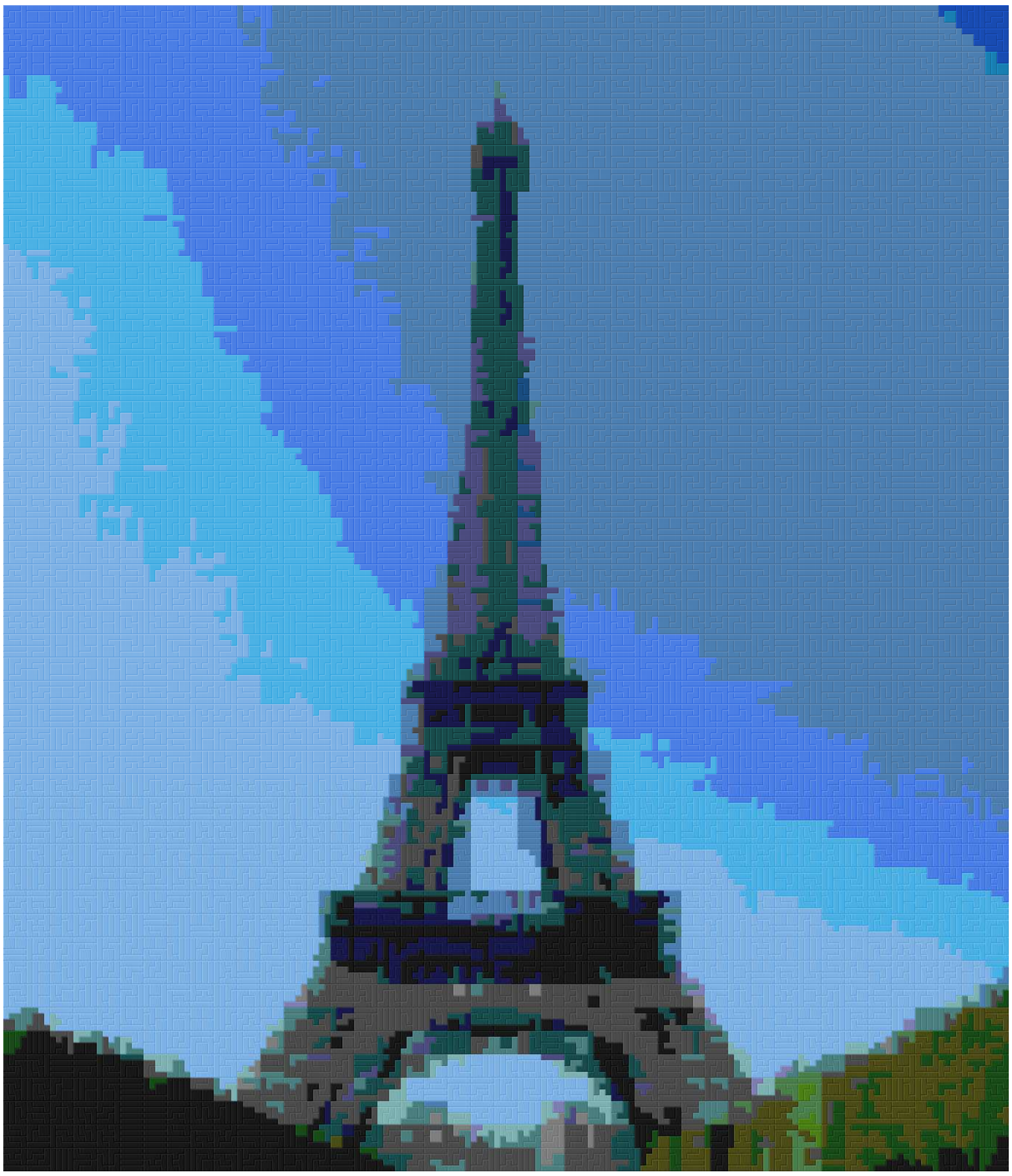
(e)

Figure 5.3 An experimental result. (a) A tetromino-based mosaic image. (b) A stego-tetromino-based mosaic image created from (a) with the watermark shown in (c) embedded. (c) The watermark. (d) A watermark extracted from (b) with a correct key. (e) A watermark extracted from (b) with a wrong key (continued).

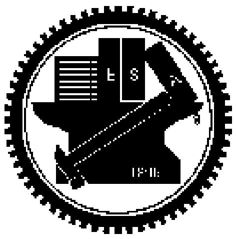


(a)

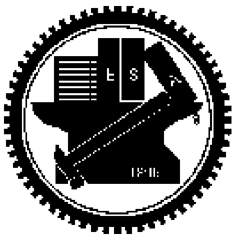
Figure 5.4 Another experimental result. (a) A tetromino-based mosaic image. (b) A stego-tetromino-based mosaic image created from (a) with the watermark shown in (c) embedded. (c) The watermark. (d) A watermark extracted from (b) with a correct key. (e) A watermark extracted from (b) with a wrong key.



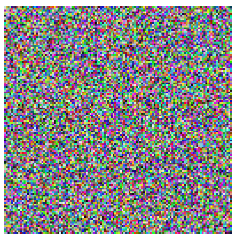
(b)



(c)



(d)



(e)

Figure 5.4 Another experimental result. (a) A tetromino-based mosaic image. (b) A stego-tetromino-based mosaic image created from (a) with the watermark shown in (c) embedded. (c) The watermark. (d) A watermark extracted from (b) with a correct key. (e) A watermark extracted from (b) with a wrong key (continued).

Chapter 6
Removable Lossless Visible Watermarking in Tetromino-based Mosaic Images

6.1 Idea of Proposed Method

In order to protect copyrights of images, many digital watermarking techniques have been designed to embed a pre-defined watermark into an image. Digital watermarking techniques are classified into two main types, *visible* and *invisible* watermarking, by their visual characteristics. Different from invisible watermarking techniques, visible watermarking techniques are used to convey ownership information directly and deter further copyright violations. As a result, visible watermarking is a suitable way to protect copyrights of art images.

Removable visible watermarking methods not only are used to embed visible watermarks for copyright protection but also allow legal users to remove the watermarks and restore the original contents. However, most removable visible watermarking techniques are *lossy*, that is, the recovered contents are changed with some distortions. Researches of removable lossless visible watermarking are hardly mentioned in the literature. Chen and Tsai [32] proposed a removable lossless visible watermarking method by replacing the colors according to a mapping between two color palettes in palette images. The detail of their proposed method will be reviewed in Section 6.2.

Having a fixed number of colors is a common feature between palette images and tetromino-based mosaic images. Based on the concept of Chen and Tsai's method [32], we will describe the proposed removable lossless visible watermarking method in tetromino-based mosaic images in this chapter.

6.2 Review of A Removable Lossless Visible Watermarking Technique for Palette images

Chen and Tsai [32] proposed a removable lossless visible watermarking method in palette images. A palette is a limited color collection which is stored in a palette image, such as a Graphics Interchange Format (GIF) image. GIF images are chosen as the input palette images in their method. Every GIF image contains at most 256 colors which are stored in an 8-bit color palette. Each color in the palette is assigned a number, called an *index*. Every pixel in the GIF image is assigned an index number referring to a color in the color palette. Furthermore, GIF images are compressed to reduce the file size without degrading the visual quality.

Before describing their method, some definitions of terms are given to help the understanding of the remainder of this section.

1. *Watermark area*: an area in a cover image where a watermark is embedded.
2. *Non-watermark area*: an area outside the watermark area.
3. *Black embedded pixels*: pixels of the cover image in the watermark area, denoted as I_b.
4. *White embedded pixels*: pixels of the cover image in the non-watermark area, denoted as I_w.

Each visible watermark used in this method is assumed to be a binary image. In order to enable the embedded watermark area to look more obvious, Chen and Tsai [32] replaced the colors of I_b with other visually different colors. This replacement was achieved by finding the farthest color among the palette colors of the cover image to replace with. First, the cover image was divided into a watermark area and a non-watermark one. Black embedded pixels I_b were replaced with other colors and white embedded pixels I_w were kept unchanged. Then, a *raw color palette* is constructed by identifying the colors of I_w and counting the occurrences of these colors. The colors of I_w were denoted as $C = \{C_0, C_1, \ldots, C_{255}\}$ and the occurrences of C were denoted as $O = \{O_0, O_1, \ldots, O_{255}\}$. The raw color palette was sorted by the O_i of O in a descending order to generate a *sorted color palette* P_s. An *adjusted Euclidean distance* by considering the concept of weighting, called a *weighted Euclidean distance*, was proposed. The traditional Euclidean distance between two colors C1 and C2 is defined in Formula 6.1, and the weighted one of color C_i in the color palette is defined in Formula 6.2 below:

$$\mu(C_1, C_2) = \sqrt{(R_1 - R_2)^2 + (G_1 - G_2)^2 + (B_1 - B_2)^2}\ ; \qquad (6.1)$$

$$\mu(C_i) = \frac{\sum_{n=0}^{255} O_n \times \mu(C_i, C_n)}{\sum_{n=0}^{255} O_n}\ . \qquad (6.2)$$

Second, they found the farthest color from the sorted color palette by using the weighted Euclidean distance (Formula 6.2) to choose the color which has the largest weighted Euclidean distance. The farthest color was denoted by C_f, which was the most different color from the colors of I_w. Then, a *rearranged color palette* P_r was set up by placing C_f in the first element. The second element of P_r is filled by choosing a color C_a which is closest to C_f and adding it to P_r. The remainder of P_r is filled by performing the previous step until the whole palette is accomplished. Finally, a one-to-one mapping between two palettes P_s and P_r is constructed. If a color C_m was labeled by index m in P_s, then this color is mapped to a color C_m' which was labeled by index m in P_r. All black embedded pixels in the cover image are replaced to generated a watermarked image by the above-mentioned replacement. Furthermore, a lossless recovery of the watermarked image was designed which is an inverse process of the proposed watermark embedding process.

Algorithm 6.1 shows a generating and mapping process of two color palettes. The details of the visible watermark embedding process in palette images and the lossless recovery process of watermarked palette images are described in Algorithms 6.2 and 6.3, respectively. An experimental result of applying this method is shown in Figure 6.1.

Algorithm 6.1. One-to-one mapping process of two color palettes.

Input: a raw color palette P_a, a set of colors $C = \{C_0, C_1, \ldots, C_n\}$ of P_a, and a set of occurrences $O = \{O_0, O_1, \ldots, O_n\}$ of C.

Output: a sorted color palette P_s and a rearranged color palette P_r.

Steps:

Step 1. Sort P_a by the occurrences O in a descending order to obtain a sorted color palette P_s.

Step 2. Create a rearranged color palette P_r, which has the same size of P_s.

Step 3. Calculate weighted Euclidean distances D_w of every color in P_s by Formula 6.1.

Step 4. Use D_w to choose a color C_f which has the largest value of D_w to be the first color of P_r.

Step 5. Calculate Euclidean distances D between C_f and the other colors in P_s by

```
        Formula 6.2.
Step 6. Use $D$ to construct $P_r$ by rearranging $P_s$ from the closest color of $C_f$ to the
        farthest color of $C_f$.
```

```
Algorithm 6.2. Visible watermark embedding process in palette images.
Input: a cover palette image $I$, a raw palette $P_a$ of $I$, a binary watermark $W$, and a
        secret key $K$.
Output: a watermarked palette image $I'$.
Steps:
Step 1. For each pixel $p$ in $I$, perform the following steps.
      1.1. If $p$ is in the watermark area, add $p$ into a set of black embedded pixels $I_b$.
      1.2. If $p$ is in the non-watermark area, add $p$ into a set of white embedded pixels
           $I_w$.
Step 2. Obtain a set of colors $C = \{C_0, C_1, \ldots, C_{255}\}$ from $P_a$.
Step 3. For every pixel in $I_w$, count the occurrences $O = \{O_0, O_1, \ldots, O_{255}\}$ of the
        colors $C = \{C_0, C_1, \ldots, C_{255}\}$, respectively.
Step 4. Use $P_a$, $C$, and $O$ to generate a sorted color palette $P_s$ and a rearranged color
        palette $P_r$ by Algorithm 6.1.
Step 5. For each pixel $x$ in $I_b$, embed $W$ into $I$ by the following steps.
      5.1. Obtain a color $C_m$ of $x$.
      5.2. Search $C_m$ in $P_s$ to obtain a color index $m$ of $C_m$.
      5.3. Obtain a color $C_m'$ from $P_r$ by the index $m$.
      5.4. Replace $C_m$ of $x$ with $C_m'$.
Step 6. Randomly pair every two pixels of $I_b$ together using $K$ to form $|I_b|/2$ pairs of
        pixels, $A$.
```

Step 7. For each pair a in A, swap the two colors of a to generate a watermarked palette image I'.

Algorithm 6.3. Lossless recovery process of watermarked palette images.

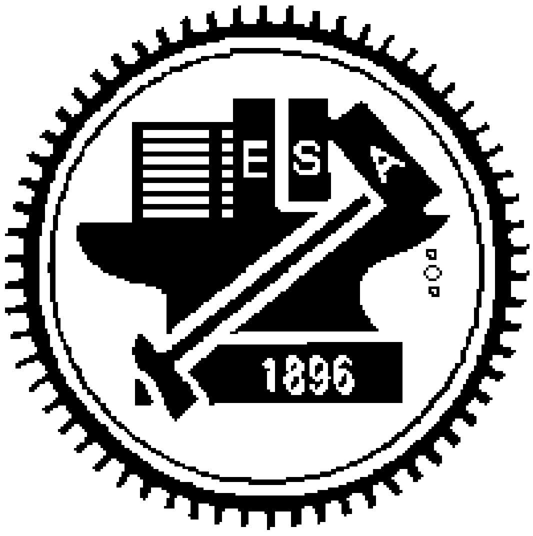
Input: a watermarked palette image I, a raw palette P_a of I, and a binary watermark W.

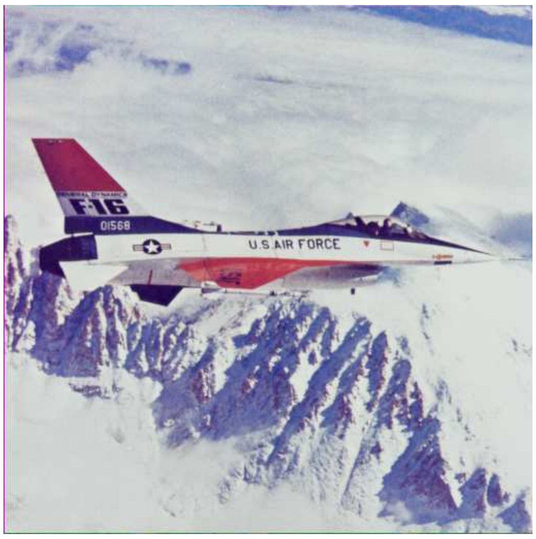
Output: a recovered palette image I'.

Steps:

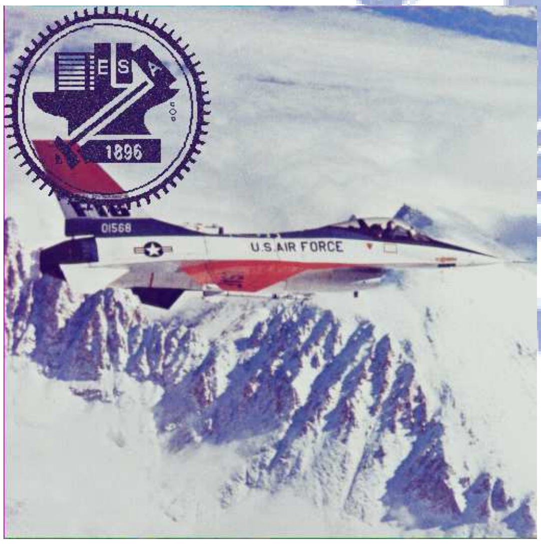
```
Step 1. For each pixel $p$ in $I$, perform the following steps.
    1.1. If $p$ is in the watermark area, add $p$ into a set of black embedded pixels $I_b$.
    1.2. If $p$ is in the non-watermark area, add $p$ into a set of white embedded pixels $I_w$.
Step 2. Obtain colors $C = \{C_0, C_1, \ldots, C_{255}\}$ from $P_a$.
Step 3. For every pixel in $I_w$, count the occurrences $O = \{O_0, O_1, \ldots, O_{255}\}$ of the colors $C = \{C_0, C_1, \ldots, C_{255}\}$, respectively.
Step 4. Use $P_a$, $C$, and $O$ to generate a sorted color palette $P_s$ and a rearranged color palette $P_r$ by Algorithm 6.1.
Step 5. Randomly pair every two pixels of $I_b$ together using $K$ to form $|I_b|/2$ pairs of pixels, $A$.
Step 6. For each pair $a$ in $A$, swap the two colors of $a$.
Step 7. For each pixel $x$ in $I_b$, remove $W$ from $I$ to generate a recovered palette image $I'$ by the following steps.
    7.1. Obtain a color $C_n$ of $x$.
    7.2. Search $C_n$ in $P_r$ to obtain a color index $n$ of $C_n$.
    7.3. Obtain a color $C_n'$ from $P_s$ by the index $n$.
```

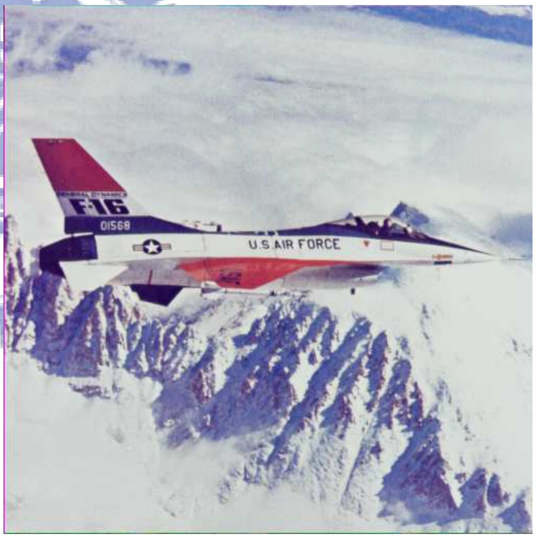
7.4. Replace C_n of x with C_n'.





(a) (b)





(c) (d)

Figure 6.1 An experimental result of Chen and Tsai's method [32]. (a) A binary watermark image. (b) A color palette image. (c) A watermarked image created by embedding the visible watermark (a) into (b). (d) A lossless recovered image created by removing the visible watermark (a) from (c).

6.3 Proposed Method for Removable Lossless Visible Watermarking in Tetromino-based Mosaic images

6.3.1 Visible Watermark Embedding Process

In this section, we describe the proposed visible watermark embedding method for tetromino-based mosaic images, which is based on Chen and Tsai's study [32]. Different from their method, the embedded watermark is hidden in tetromino-based mosaic images. That is to say, we do not need to provide the watermark during the proposed process of lossless recovery of the original image which is used to embed watermark. First, a raw palette is constructed by counting the colors of a tetromino-based mosaic image. By using the raw palette, a process of generation of two color palettes, which are denoted as P_s and P_r, and a mapping process between them are performed for watermarking. Then, for each tetromino in the white embedded pixels I_w, a binary watermark is hidden by the data hiding process using small tetromino color shiftings. At last, we replace the colors of the black embedded pixels I_b by using the one-to-one mapping of P_s and P_r. The detailed algorithm of the proposed watermark embedding method is described as follows.

Algorithm 6.4. Visible watermark embedding in tetromino-based mosaic images.

Input: a tetromino-based mosaic image I, a binary watermark W, and a secret key K.

Output: a watermarked tetromino-based mosaic image I'.

Steps:

Step 1. For each tetromino t in I, perform the following steps.

1.1. If t is in the watermark area, add all pixels in t into a set of black embedded pixels I_b.

1.2. If t is in the non-watermark area, add all pixels in t into a set of white embedded pixels I_w.

Step 2. Create a raw color palette P_a by counting colors $C = \{C_0, C_1, \ldots, C_n\}$ from I and denote the number of color elements of P_a as n.

Step 3. For every pixel in I_w, count the occurrences $O = \{O_0, O_1, \ldots, O_n\}$ of colors $C = \{C_0, C_1, \ldots, C_n\}$, respectively.

Step 4. Use P_a, C, and O to generate a sorted color palette P_s and a rearranged color palette P_r by Algorithm 6.1.

Step 5. For each tetromino in I_w, hide W in I_w by the data hiding process using small tetromino color shiftings (Algorithm 5.1).

Step 6. For each pixel p in I_b, embed W into I by the following steps.

6.1. Obtain a color C_m of p.

6.2. Search C_m in P_s to obtain a color index m of C_m.

6.3. Obtain a color C_m' from P_r by the index m.

6.4. Replace C_m of p with C_m'.

Step 7. Randomly pair every two pixels of I_b together using K to form $|I_b|/2$ pairs of pixels A.

Step 8. For each pair a in A, swap the two colors of a to generate a watermarked image I'.

6.3.2 Lossless Recovery Process of Original Images by Removing Visible Watermark

In this section, we describe the proposed lossless recovery process of the original images by removing visible watermarks. First, a binary watermark is extracted from the watermarked tetromino-based mosaic image by the lossless data recovery process, which has been described in Section 5.2.3. Then, a raw palette is constructed by counting the colors of the watermarked image. By using the raw palette, a generation of two color palettes, P_s and P_r, are performed. At last, we replace the colors of the black embedded pixels I_b to recover the original tetromino-based mosaic image by using the mapping between P_s and P_r. The detailed algorithm of the proposed lossless recovery process is described as follows.

Algorithm 6.5. Lossless recovery of tetromino-based mosaic image images.

Input: a watermarked tetromino-based mosaic image I.

Output: a recovered tetromino-based mosaic image I'.

Steps:

Step 1. Extract a binary watermark W from I by the lossless data recovery process (Algorithm 5.2)

Step 2. For each tetromino t in I, perform the following steps.

  2.1. If t is in the watermark area, add all pixels in t into a set of black embedded pixels I_b.

  2.2. If t is in the non-watermark area, add all pixels in t into a set of white embedded pixels I_w.

Step 3. Create a raw color palette P_a by counting the colors $C = \{C_0, C_1, \ldots, C_n\}$ from I and denote the number of color elements of P_a as n.

Step 4. For every pixel in I_w, count the occurrences $O = \{O_0, O_1, \ldots, O_n\}$ of colors $C = \{C_0, C_1, \ldots, C_n\}$, respectively.

Step 5. Use P_a, C, and O to generate a sorted color palette P_s and a rearranged color palette P_r by Algorithm 6.1.

Step 6. Randomly pair every two pixels of I_b together using K to form $|I_b|/2$ pairs of pixels, A.

Step 7. For each pair a in A, swap the two colors of a.

Step 8. For each pixel x in I_b, remove W from I to generate a recovered palette image I' by the following steps.

8.1. Obtain a color C_n of x.

8.2. Search C_n in P_r to obtain a color index n of C_n.

8.3. Obtain a color C_n' from P_s by the index n.

8.4. Replace C_n of x with C_n'.

6.4 Experimental Results

Figures 6.3 and 6.4 show some experimental results of applying the proposed removable visible watermarking method for a tetromino-based mosaic image. Figure 6.2 shows a binary watermark used in the experiments. Figures 6.3(b) and 6.4(b) are two watermarked tetromino-based mosaic images by the proposed watermark embedding process (Algorithm 6.4). Figures 6.3(c) and 6.4(c) are two recovered images extracted from Figures 6.3(b) and 6.4(b), respectively, with correct keys by the proposed lossless recovery process (Algorithm 6.5). Figures 6.3(d) and 6.4(d) show the two recovered images extracted from Figures 6.3(b) and 6.4(b), respectively, with wrong keys. As seen in these experimental results, the embedded watermark area looks visually different from the pixels adjacent to them. In addition, the watermark can be removed losslessly with the right key

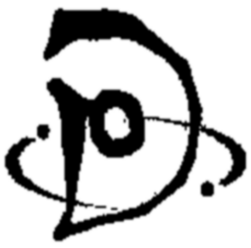
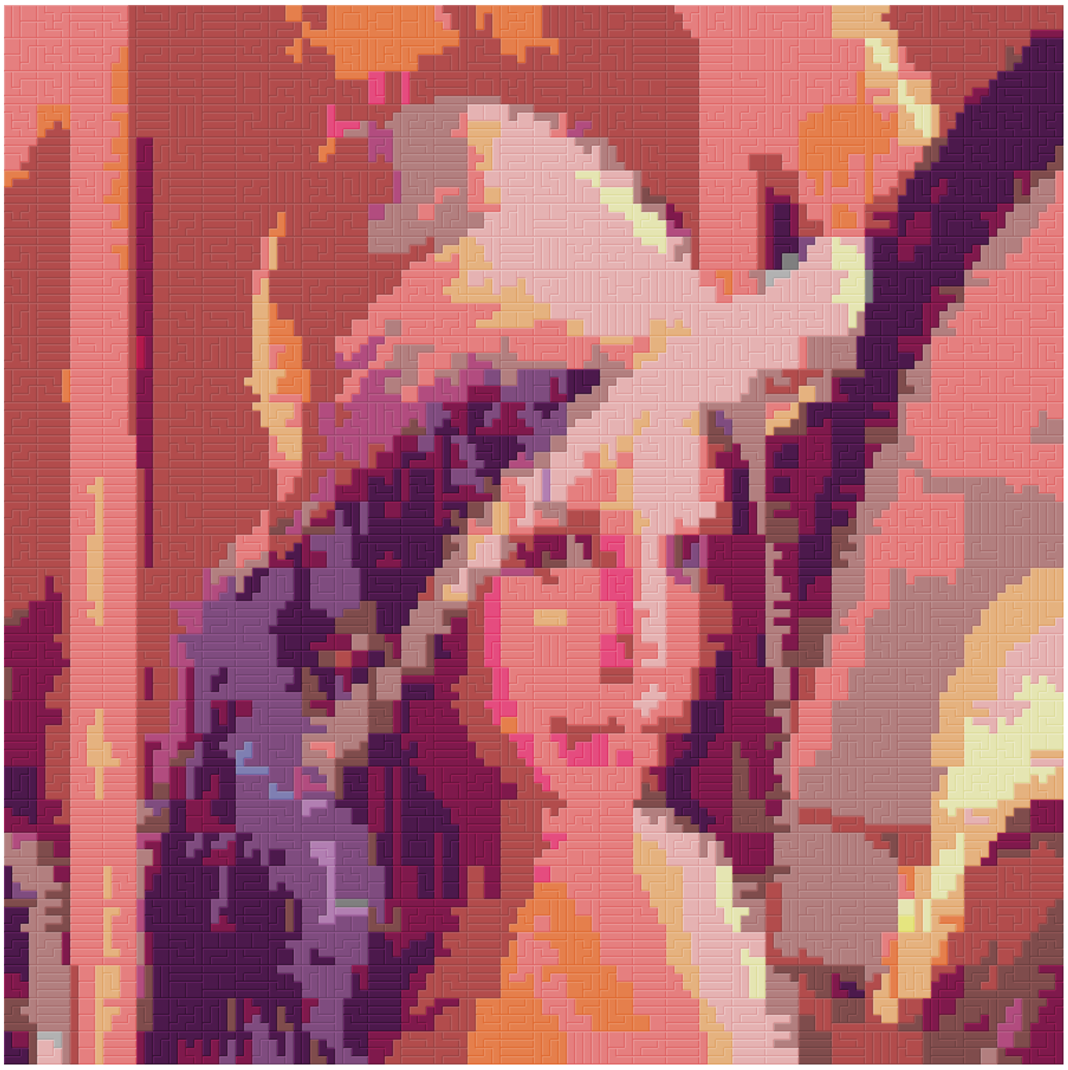
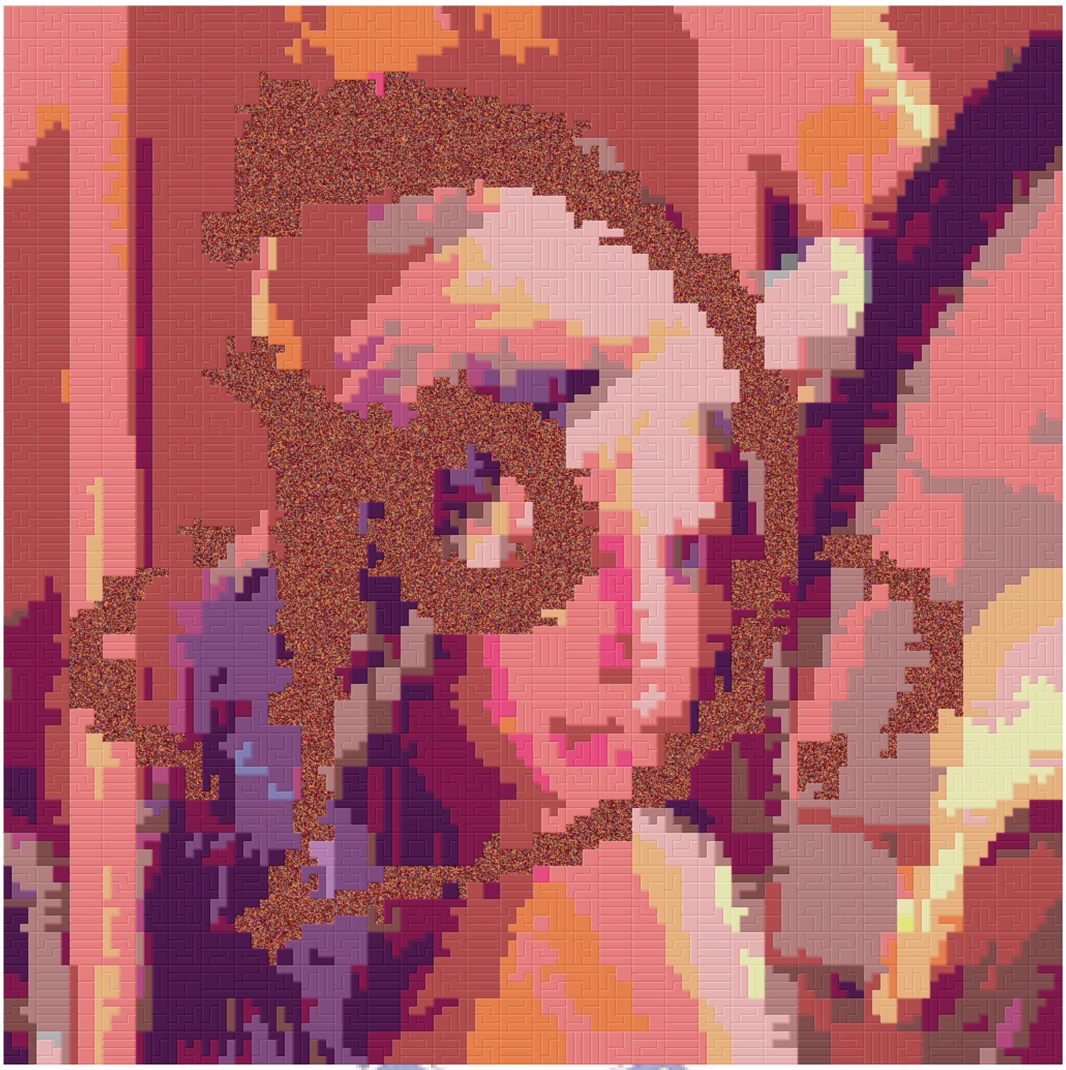


Figure 6.2 A binary watermark image of size 256×256.



(a)

Figure 6.3 An experimental result. (a) A tetromino-based mosaic image. (b) A watermarked image created from (a) with the watermark shown in Figure 6.2 embedded. (c) A recovered image created with a right key. (d) A recovered image created with a wrong key.



(b)

Figure 6.3 An experimental result. (a) A tetromino-based mosaic image. (b) A watermarked image created from (a) with the watermark shown in Figure 6.2 embedded. (c) A recovered image created with a right key. (d) A recovered image created with a wrong key (continued).

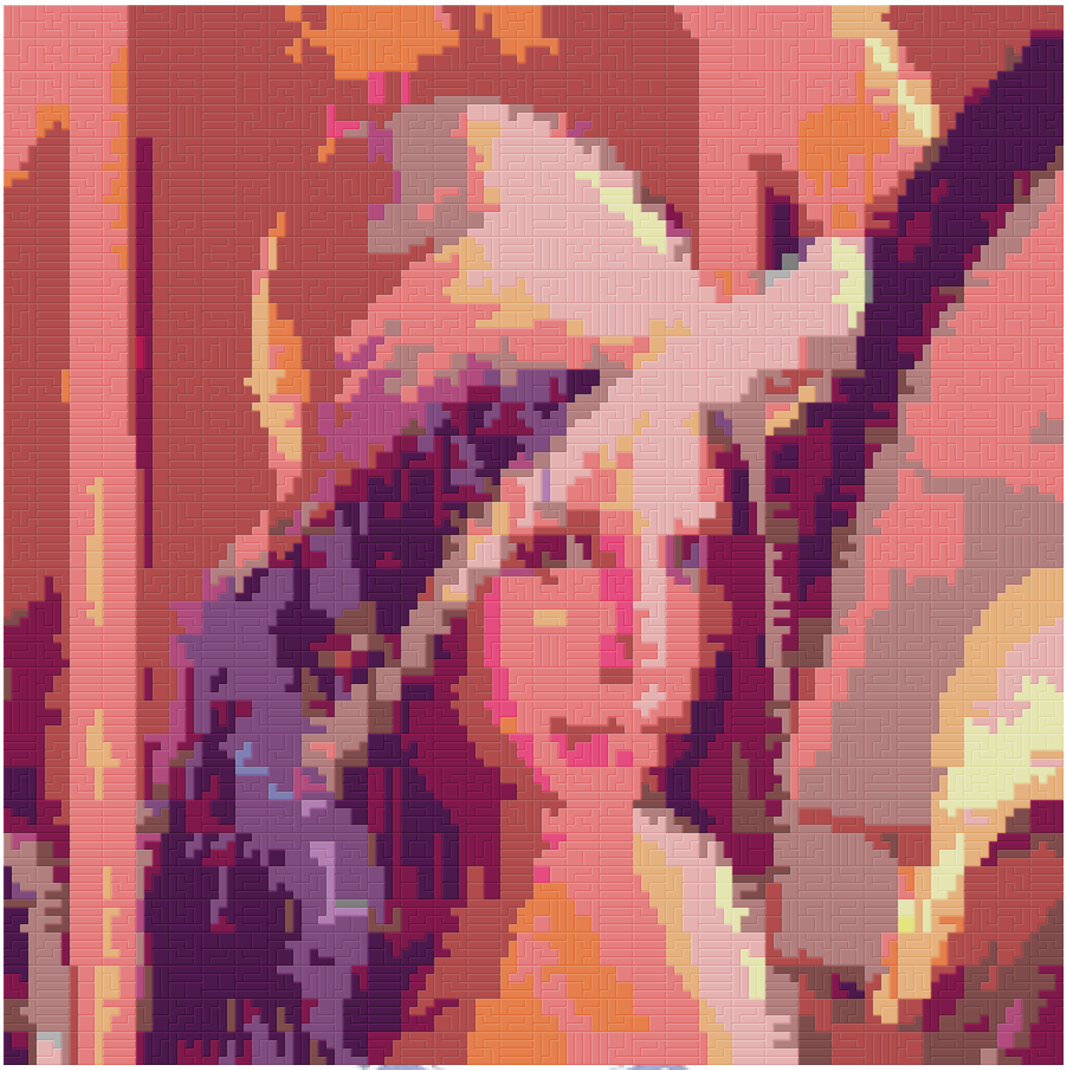
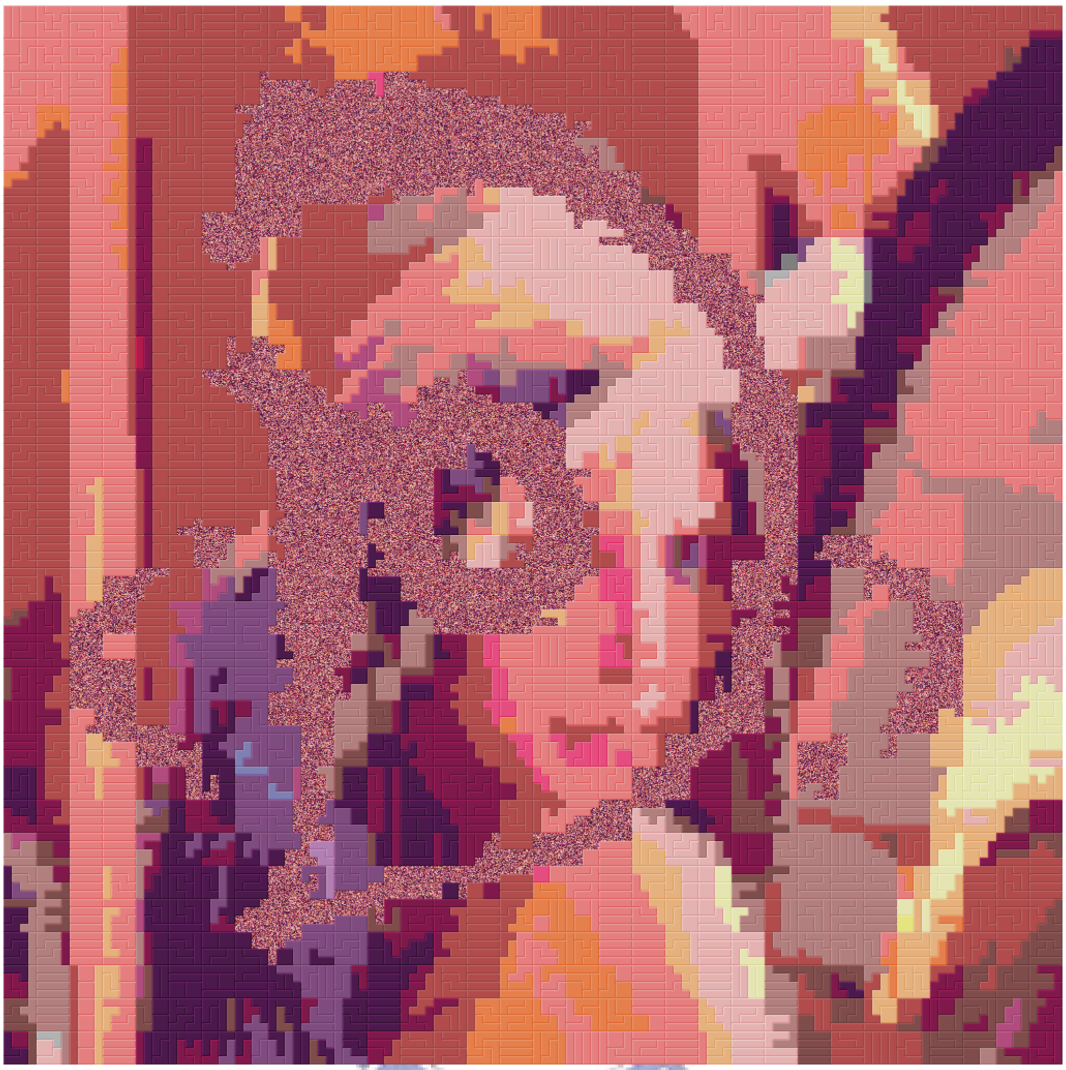
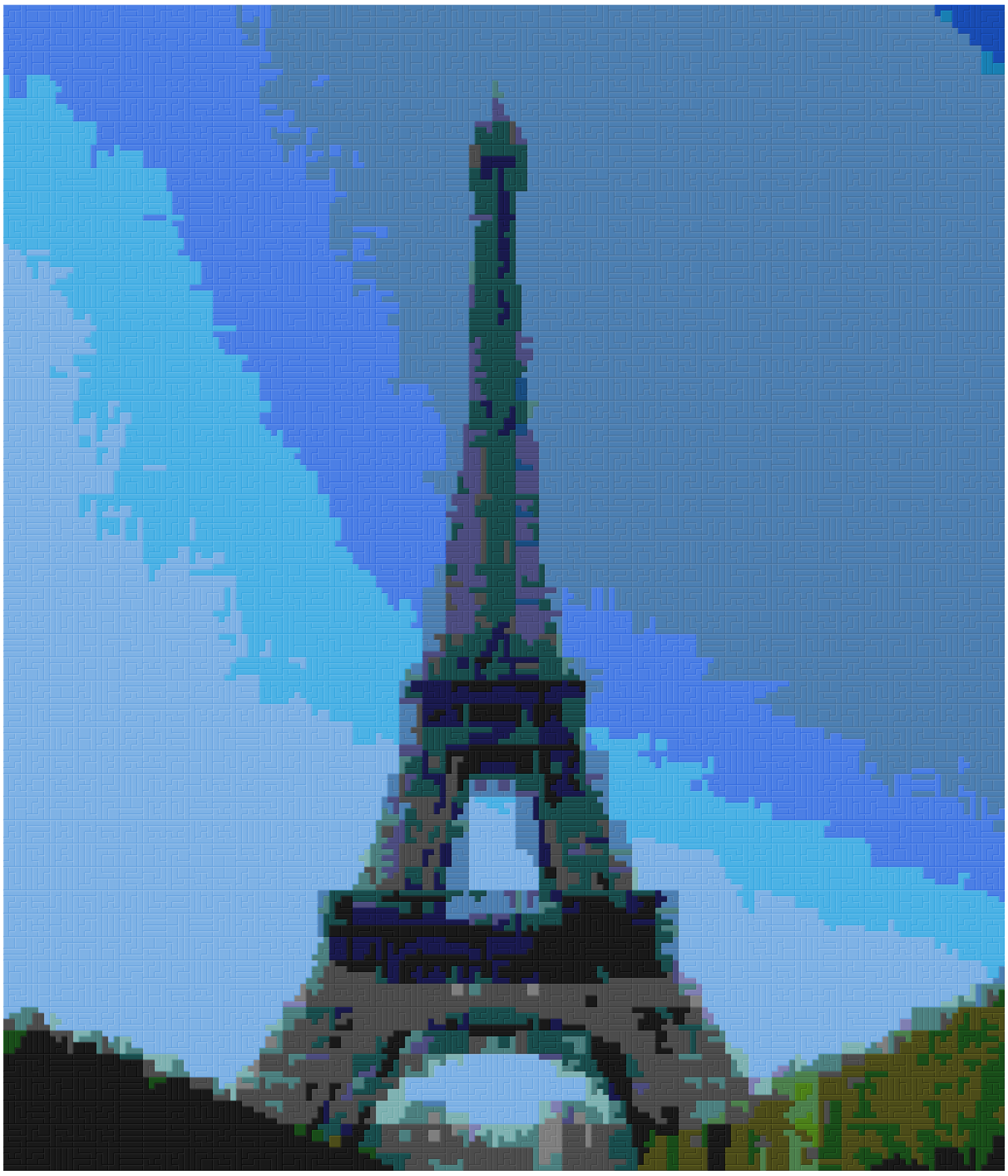

(c)

Figure 6.3 An experimental result. (a) A tetromino-based mosaic image. (b) A watermarked image created from (a) with the watermark shown in Figure 6.2 embedded. (c) A recovered image created with a right key. (d) A recovered image created with a wrong key (continued).



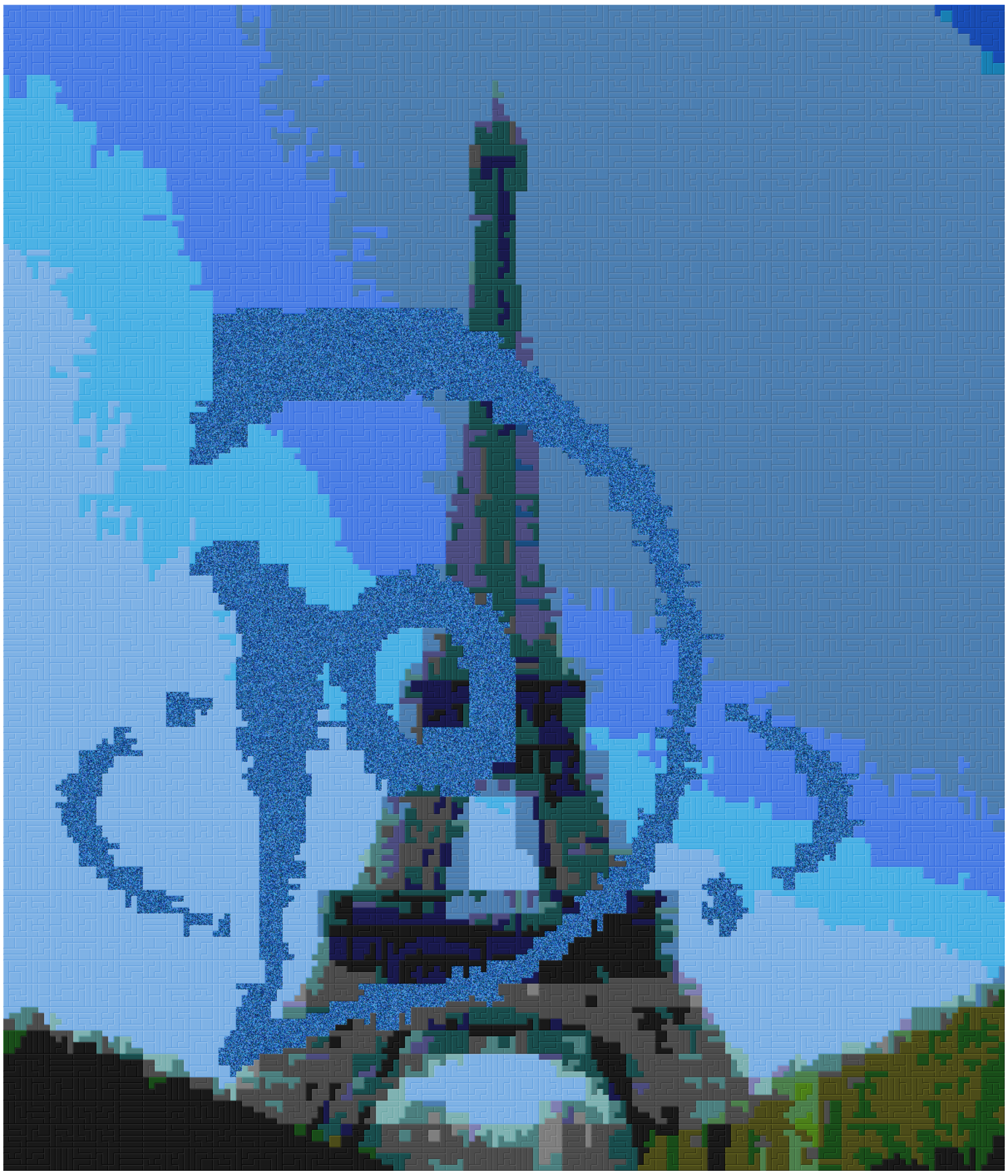
(d)

Figure 6.3 An experimental result. (a) A tetromino-based mosaic image. (b) A watermarked image created from (a) with the watermark shown in Figure 6.2 embedded. (c) A recovered image created with a right key. (d) A recovered image created with a wrong key (continued).



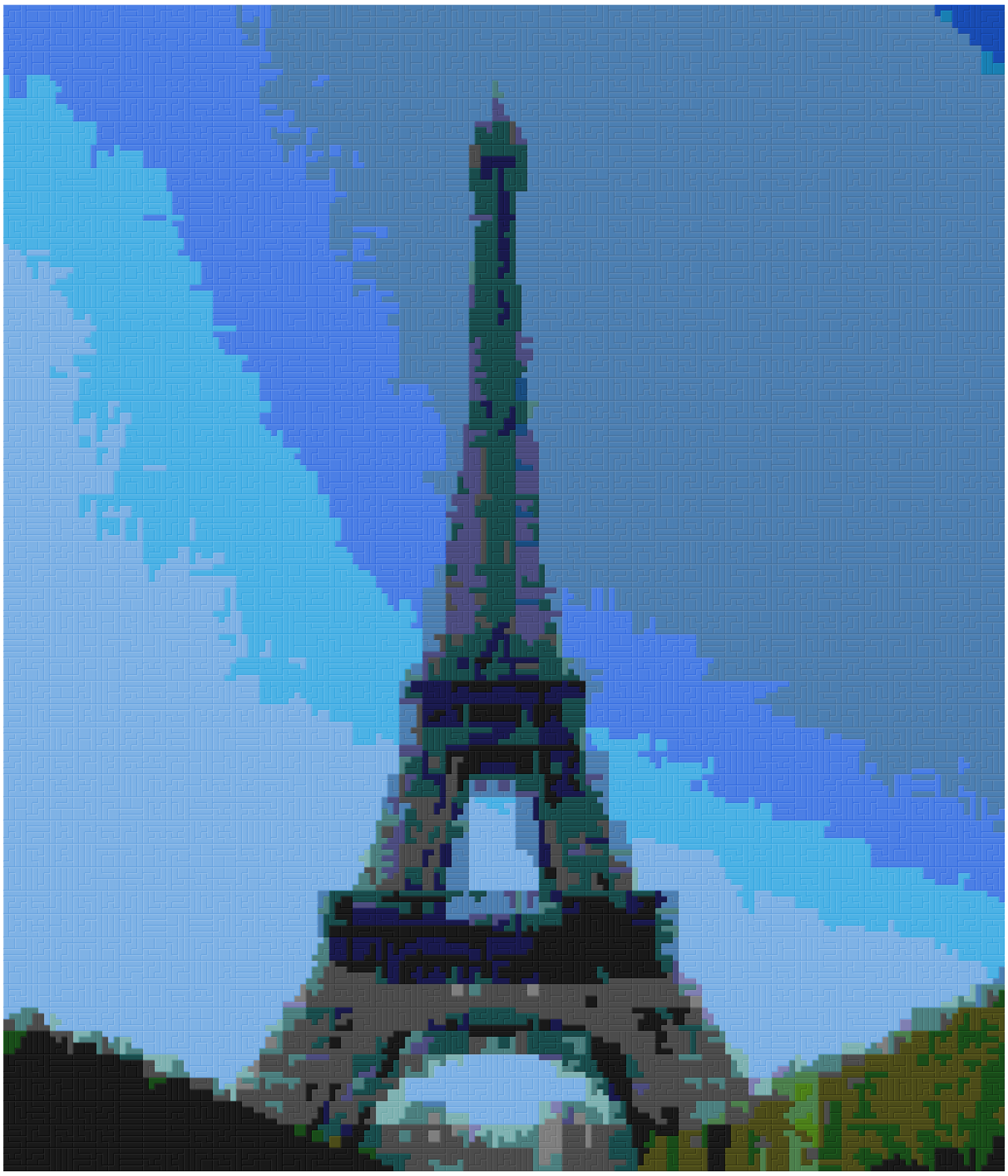
(a)

Figure 6.4 An experimental result. (a) A tetromino-based mosaic image. (b) A watermarked image created from (a) with the watermark shown in Figure 6.2 embedded. (c) A recovered image created with a right key. (d) A recovered image created with a wrong key.



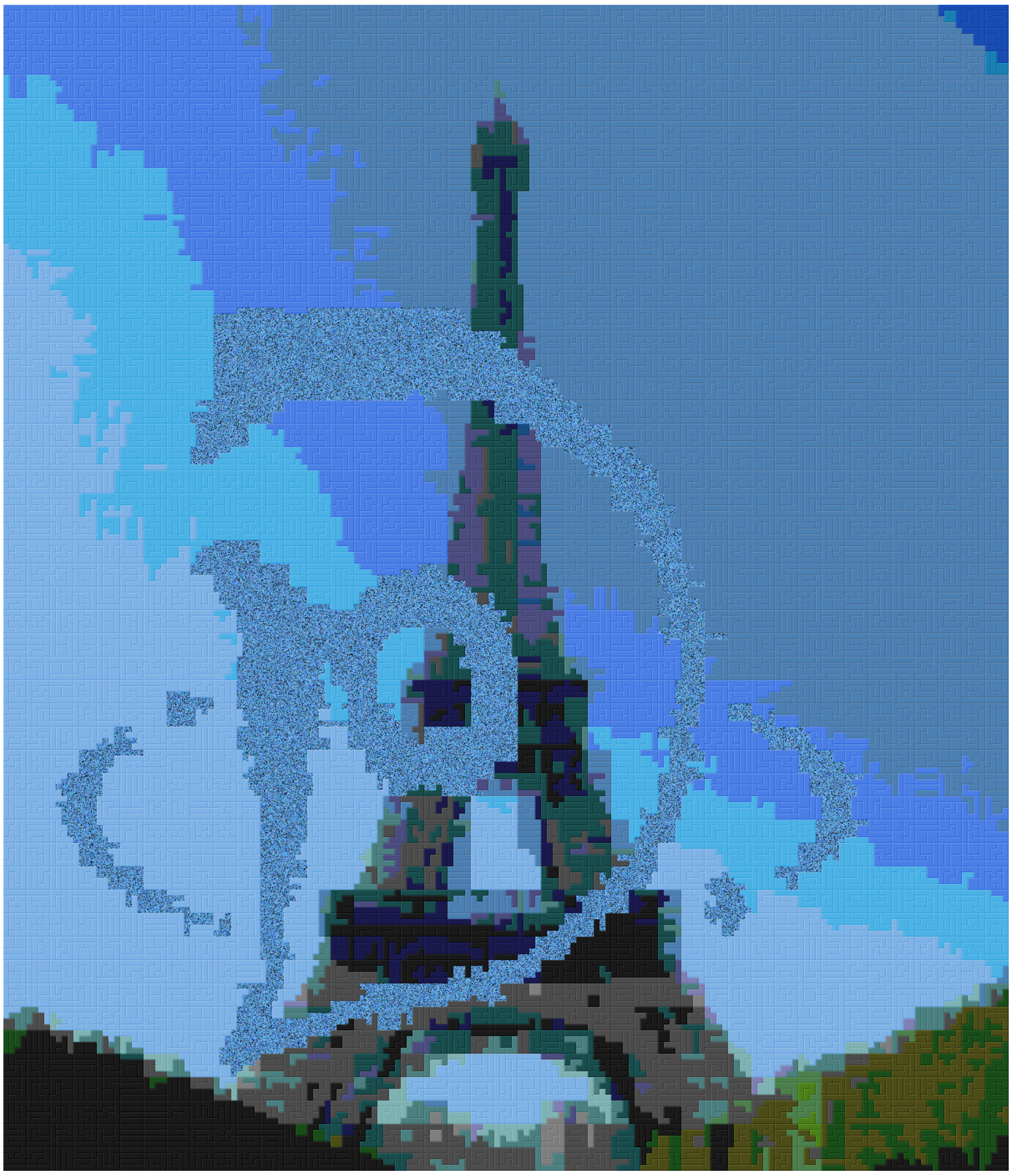
(b)

Figure 6.4 An experimental result. (a) A tetromino-based mosaic image. (b) A watermarked image created from (a) with the watermark shown in Figure 6.2 embedded. (c) A recovered image created with a right key. (d) A recovered image created with a wrong key (continued).



(c)

Figure 6.4 An experimental result. (a) A tetromino-based mosaic image. (b) A watermarked image created from (a) with the watermark shown in Figure 6.2 embedded. (c) A recovered image created with a right key. (d) A recovered image created with a wrong key (continued).



(d)

Figure 6.4 An experimental result. (a) A tetromino-based mosaic image. (b) A watermarked image created from (a) with the watermark shown in Figure 6.2 embedded. (c) A recovered image created with a right key. (d) A recovered image created with a wrong key (continued).

6.5 Discussions and Summary

In this chapter, a removable lossless visible watermarking method based on Chen and Tsai’s method [32] has been proposed. Using the properties of tetromino-based mosaic images, we can embed a binary watermark into a cover image by replacing the colors according to a mapping between two color palettes. In the mean time, the binary watermark is also embedded in the watermarked image for the use in the later recovery process. As a result, it can be used in the copyright protection of tetromino-based mosaic images. Then, a process of lossless recovery of the original tetromino-based mosaic images which are used to embed watermarks is performed by an inverse process of the watermarking method. Authorized users can remove the embedded watermark losslessly from the watermarked image without any additional information. Some experimental results were generated by the proposed watermarking algorithms to prove the feasibility of the proposed method.

Chapter 7
Image Steganography in Tetromino-based Mosaic Images by Watermarking

7.1 Idea of Proposed Method

Steganography is a science of communicating secret data between senders and receivers. By the senders, the secret data are embedded into the files of certain forms, called *cover media*, to create *stego-media* as camouflages. Then, the stego-media are transmitted through public channels, such as the Internet. The receiver acquires the stego-media from the channels and extracts the secret data from the stego-media. The advantage of steganography over cryptography is that the behavior of the communication is evident but the content of the communication is camouflaged. Invaders who intend to steal the secret may consider that these stego-media are ordinary files and ignore them easily.

As the name suggests, *image steganography* means that images are used both as the secret data and as the cover media, to perform steganography. In this chapter, we will describe the proposed image steganographic method by a watermarking technique based on the method which has been described in Chapter 6.

7.2 Proposed Method for Image Steganography in Tetromino-based Mosaic Images by Watermarking

7.2.1 Image Embedding Process

In this section, we describe the proposed image embedding process for image steganography by watermarking. At the beginning, an adapted weighted Euclidean distance derived from Formula 6.2 is proposed. A traditional Euclidean distance between two colors C_1 and C_2 and the adapted one for a color C_i in a color palette are defined in Formulas (7.1) and (7.2) , respectively, below:

$$\mu(C_1, C_2) = \sqrt{(R_1 - R_2)^2 + (G_1 - G_2)^2 + (B_1 - B_2)^2}\,; \qquad (7.1)$$

$$\mu(C_i) = \frac{\sum_{n=0}^{m} O_n \times \mu(C_i, C_n)}{\sum_{n=0}^{m} O_n}. \qquad (7.2)$$

By these two formulas, a process for generation of two color palettes, P_s and P_r, and a one-to-one mapping between them are described in Algorithm 7.1. Next, we embed a secret image into the watermark area by replacing the colors of the watermark area according to the mapping between P_s and P_r. The detailed algorithm of the proposed image embedding process by watermarking is described in Algorithm 7.2.

Algorithm 7.1. Process for generation of two color palettes.

Input: a cover palette P_c, a secret palette P_t, a set of colors $C = \{C_0, C_1, \ldots, C_m\}$ of P_c, a set of occurrences $O = \{O_0, O_1, \ldots, O_m\}$ of C, a set of colors $C' = \{C_0',$

C_1', …,C_n'} of P_t, and a set of occurrences O'= {O_0', O_1', …,O_n'} of C'.

Output: a sorted color palette P_s and a rearranged color palette P_r.

Steps:

Step 1. Sort P_t by the occurrences O' in a descending order to obtain a sorted color palette P_s.

Step 2. Create a rearranged color palette P_r, which has the same size of P_s.

Step 3. Use the colors C of P_c and the occurrences O of C to calculate weighted Euclidean distances D_w of all colors in P_s by Formula 7.1.

Step 4. Use D_w to choose a color C_f, which has the largest value of D_w, to be the first color of P_r.

Step 5. Calculate Euclidean distances D between C_f and the other colors in P_s by Formula 6.2.

Step 6. Use D to construct P_r by rearranging the colors in P_s in the order from the closest color of C_f to the farthest color of C_f.

In Step 3 of Algorithm 7.1 above, different from Algorithm 6.1, the adapted weighted Euclidean distances D_w of all the colors of the sorted palette are computed using the colors C of the cover palette P_c and the occurrences O of C. The reason for doing this is that we want to hide the secret image in the watermark area by replacing its colors in such a way to make the watermark area look more obvious. The detail of the proposed image embedding process is described in Algorithm 7.2.

Algorithm 7.2. Image embedding process for image steganography.

Input: a tetromino-based mosaic image I, a binary watermark W, a secret image S, and a secret key K.

Output: a watermarked stego-tetromino-based mosaic image I'.

Steps:

Step 1. For each pixel p in I, perform the following steps.

1.1. If p is in the watermark area, add p into a set of black embedded pixels I_b.

1.2. If p is in the non-watermark area, add p into a set of white embedded pixels I_w.

Step 2. Create a cover color palette P_c by counting the colors $C = \{C_0, C_1, \ldots, C_m\}$ from I and denote the number of color elements of P_c as m.

Step 3. Create a secret color palette P_t by counting the colors $C' = \{C_0', C_1', \ldots, C_n'\}$ from S and denote the number of color elements of P_t as n.

Step 4. For every pixel in I_w, count the occurrences $O = \{O_0, O_1, \ldots, O_n\}$ of colors $C = \{C_0, C_1, \ldots, C_n\}$, respectively.

Step 5. For every pixel in S, count the occurrences $O' = \{O_0', O_1', \ldots, O_n'\}$ of colors $C' = \{C_0', C_1', \ldots, C_n'\}$, respectively.

Step 6. Use P_c, P_t, C, C', O and O' to generate a sorted color palette P_s and a rearranged color palette P_r by Algorithm 7.1.

Step 7. Obtain a width w and a height h from the secret image S.

Step 8. Divide I_b into two areas, and name them a secret area I_s and a non-secret area I_n which have sizes of $w \times h$ and $|I_b| - w \times h$, respectively.

Step 9. For each pixel p in I_s and each pixel q in S, embed S into I by the following steps.

9.1. Obtain a color C_p of p and a color C_q of q.

9.2. Search C_q in P_s to obtain a color index i of C_q.

9.3. Obtain a color C_q' from P_r by the index i.

9.4. Replace the color C_p of p with C_q'.

Step 10. Randomly pair every two pixels of I_b together using K to form $|I_b|/2$ pairs of pixels A.

Step 11. For each pair a in A, swap the two colors of a to generate a watermarked image I'.

Step 12. For each tetromino in I_w, hide W, w, h, and P_s in I_w by the data hiding process using small tetromino color shiftings described by Algorithm 5.1.

7.2.2 Image Extraction Process

The proposed image extraction process is an inverse process of the image embedding process. The detail of the proposed image extraction process is described as follows.

Algorithm 7.3. Image extraction process for image steganography.

Input: a watermarked stego-tetromino-based mosaic image I.

Output: a secret image S.

Steps:

Step 1. Extract a binary watermark W, a width w, a height h, and a sorted palette P_s from I by the lossless data recovery process described by Algorithm 5.2.

Step 2. For each tetromino t in I, perform the following steps.

2.1. If t is in the watermark area, add all pixels in t into a set of black embedded pixels I_b.

2.2. If t is in the non-watermark area, add all pixels in t into a set of white embedded pixels I_w.

Step 3. Create a cover palette P_c by counting the colors $C = \{C_0, C_1, \ldots, C_n\}$ in I_w and denote the number of color elements of P_c as n.

Step 4. For every pixel in I_w, count the occurrences $O = \{O_0, O_1, \ldots, O_n\}$ of colors $C = \{C_0, C_1, \ldots, C_n\}$, respectively.

Step 5. Use P_c, C, C′, O and O' to generate a rearranged color palette P_r by Algorithm 7.1.

Step 6. Randomly pair every two pixels of I_b together using K to form $|I_b|/2$ pairs of pixels, A.

Step 7. For each pair a in A, swap the two colors of a.

Step 8. Divide I_b into two areas, and name them a secret area I_s and a non-secret area I_n which have sizes of $w \times h$ and $|I_b| - w \times h$, respectively.

Step 9. For each pixel p in I_b, extract each pixel q of S from I to generate the secret image S by the following steps.

9.1. Obtain a color C_p of p.

9.2. Search C_p in P_r to obtain a color index i of C_p.

9.3. Obtain a color C_p' from P_s by the index i.

9.4. Assign the color of q to C_p'.

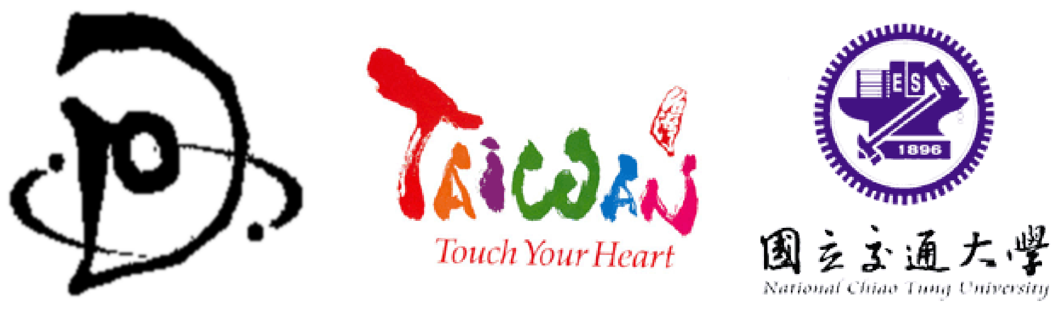
7.3 Experimental Results

Figure 7.1 shows a binary watermark and two secret images which were used in the experiments we conducted to implement the proposed method described previously. Figures 7.2 and 7.3 show some experimental results of applying Algorithms 7.2 and 7.3. Figures 7.2(b) and 7.3(b) are two watermarked stego-tetromino-based mosaic images yielded by the proposed image embedding process (Algorithm 7.2). Figures 7.2(c) and 7.3(c) are two secret images extracted from Figures 7.2(b) and 7.3(b), respectively, with correct keys by the proposed lossless recovery process (Algorithm 7.3). Figures 7.2(d) and 7.3(d) show the two secret images extracted from Figures 7.2(b) and 7.3(b), respectively, with wrong keys.

As seen in these experimental results, secret images can be embedded imperceptibly in the watermark areas, and hidden data can be extracted correctly, by the proposed method.

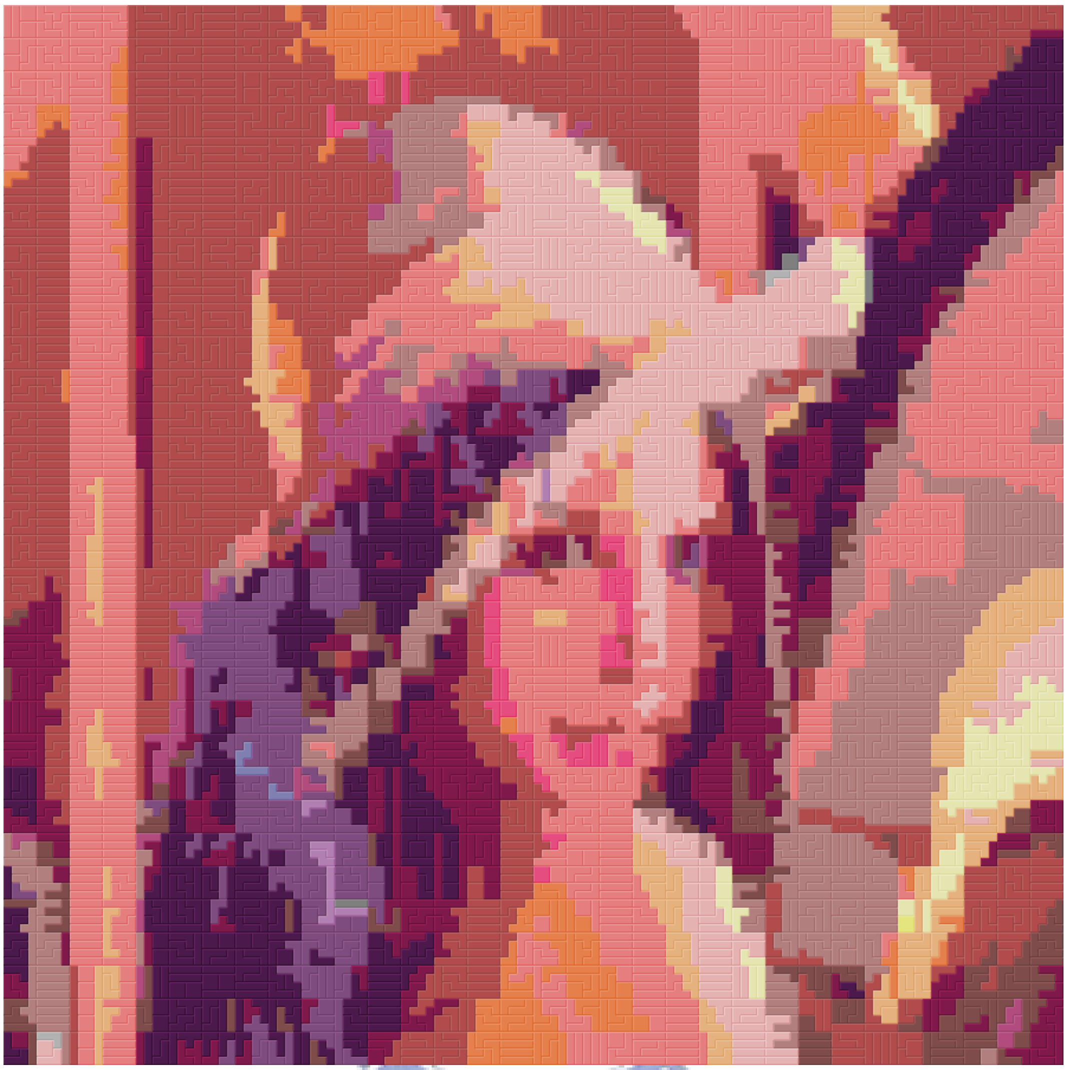
7.4 Discussions and Summary

In this chapter, an image steganographic method based on the proposed watermarking method has been proposed. Using the properties of tetromino-based mosaic images, we can hide a secret image into a cover image by replacing the colors of the watermark area according to a mapping between two color palettes. In the mean time, the binary watermark and a color palette are also embedded in the watermarked image for the use of the later image extraction process. As a result, it can be used for the covert communication via tetromino-based mosaic images. Then, an image extraction process of the secret images is performed by an inverse process of the image embedding method. Users can embed some secret images in tetromino-based mosaic images by watermarking. Some experimental results were generated by the proposed image steganographic method to prove the viability of the proposed method.



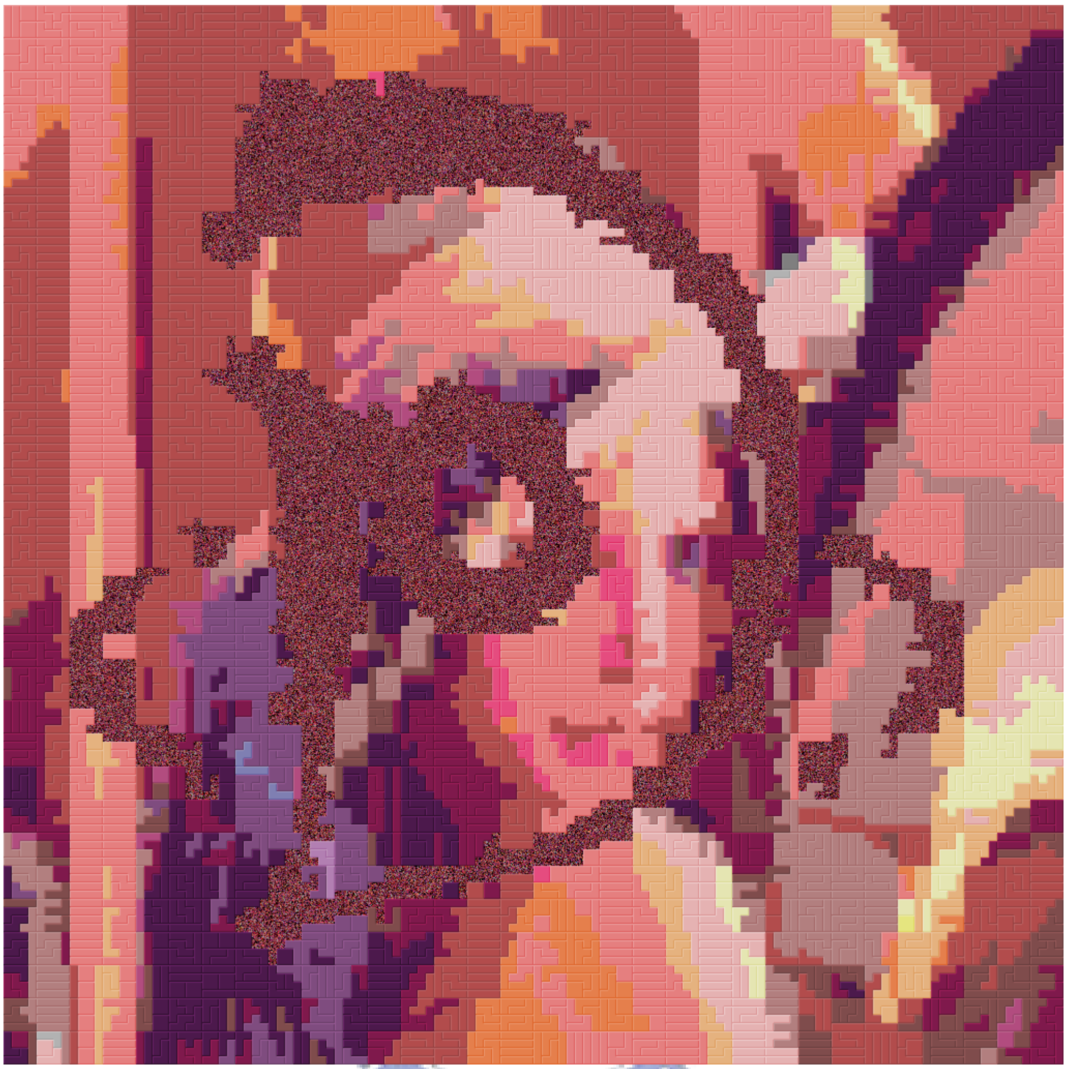
(a) (b) (c)

Figure 7.1 Input images for the proposed method. (a) A binary watermark. (b) A secret image. (c) A secret image.

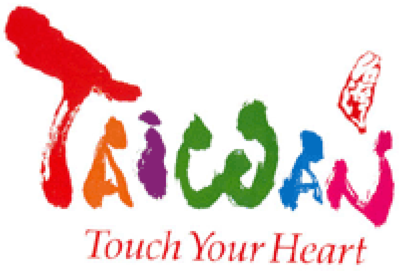


(a)

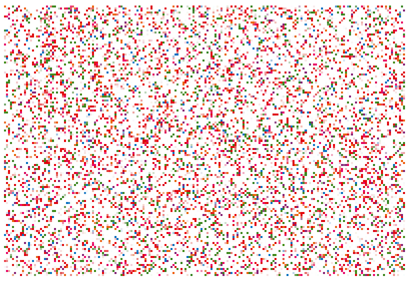
Figure 7.2 An experimental result. (a) A tetromino-based mosaic image. (b) A watermarked stego-image created from (a) with a secret image shown in Figure 7.1(b) embedded. (c) The secret image extracted with a right key. (d) The secret image extracted with a wrong key.



(b)

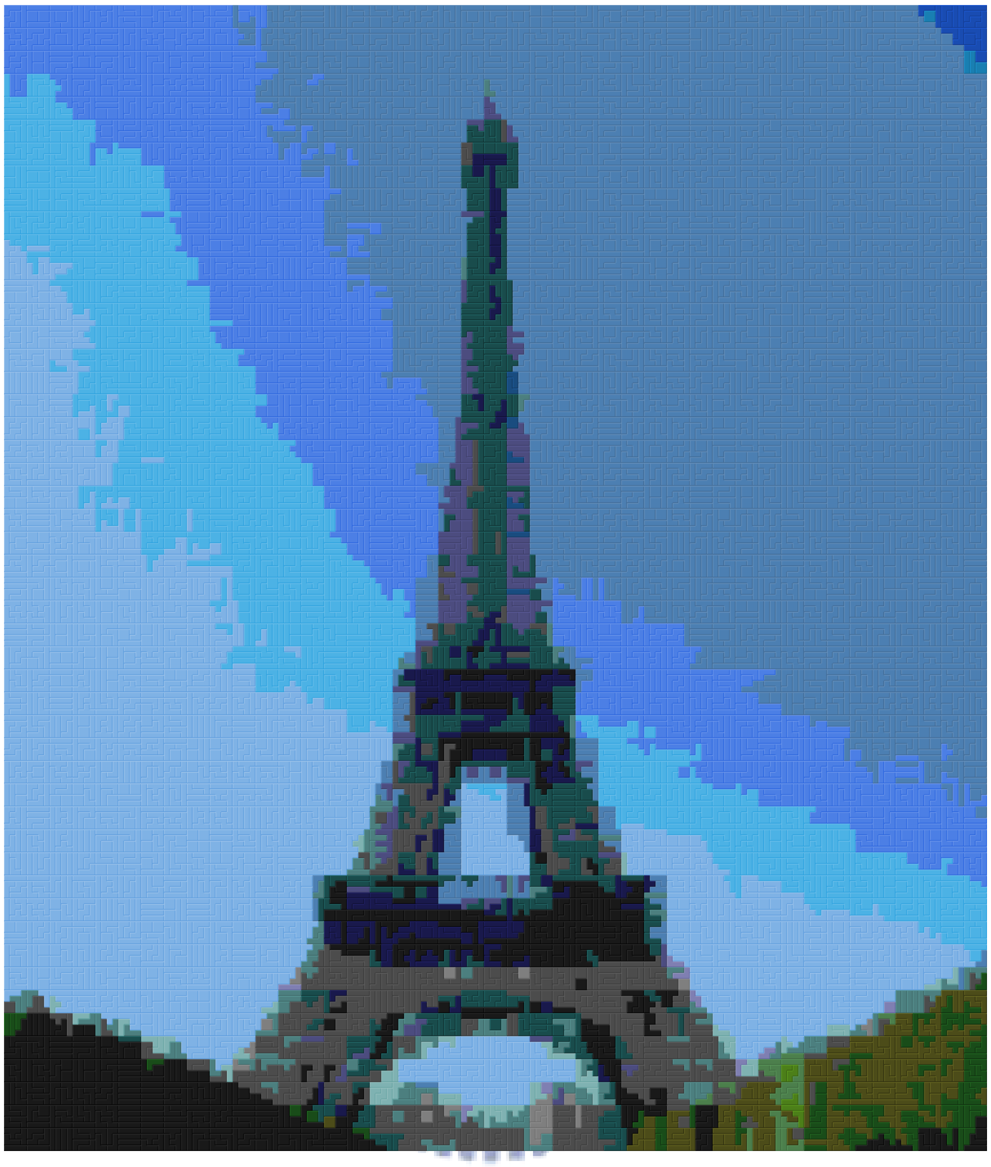


(c)



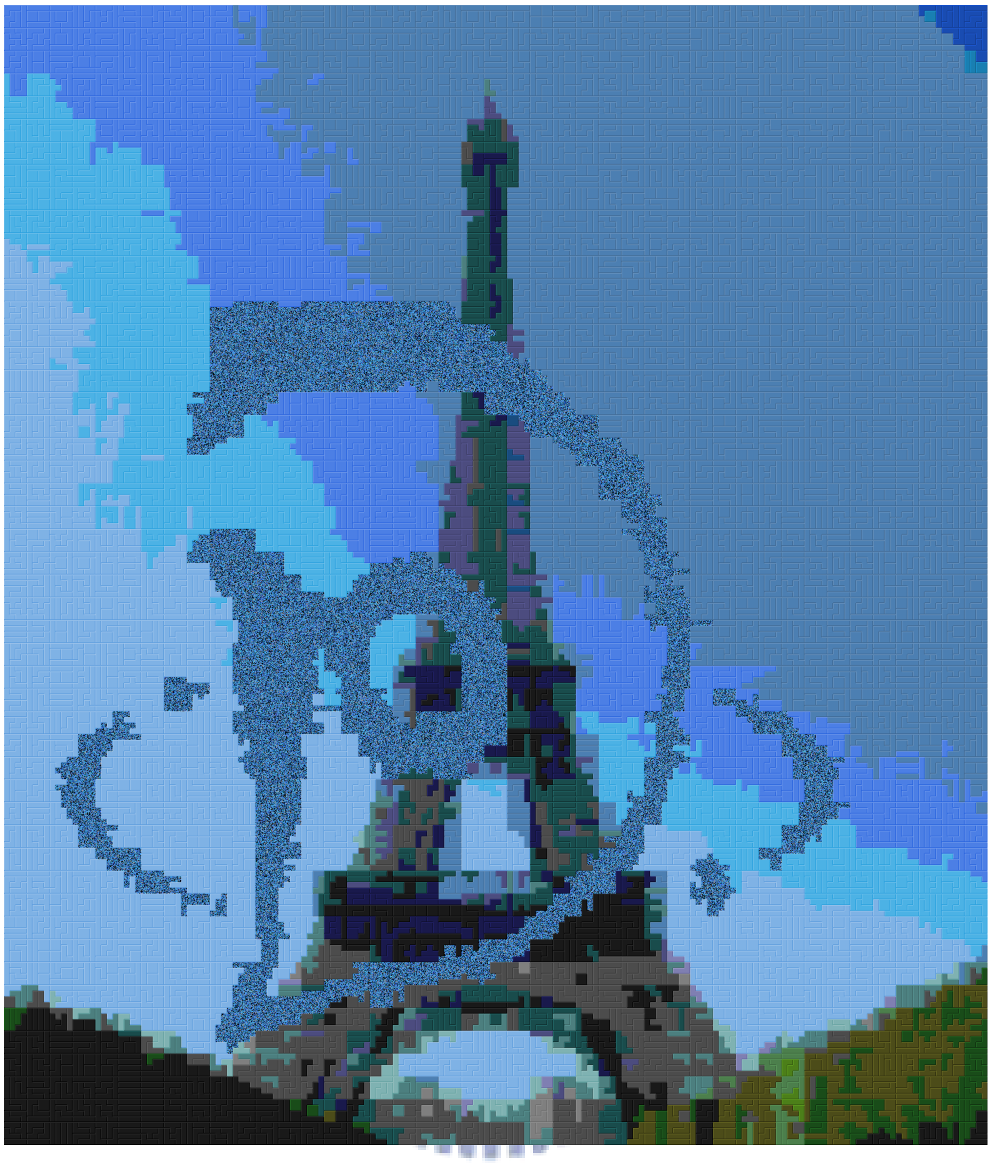
(d)

Figure 7.2 An experimental result. (a) A tetromino-based mosaic image. (b) A watermarked stego-image created from (a) with a secret image shown in Figure 7.1(b) embedded. (c) The secret image extracted with a right key. (d) The secret image extracted with a wrong key (continued).

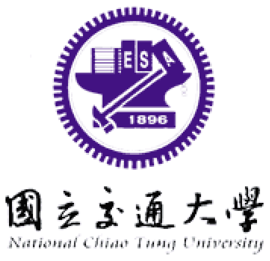


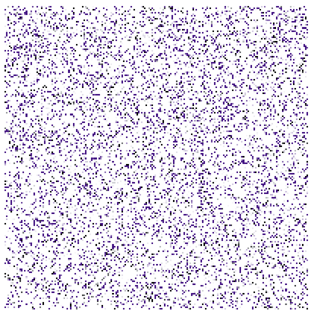
(a)

Figure 7.3 Another experimental result. (a) A tetromino-based mosaic image. (b) A watermarked stego-image created from (a) with a secret image shown in Figure 7.1(c) embedded. (c) The secret image extracted with a right key. (d) The secret image extracted with a wrong key.



(b)





(c) (d)

Figure 7.3 Another experimental result. (a) A tetromino-based mosaic image. (b) A watermarked stego-image created from (a) with a secret image shown in Figure 7.1(c) embedded. (c) The secret image extracted with a right key. (d) The secret image extracted with a wrong key (continued).

Chapter 8
Conclusions and Suggestions for Future Works

8.1 Conclusions

In this study, we have proposed methods for art image creation and data hiding in them. Users can easily generate art images and embed secret data in them simultaneously. The embedded data can be a secret message, a watermark, or a secret image. We can apply these methods for various information hiding applications, such as covert communication, copyright protection, etc.

A novel type of mosaic image which is called *tetromino-based mosaic image* is created in this study. In addition, some information hiding techniques are also developed for covert communication, watermarking, and image steganography applications.

In the creation process of tetromino-based mosaic images, we found out all the possible combinations of tetrominoes by the proposed tree enumerating algorithm. Then, a tetromino database was constructed with different colors and tetromino combinations for the creation of tetromino-based mosaic images. Finally, we also proposed a border enhancement process for improving visual effects in tetromino-based mosaic images.

For data hiding in tetromino-based mosaic images, we utilized two features of the mosaic images to embed data. One is the combinations of tetrominoes, and another is the color values of tetrominoes. We have proposed a data hiding technique by using distinct combinations of tetrominoes. To improve the visual quality of stego-tetromino-based mosaic images, an edge fitting method was used to enhance edge effects by adjusting combinations of tetrominoes using the information obtained by edge detection. In addition, the feature of the colors of tetrominoes was also used to hide data by increasing or decreasing color values of tetrominoes slightly. Using a lossless data recovery process which combines the functions of data extraction and lossless cover image recovery, we can extract data from stego-images and recover the original tetromino-based mosaic image without any distortion.

For watermarking in tetromino-based mosaic images, we have proposed a removable lossless watermarking method by replacing the colors according to a mapping between two color palettes. Based on this invertible watermarking method, we have proposed an image steganographic method by replacing the colors of the watermark area according to the proposed mapping between two color palettes.

According to our research, we can claim that if there is an image feature of an art image which can be modified and detected, a corresponding data hiding technique can be developed. And the three applications of information hiding, copyright protection, covert communication, and image authentication, can be achieved.

8.2 Suggestions for Future Works

In this study, we have proposed a method for the creation of tetromino-based mosaic images, as well as two data hiding techniques for tetromino-based mosaic images. Furthermore, a removable lossless watermarking method and an image steganographic method by watermarking have also been proposed. However, there are still some interesting topics which are worth further study. They are listed as follows.

For tetromino-based mosaic image creation:

1. Creating combinations of tetrominoes with different sizes and tiling styles to make the tetromino-based mosaic images have various visual effects.
2. Enhancing the performance of creation of tetromino-based mosaic images.
3. Applying tetromino-based mosaic images on characters or geometric totems.
4. Producing mosaic images with other shapes based on computer graphics.
5. Enhancing the performance of the tree enumerating algorithm for finding all possible tetromino combinations by applying A^* search algorithm.

For data hiding in tetromino-based mosaic images:

1. Combining two proposed data hiding methods and applying them on mosaic images to enlarge the embedding capacity.
2. Using more tetromino features such as position, tetromino border, etc. for embedding more data into tetromino-based mosaic images.
3. Extending the proposed methods to survive some attacks such as scaling, rotation, JPEG compression, etc.

For watermarking and image steganography in tetromino-based mosaic images:

1. Enhancing the robustness of the proposed watermarking method.
2. Creating different types of watermark effects in tetromino-based mosaic images.
3. Enlarging the data capacity of the proposed image steganographic method by utilizing other features of tetromino-based mosaic images.

References

[1] R. Breukelaar, E. D. Demaine, S. Hohenberger, H. J. Hoogeboom, W. A. Kosters, and David Liben-Nowell, "Tetris is hard, even to approximate," *International Journal of Computational Geometry and Applications*, Vol. 14, pp. 41-68, 2004.

[2] S. Battiato, G. Di Blasi, G. M. Farinella and G. Gallo, "Digital mosaic frameworks - an overview," *Proceedings of Eurographics Italian Chapter,* Catania, Italy, February 2006.

[3] C. Close, and J. Yau, *Recent Paintings*. New York: Pace Wildenstein, 2005.

[4] L. Harmon, "The recognition of faces," *Science American*, Vol. 229, No. 5, pp. 70-82, 1973.

[5] S. Dali, *The Salvador Dali Museum Collection*. Bulfinch Press, 1996.

[6] P. Haeberli, "Paint by numbers: abstract image representations," *Proceedings of SIGGRAPH 90*, pp. 207-214, Dallas, USA, 1990.

[7] A. Hausner, "Simulating decorative mosaics," *Proceedings of SIGGRAPH 01*, pp. 573-580, Los Angeles, USA, August 2001.

[8] K. Hoff, J. Keyser, M. Lin, D. Manocha and T. Culver, "Fast computation of generalized voronoi diagrams using graphics hardware," *Proceedings of SIGGRAPH 99*, pp. 277-286, Los Angeles, USA, August 1999.

[9] S. Lloyd, "Least square quantization in PCM," *IEEE Transactions on Information Theory 28*, pp. 129-137, 1982.

[10] Y. Dobashi, T. Haga, H. Johan and T. Nishita, "A method for creating mosaic images using voronoi diagrams," *Proceedings of Eurographics 02*, pp. 341–348, Saarbrucken, Germany, September 2002.

[11] G. M. Faustino and L. H. De Figueiredo, "Simple adaptive mosaic effects," *Proceedings of SIBGRAPI 2005*, Natal, Brazil, August 2005.

[12] G. D. Blasi and G. Gallo, “Artificial mosaics,” *The Visual Computer*, Vo. 21, pp. 373-383, 2005.

[13] G. Elber and G. Wolberg, “Rendering traditional mosaics,” *The Visual Computer*, Vol. 19, pp. 67-78, 2003.

[14] R. Silvers and M. Hawley, *Photomosaics*. Henry Holt and Company, 1997.

[15] J. Kim and F. Pellacini, “Jigsaw image mosaics,” *Proceedings of SIGGRAPH 02*, pp. 657-664, San Antonio, USA, July 2002.

[16] Y. J. Gi, Y. S. Park, S. H. Seo1 and K. H. Yoon, “Mosaic rendering using colored paper,” *The 7th International Symposium on Virtual Reality, Archaeology and Cultural Heritage(VAST)*, pp. 25-30, Baltimore, USA, October 2006.

[17] N. F. Johnson and S Jajodia, “Exploring steganography: seeing the unseen,” *IEEE Computer*, pp. 26-34, 1998.

[18] Y. C. Chiu and W. H. Tsai, “Copyright protection by watermarking for color images against rotation and scaling attacks using peak detection and synchronization in discrete fourier transform domain,” *Proceedings of 3rd Workshop on Digital Archives Technologies*, pp. 207-213, Taipei, Taiwan, August 2004.

[19] Z. Ni, Y. Q. Shi, N. Ansari, and W. Su, “Reversible data hiding,” *Proceedings. IEEE Int. On Cirsuits and Systems*, Vol. 2, pp. 912-915, Bangkok, Thailand, May 2003.

[20] Y. J. Cheng, C. Y. Yin, D. C. Wu, and W. H. Tsai, “Copyright protection and authentication of image information in digital libraries,” *Proceedings of 2001 Conference on Trends and Perspectives in Digital libraries and Digital Museums*, Vol. 3, pp. 1-11, Hsinchu, Taiwan, 2001.

[21] W. L. Lin and W. H. Tsai, “Data hiding in image mosaics by visible boundary regions and its copyright protection application against print-and-scan attacks,”

Proceedings of International Computer Symposium 2004, Taipei, Taiwan, December 2004.

[22] S. C. Hung, D. C. Wu, and W. H. Tsai, "Data hiding in stained glass images," *Proceedings of 2005 International Symposium on Intelligent Signal Processing and Communications Systems*, pp. 129-132, Hong Kong, December 2005.

[23] C. Y. Hsu and W. H. Tsai, "Creation of a new type of image - circular dotted image - for data hiding by a dot overlapping scheme," *Proceedings of 2006 Conference on Computer Vision, Graphics and Image Processing*, Taoyuan, Taiwan, August 2006.

[24] T. C. Wang and W. H. Tsai, "Creation of tile-overlapping mosaic images for information hiding," *Proceedings of 2007 National Computer Symposium*, pp. 119-126, Taichung, Taiwan, December 2007.

[25] Y. K. Lee and L. H. Chen, "High capacity image steganographic model," *IEE Proceedings of Vision Image Signal Process*, pp. 288-294, June 2000.

[26] D. L. Currie and C. E. Irvine, "Surmounting the effects of lossy compression on steganography," *Proceedings of National Information System Security Conference*, pp. 194-201, Baltimore, USA, October 1996.

[27] J. Fridrich and R. Du, "Secure steganographic methods for palette images," *Proceedings of the Information Hiding Workshop*, Vol. 1768, pp. 61–76, Dresden, Germany, 2000.

[28] L. M. Marvel, C. G. Boncelet Jr. and C. T. Retter, "Spread spectrum image steganography," *IEEE Transactions on Image Processing*, Vol. 8, pp. 1075–1083, 1999.

[29] J. Fridrich and M. Goljan, "Digital image steganography Using Stochastic Modulation," *Proceedings of SPIE Electronic Imaging*, pp. 21–24, Santa Clara, USA, January 2003.

[30] D. C. Wu and W. H. Tsai, “A steganographic method for images by pixel-value differencing,” *Pattern Recognition Letters*, Vol. 24, No. 9-10, pp. 1623-1636, 2003.

[31] J. Canny, “A computational approach to edge detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 8, No. 6, 1986.

[32] P. P. Chen and W. H. Tsai, “Copyright protection of palette images by a robust lossless visible watermarking technique,” *Proceedings of 5th Workshop on Digital Archives Technologies*, Taipei, Taiwan, August 2006.