


國立交通大學

資訊科學與工程研究所

碩士論文

線條化藝術畫之自動產生與其在資訊隱藏上之應用



A Study on Automatic Generations of Line-based
Computer Art Images and Their Applications for
Information Hiding

研究生：劉珊君

指導教授：蔡文祥 教授

中華民國一百年六月

線條化藝術畫之自動產生與其在資訊隱藏上之應用
A Study on Automatic Generations of Line-based Computer Art Images
and Their Applications for Information Hiding

研究生：劉珊君

Student: Shan-Chun Liu

指導教授：蔡文祥

Advisor: Prof. Wen-Hsiang Tsai

國立交通大學
資訊科學與工程研究所
碩士論文



A Thesis
Submitted to Institute of Computer Science and Engineering
College of Computer Science
National Chiao Tung University
In partial Fulfillment of the Requirements
For the Degree of
Master
In
Computer Science

June 2011

Hsinchu, Taiwan, Republic of China

中華民國一百年六月

線條化藝術畫之自動產生與其在資訊隱藏上之應用

研究生：劉珊君

指導教授：蔡文祥 博士

國立交通大學資訊科學與工程研究所

摘要

本論文研究探討了三種藝術影像的自動產生與資訊隱藏技術。這三種不同類型的藝術影像分別是「線條化類立體主義畫作」，「條狀化類未來主義畫作」，與「方塊化類新造型主義畫作」。除了建立一套自動產生這三種藝術畫的系統外，並利用其在影像處理上的特性，各提出了一種資訊隱藏的技術，以達到秘密傳輸之應用。第一種藝術畫—線條化類立體主義畫作的產生方式，是將一張原始影像用霍夫轉換方式找到影像中主要的線條，再用這些線條重新組合出立體單色形塊的新藝術。進而根據人類視覺對平均區塊顏色的敏感度較低這項特性，在維持區塊的顏色平均下，決定各個像素重新填色的方式，來達到藏入秘密訊息的效果。第二種藝術畫—條狀化類未來主義畫作的產生方式，是先將一張原始影像分割成許多單色大區塊，並依其方向特徵加以切割成條狀。另利用此藝術畫留白畫風的特性來產生不同填色的順序，達到藏入秘密訊息的效果。最後一種藝術畫—方塊化類新造型主義畫作的產生方式，是將一張原始影像用二元空間切割方式及相互信息(mutual information)進行遞迴式水平或垂直切割。本論文提出了兩個方法來利用此藝術畫來進行秘密傳輸：第一種是在二元空間分割過程中產生此影像的二元分割樹，利用微調葉節點的區塊平均色方式來達到藏入秘密訊息的效果；第二種是在區塊填色的時候，用二元空間切割方式逐步填色，用不同填色方向來達到藏入秘密訊息的效果。

除了上述的方法外，我們還提出了幾個增加安全性的方法，確保藏入的秘密資訊不被駭客發現。這些方法皆有實驗結果證明它們在視覺方面達到預期效果，以及在資訊隱藏技術上的可行性。

A Study on Automatic Generations of Line-based Computer Art Images and Their Applications for Information Hiding

Student: Shan-Chun Liu

Advisor: Wen-Hsiang Tsai

Institute of Computer Science and Engineering

National Chiao Tung University

ABSTRACT

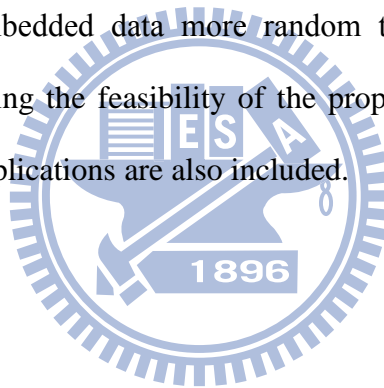
In this study, three kinds of line-type computer art images are created, called line-based Cubism-like image, strip-based Futurism-like image, and rectangle-based Neo-Plasticism-like image. Also proposed are three data hiding techniques for covert communication via these types of art images, respectively.

In the creation of line-based Cubism-like images, the longer line segments in a source image are detected and rearranged to form a new 3D-like shape for each color component in the image. In a process of re-coloring the regions in the new image, a data hiding technique is designed skillfully to embed a secret message into the image by keeping the average color of the region unchanged.

In the strip-based Futurism-like image creation process, the boundary chain codes of each region yielded by image segmentation are utilized to analyze the region characteristics such as corner points and region directions. By drawing the edges formed by the found corner points, an effect of polygon approximation of the prominent regions in the source image is obtained. Then, each region is partitioned into strips in accordance with the extracted region direction, and a given secret message is hidden into the resulting image by coloring the strips with the white color or the region's average color in a random fashion.

To generate a rectangle-based Neo-Plasticism-like image, a binary space-partition scheme is used to partition a given image into multiple rectangles with the maximum mutual information. Furthermore, two methods are proposed to hide a secret message into the generated art image. One is to limit the image partitioning directions (horizontal and vertical) to follow an alternative order and hide the secret messages into the leaf nodes of a partition tree yielded by the binary space-partition scheme. The other method does not limit the partitioning direction order, and fills the regions with horizontal or vertical color lines to embed the secret message.

In addition, for each art image, the user is allowed to select some parameters to create his/her favorite art images. Various security enhancement measures are also proposed to make the embedded data more random to prevent hackers' attacks. Experimental results showing the feasibility of the proposed methods for art image creation and data hiding applications are also included.



ACKNOWLEDGEMENTS

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

Appreciation is also given to the colleagues of the Computer Vision Laboratory in the Institute of Computer Science and Engineering at National Chiao Tung University for their suggestions and help during her thesis study.

Finally, the author also extends her profound thanks to her dear family and boyfriend for their lasting love, care, and encouragement.



CONTENTS

| | |
|------------------------------------|-------------|
| ABSTRACT (in Chinese) | i |
| ABSTRACT (in English) | ii |
| ACKNOWLEDGEMENTS | iv |
| CONTENTS | v |
| LIST OF FIGURES | viii |
| LIST OF TABLES | xiii |

Chapter 1 Introduction1

| | |
|--|----|
| 1.1 Motivation and Background..... | 1 |
| 1.1.1 Motivation of Study..... | 1 |
| 1.1.2 Introduction to Western Art..... | 2 |
| 1.2 General Review of Related Works..... | 5 |
| 1.3 Overview of Proposed Methods..... | 5 |
| 1.3.1 Definitions of Terms..... | 5 |
| 1.3.2 Brief Descriptions of Proposed Methods for Creation of and Hiding Information in Line-based Cubism-like Images..... | 7 |
| 1.3.3 Brief Descriptions of Proposed Methods for Creation of and Hiding Information In Strip-based Futurism-like Images..... | 8 |
| 1.3.4 Brief Descriptions of Proposed Methods for Creation of and Hiding Information in Rectangle-based Neo-Plasticism-like Images..... | 9 |
| 1.4 Contributions..... | 11 |
| 1.5 Thesis Organization..... | 12 |

Chapter 2 Review of Related Works13

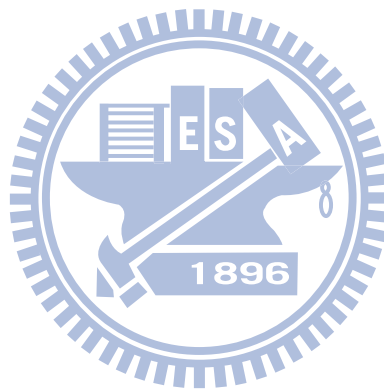
| | |
|---|----|
| 2.1 Previous Studies on Creations and Applications of Computer Art Images..... | 13 |
| 2.2 General Review on Information Hiding Techniques..... | 16 |
| 2.3 Previous Studies on Information Hiding Techniques via Art Images .. | 18 |

Chapter 3 Line-based Cubism-like Image --- A New Type of Image and Its Application to Data Hiding by Invisible Reversible Pixel Re-Coloring22

| | |
|---|----|
| 3.1 Overview of Proposed Method..... | 22 |
| 3.2 Proposed Line-based Cubism-like Image Creation Process..... | 23 |

| | | |
|------------------|---|-----------|
| 3.2.1 | Idea of Proposed Creation Technique..... | 23 |
| 3.2.2 | Proposed Art Image Creation Process | 24 |
| 3.2.3 | Experimental Results..... | 28 |
| 3.3 | Proposed Technique for Data Hiding in Line-based Cubism-like Images by Invisible Reversible Pixel Re-Coloring | 33 |
| 3.3.1 | Idea of Proposed Data Hiding Technique..... | 33 |
| 3.3.2 | Proposed Data Hiding Process | 37 |
| 3.3.3 | Proposed Secret Message Extraction Process | 41 |
| 3.3.4 | Security Consideration | 43 |
| 3.3.5 | Experimental Results..... | 44 |
| 3.4 | Summary | 45 |
| Chapter 4 | Strip-based Futurism-like Image --- A New Type of Image and Its Application to Data Hiding by Variable Sub-Region Coloring | 50 |
| 4.1 | Overview of Proposed Method..... | 50 |
| 4.2 | Proposed Strip-based Futurism-like Image Creation Process | 51 |
| 4.2.1 | Idea of Proposed Creation Technique..... | 51 |
| 4.2.2 | Proposed Art Image Creation Process | 52 |
| 4.2.3 | Experimental Results..... | 60 |
| 4.3 | Proposed Technique for Data Hiding in Strip-based Futurism-like Images by Variable Sub-Region Coloring..... | 64 |
| 4.3.1 | Idea of Proposed Data Hiding Technique..... | 64 |
| 4.3.2 | Proposed Data Hiding Process | 64 |
| 4.3.3 | Proposed Secret Message Extraction Process | 68 |
| 4.3.4 | Security Consideration | 70 |
| 4.3.5 | Experimental Results..... | 70 |
| 4.4 | Summary | 71 |
| Chapter 5 | Rectangle-based Neo-Plasticism-like Image --- A New Type of Image and Its Application to Data Hiding by Binary Space Partitioning..... | 76 |
| 5.1 | Overview of Proposed Methods | 76 |
| 5.2 | Proposed Rectangle-based Neo-Plasticism-like Image Creation Process | 77 |
| 5.2.1 | Idea of Proposed Creation Technique..... | 77 |
| 5.2.2 | Proposed Art Image Creation Process | 81 |
| 5.2.3 | Experimental Results..... | 83 |
| 5.3 | Proposed Technique for Data Hiding in Rectangle-based | |

| | |
|--|------------|
| Neo-Plasticism-like by Binary Space Partitioning | 86 |
| 5.3.1 Idea of Proposed Data Hiding Technique | 86 |
| 5.3.2 Proposed Data Hiding Process | 88 |
| 5.3.3 Proposed Secret Message Extraction Process | 94 |
| 5.3.4 Security Consideration | 97 |
| 5.3.5 Experimental Results | 98 |
| 5.4 Summary | 99 |
| Chapter 6 Conclusions and Suggestions for Future Works..... | 108 |
| 6.1 Conclusions | 108 |
| 6.2 Suggestions for Future Works | 109 |
| References | 112 |



LIST OF FIGURES

| | | |
|------------|--|----|
| Figure 1.1 | Paintings of Cubism. (A) Les Demoiselles D'avignon (“The young ladies of Avignon”), Pablo Picasso, 1907. (B) Oberweimar, Lyonel Feininger, 1921. | 3 |
| Figure 1.2 | Paintings of Futurism. (A) The traveler, Lyubov Sergeyevna Popova, 1915. (B) In the hold, David Garshen Bomberg, 1914. | 4 |
| Figure 1.3 | Neo-Plasticism paintings of Piet Mondrian. (A) Composition A, 1920. (B) Composition with gray and light brown, 1918. | 4 |
| Figure 1.4 | Proposed creation process of line-based Cubism-like image. | 7 |
| Figure 1.5 | Proposed data hiding process by invisible reversible pixel re-coloring. .. | 8 |
| Figure 1.6 | Proposed creation process of strip-based Futurism-like image. | 9 |
| Figure 1.7 | Proposed data hiding process by variable sub-region coloring. | 10 |
| Figure 1.8 | Proposed creation process of rectangle-based Neo-Plasticism-like image. | 10 |
| Figure 1.9 | Proposed data hiding process by the use of a binary partition tree. | 11 |
| Figure 2.1 | The images created by Hertzmann [2] and Hertzmann [3]. (A) An image with the effect of watercolor painting. (B) An image with the effect of oil painting. | 14 |
| Figure 2.2 | Other types of art images. (A) A pen-and-ink drawing from Salisbury [4]. (B) A stipple image from Mould [5]. (C) A stained glass image from Mould [6]. | 14 |
| Figure 2.3 | Some tile mosaic images created by Hausner [7]. | 15 |
| Figure 2.4 | Images created by Haeberli’s method [8]. | 15 |
| Figure 2.5 | Images created by Song, et al. [9]. | 16 |
| Figure 2.6 | Image mosaics. (A) An image mosaic created from Lin and Tsai [15]. (B) An image mosaic created from Wang and Tsai [16]. | 19 |
| Figure 2.7 | A secret-fragment-visible mosaic image created with Lai and Tsai’s method [17]. (A) Original image. (B) Generated secret-fragment-visible mosaic image. | 19 |
| Figure 2.8 | Art images created by Hsu and Tsai [18]. (A) A digital puzzle image. (B) A digital pointillistic image. (C) A digital circular-dotted image. | 20 |
| Figure 2.9 | Two examples of art images. (A) A stained glass image from Hung and Tsai [19]. (B) A tetromino-based mosaic from Chang and Tsai [20]. | 21 |
| Figure 3.1 | Process of crossing-image line creation. | 26 |
| Figure 3.2 | An experimental result of varying the threshold values of D_{\min} and L_{\min} . (A) A source image with size 1024×768. (B) A Cubism-like image | |

created from (A) With initial $D_{\min} = 102$ and initial $L_{\min} = 102$. (C) A Cubism-like image created from (A) with $D_{\min} = 20$ and $L_{\min} = 102$. (D) A Cubism-like image created from (A) with $D_{\min} = 102$ and $L_{\min} = 20$. (E) A Cubism-like image created from (A) with $D_{\min} = 200$ and $L_{\min} = 102$. (F) A Cubism-like image created from (A) with $D_{\min} = 102$ and $L_{\min} = 200$27

Figure 3.3 Experimental images. (A) A source image with size 1024×768. (B) Initial $D_{\min} = 102$ and initial $L_{\min} = 102$. (C) $(D_{\min}, L_{\min}) = (51, 51)$. (D) $(D_{\min}, L_{\min}) = (51, 102)$. (E) $(D_{\min}, L_{\min}) = (51, 204)$. (F) $(D_{\min}, L_{\min}) = (102, 51)$. (G) $(D_{\min}, L_{\min}) = (102, 102)$. (H) $(D_{\min}, L_{\min}) = (102, 204)$. (I) $(D_{\min}, L_{\min}) = (204, 51)$. (J) $(D_{\min}, L_{\min}) = (204, 102)$. (K) $(D_{\min}, L_{\min}) = (204, 204)$. (L) A better choice of 9 images to fit the abstract style of Figure 3.3(A) is $D_{\min}=102$ and $L_{\min}=51$29

Figure 3.4 Experimental images. (A) A source image with size 1024×768. (B) Initial $D_{\min} = 102$ and initial $L_{\min} = 102$. (C) $(D_{\min}, L_{\min}) = (51, 51)$. (D) $(D_{\min}, L_{\min}) = (51, 102)$. (E) $(D_{\min}, L_{\min}) = (51, 204)$. (F) $(D_{\min}, L_{\min}) = (102, 51)$. (G) $(D_{\min}, L_{\min}) = (102, 102)$. (H) $(D_{\min}, L_{\min}) = (102, 204)$. (I) $(D_{\min}, L_{\min}) = (204, 51)$. (J) $(D_{\min}, L_{\min}) = (204, 102)$. (K) $(D_{\min}, L_{\min}) = (204, 204)$. (L) A better choice of 9 images to fit the abstract style of Figure 3.4(A) is $D_{\min}=102$ and $L_{\min}=51$30

Figure 3.5 Experimental images. (A) A source image with size 768×1024. (B) Initial $D_{\min} = 102$ and initial $L_{\min} = 102$. (C) $(D_{\min}, L_{\min}) = (51, 51)$. (D) $(D_{\min}, L_{\min}) = (51, 102)$. (E) $(D_{\min}, L_{\min}) = (51, 204)$. (F) $(D_{\min}, L_{\min}) = (102, 51)$. (G) $(D_{\min}, L_{\min}) = (102, 102)$. (H) $(D_{\min}, L_{\min}) = (102, 204)$. (I) $(D_{\min}, L_{\min}) = (204, 51)$. (J) $(D_{\min}, L_{\min}) = (204, 102)$. (K) $(D_{\min}, L_{\min}) = (204, 204)$. (L) A better choice of 9 images to fit the abstract style of Figure 3.5(A) is $D_{\min}=102$ and $L_{\min}=102$32

Figure 3.6 The process of transforming M' into M'' with a length of a multiple of three by appending an ending pattern.38

Figure 3.7 An experimental result. (A) A source image (cover image). (B) A Cubism-like image without secret message embedding. (C) A stego-image of (A) by embedding the secret message “Hi, I am Helen. Nice to meet you!” with the secret key “door.”46

Figure 3.8 Extracting the secret message with the right secret key “door.”47

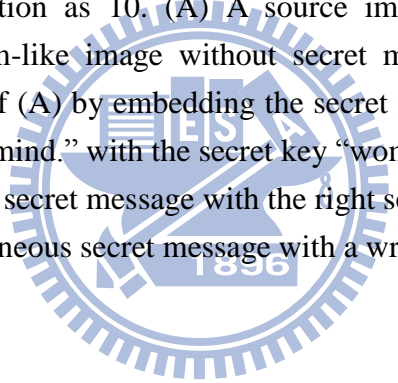
Figure 3.9 Extracted erroneous secret message with a wrong key “doo.”47

Figure 3.10 An experimental result. (A) A source image (cover image). (B) A Cubism-like image without secret message embedding. (C) A stego-image of (A) by embedding the secret message “Meet me at 21:30.

| | | |
|-------------|--|----|
| | See you.” with the secret key “test.” | 48 |
| Figure 3.11 | Extracting the secret message with the right secret key “test.” | 49 |
| Figure 3.12 | Extracted erroneous secret message with a wrong key “tete.” | 49 |
| Figure 4.1 | Chain codes. (A) The eight directions given by Freeman chain codes [23]. (B) The new eight directions used in this study. | 53 |
| Figure 4.2 | An example of looping error of chain codes. (A) A case of chain codes which yields a looping error. the pink grid is the start pixel of chain code tracing. (B) An erosion result of (A). (C) A dilation result of (B). | 54 |
| Figure 4.3 | An example of finding corner points. (A) An example of chain codes. (B) The direction changes in X coordinate. (C) The direction changes in Y coordinate. (D) The result of connecting all corner points. | 56 |
| Figure 4.4 | An example of polygon approximation by using the chain codes. (A) A source image. (B) The result of image segmentation by region merging of (A). (C) The result of polygon approximation by connecting the corner points of each region of (B). | 61 |
| Figure 4.5 | Experimental images. (A) A source image. (B) A Futurism-like image created by (A) with strip width as small. (C) A Futurism-like image created by (A) with strip width as middle. (D) A Futurism-like image created by (A) with strip width as large. | 61 |
| Figure 4.6 | Experimental images. (A) A source image. (B) A Futurism-like image created by (A) with strip width as small. (C) A Futurism-like image created by (A) with strip width as middle. (D) A Futurism-like image created by (A) with strip width as large. | 62 |
| Figure 4.7 | Experimental images. (A) A source image. (B) A Futurism-like image created by (A) with strip width as small. (C) A Futurism-like image created by (A) with strip width as middle. (D) A Futurism-like image created by (A) with strip width as large. | 63 |
| Figure 4.8 | Experimental results of drawing strips in another way. (A) Three source images. (B) The experimental results. | 63 |
| Figure 4.9 | An experimental result. (A) A source image (cover image). (B) A Futurism-like image without secret message embedding. (C) A stego-image of (A) by embedding the secret message “Make hay while the sun shines.” with the secret key “Sun.” | 72 |
| Figure 4.10 | Extracting the secret message with the right secret key “Sun.” | 73 |
| Figure 4.11 | Extracted erroneous secret messages with a wrong key “SunSun.” | 73 |
| Figure 4.12 | An experimental result with strip width as middle. (A) A source image (cover image). (B) A Futurism-like image without secret message embedding. (C) A stego-image of (A) by embedding the secret message | |

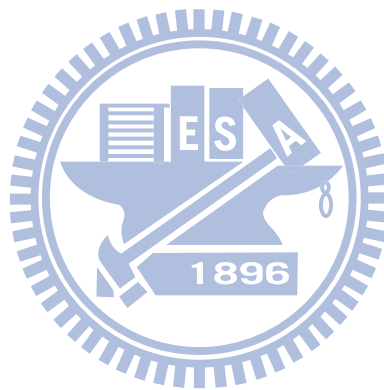
| | | |
|-------------|--|-----|
| | “Learning makes life sweet.” with the secret key “tower.” | 74 |
| Figure 4.13 | Extracting the secret message with the right secret key “tower.” | 75 |
| Figure 4.14 | Extracted erroneous secret messages with a wrong key “tower321.” | 75 |
| Figure 5.1 | An example of partitioning..... | 79 |
| Figure 5.2 | An example about the similarity measure. (A) A source image (B) The result of (A) with the partition iteration to be 30. (C) The result of (A) with the partition iteration to be 30 by using 0.05 as the lower bound of similarity measure..... | 80 |
| Figure 5.3 | Experimental images. (A) A source image. (B) A Neo-Plasticism-like image created by (A) with partition iteration to be 5. (C) A Neo-Plasticism-like image created by (A) with partition iteration to be 10. (D) A Neo-Plasticism-like image created by (A) with partition iteration to be 15..... | 84 |
| Figure 5.4 | Experimental images. (A) A source image. (B) A Neo-Plasticism-like image created by (A) with partition iteration to be 5. (C) A Neo-Plasticism-like Image created by (A) with partition iteration to be 10. (D) A Neo-Plasticism-like image created by (A) with partition iteration to be 15..... | 85 |
| Figure 5.5 | Experimental images. (A) A source image. (B) A Neo-Plasticism-like image created by (A) with partition iteration to be 5. (C) A Neo-Plasticism-like image created by (A) with partition iteration to be 10. (D) A Neo-Plasticism-like image created by (A) with partition iteration to be 15..... | 85 |
| Figure 5.6 | An example of building a partition tree. (A) A partitioned region with the red line as the first partition line. (B) A partition tree which is built according to (A). (C) A partitioned region with alternate order of horizontal and vertical direction. (D) A partition tree which is built according to (C). | 87 |
| Figure 5.7 | An example of coloring by binary space partition order..... | 88 |
| Figure 5.8 | An experimental result by using the first data hiding method with partition iteration as 10. (A) A source image (cover image). (B) A Neo-Plasticism-like image without secret message embedding. (C) A stego-image of (A) by embedding the secret message “Enjoy your own life without comparing it with that of another.” with the secret key “life.” | 100 |
| Figure 5.9 | Extracting the secret message with the right secret key “life.” | 101 |
| Figure 5.10 | Extracted erroneous secret message with a wrong key “live.”..... | 101 |
| Figure 5.11 | An experimental result by using the first data hiding method with | |

| | | |
|-------------|---|-----|
| | partition iteration as 15. (A) A source image (cover image). (B) A Neo-Plasticism-like image without secret message embedding. (C) A stego-image of (A) by embedding the secret message “Life is not an exact science, it is an art.” with the secret key “flower.”..... | 102 |
| Figure 5.12 | Extracting the secret message with the right secret key “flower.” | 103 |
| Figure 5.13 | Extracted erroneous secret message with a wrong key “flowes.” | 103 |
| Figure 5.14 | An experimental result by using the second data hiding method with partition iteration as 8. (A) A source image (cover image). (B) A Neo-Plasticism-like image without secret message embedding. (C) A stego-image of (A) by embedding the secret message “Everybody is a moon, and has a dark side which he never shows to anybody.” with the secret key “moon.” | 104 |
| Figure 5.15 | Extracting the secret message with the right secret key “moon.” | 105 |
| Figure 5.16 | Extracted erroneous secret message with a wrong key “moo.” | 105 |
| Figure 5.17 | An experimental result by using the second data hiding method with partition iteration as 10. (A) A source image (cover image). (B) A Neo-Plasticism-like image without secret message embedding. (C) A stego-image of (A) by embedding the secret message “No beauty is like the beauty of mind.” with the secret key “woman.” | 106 |
| Figure 5.18 | Extracting the secret message with the right secret key “woman.” | 107 |
| Figure 5.19 | Extracted erroneous secret message with a wrong key “womal.” | 107 |



LIST OF TABLES

| | | |
|-----------|---|----|
| Table 4.1 | A table between the direction of chain code and the change of X and Y coordinate..... | 55 |
| Table 4.2 | A mapping table between two bits of original data and three bits of new data..... | 65 |



Chapter 1

Introduction

1.1 Motivation and Background

1.1.1 Motivation of Study

With the development of the computer technology, the Internet is used widely in people's daily life nowadays. Through the Internet, many social network services have been built, including communication, information sharing, etc. For example, many people share their life photos on web albums through websites like Flickr and Picasa. However, the convenient network technologies also bring some drawbacks in communication security. Malicious hackers might steal or tamper with the data contents on the Internet. Besides, certain important information, such as bank account passwords, intelligence data, and so on, is extremely private and should be protected cautiously. In order to avoid divulging secret data, how to protect such information has become an important issue in people's daily life.

On the other hand, the evolution of human civilization is often accompanied with the development of art. For example, in the history of Western art, different periods and different thoughts affect the creation of different art schools. In recent years, unlike the artificial art, some visual arts are produced by computers, creating a new type of art, called *computer art*. An example is *mosaic image* which is composed of small pieces with the form of rectangular blocks or irregular shapes.

Different from traditional information hiding techniques, we try to combine

approaches to data hiding and art image creation for information protection in this study. Using art images as disguises of secret data, people may be attracted to appreciate their artistic contents, and thus ignore other non-artistic properties of the image yielded by information hiding. Therefore, it is advantageous to integrate data hiding techniques into art image creation to hide information.

Specifically, we try to design new techniques for creating new types of computer art. In the Western art history, the characteristics of different art schools have been influenced by one another, so common features exist in some art styles of the schools. For example, one of the common ideas of the Neo-Plasticism, Cubism, and Futurism is the use of the line feature. This idea emphasizes creating line artworks for different reasons like speed, space partitioning, or spatial reorganization. Accordingly, we propose in this study three different kinds of line-based computer art named *line-based Cubism-like image*, *strip-based Futurism-like image*, and *rectangle-based Neo-Plasticism-like image*, respectively. For each type of computer art we design, we propose also a new technique of data hiding to embed secret messages into images of the type during the creation process of them. By using such art images as camouflages, people will tend to believe that an art image of these kinds is only an artistic production and so ignore the secret data embedded in it. Additionally, all the proposed data hiding methods should take advantages of the characteristics of the new types of images.

1.1.2 Introduction to Western Art

The Western art coming mainly from European countries exist for thousands of years. Different art schools were created in different periods with different thoughts, and the features of different art types have been influenced by one another. In the 20th

century, modern art threw the traditions of the past and began to develop in its own way. Modern artists created their artworks with new viewpoints and fresh ideas about the nature of materials. In general, the trend toward abstraction is the main characteristic of the modern art.

For example, Cubism artists break up, analyze, and re-assemble objects in abstract forms. Main artists of Cubism include Pablo Picasso and Lyonel Feininger. Instead of depicting objects from one viewpoint, Cubism describes the subject from multiple viewpoints and represents it in different spaces. By intersecting at random angles of objects, each painting of Cubism seems to be composed of intersecting lines and fragmented shapes. A main characteristic of Cubism is to constitute a new three-dimensional shape of a certain identity by combining lines and objects after destructing its natural form. Some examples of Cubism artworks are shown in Figure 1.1.



(a)



(b)

Figure 1.1 Paintings of Cubism. (a) Les Femmes d'Alger (“The Young Ladies of Avignon”), Pablo Picasso, 1907. (b) Oberweimar, Lyonel Feininger, 1921.

Secondly, Futurism creates essentially artworks of a synthetic style, with more emphasis on technology, movement, and action. Like Cubism, Futurism transforms concrete shapes into abstraction with multiple viewpoints. Moreover, the direction of movement is regarded important in the artwork and is formed by sharp lines. Artists

such as Natalia Goncharova, Lyubov Popova, and David Bomberg are major artists who created pictures of Futurism. Two of their works are shown in Figure 1.2.



(a)



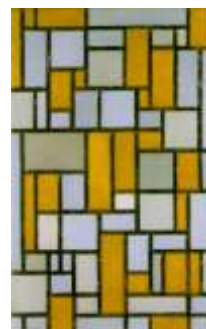
(b)

Figure 1.2 Paintings of Futurism. (a) The Traveler, Lyubov Sergeyevna Popova, 1915. (b) In the hold, David Garshen Bomberg, 1914.

By Neo-Plasticism, the painters of the school focused on simplicity and abstraction, and reduced the image content to geometric shapes by using only straight horizontal and vertical lines and rectangular shapes. More than this, their formal composition was limited to three primary colors — red, yellow, and blue, and three non-primary ones — black, white, and grey. Overall, Neo-Plasticism is the art composed of plane, lines, rectangles, and a limited number of colors. The principal artist of this group is Piet Mondrian. Two artworks of his are shown in Figure 1.3.



(a)



(b)

Figure 1.3 Neo-Plasticism paintings of Piet Mondrian. (a) Composition A, 1920. (b) Composition with Gray and Light Brown, 1918.

1.2 General Review of Related Works

In the past decade, many information hiding techniques have been proposed for various purposes, such as copyright protection, covert communication, authentication, etc. The main idea of data hiding is to embed secret messages imperceptibly into given media, so that in most cases people will not notice the existence of the hidden data. Some of the data hiding techniques were implemented via the use of images. As the art is closely related to human life, the topics of the automatic creation of art images often arouse interests of people. Many techniques, which have been proposed for the creation of computer art, will be reviewed in detail in Chapter 2. In addition, art images can be used for the purpose of data hiding as carriers of information. Some methods combining information hiding with art image creation have been developed in recent years, which will also be reviewed in Chapter 2.

1.3 Overview of Proposed Methods

In this study, we propose methods for creating three new kinds of art images. They are line-based Cubism-like image, strip-based Futurism-like image, and rectangle-based Neo-Plasticism-like image. First of all, a scheme for creation of each type is presented. And for each type, an information hiding method for covert communication is proposed using the specific characteristics of the image creation process. Brief descriptions of these methods are described as follows after some terms are defined.

1.3.1 Definitions of Terms

The definitions of some related terms used in this study are introduced in the

following.

1. *Source image*: a source image is an image chosen to produce a line-based Cubism-like image, a strip-based Futurism-like image, or a rectangle-based Neo-Plasticism-like image.
2. *Computer art image*: a computer art image is a non-photorealistic image created from a source image.
3. *Cubism-like image*: a Cubism-like image is obtained by rearranging the lines yielded by the Hough transform.
4. *Futurism-like image*: a Futurism-like image is an image created by re-partitioning the source image in accordance with the region directions (horizontal, vertical, or diagonal).
5. *Neo-Plasticism-like image*: a Neo-Plasticism-like image is an image composed of rectangular tiles obtained by binary space partitioning.
6. *Cover image*: a cover image is a medium into which a secret message for covert communication is to be embedded.
7. *Stego-image*: A stego-image is produced by embedding a secret message into a cover image.
8. *Creation process*: a creation process produces a Cubism-like image, a Futurism-like image, or a Neo-Plasticism-like image from a secret image.
9. *Embedding process*: an embedding process hides a secret message into a cover image.
10. *Extraction process*: an extraction process retrieves a secret message from a stego-image.

1.3.2 Brief Descriptions of Proposed Methods for Creation of and Hiding Information in Line-based Cubism-like Images

The proposed process for creation of line-based Cubism-like images is illustrated in Figure 1.4. As shown, we find longer line segments in the source image by the Hough transform. Then, we connect the line segments and extend them to the image boundaries. A line-based Cubism-like image is generated after re-coloring the regions which are produced by line rearrangement.

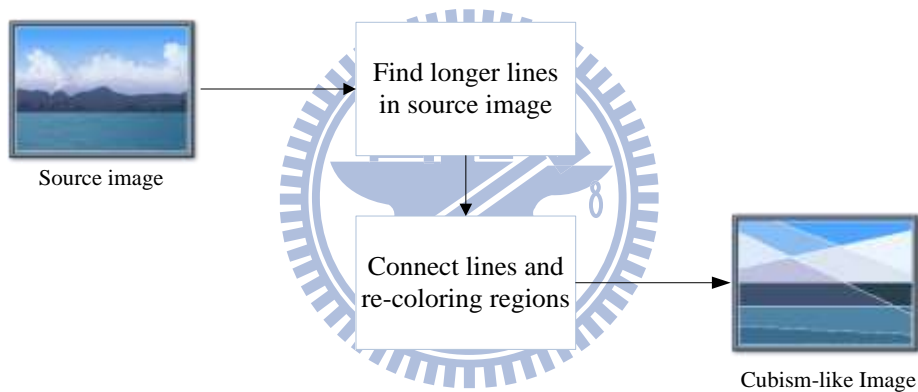


Figure 1.4 Proposed creation process of line-based Cubism-like image.

Based on the above-described creation process, we propose a data hiding technique via line-based Cubism-like images by using an invisible reversible pixel re-coloring technique. The data hiding process is shown in Figure 1.5. First, we transform a secret message to be hidden into a bit string. Secondly, in the re-coloring process, we compute the new color of each pixel to keep the average of the region color unchanged, and re-color the pixel imperceptibly. In this way of invisible reversible pixel re-coloring, a stego-image with the embedded secret message is created. Detailed descriptions of the above processes will be given in Chapter 3.

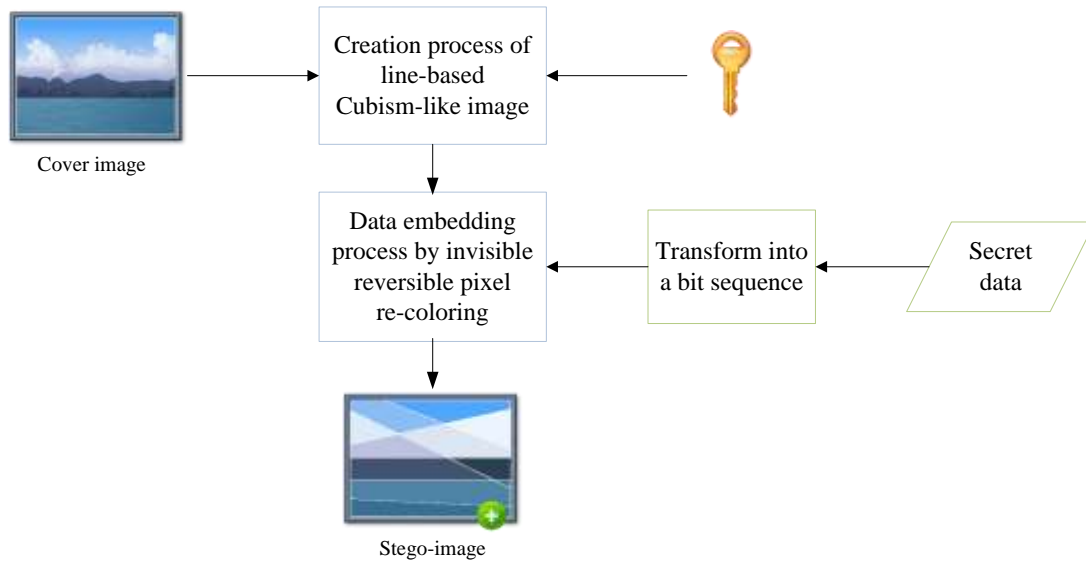


Figure 1.5 Proposed data hiding process by invisible reversible pixel re-coloring.

1.3.3 Brief Descriptions of Proposed Methods for Creation of and Hiding Information in Strip-based Futurism-like Images

The proposed creation process of strip-based Futurism-like images is illustrated in Figure 1.6. First of all, we utilize a region merging scheme for image segmentation. Then, for each region produced, we extract some region characteristics such as the corner points and the region direction by analyzing the chain codes of the region boundary. With the corner points of each region, we obtain a polygon approximation of the region by adjusting the region edge. Finally, we partition each region into strips based on the extracted region direction.

In addition, data hiding into the created Futurism-like art image is achieved in the process of creating the strips in the image. As illustrated in Figure 1.7, the process is similar to the one for hiding data into a line-based Cubism-like image. After partitioning each region into several strips, we hide a given secret message by

coloring the sub-region with white color or the original color. Furthermore, a user key is used to strengthen the security of hidden data by randomizing the processing order of the regions. In Chapter 4, the details of the above processes will be introduced.

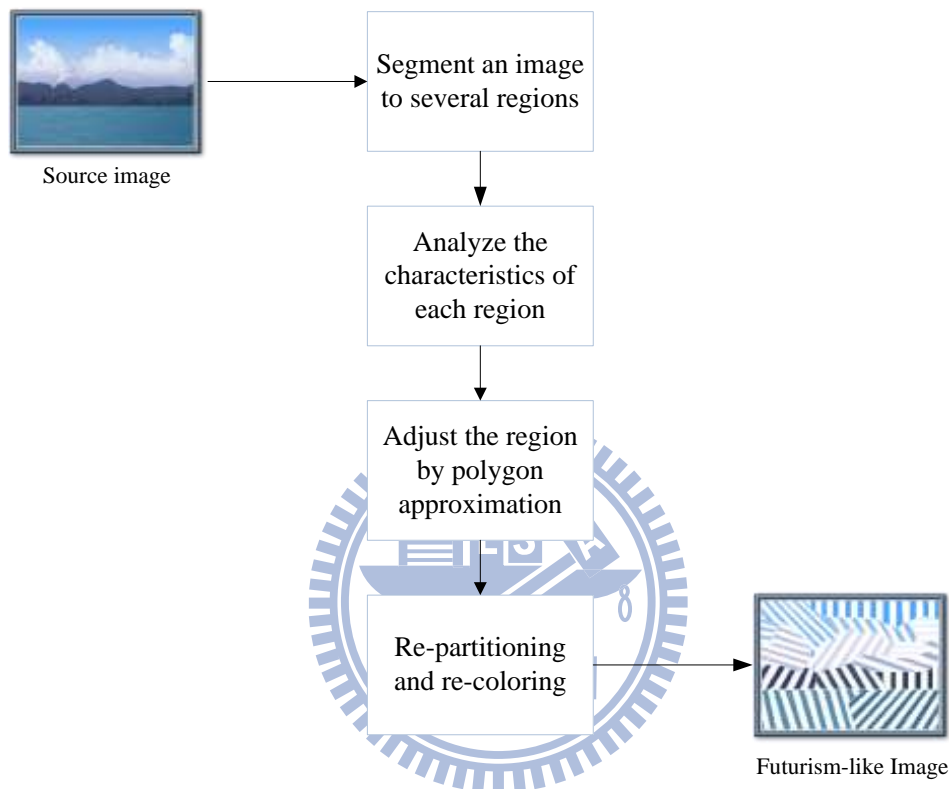


Figure 1.6 Proposed creation process of strip-based Futurism-like image.

1.3.4 Brief Descriptions of Proposed Methods for Creation of and Hiding Information in Rectangle-based Neo-Plasticism-like Images

A method to create rectangle-based Neo-Plasticism image is proposed in this study. It is based on an algorithm of binary space partitioning. As illustrated by Figure 1.8, a binary partition tree is built first by finding the maximum of the mutual information (MI) which is a measure about the intensities and the spatial positions of

the rectangles resulting from the partitioning result of a given image. Each leaf node of the tree is a rectangular region, and a rectangle-based Neo-Plasticism image is created accordingly after coloring the regions. Besides, by different numbers of iterations of partitioning, we can create art images with different abstract levels.

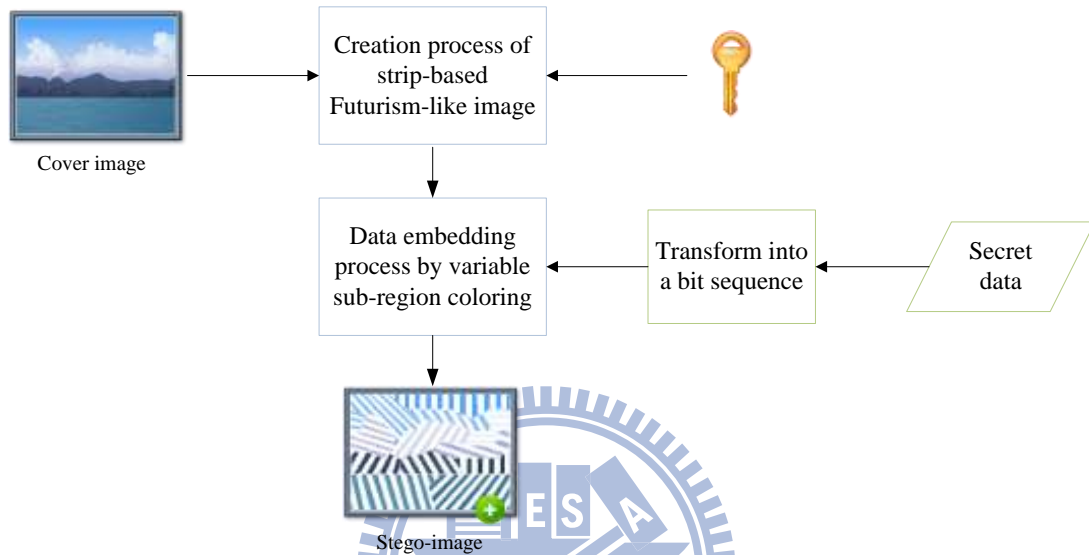


Figure 1.7 Proposed data hiding process by variable sub-region coloring.

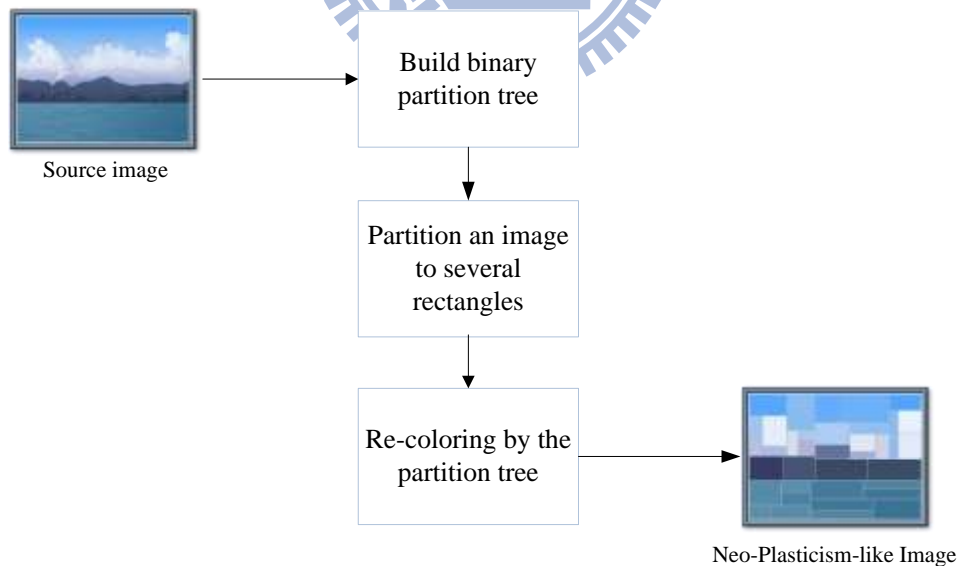


Figure 1.8 Proposed creation process of rectangle-based Neo-Plasticism-like image.

In this art image, we propose two methods to hide the secret messages. The first method of information hiding is described in Figure 1.9. The timing of using a secret

key is different from those in the generation of the above-mentioned two new types of art images. In the generation process of the binary partition tree, we use the key to decide the priority of the growing direction of the tree. Then, we hide the secret message in the leaf nodes of the partition tree in the process of art creation. The second method is to embed secret messages by filling the regions with horizontal or vertical color lines. The detailed algorithms of the above processes will be stated in Chapter 5.

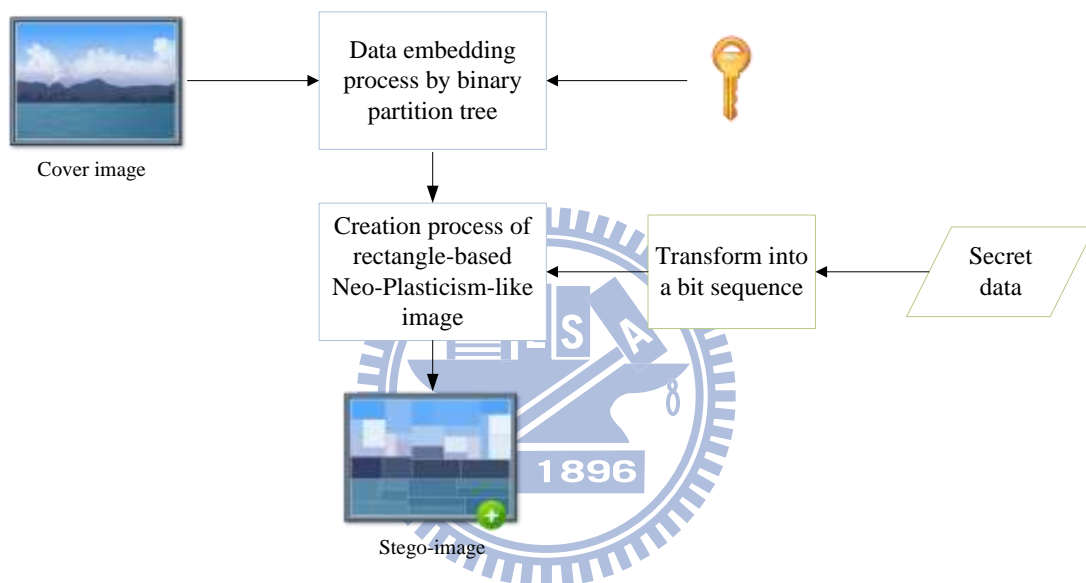


Figure 1.9 Proposed data hiding process by the use of a binary partition tree.

1.4 Contributions

Some major contributions of this study are listed in the following.

1. A method for creation of a new type of art image, called line-based Cubism-like image, is proposed.
2. A method is proposed to create a new type of art image, called strip-based Futurism-like image.
3. A method to create rectangle-based Neo-Plasticism-like images is proposed.
4. A method is proposed to hide data in line-based Cubism-like images by

invisible reversible pixel re-coloring.

5. A method for hiding data in strip-based Futurism-like images by variable sub-region coloring is proposed.
6. A method to embed secret data in rectangle-based Neo-Plasticism-like images by building a limited binary partition tree is proposed.
7. A method is proposed to embed secret data in rectangle-based Neo-Plasticism-like images by coloring the regions with horizontal or vertical direction.
8. A method to enhance the security of the hiding process by randomizing the processing order of the regions is proposed.
9. A method to enhance the security of the hiding process by randomizing the growing direction of the above-mentioned binary partition tree is proposed.

1.5 Thesis Organization

The remainder of this thesis is organized as follows. In Chapter 2, we review the related works about the techniques of data hiding and the creation of art images. In Chapter 3, the proposed method for creation of line-based Cubism-like images and the application of it to covert communication by invisible reversible pixel re-coloring are described. Similarly, we introduce the proposed method for creation of strip-based Futurism-like images and covert communication by variable sub-region coloring via such images in Chapter 4. In Chapter 5, the proposed method for creation of rectangle-based Neo-Plasticism-like images and the technique of information hiding via such images by building the binary partition tree and re-coloring the rectangular regions are described. Finally, conclusions of our study and suggestions for future works are given in Chapter 6.

Chapter 2

Review of Related Works

2.1 Previous Studies on Creations and Applications of Computer Art Images

In recent years, the topic of creating art images via the use of computers often arouses interests of people. More and more researchers investigate the problem of how to combine the computer technology and the art image creation for various applications, from semi-automatically to automatically. With the increasing maturation of computer technologies, creating art image automatically is the main development goal nowadays. Hertzmann [1] surveys many ideas of creating art images by *stroke-based rendering* (SBR) which is defined to be an automatic approach to creating non-photorealistic imagery by placing discrete elements like paint strokes and stipples. He also surveyed several SBR algorithms and styles such as painting, pen-and-ink drawing, tile mosaics, and so on. The common goal of these image styles is to make art images look like some other types of images. For example, two images created by watercolor painting and oil painting in Hertzmann [2] and Hertzmann [3], respectively, are shown in Figure 2.1. Some other types of art images are shown in Figure 2.2, where Figure 2.2(a) is an image created by pen-and-ink illustration proposed by Salisbury [4], Figure 2.2(b) is a stipple image via a stipple placement method proposed by Mould [5], and Figure 2.3(c) shows a stain-glass

image created by an image filter presented in Mould [6].

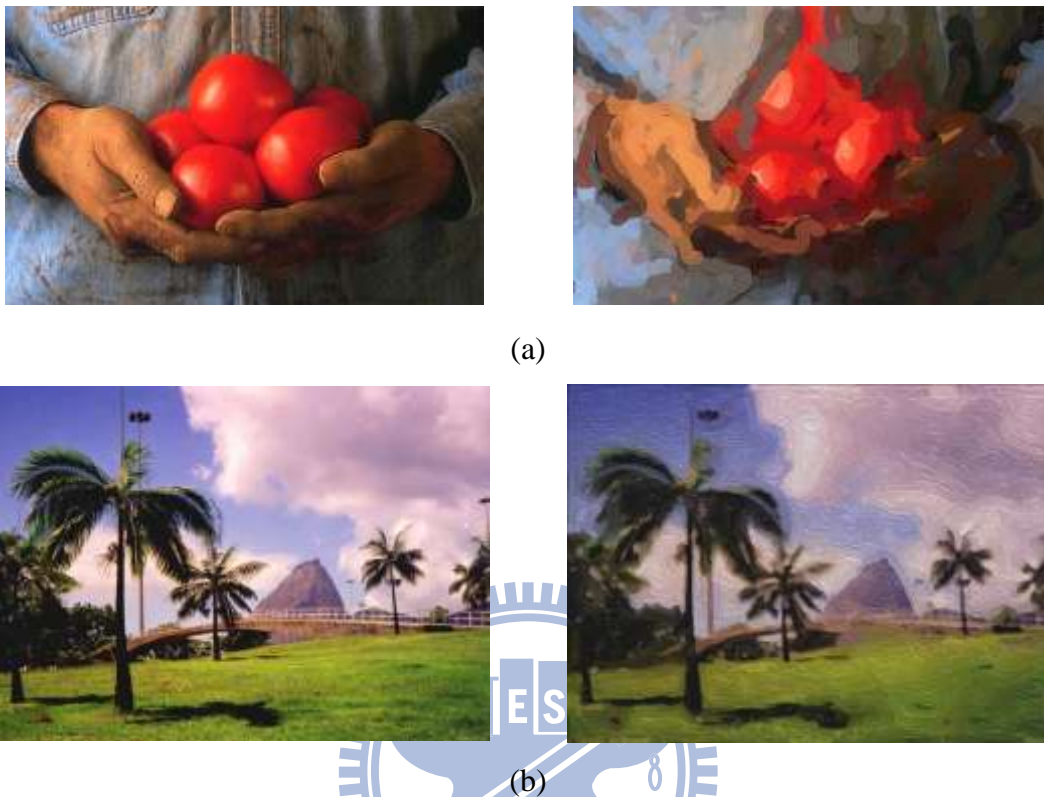


Figure 2.1 The images created by Hertzmann [2] and Hertzmann [3]. (a) An image with the effect of watercolor painting. (b) An image with the effect of oil painting.

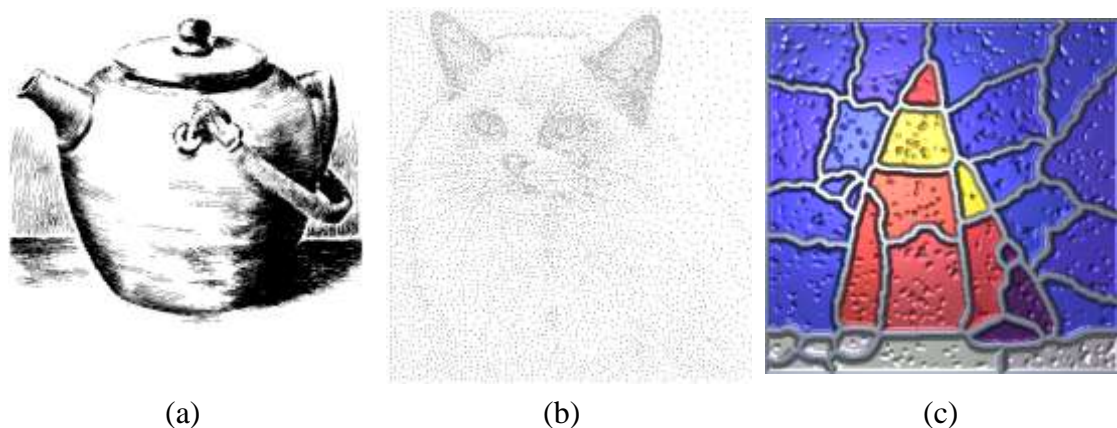


Figure 2.2 Other types of art images. (a) A pen-and-ink drawing from Salisbury [4]. (b) A stipple image from Mould [5]. (c) A stained glass image from Mould [6].

Moreover, another type of art images is mosaic image. Mosaic images are the art of creating works, each being composed of small shapes, such as squares, circles,

triangles, and so on. Different from the fixed direction of mosaic arrangement, Hausner [7] creates a tile mosaic image by placing tiles to follow the edges to make the image smoother. Figure 2.3 shows some examples from Hausner [7].



Figure 2.3 Some tile mosaic images created by Hausner [7].

Another important criterion for art image creation is to limit the number of strokes so that the resulting image looks like an abstract painting, such as the images shown in Figure 2.4 which come from Haeberli [8]. Besides, Song, et al. [9] produces an abstract synthetic art by fitting shapes like triangles or rectangles to regions in segmented images, as shown in Figure 2.5.

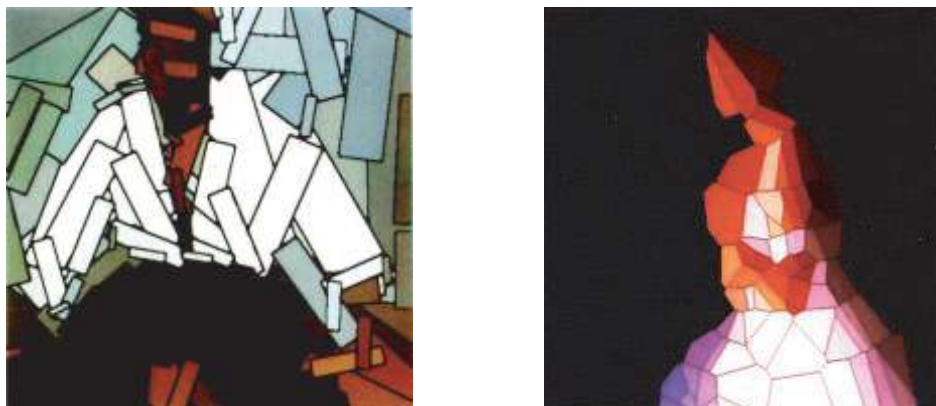


Figure 2.4 Images created by Haeberli's method [8].

In this study, we will focus on creating *abstract-type* images. An abstract-type image does not show a good match to the source image; however, it emphasizes the global trend or distribution of the image. The result will not look like a painting, but keep some properties of the original image. Furthermore, we try to combine some styles of Western art to create our computer art, like the use of the line feature in the Cubism, Futurism, and Neo-Plasticism schools. As a result, three different abstract-type line-dominated art images are generated in this study, namely, line-based Cubism-like image, strip-based Futurism-like image, and rectangle-based Neo-Plasticism-like image. The detailed descriptions of the creation processes for these types of images will be described in subsequent chapters.



Figure 2.5 Images created by Song, et al. [9].

2.2 General Review on Information Hiding Techniques

Information hiding is a technique which embeds data imperceptibly into cover images, so that people will not perceive the existence of the hidden data. Many

information hiding techniques have been proposed for various purposes such as covert communication, authentication, or steganography. Moreover, information hiding techniques often utilize the weaknesses of human visual system. A well-known method is least significant bit (LSB) modification which changes the LSBs of the pixels of an image to embed information. For instance, Chan and Cheng [10] presented a data hiding method by simple LSB substitution, and Wu and Tsai [11] proposed an information hiding method according to a human vision model. They hid the secret messages in the smooth areas of an image based on the characteristics of human vision, so that the image can arouse no notice from observers.

On the other hand, the topic of data hiding via images can be classified into three groups, namely, the spatial-domain method, the frequency-domain method, and the combination of them [12]. Generally, a method in the spatial domain is sensitive against attacks like compression, but its implementation is simple. In the frequency domain, a hiding technique overcomes the problem related to robustness found in the spatial domain, but sometimes produces more distortion. For example, Ni, et al. [13] presented a reversible data hiding algorithm for embedding data in the spatial domain by using the zero or the minimum point of the histogram and slightly modifying the pixel values. Xuan, et al. [14] proposed an approach to hiding secret data into one (or more) middle bitplane(s) of the integer wavelet transform coefficients in the middle and high subbands of the frequency domain. No matter what types they belong to, most of these researches are based on pixel-wise or block-wise operations and make use of few image features.

In this study, data hiding methods using individual features of art images will be proposed. Unlike the traditional methods of data hiding via images, we will hide data in the creation process of art images by modifying the average RGB value of each region of an image or by building the structure of a partition tree. More than this, we

will also implement some schemes to enhance the security of hiding by randomizing the direction of tree building or the order of coloring. In the following chapters, the details will be described.

2.3 Previous Studies on Information Hiding Techniques via Art Images

The combination of information hiding techniques and art image creation is a new idea of information security technology. Techniques based on this idea utilize the characteristics of the creation process of the art image to embed extra information in the generated images. Due to this way of camouflage, secret data can so be kept or transmitted covertly and securely. In addition, hackers will also tend to get unaware of the secret embedded in such images and this reduces the danger of being stolen or being tampered with.

Specifically, Lin and Tsai [15] proposed algorithms to embed secret messages in image mosaics by adding visible boundary regions to the four sides of tiles and modifying the histogram of tile images. Wang and Tsai [16] presented a data hiding technique for image mosaics as well. By varying the overlapping degrees of adjacent tile images, the method can create a new-style mosaic image in which bits of the message data are embedded. Some resulting images created via these two methods are shown in Figure 2.6. Different from the intuitive idea of image mosaics, Lai and Tsai [17] created a new type of mosaic image, called *secret-fragment-visible mosaic*, which is reconstituted with rectangle fragments yielded by partitioning of the original image. A method to embed secret messages is proposed by switching the relative positions of tile images which have similar colors in an identical bin of the histogram. The resulting mosaic image is still a meaningful image like another one, as shown in

Figure 2.7.

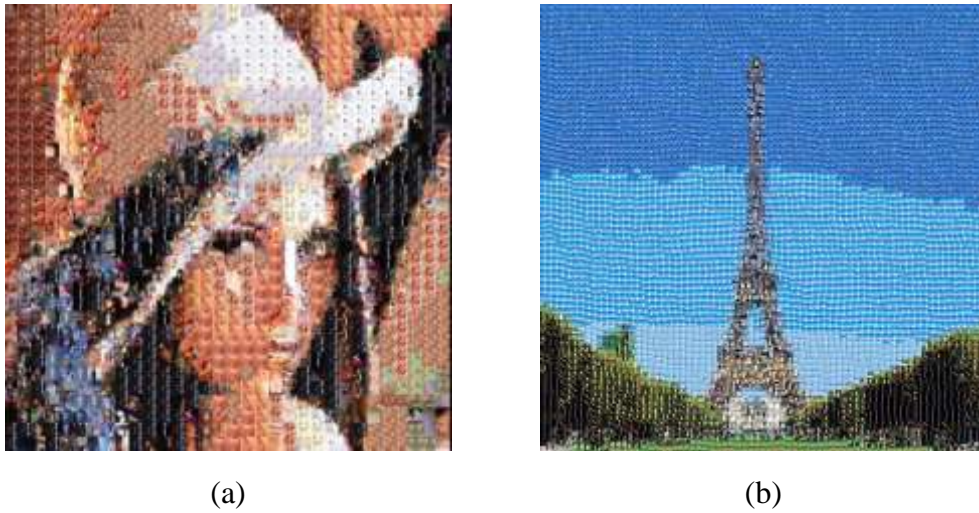


Figure 2.6 Image mosaics. (a) An image mosaic created from Lin and Tsai [15]. (b) An image mosaic created from Wang and Tsai [16].



Figure 2.7 A Secret-fragment-visible mosaic image created with Lai and Tsai' method [17]. (a) Original image. (b) Generated secret-fragment-visible mosaic image.

In addition to previous methods, numerous researches on combining other types of art images and data hiding have been given. Hsu and Tsai [18] presented three new types of art images and three methods to hide secret information in art images by using the features of the creation process. The first type of image, digital puzzle image, is generated to embed data by modifying the orientations, sizes, and angles of the

puzzle pieces. Second, in the new type of pointillistic image, palette colors are used for data hiding by varying the RGB values of each color dot of the pointillistic image. And the last, a new art image called circular-dotted image is created to embed secret messages by using the drawing order of the circular dots and a circular dot overlapping scheme. Some examples of the art images created by Hsu and Tsai [18] are shown in Figure 2.8.

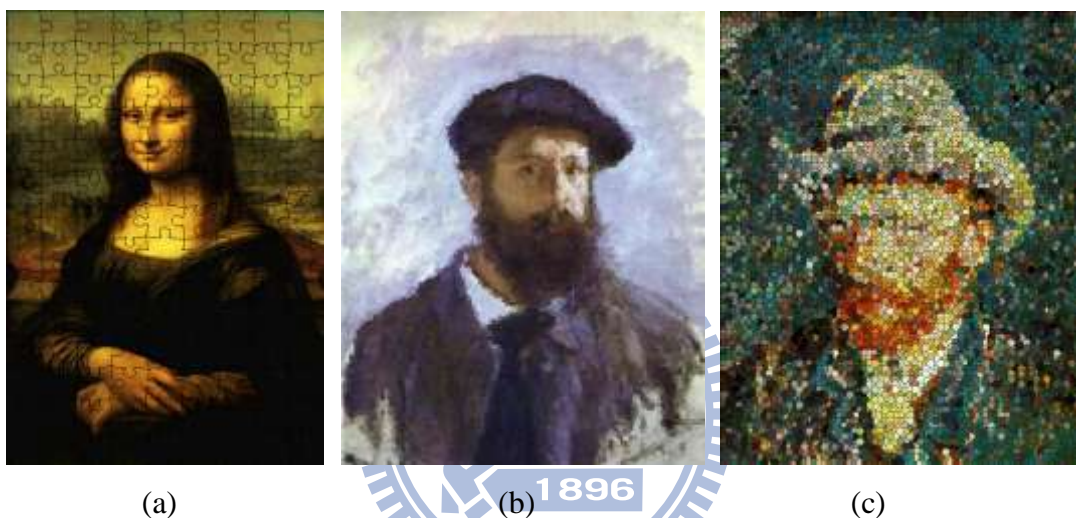


Figure 2.8 Art images created by Hsu and Tsai [18]. (a) A digital puzzle image. (b) A digital pointillistic image. (c) A digital circular-dotted image.

Additionally, an information hiding approach was proposed through the use of stained glass images by Hung and Tsai [19]. The secret data can be hidden in stained glass images by modifying the tree structure used in the creation process. A result generated by the method is shown in Figure 2.9(a). Chang and Tsai [20] created a new type of art image, called tetromino-based mosaic, which is composed of tetrominoes of the Tetris game. A tetromino is a geometric shape composed of four squares which is connected orthogonally. By the composition of geometric forms, tetrominoes can be combined to fit into a fixed shape (rectangles mostly) to form blocks which then are used to fill an image plane. A data hiding method is proposed by using distinct

combinations and color shifting of the tetromino elements. An image yielded by Chang and Tsai [20] is shown in Figure 2.9(b).



(a)



(b)

Figure 2.9 Two examples of art images. (a) A stained glass image from Hung and Tsai [19]. (b) A tetromino-based mosaic from Chang and Tsai [20].

In this study, we also propose new methods which combine information hiding techniques and art image creation to achieve covert communication. By utilizing the characteristics of the creation processes of three art images, which are line-based Cubism-like image, strip-based Futurism-like image, and rectangle-based Neo-Plasticism-like image, the images can be transmitted or kept with the secret data embedded without arousing attention from other people.

Chapter 3

Line-based Cubism-like Image --- A New Type of Image and Its Application to Data Hiding by Invisible Reversible Pixel Re-coloring

3.1 Overview of Proposed Method

In this chapter, we describe how we create a type of art image like Cubism paintings automatically via the use of a computer, and we name this kind of art image *line-based Cubism-like image*. By this type of art image, we try to keep a characteristic of the Cubism art — *multiple viewpoints* — by the use of the *line* feature. By rearranging lines in a given image, which are yielded by applying the Hough transform to the image, a line-based Cubism-like image is created, which includes a new three-dimensional shape of each identity in the given image. In Section 3.2, the proposed method for automatic creation of line-based Cubism-like images will be described in detail.

In order to achieve the purpose of hiding information in this type of art image, we propose also a data hiding technique in this study. A given message is embedded into a line-based Cubism-like image during the stage of region coloring in the creation process of the image. We assign a new color to each image pixel by keeping unchanged the average of the color in the region which includes the pixel, and

re-coloring the pixel without causing a perceptible change. Furthermore, a technique is proposed to enhance the security of the hidden data by randomizing the processing order of the regions.

3.2 Proposed Line-based Cubism-like Image Creation Process

3.2.1 Idea of Proposed Creation Technique

Cubism artists transform a natural scene into geometric forms by breaking up, analyzing, and re-assembling objects in the scene. In addition, with the scene objects rearranged to intersect at random angles, each painting of Cubism seems to be composed of intersecting lines and fragmented shapes in an abstract style. The idea of the proposed art image creation method is inspired by this concept of Cubism, as mentioned previously.

In the creation process of a line-based Cubism-like image from a given image, at first we find the longer line segments in the source image by the Hough transform. Then, we connect the line segments and extend them to reach the image boundaries. Finally, we generate the desired art image via the operations of line segment merging and region re-coloring. This process accomplishes the goal of transforming the input image into an abstract form since the lines of the created Cubism-like image tend to constitute the skeleton of the objects in the source image as observed from according to our experimental results. The detailed algorithms of the above-mentioned processes are described in the following sections.

3.2.2 Proposed Art Image Creation Process

In this section, we present an algorithm which implements the idea of proposed Cubism-like image creation. Basically, in the process of line detection, we find edges of the source image by utilizing the Canny edge detection technique [21], and then perform the Hough transform on the edge detection result to obtain longer line segments in the source image. By extending and recombining these longer line segments, a desired Cubism-like image is created. The detailed algorithm is given as follows.

Algorithm 3.1: *line-based Cubism-like image creation.*

Input: a source image S , and two threshold values — the minimum length L_{\min} of a line segment, and the minimum distance D_{\min} between two lines.

Output: a line-based Cubism-like image C .

Steps.

Stage 1 --- creating crossing-image lines.

- Step 1. Perform Canny edge detection to find the edges E_1, E_2, \dots, E_n in source image S , resulting in a new image S' .
- Step 2. Implement the following steps to find out longer line segments in S' .
 - 2.1 Find the line segments L_1, L_2, \dots, L_m , in S' by applying the Hough transform on S' , yielding a second new image S'' of the line type.
 - 2.2 Select those line segments in S'' with their lengths larger than the threshold L_{\min} .
 - 2.3 Compare every line pair L_i and L_j with $i \neq j$ in S'' in the following way:
if the distance D_{ij} between L_i and L_j is smaller than D_{\min} , then delete L_i
if the length of L_i is smaller than that of L_j ; or delete L_j , otherwise.

Step 3. Extend each of the remaining line segments in S'' to the boundaries of S'' , and regard the source image S as being partitioned by the extended lines into regions.

Stage 2 --- re-coloring image regions.

Step 4. Create a binary image T with the same size as that of S with the initial pixel values all set to be 0.

Step 5. Fill the value of 1 into those pixels in T which correspond to those lying on each of the extended line segments in S'' .

Step 6. Implement following steps to recolor the regions in S .

6.1 Perform region growing in the binary image T in a raster-scan order, and segment out 0-valued regions, R_1, R_2, \dots, R_k , each of which is enclosed by a group of 1-valued line segments in S'' .

6.2 Compute the area A_i of each segmented region R_i in T and the average RGB color values (C_{ir}, C_{ig}, C_{ib}) of the corresponding region R_i' in S using A_i , and re-color each pixel in R_i' of S by the color values $(C_{ir}, C_{ig}, C_{ib}), i = 1, 2, \dots, k$.

6.3 Re-color all lines in S corresponding to the 1-valued extended lines in T by the white color.

Step 7. Take the final S as the desired line-based Cubism-like image C .

The above algorithm of line-based Cubism-like image creation, as illustrated in Figure 3.1, is composed of two stages. In Stage 1, we perform line detection to obtain the longer lines in a source image S . By Canny edge detection, we get a group of edge points. For the purpose of finding prominent line features in S , we use two thresholds to select the longer and sufficiently-separate lines from those line segments yielded by applying the Hough transform to the group of edge points. The first threshold L_{min} is

used to filter out short line segments. The other threshold D_{min} is used to filter out extended lines which are too close to other longer lines. The final step in this stage is to extend each of the remaining line segments to cross the image, with the two line ends reaching the image boundaries.

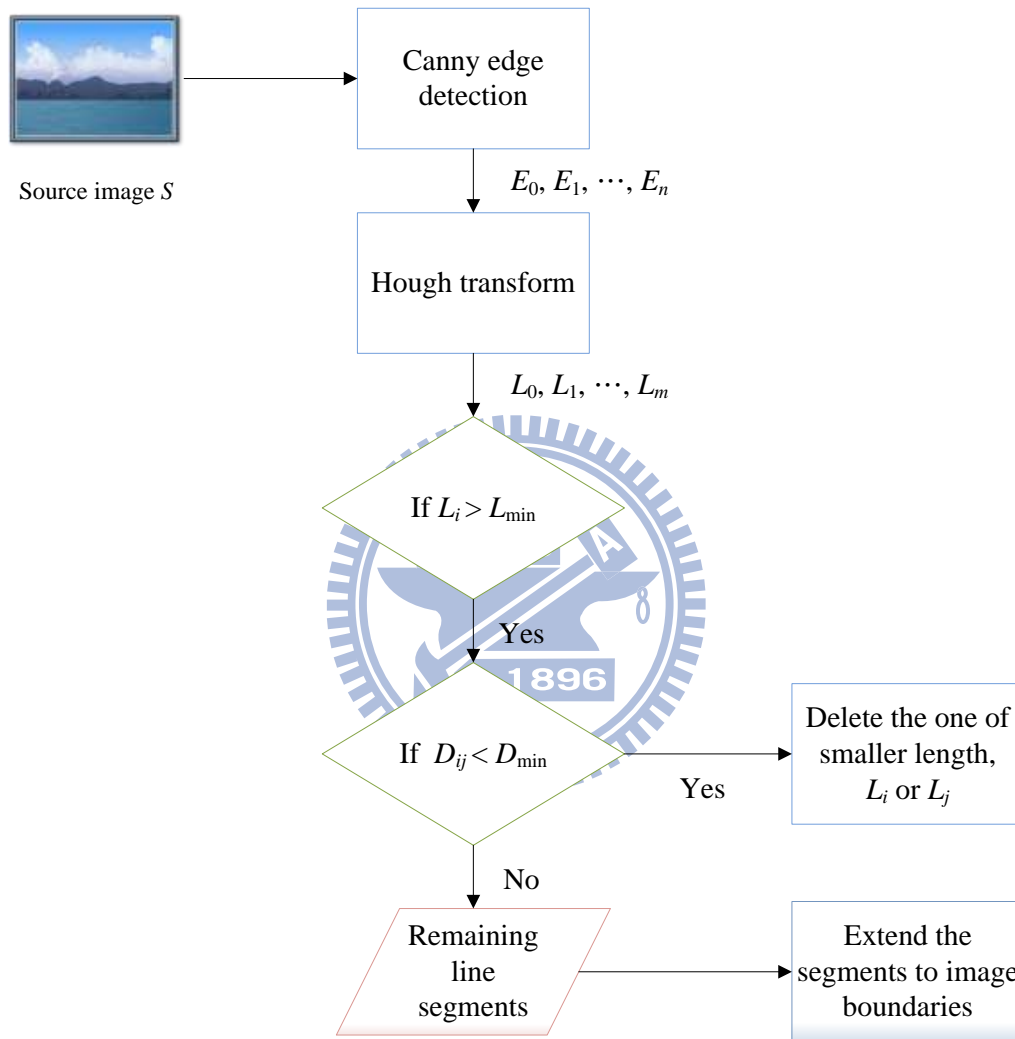


Figure 3.1 Process of crossing-image line creation.

In this study, after considering the mutual influence between the image size and the line length, we use one-tenth of the longer boundary length of the image as the initial value of L_{min} and D_{min} . A series of experiments about the effects of varying the values of L_{min} and D_{min} have been conducted, and an experimental result is shown in Figure 3.2. In these resulting images, we can find that if we take a smaller initial value

of L_{\min} , the number of extracted lines will increase. With more lines, the complexity of the created Cubism-like image also increases, giving an impression closer to the original image content. On the other hand, fewer lines make the Cubism-like image simpler and more abstract. The effect of changing the initial value of D_{\min} is similar to that of L_{\min} .

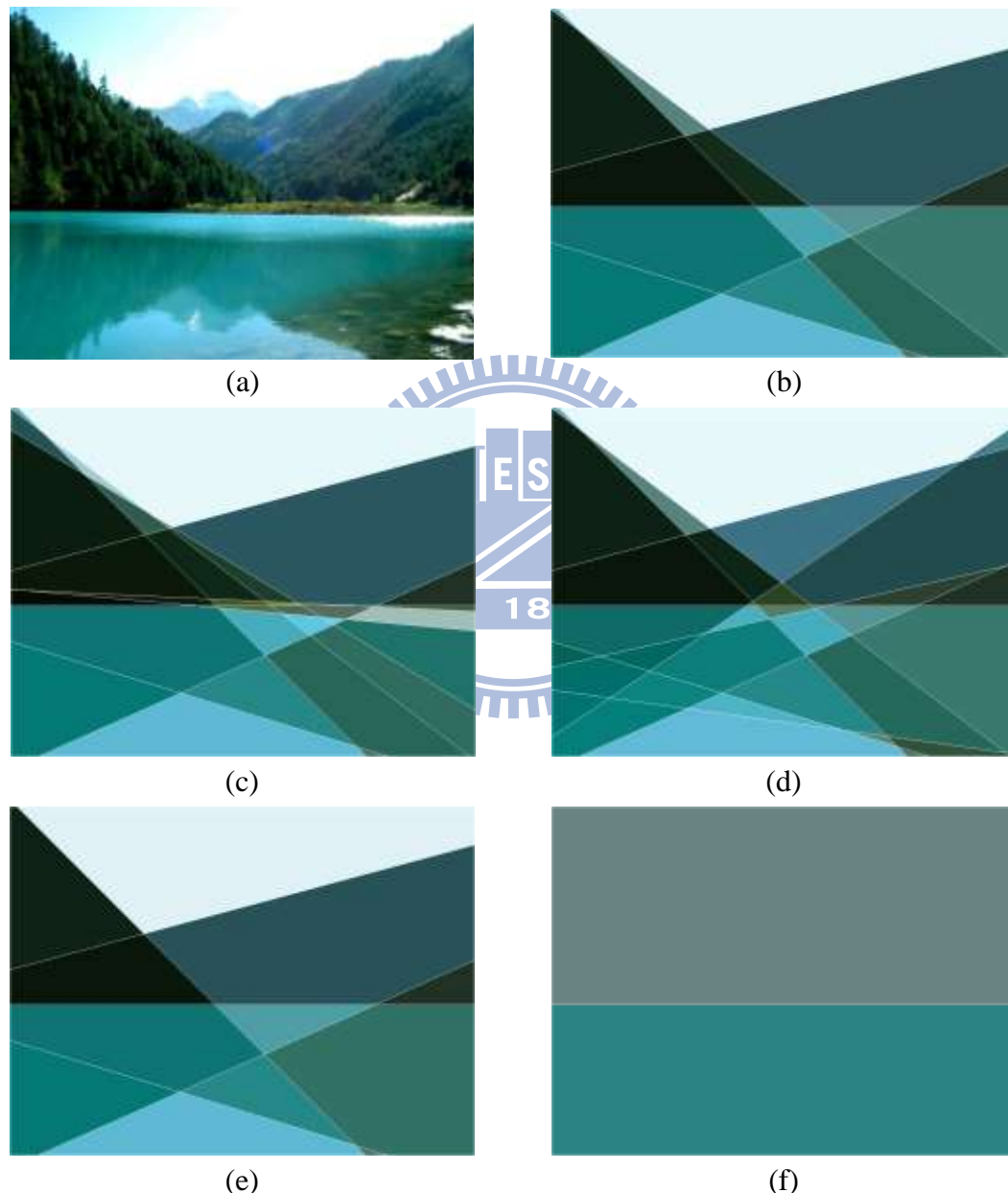


Figure 3.2 An experimental result of varying the threshold values of D_{\min} and L_{\min} . (a) A source image with size 1024×768 . (b) A Cubism-like image created from (a) with initial $D_{\min} = 102$ and initial $L_{\min} = 102$. (c) A Cubism-like image created from (a) with $D_{\min} = 20$ and $L_{\min} = 102$. (d) A Cubism-like image created from (a) with $D_{\min} = 102$ and $L_{\min} = 20$. (e) A Cubism-like image created from (a) with $D_{\min} = 200$ and $L_{\min} = 102$. (f) A Cubism-like image created from (a) with $D_{\min} = 102$ and $L_{\min} = 200$.

In Stage 2, with the extended line segments, the source image S is regarded as being partitioned into regions. By region growing, we segment out these regions and calculate the area and the average RGB color of each of them. Finally, a line-based Cubism-like image C is created by re-coloring these regions with the average color and all the lines with the white color.

3.2.3 Experimental Results

According to the above discussions, we see that different selections of the two threshold values L_{\min} and D_{\min} will result in totally different effects. However, it is difficult to decide which result is better than the others because the decision is obviously dependent on the different feelings of people for art. Therefore, in this study we just offer a series of results yielded by the use of different sets of thresholds for the user to choose. Specifically, we use the *normalized* initial thresholds of 1/10 of the length of the longer image boundary as the center, and vary each threshold to be twice and half of its initial value, in addition to the initial one. As a result, each threshold has three choices, resulting in nine choices of the two threshold values.

Then, we generate nine art images, each corresponding to one of the nine threshold combinations, for the user to choose his/her favorite one among them. Besides, we also provide the option of choosing normalized thresholds for users, and then we produce the nine sets of threshold combinations as described above based on the choice of the user. Some Cubism-like images created by the above-proposed algorithms with nine results yielded by the use of different threshold combinations for each input source image are given in Figures 3.3 through 3.5. For simplification, we use the expression (D_{\min}, L_{\min}) to show a combination of the two thresholds in the captions of the figures. As seen in the images, we can see an abstract style of Cubism

shown by the created lines and regions.

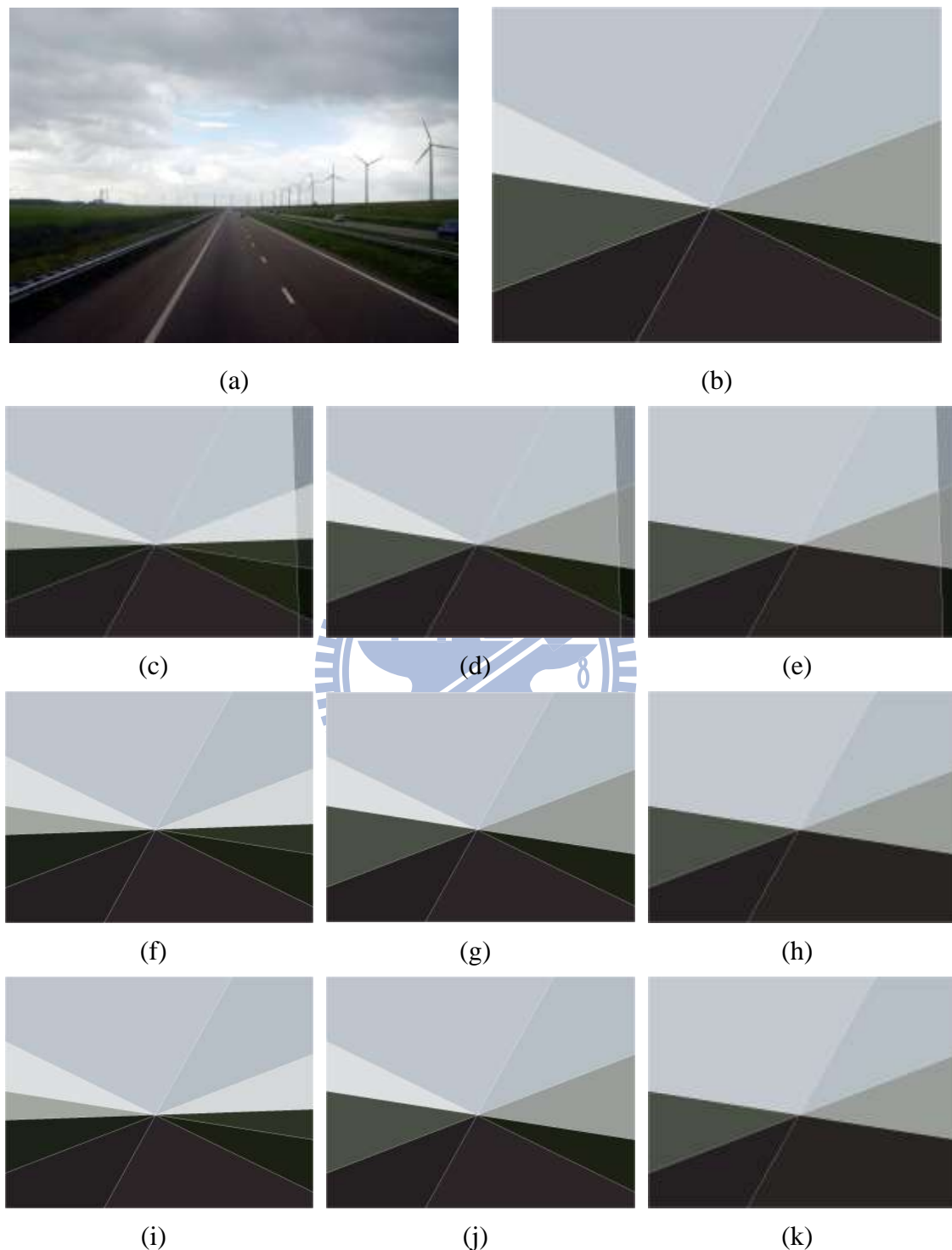
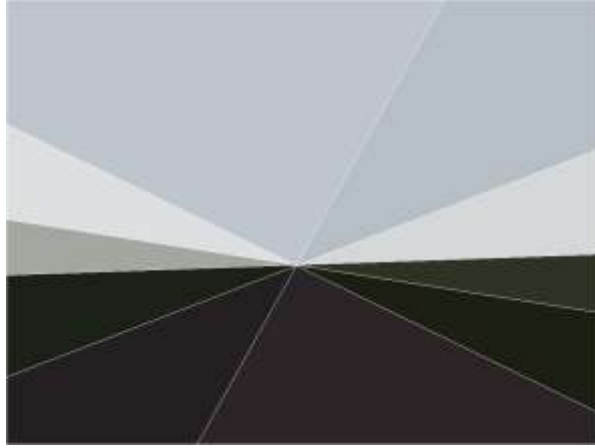
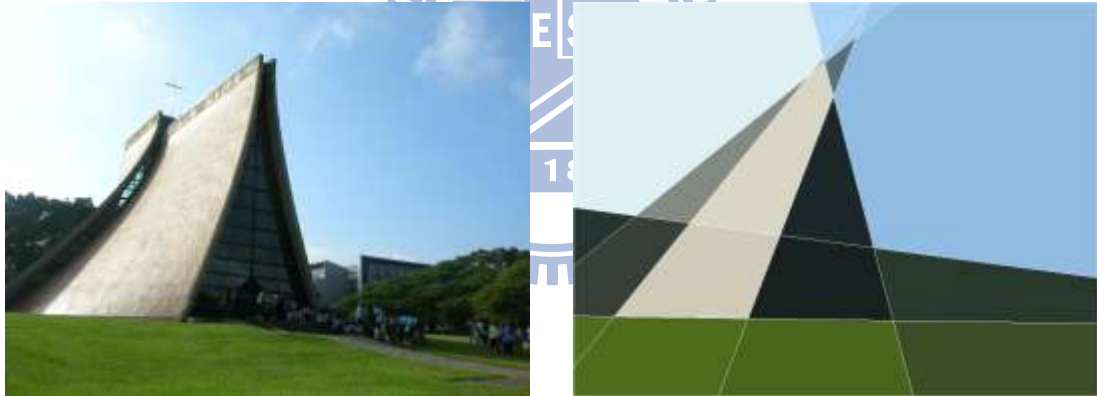


Figure 3.3 Experimental results. (a) A source image with size 1024×768 . (b) Initial $D_{\min} = 102$ and initial $L_{\min} = 102$. (c) $(D_{\min}, L_{\min}) = (51, 51)$. (d) $(D_{\min}, L_{\min}) = (51, 102)$. (e) $(D_{\min}, L_{\min}) = (51, 204)$. (f) $(D_{\min}, L_{\min}) = (102, 51)$. (g) $(D_{\min}, L_{\min}) = (102, 102)$. (h) $(D_{\min}, L_{\min}) = (102, 204)$. (i) $(D_{\min}, L_{\min}) = (204, 51)$. (j) $(D_{\min}, L_{\min}) = (204, 102)$. (k) $(D_{\min}, L_{\min}) = (204, 204)$. (l) A better choice of 9 images to fit the abstract style of Figure 3.3(a) is $D_{\min}=102$ and $L_{\min}=51$.



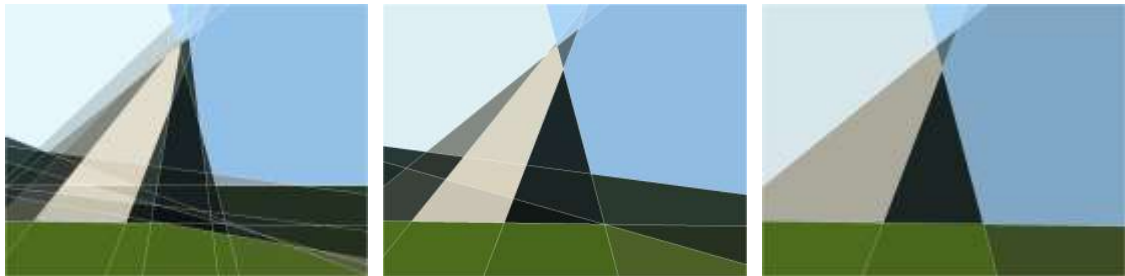
(l)

Figure 3.3 Experimental results. (a) A source image with size 1024×768 . (b) Initial $D_{\min} = 102$ and initial $L_{\min} = 102$. (c) $(D_{\min}, L_{\min}) = (51, 51)$. (d) $(D_{\min}, L_{\min}) = (51, 102)$. (e) $(D_{\min}, L_{\min}) = (51, 204)$. (f) $(D_{\min}, L_{\min}) = (102, 51)$. (g) $(D_{\min}, L_{\min}) = (102, 102)$. (h) $(D_{\min}, L_{\min}) = (102, 204)$. (i) $(D_{\min}, L_{\min}) = (204, 51)$. (j) $(D_{\min}, L_{\min}) = (204, 102)$. (k) $(D_{\min}, L_{\min}) = (204, 204)$. (l) A better choice of 9 images to fit the abstract style of Figure 3.3(a) is $D_{\min}=102$ and $L_{\min}=51$. (Continued.)



(a)

(b)



(c)

(d)

(e)

Figure 3.4 Experimental results. (a) A source image with size 1024×768 . (b) Initial $D_{\min} = 102$ and initial $L_{\min} = 102$. (c) $(D_{\min}, L_{\min}) = (51, 51)$. (d) $(D_{\min}, L_{\min}) = (51, 102)$. (e) $(D_{\min}, L_{\min}) = (51, 204)$. (f) $(D_{\min}, L_{\min}) = (102, 51)$. (g) $(D_{\min}, L_{\min}) = (102, 102)$. (h) $(D_{\min}, L_{\min}) = (102, 204)$. (i) $(D_{\min}, L_{\min}) = (204, 51)$. (j) $(D_{\min}, L_{\min}) = (204, 102)$. (k) $(D_{\min}, L_{\min}) = (204, 204)$. (l) A better choice of 9 images to fit the abstract style of Figure 3.4(a) is $D_{\min}=102$ and $L_{\min}=51$.

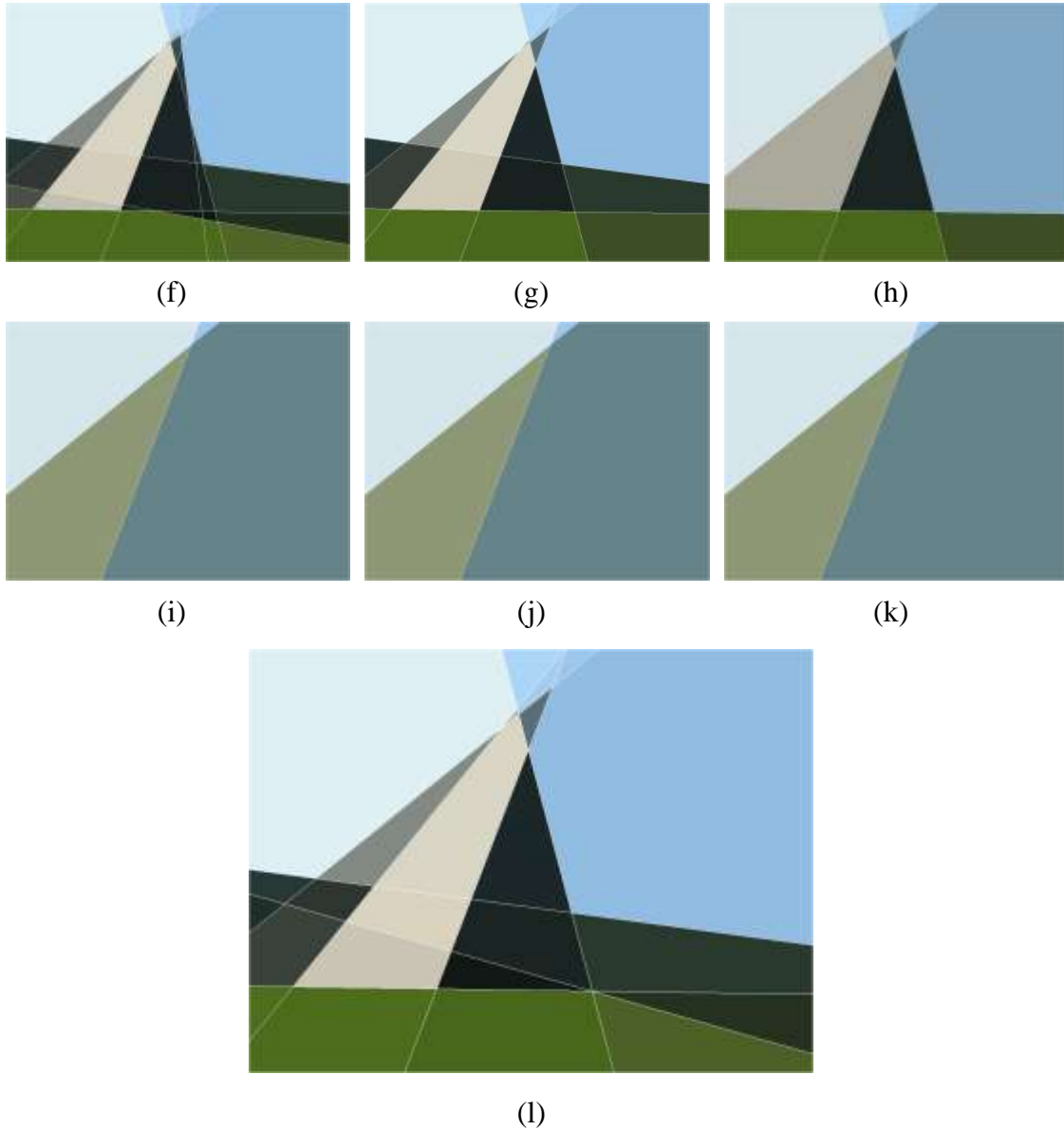
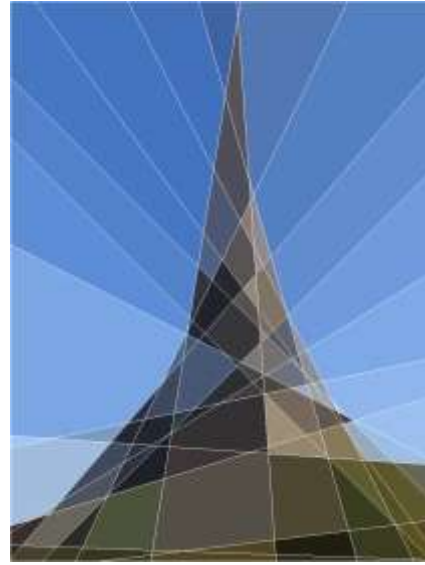


Figure 3.4 Experimental results. (a) A source image with size 1024×768 . (b) Initial $D_{\min} = 102$ and initial $L_{\min} = 102$. (c) $(D_{\min}, L_{\min}) = (51, 51)$. (d) $(D_{\min}, L_{\min}) = (51, 102)$. (e) $(D_{\min}, L_{\min}) = (51, 204)$. (f) $(D_{\min}, L_{\min}) = (102, 51)$. (g) $(D_{\min}, L_{\min}) = (102, 102)$. (h) $(D_{\min}, L_{\min}) = (102, 204)$. (i) $(D_{\min}, L_{\min}) = (204, 51)$. (j) $(D_{\min}, L_{\min}) = (204, 102)$. (k) $(D_{\min}, L_{\min}) = (204, 204)$. (l) A better choice of 9 images to fit the abstract style of Figure 3.4(a) is $D_{\min}=102$ and $L_{\min}=51$. (Continued.)

In Figures 3.3 through 3.5, Figure 3.3(a), 3.4(a), and 3.5(a) are the source images, and Figures 3.3(b), 3.4(b), and 3.5(b) show the results generated with initial thresholds. Figures 3.3(c) through (k), 3.4(c) through (k), and 3.5(c) through (k) are the experimental results with nine combinations of the thresholds. Finally, Figures 3.3(l), 3.4(l), and 3.5(l) are better choices from the respective nine results.



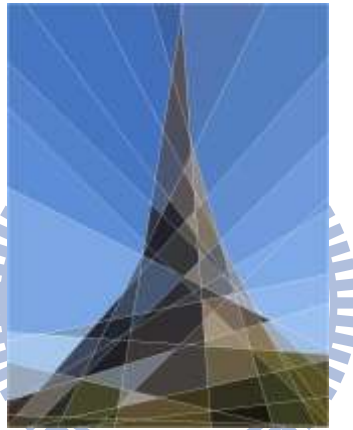
(a)



(b)



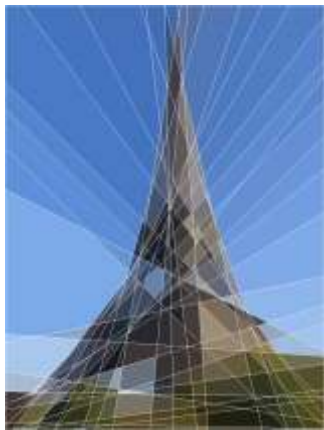
(c)



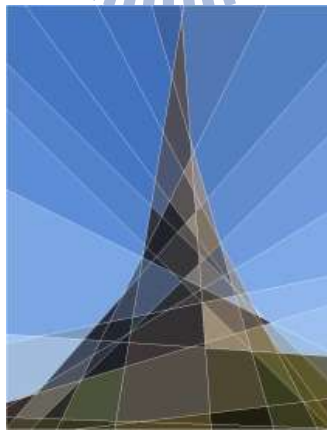
(d)



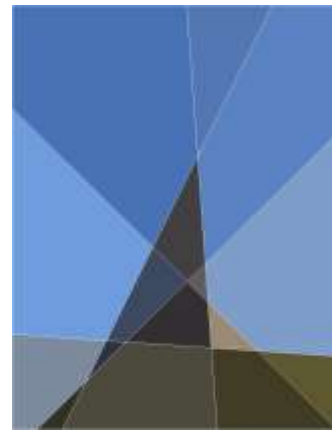
(e)



(f)



(g)



(h)

Figure 3.5 Experimental results. (a) A source image with size 768×1024 . (b) Initial $D_{\min} = 102$ and initial $L_{\min} = 102$. (c) $(D_{\min}, L_{\min}) = (51, 51)$. (d) $(D_{\min}, L_{\min}) = (51, 102)$. (e) $(D_{\min}, L_{\min}) = (51, 204)$. (f) $(D_{\min}, L_{\min}) = (102, 51)$. (g) $(D_{\min}, L_{\min}) = (102, 102)$. (h) $(D_{\min}, L_{\min}) = (102, 204)$. (i) $(D_{\min}, L_{\min}) = (204, 51)$. (j) $(D_{\min}, L_{\min}) = (204, 102)$. (k) $(D_{\min}, L_{\min}) = (204, 204)$. (l) A better choice of 9 images to fit the abstract style of Figure 3.5(a) is $D_{\min}=102$ and $L_{\min}=102$.

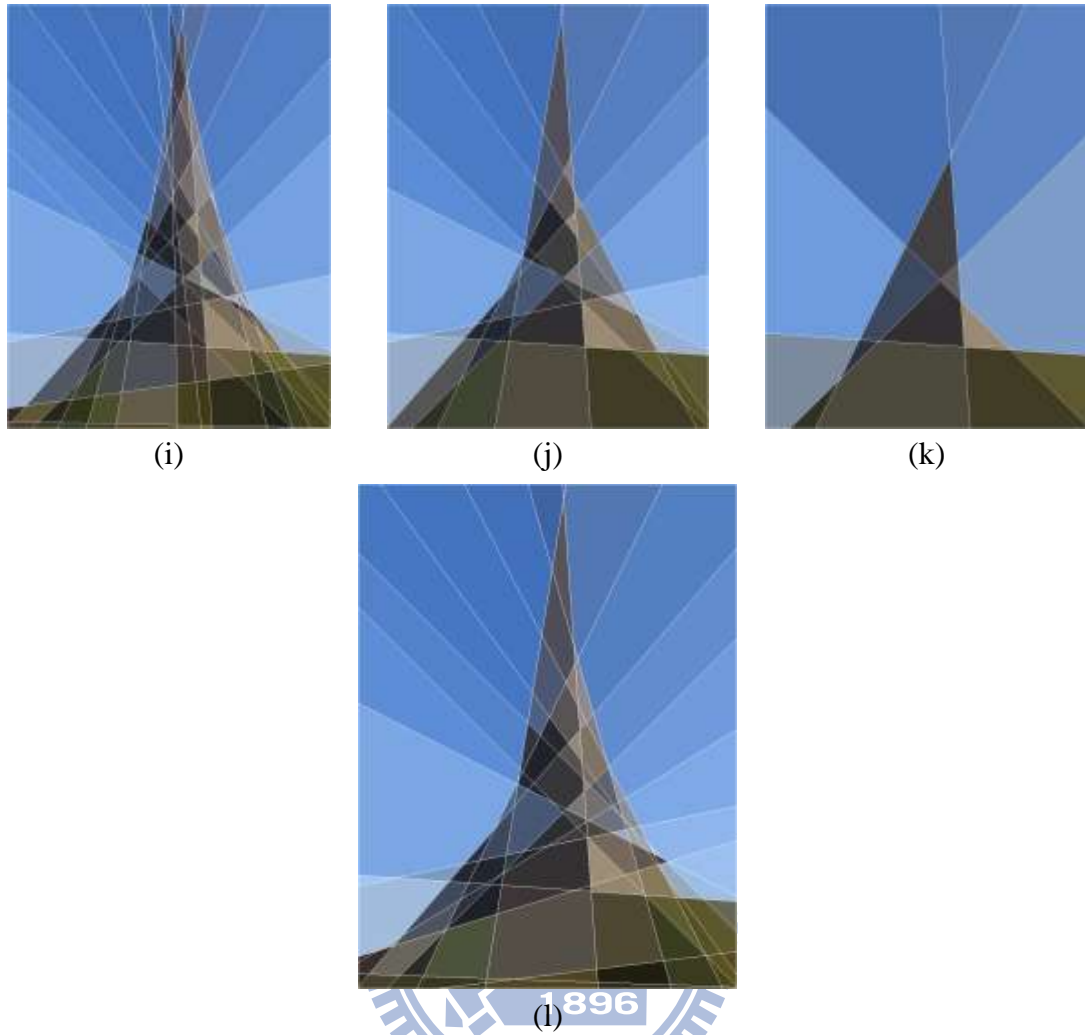


Figure 3.5 Experimental results. (a) A source image with size 768×1024 . (b) Initial $D_{\min} = 102$ and initial $L_{\min} = 102$. (c) $(D_{\min}, L_{\min}) = (51, 51)$. (d) $(D_{\min}, L_{\min}) = (51, 102)$. (e) $(D_{\min}, L_{\min}) = (51, 204)$. (f) $(D_{\min}, L_{\min}) = (102, 51)$. (g) $(D_{\min}, L_{\min}) = (102, 102)$. (h) $(D_{\min}, L_{\min}) = (102, 204)$. (i) $(D_{\min}, L_{\min}) = (204, 51)$. (j) $(D_{\min}, L_{\min}) = (204, 102)$. (k) $(D_{\min}, L_{\min}) = (204, 204)$. (l) A better choice of 9 images to fit the abstract style of Figure 3.5(a) is $D_{\min}=102$ and $L_{\min}=102$. (Continued.)

3.3 Proposed Technique for Data Hiding in Line-based Cubism-like Images by Invisible Reversible Pixel Re-coloring

3.3.1 Idea of Proposed Data Hiding Technique

The proposed method of using Cubism-like images for covert communication is

described in this section. In the proposed Cubism-like image creation process as presented by Algorithm 3.1 above, we re-color image regions with the respective average colors of the regions. Due to the nature of the human visual system, people cannot sense small changes in the appearance of a color image, such as color alternations or edge shiftings. Accordingly, we implement a method to hide secret a message in a cover image (a cubism-like color stego-image generated by the proposed method described previously) by slightly changing the RGB color values of the pixels in each region of the cover image. As a result, people will not be able to distinguish between the cover image and a stego-one. It is in this way that we achieve the goal of data hiding in the proposed line-based Cubism-like art image.

Besides, for the reason of reversibility in the hidden data extraction process, a region re-coloring technique is proposed, which keeps the average color of each region unchanged. Consequently, we can restore the color information of the pixels of the stego-image, and extract the secret messages embedded in them. In the following sections, these mentioned techniques used in the proposed method for data hiding in the line-based Cubism-like image will be described in detail.

More specifically, in the proposed data hiding process, after the step of hiding the message bits into a color channel, the pixel colors in a region will be changed via color shifting, and the average color of the region will also be influenced. In order to keep the average color unchanged, we must limit the number of embedded message bits. For this purpose, it is found in the study that the property of *rounding-off* may be utilized. Specifically, when computing the average color C of a region, all the computed results in the range between $C - 0.5$ and $C + 0.5$ will be rounded to be an identical value since RGB color values used in this study are *integer numbers*. Accordingly, we can acquire the maximum number of embedded message bits in a region R by an equation derived as follows.

Assume that A is the area of region R (in unit of pixel), $C_{r1}, C_{r2}, \dots, C_{rm}$ are the R-color values of the n pixels in region R , and C_r is the average R-color value of region R . Also, let N_0 and N_1 denote the total numbers of bits of 0's and 1's embedded in R . At first, the average R-color value C_r of all the pixels in R may be computed by

$$C_r = (C_{r1}, C_{r2}, \dots, C_{rm})/A.$$

Therefore, we have

$$C_r A = C_{r1} + C_{r2} + \dots + C_{rm}.$$

Furthermore, we hide the secret message bits by shifting the average R-color value of each pixel in this study. We assume that when hiding a bit of 0 into a pixel P in region R with average R-color value C_r , the pixel's color C_r is decreased by 1; and when hiding a bit of 1 into P , the pixel's color C_r is increased by 1. Therefore, if N_0 0's and N_1 1's are embedded into the pixels of region R , then it can be figured out that the average color value C_r of R will increase for the amount of $N_1 - N_0$ (or equivalently, decrease for the amount of $N_0 - N_1$), so that the new average color C_r' becomes:

$$\begin{aligned} C_r' &= \frac{C_{r1} + C_{r2} + \dots + C_{rm} + N_1 - N_0}{A} \\ &= \frac{C_r A + N_1 - N_0}{A}. \end{aligned}$$

To keep the new average color C_r' equal to the original one C_r , we have to limit the values of the two numbers N_1 and N_0 according to the above-mentioned rounding-off property, resulting in the following formula:

$$C_r - \frac{1}{2} < C_r' = \frac{C_r A + N_1 - N_0}{A} < C_r + \frac{1}{2}$$

which can be reduced to be

$$-\frac{A}{2} < N_1 - N_0 < \frac{A}{2},$$

or equivalently, to be

$$|N_1 - N_0| < \frac{A}{2}. \quad (3.1)$$

As a summary, the values of N_1 and N_0 together are limited by the total number A of pixels in the region, implying that the data hiding capacity is also restricted by it. In the best case, we know that the maximum number of embeddable bits in a region is just the number A of pixels in the region, that is,

$$N_1 + N_0 = A \quad (3.2)$$

Now, according to Equations (3.1) and (3.2), we can derive the range of N_1 and N_0 to be as follows (the details omitted):

$$\frac{A}{4} < N_0 < \frac{3A}{4} \quad (3.3)$$

$$\frac{A}{4} < N_1 < \frac{3A}{4} \quad (3.4)$$

That is, if N_1 and N_0 satisfy the ranges specified by Equation (3.3) and (3.4), we can get the maximum number of embeddable bits in a region. In the extreme case where the digit sequence is composed of all 0's or all 1's, the upper bound of the data hiding capacity will be reached, which is $\left\lfloor \frac{A}{2} \right\rfloor - 1$, as can be figured out from Equation (3.1).

In this study, we keep the average region color unchanged for two reasons. The first, as mentioned previously, is to make possible recovery of the region information in the data extraction process, where we use the average color as a basis to extract the hidden secret message. The detail can be seen in Section 3.3.3. The other reason is to yield a visual mosaic effect such that the region color that people see is *almost the same*, despite of the color shifting inside the region.

3.3.2 Proposed Data Hiding Process

Based on the creation process of the line-based Cubism-like image, we propose a data hiding method in this section. According to the last step of Algorithm 3.1 described previously, we re-color each region of the input image by the average color of the region to generate a Cubism-like image. For the purpose of hiding data in a generated Cubism-like image, we try to modify this re-coloring process to hide secret messages. Specifically, the data hiding process is composed of two main stages. First at all, we transform the secret message into a digit sequence and append an ending pattern (with at least one digit) at the end of the digit sequence to keep the sequence length a multiple of three, as shown in Figure 3.6. By the ending pattern, we can determine where the message ends in a sequence of extracted bits in the message extraction process. Then, we obtain the information of two parameters of each region, namely, the region area and the average RGB color values in the region, by performing Algorithm 3.1.

Furthermore, we use a secret key to randomize the order of re-coloring of the regions in the input image, and take the resulting new sequence as the order for data hiding. For each region, we compute the maximum data hiding capacity first. In order to keep the average RGB color of the region unchanged, we limit the embedded amount of message bit data in each region. After getting the maximum data hiding capacity, we embed the message bits by shifting the RGB color values of the pixels in each region according to the above-mentioned data hiding order. After the digit sequence is exhausted, there might exist regions in which no message bit is embedded. We have to deal further with these intact regions to keep the coloring style of all regions consistent. By creating a new binary string whose size is the upper bound of the data hiding capacity and constituting randomly a string with 0's and 1's by the

secret key, we use the same method to re-color the pixels according to the resulting binary string. At the end, a stego-image is generated with the secret message embedded. The detailed algorithm is given as follows.

Algorithm 3.2: embedding a secret message into a line-based Cubism-like image.

Input: a cover image S , a secret key K , a secret message M , and two threshold values — the minimum length L_{\min} of a line segment, and the minimum distance D_{\min} between two lines.

Output: a stego-image I into which M is embedded.

Steps.

Stage 1 --- embedding a secret message M .

- Step 1. Transform the secret message M , eight bits for each character, into a bit sequence M' , and randomize the bits in M' by the secret key K .
- Step 2. Transform each bit of M' into a digit, resulting in a digit sequence M'' , and append an ending pattern with at least one and no more than three identical digits other than 0's and 1 (such as 2, 22, or 222) at the end of M'' to form a new digit sequence with its length being a multiple of three, as shown in Figure 3.6.

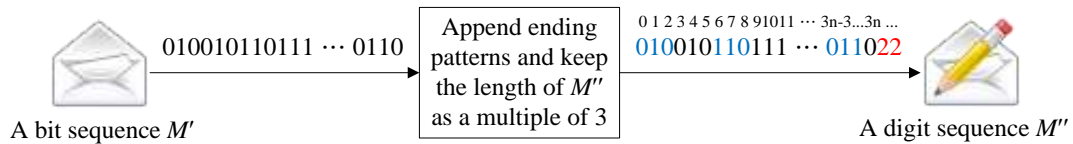


Figure 3.6 The process of transforming M' into M'' with a length of a multiple of three by appending an ending pattern.

- Step 3. Divide M'' into a series of 3-digit segments m_1, m_2, \dots, m_n .
- Step 4. Perform Algorithm 3.1, using the input cover image S as the source image,

to obtain the information of two parameters of the regions R_1, R_2, \dots, R_k in S , namely, the areas A_1, A_2, \dots, A_k and the average RGB color values $(C_{1r}, C_{1g}, C_{1b}), (C_{2r}, C_{2g}, C_{2b}), \dots, (C_{kr}, C_{kg}, C_{kb})$ of the regions R_1, R_2, \dots, R_k , respectively.

Step 5. Rearrange randomly the coloring order of the regions by the secret key K , resulting in a new coloring sequence $C_s = \{R_1', R_2', \dots, R_k'\}$; and change accordingly the corresponding orders of the areas and the average RGB color values, resulting in the new orders of A_1', A_2', \dots, A_k' and $(C_{1r}', C_{1g}', C_{1b}'), (C_{2r}', C_{2g}', C_{2b}'), \dots, (C_{kr}', C_{kg}', C_{kb}')$, respectively.

Step 6. Calculate the maximum data hiding capacity Q_i of each region R_i' in accordance with the new coloring sequence C_s by the following steps with the initial value of Q_i being set to be zero.

6.1 Assign each 3-digit segment m_i of the digit sequence M'' into one of six groups $N_{r0}, N_{r1}, N_{g0}, N_{g1}, N_{b0},$ and N_{b1} in the following way, assuming that the three digits in m_i are denoted as $b_r, b_g,$ and b_b :

- (a) increase N_{r0} by 1 if $b_r = 0$; and increase N_{r1} by 1 if $b_r = 1$;
- (b) increase N_{g0} by 1 if $b_g = 0$; and increase N_{g1} by 1 if $b_g = 1$;
- (c) increase N_{b0} by 1 if $b_b = 0$; and increase N_{b1} by 1 if $b_b = 1$.

6.2 Compute the maximum data hiding capacity Q_i of each region R_i' by the following operations.

- (a) If $|N_{r1} - N_{r0}| \geq \frac{A_i'}{2}$, take the current Q_i to be the maximum data hiding capacity.
- (b) If $|N_{g1} - N_{g0}| \geq \frac{A_i'}{2}$, take the current Q_i to be the maximum data hiding capacity .

(c) If $|N_{b1} - N_{b0}| \geq \frac{A_i'}{2}$, take the current Q_i to be the maximum data hiding capacity.

(d) Increase the data hiding capacity Q_i by 3 if Q_i has not been set to be the maximum data hiding capacity.

6.3 Repeat Steps 6.1 and 6.2 until Q_i has been taken to be the maximum data hiding capacity or until the digit sequence M'' is exhausted.

Step 7. Perform the following steps to embed the secret message bits into each region R_i' .

7.1 Reorder randomly the pixels in R_i' , which is initially in a raster-scan order, into a *hiding sequence* $H_s = \{p_1, p_2, \dots, p_t\}$, by using the secret key K and the index i of R_i' in the coloring sequence C_s as the seed for the randomization process.

7.2 Embed each 3-digit segment m_t of the digit sequence M'' into a pixel p_x of R_i' according to the hiding sequence H_s in the following way.

(a) Obtain new average RGB color values $(C_{ir}'', C_{ig}'', C_{ib}'')$ of each pixel p_x in S by shifting in order the original average RGB color values $(C_{ir}', C_{ig}', C_{ib}')$ of p_x in the following way:

- i. if the value of m_t is 0, decrease by 1 the first unchanged color value, say C_{ih}' , in the original average color values $(C_{ir}', C_{ig}', C_{ib}')$;
- ii. if the value of m_t is 1, increase C_{ih}' by 1;
- iii. if the value of m_t is 2, do nothing to the original average color values $(C_{ir}', C_{ig}', C_{ib}')$.

(b) Re-color the pixel p_x by the new RGB color values $(C_{ir}'', C_{ig}'', C_{ib}'')$.

(c) Decrease the data hiding capacity Q_i by 3.

(d) Repeat the above two steps until the maximum data hiding capacity Q_i is exhausted.

Step 8. Repeat Steps 6 and 7 if the digit sequence of M'' is not exhausted.

Stage 2 --- dealing with intact regions.

Step 9. Perform the following steps to deal with each intact region R_j' with area A_j' which has not been used for message bit embedding so far.

9.1 Use the secret key K to create a binary string B with size $\left\lfloor \frac{A_j'}{2} \right\rfloor - 1$

for R_j' , which is composed of a random sequence of 0's and 1's.

9.2 Perform Steps 6 through 8 to re-color the pixels of R_j' to embed the binary string B .

Step 10. Take the final S as the desired stego-image I .

In above algorithm, wrap-around problems might occur in Steps of 7.2(a)-i and 7.2(a)-ii when the average RGB color values are (255, 255, 255) or (0, 0, 0). In this case, we will obtain a bad stego-image with some noise (black or white image points) after 1 is added to 255 or 1 is subtracted from 0. To avoid such extreme cases, we adjust the extreme average color values of (255, 255, 255) and (0, 0, 0) to be (254, 254, 254) and (1, 1, 1), respectively before data hiding. Such slight color alternations in the generated stego-image cause nearly no visual effect to human vision.

3.3.3 Proposed Secret Message Extraction Process

In the proposed secret message extraction process, first we recover the coloring sequence in the stego-image. By a region growing scheme, we get the information of

the regions with an initial order sequence. Then, we retrieve the coloring sequence by using the secret key. Moreover, in the process of region growing, we also obtain the area and the average RGB color of each region in the stego-image. Based on the average RGB color values, we can retrieve accordingly the secret message embedded in the stego-image. The algorithm of secret data extraction is described in detail as follows.

Algorithm 3.3: extracting a secret message from a stego-image.

Input: a stego-image S , and a secret key K identical to that used in Algorithm 3.2.

Output: the secret message M supposedly embedded in S .

Steps.

Stage 1 --- retrieving the information of the stego-image S .

Step 1. Perform the region growing scheme in a raster-scan order to segment out regions, R_1, R_2, \dots, R_k , in S , and obtain the information about the area A_1, A_2, \dots, A_k and the average RGB color values $(C_{1r}, C_{1g}, C_{1b}), (C_{2r}, C_{2g}, C_{2b}), \dots, (C_{kr}, C_{kg}, C_{kb})$ of the regions, respectively.

Step 2. Retrieve the coloring order of regions by the secret key K , denoted as coloring sequence $C_s = \{R_1', R_2', \dots, R_k'\}$, and change the corresponding orders of the areas and the average RGB color values, resulting in the new order sequences of areas A_1', A_2', \dots, A_k' and average color values $(C_{1r}', C_{1g}', C_{1b}'), (C_{2r}', C_{2g}', C_{2b}'), \dots, (C_{kr}', C_{kg}', C_{kb}')$, respectively.

Stage 2 --- extracting the embedded secret message M .

Step 3. Create an *empty* digit sequence Q initially.

Step 4. Perform the following steps to extract the secret message M from S .

4.1 Extract the pixel order in R_i' which is initially in a raster-scan order, denoted as a recovery sequence $H_s = \{p_1, p_2, \dots, p_i\}$, by using the

secret key K and the index of R_i' in the coloring sequence C_s as the seed of the randomization process.

4.2 Obtain the RGB color values of each pixel p_x of R_i' according to the hiding sequence H_s and denote them by $(C_{ir}'', C_{ig}'', C_{ib}'')$.

4.3 Acquire three values Q_{ir} , Q_{ig} , and Q_{ib} from the difference between the RGB color values $(C_{ir}'', C_{ig}'', C_{ib}'')$ of each pixel p_x and the average RGB color values $(C_{ir}', C_{ig}', C_{ib}')$, respectively, by the following way:

(a) if any of the color values $(C_{ir}'', C_{ig}'', C_{ib}'')$ is smaller than the corresponding average color value in $(C_{ir}', C_{ig}', C_{ib}')$, then set the corresponding secret value Q_{ir} , Q_{ig} , or Q_{ib} to be 0;

(b) if any of the color values $(C_{ir}'', C_{ig}'', C_{ib}'')$ is larger than the average color $(C_{ir}', C_{ig}', C_{ib}')$, then set the secret value Q_{ir} , Q_{ig} , or Q_{ib} to be 1;

(c) if the colors $(C_{ir}'', C_{ig}'', C_{ib}'')$ and the average colors $(C_{ir}', C_{ig}', C_{ib}')$ are the same, respectively, then set the secret value Q_{ir} , Q_{ig} , or Q_{ib} to be 2 (the ending pattern).

4.4 Store the three digits Q_{ir} , Q_{ig} , and Q_{ib} into Q in order.

4.5 Repeat Steps 4.1 through 4.4 until a value in Q equal to 2 (the ending pattern) is encountered.

Step 5. Use the secret key K to reorder Q .

Step 6. Transform every eight values of Q into a decimal number, resulting in a new sequence Q' , and then transform Q' into characters as the desired secret message M .

3.3.4 Security Consideration

As mentioned previously, people can extract an embedded secret message from a

Cubism-like image by the proposed extraction process described by Algorithm 3.3. In the proposed method, we assume that both the sender and the receiver have the secret key beforehand. Without the right secret key, any malicious user cannot extract the secret message successfully. However, some malicious users may observe the regularity of these pixels to guess the secret message by trial and error. We try to prevent this possibility in advance.

To ensure the security of the proposed technique, in addition to using a secret key to encrypt and protect the secret message, we also use it to randomize the processing order of the regions and the hiding order of the pixels. Besides, for the intact regions, we also imitate the hiding process to embed random sequences in them, and malicious users may be misled to guess the secret message erroneously using data in them. With the above-mentioned schemes for security enhancement, we can decrease the risk for the message to be stolen by malicious users.

3.3.5 Experimental Results

Figures 3.7 through 3.12 show some experimental images of applying the proposed data hiding method in a Cubism-like image. Figure 3.7(a) and Figure 3.10(a) are source images, which are also cover images for data hiding. Figure 3.7(b) and Figure 3.10(b) are Cubism-like images without secret message embedding. Figure 3.7(c) is a stego-image into which a secret message “Hi, I am Helen. Nice to meet you!” has been embedded with the secret key “door”. Figure 3.10(c) is a stego-image into which a secret message “Meet me at 21:30. See you.” has been embedded with the secret key “test”.

The secret message will be retrieval only if the right key is used in the secret message extraction process, like Figure 3.8 and 3.11. If someone uses a wrong secret

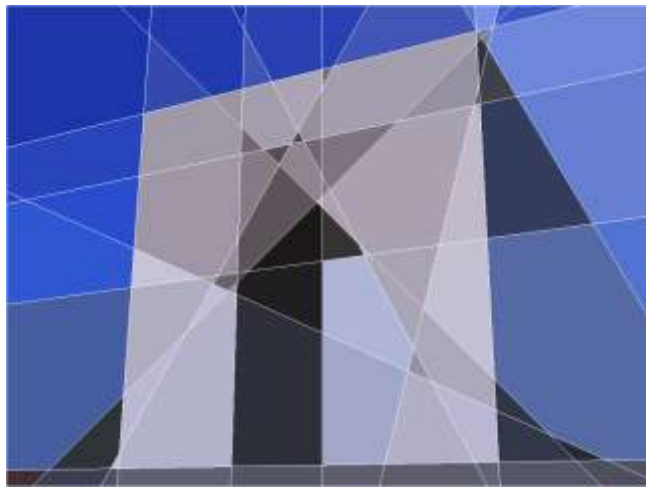
key in a secret extraction process, the extraction will fail as shown in Figure 3.9 and 3.12.

3.4 Summary

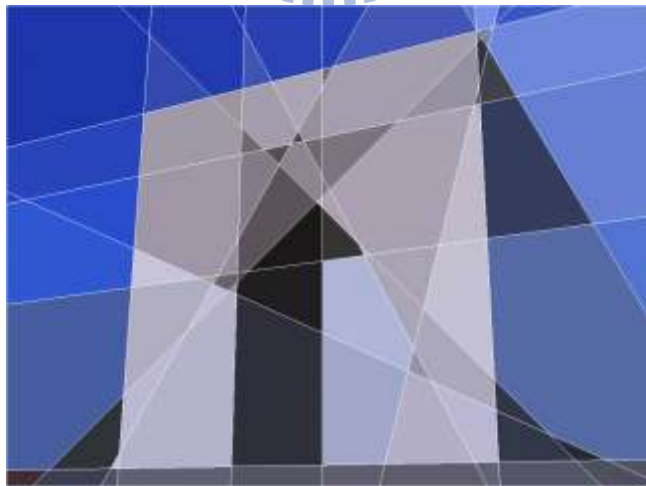
In this study, we propose a new type of computer art, called line-based Cubism-like image. To create it, we propose an automatic method to find the longer line segments in a source image by utilizing the Canny edge detection technique [21] and the Hough transform method [22]. After rearranging the longer line segments according to line length and distance parameters, a line-based Cubism-like image is created, which accomplishes the goal of producing an abstract form of the source image, and includes a new three-dimensional shape of each identity in the given image. By utilizing the characteristic of the Cubism-like image creation process, we propose further a data hiding technique for covert communication in the stage of region re-coloring in image creation process. Based on shifting colors slightly, we embed secret message bits into the pixels of regions by keeping the average of the region color unchanged. Furthermore, considering the security of embedded messages, we use a secret key to randomize the secret message bit sequence and the message bit processing order. Because the resulting color difference between the Cubism-like image and stego-image is small, users cannot notice the existence of the hidden data. Consequently, the proposed information hiding method is feasible for covert communication.



(a)



(b)



(c)

Figure 3.7 An experimental result. (a) A source image (cover image). (b) A Cubism-like image without secret message embedding. (c) A stego-image of (a) by embedding the secret message “Hi, I am Helen. Nice to meet you!” with the secret key “door.”

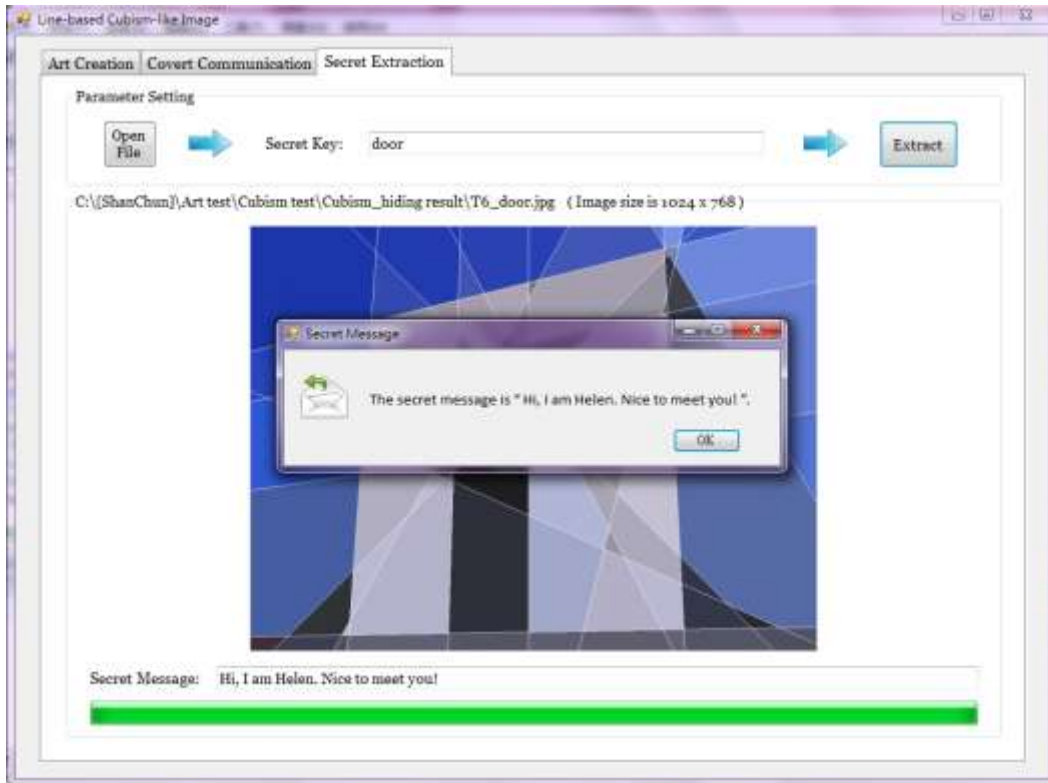


Figure 3.8 Extracting the secret message with the right secret key “door.”

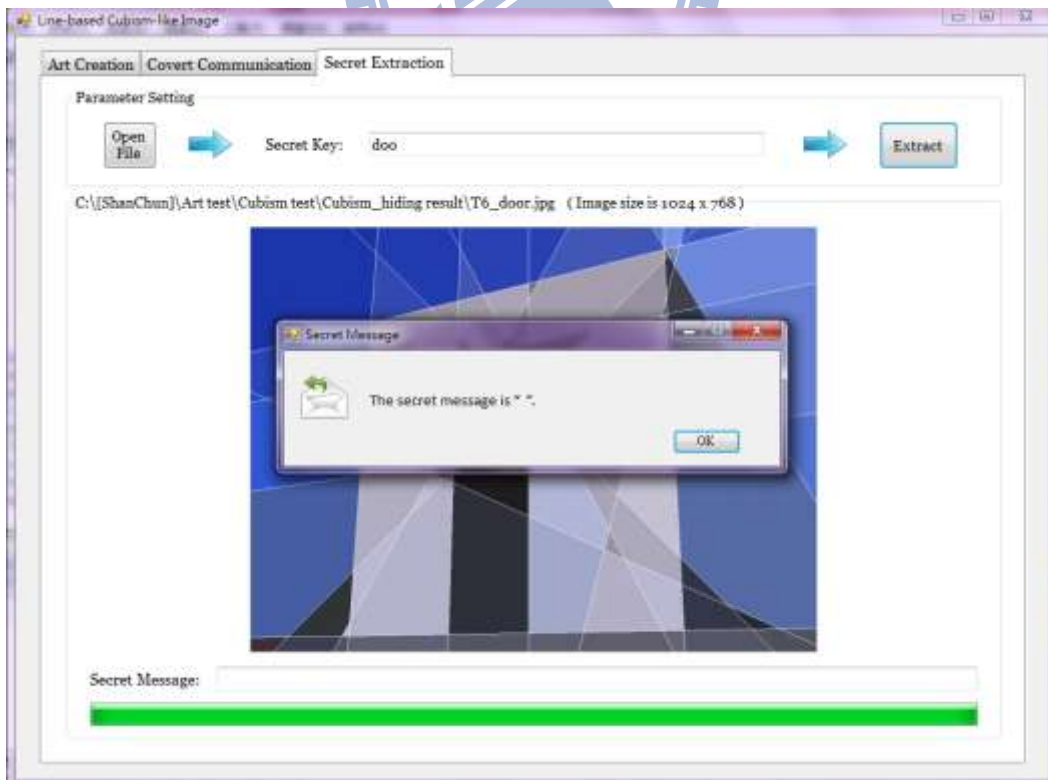
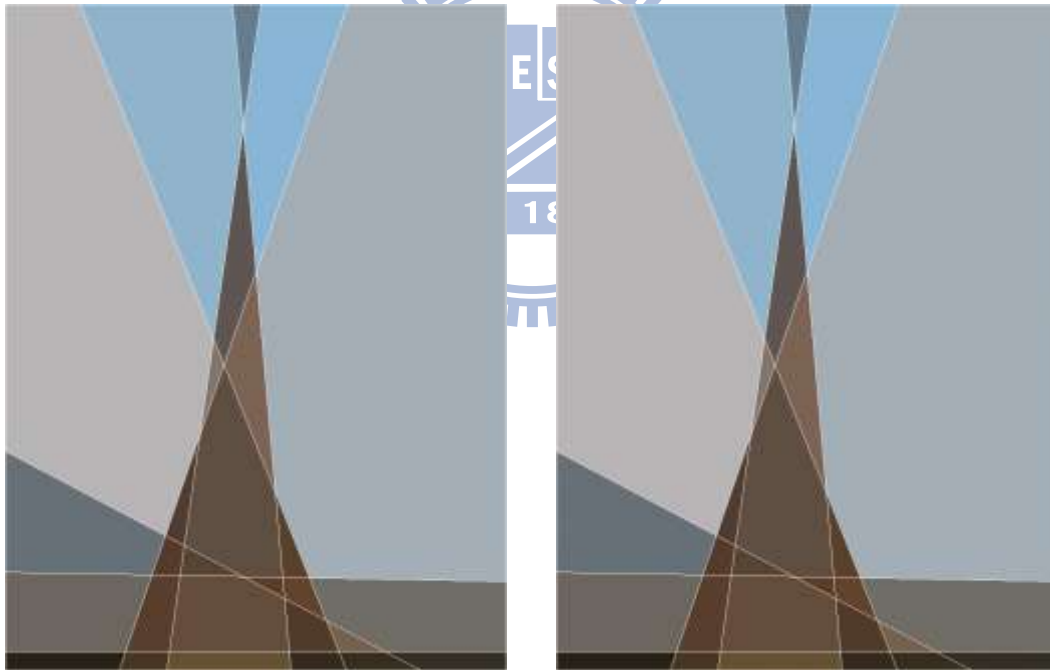


Figure 3.9 Extracted erroneous secret message with a wrong key “doo.”



(a)



(b)

(c)

Figure 3.10 An experimental result. (a) A source image (cover image). (b) A Cubism-like image without secret message embedding. (c) A stego-image of (a) by embedding the secret message “Meet me at 21:30. See you.” with the secret key “test.”

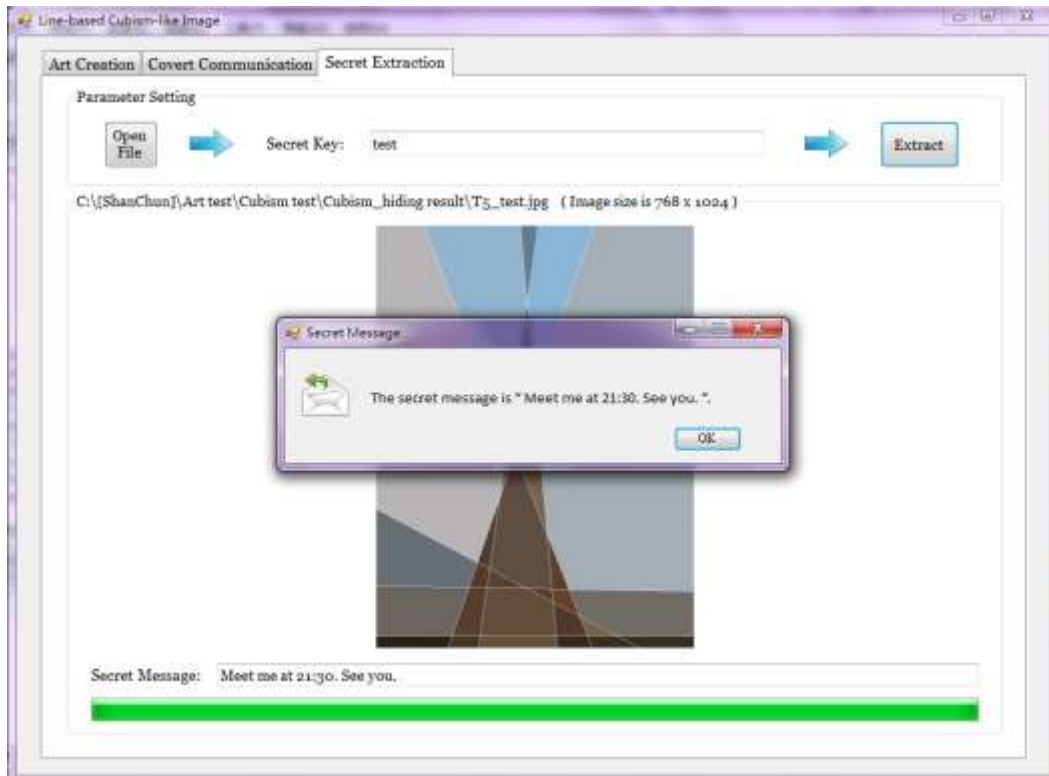


Figure 3.11 Extracting the secret message with the right secret key “test.”

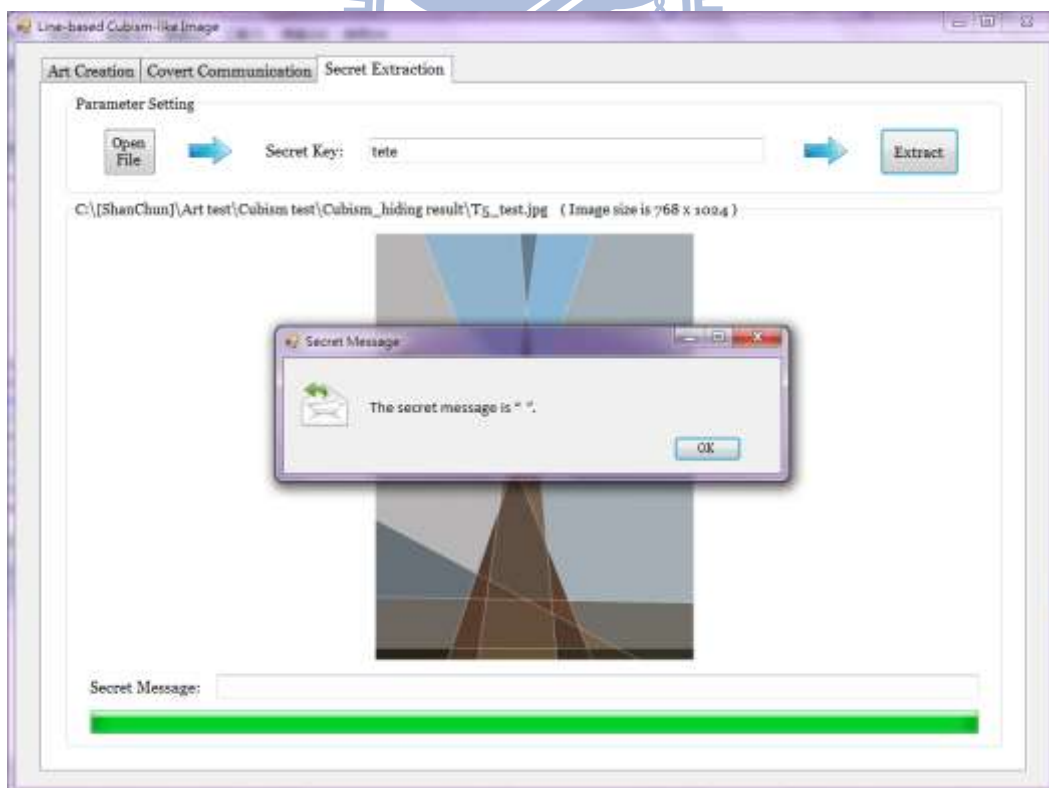


Figure 3.12 Extracted erroneous secret message with a wrong key “tete.”

Chapter 4

Strip-based Futurism-like Image --- A New Type of Image and Its Application to Data Hiding by Variable Sub-region Coloring

4.1 Overview of Proposed Method

A new type of computer art image, namely, strip-based Futurism-like image, is proposed in this study, which can be created automatically. The proposed method to create such images is described in this chapter. The aim is to pursue the sense of motion emphasized by the Futurism school. To create an image of this type, firstly we segment out prominent regions from a given image by a region merging scheme. Then, we extract some region features, such as corner point and region direction, by analyzing the chain codes of the region boundaries. Using the features, we extract the direction of the each region in the image. Finally, we partition each region into strips according to the extracted direction of the region. By using the strips of different directions in each region, an art image can be created, which looks like the type of motion pursued by Futurist painters by making the regions in the image look more vivid. The proposed method for automatic creation of such strip-based Futurism-like images will be described in detail in Section 3.2.

In addition, we propose also a data hiding technique for embedding secret messages into created Futurism-like art images in this study. This data hiding

technique is conducted in the process of creating the strips in the image. After partitioning each region into several strips, we hide a given secret message by coloring the sub-region with white color or the original color alternatively or randomly. Moreover, a scheme is proposed to enhance the security of the hidden data by randomizing the order of processing the regions and the bit data in the secret message.

4.2 Proposed Strip-based Futurism-like Image Creation Process

4.2.1 Idea of Proposed Creation Technique

Like Cubism, Futurist painters transform concrete shapes into abstraction with multiple viewpoints, but they emphasize more on the sense of movement and action. In this study, we follow the idea of movement in Futurism to create a type of art image, called Strip-based Futurism-like image. In order to accomplish the concept of motion, we use lines to represent the direction of a region, and form strips in regions by multiple lines with the same direction or roughly being so. When strips with different directions are gathered together, they are like a clash in the image which forms another aesthetic perception and arouses people's vivid feeling of the image.

In the following sections, we describe how to obtain the directions of regions in the creation process of a strip-based Futurism-like image from a given image. At first, we perform image segmentation by region merging to obtain the regions in the image. We then utilize polygon approximation to get simple geometric shapes of the regions. By analyzing the chain codes of the region boundary, we acquire some region features like the corner points and the region directions. Finally, a desired art image is

generated by partitioning each region into strips according to the region direction and coloring each strip with the original or white color alternatively.

In conclusion, the direction of movement is regarded an important factor in the art image, and we follow this concept in the proposed method to create the strip-based Futurism-like art image.

4.2.2 Proposed Art Image Creation Process

The main concept of the proposed method for Futurism-like image creation is to find the direction of each region in a given image and partition the region by its direction. In more detail, in the process of art image creation, firstly we segment the source image into regions by utilizing a region merging technique. After dividing the given image into several blocks in this way, we merge similar blocks based on some thresholds. To reduce the computation time, we transform 3D RGB colors into 1D ones via a function described by the following equation proposed by Lai and Tsai [17] and use this 1-D color in measuring the homogeneity of a region:

$$h(r', g', b') = b' + 8 \times r' + 64 \times g'. \quad (4.1)$$

After merging the blocks by comparing their 1-D colors according to a merging threshold value, we perform polygon approximation to make the region contours simpler by the use of the Freeman chain codes [23], as shown in Figure 4.1(a). In this study, we adjust the direction sequence of the Freeman chain codes for the convenience of computation by taking the left-upper point as the origin point (0, 0) of the image. The new direction sequence of the Freeman chain codes is shown in Figure 4.1(b).

Before using the chain codes, we apply erosion and dilation operations to the

Freeman chain codes to remove chain code looping in them. Firstly, we use the erosion operation to delete the isolated points. Then, we use the dilation to extend the region yielded by erosion to recover approximately the shape of the original regions. In this way, we can remove any looping in the chain codes. However, the erosion operation sometimes will partition a single region into multiple regions, which will not be re-connected back to be a single region. For example, in Figure 4.2(a), we show a case (shown as red blocks) that will cause the looping error in finding chain codes. Figure 4.2(b) shows the result of erosion (drawn as purple blocks) from Figure 4.2(a), and it is seen that the original region is divided into two. Figure 4.2(c) is the result of dilation (drawn as green blocks) from Figure 4.2(b).

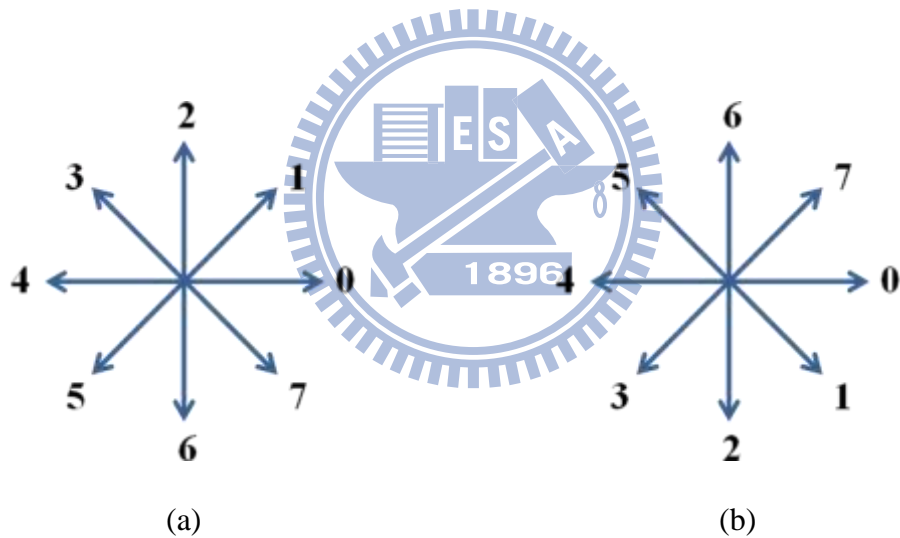


Figure 4.1 Chain codes. (a) The eight directions given by Freeman chain codes [23]. (b) The new eight directions used in this study.

Furthermore, we use chain codes to represent the shape of each region and utilize an expression (A_i, B_i) to record the information of a group of identical chain codes, called *coherent chain codes*, where A_i is the direction of the coherent chain code values i in a group and B_i is the number of the coherent chain code values i in the group. For example, given a chain code sequence $\{1100000222235555550777\}$, it can be represented as $\{(1, 2), (0, 5), (2, 4), (3, 1), (5, 7), (0, 1), (7, 3)\}$.

Then, we analyze the chain codes of each region to obtain the corner points of the region. Sanchez-Cruz [24] proposed a new method to represent shapes with only three symbols $C = \{0, 1, 2\}$ to save storage efficiently. The first symbol 0 means that the direction is the same as the previous one; the second, 1, indicates a direction change upward with the previous one (increasing the value of the vertical direction by regarding the previous one as horizontal); and the last one, 2, means to change backward with respect to the previous one (decreasing the value of the vertical direction by regarding the previous one as horizontal). To obtain the corner points, we use instead in this study a set $C = \{-1, 0, +1\}$ with three symbols to represent the different changes of directions in X and Y coordinates according to the above-mentioned idea of Sanchez-Cruz [24].

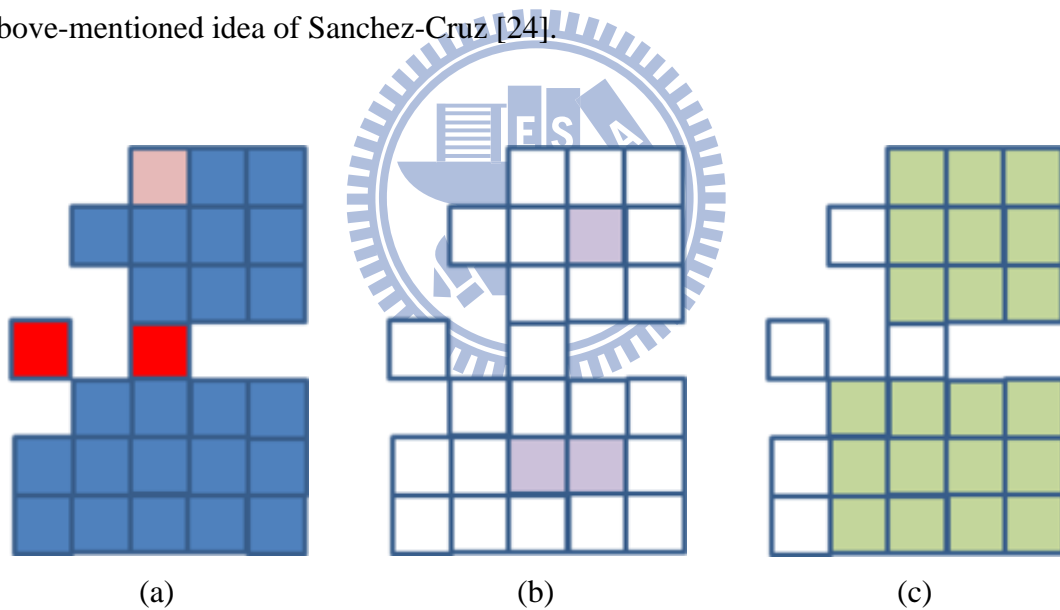


Figure 4.2 An example of looping error of chain codes. (a) A case of chain codes which yields a looping error. The pink grid is the start pixel of chain code tracing. (b) An erosion result of (a). (c) A dilation result of (b).

As an example, according to the direction sequence of Figure 4.1(b), we can create a table to record different direction changes, as shown in Table 4.1. Obviously, a direction change appears when the X or Y coordinates of the chain code change from $+1$ to -1 or from -1 to $+1$. Accordingly, corner points may be detected at these

positions of direction changes. In this way, we can find corner points from the chain codes of each region.

With the corner points of the region found, we can utilize them to adjust the edge of the region to get a polygon approximation of the region. Basically, we connect corner points to form new regions with simpler geometric shapes. For example, as shown in Figure 4.3(a), we use the point drawn in pink color which is nearest to the origin point as the start point of the sequence of chain codes, and the sequence of chain codes is “012222335567676.” Besides, we transform the sequence of chain codes into a sequence of *chain code segments* $\{(0, 1), (1, 1), (2, 4), (3, 2), (5, 2), (6, 1), (7, 1), (6, 1), (7, 1), (6, 1)\}$. Figure 4.3(b) shows the changes of X coordinates, and we can see that two corner points (shown as green blocks) appear at the corner from chain code 2 of the 3rd segment to chain code 3 of the 4th segment and at that from the 6th segment to the 7th segment. Figure 4.3(c) shows the changes of Y coordinates, and one corner point (shown as a green block) appears at the corner from the 4th segment to the 5th segment. As the start point is also a corner point, we can orderly connect all the corner points to obtain a new shape, as shown in Figure 4.3(d).

Table 4.1 A table between the direction of chain code and the change of X and Y coordinate.

| Direction | The change of X | The change of Y |
|-----------|-------------------|-------------------|
| 0 | + 1 | 0 |
| 1 | + 1 | + 1 |
| 2 | 0 | + 1 |
| 3 | - 1 | + 1 |
| 4 | - 1 | 0 |
| 5 | - 1 | - 1 |
| 6 | 0 | - 1 |
| 7 | + 1 | - 1 |

Finally, we utilize the corner points to find the direction of a region. Each pair of the corner point forms a vector. We classify these vectors into 10 groups which are

1/10 of 180 degrees, and we take the group with the maximum number of vectors as the desired direction of the region. According to the angles and directions of the vectors, we can partition the region into multiple strips. By coloring the strips, a strip-based Futurism-like image is created finally. A detailed algorithm describing this process is given as follows.

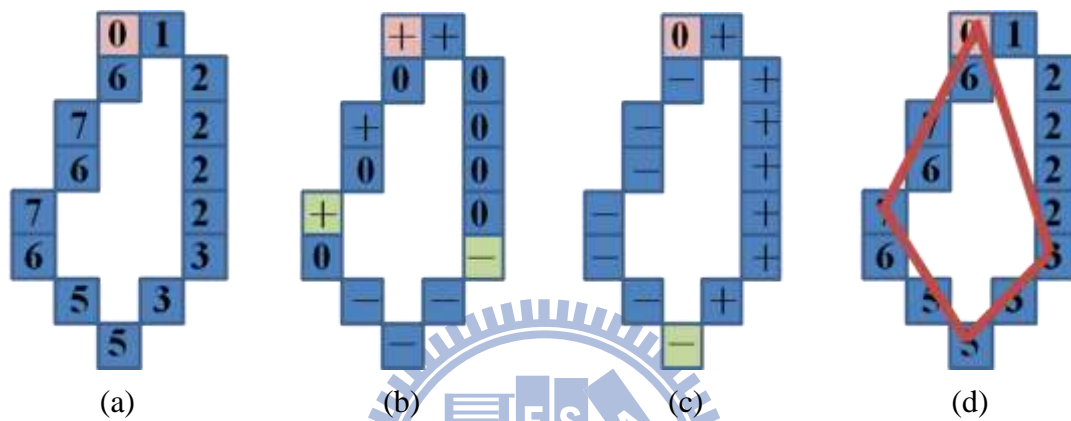


Figure 4.3 An example of finding corner points. (a) An example of chain codes. (b) The direction changes in X coordinate. (c) The direction changes in Y coordinate. (d) The result of connecting all corner points.

Algorithm 4.1: *strip-based Futurism-like image creation.*

Input: a source image S , and two threshold values — the strip width threshold W and the merging threshold T_m .

Output: a strip-based Futurism-like image F .

Steps.

Stage 1 --- finding the region direction.

Step 1. Compute the h -color h_c of the RGB color values (R_c, G_c, B_c) of each pixel in source image S by Equation (4.1) above, resulting in the following equation:

$$h(R_c, G_c, B_c) = B_c + 8 \times R_c + 64 \times G_c.$$

Step 2. Perform the following steps to conduct region merging to obtain the regions

R_1, R_2, \dots, R_n in S .

2.1 Divide S into 8×8 blocks.

2.2 Calculate the average h -color of each block.

2.3 Merge two blocks into one if the difference between their average h -colors is smaller than the merging threshold T_m .

2.4 Apply the erosion operation to delete isolated points, like Figure 4.2(b), and apply the dilation operation to remove chain code looping, like Figure 4.2(c).

2.5 Find the resulting regions by region growing and denote them by R_1, R_2, \dots, R_n .

Step 3. Perform the following steps to find the chain codes of each region $R_i, i = 1, 2, \dots, n$.

3.1 Set the point nearest to the origin point as the start point (s_{xi}, s_{yi}) of the chain codes of R_i .

3.2 Search the boundary of region R_i to get the sequence S_1, S_2, \dots, S_n of the chain codes of R_i by the directions illustrated in Figure 4.1(b) until the point reaches the start point.

3.3 Transform the sequence S_1, S_2, \dots, S_n into a sequence of chain code segments $E_i = \{(A_1, B_1), (A_2, B_2), \dots, (A_m, B_m)\}$.

Step 4. Perform the following steps to conduct polygon approximation using the sequence of chain code segments of each region $R_i, i = 1, 2, \dots, n$.

4.1 Analyze the chain codes $A_c, c = 1, 2, \dots, m$, of the chain code segment sequence $E_i = \{(A_1, B_1), (A_2, B_2), \dots, (A_m, B_m)\}$ of R_i to get the direction change sequences X_1, X_2, \dots, X_m and Y_1, Y_2, \dots, Y_m of the X and Y coordinates, respectively, according to Table 4.1.

4.2 Acquire the corner points in R_i by the following operations.

- (a) Create a corner point sequence $P_i = \{(P_{x1}, P_{y1}), (P_{x2}, P_{y2}), \dots, (P_{xk}, P_{yk})\}$ to record to the positions of the corner points to be found.
- (b) Assume that the start point (s_{xi}, s_{yi}) of each region R_i is the first corner point.
- (c) Set a segment c as a corner segment c' when the chain code value A_c of the sequence of chain code segments E_i , in the direction change sequence X_1, X_2, \dots, X_m changes from -1 to $+1$ or from $+1$ to -1 .
- (d) Set a segment c as a corner segment c' when the chain code value A_c of the sequence of chain code segment E_i in the direction change sequence Y_1, Y_2, \dots, Y_m changes from -1 to $+1$ or from $+1$ to -1 .
- (e) Obtain the position (P_{xk}, P_{yk}) of a corner point k in the corner point sequence P_i from a corner segment c by the following way, assuming that each of the functions F_x and F_y computes respectively, the changes of X and Y coordinates in direction according to Table 4.1, and that the previous corner segment is denoted as b' :

$$P_{xk} = P_{xk-1} + \sum_{i=b'+1}^{c'} F_x(A_i)B_i ;$$

$$P_{yk} = P_{yk-1} + \sum_{i=b'+1}^{c'} F_y(A_i)B_i .$$

- (f) Delete a corner point if the distance between two corner points is smaller than a quarter of the shorter of the region width and height.
- (g) Set the last point in the segment c as a corner point when the length B_c in the X direction of the chain code segment E_i is larger than a half of the region width or when the length B_c in the Y direction of the chain code segment E_i is larger than a half of the region height.

4.3 Connect the corner points of each region R_i in order as new edges of region R_i .

4.4 Obtain new regions, called *adjusted regions*, from the new edges and denote them as R_1', R_2', \dots, R_k' by filling pixels into the interiors of them and deleting the original pixels, which belong to the original regions but outside the adjusted regions.

4.5 Merge the remaining regions, which are residues after the adjustment of the last step, into the *nearest* adjusted regions R_1', R_2', \dots, R_k' according to the measure of distance between the average h -colors of the remaining region and the adjusted region.

Step 5. Re-compute the chain codes and the corner points of R_1', R_2', \dots, R_k' by repeating Steps 3 and 4.

Step 6. Perform the following steps to obtain the region directions of the adjusted regions R_1', R_2', \dots, R_k' , respectively, via the corner points in them.

6.4 Acquire an edge vector by connecting each pair of consecutive corner points in order.

6.5 Classify the edge vectors into ten groups, each group covering the range of 18 degrees in orientation.

6.6 Find the group with the maximum number of vectors, and take the direction of the vectors in the group as the *representative* direction of the region.

Stage 2 --- generating the desired Futurism-like image.

Step 7. Partition each adjusted region R_i' ($i = 1$ through k) into several strips according to the region direction of R_i' and the strip width W .

Step 8. Compute the average color C_1', C_2', \dots, C_k' of each adjusted region R_i' ($i = 1$ through k).

- Step 9. Color the *partitioning line* between every two strips by the black color.
- Step 10. Color the strips in each R_i' ($i = 1$ through k) with the average color C_i' or the white color *alternatively*.
- Step 11. Draw the pixels of the chain-code boundary of each region by the black color.
- Step 12. Take the final S as the desired strip-based Futurism-like image F .

In the above algorithm of strip-based Futurism-like image creation, we focus on the process of polygon approximation and finding the region direction by applying the chain codes. By connecting the corner points yielded by analyzing the chain codes, we can obtain a simpler shape of each region in the input image. The effect of polygon approximation also reaches the *geometric* style of Futurism school, as can be seen from the example of experimental results shown in Figure 4.4. Furthermore, we partition each region by its direction to pursue the *motion* style of Futurism. Some more experimental results will be given in the next section.

4.2.3 Experimental Results

According to the above methods, we set a *threshold of strip widths* to decide the width of each strip in each region. Because the feeling of beauty is different to everyone, we allow a user to select the width of strips in this study. People have three choices, which are small, middle, large strip widths. Different choices will result in different effects, so the user can choose their favorite type. Some created Futurism-like images using the above-proposed algorithms are given in Figures 4.5 through 4.7. From the images, we can see an abstract style of Futurism shown by the created simpler regions and strips in them.

Additionally, we can adjust the direction of strips in a region in other ways. For

example, we can use a strip line as a basis, and then change the angles of other strip lines with one end of the basis line as a pivot point. Some results produced in this way are shown in Figure 4.8. As can be observed, the images become vivid in another flavor. In the future, we may combine this way of strip generation with the previous one to increase the diversity of art image contents.



Figure 4.4 An example of polygon approximation by using the chain codes. (a) A source image. (b) The result of image segmentation by region merging of (a). (c) The result of polygon approximation by connecting the corner points of each region of (b).



Figure 4.5 Experimental results. (a) A source image. (b) A Futurism-like image created by (a) with strip width as small. (c) A Futurism-like image created by (a) with strip width as middle. (d) A Futurism-like image created by (a) with strip width as large.



(c)

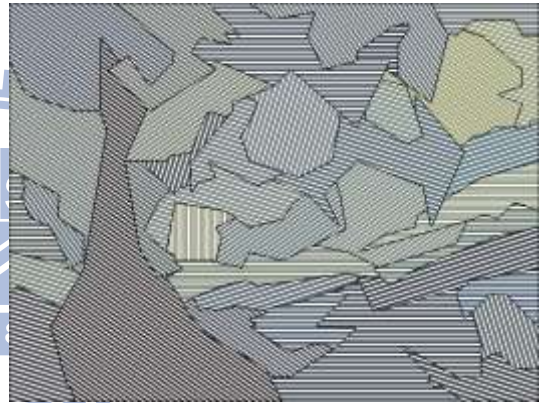


(d)

Figure 4.5 Experimental results. (a) A source image. (b) A Futurism-like image created by (a) with strip width as small. (c) A Futurism-like image created by (a) with strip width as middle. (d) A Futurism-like image created by (a) with strip width as large. (Continued.)



(a)



(b)



(c)



(d)

Figure 4.6 Experimental results. (a) A source image. (b) A Futurism-like image created by (a) with strip width as small. (c) A Futurism-like image created by (a) with strip width as middle. (d) A Futurism-like image created by (a) with strip width as large.

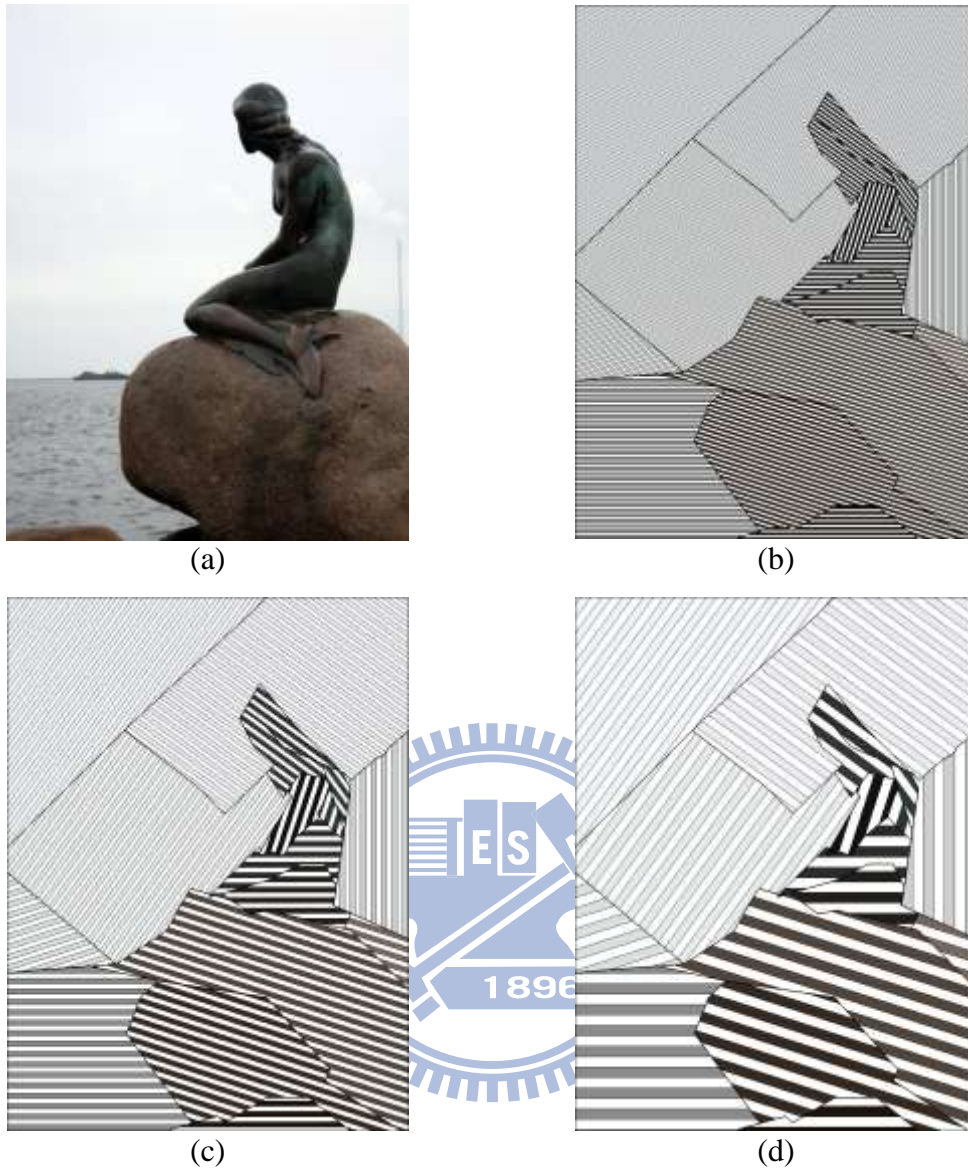


Figure 4.7 Experimental results. (a) A source image. (b) A Futurism-like image created by (a) with strip width as small. (c) A Futurism-like image created by (a) with strip width as middle. (d) A Futurism-like image created by (a) with strip width as large.



Figure 4.8 Experimental results of drawing strips in another way. (a) Three source images. (b) The experimental results.

4.3 Proposed Technique for Data Hiding in Strip-based Futurism-like Images by Variable Sub-region Coloring

4.3.1 Idea of Proposed Data Hiding Technique

In this section, the proposed method of using Futurism-like images for covert communication is presented. As described in Algorithm 4.1, we color the strips of each region with the region's average color and the white color alternatively. However, like some Futurists' style of *leaving white space* in paintings, it is also appropriate to color them occasionally with *successive* white or average colors. Accordingly, we propose a data hiding method in a cover image by coloring the strips of each region with a *message-dependent random order* of the white and the average colors. Consequently, people will feel the resulting stego-image as an art painting, and take no notice of the embedded message. It is in this way that we achieve the goal of data hiding in the proposed strip-based Futurism-like art image.

Additionally, in some extreme cases, the strips in a region may all be colored with the same color (the average color or the white color only), resulting in an art image felt strange by people. In order to avoid these situations, we adjust the secret message sequence by transforming every two bits into three one before embedding the message bits. As a result, in the stego-image, rare extreme cases will occur. The details of these techniques used in the proposed method for data hiding in the strip-based Futurism-like image will be given in the following sections.

4.3.2 Proposed Data Hiding Process

As mentioned previously, in order to implement the concept of covert

communication, we do not color the strip in a fixed alternative order, as mentioned previously. By using the bits of the secret message as the strip coloring order, we can hide the secret message and in the meantime achieve creation of varying styles in the resulting images. To avoid the extreme case of coloring a region with a successive sequence of identical colors, before hiding the messages, we adjust the secret message sequence by creating a mapping table to map every two bits of the original data to three bits, as shown in Table 4.2. Accordingly, we can increase the security and complexity by utilizing a secret key to randomize the mapping orders specified in the table.

Table 4.2 A mapping table between two bits of original data and three bits of new data.

| Original two bits | New three bits |
|-------------------|----------------|
| 00 | 101 |
| 01 | 100 |
| 10 | 110 |
| 11 | 010 |
| Ending pattern | 011 |

Besides, in order to recover the secret message, we have to record the start point of each region as the beginning of the chain code sequence. Also should be recorded are the next boundary points in order to avoid getting the wrong chain code sequence in the message extraction process. The detailed algorithm is described as follows.

Algorithm 4.2: *embedding a secret message into a strip-based Futurism-like image.*

Input: a cover image S , a secret key K , a secret message M , and two parameter values

— the strip width L_s and the merging threshold T_m .

Output: a stego-image I into which M is embedded.

Steps.

Stage 1 --- embedding a secret message M .

Step 1. Transform the secret message M , eight bits for each character, into a bit

sequence M' , and randomize the bits in M' by the secret key K .

- Step 2. Create a 2-bit-to-3-bit mapping table as shown in Table 4.2, with the mapping orders randomized by the secret key K .
- Step 3. Transform every two bits of M' into three bits according to Table 4.2, resulting in a new sequence M'' , and append an ending pattern (which is 011 in Table 4.2 is used) at the end of M'' to form a new bit sequence with a length of $(M'/2) \times 3 + 3$.
- Step 4. Divide M'' into a series of 3-digit segments m_1, m_2, \dots, m_n .
- Step 5. Perform Algorithm 3.1, using the input cover image S as the source image, to obtain the information of some parameters of the regions R_1, R_2, \dots, R_k in S , namely, the start points $(S_{x1}, S_{y1}), (S_{x2}, S_{y2}), \dots, (S_{xk}, S_{yk})$, the region directions D_1, D_2, \dots, D_k and the average RGB color values $(C_{1r}, C_{1g}, C_{1b}), (C_{2r}, C_{2g}, C_{2b}), \dots, (C_{kr}, C_{kg}, C_{kb})$ of the regions R_1, R_2, \dots, R_k , respectively.
- Step 6. Rearrange randomly the coloring order of the regions by the secret key K , resulting in a new coloring sequence $C_s = \{R_1', R_2', \dots, R_k'\}$; and change accordingly the corresponding orders of the start points, the region directions and the average RGB color values, resulting in the new orders of $(S_{x1}', S_{y1}'), (S_{x2}', S_{y2}'), \dots, (S_{xk}', S_{yk}'), D_1', D_2', \dots, D_k'$, and $(C_{1r}', C_{1g}', C_{1b}'), (C_{2r}', C_{2g}', C_{2b}'), \dots, (C_{kr}', C_{kg}', C_{kb}')$, respectively.
- Step 7. Partition each region R_i' into strips by the following steps.
- 7.1 Divide the region R_i' into strips by the region direction D_i' and the strip width L_s .
 - 7.2 Color the partitioning line by the black color.
 - 7.3 Color the strips in order according to the secret message sequence M'' in the following ways:
 - (a) color an uncolored strip with the nearly-white color (254, 255, 254)

if the bit of M'' is 0; or

(b) color an uncolored strip with the average color $(C_{ir}', C_{ig}', C_{ib}')$ if the bit of M'' is 1;

7.4 Repeat Steps 7.1 through 7.3 until the bit sequence M'' is exhausted, or until all regions are embedded but the bit sequence M'' is not exhausted; and for the latter case, show a message to tell the user the shortage of data-embedding space and exit.

Step 8. Partition other intact region R_j' , which has not been used for message bit embedding so far, into strips and color the strips with the average color $(C_{jr}', C_{jg}', C_{jb}')$ or the white color *alternatively*.

Step 9. Hide the start point (S_{xi}', S_{yi}') of each region R_i' , which is also the first point of the chain-code boundary points, (P_{x1}', P_{y1}') , and the next chain-code boundary points (P_{x2}', P_{y2}') by coloring them with the colors $(0, 0, 1)$ and $(1, 0, 0)$, respectively.

Step 10. Color the pixels of the chain-code boundary of each region by the black color.

Step 11. Take the final S as the desired stego-image I .

In the above algorithm, we use some colors to record the information such as the start points and the next boundary points, so the average color in each region will be adjusted in advance. In order to extract the right information, if one of the average RGB color values C_r , C_g , and C_b of a region is computed to be 0, which are used as special colors for various purposes in the algorithm above, reset it to be 1 before data hiding. To the user, the resulting slight color alternations in the generated stego-image cause nearly no visual effect.

4.3.3 Proposed Secret Message Extraction Process

Before a data extraction process, we have to find the start points of each region, and use the second points connecting to start points to get the first direction of the chain code. After obtaining the information, we can extract the all chain code sequence and recover the region. Then, we can use the chain code to reconstruct the process of partitioning the strips and acquire the secret message. In addition, we also have to extract the mapping orders in the mapping table by utilizing the secret key. Based on the mapping table, we can retrieve accordingly the secret message embedded in the stego-image. The detailed algorithm of secret data extraction is given as follows.

Algorithm 4.3: *extracting a secret message from a stego-image.*

Input: a stego-image S , and a secret key K identical to that used in Algorithm 4.2.

Output: the secret message M supposedly embedded in S .

Steps.

Stage 1 --- retrieving the information of the stego-image S .

Step 1. Scan the stego-image S to obtain the start points $(S_{x1}, S_{y1}), (S_{x2}, S_{y2}), \dots, (S_{xn}, S_{yn})$ and the next boundary points $(P_{x1}, P_{y1}), (P_{x2}, P_{y2}), \dots, (P_{xn}, P_{yn})$ in S by checking their colors:

1.1 Mark each pixel with color $(0, 0, 1)$ as a start point;

1.2 Mark each pixel with color $(1, 0, 0)$ as a next boundary point.

Step 2. Mark a pixel with black color $(0, 0, 0)$ as a boundary pixel of a region.

Step 3. Retrieve the coloring order of the regions by using the secret key K and denoted the retrieved coloring sequence as $C_s = \{R_1, R_2, \dots, R_n\}$.

Step 4. Acquire the chain code sequence of each region R_i via the start points $(S_{xi},$

S_{yi}) and the next boundary point (P_{xi}, P_{yi}) by Performing Step 3 of Algorithm 4.1.

Step 5. Perform Step 6 of Algorithm 4.1 to obtain the region direction D_1, D_2, \dots, D_n of each region.

Step 6. Obtain the strip width L_s by checking the distance of two neighboring lines of a strip in a region.

Stage 2 --- extracting the embedded secret message M .

Step 7. Create an *empty* bit sequence Q initially.

Step 8. Create a bit mapping table as shown in Table 4.2 with the mappings decided by the secret key K .

Step 9. Extract the secret message by reversing the process of partitioning strips in the following steps.

9.1 Use the region direction and strip width L_s to divide the region into strips.

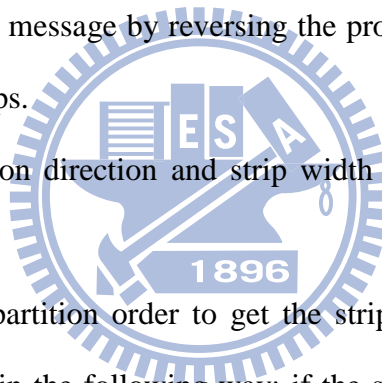
9.2 Follow the partition order to get the strip color, and extract a secret message bit in the following way: if the strip color is the white color, set the secret value to be 0; otherwise, to be 1.

9.3 Transform every three extracted bits into two bits of Q according to the bit mapping table.

9.4 Repeat Steps 9.1 to 9.3 until the three extracted bits is equal to the end pattern, 011.

Step 10. Use the secret key K to reorder Q .

Step 11. Transform every eight values of Q into a decimal number, resulting in a new sequence Q' , and then transform Q' into characters as the desired secret message M .



4.3.4 Security Consideration

The security of the proposed method is based on the random order of coloring the regions in the image creation process. By using the secret key, we can encrypt and reorder the secret message bits before data hiding. Moreover, we also use the secret key to randomize the corresponding order of the secret bits and create a mapping table (Table 4.2), so that without the key a malicious user cannot obtain the correct secret message. In these ways, we can strongly protect the secret messages using the proposed data hiding method; it is nearly impossible for a malicious user to retrieve the embedded secret message correctly even when the proposed algorithms are available to him/her.

4.3.5 Experimental Results

Some experimental results of applying the proposed method for covert communication via a Futurism-like image are shown in Figures 4.9 through 4.14. Figure 4.9(a) and Figure 4.12(a) are the source images, which are also cover images for data hiding. Figure 4.9(b) and Figure 4.12(b) are the Futurism-like images without an embedded secret message. By using Figure 4.9(a) and a secret key “Sun”, we can generate a stego-image into which a secret message “Make hay while the sun shines.” is embedded, as shown by Figure 4.9(c). Figure 4.12(c) is a stego-image into which a secret message “Learning makes life sweet.” has been embedded with the secret key “tower”. As shown in Figures 4.10 and 4.13, by using the right key, the secret messages are retrieved successfully. If we use a wrong secret key in the process of secret message extraction, the extracted secret messages will be incorrect, like Figures 4.11 and 4.14.

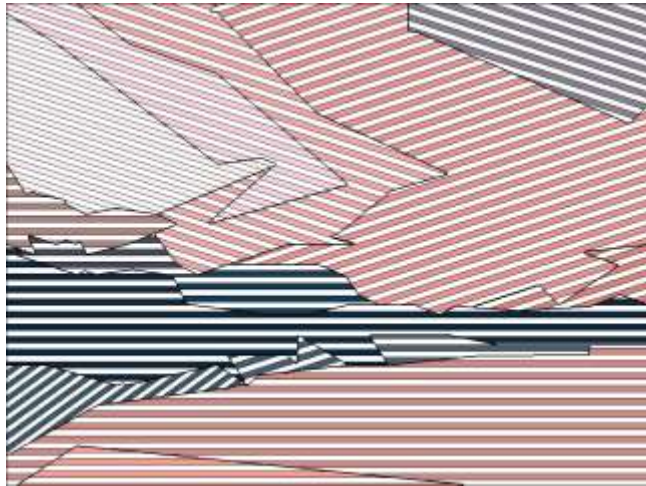
4.4 Summary

In this study, a type of computer art, called strip-based Futurism-like image is proposed. In order to create this style of art image, we have proposed an automatic method to make the region shapes in an input source image simpler by connecting the main corner points yielded by analyzing the chain codes of the region boundaries. Before obtaining the chain codes, we pre-process the input image by erosion and dilation operations to avoid some chain code looping cases. With the corner points, we can obtain the region direction by analyzing the direction of region edges. After partitioning each region into strips with its direction, we can obtain an abstract form with simple geometric shapes and the motion effect of the source image.

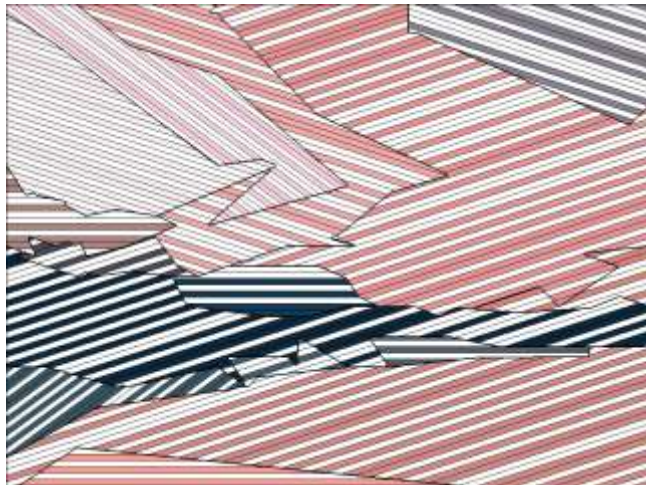
Furthermore, we have proposed a data hiding technique for covert communication in the stage of strip coloring in the image creation process. Based on variable coloring, we can embed a secret message into the strips of the regions in the image. To avoid coloring region strips with successive identical colors, we use a mapping table to transform the original message into another with smaller probabilities of successive identical bits. Besides, we randomize the secret message bit sequence and its processing order by a secret key to enhance the security of data hiding. Because the resulting Futurism-like image is like an abstract art image, people have no suspicion about the existence of the hidden message. As a result, the proposed data hiding method is feasible for covert communication applications.



(a)



(b)



(c)

Figure 4.9 An experimental result. (a) A source image (cover image). (b) A Futurism-like image without secret message embedding. (c) A stego-image of (a) by embedding the secret message “Make hay while the sun shines.” with the secret key “Sun.”

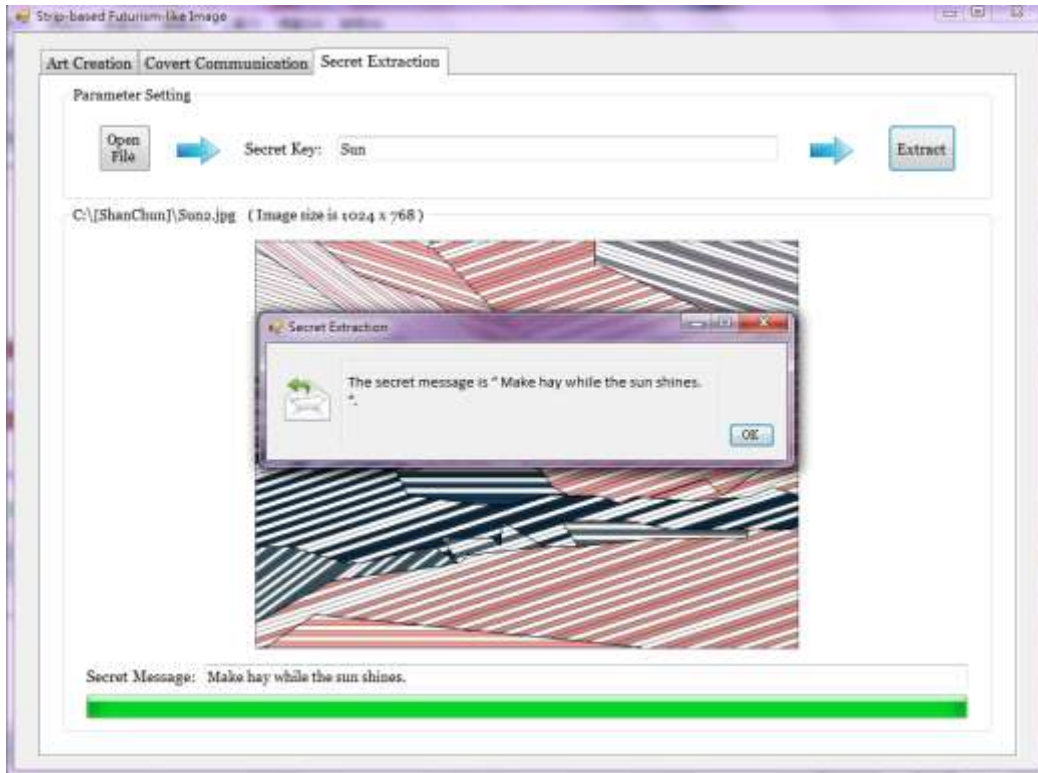


Figure 4.10 Extracting the secret message with the right secret key “Sun.”

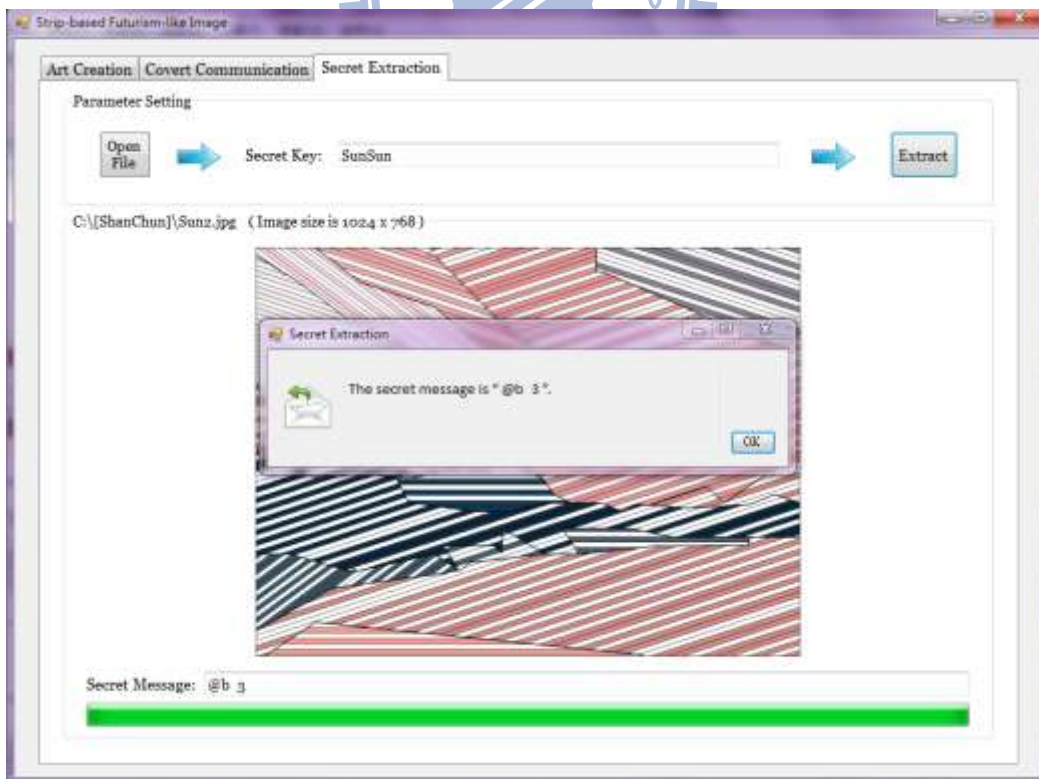


Figure 4.11 Extracted erroneous secret messages with a wrong key “SunSun.”



(a)



(b)

(c)

Figure 4.12 An experimental result with strip width as middle. (a) A source image (cover image). (b) A Futurism-like image without secret message embedding. (c) A stego-image of (a) by embedding the secret message “Learning makes life sweet.” with the secret key “tower.”

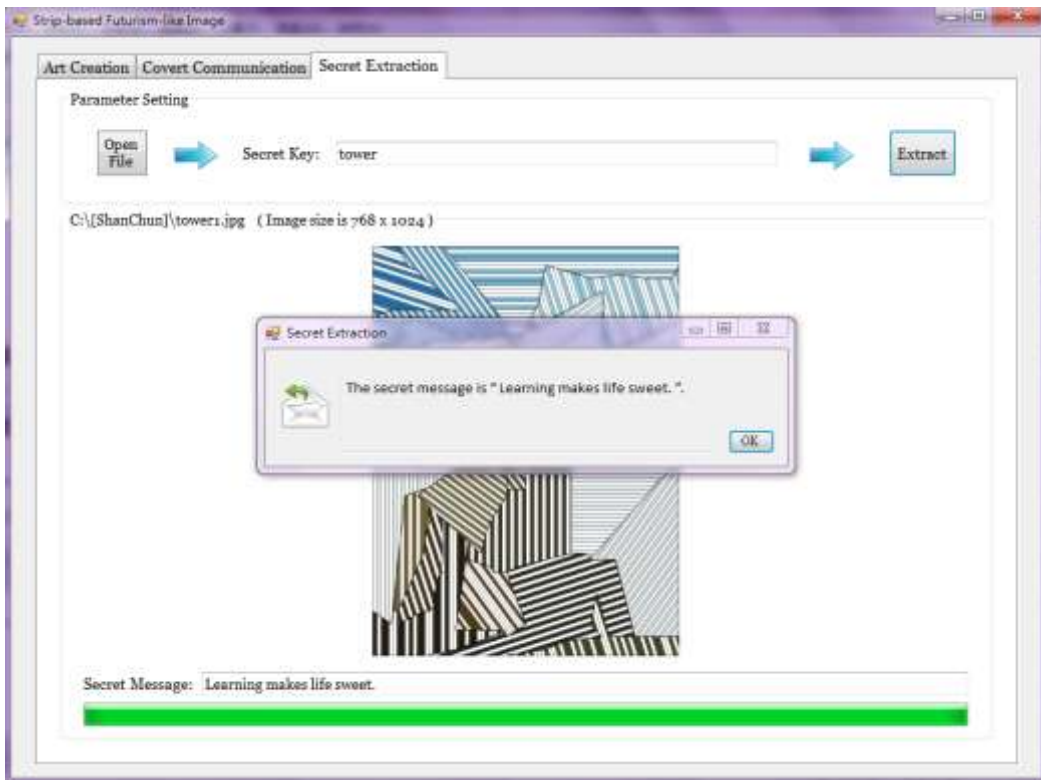


Figure 4.13 Extracting the secret message with the right secret key “tower.”

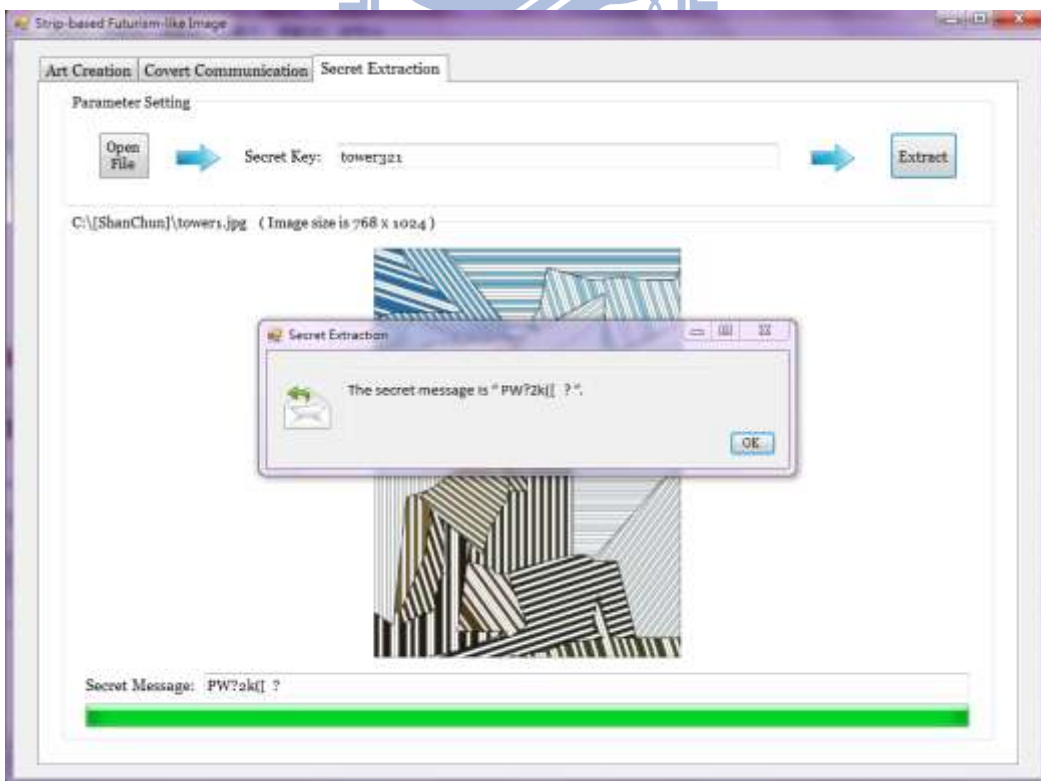


Figure 4.14 Extracted erroneous secret messages with a wrong key “tower321.”

Chapter 5

Rectangle-based Neo-Plasticism-like Image --- A New Type of Image and Its Application to Data Hiding by Binary Space Partitioning

5.1 Overview of Proposed Methods

The idea of the proposed methods described in this chapter comes from the paintings of Piet Mondrian who was an important contributor to the Neo-Plasticism art movement of the 20th century. He reduced the image content to geometric shapes by using only straight horizontal and vertical lines and rectangular shapes. In this chapter, we describe how to imitate the abstract style via a computer in the proposed methods, and we name this kind of image *rectangle-based Neo-Plasticism-like image*. To keep the characteristics of Neo-Plasticism such as lines, rectangles, and simple colors, we apply the idea of binary space partitioning to partition the image into multiple rectangles by using straight horizontal and vertical lines. Finally, a rectangle-based Neo-Plasticism-like image is created, which is like being composed of multiple-size rectangles. In Section 5.2, the detailed algorithm for automatic creation of such rectangle-based Neo-Plasticism-like images will be described.

Furthermore, in this study, we propose also an information hiding technique for embedding secret messages into created Neo-Plasticism-like art images. In the process of building the binary partition tree, a given message is embedded into a

rectangle-based Neo-Plasticism-like image, which influences the construction of the binary partition tree. According to the different tree, the partitioning result of the Neo-Plasticism-like image will also be different, but the characteristics of Neo-Plasticism school are still kept. Besides, we randomize the building priority of the two sides (either the upper and lower sides, or the left and right sides) of a partitioning line by using a secret key to enhance the security of the hidden data.

5.2 Proposed Rectangle-based Neo-Plasticism-like Image Creation Process

5.2.1 Idea of Proposed Creation Technique

Proponents of Neo-Plasticism express an ideal idea of spiritual harmony and order. They pursue pure abstraction by reducing a natural scene to the essentials of form and color such as vertical and horizontal lines, three primary colors, and three non-primary colors. The idea of the proposed art image creation method is inspired by this concept of Neo-Plasticism, as mentioned previously.

In the creation process of a rectangle-based Neo-Plasticism-like image from a given image, at first we have to use some criteria to decide where to partition in the image. Rigau, et al. [25] proposed a method to conduct image segmentation via the use of the mutual information (MI) of the spatial positions and the image intensities. In this study, we use the same concept to partition the image into multiple rectangles.

In information theory, the mutual information of two random variables is a quantity that measures the mutual dependence of the two variables. High mutual

information indicates the high dependence between two variables. The mutual information between two discrete random variables X and Y is defined as

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x, y) \log\left(\frac{p(x, y)}{p(x)p(y)}\right), \quad (5.1)$$

where $p(x, y)$ is the joint probability distribution function of X and Y , and $p_1(x)$ and $p_2(y)$ are the probability distribution functions of X and Y , respectively. Furthermore, let $p(x, y) = p(y|x)p(x)$. Then Equation (5.1) can be reduced to be

$$I(X;Y) = \sum_{x \in X} p(x) \sum_{y \in Y} p(y|x) \log\left(\frac{p(y|x)}{p(y)}\right). \quad (5.2)$$

In this study, we use the same type of mutual information between the spatial positions and the image intensities proposed by Rigau, et al. [25]. Given an image with N pixels and an intensity histogram with n_i pixels in bin i , we define X to represent the bins of the histogram with probability distribution $p(x_i) = n_i/N$; and then Y to be the spatial region with probability distribution $p(y_j) = N_j/N$, assuming that N_j is the area of region j . As shown in Figure 5.1, assume that we partition a region into two sub-regions y_1 and y_2 with the area N_1 and N_2 via a vertical line and the number of bins is 128. Then we can compute the value of mutual information by

$$I(X;Y) = \sum_{i=0}^{127} p(x_i) \left(p(y_1|x_i) \log\left(\frac{p(y_1|x_i)}{p(y_1)}\right) + p(y_2|x_i) \log\left(\frac{p(y_2|x_i)}{p(y_2)}\right) \right).$$

Accordingly, for each partition line (vertical and horizontal), we apply Equation (5.2) to calculate the mutual information. And then we find the maximum mutual information, which means the maximum information quantity between spatial position and intensities, to obtain an appropriate position to partition.

In more detail, in each region, we have to compute the probability of bins of the intensity histogram, so the computation is huge. In order to reduce the computation time, first we transform the 3-D RGB color into the 1-D color via Equation (4.1)

proposed previously, and use this 1-D color as the intensity of the pixels in the image. Besides, we quantize the color to limit the number of bins to be 128. Finally, we speed up the computation of mutual information by the following two ways: (1) trimming the first 10 pixels and the last 10 of each of the four boundaries of the region because the partition line will normally not to be found near the two ends of any boundary; (2) rescaling the image down to 1/9 for computing the mutual information.

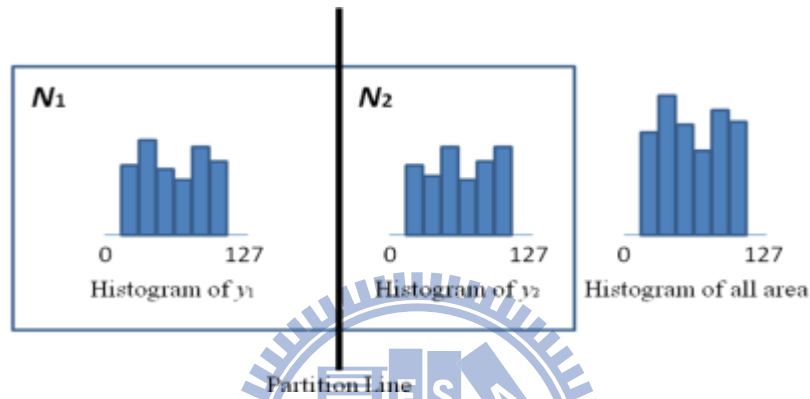


Figure 5.1 An example of partitioning.

As mentioned above, we use the mutual information to divide the image. However, if the region is smooth and similar, it is not necessary to partition it. Therefore, we use a criterion to determine whether to partition a region or not. According the existing information --- the intensity histogram, we use the following equation to be a similarity measure of the partition result with respect to the original region:

$$\frac{\sum_{i=0}^{127} |H_1(i) - H_2(i)|}{N} \times \frac{B_n}{B} \quad (5.3)$$

where N is the number of pixels in a region, H_1 is the intensity histogram of a sub-region after partitioning with a line, H_2 is that of the other one, B is the number of bins of the intensity histogram, and B_n is the number of bins with non-zero values. If

two sub-regions in the intensity histogram are completely different and the value of each bin is not zero, the value of this similarity measure will be 1. In this study, we use 0.05 as the lower bound of similarity measures. Accordingly, a region with smooth color will not be partitioned, as shown in Figure 5.2. By coloring partitioning regions recursively according to this similarity measure, a rectangle-based Neo-Plasticism image is created. This process accomplishes the concept of Neo-Plasticism school by transforming the input image into an abstract form. In the following sections, the proposed method of the above-mentioned processes will be described in detail.

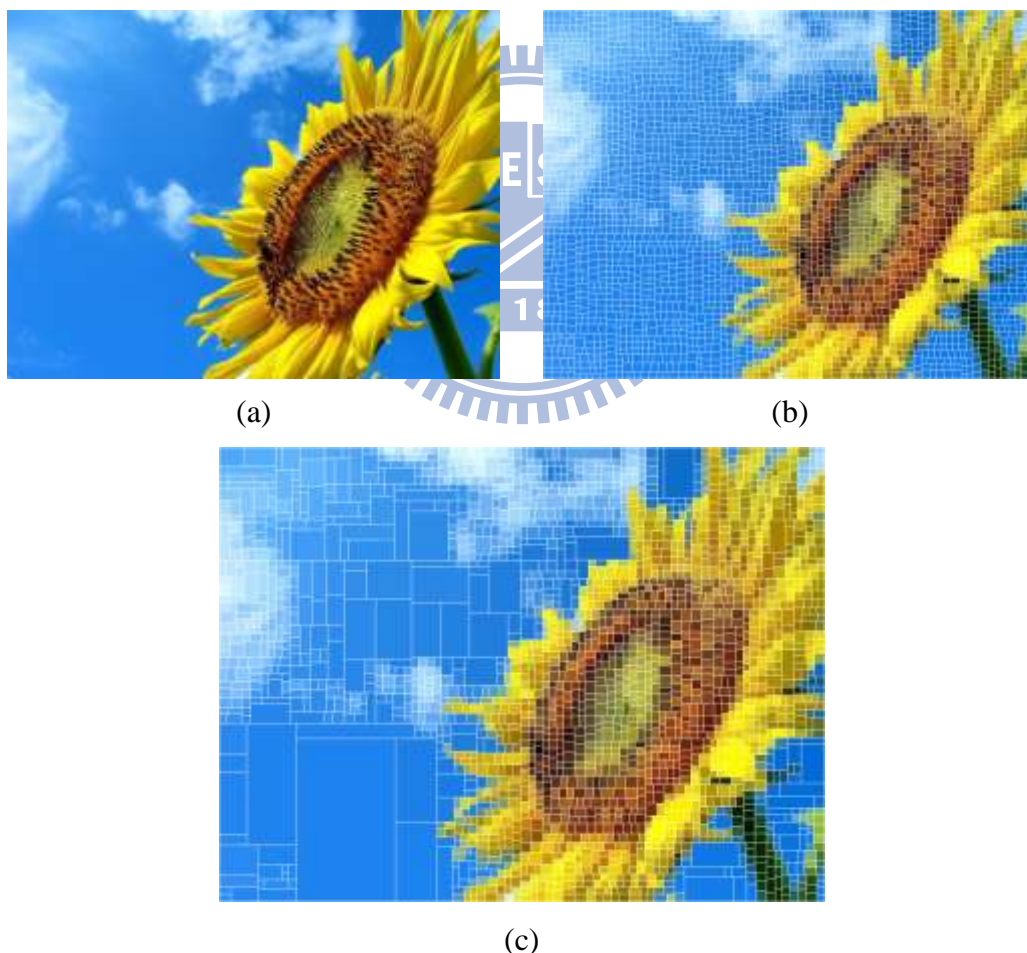


Figure 5.2 An example about the similarity measure. (a) A source image (b) The result of (a) with the partition iteration to be 30. (c) The result of (a) with the partition iteration to be 30 by using 0.05 as the lower bound of similarity measure.

5.2.2 Proposed Art Image Creation Process

In this section, an algorithm is proposed to implement the proposed idea of Neo-Plasticism-like image creation. From a given image, we calculate the mutual information in each partition process according to Equation (5.2), and then partition the image at a position in the image, where the corresponding mutual information value is the maximum to make the image into two sub-regions. Then, for the two sub-regions, we repeat the above process to find the maximum mutual information to partition the image iteratively. By coloring each of the partitioned regions with its average color, a desired Neo-Plasticism-like image is created. The detailed algorithm is given as follows.

Algorithm 5.1: *rectangle-based Neo-Plasticism-like image creation.*

Input: a source image S , and a threshold of partition iterations, I .

Output: a rectangle-based Neo-Plasticism-like image O .

Steps.

Stage 1 --- partitioning the image.

Step 1. Compute the h -color h_c of the RGB color values (R_c, G_c, B_c) of each pixel in source image S by Equation (4.1), resulting in the following equation:

$$h(R_c, G_c, B_c) = B_c + 8 \times R_c + 64 \times G_c.$$

Step 2. Quantize the h -color into the range of 0 to 127.

Step 3. Assume that the source image S is the first input region R .

Step 4. Create the probability density function $P_{\text{all}}(c_z)$ of the h -color intensity histogram C_{all} of the input region R .

Step 5. Perform the following steps to partition the input region R with the region boundaries being described by the two corners at coordinates $(X_{\text{min}}, Y_{\text{min}})$

and (X_{max}, Y_{max}) .

5.1 Calculate the maximum mutual information in the horizontal direction from $Y_{min} + 10$ to $Y_{max} - 10$ in the following way.

- (a) Set $Y_{min} + 10 + i$ as a partition line l ($i=0, 3, 6, \dots$).
- (b) Compute the probability density function $P_{c1}(c_x)$ and $P_{c2}(c_y)$ of two intensity histograms C_1 and C_2 of two sub-regions (the upper side and lower side of partition line l) and the probability density function P_{A1} and P_{A2} of the areas A_1 and A_2 of two sub-regions.
- (c) Calculate the mutual information by Equation (5.2) to be

$$M_{hm} = \sum_{x=0, y=0, z=0}^{127} P_{all}(c_z) [P_{c1}(c_x) \log\left(\frac{P_{c1}(c_x)}{P_{A1}}\right) + P_{c2}(c_y) \log\left(\frac{P_{c2}(c_y)}{P_{A2}}\right)].$$

- (d) Select the maximum mutual information M_{hmax} from $M_{h1}, M_{h2}, \dots, M_{hm}$ and record the corresponding partition line as h .
- (e) Calculate the similarity value E_h with partition line h by Equation (5.3).

5.2 Calculate the maximum mutual information in the vertical direction from $X_{min} + 10$ to $X_{max} - 10$ in the following way.

- (a) Set $X_{min} + 10 + i$ as a partition line l ($i=0, 3, 6, \dots$).
- (b) Compute the probability density function $P_{c1}(c_x)$ and $P_{c2}(c_y)$ of two intensity histograms C_1 and C_2 of two sub-regions (the left side and right side of partition line l) and the probability density function P_{A1} and P_{A2} of the areas A_1 and A_2 of two sub-regions.
- (c) Calculate the mutual information by Equation (5.2).

$$M_{vm} = \sum_{x=0, y=0, z=0}^{127} P_{all}(c_z) [P_{c1}(c_x) \log\left(\frac{P_{c1}(c_x)}{P_{A1}}\right) + P_{c2}(c_y) \log\left(\frac{P_{c2}(c_y)}{P_{A2}}\right)].$$

- (d) Select the maximum mutual information M_{vmax} from M_{v1}, M_{v2}, \dots ,

M_{vn} and record the partition line as v .

(e) Calculate the similarity value E_v with partition line v by Equation (5.3).

5.3 Compare two maximum mutual information M_{hmax} and M_{vmax} of the horizontal and vertical directions and choose the maximum one as M_{max} with the corresponding position as C_{max} .

5.4 Partition the image by a line at the position of C_{max} if the similarity value (E_h or E_v) of M_{max} is bigger than 0.05.

5.5 Repeat Steps 4 and 5 with each of the two sub-regions as the input region R , and decrease partition iteration threshold I by 1 until it reaches the value of 0.

Stage 2 --- coloring image regions.

Step 6. Compute the average color of each partitioned region.

Step 7. Color each partitioned region by its average color.

Step 8. Color the partition lines by the white color.

Step 9. Take the final S as the desired rectangle-based Neo-Plasticism-like image O .

5.2.3 Experimental Results

In the above-mentioned method of art image creation, we know that different selections of the partition iteration number will result in different effects. In this study, we allow the user to select this iteration number because everybody's feeling for art is different. Some created Neo-Plasticism-like images yielded by different numbers of partition iterations using the above-proposed algorithms are shown in Figures 5.3 through 5.5. From the images, we can see an abstract style of Neo-Plasticism shown by using smaller partition iteration numbers. On the other hand, by using the larger

partition iteration numbers, the image will be more similar to the source image with a mosaic effect.

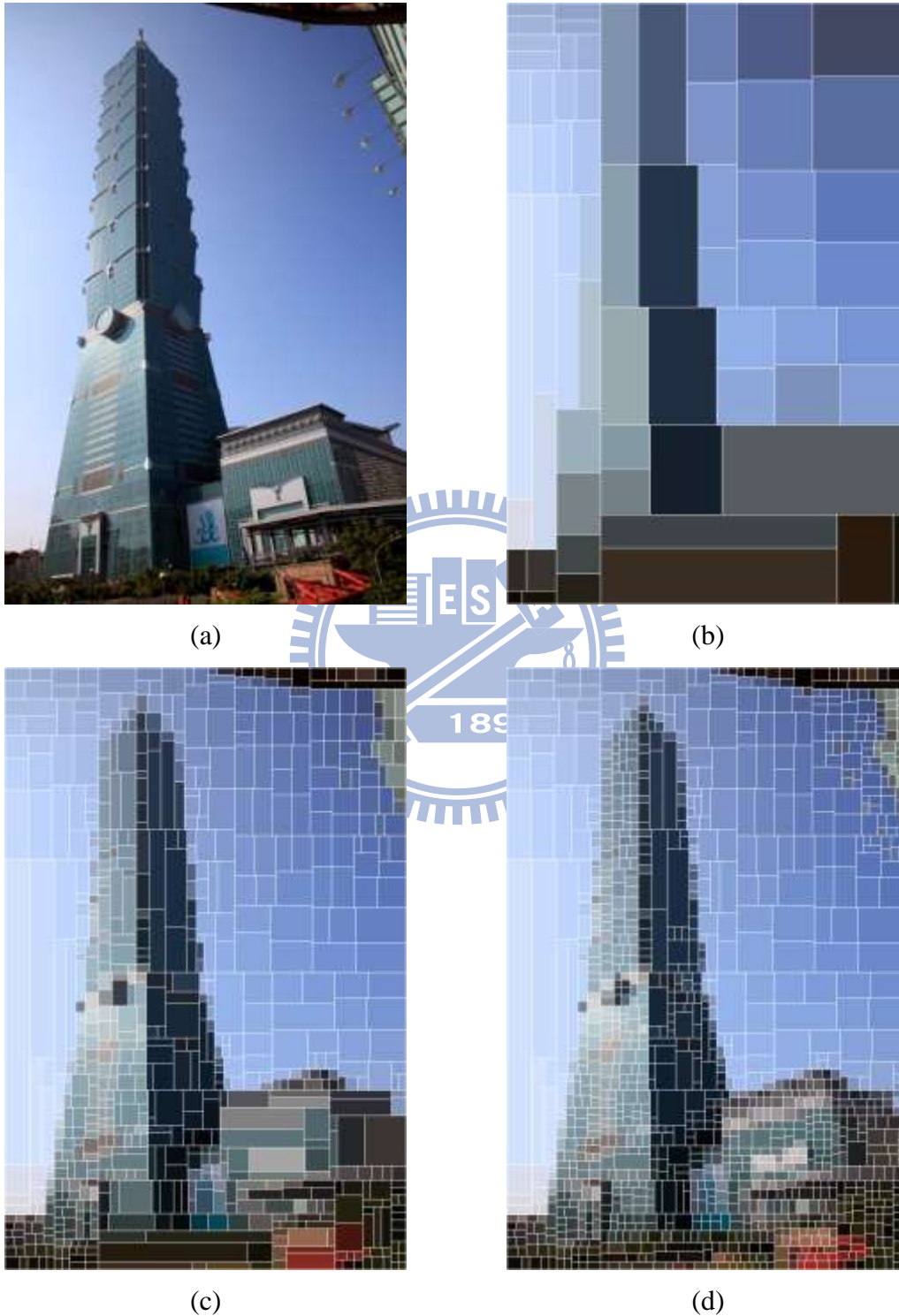


Figure 5.3 Experimental results. (a) A source image. (b) A Neo-Plasticism-like image created by (a) with partition iteration to be 5. (c) A Neo-Plasticism-like image created by (a) with partition iteration to be 10. (d) A Neo-Plasticism-like image created by (a) with partition iteration to be 15.

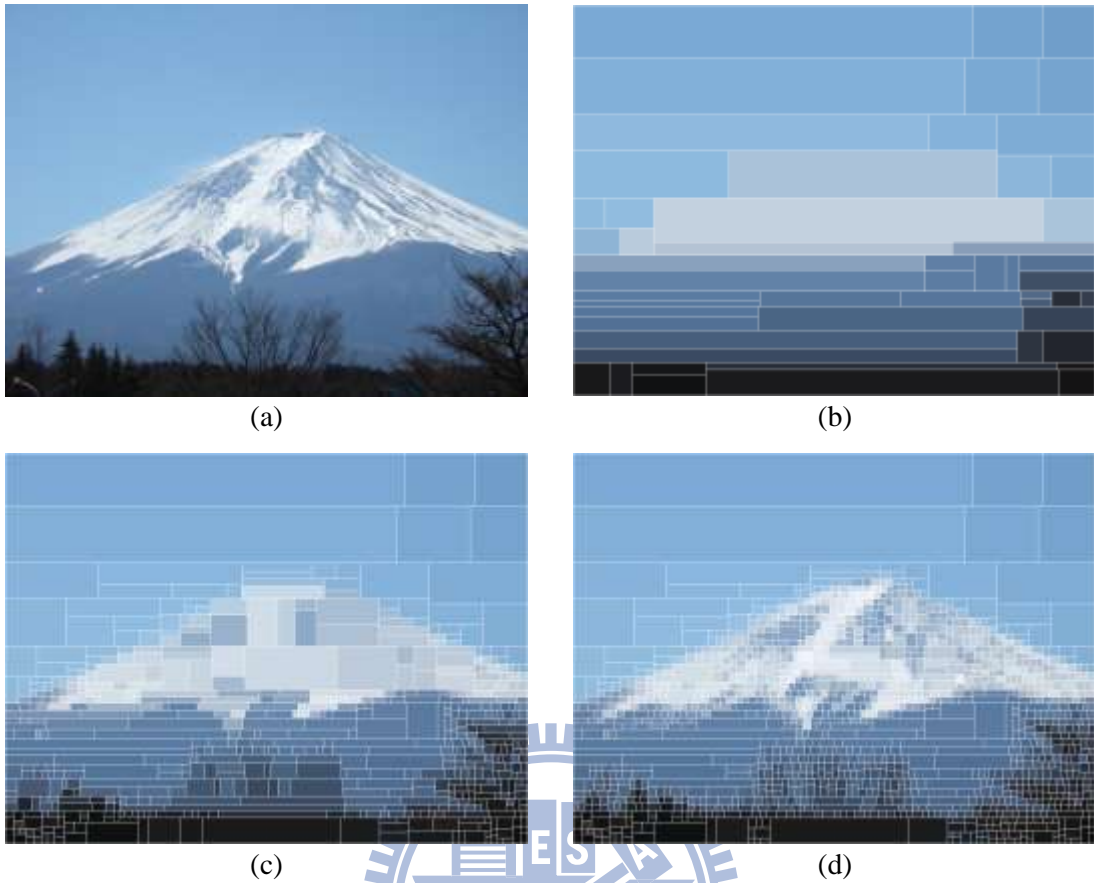


Figure 5.4 Experimental results. (a) A source image. (b) A Neo-Plasticism-like image created by (a) with partition iteration to be 5. (c) A Neo-Plasticism-like image created by (a) with partition iteration to be 10. (d) A Neo-Plasticism-like image created by (a) with partition iteration to be 15.

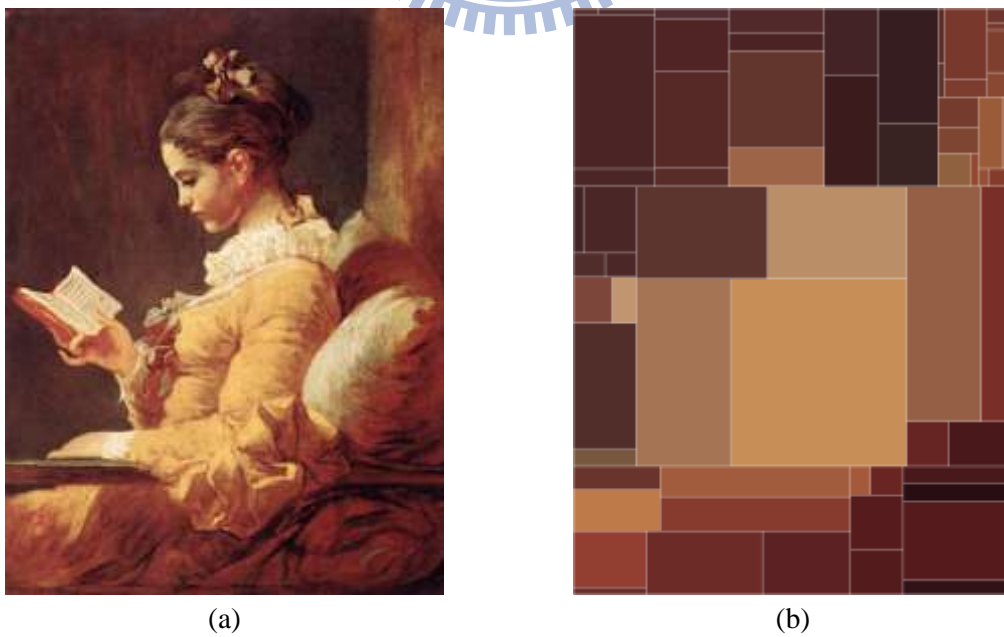


Figure 5.5 Experimental results. (a) A source image. (b) A Neo-Plasticism-like image created by (a) with partition iteration to be 5. (c) A Neo-Plasticism-like image created by (a) with partition iteration to be 10. (d) A Neo-Plasticism-like image created by (a) with partition iteration to be 15.



(c)



(d)

Figure 5.5 Experimental results. (a) A source image. (b) A Neo-Plasticism-like image created by (a) with partition iteration to be 5. (c) A Neo-Plasticism-like image created by (a) with partition iteration to be 10. (d) A Neo-Plasticism-like image created by (a) with partition iteration to be 15. (Continued.)

5.3 Proposed Technique for Data Hiding in Rectangle-based Neo-Plasticism-like by Binary Space Partitioning

5.3.1 Idea of Proposed Data Hiding Technique

The proposed method of using Neo-Plasticism-like images for covert communication is presented in this section. In the proposed Neo-Plasticism-like image creation process as described by Algorithm 5.1 above, we can see that a given image is partitioned by the binary space partitioning scheme which finds the maximum mutual information to decide the partition line. After partitioning the given

image for several times, we can build a partition tree to record the partitioning result and each leaf node of the tree is a rectangular region (partitioned region), as shown in Figures 5.6(a) and 5.6(b). Accordingly, we can embed a secret message into this tree structure. By limiting the partition directions (horizontal or vertical) to be in an alternative order, we can obtain the identical tree in the recovery process, as shown in Figures 5.6(c) and 5.6(d).

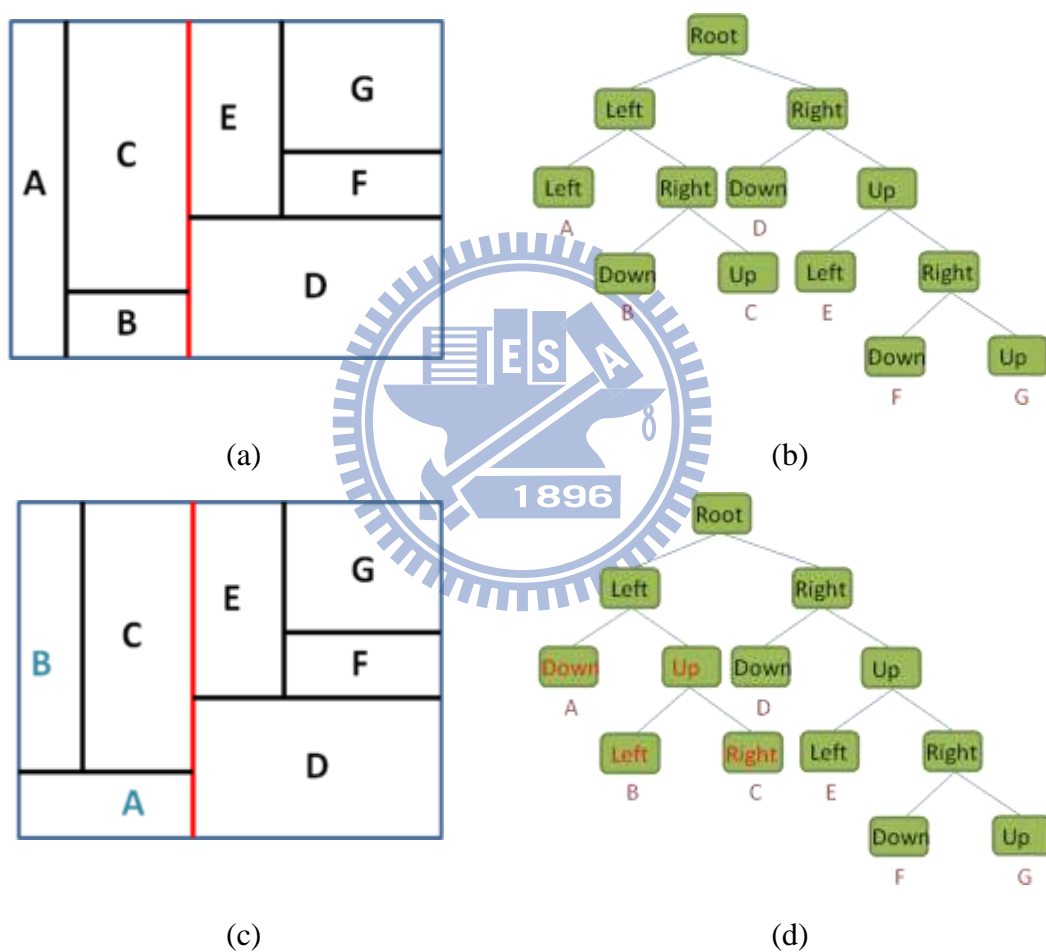


Figure 5.6 An example of building a partition tree. (a) A partitioned region with the red line as the first partition line. (b) A partition tree which is built according to (a). (c) A partitioned region with alternate order of horizontal and vertical direction. (d) A partition tree which is built according to (c).

In this study, we propose two methods to hide to secret messages. One, called Method 1, embeds the message into the LSBs of the average RGB color values of

each partitioned region based on the limited tree structure when the partition iteration number is large. However, an extreme situation occurs when the partitioned regions are few. In that case, we can only embed a few message bits. The other method, called Method 2, is proposed for the case of a small partition iteration number by coloring the regions with a binary space-partition order. Moreover, this method is not limited by the partition direction. As shown in Figure 5.7, we use two similar average colors to represent the vertical and horizontal colors, and color the region by these colors with some limited order. We embed the secret bit 0 by coloring the horizontal line at the middle position, and 1 by the vertical line. For example, we embed the secret message bits “010010” into the region D in Figure 5.7. Due to the characteristic that the vertical and horizontal colors are similar, users cannot notice the difference in the resulting image. In the following sections, these techniques used in the proposed method for data hiding in the rectangle-based Neo-Plasticism-like image will be described in detail.

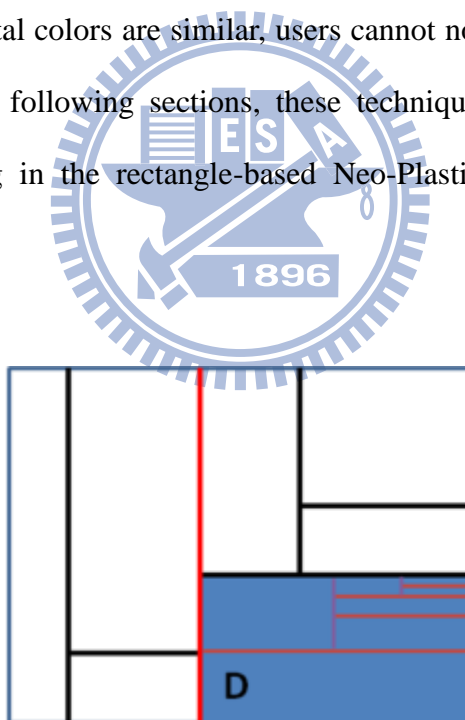


Figure 5.7 An example of coloring by binary space partition order.

5.3.2 Proposed Data Hiding Process

As mentioned previously, in order to implement the concept of covert communication, we propose two data hiding methods based on the creation process of

the strip-based Neo-Plasticism-like image. According to the process of partitioning the image, we can build a partition tree to record the partitioning result, as shown in Figure 5.6(b), and each leaf node of the tree is just a partitioned region. By using fixed partition directions, an identical partition tree will be built. Specifically, we make the partition in the alternative order of horizontal and vertical directions. Finally, we hide the secret message bits into the LSBs of the average colors of the partitioned regions.

A disadvantage of this method is that the partition directions will be limited, and may cause conflict to the content of the original image. The conflict will decrease when the partition iteration number is large, because the image will be partitioned into more small rectangles. Besides, if the partition iteration number is small, the amount of embedded message bits resulting from the first method will also be less. The details are described in Algorithm 5.2 below.

The second method is proposed for hiding a larger amount of secret message bits, and this method can remove the limit on the partition direction. Due to the characteristic of the human visual system, we can slightly change the color in each region without arousing notice from an observer. We color the regions with some fixed order according to a binary space-partition scheme. We use two similar colors to color the regions with vertical or horizontal directions, and build a binary coloring tree for each region. For the purpose of successful recovery, in each coloring, we limit the coloring position to be the one in the middle of region. If the sub-region is smaller than 2×2 , we do not color it. Finally, we color the un-colored pixels in a region by the average color. The detailed algorithm is described in Algorithm 5.3. Furthermore, for the two data hiding methods, we use a secret key to decide the priority of the building direction (upper and lower sides, or left and right sides) of the tree.

Algorithm 5.2: embedding a secret message into a rectangle-based

Neo-Plasticism-like image by Method 1.

Input: a cover image S , a secret key K , a secret message M , and a partition iteration threshold I .

Output: a stego-image O into which M is embedded.

Steps.

Stage 1 --- transforming the secret message M into 3-digit segments.

Step 1. Transform the secret message M , eight bits for each character, into a bit sequence M' , and randomize the bits in M' by the secret key K .

Step 2. Transform each bit of M' into a digit, resulting in a digit sequence M'' , and append an ending pattern with at least one and no more than three identical digits other than 0's and 1 (such as 2, 22, or 222) to the end of M'' to form a new digit sequence with its length being a multiple of three, as illustrated in Figure 3.6.

Step 3. Divide M'' into a series of 3-digit segments m_1, m_2, \dots, m_n .

Stage 2 --- partitioning the source image and building a partition tree.

Step 4. Perform Steps 4 and 5 of Algorithm 5.1, using the input cover image S as the input region and limiting the partition directions (horizontal or vertical) to be in an alternative order, to obtain the partitioned regions R_1, R_2, \dots, R_k .

Step 5. Use the secret key K to determine the priority of the partition direction (upper and then lower sides, or left and then right sides) of the tree.

Step 6. Build a partition tree T according to the priority of the partition direction and the partition result of Step 4.

Stage 3 --- using a stack to obtain a hiding sequence.

Step 7. Set up a stack to record a hiding sequence H of the leaf nodes of the partition tree T to hide the message later in the following way.

- 7.1 Push the root of partition tree T into the stack.
- 7.2 Pop the top value v in the stack and push two children of v into the stack according to the priority (the one with the highest priority is pushed in at last).
- 7.3 Repeat the above steps until all the nodes in the partition tree T are visited.
- 7.4 Take the sequence in the stack to be the desired hiding sequence H .

Stage 4 --- embedding a secret message M .

- Step 8. Compute the average RGB color values $(C_{1r}, C_{1g}, C_{1b}), (C_{2r}, C_{2g}, C_{2b}), \dots, (C_{kr}, C_{kg}, C_{kb})$ of the regions R_1, R_2, \dots, R_k , respectively.
- Step 9. Embed each 3-digit segment m_t of the digit sequence M'' into a region R_i according to the hiding sequence H in the following way.
 - 9.1 Adjust the LSBs of the RGB color values of all pixels in R_i by the segment m_t , assuming that the three digits in m_t are denoted as b_r, b_g , and b_b .
 - (a) Set the LSB of the R color value to be 0 if $b_r = 0$; otherwise, to be 1.
 - (b) Set the LSB of the G color value to be 0 if $b_g = 0$; otherwise, to be 1.
 - (c) Set the LSB of the B color value to be 0 if $b_b = 0$; otherwise, to be 1.
 - 9.2 Color the pixel in the middle of the region with different colors as the ending signal when the digit sequence has the ending pattern.
 - (a) If the ending pattern is 2, flip the second LSB of the R color value.
 - (b) If the ending pattern is 22, flip the second LSB of the RG color

values.

- (c) If the ending pattern is 222, flip the second LSB of the RGB color values.

Step 10. Color the horizontal lines in the cover image S by (255, 255, 254) and the vertical lines by (255, 255, 255).

Step 11. Take the final S as the desired stego-image O .

Algorithm 5.3: embedding a secret message into a rectangle-based

Neo-Plasticism-like image by Method 2.

Input: a cover image S , a secret key K , a secret message M , and a partition iteration threshold I .

Output: a stego-image O into which M is embedded.

Steps.

Stage 1 --- embedding a secret message M .

Step 1. Transform the secret message M , eight bits for each character, into a bit sequence M' , and randomize the bits in M' by the secret key K .

Step 2. Perform Steps 4 and 5 of Algorithm 5.1, using the input cover image S as the input region to obtain partitioned regions R_1, R_2, \dots, R_k which are initially in a raster-scan order.

Step 3. Rearrange randomly the coloring order of the regions by the secret key K , resulting in a new coloring sequence $C_s = \{R_1', R_2', \dots, R_k'\}$.

Step 4. Use the secret key K to determine the priority of the partition direction (upper and then lower sides, or left and then right sides) of the tree.

Step 5. Compute the average RGB color values $(C_{1r}, C_{1g}, C_{1b}), (C_{2r}, C_{2g}, C_{2b}), \dots, (C_{kr}, C_{kg}, C_{kb})$ of regions R_1', R_2', \dots, R_k' , respectively.

Step 6. Embed the bit sequence M' into each region R_i' according to the coloring

sequence C_s by the following steps.

6.1 Color the region R_i' in the following way, using a coloring stack T_i according to the priority of the partition direction.

- (a) Push the region into stack T_i .
- (b) Pop the top region v in the stack.
- (c) Color the horizontal line at the middle of region v by the color $(C_{ir} + 1, C_{ig}, C_{ib})$ if the message bit is 0 and the area of region v is bigger than 2×2 .
- (d) Color the vertical line at the middle of region v by the color $(C_{ir}, C_{ig}, C_{ib} + 1)$ if the message bit is 1 and the area of region v is bigger than 2×2 .
- (e) Push two sub-regions of v into the stack according to the priority (the one with the highest priority is pushed in at last).
- (f) Repeat Steps (b) through (e) until the bit sequence of M' is exhausted.
- (g) Color the vertical line at the middle of region v by the color $(C_{ir}, C_{ig} + 1, C_{ib})$ as the ending signal if the bit sequence of M' is exhausted and the area of region v is bigger than 2×2 .

6.2 Color the remaining pixels in the region R_i' by the average color (C_{ir}, C_{ig}, C_{ib}) .

Step 7. Repeat Step 6 until the bit sequence of M' is exhausted.

Stage 2 --- dealing with intact regions.

Step 8. Perform Step 6 to color the intact regions by giving a random bit sequence by the secret key K .

Step 9. Color the horizontal lines in the cover image S by $(255, 255, 255)$ and the vertical lines by $(255, 255, 254)$.

Step 10. Take the final S as the desired stego-image O .

As seen in the above-mentioned algorithms, we use some colors (255, 255, 254) and (255, 255, 255) to record information like the vertical and horizontal partition lines, so the average color in each region will be adjusted in advance. In order to extract the right information from LSBs of each color, if one of the average RGB color values of a region is computed to be 255 or 254, which are used as special colors for various purposes in the algorithm above, the color is reset to be 253 or 252 by flipping the second LSB before data hiding. The resulting slight color shifting in the generated stego-image causes nearly no visual effect for human beings.

5.3.3 Proposed Secret Message Extraction Process

In the proposed secret message extraction process, first we have to determine which data hiding method the stego-image was created by. By extracting the color of the horizontal and vertical lines in the image, we can obtain the data hiding method used in this image. For different data hiding methods, we use different ways to recover the hiding sequence in the stego-image. By reconstructing the partition tree, we acquire the information of the hiding sequence. Accordingly, we can retrieve the secret message embedded in the stego-image. The algorithm of secret data extraction is described in detail as follows.

Algorithm 5.4: *extracting a secret message from a stego-image.*

Input: a stego-image S , and a secret key K identical to that used in Algorithms 5.2 or 5.3.

Output: the secret message M supposedly embedded in S .

Steps.

Stage 1 --- retrieving the information of the stego-image S .

- Step 1. Scan the partition lines in the vertical and horizontal direction; if the color in the horizontal line is (255, 255, 254), the hiding method is Method 1, H_1 ; otherwise, the hiding method is Method 2, H_2 .
- Step 2. Acquire the priority of the partition direction (upper and then lower sides, or left and then right sides) of the tree by using the secret key K .
- Step 3. Create an *empty* bit sequence Q initially.
- Step 4. Perform Step 5 to extract the secret message M if the hiding method is H_1 ; otherwise, perform Steps 6 through 8.

Stage 2 --- extracting the embedded secret message M .

- Step 5. Extract the secret message M from S by reconstructing the hiding sequence R in the following way.
- 5.1 Find the first partition line l_0 by scanning the vertical and horizontal directions.
- 5.2 Set up a recovery stack A to obtain the hiding sequence R .
- 5.3 Push the first partition line l_0 into the stack A .
- 5.4 Pop up the top value v in the stack A .
- 5.5 Push the sub-regions of two sides of l_0 into the stack according to the priority (the one with the highest priority is pushed in at last).
- 5.6 Find the partition line l_i of the top value v in the stack A and pop up the top value v in the stack A .
- 5.7 Push the sub-regions of two sides of l_i into the stack according to the priority (the one with the highest priority is pushed in at last).
- 5.8 Put a region i which does not contain any partition line into the hiding sequence R .
- 5.9 Extract three bits Q_r , Q_g , and Q_b from the LSBs of the RGB color

values of the region i .

5.10 Repeat Steps 5.6 through 5.9 until the B color value of the middle pixel in the region is different from the average B color of the region.

5.11 Store the three bits Q_{ir} , Q_{ig} , and Q_{ib} into Q in order.

Step 6. Perform the region growing scheme in a raster-scan order to segment out regions, R_1, R_2, \dots, R_k , in S , and obtain the information about the boundaries $(X_{\min 1}, X_{\max 1}, Y_{\min 1}, Y_{\max 1}), (X_{\min 2}, X_{\max 2}, Y_{\min 2}, Y_{\max 2}), \dots, (X_{\min k}, X_{\max k}, Y_{\min k}, Y_{\max k})$.

Step 7. Retrieve the coloring order of the regions by the secret key K , denoted as the coloring sequence $C_s = \{R'_1, R'_2, \dots, R'_k\}$, and change the corresponding orders of the boundaries $(X_{\min 1}', X_{\max 1}', Y_{\min 1}', Y_{\max 1}'), (X_{\min 2}', X_{\max 2}', Y_{\min 2}', Y_{\max 2}'), \dots, (X_{\min k}', X_{\max k}', Y_{\min k}', Y_{\max k}')$.

Step 8. Extract the secret message M from S according to the coloring sequence C_s by the following steps.

8.1 Find the first partition line l_0 by scanning the vertical and horizontal directions in a region R'_i .

8.2 Extract a bit Q_i in Q to be 0 if the first partition line l_0 is horizontal in direction; otherwise, set Q_i to be 1.

8.3 Obtain the colors C_v and C_h of the vertical and horizontal lines in region R'_i .

8.4 Set up a recovery stack A to obtain the hiding sequence R .

8.5 Push the first partition line l_0 into the stack A .

8.6 Pop up the top region v in the stack A .

8.7 Push the sub-regions of two sides of l_0 into the stack if the area of sub-region is larger than 2×2 according to the priority (the one with the highest priority is pushed in at last).

- 8.8 Find the partition line l_i of the top value v in the stack A and pop up the top value v in the stack A .
- 8.9 Extract a bit Q_i in Q to be 0 if the partition line l_i is horizontal direction; otherwise, set Q_i to be 1.
- 8.10 Push the sub-regions of two sides of l_i into the stack if the area of sub-region is larger than 2×2 according to the priority (the one with the highest priority is pushed in at last).
- 8.11 Repeat Steps 8.8 through 8.10 until the R color value of a vertical line in the region is different from the R color of C_v .

Step 9. Use the secret key K to reorder Q .

Step 10. Transform every eight values of Q into a decimal number, resulting in a new sequence Q' , and then transform Q' into characters as the desired secret message M .

5.3.4 Security Consideration

Users can extract an embedded secret message by the proposed extraction process described by Algorithm 5.4, as mentioned previously. To ensure the security of the proposed technique, in addition to using a secret key to encrypt the secret message, we also use it to randomize the building priority (upper and then lower sides, or left and then right sides) of the partition tree. Besides, for the intact regions in the second data hiding method, we also imitate the hiding process to embed random sequences in them, and malicious users may be misled to guess the secret message erroneously. With the above-mentioned schemes for security enhancement, we can strongly protect the secret messages using the proposed data hiding methods and reduce the risk for the message to be stolen by malicious users.

5.3.5 Experimental Results

Some experimental results of applying the proposed method for covert communication via a Neo-Plasticism-like image are shown in Figures 5.8 through 5.19. Figures 5.8 through 5.13 show the results by using the first data hiding method, and Figures 5.14 through 5.19 were yielded by the second method. Figure 5.8(a), Figure 5.11(a), Figure 5.14(a), and Figure 5.17(a) are the source images, which are also cover images for data hiding. Figure 5.8(b), Figure 5.11(b), Figure 5.14(b), and Figure 5.17(b) are the generated Neo-Plasticism-like images without an embedded secret message. By using Figure 5.8(a) and a secret key “life,” we can generate a stego-image into which a secret message “Enjoy your own life without comparing it with that of another.” is embedded, as shown by Figure 5.8(c). Figure 5.11(c) is a stego-image into which a secret message “Life is not an exact science, it is an art.” has been embedded with the secret key “flower.” From these images, we can see that the partition lines in the stego-image are different from those in the image without secret message embedding. By limiting the order of the partitioning lines, we can embed the secret data into the image.

Similarly, Figure 5.14(c) is a stego-image into which a secret message “Everybody is a moon, and has a dark side which he never shows to anybody.” has been embedded with the secret key “moon”. Figure 5.17(c) is a stego-image into which a secret message “No beauty is like the beauty of mind.” has been embedded with the secret key “woman”. As seen in the images, the resulting color difference between the Neo-Plasticism-like image and the stego-image is small, and the partitioning result is the same.

As shown in Figures 5.9, 5.12, 5.15, and 5.18, the secret message may be retrieved only if the right key is used in the secret message extraction process. If we

use a wrong secret key in the process, the extracted secret message will fail, like the cases seen in Figures 5.10, 5.13, 5.16 and 5.19.

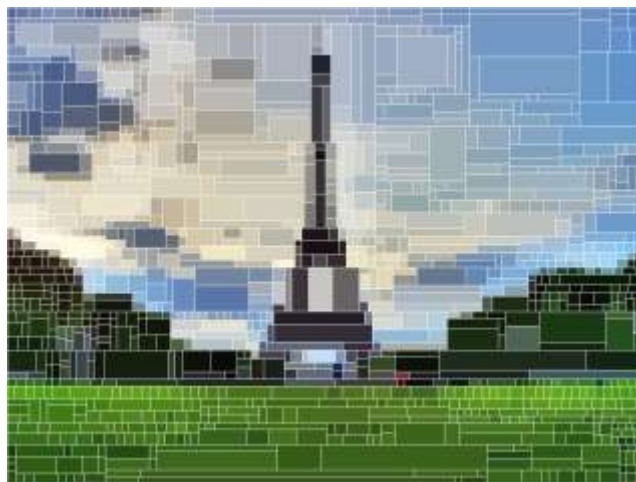
5.4 Summary

In this study, a type of computer art, called rectangle-based Neo-Plasticism-like image, is proposed. To create it, we propose an automatic method which applies binary-space partitioning to a source image by finding the maximum mutual information of the spatial positions and the image intensities according to Rigau, et al. [25]. After partitioning the image into multiple rectangles by a number of partition iterations and coloring the rectangles with their average colors, a rectangle-based Neo-Plasticism-like image is created, which appears to have the abstract style of Neo-Plasticism by using only straight horizontal and vertical lines and rectangular shapes in the resulting image.

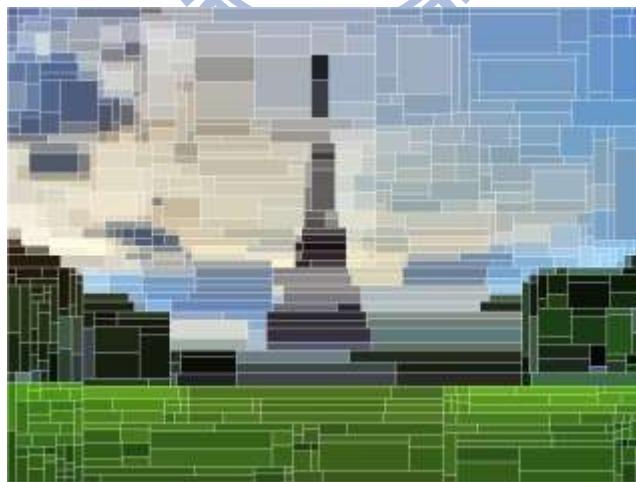
Furthermore, we have proposed two data hiding techniques for covert communication by utilizing the characteristic of the Neo-Plasticism-like image creation process. The first proposed method builds a partition tree according to a certain limit on the tree generation order. The other method is like the first one. Method 2 builds a coloring tree to color the regions of the source image. By coloring the regions in the vertical or horizontal directions, we do not limit the partition order in the source image. Besides, we use a secret key to randomize the secret message bit sequence and the priority of the partition directions of the partition tree to enhance the security of data hiding. Because the resulting Neo-Plasticism-like image is like an abstract art image as well, users cannot notice the existence of the hidden data. Accordingly, the proposed information hiding methods are feasible for covert communication.



(a)



(b)



(c)

Figure 5.8 An experimental result by using the first data hiding method with partition iteration as 10. (a) A source image (cover image). (b) A Neo-Plasticism-like image without secret message embedding. (c) A stego-image of (a) by embedding the secret message “Enjoy your own life without comparing it with that of another.” with the secret key “life.”

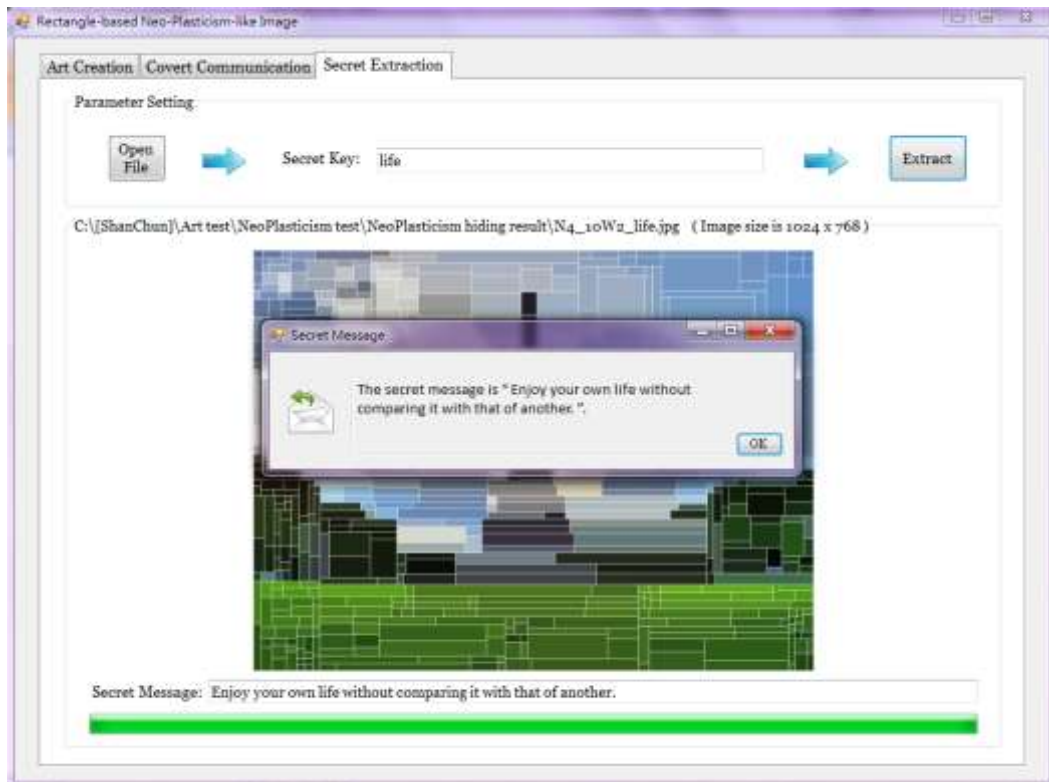


Figure 5.9 Extracting the secret message with the right secret key “life.”

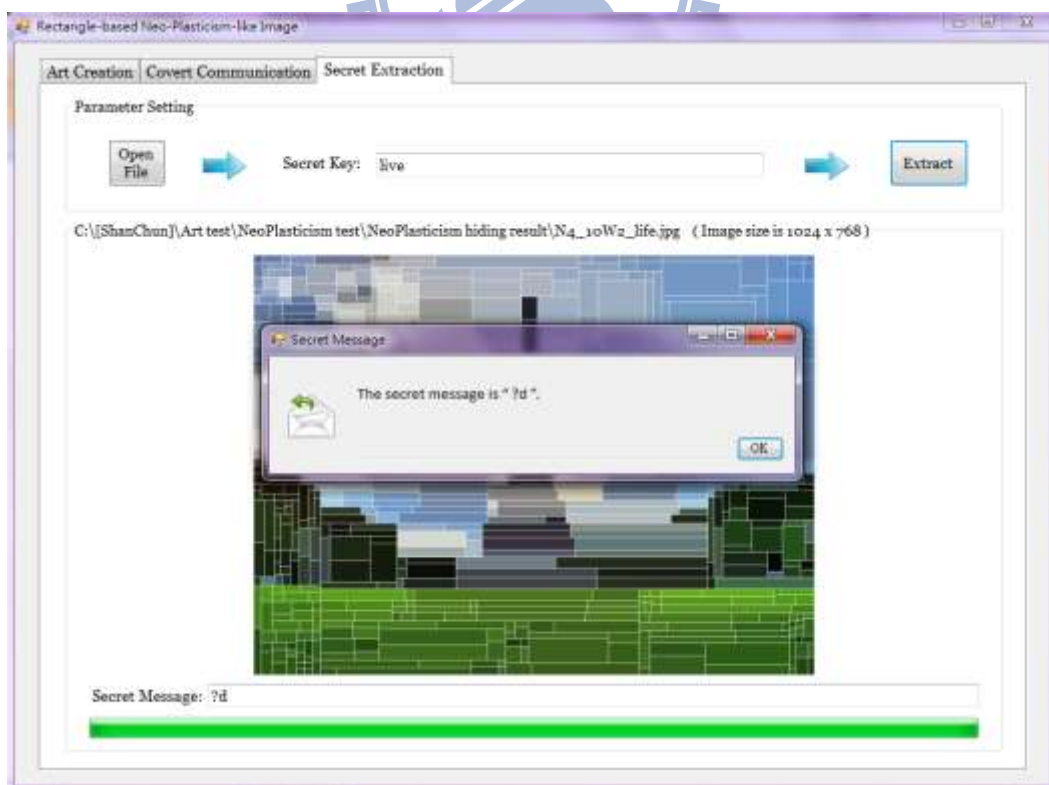


Figure 5.10 Extracted erroneous secret message with a wrong key “live.”



(a)



(b)



(c)

Figure 5.11 An experimental result by using the first data hiding method with partition iteration as 15. (a) A source image (cover image). (b) A Neo-Plasticism-like image without secret message embedding. (c) A stego-image of (a) by embedding the secret message “Life is not an exact science, it is an art.” with the secret key “flower.”

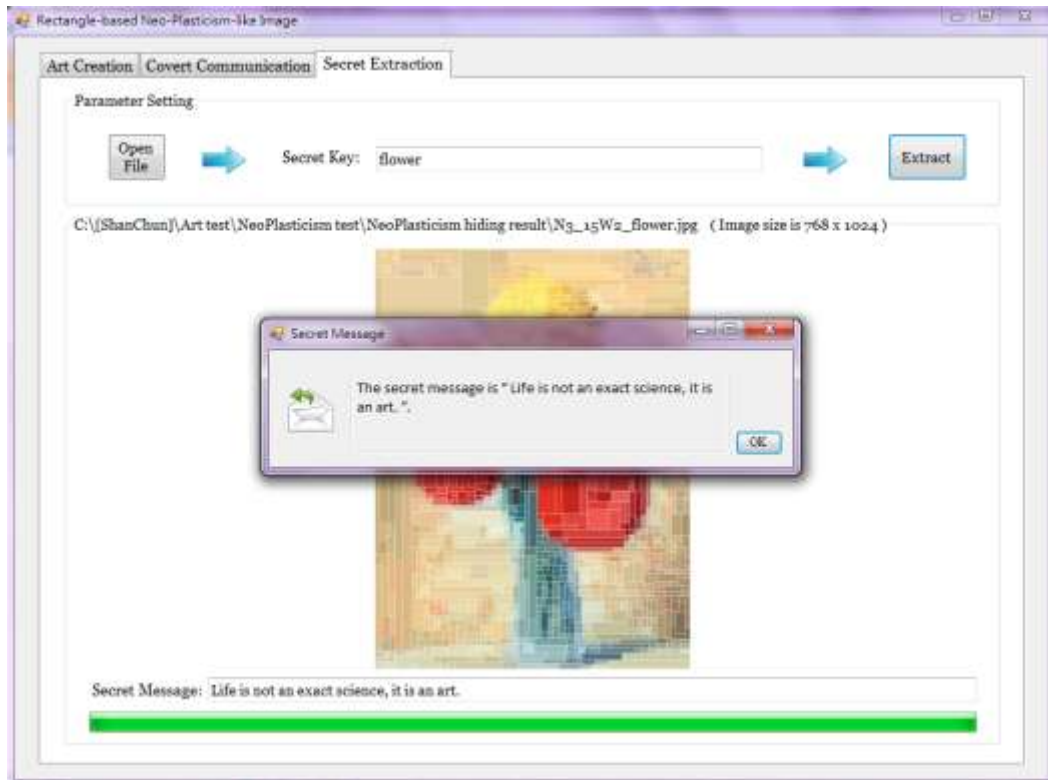


Figure 5.12 Extracting the secret message with the right secret key “flower.”

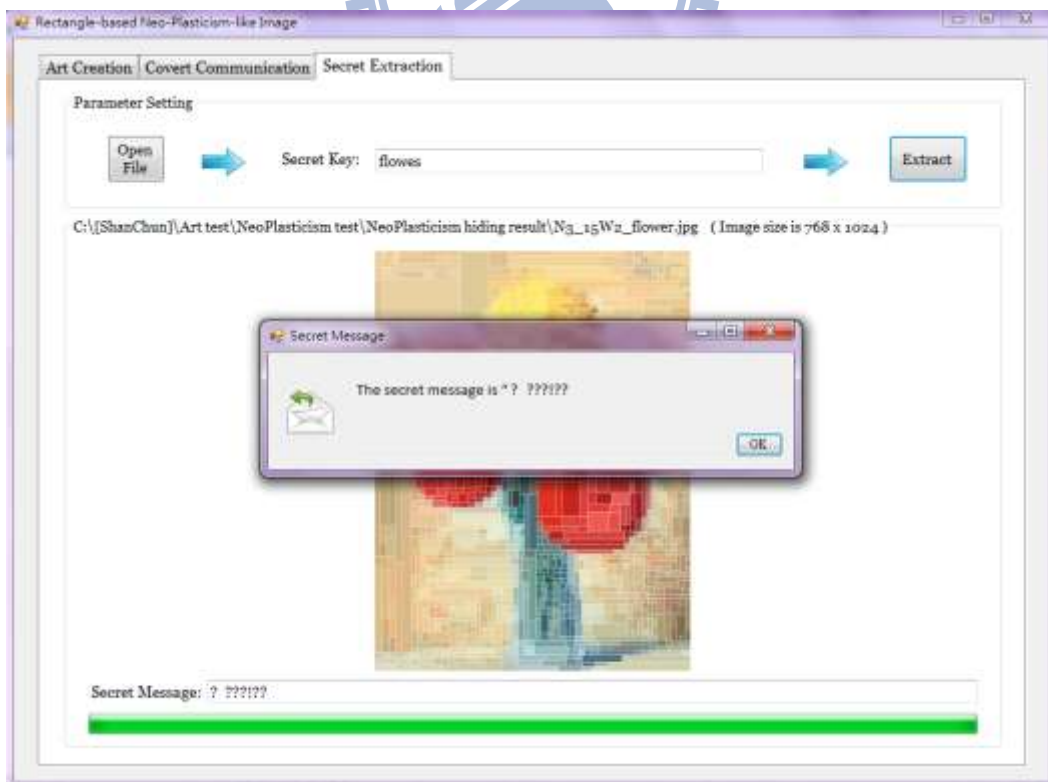
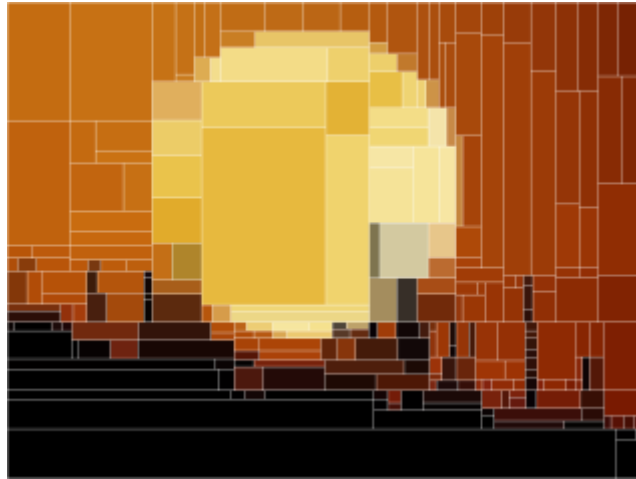


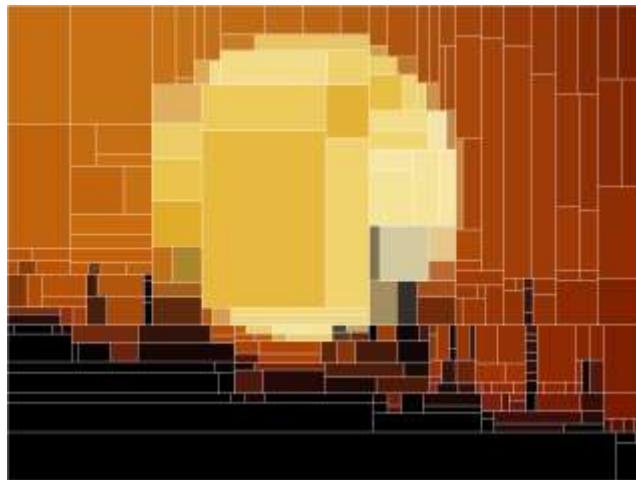
Figure 5.13 Extracted erroneous secret message with a wrong key “flowes.”



(a)



(b)



(c)

Figure 5.14 An experimental result by using the second data hiding method with partition iteration as 8. (a) A source image (cover image). (b) A Neo-Plasticism-like image without secret message embedding. (c) A stego-image of (a) by embedding the secret message “Everybody is a moon, and has a dark side which he never shows to anybody.” with the secret key “moon.”

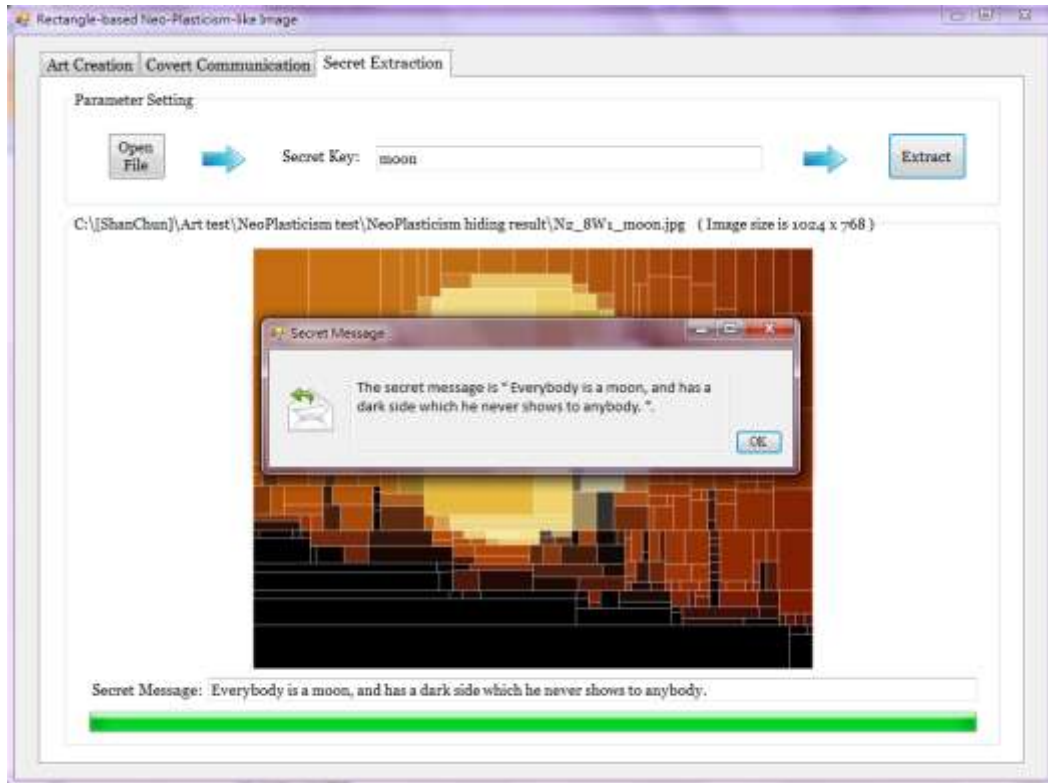


Figure 5.15 Extracting the secret message with the right secret key “moon.”

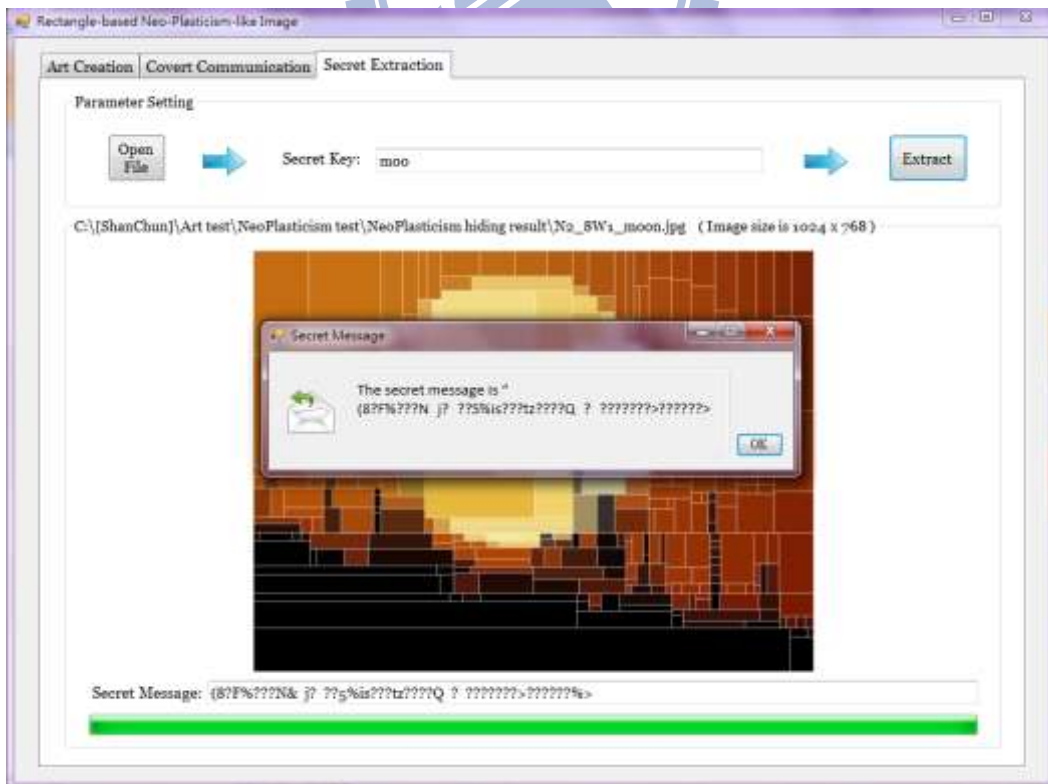


Figure 5.16 Extracted erroneous secret message with a wrong key “moo.”



Figure 5.17 An experimental result by using the second data hiding method with partition iteration as 10. (a) A source image (cover image). (b) A Neo-Plasticism-like image without secret message embedding. (c) A stego-image of (a) by embedding the secret message “No beauty is like the beauty of mind.” with the secret key “woman.”

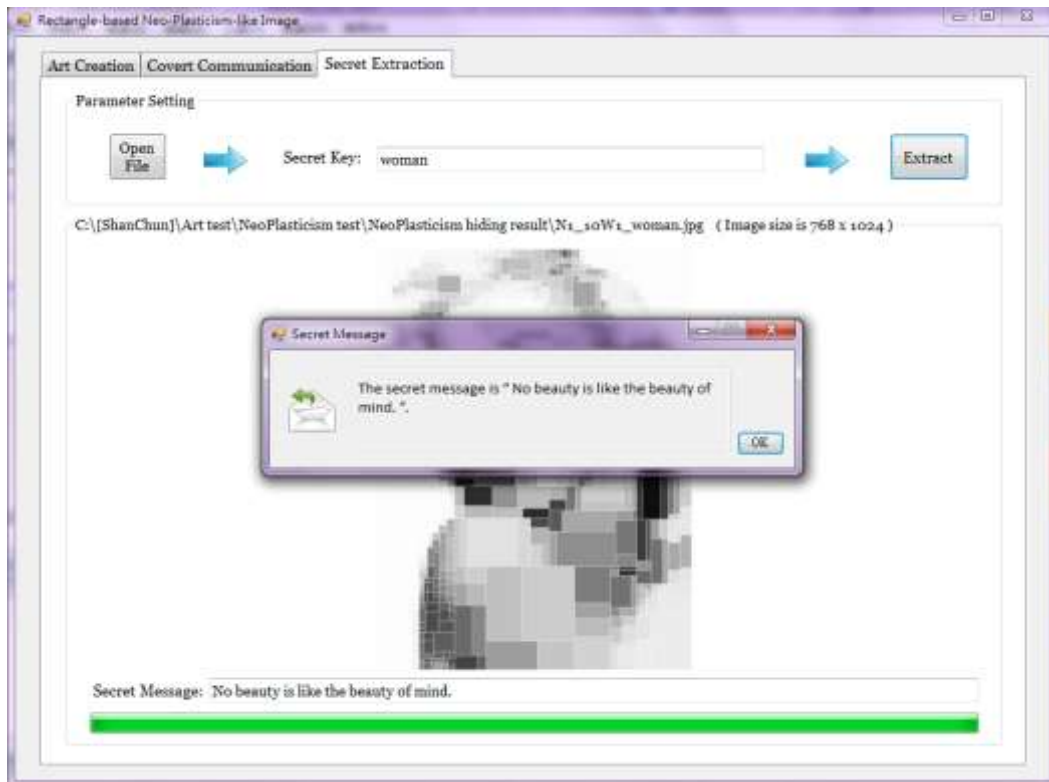


Figure 5.18 Extracting the secret message with the right secret key “woman.”

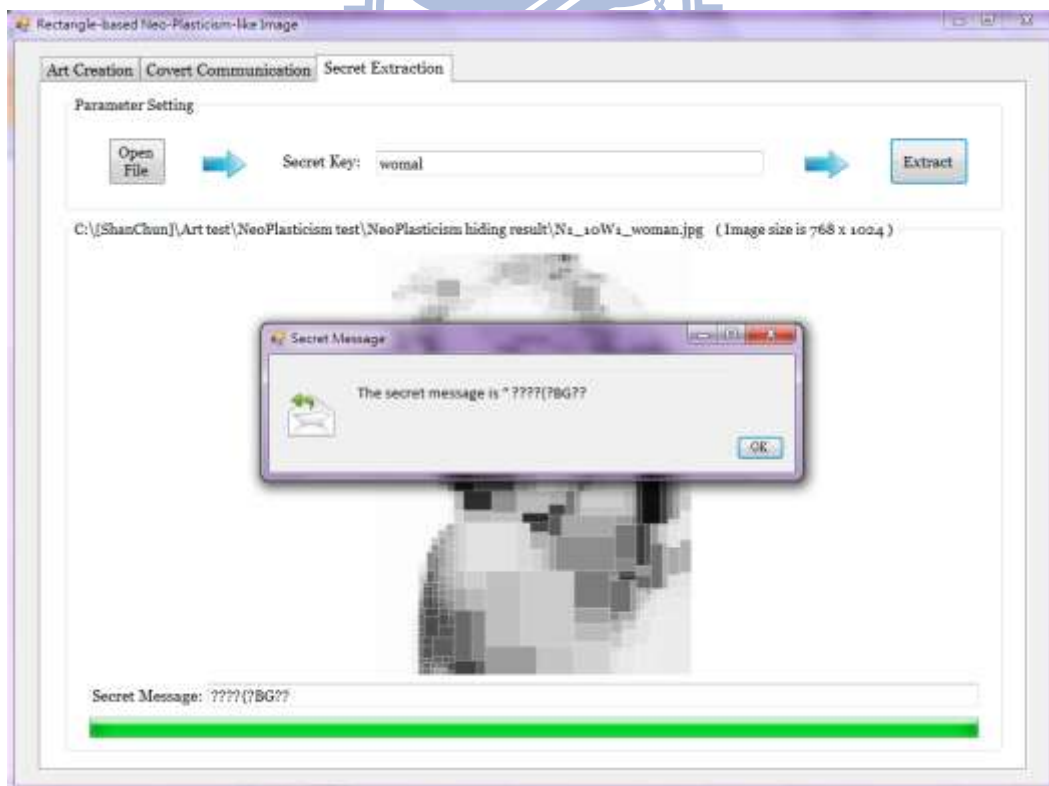


Figure 5.19 Extracted erroneous secret message with a wrong key “womal.”

Chapter 6

Conclusions and Suggestions for Future Works

6.1 Conclusions

In this study, we have proposed three different kinds of line-based computer art, named line-based Cubism-like image, strip-based Futurism-like image, and rectangle-based Neo-Plasticism-like image. Also proposed were three data hiding techniques for covert communication for these types of art image, respectively. Therefore, users can easily and simultaneously generate art images and embed secret messages in them.

Different from traditional image data hiding techniques, we hide secret messages in the individual features of art images. For the line-based Cubism-like image creation process, first, we find longer line segments in the source image by the Hough transform. Then, we connect the line segments and extend them to the image boundaries. In the region re-coloring process, we compute the new color of each pixel to keep the average color of the region unchanged, and re-color the pixel invisibly. In this way of invisible reversible pixel re-coloring, a Cubism-like image with the embedded secret message may be created.

In the creation of strip-based Futurism-like images, we utilize a region merging scheme for image segmentation. Then, we extract some region characteristics such as the corner point and the region direction by analyzing the chain codes of the region boundary. By adjusting the region edge with the corner points of each region, we

acquire the effect of polygon approximation in each region. Finally, we partition each region into strips in accordance with the extracted region direction, and hide a given secret message by coloring the sub-region with the white color or the average color.

A rectangle-based Neo-Plasticism-like image is an image created by imitating the style of the Neo-Plasticism style via a computer. Based on an algorithm of binary space partitioning, a given image can be partitioned into multiple rectangles by finding the maximum of the mutual information (MI) which is a measure about the intensities and the spatial positions of the rectangle. For the Neo-Plasticism-like image, we propose two methods to hide the secret messages. One is to limit the partitioning direction order, and we build a partition tree with each leaf node of the tree being a rectangular region. Accordingly, we can embed the secret data into the LSBs of the RGB color values of the rectangular regions. The other method does not change the partition direction order, and fill the regions with horizontal or vertical color lines to embed the message into the regions.

Furthermore, because the feeling of art is different to everyone, for each art image, we use some thresholds for users to select in order to create their favorite art image in this study. To ensure the security of the proposed technique, we have also proposed some methods to reduce the risk for the message to be stolen by attackers or hackers.

6.2 Suggestions for Future Works

In this study, we have proposed some methods for creation of line-based Cubism-like images, strip-based Futurism-like images, and rectangle-based Neo-Plasticism-like images, as well as the data hiding techniques for these three types of art images. However, there are still several interesting topics worth further study as

listed in the following.

For line-based Cubism-like images:

1. Adjusting the data hiding method to solve the wrap-around problems, but keep the average color unchanged.
2. Using some criterion to select the favorite art image for people via a computer automatically.
3. Adding different line widths to increase the diversity of an art image.

For strip-based Futurism-like images:

1. Enlarging the data capacity of the proposed data hiding method.
2. Enhancing the performance speed of the Futurism-like image creation process.
3. Producing Futurism-like images with combination of different strip directions.

For rectangle-based Neo-Plasticism-like images:

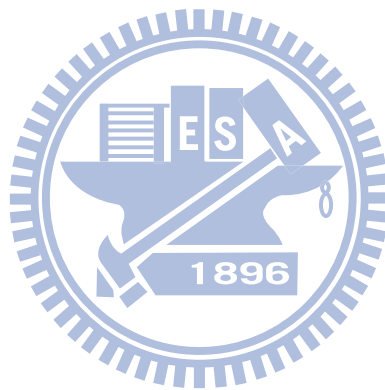
1. Enhancing the performance speed of the Neo-Plasticism-like image partition process.
2. Combining two proposed data hiding methods and applying them to enlarge the embedding capacity.

For all art images:

1. Finding more characteristics of Cubism, Futurism, or Neo-Plasticism to create art images more like their styles.
2. Investigating other types of art image creation methods and searching for specific image features for information hiding.
3. Developing visible watermarking techniques based on three art types proposed in this study.
4. Extending the proposed methods to survive some attacks such as scaling, rotation,

print-and-scan, etc.

5. Allowing the system to choose the art image style automatically according to the secret message length.



References

- [1] A. Hertzmann, "A survey of stroke-based rendering," *IEEE Computer Graphics and Applications*, vol. 23, no. 4, July-Aug. 2003, pp. 70-81.
- [2] A. Hertzmann, "Painterly rendering with curved brush strokes of multiple sizes," *Proceedings of 1998 International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 1998)*, Orlando, Florida, USA, July 1998, pp. 453-460.
- [3] A. Hertzmann, "Fast paint texture," *Proceedings of 2002 International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2002)*, Annecy, France, June 3-5, 2002, pp. 91-96.
- [4] M. P. Salisbury, M. T. Wong, J. F. Hughes, and D. H. Salesin, "Orientable textures for image-based pen-and-ink illustration," *Proceedings of 1997 International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 1997)*, Los Angeles, California, USA, 1997, pp. 401-406.
- [5] D. Mould, "Stipple placement using distance in a weighted graph," *Proceedings of International Symposium on Computational Aesthetics in Graphics, Visualization and Imaging*, Banff, Alberta, Canada, 2007, pp. 45-52.
- [6] D. Mould, "A stained glass image filter," *Proceedings of 14th Eurographics Workshop on Rendering*, Leuven, Belgium, 2003, pp. 20-25.
- [7] A. Hausner, "Simulating decorative mosaics," *Proceedings of 2001 International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 01)*, Los Angeles, California, USA, August 2001, pp. 573-580.
- [8] P. Haeberli, "Paint by numbers: abstract image representations," *Proceedings of 1990 International Conference on Computer Graphics and Interactive*

- Techniques (SIGGRAPH 1990)*, Dallas, Texas, USA, 1990, pp. 207-214.
- [9] Y. Z. Song, P. L. Rosin, P. M. Hall, and J. Collomosse, "Arty shapes," *Proceedings of the Computational Aesthetics in Graphics, Visualization and Imaging*, Lisbon, Portugal, 2008, pp. 65-72.
- [10] C. K. Chan and L. M. Cheng, "Hiding data in images by simple LSB substitution," *Pattern Recognition*, vol. 37, March 2004, pp. 469-474.
- [11] D. C. Wu and W. H. Tsai, "Embedding of any type of data in images based on a human visual model and multiple-based number conversion," *Pattern Recognition Letters*, vol. 20, August 1999, pp. 1511-1517.
- [12] 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*, Taipei, Taiwan, August 2004, pp. 207-213.
- [13] Z. Ni, Y. Q. Shi, N. Ansari, and W. Su, "Reversible data hiding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 3, March 2006, pp. 354-362.
- [14] G. Xuan, J. Zhu, J. Chan, Y. Q. Shi, Z. Ni, and W. Su, "Distortionless data hiding based on integer wavelet transform," *IEEE Electronics Letters*, vol. 38, no. 25, December 2002, pp. 1646-1648.
- [15] 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 2004 International Computer Symposium (ICS 2004)*, Taipei, Taiwan, December 15-17, 2004, pp. 449-454.
- [16] C. C. Wang and W. H. Tsai, "Creation of tile-overlapping mosaic images for information hiding," *Proceedings of 2007 National Computer Symposium*,

- Taichung, Taiwan, December 20- 21, 2007, pp. 119-126.
- [17] I. J. Lai and W. H. Tsai, "Secret-fragment-visible mosaic image - a new computer art and its application to information hiding," *IEEE Transactions on Information Forensics and Security*, accepted and to appear.
- [18] 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 13-15, 2006.
- [19] 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*, Hong Kong, June 2005, pp. 129-132.
- [20] C. P. Chang and W. H. Tsai, "Creation of a new type of art image—tetromino-based mosaic image—and protection of its copyright by losslessly-removable visible watermarking", *Proceedings of 2009 National Computer Symposium*, Taipei, Taiwan, November 27-28, 2009, pp. 577-586.
- [21] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, 1986, pp. 679-698.
- [22] R. C. Gonzalez and R. E. Woods, *Digital image processing*. 2nd ed., Prentice Hall, Upper saddle river, New Jersey, 2002.
- [23] H. Freeman, "On the encoding of arbitrary geometric configuration," *IRE Transactions on Electronic Computers*, vol. 10, no. 2, 1961, pp. 260-268.
- [24] H. Sanchez-Cruz, "A proposal method for corner detection with an orthogonal three-direction chain code," *Lecture Notes in Computer Science (ACIVS 2006)*, vol. 4179, Berlin, Germany, 2006, pp. 161-172.
- [25] J. Rigau, M. Feixas, and M. Sbert, "An information theoretic framework for image segmentation," *Proceedings of IEEE International Conference on Image*

