

國立交通大學

資訊學院

資訊科學與工程研究所

博士論文

「富含訊息多媒體」——一種普及溝通
之新工具

**Message-rich Multimedia – A New Tool for
Pervasive Communication**

研究生：李雅琳

指導教授：蔡文祥博士

中華民國 一百零二年 十一月

「富含訊息多媒體」 - 一種普及溝通
之新工具

**Message-rich Multimedia – A New Tool for
Pervasive Communication**

研 究 生：李 雅 琳

Student: Ya-Lin Lee

指 導 教 授：蔡 文 祥 博 士

Advisor: Dr. Wen-Hsiang Tsai

國立交通大學資訊學院
資訊科學與工程研究所
博士論文

A Dissertation 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 Doctor of Philosophy
in Computer Science and Engineering

November 2013
Hsinchu, Taiwan, 300
Republic of China

中華民國 一 百 零 二 年 十 一 月

「富含訊息多媒體」 - 一種普及溝通之新工具

研究生：李雅琳

指導教授：蔡文祥博士

國立交通大學資訊學院

資訊科學與工程研究所

摘要

隨著資訊科技的進步，越來越多的裝置被設計出來跟周遭的環境互動，以做各種普及溝通之應用。普及溝通是指人們可以和生活周遭的物體於任何地點與時間做資訊交換；而存在於生活周遭的許多物體可用來容納資訊，達到普及溝通的目的。本論文定義「富含訊息多媒體」，並探討以這種多媒體做普及溝通的方法。

此外，資訊隱藏可將訊息嵌入多媒體中，因此資訊隱藏為達到普及溝通的其一重要技術。然而，目前大部分的數位裝置，如智慧型手機與平板，並不能感知周遭環境的內容，意即他們不能瞭解周遭環境所含有的資訊，因此需要嶄新之資訊隱藏技術，以應用這些裝置和多種不同的富含訊息多媒體做溝通，達到普及溝通之目的。本論文提出了五種富含訊息的多媒體，包含：大張影像、加密影像、文字形式之協作文件，以及特別設計之兩種影像的硬刻版本，並提出對應於此五種富含訊息多媒體的新資訊隱藏方法。

首先，本論文提出一種可將一秘密影像隱藏於任一同樣大小之目標影像之中的資訊隱藏方法。本方法利用色彩轉換技術，建立一「可視秘密碎片馬賽克影像」，該影像看起來類似使用者挑選的目標影像，可將秘密影像隱藏起來成為一馬賽克影像，此種馬賽克影像不僅可有傳輸秘密影像之功能，更解決了傳統資訊隱藏方法無法於影像中嵌入大量資訊的問題。接著，針對加密影像，本論文利用兩次影像加密及空間相關性之比對技術來隱藏資訊，對每一區塊畫素的幾個LSBs 做加密達到嵌入一位元資訊的作用。此方法解決了先前已發表兩個技術會遭遇到的平滑影像問題。

除了影像以外，本論文亦提出一可於協作平台隱藏資訊的新方法。此方法能根據秘密訊息產生擬造的修訂歷史，並將秘密訊息隱藏於模擬過程之中。能達

此作用，主要是利用多人協作的幾個特性來隱藏資訊，包含：每個修訂版本的作者、被更改文字序列的數量、被更改文字序列的內涵、取代被更改文字序列的文字序列。此外，還利用一下載的 XML 格式之英文版維基百科來建置模擬多人協作之資料庫。此一所提方法提供了一種利用協作平台進行秘密通訊與安全保存秘密訊息之應用。

最後，本研究另外提出了可從事「自動識別與資料抓取」之兩種資訊隱藏方法，讓普及溝通實現於特別設計之兩種影像的硬刻版本上。此兩種影像之第一種為富含訊息之字元影像，第二種為富含訊息之編碼影像，這些影像可列印於紙上或顯示於螢幕或電視上。詳言之，富含訊息之字元影像其建立方式是先將字元訊息切碎，再產生跟目標影像一樣大的字元影像，接著利用一區塊亮度調變方法來改變字元影像每塊碎片的亮度值，最後將調變之位元影像注入於目標影像之中。而富含訊息之編碼影像其建立方式是先將一訊息轉換為一由二元圖樣區塊所組成的圖樣影像，並同樣透過區塊亮度調變方法，改變每塊圖樣區塊的亮度值，最後將調變之圖樣影像注入於目標影像之中。這兩種影像擁有類似條碼及 QR 碼的功能，且其外觀看起來類似一任一選擇的目標影像，故可達到普及溝通的效果。

以上所提出的方法皆為創新之作，深入的理論分析及實驗結果顯示這些方法皆具有可行性及實用性。

Message-rich Multimedia – A New Tool for Pervasive Communication

Student: Ya-Lin Lee

Advisor: Dr. Wen-Hsiang Tsai

Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

Abstract

With the advance of information technologies, more and more devices are designed to interact with environments for various pervasive communication applications; thereby people can exchange information with the identities existing in the environment everywhere and anytime. Many kinds of identities exist in the environment can be utilized to accommodate information for the purpose of pervasive communication. In this study, identities called message-rich multimedia are defined and investigated for pervasive computing.

Data hiding can be employed to embed message information into multimedia existing in various application environments, creating message-rich multimedia as the result; therefore, data hiding is regarded as one of the key techniques to achieve pervasive communication. In addition, most existing digital devices like smart phones and tablets are “unaware” of the environment context, i.e., they cannot “understand” the environmental surrounds even if they can “see” them by image taking with the built-in cameras. Therefore, numerous possibilities for achieving pervasive computing through uses of these devices by data hiding techniques via various message-rich multimedia are worth investigation.

In this dissertation study, five types of message-rich multimedia are proposed, including: 1) image with large data volumes; 2) encrypted image; 3) text-typed collaborative writing work; and 4) hard copies of two types of specially-designed images; and five new data hiding techniques creating respectively these types of message-rich multimedia are designed.

Firstly, a new large-volume data hiding method for hiding a secret image into any target image of the same size is proposed. The method creates automatically from an arbitrarily-selected target image a so-called secret-fragment-visible mosaic image as a disguise of the given secret image. Based on color transformation, the method not only creates mosaic images useful for secure image communication, but also provides a new way to solve the difficulty of hiding secret images with huge data volumes into target images. Next, via encrypted images, a new data hiding method based on the techniques of double image encryption and spatial correlation comparison is proposed as well, solving a problem encountered by two previously-proposed methods when dealing with flat cover images. Specifically, the proposed method encrypts the LSBs of each block pixel in a given encrypted image to embed a message bit.

In addition to dealing with images, a new data hiding method via collaboratively-written articles with camouflaged revision history records for use on collaborative writing platforms is proposed. Characteristics of article revisions are identified subtly and used to embed secret messages, including the author of each revision, the number of corrected word sequences, the content of the corrected word sequences, and the word sequences replacing the corrected ones. An English Wikipedia XML dump is utilized to construct a database for forging the revisions by data hiding techniques. The proposed method is useful for covert communication or secure keeping of secret messages via collaborative writing platforms.

Finally, two other data hiding techniques for automatic identification and data capture applications are proposed to enable pervasive communication via hard copies of two types of specially-created images, where the first type is message-rich character image and the second message-rich code image. These image copies may be printed versions on papers or displayed versions on monitors or TVs. Specifically, a digital message-rich character image is created from an arbitrarily-selected target image for use as a carrier of a given message by fragmenting the shapes of the composing characters of the message and injecting the resulting character fragments randomly into the target image by a block luminance modulation scheme. And a message-rich code image is created by converting a given message into a pattern image composed of binary pattern blocks and injecting the resulting pattern image into the target image by a block luminance modulation scheme. With functions similar to those of barcodes or QR codes, the created two types of message-rich images look similar to the target image, achieving the effect of pervasive communication.

The feasibility and effectiveness of all the proposed methods are demonstrated by theoretical analyses and good experimental results.



Acknowledgements

The author is in hearty appreciation of the continuous guidance, discussions, and support from her advisor, Dr. Wen-Hsiang Tsai, not only in the development of this dissertation study, but also in every aspect of her personal growth. The author would also like to acknowledge the very helpful comments and suggestions from the members of the oral defense committee as well as those from the reviewers for parts of this dissertation that were submitted for journal publication.

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 helps during her dissertation study. The author would like to acknowledge as well the financial support received from the National Science Council during the course of this dissertation study.

Finally, the author also extends her profound thanks to her dear family and boy friend for their lasting love, care, and encouragement. This dissertation is dedicated to them.

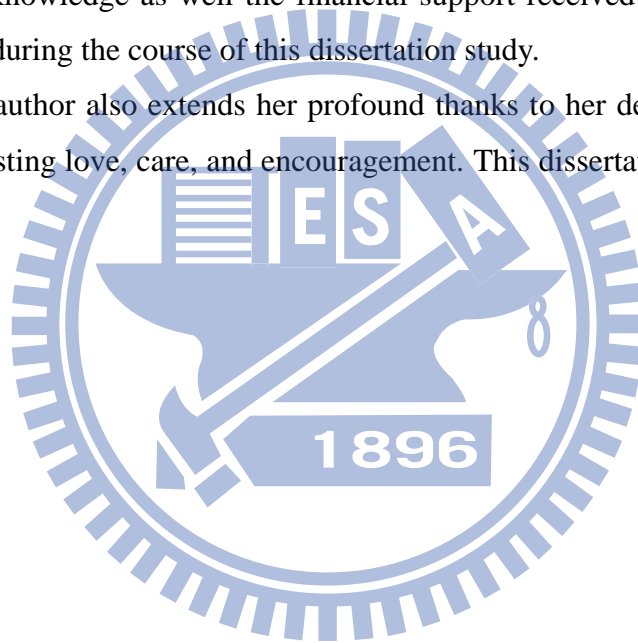


Table of Contents

摘要.....	i
Abstract.....	iii
Acknowledgements	vi
Table of Contents	vii
List of Figures.....	x
List of Tables	xvii
Chapter 1 Introduction.....	1
1.1 Background and Motivation.....	1
1.2 Issues in Study of Message-rich Multimedia.....	2
1.3 Survey of Related Works	3
1.3.1 Review of techniques for data hiding via images.....	3
1.3.2 Review of techniques for data hiding via image barcodes	4
1.3.3 Review of techniques for data hiding via text documents.....	5
1.3.4 Review of techniques for barcode reading	6
1.4 Overview of Proposed Techniques and Ideas	6
1.4.1 Data hiding by nearly-reversible color transformation.....	6
1.4.2 Data hiding by techniques of image encryption and spatial correlation comparison	7
1.4.3 Data hiding via revision history records on collaborative writing platforms	8
1.4.4 Data hiding via message-rich character images.....	8
1.4.5 Data hiding via message-rich code images.....	9
1.5 Dissertation Organization.....	11
Chapter 2 A New Data Hiding Technique via Secret-fragment-visible Mosaic Images by Nearly-reversible Color Transformation	12
2.1 Introduction.....	12
2.2 Idea of Proposed Method	14
2.3 Problems and Proposed Solutions for Mosaic Image Creation.....	15
2.4 Algorithms of Proposed Method.....	21
2.5 Experimental Results	25
2.6 Security Considerations	30

2.7 Summary	33
Chapter 3 A New Data Hiding Technique via Encrypted Images by Image Encryptions and Spatial Correlation Comparisons	34
3.1 Introduction	34
3.2 Review of Existing Methods	35
3.3 Proposed Method	39
3.3.1 Message embedding	39
3.3.2 Message extraction and image recovery	39
3.4 Experimental Results	42
3.5 Summary	46
Chapter 4 A New Data Hiding Technique via Revision History Records on Collaborative Writing Platforms	47
4.1 Introduction	47
4.2 Basic Idea of Proposed Method	49
4.3 Data Hiding via Revision History	52
4.3.1 Collaborative writing database construction	52
4.3.2 Secret message embedding	54
4.3.3 Secret message extraction	67
4.4 Experimental Results	69
4.5 Security Consideration	79
4.5.1 Camouflage	79
4.5.2 Randomness	80
4.5.3 Possible extensions for the proposed method using natural language processing methods	82
4.6 Summary	82
Chapter 5 A New Data Hiding Technique via Message-rich Character Image for Automatic Identification and Data Capture Applications	84
5.1 Introduction	84
5.2 Idea of Proposed Method	86
5.3 Generation of Message-rich Character Image	87
5.3.1 Message image creation	87
5.3.2 Block luminance modulation	87
5.3.3 Algorithm for message-rich character image creation	92
5.4 Message Extraction	93

5.4.1	Message-rich character image localization and inverse perspective transform	93
5.4.2	Block number identification and block segmentation	94
5.4.3	Binarization and optical character recognition	95
5.4.4	Message extraction algorithm	96
5.5	Experimental Results	97
5.6	Summary	100
Chapter 6 A New Data Hiding Technique via Message-rich code Image for		
Automatic Identification and Data Capture Applications..... 102		
6.1	Introduction	102
6.2	Idea of Proposed Method	103
6.3	Generation of Message-rich Code Image	104
6.3.1	Pattern image creation	104
6.3.2	Block luminance modulation	107
6.3.3	Algorithm for message-rich code image creation	108
6.4	Message Extraction	110
6.4.1	Localization of message-rich code image and inverse perspective transform	110
6.4.2	Block number identification and block segmentation	110
6.4.3	Binarization and recognition of pattern blocks	111
6.4.4	Message extraction algorithm	114
6.5	Experimental Results	116
6.6	Summary	124
Chapter 7 Conclusions and Suggestions for Future Studies..... 126		
7.1	Conclusions	126
7.2	Suggestions for Future Studies	129
References..... 131		
Vitae 138		
List of Publications of Ya-Lin Lee..... 139		

List of Figures

Figure 1.1. A result yielded by proposed method. (a) Secret image. (b) Target image. (c) Secret-fragment-visible mosaic image created from (a) and (b).	7
Figure 1.2. Example of created message-rich character image. (a) Target image. (b) Created message-rich character image.....	9
Figure 1.3. Example of created message-rich code image. (a) Target image. (b) Created message-rich code image.....	10
Figure 2.1. Illustration of creation of secret-fragment-visible mosaic image proposed in [39]......	13
Figure 2.2. Flow diagram of the proposed method.	15
Figure 2.3. Illustration of fitting tile images into target blocks.	17
Figure 2.4. Illustration of effect of rotating tile images before fitting them into target blocks. (a) Secret image. (b) Target image. (c) Mosaic image created from (a) and (b) without block rotations (with RMSE = 23.261). (d) Mosaic image created from (a) and (b) with block rotations (with RMSE = 20.870).	18
Figure 2.5. An experimental result of mosaic image creation. (a) Secret image. (b) Target image. (c) Mosaic image created with tile image size 8×8. (d) Recovered secret image using a correct key with RMSE = 0.948 with respect to secret image (a). (e) Recovered secret image using a wrong key. (f)-(i) Mosaic images created with different tile image sizes 16×16, 24×24, 32×32, and 40×40.....	26
Figure 2.6. Comparison of results of Lai and Tsai [39] and proposed method. (a) Secret image. (b) Target image. (c) Mosaic image created from (a) and (b) by [39] with RMSE = 47.651. (d) Mosaic image created from (a) and (b) by proposed method with RMSE = 33.935.....	27
Figure 2.7. Two other experimental results of mosaic image creation. (a) and (d) Secret images. (b) and (e) Target images. (c) and (f) Mosaic images created from (a) and (b), and (d) and (e), respectively, with tile size 8×8. (g) and (h) Zoom-out images of red square regions of (c) and (f), respectively.....	28
Figure 2.8. Created mosaic images with the same secret image. (a) Secret image. (b) Mosaic image created from (a) and Figure 2.7(b) with RMSE = 26.067. (c) Mosaic image created from (a) and Figure 2.7(e) with RMSE = 33.102.	29

Figure 2.9. Created mosaic images with the same secret image shown in Fig. 5(a) and small-sized target images. (a) Created image for target image shown in Fig. 5(b) with size 768×1024 . (b) Created image for target image shown in Fig. 5(b) but with size reduced to $(1/5) \times (1/5)$. (c) Created image for target image shown in Fig. 5(b) but with size reduced to $(1/10) \times (1/10)$30

Figure 2.10. Plots of trends of various parameters versus different tile image sizes (8×8 , 16×16 , 32×32) with input secret images shown previously and coming from a large dataset. (a) RMSE values of created mosaic images with respect to target images. (b) Numbers of required bits embedded for recovering secret images. (c) RMSE values of recovered secret images with respect to original ones. (d) MSSIM values of created mosaic images with respect to target images.31

Figure 2.11. Correct permutations of tile images in the mosaic image without recovering the original color characteristics. (a) The correct permutation of tile images of Figure 1.1(c). (b) The correct permutation of tile images of Figure 2.7(c).33

Figure 3.1. Recovery results showing problems of [55] and [56] with block size 8×8 , incorrectly-recovered blocks marked as white, and error denoted by *err*. (a) Input flat X-ray image. (b) Result with *err* = 50.81% yielded by [55]. (c) Result with *err* = 44.53% yielded by [56]. (d) Result with *err* = 0% yielded by proposed method. (e) Input original image of (a). (f) Result with *err* = 16.53% yielded by [55]. (g) Result with *err* = 14.99% yielded by [56]. (h) Result with *err* = 0% yielded by proposed method.38

Figure 3.2. Illustration of block contents for computing $|H_{m,n,0'} - H_{x,y,1'}|$ for 4×4 blocks, where currently-processed adjacent block of $B_{m,n'}$ is $B_{m+1,n'}$ 40

Figure 3.3. Recovery results showing effects of using both recovered and unrecovered blocks for measuring smoothness of 8×8 blocks, with incorrectly-recovered blocks marked as white, and error rate denoted by *err*. (a) Result with *err* = 2.66% yielded by [56]. (b) Result with *err* = 0.46% yielded by proposed method without using side-match. (c) Result with *err* = 0.27% yielded by proposed method using only recovered blocks in side-match scheme. (d) Result with *err* = 0.22% yielded by proposed method using both recovered and unrecovered blocks in side-match scheme.41

Figure 3.4. Four test images of size 512×512.....	42
Figure 3.5. Comparisons of bit-extraction error rates yielded by proposed method with those yielded by [55] and [56] versus different block sizes. (a) Error rates with cover image Figure 3.4(a). (b) Error rates with cover image Fig. Figure 3.4(b). (c) Error rates with cover image Figure 3.4(c). (d) Error rates with cover image Figure 3.4(d).	43
Figure 3.6. Comparisons of execution time in message embedding required by proposed method with that required by [55] and [56] versus different block sizes.....	44
Figure 3.7. Recovery results showing effects of using different numbers N_L of LSBs for 8×8 blocks with incorrectly-recovered blocks marked as white and error rate denoted by r . (a) Cover image. (b) Decrypted image with message embedded. (c) Result with $r = 0.90\%$ yielded by proposed method for $N_L = 3$. (d) Result with $r = 0.07\%$ yielded by proposed method for $N_L = 4$. (e) Result with $r = 12.87\%$ yielded by [55] for $N_L = 3$. (f) Result with $r = 29.27\%$ yielded by [55] for $N_L = 4$. (g) Result with $r = 10.21\%$ yielded by [56] for $N_L = 3$. (h) Result with $r = 27.76\%$ yielded by [56] for $N_L = 4$	45
Figure 4.1. Basic idea of proposed method that generates a revision history of a stego-document as a camouflage for data hiding.....	48
Figure 4.2. A screenshot of the revision history of an article about computer vision on Wikipedia.....	49
Figure 4.3. A screenshot of two consecutive revisions of an article about computer vision on Wikipedia.	51
Figure 4.4. Flow diagram of the proposed method.....	51
Figure 4.5. Illustration of used terms and notations.....	52
Figure 4.6. An example of found correction pairs between D_i and D_{i-1}	53
Figure 4.7. Illustration of encoding authors of revisions for data hiding.	55
Figure 4.8. Illustration of the dependency problem. (a) Revision D_{i-1} and candidate set Q_r where the dependent word sequences are surrounded by red squares. (b) Set I that corresponds to the set Q_r for solving the dependency problem.....	57
Figure 4.9. Illustration of the selection problem. (a) Huffman codes for the word sequences and the message bits that are encountered in the selection problem.	

(b) Dividing of the word sequences into groups to solve the selection problem.	59
Figure 4.10. Illustration of the consecutiveness problem. (a) An example for illustration of the consecutiveness problem. (b) Choosing splitting points randomly to solve the consecutiveness problem.....	61
Figure 4.11. The number of entries of chosen sets with the size from 2 to 40.	71
Figure 4.12. An example of generated stego-documents on constructed Wiki site with input secret message “Art is long, life is short.” (a) Cover document. (b) Revision history (c) Stego-document. (d) Previous revision of revision of (e) with words in red being those corrected to be new words in revision of (e) in red. (e) Newest revision of created stego-document. (f) Correct secret message extracted with the right key “1234.” (g) Wrong extracted secret message with a wrong key “123.”.....	73
Figure 4.13. The embedding capacities. (a) Embedding capacities of documents with chosen sets of different sizes. (b) Embedding capacities of documents with different number of revisions.....	75
Figure 4.14. Comparison of embedding capacities yielded by Liu and Tsai [64] and proposed method using different numbers of revisions.....	76
Figure 4.15. An example to show the interoperability of the proposed method which can be applied on Chinese articles.....	77
Figure 5.1. Examples of commonly-used barcodes. (a) Code 39. (b) PDF 417. (c) QR code. (d) Data matrix code.....	85
Figure 5.2. Illustration of proposed method.....	86
Figure 5.3. Message-rich character image generation. (a) Image of character “T.” (b) Ending pattern. (c) Target image. (d) Message image. (e) Y-channel of (c). (f) Modulated message image. (g) Zoom-out of red square region in (f). (h) Resulting printed message-rich character image.	89
Figure 5.4. Modulated character-fragments resulting from uses of different contrast threshold values of δ for the difference between the two representative values r_1 and r_2 . (a) $\delta = 0$. (b) $\delta = 10$. (c) $\delta = 20$. (d) $\delta = 30$. (e) $\delta = 40$. (f) $\delta = 50$	90
Figure 5.5. Localization and correction of perspective distortion in captured message-rich character image. (a) Localized message-rich character image	

portion (enclosed by red rectangle). (b) Result of perspective distortion correction applied to red portion region in (a).	93
Figure 5.6. Message extraction. (a) Captured modulated message image I_M'' . (b) Gradient values of (a). (c) Average gradient values of pixels on candidate spitting lines for different N_s . (d) Image division result according to determined number of blocks $N_s = 16$. (e) Fragment reordering result of (d). (f) Binarization result of (e). (g) OCR result of (f). (h) Extracted message.	95
Figure 5.7. Created message-rich character images. (a)-(c) Test target images. (d)-(f) Resulting message-rich character images with $N_s = 32$ and $\delta = 40$	98
Figure 5.8. Plots of trends of results using various parameters. (a) Accuracy rates of extracted messages with different contrast threshold values δ , with #blocks $N_s = 16$. (b) RMSE values of created message-rich character images with respect to target images for different contrast threshold values of δ , with #blocks $N_s = 16$. (c) Accuracies of extracted messages with different #blocks N_s , where contrast threshold value $\delta = 40$	99
Figure 5.9. Robustness of proposed method. (a) A captured message-rich character image under defacement attack. (b) A captured message-rich character image under another defacement attack. (c) A message-rich character image captured from a monitor screen.	100
Figure 6.1. Examples of message-rich images yielded by the method in Chapter 5 and proposed method. (a) Target image. (b) Message-rich character image created by the method in Chapter 5. (c) Message-rich code image created by proposed method.	103
Figure 6.2. Illustration of major steps of two phases of proposed method.	104
Figure 6.3. An example of undistinguishable binary code patterns.	105
Figure 6.4. Performing proposed bit expansion scheme on every three message bits to yield eight binary code patterns represented by pattern blocks.	106
Figure 6.5 Message-rich code image generation. (a) Target image. (b) Pattern image I_P . (c) Y-channel of (a). (d) Modulated pattern image. (e) Zoom-out of red square region in (d). (f) Resulting message-rich code image.	107
Figure 6.6. Modulated pattern block resulting from uses of different contrast threshold values of δ for the absolute difference between the two adjusted representative	

values r_1' and r_2' . (a) $\delta = 0$. (b) $\delta = 5$. (c) $\delta = 10$. (d) $\delta = 20$. (e) $\delta = 30$. (f) $\delta = 40$ 109

Figure 6.7. Localization and correction of perspective distortion in captured message-rich code image. (a) Localized message-rich code image portion (enclosed by red rectangle). (b) Result of perspective distortion correction applied to red portion region in (a). 110

Figure 6.8. Block number identification. (a) Captured modulated pattern image I_P'' . (b) Gradient values of (a). (c) Average gradient values of pixels on candidate spitting lines for different N_s . (d) Image division result according to determined number of unit blocks, $N_s = 64$ 112

Figure 6.9. Binarization and code-pattern recognition. (a) Captured modulated pattern image. (b) Binarization result of (a). (c) Result of code-pattern recognition of (b). (d) Extracted message. 115

Figure 6.10. Created message-rich code images. (a), (c), and (e) Target images. (b), (d) and (f) Resulting message-rich code images with $N_s = 128$ and $\delta = 40$ 117

Figure 6.11. Created message-rich code images with different contrast threshold values of δ , where $N_s = 64$. (a) Resulting message-rich code image with RMSE = 66.35 and accuracy rate = 85.60%, where $\delta = 0$. (b) Resulting code image with RMSE = 66.57 and accuracy rate = 98.97%, where $\delta = 20$. (c) Resulting code images with RMSE = 68.47 and accuracy rate = 100%, where $\delta = 40$. (b) Resulting images with RMSE = 72.27 and accuracy rate = 100%, where $\delta = 60$ 118

Figure 6.12. Plots of trends of results using various parameters. (a) Accuracy rates of extracted messages with different contrast threshold values δ , with #unit blocks $N_s = 32$. (b) RMSE values of created message-rich code images with respect to target images for different contrast threshold values of δ , with #unit blocks $N_s = 32$. (c) Accuracy rates of extracted messages with different #unit blocks N_s with contrast threshold $\delta = 40$. (d) RMSE values of created message-rich code images with respect to target images with different #unit blocks N_s and contrast threshold $\delta = 40$ 119

Figure 6.13. Created message-rich code images with different #unit blocks N_s , where contrast threshold value $\delta = 40$. (a) Resulting message-rich code image with RMSE = 47.66 and accuracy rate = 100%, where $N_s = 16$. (b) Resulting

message-rich code image with RMSE = 44.63 and accuracy rate = 100%, where $N_s = 32$. (c) Resulting message-rich code image with RMSE = 42.05 and accuracy rate = 100.00%, where $N_s = 64$. (d) Resulting message-rich code image with RMSE = 39.43 and accuracy rate = 99.11%, where $N_s = 128$...

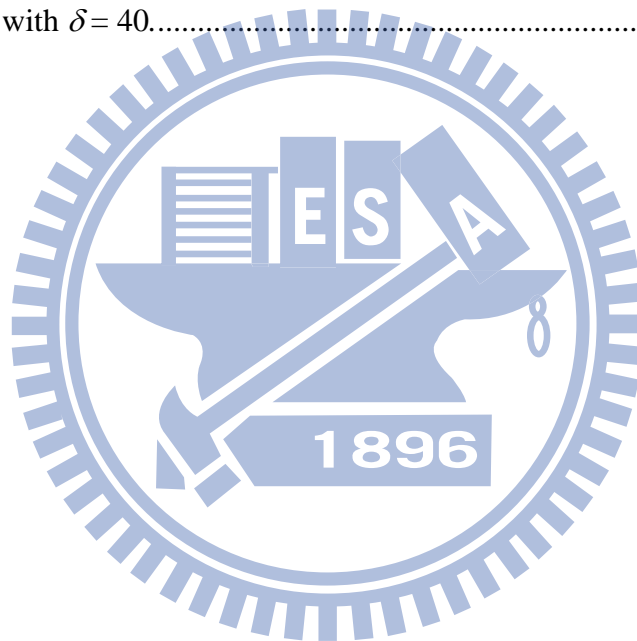
Figure 6.14. Binarized captured message-rich images created by method in Chapter 5 and proposed method in this chapter and respective message extraction accuracy rates, where the target image of these resulting images is Figure 6.10(c). (a) Binarized image by method in Chapter 5 with $N_s = 32$ and accuracy rate = 98.61%. (b) Binarized image by proposed method in this chapter with $N_s = 32$ and accuracy rate = 99.80%. (c) Binarized image by method in Chapter 5 with $N_s = 64$ and accuracy rate = 41.25%. (d) Binarized image by proposed method in this chapter with $N_s = 64$ and accuracy rate = 99.76%..... 122

Figure 6.15. Performing another bit expansion scheme on every three message bits to yield 14 binary code patterns represented by pattern blocks. 123

Figure 6.16. Results yielded by using two different bit expansion schemes with $N_s = 64$ and $\delta = 20$. (a) Pattern image yielded by the original bit expansion scheme. (b) Pattern image yielded by the new bit expansion scheme. (c) Message-rich code image yielded by the original bit expansion scheme with RMSE = 55.97. (d) Message-rich code image yielded by the new bit expansion scheme with RMSE = 55.44. 124

List of Tables

Table 4.1. Top twenty frequently used correction pairs.....	70
Table 4.2. Some correction pairs each with more than one word either in the original word sequence or in the new word sequence.....	71
Table 4.3. An example of a chosen set with the new word sequence “such as”.....	72
Table 4.4. The information of experimental documents.....	74
Table 4.5. Comparison of methods for data hiding via texts.....	78
Table 6.1. An example of code pattern recognition.....	114
Table 6.2. Comparison of results of proposed method in this chapter and method in Chapter 5 with $\delta = 40$	121



Chapter 1

Introduction

1.1 Background and Motivation

With the advance of information technologies, more and more devices are designed to interact with environments for various pervasive communication applications; thereby people can exchange information with the identities existing in the environment everywhere and anytime [1]–[2]. For example, one can use the camera on a smart phone to scan a QR code on a merchandise item and obtain the detailed related information. Recently, Davis [3] proposed a new concept, called *signal rich art*: the art that communicates its identity to context-aware devices, through *data hiding techniques* mainly, to realize pervasive communication.

In our daily life, many kinds of *identities* existing in the environment can be utilized to accommodate information for the purpose of pervasive communication. However, identities discussed in [3] are mainly those with artistic flavors, such as illustrations, posters, sculptures. It is desirable in this study to explore various types of multimedia, such as images, texts, hard copies, advertisements, displays on monitors or TVs, etc., for pervasive computing. Messages are expected to be injected into such identities, like the information encoded into the QR codes, and can be extracted by people using a “message reader.” We call such multimedia *message-rich multimedia* in this study.

Moreover, data hiding is a type of technique which can embed messages into multimedia existing in our daily life for various applications, creating message-rich multimedia which achieve the effect of pervasive communication. A lot of data hiding techniques have been developed in the past decade [4]–[5]. They may be regarded to play key roles in our study of pervasive communication via message-rich multimedia.

However, current progresses towards our new vision of technology advancement — pervasive communication by message-rich multimedia — are impeded by computers, networks, and digital devices that are largely *unaware* of the environmental context [3]; that is, they cannot “understand” the environmental surrounds even if they can “see” them by image taking with the built-in cameras. Therefore, numerous possibilities for achieving pervasive computing by data hiding

techniques via message-rich multimedia are still open research topics worth studying. It is desired in this study to solve possible issues which might be encountered in the study of message-rich multimedia.

1.2 Issues in Study of Message-rich Multimedia

About issues which might be encountered in the study of message-rich multimedia, firstly it is well known that conventional data hiding methods often face the difficulty to embed a large amount of message data into a single image [4]-[16]. Up to now, most existing methods can hide only text messages or images with small data volumes into cover images. Specifically, if one wants to hide a secret image into a cover image of the same size, the secret image must be compressed greatly in advance, resulting in the undesired effect of unrecoverability of the original higher-quality secret image. This study tries to solve this issue of transmitting images with large data volumes secretly without degrading the original quality of the secret image.

Next, an image may contain private or confidential information that is usually encrypted before being transmitted on the Internet to ensure its security. However, designs of most conventional data hiding methods are based on the properties of natural images so that they are not suitable for use in embedding messages into *encrypted* images which usually appear to be noise or random data. Hence, this dissertation study is devoted, as the second goal, to this issue of embedding messages in encrypted images.

Thirdly, attacking the weaknesses of human auditory and visual systems, most researches on data hiding focused on *non-text* multimedia as cover media. Less data hiding techniques using text-type cover media have been proposed. Recently, more and more collaborative writing platforms are becoming popular, and some of them have been exploited for data hiding applications. However, most of the data hiding methods can only be applied to documents with single authors and single revision versions [29]-[38], meaning that they are not suitable for hiding data on collaborative writing platforms. Therefore, a third goal of this study is to design new data hiding methods which can hide data into documents created on collaborative writing platforms.

Moreover, conventional data hiding methods can be employed to transfer data though “digital files” only, such as images and text documents; they are “incompetent” for enabling pervasive communication when one wants to interact with the

environmental surround. A type of data hiding, called *hardcopy data hiding*, have been proposed, which embeds information into *image barcodes* using halftone techniques [17]-[19], and the encoded information can survive “*print-and-scan attacks*.” However, if one uses a mobile device to capture images of *hardcopy image barcodes*, the information might not be decoded successfully since the captured image will suffer from additional types of distortions other than those acquired by scanning. Therefore, also as a goal of this study it is desirable to devise new *automatic identification and data capture* techniques via the use of hard copies of *message-rich images* that have functions similar to barcode or QR-code reading, with the generated hard copies of the images looking visually similar to pre-selected target images, achieving the effect of pervasive communication once again in different ways.

In summary, the goals of this dissertation study are to propose data hiding techniques to create various types of message-rich multimedia for pervasive communication, including: 1) image with large data volumes; 2) encrypted image; 3) text-typed collaborative writing work; and 4) hard copies of images. Fulfillments of aforementioned goals of this dissertation study together will be expected to enhance the state-of-art studies on data hiding techniques, yielding a new vision of pervasive communication and a further step of extending its applications.

1.3 Survey of Related Works

Works related to this study are categorized into several directions and reviewed as follows.

1.3.1 Review of techniques for data hiding via images

Data hiding is useful for applications like covert communication, copyright protection, document authentication, secret keeping, etc., and is a key component to achieve the function of pervasive communication as mentioned previously. Recently, many methods for data hiding via images have been proposed. Petitcolas [4] and Bender *et al.* [5] made good surveys of data hiding techniques via images, which may be classified into two major types: *spatial-domain based* and *transform-domain based*.

Spatial-domain based methods hide messages directly into the spatial-domain data of given images, such as LSB substitution, histogram modification, difference expansion, etc. [6]-[10]. For example, Chan and Cheng [6] proposed a simple LSB

substitution method that applies an optimal pixel adjustment process to the input image. Ni *et al.* [7] and Lee and Tsai [8] proposed histogram modification methods, each of which shifts some values in the histogram around the peak to embed secret messages. Tian [9] proposed a difference expansion method that explores data redundancy in an image to achieve a high embedding capacity. Hu *et al.* [10] proposed another difference expansion method that utilizes horizontal as well as vertical difference images for data embedding.

Transform-domain based methods hide messages into the transform-domain data of given images, using transformations like discrete cosine, integer wavelet, etc. [11]-[14]. For example, Fridrich *et al.* [11] proposed two discrete cosine transform (DCT) based methods that compress JPEG coefficients or modify quantization matrices to embed messages. Chang *et al.* [12] proposed another DCT based method that uses two successive zero coefficients of the medium frequency components in each block to hide messages. Lee *et al.* [13] proposed an integer wavelet transform based method that embeds a watermark into the high-frequency wavelet coefficients of each block. Lin *et al.* [14] proposed a data hiding method for copyright protection based on the use of the so-called significant differences of the blocks of the wavelet coefficients during the wavelet coefficient quantization process.

1.3.2 Review of techniques for data hiding via image barcodes

Another type of data hiding, which is called “*hardcopy*” *data hiding*, can embed information into so-called *image barcodes* using halftone techniques [17]-[19]. These image barcodes have the visual appearances of other images and the encoded information can be decoded from their hardcopy versions acquired by *scanners*. That is, the encoded information can survive “*print-and-scan attacks*.” For example, Bulan *et al.* [17] proposed a framework for data hiding in images printed with clustered dot halftones via a pattern orientation modulation technique. Bulan and Sharma [18] proposed another pattern orientation modulation technique that utilizes three printing channels and modulates the orientations of elliptical-shaped dots for data encoding. Damera-Venkata *et al.* [19] proposed a block-error diffusion method that embeds information into hardcopy images by using dot-shape modulation.

1.3.3 Review of techniques for data hiding via text documents

Attacking the weaknesses of human auditory and visual systems, many researches on data hiding focused on non-text cover media. Less data hiding techniques using text-type cover media have been proposed. Bennett [28] made a good survey about hiding data in text and classified related techniques into three categories: *format-based methods*, *random and statistical generation*, and *linguistic methods*.

Format-based methods use the physical formats of documents to hide messages. Some of them utilize spaces in documents to encode message data. For example, Alattar and Alattar [29] proposed a method that adjusts the distances between words or text lines using spread-spectrum and BCH error-correction techniques, and Kim *et al.* [30] proposed a word-shift algorithm that adjusts the spaces between words based on concepts of word classification and statistics of inter-word spaces. Some other methods utilize non-displayed characters to hide messages, such as Lee and Tsai [31] which encodes message bits using special ASCII codes and hides the result between the words or characters in PDF files.

Random and statistical methods generate directly camouflage texts with hidden messages to prevent the attack of comparing the camouflage text with a known plaintext. For example, Wayner [32]-[33] proposed a method for text generation based on the use of context-free grammars and tree structures. Another method available on a website [34] extends this idea to generate fake spam emails with hidden messages, which are usually ignored by people.

Linguistic methods use written natural languages to conceal secret messages. For example, Chapman *et al.* [35] proposed a synonym replacement method that generates a cover text according to a secret message using certain sentence models and a synonym dictionary. Bolshakov [36] extended the synonym replacement method by using a specific synonymy dictionary and a very large database of collocations to create a cover text, which is more believable to a human reader. Shirali-Shahreza and Shirali-Shahreza [37] proposed a third synonym replacement method that hides data in a text by substituting words which have different terms in the UK and the US. Stutsman *et al.* [38] proposed a method to hide messages in the noise that is inherent in natural language translation results without the necessity of transmitting the source text for decoding.

1.3.4 Review of techniques for barcode reading

In addition to data hiding, the use of the barcode is another technique for pervasive communication, where a barcode is usually attached to objects for various identification purposes, and represents machine-readable data by patterns of lines, rectangles, dots, etc. To extract the data encoded into barcodes, such as Code 39 [20], PDF417 [21], QR code [22], data matrix code [23], etc., several barcode reading techniques have been proposed in the past. Ouaviani *et al.* [24] proposed an image processing framework for 2D barcode reading, including four main phases: region of interest detection, code localization, code segmentation, and decoding. Zhang *et al.* [25] proposed a real-time barcode localization method by using a two-stage processing, where the barcode is found first through a region-based analysis of low-resolution images and then read and analyzed in their original resolutions. Yang *et al.* [26] proposed another accurate barcode localization method by using the prior knowledge of the barcode to obtain the initially localized corners, and then using a post-localization process to find the accurate corner locations. Yang *et al.* [27] proposed an adaptive thresholding technique for the binarization of the barcode image by constructing a dynamic search window centered at the nearest edge pixel of the pixel to be binarized.

1.4 Overview of Proposed Techniques and Ideas

In this section, we describe the main ideas and techniques of the proposed data hiding techniques via various message-rich multimedia.

1.4.1 Data hiding by nearly-reversible color transformation

A new large-volume data hiding method is proposed, which creates automatically from an arbitrarily selected target image a so-called secret-fragment-visible mosaic image as a disguise of a given secret image, achieving the effect of hiding a secret image into any target image of the same size. Specifically, after a target image is selected arbitrarily, the given secret image is first divided into rectangular fragments called *tile images*, which then are fit into similar blocks in the target image, called *target blocks*, according to a similarity criterion based on color variations. Next, the color characteristic of each tile image is transformed to be that of the corresponding target block in the target image, resulting in a mosaic image which looks like the target image. Figure 1.1 shows a result yielded by the proposed method.



Figure 1.1. A result yielded by proposed method. (a) Secret image. (b) Target image. (c) Secret-fragment-visible mosaic image created from (a) and (b).

The proposed method removes the weakness found in [39] that requires a large image database for the user to select a color-similar target image for each input secret image while keeping its merit of high-volume data embedding capability. That is, the proposed method can hide a secret image into *any pre-selected* target image of the same size to create a secret-fragment-visible mosaic image *without the need of a database*. The method not only creates mosaic images useful for secure keeping of secret images, but also provides a new way to solve the difficulty of hiding secret images with huge data volumes into target images.

1.4.2 Data hiding by techniques of image encryption and spatial correlation comparison

A new data hiding method on encrypted images based on the techniques of double image encryption and spatial correlation comparison is proposed. The proposed method solves a problem encountered in the two previously-proposed methods [55]-[56] when dealing with flat cover images. In this study, the LSBs of each block pixel in an encrypted image are *encrypted further* to embed one message bit, so the aforementioned problem encountered in [55] and [56] caused by flat cover images is solved.

Furthermore, *four* LSBs of each pixel of a block in the encrypted image are utilized for message embedding. Also, a side-match scheme that utilizes the spatial correlations of *both* recovered and unrecovered blocks is proposed to decrease the bit-extraction error rate, in contrast with [56] which utilizes the spatial correlations of

recovered blocks only. Experimental results showing the proposed method greatly improves the performance of the two previously-proposed methods in dealing with flat cover images.

1.4.3 Data hiding via revision history records on collaborative writing platforms

A new data hiding method via collaboratively-written articles with forged revision history records on collaborative writing platforms is proposed. The hidden message is camouflaged as a stego-document consisting of a stego-article and a revision history created through a simulated process of collaborative writing. The revisions are forged using a database constructed by mining the word sequences used in real cases from an English Wikipedia XML dump. Four characteristics of article revisions are identified and utilized to embed secret messages, including the author of each revision, the number of corrected word sequences, the content of the corrected word sequences, and the word sequences replacing the corrected ones. Related problems arising in utilizing these characteristics for data hiding are identified and solved skillfully, resulting in an effective multi-way method for hiding secret messages into the revision history.

To create more realistic revisions, Huffman coding based on the word sequence frequencies collected from Wikipedia is applied to encode the word sequences. Therefore, the resulting stego-document is more realistic than other text data hiding methods. The proposed method is useful for covert communication or secure keeping of secret messages on collaborative writing platforms. Moreover, to the best of our knowledge, this is the first work that can simulate the collaborative writing process with multiple authors and revisions and utilize the characteristics in the collaborative writing process effectively for message embedding.

1.4.4 Data hiding via message-rich character images

A new data hiding method via message-rich character images is proposed, where the character image is a new kind of message-rich multimedia and may be printed as a hardcopy for use in applications of pervasive communication. Figure 1.2 shows an example of the created message-rich character image yielded by the proposed method. The created image is then “re-imaged” by a mobile-phone camera and “understood”

by some *automatic identification and data capture* (AIDC) techniques [40] proposed in this study.

Specifically, a message-rich character image is created from a target image used as a carrier of a given message by fragmenting the shapes of the composing *characters* of the message and “injecting” the resulting character fragments randomly into the target image by a block luminance modulation scheme. Each message-rich character image so created has the visual appearance of the corresponding pre-selected target image.

Message-rich character images may be of the forms of documents, labels, posters, etc. Also, such images may have the visual appearances of artistic-flavored photos, pictures, paintings, which are more attractive to humans than those produced by conventional AIDC techniques, like barcodes, QR-codes, etc. Moreover, the image not only can be printed on papers but also can be displayed on various types of screens for various uses. In addition, the message-rich character image can endure more types of distortions like perspective transformation, noise, screen blurring, etc. than the existing hardcopy image barcode methods.



Figure 1.2. Example of created message-rich character image. (a) Target image. (b) Created message-rich character image.

1.4.5 Data hiding via message-rich code images

Another new data hiding method via message-rich code images is proposed, where the message-rich code image is a new kind of message-rich multimedia. It may also be printed as a hardcopy for use in applications of pervasive communication just

like the use of the message-rich character image. The proposed method improves the previous method presented in Section 1.4.4 above, as described in the following.

As shown in Figure 1.2(b), each message-rich character image contains many small character fragments with undesired visual effects. Also, it requires an optical character recognition (OCR) scheme to extract the embedded message. Moreover, to keep the resolution in the captured image sufficiently good for correct message extraction, the size of each block cannot be too small. In order to solve these problems, instead of transforming the given message to be embedded into a message image, the proposed method converts a given message into a bit stream of codes first, which is then represented by binary *pattern blocks*, each being composed of 2×2 *unit blocks*. A block luminance modulation scheme is then applied to each pattern block to yield a message-rich code image with the visual appearance of a pre-selected target image. Figure 1.3 shows an example of the created message-rich code image yielded by the proposed method.

The proposed method has the following additional merits: (1) the yielded message-rich code image has a much better visual appearance of the target image; (2) the accuracy rate of message extraction from the generated code image is higher; and (3) the message extraction speed is higher.

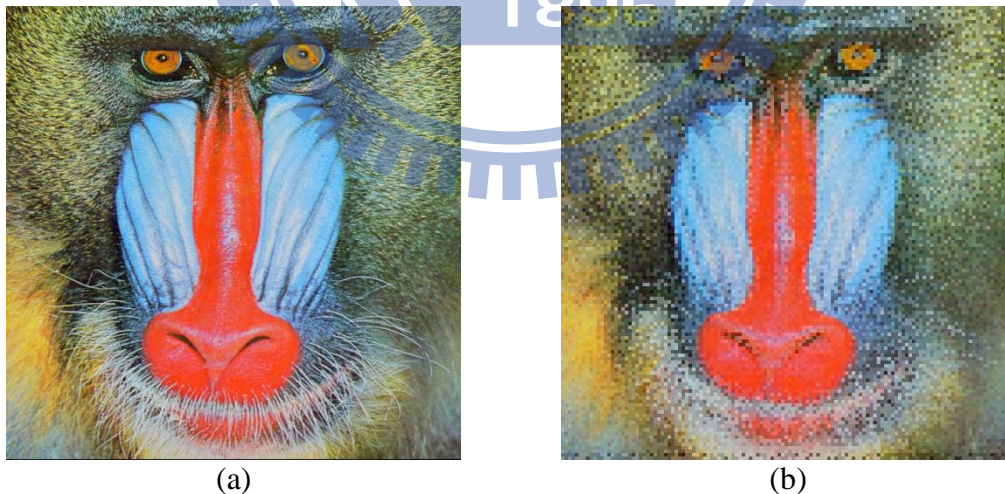


Figure 1.3. Example of created message-rich code image. (a) Target image. (b) Created message-rich code image.

1.5 Dissertation Organization

The remainder of this dissertation is organized as follows. In Chapter 2, the proposed large-volume data hiding technique for secure image transmission is described. In Chapter 3, the proposed new data hiding technique on encrypted images based on the techniques of double image encryption and spatial correlation comparison is described. The proposed new data hiding technique via creations of fake collaboratively-written documents on collaborative writing platforms is presented in Chapter 4. In Chapter 5, the proposed new data hiding technique via message-rich character images is described, while the proposed new data hiding technique via message-rich code images is described in Chapter 6. Finally, conclusions of this study and some suggestions for future researches are included in the last chapter.



Chapter 2

A New Data Hiding Technique via Secret-fragment-visible Mosaic Images by Nearly-reversible Color Transformation

2.1 Introduction

Data hiding is useful for applications like covert communication, copyright protection, document authentication, secret keeping, etc. Many methods for data hiding via images have been proposed [6]-[16]. In order to reduce the distortion of the resulting image, an upper bound for the distortion value is usually set on the payload of the cover image. A discussion on this rate-distortion issue can be found in [41]. Thus, a main issue of the methods for hiding data in images is the difficulty to embed a large amount of message data into a single image.

Specifically, if one wants to hide a secret image into a cover image with the same size, the secret image must be highly compressed in advance. For example, for a data hiding method with an embedding rate of 0.5 bits per pixel, a secret image with 8 bits per pixel must be compressed at a rate of at least 93.75% beforehand in order to be hidden into a cover image. But, for many applications, such as keeping or transmitting medical pictures, military images, legal documents, etc., that are valuable with no allowance of serious distortions, such data compression operations are usually impractical. Moreover, most image compression methods, such as JPEG compression, are not suitable for line drawings and textual graphics, where sharp contrasts between adjacent pixels are often destructed to become noticeable artifacts after being compressed [42].

Therefore, most existing methods can hide only text messages or images with small data volumes into cover images. However, in a recently published paper by Lai and Tsai [39], a new type of computer art image was presented, called secret-fragment-visible mosaic image, which is the result of rearrangement of the fragments of a secret image in disguise of another image called *target image* pre-selected from a database. The above-mentioned difficulty of hiding a huge volume of image data behind a cover image is solved *automatically* by the use of this

type of mosaic image, where a secret image of the same size is hidden in the mosaic image *without* any compression.

In more detail, as illustrated by Figure 2.1, a given secret image is first “chopped” into tiny rectangular fragments, and a target image with a similar color distribution is selected from a database. Then, the fragments are arranged by using a fast greedy algorithm to fit into the blocks of the target image, yielding an image with a mosaic appearance looking like the target image. The mosaic image preserves all the secret image fragments in appearance, but no one can figure out what the original secret image looks like due to the tiny sizes and the randomness of the re-arranged fragments. The method may be adopted as a new way for secure keeping of secret images.

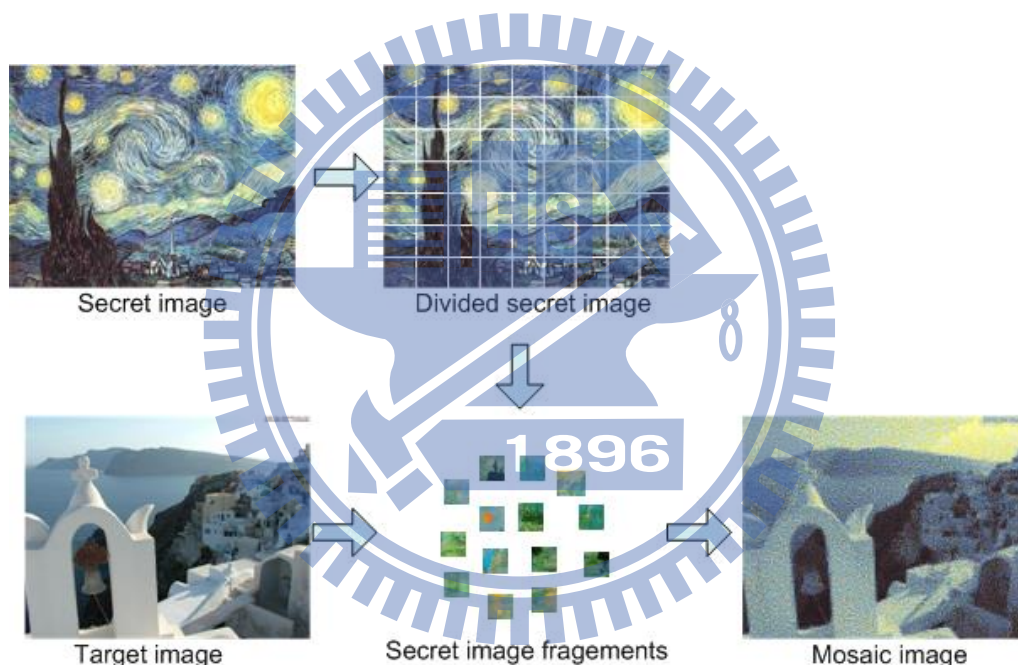


Figure 2.1. Illustration of creation of secret-fragment-visible mosaic image proposed in [39].

However, using their method, the user is *not* allowed to select freely his/her favorite image for use as the target image. It is therefore desired in this study to remove this weakness of the method while keeping its merit, that is, it is aimed to design a new method to transform a secret image into a secret-fragment-visible mosaic image of the same size that has the visual appearance of *any freely-selected* target image *without the need of a database*.

Specifically, after a target image is selected arbitrarily, the given secret image is first divided into rectangular fragments called *tile images*, which then are fit into

similar blocks in the target image, called *target blocks*, according to a similarity criterion based on color variations. Next, the color characteristic of each tile image is transformed to become that of the corresponding target block in the target image, resulting in a mosaic image which looks like the target image. Relevant schemes are also proposed to conduct *nearly lossless* recovery of the original secret image from the resulting mosaic image. The proposed method is new in the fact that it can transform a secret image into a disguising mosaic image without compression, while other data hiding methods must hide a highly compressed version of the secret image into a cover image when the secret image and the cover image have the same data volume.

2.2 Idea of Proposed Method

The proposed method includes two main phases as shown by the flow diagram of Figure 2.2: 1) mosaic image creation; and 2) secret image recovery.

In the first phase, a mosaic image is yielded, which consists of the fragments of an input secret image with color corrections according to a similarity criterion based on color variations. The phase includes four stages as described in the following.

Stage 1-1 – fit the tile images of the secret image into the target blocks of a pre-selected target image.

Stage 1-2 – transform the color characteristic of each tile image in the secret image to become that of the corresponding target block in the target image.

Stage 1-3 – rotate each tile image into a direction with the minimum RMSE value with respect to its corresponding target block.

Stage 1-4 – embed relevant information into the created mosaic image for future recovery of the secret image.

And in the second phase, the embedded information is extracted to recover nearly losslessly the secret image from the generated mosaic image. The phase includes two stages as described in the following.

Stage 2-1 – extract the embedded information for secret image recovery from the mosaic image.

Stage 2-2 – recover the secret image using the extracted information.

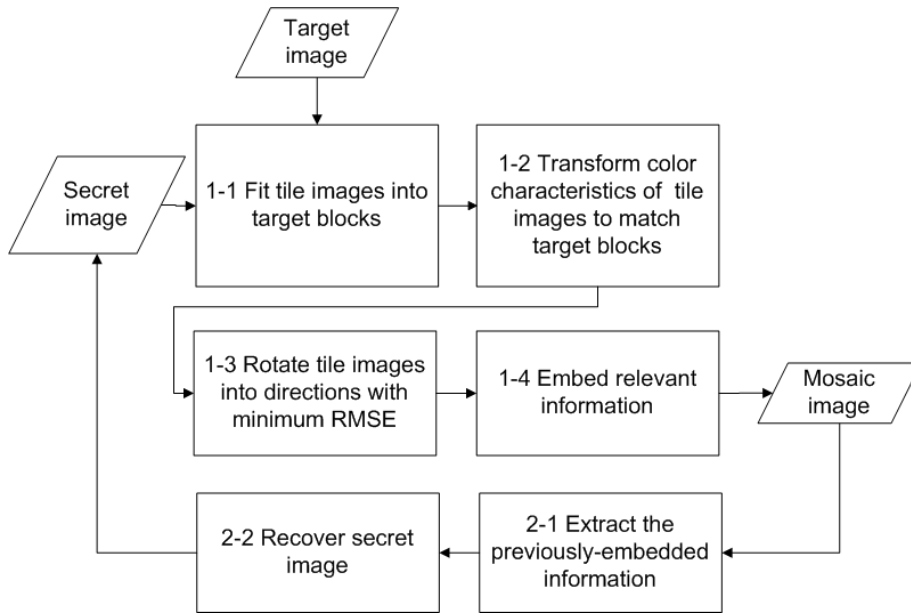


Figure 2.2. Flow diagram of the proposed method.

2.3 Problems and Proposed Solutions for Mosaic Image Creation

Problems encountered in generating mosaic images are discussed in this section with solutions to them proposed.

A. Color Transformations between Blocks

In the first phase of the proposed method, each tile image T in the given secret image is fit into a target block B in a pre-selected target image. Since the color characteristics of T and B are different from each other, how to change their color distributions to make them look alike is the main issue here. Reinhard *et al.* [43] proposed a color transfer scheme in this aspect, which converts the color characteristic of an image to be that of another in the $\alpha\beta$ color space. This idea is an answer to the issue and is adopted in this study, except that the RGB color space instead of the $\alpha\beta$ one is used to reduce the volume of the required information for recovery of the original secret image.

More specifically, let T and B be described as two pixel sets $\{p_1, p_2, \dots, p_n\}$ and $\{p'_1, p'_2, \dots, p'_n\}$, respectively. Let the color of each p_i be denoted by (r_i, g_i, b_i) and that of each p'_i by (r'_i, g'_i, b'_i) . At first, we compute the means and standard deviations of T and B , respectively, in each of the three color channels R, G, and B by the following formulas:

$$\mu_c = \frac{1}{n} \sum_{i=1}^n c_i, \quad \mu'_c = \frac{1}{n} \sum_{i=1}^n c'_i; \quad (1)$$

$$\sigma_c = \sqrt{\frac{1}{n} \sum_{i=1}^n (c_i - \mu_c)^2}, \quad \sigma'_c = \sqrt{\frac{1}{n} \sum_{i=1}^n (c'_i - \mu'_c)^2} \quad (2)$$

where c_i and c'_i denote the C-channel values of pixels p_i and p'_i , respectively, with $c = r, g, \text{ or } b$ and $C = R, G, \text{ or } B$. Next, we compute new color values (r_i'', g_i'', b_i'') for each p_i in T by:

$$c_i'' = q_c (c_i - \mu_c) + \mu'_c, \quad (3)$$

where $q_c = \sigma'_c / \sigma_c$ is the *standard deviation quotient* and $c = r, g, \text{ or } b$. It can be verified easily that the new color mean and variance of the resulting tile image T' are equal to those of B , respectively. To compute the original color values (r_i, g_i, b_i) of p_i from the new ones (r_i'', g_i'', b_i'') , we use the following formula which is the inverse of :

$$c_i = (1/q_c)(c_i'' - \mu'_c) + \mu_c. \quad (4)$$

Furthermore, we have to embed into the created mosaic image sufficient information about the new tile image T' for use in the later stage of recovering the original secret image. For this, theoretically we can use (4) to compute the original pixel value of p_i . However, the involved mean and standard deviation values in the formula are all real numbers, and it is impractical to embed real numbers, each with many digits, in the generated mosaic image. Therefore, we limit the numbers of bits used to represent relevant parameter values in (3) and (4). Specifically, for each color channel we allow each of the means of T and B to have 8 bits with its value in the range of 0 to 255, and the standard deviation quotient q_c in (3) to have 7 bits with its value in the range of 0.1 to 12.8. That is, each mean is changed to the closest value in the range of 0 to 255, and each q_c is changed to the closest value in the range of 0.1 to 12.8. We do *not* allow q_c to be 0 because otherwise the original pixel value cannot be recovered back by (4) for the reason that $1/q_c$ in (4) is not defined when $q_c = 0$.

B. Choosing Appropriate Target Blocks and Rotating Blocks to Fit Better with Smaller RMSE Value

In transforming the color characteristic of a tile image T to be that of a corresponding target block B as described above, how to choose an appropriate B for each T is an issue. For this, as shown in Figure 2.3, we use the standard deviation of the colors in the block as a measure to select the most similar B for each T . Specially, we sort all the tile images to form a sequence, S_{tile} , and all the target blocks to form another, S_{target} , according to the *average* values of the standard deviations of the three color channels. Then, we fit the first in S_{tile} into the first in S_{target} , fit the second in S_{tile} into the second in S_{target} , and so on.

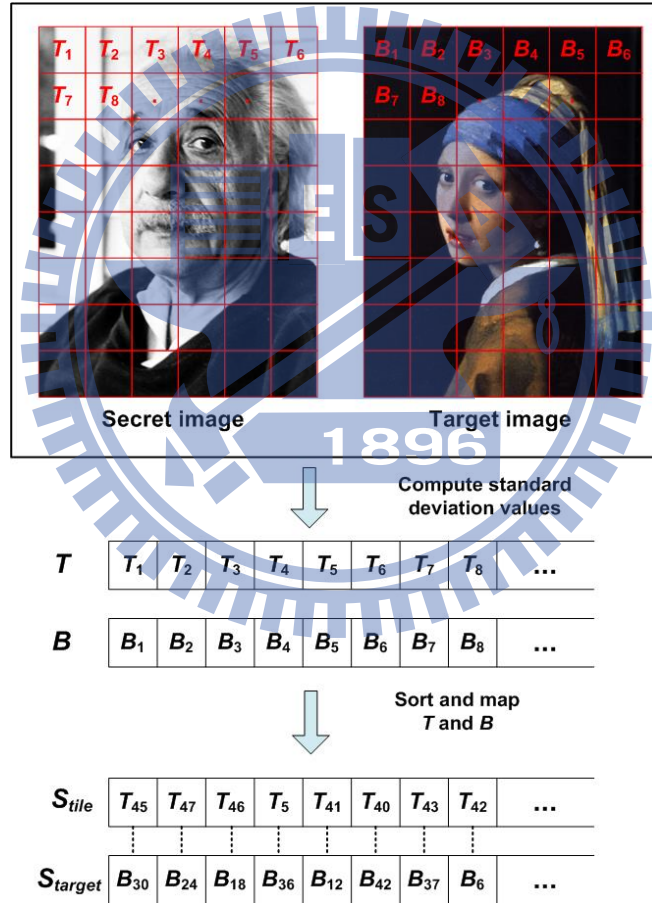


Figure 2.3. Illustration of fitting tile images into target blocks.

Additionally, after a target block B is chosen to fit a tile image T and after the color characteristic of T is transformed, we conduct a further improvement on the color similarity between the resulting tile image T' and the target block B by rotating T' into one of the four directions, 0° , 90° , 180° , and 270° , which yields a rotated

version of T' with the *minimum* root mean square error (RMSE) value with respect to B among the four directions for final use to fit T into B . Furthermore, the color similarity between the resulting tile image T' and the target block B is measured in the *luminance channel only*, instead of in the RGB three color channels, to reduce the execution time of the proposed method. Figure 2.4 shows a result of applying this block rotation scheme to the secret image and the target image shown in Figures 2.4(a) and 2.4(b), respectively, where Figure 2.4(c) is the mosaic image created without applying this scheme and Figure 2.4(d) is the one created instead. It can be seen that Figure 2.4(d) has a better fitting result with a smaller RMSE value than Figure 2.4(c).

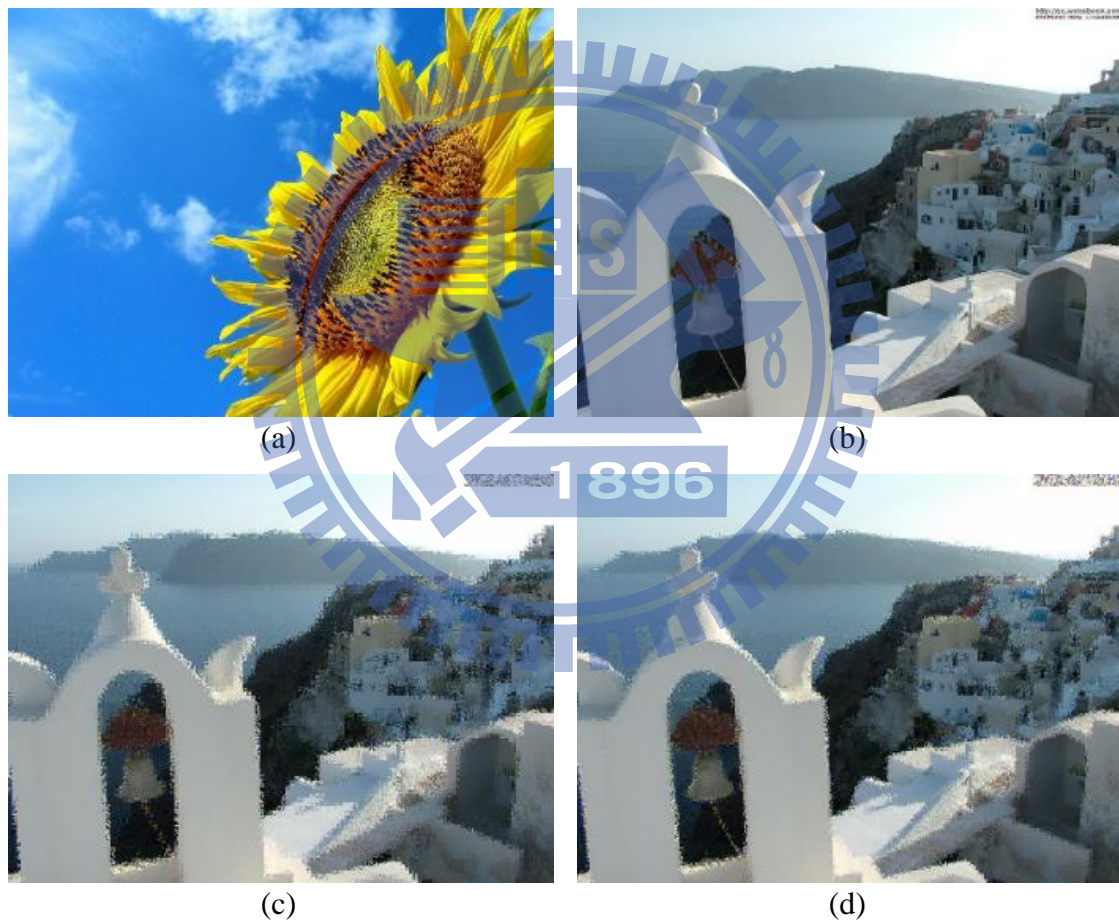


Figure 2.4. Illustration of effect of rotating tile images before fitting them into target blocks. (a) Secret image. (b) Target image. (c) Mosaic image created from (a) and (b) without block rotations (with RMSE = 23.261). (d) Mosaic image created from (a) and (b) with block rotations (with RMSE = 20.870).

C. Handling Overflows/Underflows in Color Transformation

After the color transformation process is conducted as described previously, some pixel values in the new tile image T' might have *overflows* or *underflows*. To deal with this problem, we convert such values to be non-overflow or non-underflow ones and record the value differences as *residuals* for use in later recovery. Specifically, we convert all the transformed pixel values in T' not smaller than 255 to be 255, and all those not larger than 0 to be 0. Next, we compute the differences between the original pixel values and the converted ones as the residuals and record them as part of the information associated with T' . Accordingly, the pixel values which are just on the bound of 255 or 0, however, cannot be distinguished from those with overflow/underflow values during later recovery since all the pixel values with overflows/underflows are converted to be 255 or 0 now. To remedy this, we define the residuals of those pixel values which are on the bound to be “0” and record them as well.

But as can be seen from (3), the ranges of possible residual values are unknown, and this causes a problem of deciding how many bits should be used to record a residual. To solve this problem, we record the residual values in the *un-transformed* color space rather than in the transformed one. That is, by using the following two formulas we compute first the smallest possible color value c_S (with $c = r, g, \text{ or } b$) in T that becomes larger than 255 as well as the largest possible value c_L in T that becomes smaller than 0, respectively, after the color transformation process has been conducted:

$$\begin{aligned} c_S &= \lceil (1/q_c)(255 - \mu_c') + \mu_c \rceil; \\ c_L &= \lfloor (1/q_c)(0 - \mu_c') + \mu_c \rfloor. \end{aligned} \quad (5)$$

Next, for an un-transformed value c_i which yields an overflow after the color transformation, we compute its residual as $|c_i - c_S|$; and for c_i which yields an underflow, we compute its residual as $|c_L - c_i|$. Then, the possible values of the residuals of c_i will all lie in the range of 0 to 255 as can be verified. Consequently, we can simply record each of them with 8 bits. And finally, because the residual values are *centralized around zero*, we use further in this study *the Huffman encoding scheme* to encode the residuals in order to reduce the numbers of required bits to represent them.

D. Embedding Information for Secret Image Recovery

In order to recover the secret image from the mosaic image, we have to embed relevant recovery information into the mosaic image. For this, we adopt a technique proposed by Coltuc and Chassery [44] and apply it to the least significant bits of the pixels in the created mosaic image to conduct data embedding. Unlike the classical LSB replacement methods [6], [45], [46], which substitute LSBs with message bits directly, the reversible contrast mapping method [44] applies simple integer transformations to pairs of pixel values. Specifically, the method conducts forward and backward integer transformations as follows, respectively, where (x, y) are a pair of pixel values and (x', y') are the transformed ones:

$$x' = 2x - y, \quad y' = 2y - x; \quad (6)$$

$$x = \left\lfloor \frac{2}{3}x' + \frac{1}{3}y' \right\rfloor, \quad y = \left\lfloor \frac{1}{3}x' + \frac{2}{3}y' \right\rfloor. \quad (7)$$

The method yields high data embedding capacities close to the highest bit rates and has the lowest complexity reported so far.

The information required to recover a tile image T which is mapped to a target block B includes: 1) the index of B ; 2) the optimal rotation angle of T ; 3) the truncated means of T and B and the standard deviation quotients, of all color channels; and 4) the overflow/underflow residuals. These data items for recovering a tile image T are integrated as a five-component bit stream of the form

$$M = t_1 t_2 \dots t_m r_1 r_2 m_1 m_2 \dots m_{48} q_1 q_2 \dots q_{21} d_1 d_2 \dots d_k,$$

where the bit segments $t_1 t_2 \dots t_m$, $r_1 r_2$, $m_1 m_2 \dots m_{48}$, $q_1 q_2 \dots q_{21}$, and $d_1 d_2 \dots d_k$ represent the values of the index of B , the rotation angle of T , the means of T and B , the standard deviation quotients, and the residuals, respectively.

In more detail, the numbers of required bits for the five data items in M are discussed below: 1) the index of B needs m bits to represent, with m computed by: $m = \lceil \log[(W_S \times H_S) / N_T] \rceil$, where W_S and H_S are respectively the width and height of the secret image S , and N_T is the size of the target image T ; 2) it needs two bits to represent the rotation angle of T because there are four possible rotation directions; 3) 48 bits are required to represent the means of T and B because we use eight bits to represent a mean value in each color channel; 4) it needs 21 bits to represent the quotients of T over B in the three color channels with each channel requiring 7 bits;

and 5) the total number k of required bits for representing all the residuals depends on the number of overflows or underflows in T' .

Then, the above-defined bit streams of all the tile images are concatenated in order further into a *total bit stream* M_t for the entire secret image. Moreover, in order to protect M_t from being attacked, we encrypt it with a secret key to obtain an encrypted bit stream M_t' , which is finally embedded into the pixel pairs in the mosaic image using the method of Coltuc and Chassery [44] described above. It may require more than one iteration in the encoding process since the length of M_t' may be larger than the number of pixel pairs available in an iteration. A plot of the statistics of the numbers of required bits for secret image recovery is shown in Figure 2.10(b).

Moreover, we have to embed as well some related information about the mosaic image generation process into the mosaic image for use in the secret image recovery process. Such information, described as a bit stream I like M mentioned previously, includes the following data items: 1) the number of iterations conducted in the process for embedding the bit stream M_t' ; 2) the total number of used pixel pairs in the last iteration for embedding M_t' ; and 3) the Huffman table for encoding the residuals.

With the bit stream M_t' embedded into the mosaic image, we can recover the secret image back as will be described later. It is noted that some *loss* will be incurred in the recovered secret image, or more specifically, in the color transformation process using (3) where each pixel's color value c_i is multiplied by the standard deviation quotient q_c and the resulting real value c_i'' is truncated to be an integer in the range of 0 through 255. However, because each truncated part is smaller than the value of 1, the recovered value of c_i using (4) is still precise enough to yield a color nearly identical to its original one. Even when overflows/underflows occur at some pixels in the color transformation process, we record their residual values as described previously and after using (4) to recover the pixel value c_i , we add the residual values back to the computed pixel values c_i to get the original pixel data, yielding a nearly losslessly-recovered secret image. According to the results of the experiments conducted in this study, each recovered secret image has a very small RMSE value with respect to the original secret image, as will be shown later in Section 2.5.

2.4 Algorithms of Proposed Method

Based on the above discussions, the detailed algorithms for mosaic image creation and secret image recovery may now be described as follows.

Algorithm 2.1. Mosaic image creation.

Input: a secret image S , a pre-selected target image T , and a secret key K .

Output: a secret-fragment-visible mosaic image F .

Steps:

Stage 1 — fitting the tile images into the target blocks.

- Step 1. If the size of the target image T is different from that of the secret image S , change the size of T to be identical to that of S ; and divide the secret image S into n tile images $\{T_1, T_2, \dots, T_n\}$ as well as the target image T into n target blocks $\{B_1, B_2, \dots, B_n\}$ with each T_i or B_i being of size N_T .
- Step 2. Compute the means and the standard deviations of each tile image T_i and each target block B_j for the three color channels according to (1) and (2); and compute accordingly the average standard deviations for T_i and B_j , respectively, where $i = 1$ through n and $j = 1$ through n .
- Step 3. Sort the tile images in the set $S_{tile} = \{T_1, T_2, \dots, T_n\}$ and the target blocks in the set $S_{target} = \{B_1, B_2, \dots, B_n\}$ according to the computed average standard deviation values of the blocks; map in order the blocks in the sorted S_{tile} to those in the sorted S_{target} in a 1-to-1 manner; and reorder the mappings according to the indices of the tile images, resulting in a *mapping sequence* L of the form: $T_1 \rightarrow B_{j1}, T_2 \rightarrow B_{j2}, \dots, T_n \rightarrow B_{jn}$.
- Step 4. Create a mosaic image F by fitting the tile images into the corresponding target blocks according to L .

Stage 2 — performing color conversions between the tile images and the target blocks.

- Step 5. Create a *counting table* TB with 256 entries, each with an index corresponding to a residual value, and assign an initial value of zero to each entry (note that each residual value will be in the range of 0 to 255).
- Step 6. For each mapping $T_i \rightarrow B_{j_i}$ in sequence L , represent the means μ_c and $\mu_{c'}$ of T_i and B_{j_i} , respectively, by eight bits; and represent the standard deviation quotient q_c appearing in (3) by seven bits, according to the scheme described in Section 2.3(A) where $c = r, g, \text{ or } b$.
- Step 7. For each pixel p_i in each tile image T_i of mosaic image F with color value c_i where $c = r, g, \text{ or } b$, transform c_i into a new value c_i'' by (3); if c_i'' is not

smaller than 255 or if it is not larger than 0, then change c_i'' to be 255 or 0, respectively; compute a residual value R_i for pixel p_i by the way described in Section 2.3(C); and increment by 1 the count in the entry in the counting table TB whose index is identical to R_i .

Stage 3 – rotating the tile images.

Step 8. Compute the RMSE values of each color-transformed tile image T_i in F with respect to its corresponding target block B_j after rotating T_i into each of the directions $\theta = 0^\circ, 90^\circ, 180^\circ$ and 270° in the luminance channel; and rotate T_i into the *optimal* direction θ_0 with the smallest RMSE value.

Stage 4 – embedding the secret image recovery information.

Step 9. Construct a Huffman table HT using the content of the counting table TB to encode all the residual values computed previously.

Step 10. For each tile image T_i in mosaic image F , construct a bit stream M_i for recovering T_i in the way as described in Section 2.3(D), including the bit-segments which encode the data items of: 1) the index of the corresponding target block B_j ; 2) the optimal rotation angle θ_0 of T_i ; 3) the means of T_i and B_j and the related standard deviation quotients of all three color channels; and 4) the bit sequence for overflows/underflows with residuals in T_i encoded by the Huffman table HT constructed in Step 9.

Step 11. Concatenate the bit streams M_i of all T_i in F in a raster-scan order to form a total bit stream M_i ; use the secret key K to encrypt M_i into another bit stream M_i' ; and embed M_i' into F by the reversible contrast mapping scheme proposed in [44].

Step 12. Construct a bit stream I including: 1) the number of conducted iterations N_i for embedding M_i' ; 2) the number of pixel pairs N_{pair} used in the last iteration; and 3) the Huffman table HT constructed for the residuals; and embed the bit stream I into mosaic image F by the same scheme used in Step 11.

Algorithm 2.2. Secret image recovery.

Input: a mosaic image F with n tile images $\{T_1, T_2, \dots, T_n\}$ and the secret key K .

Output: the secret image S .

Steps:

Stage 1 — extracting the secret image recovery information.

- Step 1. Extract from F the bit stream I by a reverse version of the scheme proposed in [44] and decode them to obtain the following data items: 1) the number of iterations N_i for embedding M_i' ; 2) the total number of used pixel pairs N_{pair} in the last iteration; and 3) the Huffman table HT for encoding the values of the residuals of the overflows or underflows.
- Step 2. Extract the bit stream M_i' using the values of N_i and N_{pair} by the same scheme used in the last step.
- Step 3. Decrypt the bit stream M_i' into M_i by K .
- Step 4. Decompose M_i into n bit streams M_1 through M_n for the n *to-be-constructed* tile images T_1 through T_n in S , respectively.
- Step 5. Decode M_i for each tile image T_i to obtain the following data items: 1) the index j_i of the block B_{j_i} in F corresponding to T_i ; 2) the optimal rotation angle θ_0 of T_i ; 3) the means of T_i and B_{j_i} and the related standard deviation quotients of all color channels; and 4) the overflow/underflow residual values in T_i decoded by the Huffman table HT .

Stage 2 — recovering the secret image.

- Step 6. Recover one by one in a raster-scan order the tile images T_i , $i = 1$ through n , of the desired secret image S by the following steps: 1) rotate in the reverse direction the block indexed by j_i , namely B_{j_i} , in F through the optimal angle θ_0 and fit the resulting block content into T_i to form an *initial* tile image T_i ; 2) use the extracted means and related standard deviation quotients to recover the original pixel values in T_i according to (4); 3) use the extracted means, standard deviation quotients, and (5) to compute the two parameters c_S and c_L ; 4) scan T_i to find out pixels with values 255 or 0 which indicate that overflows or underflows, respectively, have occurred there; 5) add respectively the values c_S or c_L to the corresponding residual values of the found pixels; and 6) take the results as the final pixel values, resulting in a *final* tile image T_i .
- Step 7. Compose all the final tile images to form the desired secret image S as output.

2.5 Experimental Results

A series of experiments have been conducted to test the proposed method using many secret and target images with sizes 1024×768 or 768×1024 . To show that the created mosaic image looks like the pre-selected target image, the quality metric of root mean square error (RMSE) is utilized, which is defined as the square root of the mean square difference between the pixel values of the two images.

An example of the experimental results is shown in Figure 2.5, where Figure 2.5(c) shows the created mosaic image using Figure 2.5(a) as the secret image and Figure 2.5(b) as the target image. The tile image size is 8×8 . The recovered secret image using a correct key is shown in Figure 2.5(d) which looks nearly identical to the original secret image shown in Figure 2.5(a) with $RMSE = 0.948$ with respect to the secret image. It is noted by the way that all the other experimental results shown in this paper have small RMSE values as well, as seen in Figure 2.10(c).

Moreover, Figure 2.5(e) shows the recovered secret image using a wrong key, which is a noise image. Figures 2.5(f) through 2.5(i) show more results using different tile image sizes. It can be seen from the figures that the created mosaic image retains more details of the target image when the tile image is smaller. It can also be seen that the blockiness effect is observable when the image is magnified to be large; but if the image is observed as a whole, it still looks like a mosaic image with its appearance similar to the target image. Figure 2.10(a) also shows this fact in another way — a mosaic image created with smaller tile images has a smaller RMSE value with respect to the target image. On the other hand, the number of required bits embedded for recovering the secret image will be increased when the tile image becomes smaller, as can be seen from Figure 2.10(b).

Figure 2.6 shows a comparison of the results yielded by the proposed method with those by Lai and Tsai [39], where Figure 2.6(a) is the input secret image, Figure 2.6(b) is the selected target image, Figure 2.6(c) is the mosaic image created by Lai and Tsai [39], and Figure 2.6(d) is that created by the proposed method. It can be seen from these results that the mosaic image yielded by the proposed method has a smaller RMSE value with respect to the target image, implying that it is more similar to the target image in appearance. The other results of our experiments also show the same conclusion. And more importantly, the proposed method allows users to select their favorite images for uses as target images.



Figure 2.5. An experimental result of mosaic image creation. (a) Secret image. (b) Target image. (c) Mosaic image created with tile image size 8×8 . (d) Recovered secret image using a correct key with $RMSE = 0.948$ with respect to secret image (a). (e) Recovered secret image using a wrong key. (f)-(i) Mosaic images created with different tile image sizes 16×16 , 24×24 , 32×32 , and 40×40 .

Figure 2.7 shows two other experimental results of mosaic image creation, where the utilized secret images both contain many structures (Figure 2.7(a) is a stained-glass window painting and Figure 2.7(d) is a document image) and Figures 2.7(b) and 2.7(e) are the target images; Figures 2.7(c) and 2.7(f) are the created mosaic images with image sizes 8×8 ; and Figures 2.7(g) and 2.7(h) are the zoom-out images of the red square regions of Figures 2.7(c) and 2.7(f), respectively. It can be seen from Figures 2.7(c) and 2.7(f) that each created mosaic image still has the visual

appearance of the pre-selected target image even when the secret image contains many structural elements. Especially, the secret image of Figure 2.7(d) is a nearly black-and-white *document image*, which means that the proposed method can be utilized for secure transmissions of confidential document images as well. Moreover, it can be seen from Figures 2.7(g) and 2.7(h) that each generated mosaic image has a blocky appearance which comes from the mosaic effect because the mosaic image is composed by changing the color characteristics of the fragments of the secret image and rearranging the resulting fragments. To show the flexibility of the proposed method for a user to choose *any* target image as the reference of a secret image, we selected one secret image as shown in Figure 2.8 and two target images as shown in Figures 2.7(b) and 2.7(e), and transformed the former to have the visual appearance of each of the latter ones. The results are shown in Figures 2.8(b) and 2.8(c) from which we can see that the created mosaic images look similar to the respective target images even though the secret image is quite different from the target images in appearance.

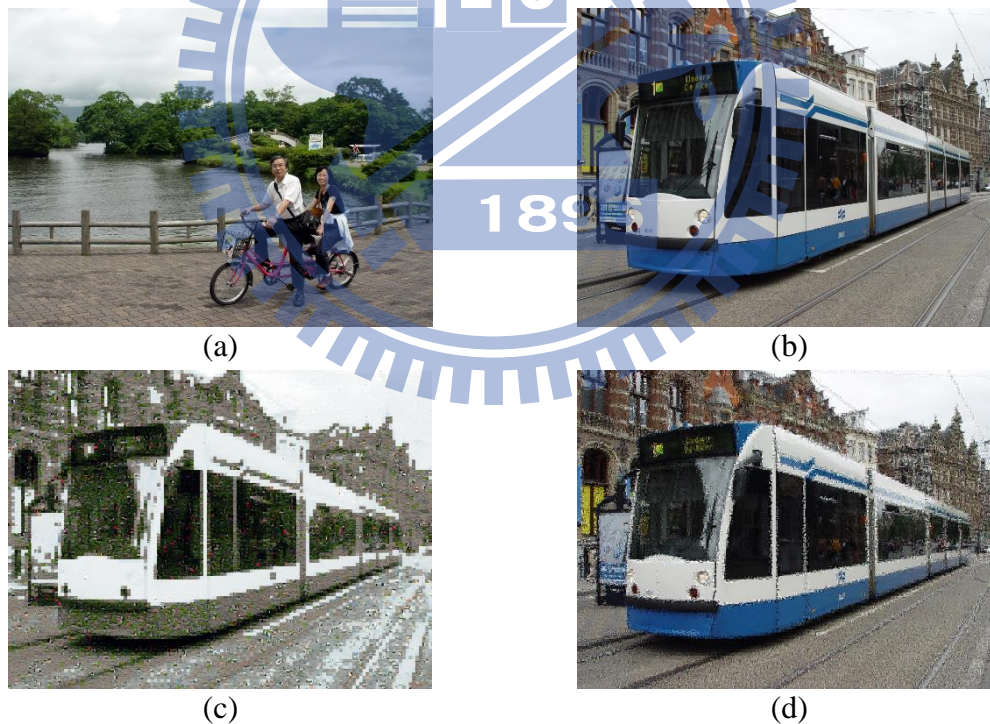


Figure 2.6. Comparison of results of Lai and Tsai [39] and proposed method. (a) Secret image. (b) Target image. (c) Mosaic image created from (a) and (b) by [39] with RMSE = 47.651. (d) Mosaic image created from (a) and (b) by proposed method with RMSE = 33.935.



Figure 2.7. Two other experimental results of mosaic image creation. (a) and (d) Secret images. (b) and (e) Target images. (c) and (f) Mosaic images created from (a) and (b), and (d) and (e), respectively, with tile size 8×8 . (g) and (h) Zoom-out images of red square regions of (c) and (f), respectively.

However, since the mosaic image is yielded by dividing the secret image into tile images and transforming their color characteristics to be those of the corresponding target blocks, the global color characteristics of a transformed tile image and its corresponding target block are the same but the color distributions of them may be quite different. Hence, although the mosaic image has the visual appearance of the target image, the details of each fragment in the mosaic image may have low similarity to those of its corresponding target block. To measure this *mosaic effect*, we adopt the metric of mean structural similarity (MSSIM) to compare the

similarity of the created mosaic image and the target image [47]. Figure 2.10(d) shows the MSSIM values of the created mosaic images with respect to the target images versus different tile image sizes, where the window size for computing the MSSIM is set to be the same as the size of the tile image. We can see from Figure 2.10(d) that the MSSIM value of the created mosaic image with respect to the target image varies from 0.2 to 0.8, which shows that the similarity of the details of the created mosaic image to those of the target image is not good enough. But, this is not the main concern of the proposed method because our goal is to create a globally visually-similar mosaic image, which contains a secret image of the same size, for the purpose of secure image transmission.



Figure 2.8. Created mosaic images with the same secret image. (a) Secret image. (b) Mosaic image created from (a) and Figure 2.7(b) with RMSE = 26.067. (c) Mosaic image created from (a) and Figure 2.7(e) with RMSE = 33.102.

A limitation of the proposed method is that the sizes of available target images should match those of possible input secret images. Specifically, if we have a very large secret image but only small target images for selections, then any selected target image should be enlarged before mosaic image creation in order to match the size of the secret image, and the created mosaic image will become *blurred*. An experimental result showing this blurring effect is presented in Figure 2.9.



Figure 2.9. Created mosaic images with the same secret image shown in Fig. 5(a) and small-sized target images. (a) Created image for target image shown in Fig. 5(b) with size 768×1024 . (b) Created image for target image shown in Fig. 5(b) but with size reduced to $(1/5) \times (1/5)$. (c) Created image for target image shown in Fig. 5(b) but with size reduced to $(1/10) \times (1/10)$.

Furthermore, as shown in Figure 2.10, we have drawn plots of the trends of various parameters versus different tile image sizes, including those for the parameters of 1) the RMSE values of the created mosaic images with respect to the target images; 2) the numbers of required bits embedded for recovering the secret images; 3) the RMSE values of the recovered secret images with respect to the original ones; and 4) the MSSIM values of created mosaic images with respect to target images.

In addition, we have conducted experiments on a set of 12 images from which a total of $12 \times 11 = 132$ secret-target image pairs are selected without repetitions, and the averages of the parameters of the 132 mosaic image creation results were also plotted in Figure 2.10 as the orange curves for comparisons.

2.6 Security Considerations

In order to increase the security of the proposed method, the embedded information for later recovery is encrypted with a secret key as seen in Algorithm 2.1. Only the receiver who has the key can decode the secret image. However, an eavesdropper who does not have the key may still try all possible permutations of the tile images in the mosaic image to get the secret image back. Fortunately, the number of all possible permutations here is $n!$, and so the probability for him/her to correctly guess the permutation is $p = 1/n!$ which is very small in value. For example, for the typical case where we divide a secret image of size 1024×768 into tile images with

block size 8×8 , the value n is $(1024 \times 768) / (8 \times 8) = 12,288$. So the probability to guess the permutation correctly without the key is $1/n! = 1/(12,288!)$. So breaking the system by this way of guessing is computationally infeasible.

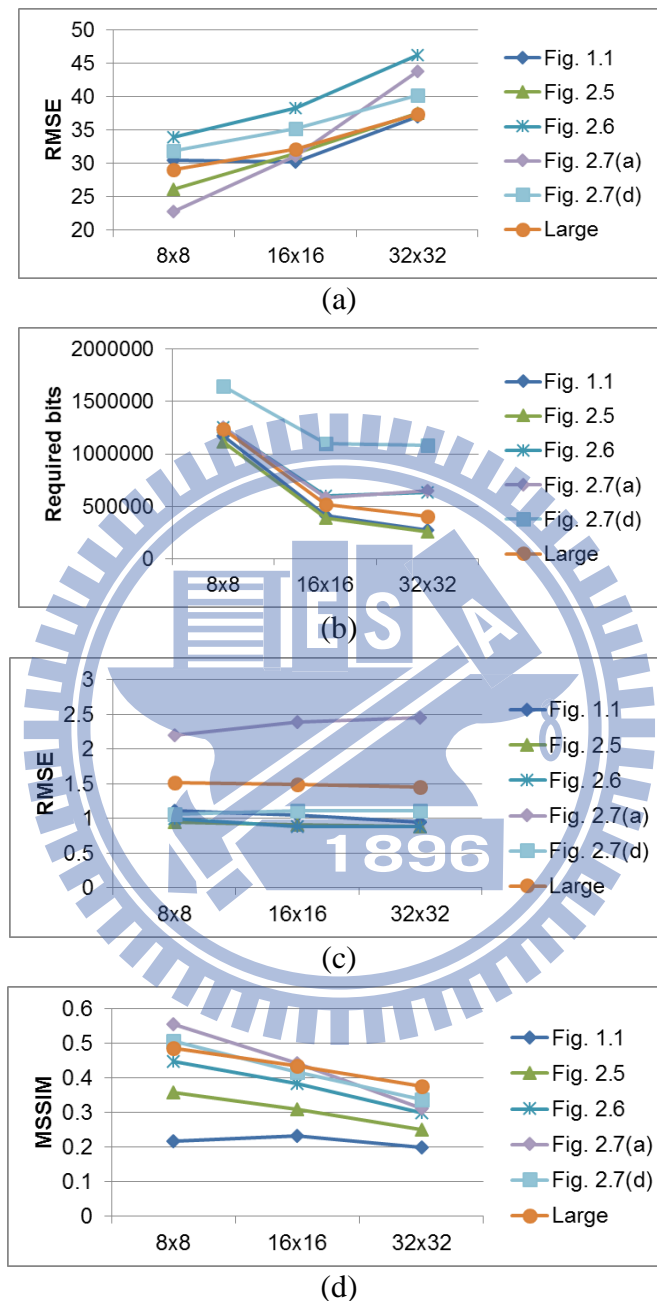


Figure 2.10. Plots of trends of various parameters versus different tile image sizes (8×8 , 16×16 , 32×32) with input secret images shown previously and coming from a large dataset. (a) RMSE values of created mosaic images with respect to target images. (b) Numbers of required bits embedded for recovering secret images. (c) RMSE values of recovered secret images with respect to original ones. (d) MSSIM values of created mosaic images with respect to target images.

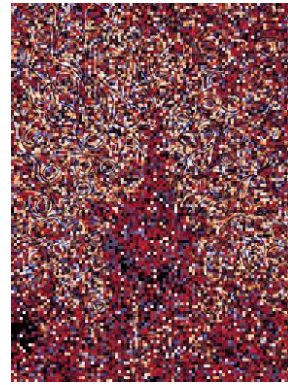
In fact, we can view the addressed problem here as a *square jigsaw puzzle problem*, which is to reconstruct a complete image from a set of unordered square puzzle parts. Recently, many methods have been proposed to try to solve this problem automatically by utilizing measures of feature-based similarity [48], dissimilarity-based compatibility [49], prediction-based compatibility [50], etc. But these state-of-art methods can only solve partially problems with limited numbers of puzzle parts automatically. Also, the jigsaw puzzle problem has been proved to be NP-complete [51], which means that we cannot solve the problem in polynomial time. In fact, the time complexity is $n! \approx \sqrt{2\pi n}(n/e)^n$ as mentioned in [51], which is too big a number as well for our case here with $n = 12,288$.

However, when n is much smaller, say smaller than 1000, some compatibility metrics may be utilized to solve the square jigsaw problem [50]. So, a large value of n should be used to increase the security of the proposed method. In addition, the addressed puzzle problem of the proposed method is more complicated than the conventional square jigsaw puzzle problem because the color characteristics of the puzzle parts have been changed, i.e., adjacent puzzle parts have different color appearances, meaning that a greedy search using color similarities between originally adjacent fragments for image reconstruction as done in conventional manual reconstruction techniques is infeasible, either.

Furthermore, even if one happens to guess the permutation correctly, such as the correctly guessed permutations shown in Figure 2.11, he/she still does not know the correct parameters for recovering the original color appearance of the secret image because such parameter information for color recovery is encrypted as a bit stream using a secret key. Even so, it still should be assumed, in the extreme case, that he/she will observe the content of the mosaic image with a correct permutation, and try to figure useful information out of it. For example, an attacker might analyze the spatial continuity of the mosaic image in order to estimate a rough version of the secret image. To increase the security of the proposed method against this type of attack, one possible way to is to use the key to randomize the important part of a secret image, such as the positions of the pixels in the secret image, before transforming the secret image into a mosaic image by the proposed method. Consequently, only authorized users with the key can know the correct secret image while an attacker cannot.



(a)



(b)

Figure 2.11. Correct permutations of tile images in the mosaic image without recovering the original color characteristics. (a) The correct permutation of tile images of Figure 1.1(c). (b) The correct permutation of tile images of Figure 2.7(c).

2.7 Summary

A new image data hiding method has been proposed, which not only can be used for secure keeping of secret images but also can be adopted as a new option to solve the difficulty of hiding images with huge data volumes into cover images. By the use of proper pixel color transformations as well as a skillful scheme for handling overflows and underflows in the converted values of the pixels' colors, secret-fragment-visible mosaic images with very high visual similarities to arbitrarily-selected target images can be created with no need of a target image database; and the original secret images can be recovered nearly losslessly from the created mosaic images. Good experimental results have shown the feasibility of the proposed method. Future studies may be directed to applying the proposed method to images of color models other than the RGB.

Chapter 3

A New Data Hiding Technique via Encrypted Images by Image Encryptions and Spatial Correlation Comparisons

3.1 Introduction

Nowadays, images from various sources are frequently utilized and transmitted through the Internet for various applications, such as online personal photograph albums, confidential enterprise archives, document storage systems, medical imaging systems, military image databases, etc. These images usually contain private or confidential information so that they should be protected from leakages during transmissions. Therefore, some methods jointing data hiding and encryption techniques have been proposed [52]-[54], in which a part of the cover media is encrypted and the other part is used for data embedding. Such methods, however, reveal undesirably the content of the second part of the cover media. To provide higher security, Zhang [55] proposed a method that can prevent people, including the data hider, from realizing the cover media content before or after the data embedding process is performed. Specifically, the cover image is encrypted *entirely*, instead of partially, by the content owner using a key, and the result is then delivered to the data hider for data embedding. Also, the original cover image can be recovered after the embedded data are extracted. Hong *et al.* [56] improved Zhang's method [55] by using a side-match scheme based on uses of spatial correlations of adjacent blocks.

Either [55] or [56] embeds a message into an encrypted image by flipping the three least significant bits (LSBs) of a portion of the pixels of each image block to embed a message bit. Data extraction and image recovery are achieved by using spatial correlations. However, such a message embedding scheme suffers from a problem which occurs when the cover image is a *flat image*, i.e., when the cover image has lots of *smooth* regions with the characteristic that the most significant bits (MSBs) of the pixels in each of such regions are all the same.

In this study, the LSBs of each block pixel in an encrypted image are *encrypted further*, rather than flipped, to embed a message bit, thereby solving the

aforementioned problem encountered in [55] and [56] which is caused by flat cover images. Also, the spatial similarity of the original LSBs of the pixels in each block is broken by the encryption function. Moreover, for each pixel of a block in the encrypted image, *four* LSBs, instead of three, are utilized for message embedding, and a side-match scheme that utilizes the spatial correlations of *both* recovered and unrecovered blocks are proposed to decrease the bit-extraction error rate, in contrast with [56] which utilizes the spatial correlations of recovered blocks only.

3.2 Review of Existing Methods

The reversible data hiding scheme proposed in Zhang [55] for grayscale images includes three phases: 1) image encryption, 2) message embedding, and 3) message extraction and image recovery. In the first phase, a cover image I is encrypted by performing the exclusive-OR (XOR) operation \oplus on all bits and their corresponding *random bits* generated by the use of an *encryption key* K_e and a random number generator PR_e . Specifically, by denoting the value of each pixel P_{ij} in I by p_{ij} , each bit of P_{ij} by $b_{i,j,k}$, and the generated random bit corresponding to $b_{i,j,k}$ by $r_{i,j,k}$, the encryption of I is conducted by replacing $b_{i,j,k}$ by $b_{i,j,k}' = b_{i,j,k} \oplus r_{i,j,k}$ for all i, j , and k , resulting in an *encrypted image* I' with pixels $p_{i,j}'$ and bits $b_{i,j,k}'$.

In the message embedding phase, I' is divided into blocks $B_{m,n}$ of size $s \times s$, with each block used to carry a bit. Specifically, firstly each pixel $P_{i,j}'$ in $B_{m,n}$ is assigned into two *random sets* S_0 and S_1 using a *data-hiding key* K_h and another random number generator PR_h . Then, if the bit to be embedded into $B_{m,n}$ is 0, the three LSBs $b_{i,j,k}'$ of each pixel $P_{i,j}'$ in S_0 are flipped to be their complements $\overline{b_{i,j,k}'}$, resulting in a new pixel $P_{i,j}''$ with value $p_{i,j}''$; else, the three LSBs of each pixel $P_{i,j}'$ in the other random set S_1 are flipped. The resulting image is denoted by I'' .

In the last phase — message extraction and image recovery, firstly I'' is decrypted to obtain a decrypted *image* I''' by performing an XOR operation \oplus on every bit $b_{i,j,k}''$ in I'' and the corresponding random bit $r_{i,j,k}$ re-generated by the encryption key K_e . By denoting the resulting *decrypted pixels* and *decrypted bits* as $P_{i,j}'''$ and $b_{i,j,k}'''$, respectively, it can be seen that the five MSBs of $P_{i,j}'''$ in I''' are *identical* to the corresponding ones of the original pixel $P_{i,j}$ in I but the *three LSBs are not*. Next, by using the data-hiding key K_h , the random sets S_0 and S_1 may be re-generated for each *decrypted block* $B_{m,n}'$ of I''' . Then, the three LSBs of the pixels

in S_0 and those in S_1 are both flipped to form blocks $H_{m,n,0}$ and $H_{m,n,1}$, respectively. It can be figured out that one of $H_{m,n,0}$ and $H_{m,n,1}$ is identical to the *original* block $B_{m,n}$ in I , and the other is an *interfered* version of $B_{m,n}$ since all the three LSBs of the latter have been flipped. Because of the spatial correlations of pixels in natural images, the original block is usually smoother than the interfered version. So the embedded bit can be extracted, as done in [55], by using a block smoothness measure f as follows:

$$f = \sum_{u=2}^{s-1} \sum_{v=2}^{s-1} |p_{u,v} - (p_{u-1,v} + p_{u,v-1} + p_{u+1,v} + p_{u,v+1}) / 4|, \quad (8)$$

where $p_{u,v}$ denotes the value of a pixel $P_{u,v}$ in the block. Specifically, by denoting the smoothness values calculated of $H_{m,n,0}$ and $H_{m,n,1}$ as $f_{m,n,0}$ and $f_{m,n,1}$, respectively, bit extraction is conducted by the rule: if $f_{m,n,0} < f_{m,n,1}$, a bit 0 is extracted and $H_{m,n,0}$ is taken as the original block; else, a bit 1 is extracted and $H_{m,n,1}$ is taken as the original block.

The reversible data hiding scheme proposed in Hong *et al.* [56] is the same as described above except that a different smoothness measure as follows with a better effect is used:

$$f' = \sum_{u=1}^s \sum_{v=1}^{s-1} |p_{u,v} - p_{u,v+1}| + \sum_{u=1}^{s-1} \sum_{v=1}^s |p_{u,v} - p_{u+1,v}|. \quad (9)$$

Moreover, a side-match scheme is adopted, using additionally the spatial correlations of *adjacent recovered blocks* to reduce the error rate of bit extraction.

A problem with [55] occurs when the five MSBs of the pixels in a block are *all identical* — a case encountered when the cover image is *flat*. The problem is: the smoothness measures $f_{m,n,0}$ and $f_{m,n,1}$ computed of $H_{m,n,0}$ and $H_{m,n,1}$, respectively, become *the same* in such a case, as proved below, and the resulting bit extraction will so be *no better than random guesses*, i.e., the probability to extract a correct bit is about 1/2.

Let the five MSBs of *each* pixel $P_{i,j}$ in the *original* block $B_{m,n}$ be denoted identically as $b_8 \sim b_4$ under the assumption of image flatness. Also, let the pixels in $H_{m,n,0}$ and $H_{m,n,1}$ be denoted as $P_{u,v,0}$ and $P_{u,v,1}$ with values $p_{u,v,0}$ and $p_{u,v,1}$, respectively. Without loss of the generality, let $H_{m,n,0}$ be the one identical to the original block $B_{m,n}$ and $H_{m,n,1}$ the interfered one. Thus, $p_{u,v,0}$ in $H_{m,n,0}$ are all identical to the corresponding pixel values $p_{u,v}$ of $B_{m,n}$, and the three LSBs of $p_{u,v,1}$ in $H_{m,n,1}$ are the flipped versions

of $p_{u,v}$ in $B_{m,n}$. That is, the five MSBs of each pixel in $H_{m,n,0}$ and $H_{m,n,1}$ are identically $b_8 \sim b_4$, respectively; and if the three LSBs of each pixel $P_{u,v,0}$ in $H_{m,n,0}$ are $b_{u,v,3}, b_{u,v,2}$, and $b_{u,v,1}$, then the three LSBs of the corresponding pixel $P_{u,v,1}$ in $H_{m,n,1}$ are just $\overline{b_{u,v,3}}$, $\overline{b_{u,v,2}}$, and $\overline{b_{u,v,1}}$. Note that $b_{u,v,k} + \overline{b_{u,v,k}} = 1$. Now, the block smoothness values $f_{m,n,0}$ and $f_{m,n,1}$ can be computed respectively according to (8), leading the following derivation:

$$\begin{aligned}
f_{m,n,1} &= \sum_{u=2}^{s-1} \sum_{v=2}^{s-1} \left| p_{u,v,1} - (p_{u-1,v,1} + p_{u,v-1,1} + p_{u+1,v,1} + p_{u,v+1,1}) / 4 \right| \\
&= \sum_{u=2}^{s-1} \sum_{v=2}^{s-1} \left| \sum_{k=1}^3 \left[\overline{b_{u,v,k}} - (\overline{b_{u-1,v,k}} + \overline{b_{u,v-1,k}} + \overline{b_{u+1,v,k}} + \overline{b_{u,v+1,k}}) / 4 \right] \times 2^{(k-1)} \right| \\
&= \sum_{u=2}^{s-1} \sum_{v=2}^{s-1} \left| \sum_{k=1}^3 [(1 - b_{u,v,k}) - ((1 - b_{u-1,v,k}) + (1 - b_{u,v-1,k}) + (1 - b_{u+1,v,k}) + (1 - b_{u,v+1,k})) / 4] \times 2^{(k-1)} \right| \\
&= \sum_{u=2}^{s-1} \sum_{v=2}^{s-1} \left| \sum_{k=1}^3 \left[b_{u,v,k} - (b_{u-1,v,k} + b_{u,v-1,k} + b_{u+1,v,k} + b_{u,v+1,k}) / 4 \right] \times 2^{(k-1)} \right| \\
&= \sum_{u=2}^{s-1} \sum_{v=2}^{s-1} \left| p_{u,v,0} - (p_{u-1,v,0} + p_{u,v-1,0} + p_{u+1,v,0} + p_{u,v+1,0}) / 4 \right| = f_{m,n,0}. \tag{10}
\end{aligned}$$

The above-mentioned problem is also found in [56], i.e., the block smoothness values $f_{m,n,0}'$ and $f_{m,n,1}'$ computed according to (9) are equal when the five MSBs of the pixels in the original block $B_{m,n}$ are *all identical*. The proof, also based on the equality $b_{u,v,k} + \overline{b_{u,v,k}} = 1$, is similar to (10) and so omitted.

Figure 3.1(a) shows a flat image with 8×8 blocks where the pixel values of *each* block are reassigned artificially to be all the same. Figure 3.1(b) shows the incorrectly-recovered blocks (marked in white) yielded by [55] with an error rate of 50.81%, and Figure 3.1(c) shows those yielded by [56] with an error rate of 44.53%. In contrast, the proposed method (described in the next section) yields a result as shown in Figure 3.1(d) with an error rate of 0%, showing the effectiveness of the proposed method. Instead of using artificially-created images, Figures 3.1(e) through 3.1(h) shows the recovery results using an original X-ray image, Figure 3.1(e), as input, where Figure 3.1(f) shows the result yielded by [55] with an error rate of 16.53%, Figure 3.1(g) shows that yielded by [56] with an error rate of 14.99%, and

Figure 3.1(h) shows that yielded by the proposed method with an error rate of 0% again.

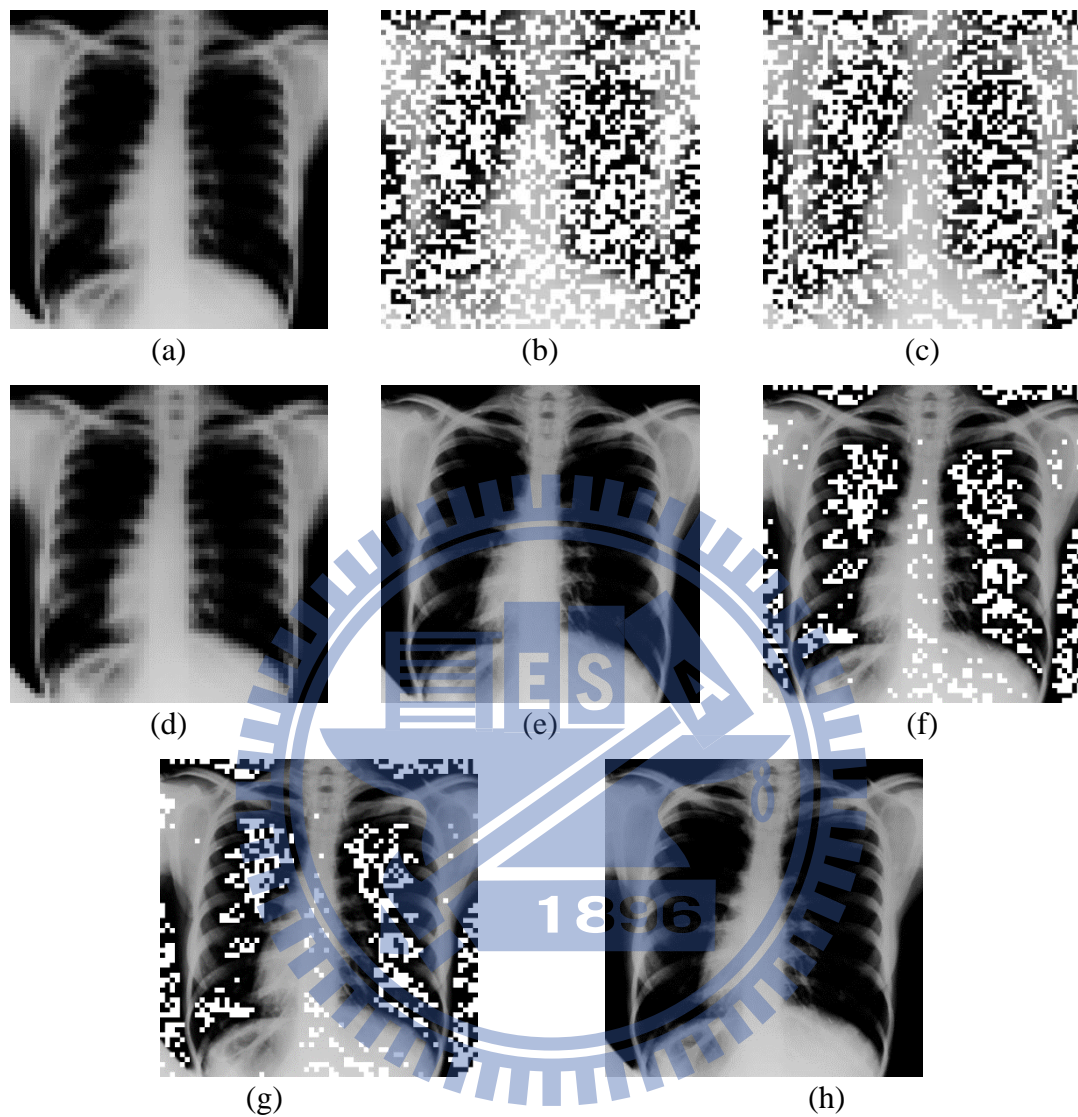


Figure 3.1. Recovery results showing problems of [55] and [56] with block size 8×8 , incorrectly-recovered blocks marked as white, and error rate denoted by *err*. (a) Input flat X-ray image. (b) Result with *err* = 50.81% yielded by [55]. (c) Result with *err* = 44.53% yielded by [56]. (d) Result with *err* = 0% yielded by proposed method. (e) Input original image of (a). (f) Result with *err* = 16.53% yielded by [55]. (g) Result with *err* = 14.99% yielded by [56]. (h) Result with *err* = 0% yielded by proposed method.

3.3 Proposed Method

3.3.1 Message embedding

In the proposed method, the cover image I is encrypted using an encryption key K_e as done in [55], and each block $B_{m,n}$ of the resulting encrypted image I' is used to carry a message bit as well. However, unlike [55] and [56] which *flip* the three LSBs of each involved pixel in $B_{m,n}$, *four* LSBs are *randomized* to increase the probability for distinguishing original blocks from altered (interfered) ones using spatial correlations while the visual appearance of the decrypted image is still kept good enough. Next, four random bits $r_{i,j,k'}$ corresponding to the four LSBs $b_{i,j,k'}$ of each pixel $P_{i,j'}$ in $B_{m,n}$ are generated using the data-hiding key K_h ; and if a bit to be embedded in $B_{m,n}$ is 0, then the four LSBs $b_{i,j,k'}$ of each pixel $P_{i,j'}$ in $B_{m,n}$ are replaced by $b_{i,j,k''} = b_{i,j,k'} \oplus r_{i,j,k'}$, resulting in the new pixel value $p_{i,j''}$; else, $p_{i,j''}$ is taken to be the old value $p_{i,j'}$ of $P_{i,j}$. Let the resulting encrypted image be denoted as I'' . The overall effect is: if the embedded bit in a block is 0, then the four LSBs of each block pixel are *encrypted twice* by the keys K_e and K_h ; else, *once* by the key K_e only. So, the embedded bit in each block may be extracted by decrypting the block content with keys K_e and K_h and measuring the spatial correlations of the block to decide if the block has been encrypted *once* or *twice* as described next.

3.3.2 Message extraction and image recovery

To extract the message embedded in the encrypted image I'' , firstly the key K_e is used to *decrypt* image I'' to obtain another, denoted by I''' , whose pixels' four MSBs are all the same as those of the pixels of the original cover image I . Next, the four corresponding random bits $r_{i,j,k'}$ are re-generated using the key K_h for the four LSBs $b_{i,j,k'''}$ of each pixel $P_{i,j''}$ in each block $B_{m,n'}$ of size $s \times s$ in I''' , and XOR operations are applied to $b_{i,j,k'''}$ and $r_{i,j,k'}$ to form another block $H_{m,n,0}$. Also, $B_{m,n'}$ itself is regarded as a *contrastive* block $H_{m,n,1}$. It can be figured out that one of $H_{m,n,0}$ and $H_{m,n,1}$ is the *original* cover block $B_{m,n}$; and the other is a *scrambled* version of $B_{m,n}$ because the four LSBs of this block's pixels have been encrypted for the second time using the key K_h . To decide which one is $B_{m,n}$, the smoothness measures of $H_{m,n,0}$ and $H_{m,n,1}$ can be utilized because: if the embedded bit is 0, then since the original *cover block* is encrypted *twice* by the keys K_e and K_h , the decrypted block $H_{m,n,0}$ using the same keys K_e and K_h will become the original block, which usually is smoother than the

scrambled version $H_{m,n,1}$; and if the embedded bit is 1, then since the original cover block is encrypted *only once* by the key K_e , the decrypted block $H_{m,n,1}$ using the same key K_e will become the original block, which is usually smoother than the scramble version $H_{m,n,0}$ as well. Moreover, to compute the block smoothness, we adopt the measure used in [56] described by (9), but, unlike the side-match scheme used in [56] which utilizes only the recovered block to compute the smoothness, a new side-match scheme using both unrecovered and recovered blocks is proposed.

In more detail, for each block $B_{x,y}'$ of the four blocks adjacent to each block $B_{m,n}'$ in I''' , if $B_{x,y}'$ is *unrecovered* yet, then the values of $f_{m,n,0}'$ and $f_{m,n,1}'$ computed according to (9) are augmented in the following way:

$$\text{set } f_{m,n,0}' = f_{m,n,0}' + \min(|H_{m,n,0}' - H_{x,y,0}'|, |H_{m,n,0}' - H_{x,y,1}'|);$$

$$\text{set } f_{m,n,1}' = f_{m,n,1}' + \min(|H_{m,n,1}' - H_{x,y,0}'|, |H_{m,n,1}' - H_{x,y,1}'|),$$

where $H_{x,y,q}'$ with $q = 0$ or 1 is generated from $B_{x,y}'$ by the same way as $H_{m,n,p}'$ with $p = 0$ or 1 is generated from $B_{m,n}'$ using the key K_h ; and as illustrated in Figure 3.2, $|H_{m,n,p}' - H_{x,y,q}'|$ with $p, q = 0, 1$ is defined by

$$|H_{m,n,p}' - H_{x,y,q}'| = \sum_{c=1}^s |b_{c,p} - b_{c,q}'|,$$

with $b_{c,p}$ denoting the value of a border pixel of block $H_{m,n,p}'$ adjacent to block $H_{x,y,q}'$; and $b_{c,q}'$ denoting the value of a border pixel of block $H_{x,y,q}'$ adjacent to block $H_{m,n,p}'$. Contrarily, if the adjacent block $B_{x,y}'$ of $B_{m,n}'$ are *recovered* as $H_{x,y,r}'$ already, then $f_{m,n,0}'$ and $f_{m,n,1}'$ are augmented in the following way:

$$\text{set } f_{m,n,0}' = f_{m,n,0}' + |H_{m,n,0}' - H_{x,y,r}'|;$$

$$\text{set } f_{m,n,1}' = f_{m,n,1}' + |H_{m,n,1}' - H_{x,y,r}'|.$$

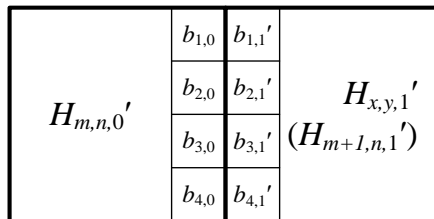


Figure 3.2. Illustration of block contents for computing $|H_{m,n,0}' - H_{x,y,1}'|$ for 4×4 blocks, where currently-processed adjacent block of $B_{m,n}'$ is $B_{m+1,n}'$.

Since blocks adjacent to $B_{m,n'}$ are all used to compute the smoothness no matter whether they are recovered or not, the resulting bit-extraction error rate will be smaller than other schemes not doing so. Figure 3.3 includes some results showing this effect for 8×8 blocks with a comparison with that yielded by [56].

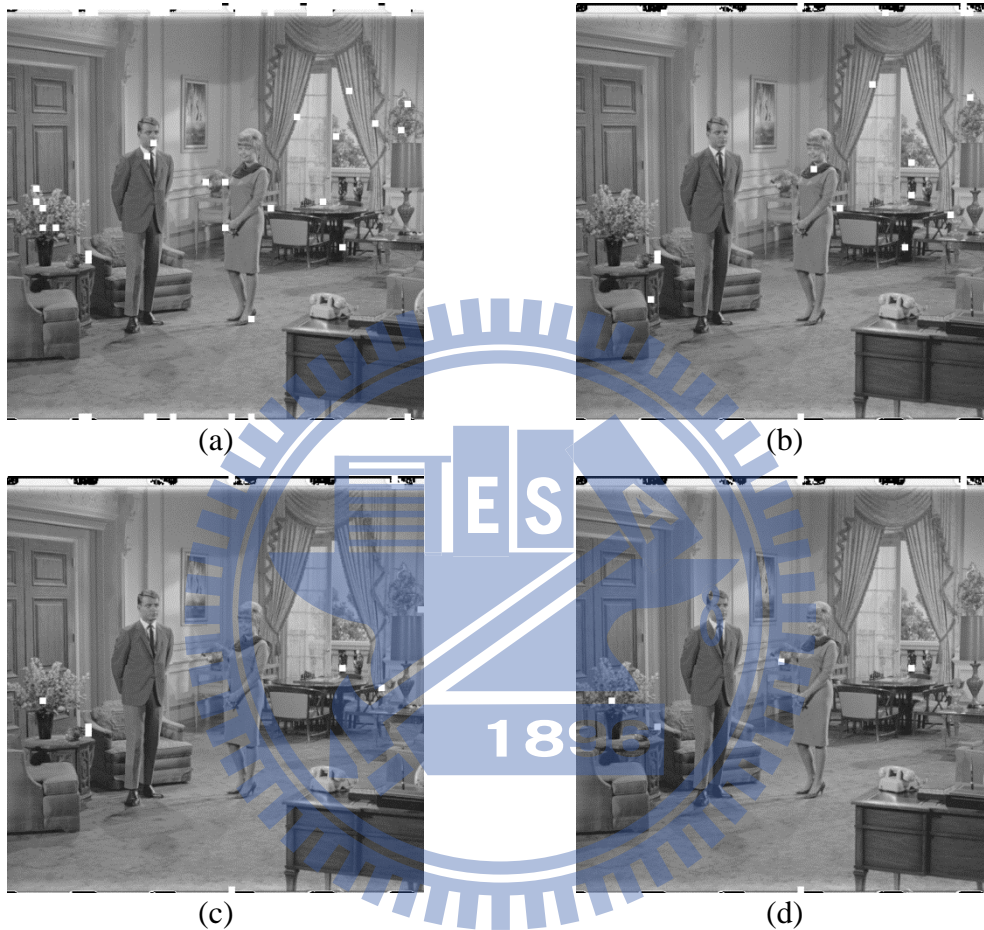


Figure 3.3. Recovery results showing effects of using both recovered and unrecovered blocks for measuring smoothness of 8×8 blocks, with incorrectly-recovered blocks marked as white, and error rate denoted by err . (a) Result with $err = 2.66\%$ yielded by [56]. (b) Result with $err = 0.46\%$ yielded by proposed method without using side-match. (c) Result with $err = 0.27\%$ yielded by proposed method using only recovered blocks in side-match scheme. (d) Result with $err = 0.22\%$ yielded by proposed method using both recovered and unrecovered blocks in side-match scheme.

3.4 Experimental Results

Four 512×512 test images, Figures 3.4(a) through 3.4(d), were used in the experiments, and the results of the proposed method are compared with those yielded by [55] and [56], as illustrated in Figure 3.5 which includes plots of the trends of bit-extraction error rates versus different block sizes $s \times s$. It is seen from Figures 3.5(a) through 3.5(d) that the error rates yielded by the proposed method are much *smaller* than those yielded by [55] and [56]. For example, for the cover image Figure 3.4(a) with block size 8×8, Figure 3.5(a) shows that the bit-extraction error rates using [55] and [56] are 12.87% and 10.21%, respectively; and that yielded by the proposed method is 0.07%. Moreover, Figure 3.5(a) shows that the error rate yielded by the proposed method is zero when s is larger than 12, but those yielded by both [55] and [56] are still larger than zero when $s = 32$.

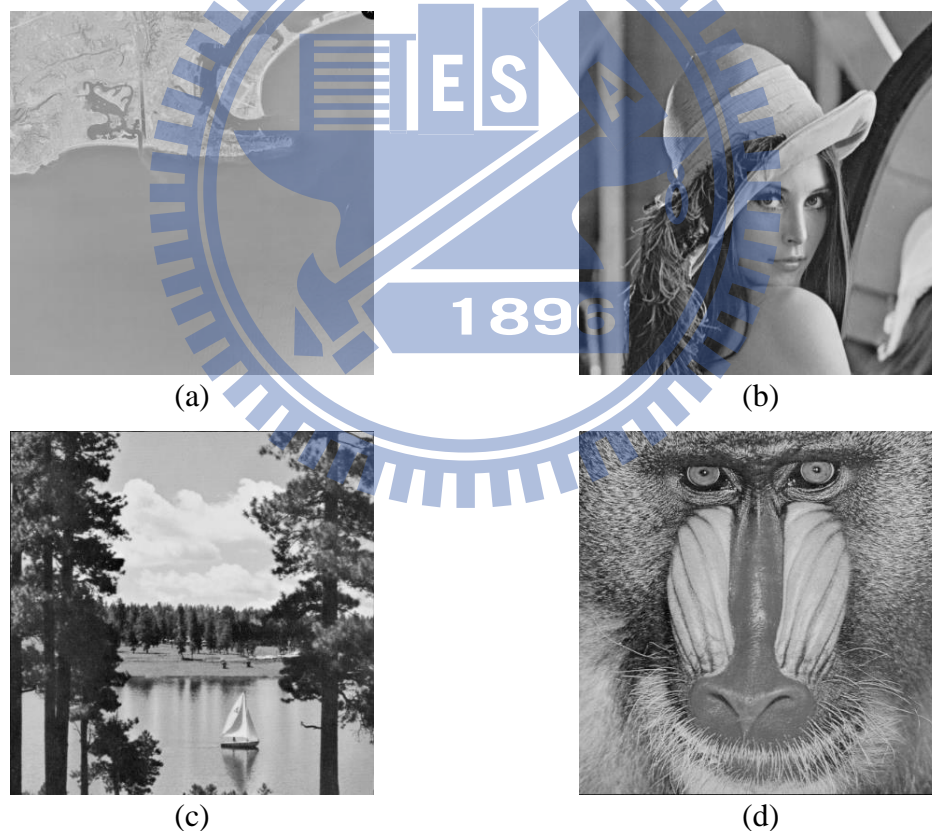


Figure 3.4. Four test images of size 512×512.

Additionally, we compare the execution time of the proposed method with that of [55] and [56] in message embedding. As mentioned previously in Sections 3.2 and 3.3.1, to embed a message bit, three LSBs of a portion of the pixels of each image

block are flipped in [55] and [56] and four LSBs of each block pixel are encrypted further, respectively. Therefore, the execution time of the proposed method and those of [55] and [56] for message embedding are all very short since the operations involve *only encryptions and flippings*, respectively. For example, for the cover image of Figure 3.4(a), Figure 3.6 shows a comparison of the execution time for message embedding required by the proposed method with those required by [55] and [56] versus different block sizes, where the execution times of them are all very fast (about 0.1 ~ 0.15s).

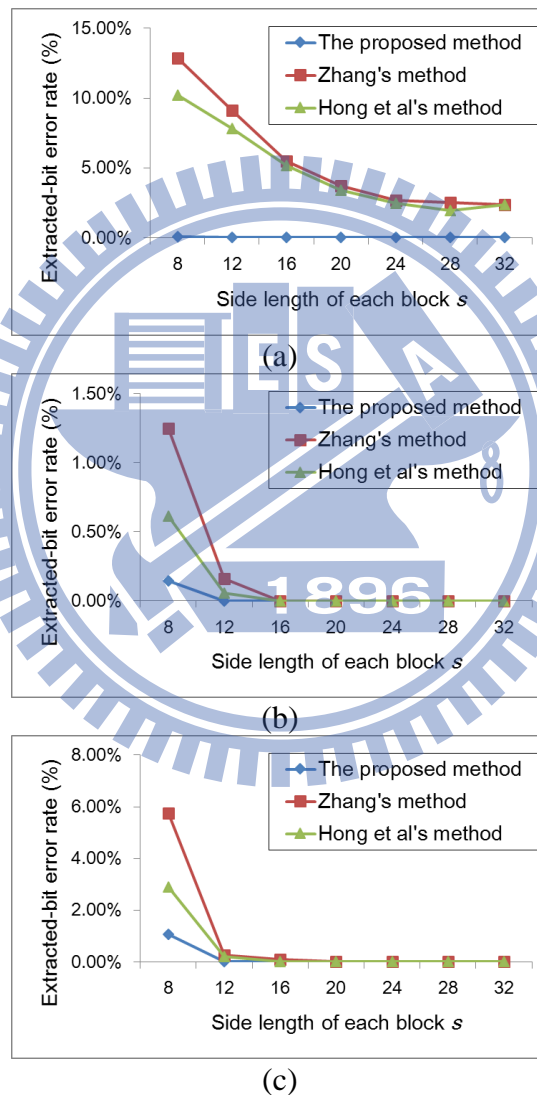
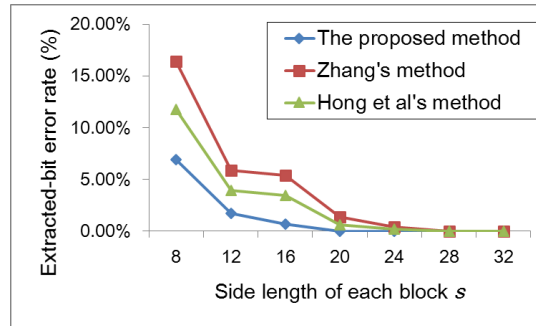


Figure 3.5. Comparisons of bit-extraction error rates yielded by proposed method with those yielded by [55] and [56] versus different block sizes. (a) Error rates with cover image Figure 3.4(a). (b) Error rates with cover image Fig. Figure 3.4(b). (c) Error rates with cover image Figure 3.4(c). (d) Error rates with cover image Figure 3.4(d).



(d)

Figure 3.5. Comparisons of bit-extraction error rates yielded by proposed method with those yielded by [55] and [56] versus different block sizes. (a) Error rates with cover image Figure 3.4(a). (b) Error rates with cover image Figure 3.4(b). (c) Error rates with cover image Figure 3.4(c). (d) Error rates with cover image Figure 3.4(d) (continued).

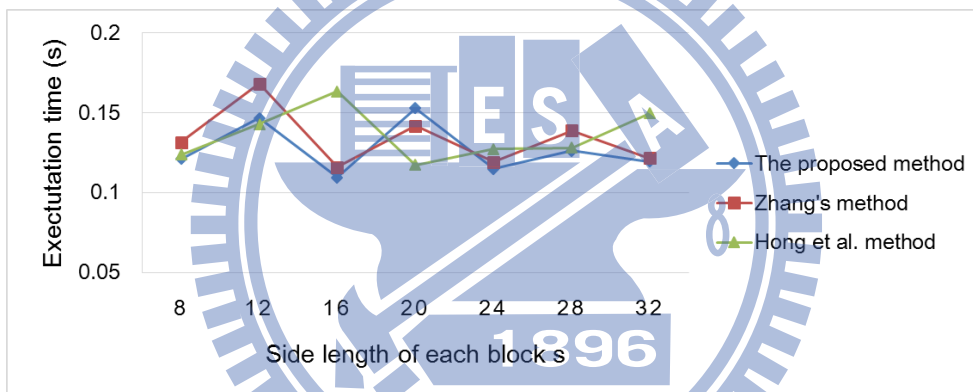


Figure 3.6. Comparison of execution time for message embedding required by proposed method with those required by [55] and [56] versus different block sizes.

Also, experiments for comparisons of the effects of using *three or four* LSBs for data embedding have also been conducted. Results for 8×8 blocks are shown in Figure 3.7, where the number of used LSBs is denoted by N_L . Specifically, the methods [55] and [56] do not perform better when $N_L = 4$ for image Figure 3.4(a) as can be seen from Figures 3.6(e) through 3.6(h). The same conclusions can be drawn for other images. Contrarily, Figures 3.6(c) and 3.6(d) show that the proposed method performs better as N_L is enlarged from 3 to be 4.

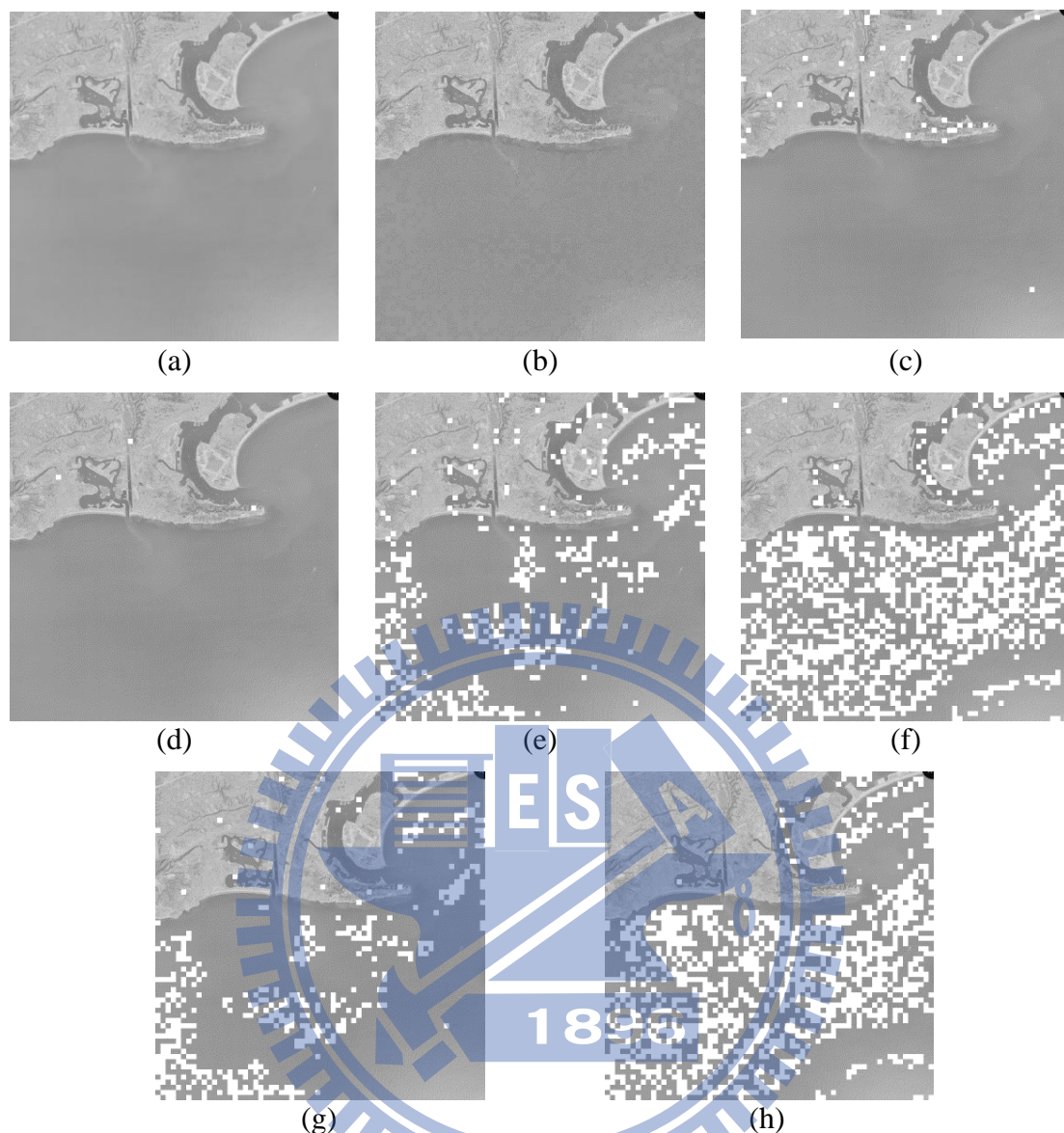


Figure 3.7. Recovery results showing effects of using different numbers N_L of LSBs for 8×8 blocks with incorrectly-recovered blocks marked as white and error rate denoted by r . (a) Cover image. (b) Decrypted image with message embedded. (c) Result with $r = 0.90\%$ yielded by proposed method for $N_L = 3$. (d) Result with $r = 0.07\%$ yielded by proposed method for $N_L = 4$. (e) Result with $r = 12.87\%$ yielded by [55] for $N_L = 3$. (f) Result with $r = 29.27\%$ yielded by [55] for $N_L = 4$. (g) Result with $r = 10.21\%$ yielded by [56] for $N_L = 3$. (h) Result with $r = 27.76\%$ yielded by [56] for $N_L = 4$.

Furthermore, the average distortion of the decrypted image with respect to the original image by using the proposed method can be computed, which is described as follows. Firstly, a decrypted pixel in the decrypted image has two possibilities: (1)

correct decryption – the same as the original pixel; or (2) incorrect decryption – a scrambled version of the original pixel, where the possibility for each case is 1/2. If the decrypted pixel is of the first case, then the average squared difference between the decrypted gray value and the original one is zero; else, the average squared difference between the decrypted gray value and the original one is $\frac{1}{16} \sum_{i=0}^{15} i^2 = 77.5$,

where i represents the difference between the decrypted gray value and the original one. The value of the PSNR of the decrypted image with respect to the original image is approximately

$$\text{PSNR} = 10 \log_{10} \frac{255^2}{(1/2) \times 0 + (1/2) \times 77.5} = 32.25 \text{ dB.}$$

Hence, the average PSNR value of the decrypted image with respect to the original image by using the proposed method is about 32.25 dB, which is not so good as those of [55] and [56] with the average PSNR value of 37.9 dB. However, the proposed method significantly reduces the bit-extraction error rate and solves the flat image problem without keeping the spatial similarity of the LSBs of the pixels in each block as mentioned previously.

3.5 Summary

A new data hiding method via encrypted images based on double image encryptions and refined spatial correlation comparison has been proposed, which does not have the weakness of two existing methods [55] and [56] in handling flat cover images. The weakness comes from the way of flipping the three LSBs of each pixel in part of each block in an encrypted image to embed a message bit. The proposed method improves this by encrypting the four LSBs of each pixel of every block instead of flipping three of them to embed a bit. Also, a refined side-match scheme utilizing the spatial correlations of both recovered and unrecovered blocks has been proposed to decrease the bit-extraction error rate, in contrast with Hong *et al.* [56] which utilizes only those of recovered blocks. Experimental results show the feasibility of the proposed method. Future studies may be directed to applying the proposed method for various information hiding purposes.

Chapter 4

A New Data Hiding Technique via Revision

History Records on Collaborative Writing

Platforms

4.1 Introduction

Recently, more and more collaborative writing platforms are available, such as Google Drive, Office Web Apps, Wikipedia, etc. On these platforms, a huge number of revisions generated during the collaborative writing process are recorded. Furthermore, many people work collaboratively on these platforms. Thus, these platforms are very suitable for data hiding applications, such as covert communication, secret data keeping, etc. It is desired to propose a new method which is useful for covert communication or secure keeping of secret messages on collaborative writing platforms. However, the above-mentioned data hiding methods via text [29]-[38] in Section 1.3.3 can only be applied to documents with single authors and single revision versions, meaning that they are not suitable for hiding data on collaborative writing platforms. Therefore, the goal of this study is to propose a new data hiding method which can hide data into documents created on *collaborative writing platforms*. In more detail, a new data hiding method is proposed, which simulates a collaborative writing process to generate a fake document, consisting of an article and its revision history, as a camouflage for message bit embedding. As shown in Figure 4.1, with the input of an article and a secret message, the proposed method utilizes multiple virtual authors to collaboratively revise the article, generating artificially a history of earlier revisions of the article according to the secret message. An ordinary reader will consider the resulting stego-document as a normal collaborative writing output, and cannot realize the existence of the secret message hidden in the document.

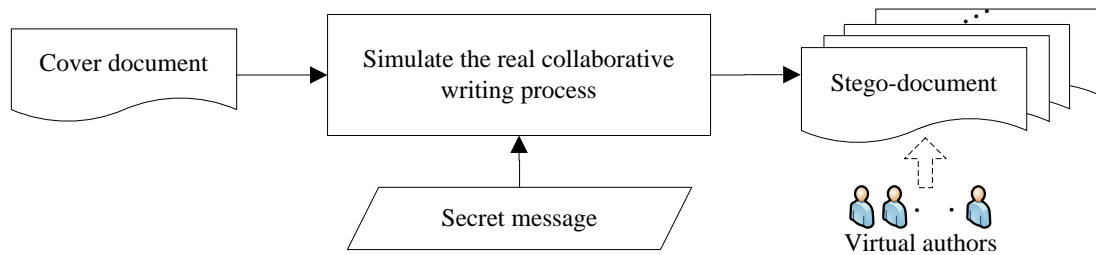


Figure 4.1. Basic idea of proposed method that generates a revision history of a stego-document as a camouflage for data hiding.

Moreover, the previously-mentioned linguistic methods [35]-[38] in Section 1.3.3 use written natural languages to generate stego-documents and can produce more innocuous stego-texts than other data hiding methods, but an issue common to them is how to find a nature way for simulating the writing process and how to obtain large-volume written data *automatically*. Hence, another goal of this study is to find a nature way to generate the revision history and to obtain large-volume collaborative writing data *automatically*. In recent years, some researches have been conducted to analyze the revision history data of Wikipedia articles for various natural language processing applications [57]-[63], such as spelling corrections, reformulations, text summarization, user edits classification, multilingual content synchronization, etc. In addition to being useful for these applications, the collaboratively written data in Wikipedia are also very suitable, as found in this study, for simulating the collaborative writing process for the purpose of data hiding since it is the largest collaborative writing platform nowadays.

In [64], Liu and Tsai proposed a data hiding method via Microsoft Word documents by the use of the change tracking function, which embeds a secret message by mimicking a pre-draft document written by an author with an inferior writing skill and encoding the secret message by choices of degenerations in the writing. Although they used three databases for degenerations, the sizes of them are quite small when compared to that of the database constructed from Wikipedia which we make use for data embedding in this study. It is noted by the way that a data hiding method can, as well known, embed more bits by making use of a larger database. Furthermore, in [64] a stego-document is generated by only two virtual persons and the change tracking data are made by the one with a better writing skill. This scenario is *insufficient* for simulating a normal collaborative writing process. Therefore, in this study we propose

a new *framework* that uses the revision-history data from Wikipedia and simulates real collaborative writing processes to hide secret messages. Four characteristics of collaborative writing processes are analyzed and utilized for message hiding, including the author of each revision, the number of corrected word sequences, the content of the corrected word sequences, and the word sequences replacing the corrected ones. The proposed method is useful for covert communication or secure keeping of secret messages on collaborative writing platforms.

4.2 Basic Idea of Proposed Method

Collaborative writing means an activity involving more than one author to create an article cooperatively on a common platform. The purposes of establishing a *collaborative writing platform* includes knowledge sharing, project management, data keeping, etc. Many collaborative writing platforms are available, such as Google Drive, Office Web Apps, Wikipedia, etc., which record revisions generated during the collaborative writing process. In general, the recorded information of a revision includes: 1) the author of the revision, 2) the time the revision was made, and 3) the content of the revision. For example, Figure 4.2 shows a screenshot of the revision history of an article about computer vision on Wikipedia.



Figure 4.2. A screenshot of the revision history of an article about computer vision on Wikipedia.

To achieve the goal of creating camouflage revisions in collaborative writing for message hiding in this study, we analyze the existing revision-history data of articles on Wikipedia, which is the largest collaborative writing platform on the Internet currently in the world. The aim is to get real and large *collaborative writing data* contributed by people all over the world and use them to create more realistic revision

histories to enhance the resulting effect of data embedding. However, since the collaborative writing process is very complicated, it is hard to find a unified model to simulate it. Many different types of modifications may be made during the collaborative writing process [57], [59], such as error corrections, paraphrasing, factual edits, etc. Moreover, different languages usually require different models to represent due to their distinctive grammatical structures. Therefore, in order to get useful collaborative writing data *automatically* from the revision history data on Wikipedia without building models manually and to generalize a method that can be applied to multiple languages, we assume that only *word sequence corrections* occur during a revision. Some characteristics in collaborative writing based on this assumption for data embedding are identified, which will be discussed in the following. It is noted that various text articles, not only in English but also in other languages, can be utilized as cover media in this study.

The revision history of each article in Wikipedia is stored in a database, and one can recover any previous revision version of the article by an interface provided on the site. As an illustration, Figure 4.3 shows a screenshot of two consecutive revisions of an article about computer vision on Wikipedia. For this study, we have collected a large set of revision-history data from Wikipedia, and in the proposed method we mine this set to get useful information about word usages in the revisions. Then, we use the acquired information to simulate a collaborative writing process, starting from a *cover article*; and generate a *stego-article* with a sequence of revisions according to the secret message and a secret key. The resulting *stego-document*, including the stego-article and the revision history, looks like a work created by a group of real authors, achieving an effect of camouflage. In contrast, we call the original article with an initially-empty history a *cover document* in the sequel.

More specifically, the proposed method includes three main phases as shown in Figure 4.4: 1) construction of a collaborative-writing database; 2) secret message embedding; and 3) secret message extraction. In the first phase, a large number of articles acquired from Wikipedia are analyzed and useful collaboratively written data about word usages are mined using a natural language processing technique. The mined data then are used to construct a database, called the *collaborative writing database*, denoted as DB_{cw} subsequently. In the second phase, with the input of a cover document, a secret message, and a secret key, a stego-document with a *fake* revision history is generated by simulating a real collaborative writing process using

DB_{cw} . The revisions in the history are supposed to be made by multiple virtual authors; and the following *characteristics* of each revision are decided by the secret message: 1) the author of the revision; 2) the number of *changed word sequences* of the revision; 3) the changed word sequences in the revision; and 4) the word sequences selected from the collaborative writing database DB_{cw} , which replace those of 3), called the *replacing word sequences* in the sequel. And in the third phase, an authorized person who has the secret key can extract the secret message from the stego-document, while those who do not have the key cannot do so. They even could not realize the existence of the secret message because the secret message is disguised as the revision history in the stego-document. Note that the second and third phases can be applied on *any* collaborative writing platforms, not just on Wikipedia; Wikipedia is merely utilized in the first phase to construct the collaborative writing database DB_{cw} in this study.

Computer vision: Difference between revisions

From Wikipedia, the free encyclopedia

[Revision as of 11:31, 12 January 2012 \(edit\)](#)
[Revision as of 12:53, 12 January 2012 \(edit\) \(undo\)](#)

Line 27: Line 27:

[[Physics]] is another field that is closely related to computer vision. **Computer** vision systems rely on [[image sensors]], which detect electromagnetic radiation which is typically in the form of either visible or infra-red light. The sensors are designed using [[solid-state physics]]. The process by which light interacts with surfaces is explained using physics. Physics explains the behavior of [[optics]] which are a core part of most imaging systems. Sophisticated [[image sensors]] even require quantum mechanics to provide a complete understanding of the image formation process. Also, various measurement problems in physics can be addressed using computer vision, for example motion in fluids.

+

[[Physics]] is another field that is closely related to computer vision. **Most** computer vision systems rely on [[image sensors]], which detect electromagnetic radiation which is typically in the form of either visible or infra-red light. The sensors are designed using [[solid-state physics]]. The process by which light interacts with surfaces is explained using physics. Physics explains the behavior of [[optics]] which are a core part of most imaging systems. Sophisticated [[image sensors]] even require quantum mechanics to provide a complete understanding of the image formation process. Also, various measurement problems in physics can be addressed using computer vision, for example motion in fluids.

Figure 4.3. A screenshot of two consecutive revisions of an article about computer vision on Wikipedia.

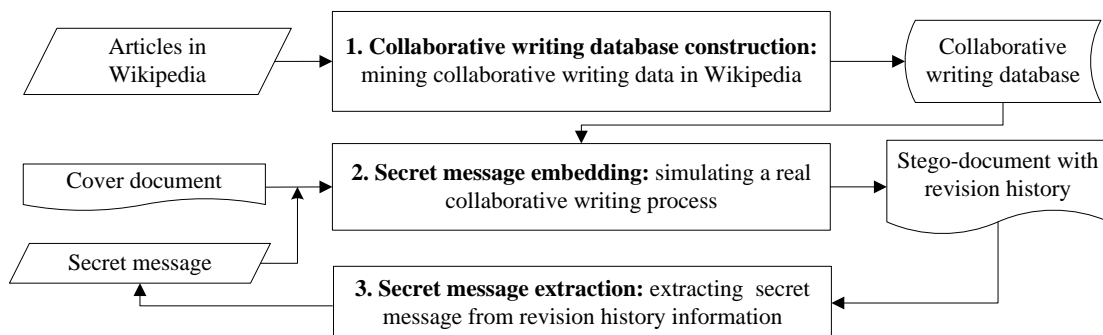


Figure 4.4. Flow diagram of the proposed method.

4.3 Data Hiding via Revision History

In this section, the details of the proposed method for using the analyzed characteristics of collaborative writing to hide secret messages are described in the following, where the first part is collaborative writing database construction, the second part is secret message embedding, and the final part is secret message extraction.

4.3.1 Collaborative writing database construction

To construct the aforementioned collaborative writing database DB_{cw} , we try to mine the revision data collected from Wikipedia. There were about 4.2 million articles in the English Wikipedia in May 2013, which is a very large knowledge repository; therefore, it is suitable to use it as a source for constructing the database DB_{cw} desired in the study. Specifically, at first we downloaded part of the English Wikipedia XML dump with the complete revision histories of all the articles on August 3, 2011. Then, we mine the useful collaborative writing data from the downloaded data set under the assumption that only *word sequence corrections* will occur during a revision.

As illustrated in Figure 4.5, each downloaded article P has a set of revisions $\{D_0, D_1, \dots, D_n\}$ in its revision history, where a newer revision D_i has a smaller index i with D_0 being the latest version of the article. For every two consecutive revisions D_i and D_{i-1} , we find all the *correction pairs* between D_i and D_{i-1} , each denoted as $\langle s_j, s_j' \rangle$, where s_j is a word sequence in revision D_i and was corrected to become another, namely, s_j' , by the author of revision D_{i-1} . Then, we collect all correction pairs so found to construct the database DB_{cw} . For example, assume $D_i =$ “National Chia Tang University” and $D_{i-1} =$ “National Chiao Tung University” as shown in Figure 4.6. Then, the correction pair $\langle s_{11}, s_{11}' \rangle = \langle$ “Chia Tang”, “Chiao Tung” \rangle is generated and included into DB_{cw} .

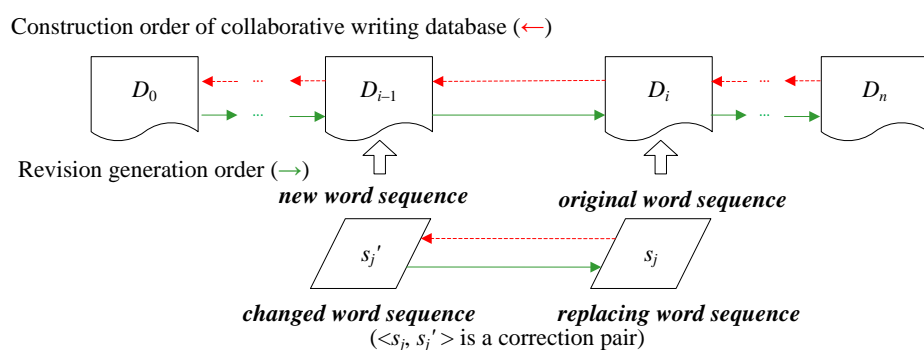
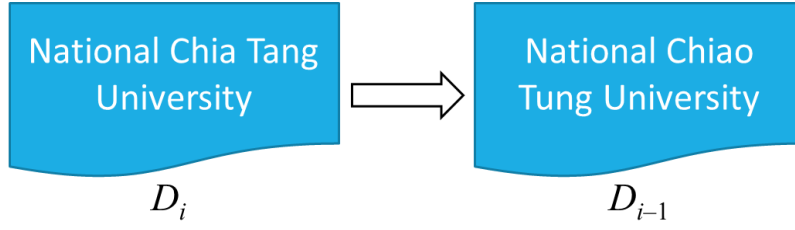


Figure 4.5. Illustration of used terms and notations.



→ Correction pair $\langle s_1, s_1' \rangle = \langle \text{"Chia Tang"}, \text{"Chiao Tung"} \rangle$

Figure 4.6. An example of found correction pairs between D_i and D_{i-1}

Moreover, about the properties of correction pairs, it was observed that if the *context* of a word sequence s_j in revision D_i is the same as that of a word sequence s_j' in revision D_{i-1} (that is, if the preceding word of s_j is the same as that of s_j' and the succeeding word of s_j is the same as that of s_j' as well), then $\langle s_j, s_j' \rangle$ is a correction pair. Accordingly, a novel algorithm is proposed in this study for finding *automatically* all of the correction pairs between every two consecutive revisions for inclusion in DB_{cw} . The algorithm is an extension of the longest common subsequence (LCS) algorithm [65]. The details are described in Algorithm 4.1.

Algorithm 4.1. Finding correction pairs.

Input: two consecutive revisions D_i and D_{i-1} in the revision history of an article P .

Output: the correction pairs between D_i and D_{i-1} .

Steps:

Stage 1 — finding the longest common subsequence.

Step 1. (*Splitting revisions into word sets*) Split D_i and D_{i-1} into two sets of words, $W = \{w_1, w_2, \dots, w_n\}$ and $W' = \{w_1', w_2', \dots, w_m'\}$, respectively.

Step 2. (*Constructing a counting table by dynamic programming*) Construct an $n \times m$ counting table T to record the lengths of the common subsequences of W and W' as follows.

- (a) Initialize all elements in table T to be zero.
- (b) Compute the values of table T from the upper left and denote the currently-processed entry in T by $T(x, y)$ with $x = 1$ and $y = 1$ initially.
- (c) If the content of w_x is identical to that of $w_{y'}$, then let $T(x, y) = T(x - 1, y - 1) + 1$; else, let $T(x, y) = \max(T(x - 1, y), T(x, y - 1))$.

- (d) If x is not larger than n , then let $x = x + 1$ and go to Step 2c); else, if y is not larger than m , then let $x = 1$ and $y = y + 1$ and go to Step 2c); else, regard table T as being filled up and continue.

Step 3. (*Finding the longest common subsequence*) Apply a backtracking procedure to table T , starting from $T(m, n)$, to find the longest common subsequence $L = \{l_1, l_2, \dots, l_t\}$, where each element l_i in L is a word common to W and W' .

Stage 2 — finding the correction pairs.

Step 4. (*Finding the correction pairs*) Starting from the first element l_1 of L with the currently-processed element in L being denoted by l_p , find the correction pairs as follows.

- (a) If the word sequence s_j in D_i with its preceding and succeeding words being l_p and l_{p+1} , respectively, is not empty and if the word sequence s'_j in D_{i-1} with the same context condition is not empty, either, then take $\langle s_j, s'_j \rangle$ as a correction pair.
- (b) Increment p by 1 and go to Step 4) until $p > t$.

We run Algorithm 4.1 for every two consecutive revisions of all the articles downloaded from Wikipedia to obtain a large set of correction pairs and write them into the database DB_{cw} . Furthermore, we count the total number N_{cp} of times that each correction pair CP is so obtained, and call the number N_{cp} the *correction count* of CP . The correction counts are also kept in the database DB_{cw} for use in the proposed data hiding process.

As a summary, we use a *record* in the database DB_{cw} to keep the following information about a correction pair $\langle s_j, s'_j \rangle$: 1) an *original* word sequence s_j ; 2) a *new* word sequence s'_j ; and 3) the correction count N_{cp} of the pair. Moreover, we define a *chosen set* of a word sequence s' in DB_{cw} to be the one which include all the correction pairs $\langle s, s' \rangle$ with s' as their identical new word sequences. For example, Table 4.3 (shown in Section 4.4) shows a chosen set of the word sequence “such as.”

4.3.2 Secret message embedding

In the phase of message embedding with a cover document D_0 as the input, the proposed system is designed to generate a stego-document D' with consecutive revisions $\{D_0, D_1, D_2, \dots, D_n\}$ by producing a *previous* revision D_i from the *current* revision D_{i-1} repeatedly until the entire message is embedded, as shown in Figure 4.5

where the direction of revision generation is indicated by the green arrows. The stego-document D' including the revision history $\{D_0, D_1, D_2, \dots, D_n\}$ then is kept on a collaborative writing platform, which may be Wikipedia or others. To simulate a collaborative writing process more realistically, we utilize the four aforementioned characteristics of revisions to “hide” the message bits into the revisions sequentially: 1) the author of the previous revision D_i , 2) the number of changed word sequences in the current revision D_{i-1} , 3) the changed word sequences in the current revision D_{i-1} , and 4) the replacing word sequences in the previous revision D_i , as described in the following.

(1) Encoding the authors of revisions for data hiding.

We encode the authors of revisions to hide message bits in the proposed method. For this, at first we select a group of simulated authors, with each author being assigned a unique code a , called *author a* . Then, if the message bits to be embedded form a code a_j , then we assign author a_j to the previous revision D_i as its author to achieve embedding of message bits a_j into D_i . For example, assume that four authors are selected and each is assigned a unique code a as shown in Figure 4.7, respectively. If the message bits a_j to be embedded is “01,” then Jessy with author code “01” is selected to be the author of the revision D_i . Moreover, every revision of D_0 through D_n will be assigned an author according to the corresponding message bits, and so an author can be assigned to conduct more than one revision or reversely no revision in the generated revisions.

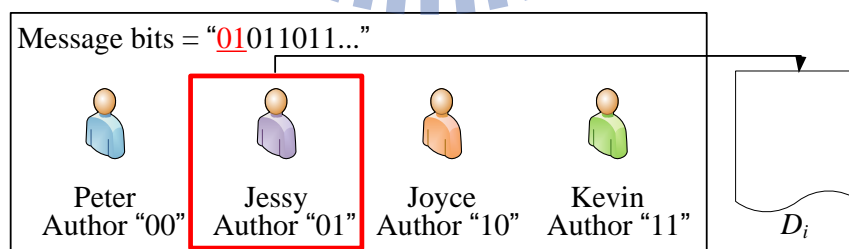


Figure 4.7. Illustration of encoding authors of revisions for data hiding.

(2) Using the number of changed word sequences for data hiding.

In the process of generating the previous revision D_i from the current one D_{i-1} , we select some word sequences in D_{i-1} and changed them into other ones in D_i . It is

desired to use as well the *number* N_g of word sequences changed in this process as a *message-bit carrier*.

To implement this aim, at first we set on the magnitude of N_g a limit N_c taken to be the *maximum* allowed number of word sequences in D_{i-1} that can be changed to yield D_i . This limitation makes the simulated step of revising D_{i-1} to become D_i look more realistic because usually not very many words are corrected in a single revision. Next, we scan the word sequences in the text of the current revision D_{i-1} sequentially and search DB_{cw} to find all the correction pairs $\langle s_j, s_j' \rangle$ with s_j' in D_{i-1} . Then, we collect all s_j' in these pairs as a set Q_r , which we call the *candidate set of word sequences for changes in D_{i-1}* . Finally, we select N_g word sequences in Q_r to form a set Q_c such that the binary version of the number N_g is just the current message bits to be embedded.

But for this process of using N_g as a message-bit carrier to be feasible, several problems must be solved beforehand, including: 1) the *dependency problem*, 2) the *selection problem*, 3) the *consecutiveness problem*, and 4) the *encoding problem*, as described in the following.

(2.1) The dependency problem.

We say that two word sequences in D_{i-1} are *dependent* if some identical words appear in both of them, and changing word sequences with this property in D_{i-1} will cause conflicts, leading to a *dependency problem* which we explain by an example as follows.

As shown in Figure 4.8(a), $D_{i-1} =$ “you are not wrong, who deem that my days have been a dream” and Q_r includes 11 word sequences denoted as q_1 through q_{11} , respectively. From Figure 4.8(a) we can see that the word sequences q_2 , q_3 , and q_5 in Q_r are dependent on the word sequence q_4 because the intersection of each of the former three with the latter one is non-empty. If we correct $q_4 =$ “are not wrong” in D_{i-1} to be another, say “is right,” then the dependent word sequences q_2 , q_3 , and q_5 in D_{i-1} *cannot* be selected and changed anymore because they include word sequences in q_4 which have already been changed and disappeared. That is, any part of a changed word sequence cannot be changed again; otherwise, a dependency problem will occur.

To avoid this problem in creating D_i from D_{i-1} , we propose a two-step scheme: 1) decompose Q_r into a set of lists, $I = \{I_1, I_2, \dots, I_u\}$, with each list I_i including a group of mutually dependent word sequences (i.e., with every word sequence in each I_i

being *dependent* on another in the same list) and every two word sequences in two different lists, respectively, in I being *independent* of each other; and 2) select only word sequences from different lists in set I and change them to construct a new revision. The details to implement the first step is described in Algorithm 4.2. After applying the first step on the set Q_r as shown in Figure 4.8(b), it will be transformed into $I = \{(q_1), (q_2, q_3, q_4, q_5), (q_6), (q_7), (q_8), (q_9, q_{10}), (q_{11})\}$ where each pair of parentheses encloses a list of mutually dependent word sequences. With I ready, we can now select word sequences from distinct lists I_i in it, such as $q_1, q_2, q_6,$ and $q_9,$ to simulate changes of word sequences in revision D_{i-1} without causing the dependency problem.

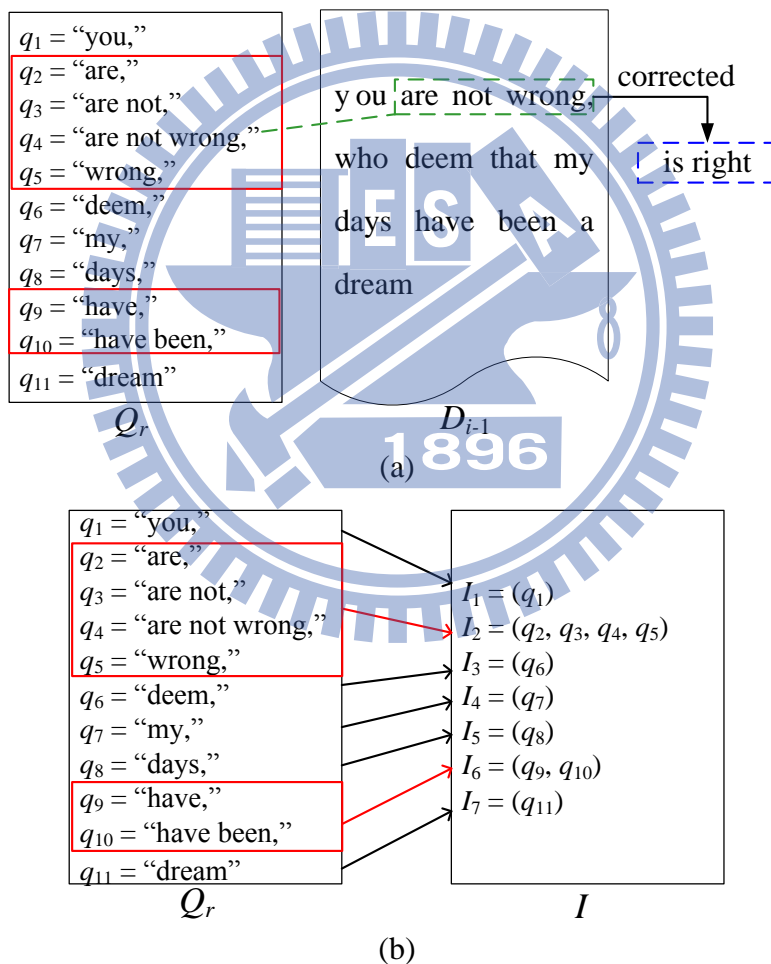


Figure 4.8. Illustration of the dependency problem. (a) Revision D_{i-1} and candidate set Q_r where the dependent word sequences are surrounded by red squares. (b) Set I that corresponds to the set Q_r for solving the dependency problem.

(2.2) The selection problem.

It is desired to select word sequences for use in the simulated revisions according to their *usage frequencies* in DB_{cw} , so that a more frequently-corrected word sequence has a larger probability to be selected, forging a more realistic revision. For this aim, following [32], [64], we adopt the Huffman coding technique to create Huffman codes uniquely for the word sequences in Q_r according to their usage frequencies, and select word sequences with their codes identical to the message bits to be embedded. Specifically, according to a property of Huffman coding, the lengths of the resulting Huffman codes of word sequences are in reverse proportion to the usage frequencies of the word sequences. So a word sequence with a shorter Huffman code will have a larger probability to be selected, which can be computed as $(1/2)^L$ where L denotes the number of bits of the code. That is, the use of Huffman coding indeed can achieve the aim of selecting word sequences in favor of those which are more frequently corrected in real cases.

But a problem arises here — after we select one word sequence q_y in this way, q_y cannot be used in the revision *again* for encoding an identical succeeding code in the message because q_y has already been changed into another word sequence, causing a problem which we call the *selection problem*. This problem comes partially from the *unique decidability* property of Huffman coding. To illustrate this problem, for the previous example as shown in Figure 4.8 again, the Huffman codes for word sequences q_1 through q_{11} are shown in Figure 4.9(a), and the message bit sequence to be embedded currently is “100100...” with the first six bits being just two repetitions of the code “100.” For this, at first we select word sequence q_4 and change it into another in the revision because the first three message bits to be embedded, “100,” are just the code for q_4 (indicated by red color). After this, the next three message bits to be embedded are *again* the code “100” (the blue color of message bits in Figure 4.9(a)); however, the corresponding word sequence q_4 cannot be selected any further because it has already been changed in the current revision version, and other word sequences cannot be selected, either, because their codes are not the same as the current message bits “100” to be embedded.

Word sequence	Huffman code
q_1	1110
q_2	0001
q_3	1010
q_4	100
q_5	001
q_6	110
q_7	011
q_8	1111
q_9	0000
q_{10}	1011
q_{11}	010

Message bits =
 100100...

(a)

Group	Word sequence	Huffman code
G_1	q_1	00
	q_2	100
	q_3	101
	q_4	11
	q_5	01
G_2	q_6	0
	q_7	11
	q_8	10
G_3	q_9	10
	q_{10}	11
	q_{11}	0

Message bits =
 100100...

(b)

Figure 4.9. Illustration of the selection problem. (a) Huffman codes for the word sequences and the message bits that are encountered in the selection problem. (b) Dividing of the word sequences into groups to solve the selection problem.

To solve this selection problem, suppose that based on the use of a key, we assign randomly the word sequences in Q_r consecutively into N_g groups G_1 through G_{N_g} , each group including multiple, but distinct, word sequences, where N_g is the number of word sequences changed in D_{i-1} . Then, starting from group G_1 , we apply Huffman coding to assign codes to all word sequences in the currently-processed group G_k according to their *usage frequencies*, and select a word sequence in G_k with its assigned code identical to the leading message bits for use in the revision. We apply this step repetitively until all groups are processed. In this process, Huffman coding is applied to *each* G_k with word sequences distinct from those in the other groups, so that the selection problem of choosing a word *twice* to change due to code repetition in the message will not happen anymore. For example, as shown in Figure

4.9(b), Q_r is divided into three groups: G_1 , G_2 , and G_3 , represented by red, blue, and green colors, respectively. Starting from G_1 , we assign Huffman codes to the elements in each group as shown in Figure 4.9(b). Then, q_2 will be selected because the code of q_2 is the same as the first three bits “100...” of the message to be embedded. Then, next in G_2 , q_8 will be selected because the message bits to be embedded are currently “100...” Finally, q_{11} in G_3 will be selected because the current message bits to be embedded are “0...” In this way, the previous problem of being unable to embed the repetitive code “100” is solved automatically. In short, by decomposing randomly the candidate set Q_r of word sequences for changes into groups and representing each group by a Huffman code, we can embed message bits sequentially by changing only one word sequence in each group without causing the selection problem.

However, the above process is insufficient; it must be modified in such a way that word sequences which have *mutual dependency relations* are divided into an identical group in order to avoid the dependency problem as discussed previously. For this aim, instead of decomposing the word sequences in Q_r directly into random groups as mentioned previously, we divide randomly the *mutually-independent* list elements of I into N_g groups, where each group is denoted by G_{Tk} . Then, we take out all the word sequences in the lists in each G_{Tk} to form a new group of word sequences, denoted as G_k , resulting again in N_g groups of word sequences. For instance, for the previous example as shown in Figure 4.8, let $N_g = 2$ and suppose that the list elements of I are decomposed randomly into two groups: $G_{T1} = \{I_1, I_2, I_3, I_4\}$ and $G_{T2} = \{I_5, I_6, I_7\}$. Then, this procedure will yield the two groups of $G_1 = \{q_1, \dots, q_7\}$ and $G_2 = \{q_8, q_9, q_{10}, q_{11}\}$.

(2.3) The consecutiveness problem.

As shown in Figure 4.10(a), for example, the word sequence “increase in” in revision D_{i-1} is seen to become “improve themselves” in revision D_i . This effect comes from two changes made during message embedding: the word sequence “increase” in D_{i-1} was changed to be “improve” in D_i ; and the word sequence “in” in D_{i-1} was changed to be “themselves” in D_i . However, because of the *consecutiveness* of the two words “improve” and “themselves” in D_i , the two changes might be considered *as a single one* during secret message extraction, i.e., the word sequence “increase in” in D_{i-1} might be regarded to have been changed to be “improve themselves” in D_i . This ambiguity causes a problem, namely, we cannot know whether a change from a word sequence in D_{i-1} to be another in D_i is from one group

or two, or equivalently, we cannot know the *true* number N_g of changed word sequences in D_{i-1} , so that we cannot extract later the embedded messages bits correctly. We call this difficulty in message extraction a *consecutiveness problem*.

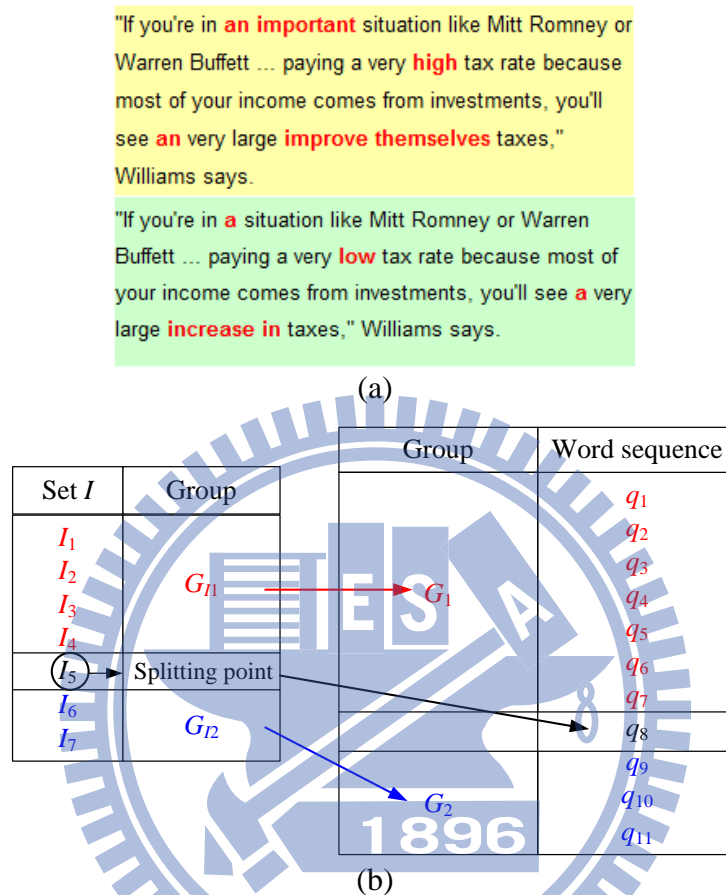


Figure 4.10. Illustration of the consecutiveness problem. (a) An example for illustration of the consecutiveness problem. (b) Choosing splitting points randomly to solve the consecutiveness problem.

Obviously, word sequences in different groups must be made *non-consecutive* in order to solve the problem. For this aim, the previously-mentioned solution to the selection problem is modified further. Specifically, by the use of a key again we choose randomly $N_g - 1$ lists, say I_1, I_2, \dots, I_n (with $N = N_g - 1$), of the set I for use as *splitting points* to divide I into N_g groups with I_i through I_n not included in any of the N_g groups. For instance, let $N_g = 2$ for the previous example as shown in Figure 4.8 and the number of splitting points may be computed accordingly to be $N_g - 1 = 1$. Consequently, as shown in Figure 4.10(b), we choose a splitting point, say I_5 , to divide the set I into two groups: G_{11} and G_{12} , both not including I_5 . The final groups of

word sequences then become: $G_1 = \{q_1, q_2, q_3, q_4, q_5, q_6, q_7\}$ and $G_2 = \{q_9, q_{10}, q_{11}\}$. Because of the existence of the splitting point $I_5 = (q_8)$, groups G_1 and G_2 are non-consecutive, and accordingly uses of them for creating word sequence changes in revisions will now cause no consecutiveness problem.

(2.4) The encoding problem.

The issue up to now is how to determine the aforementioned number N_g of word sequences to be changed in D_{i-1} . Although a limit N_c is set for N_g , the maximum number N_m of word sequences that can be selected in D_{i-1} may even be smaller than N_c . Therefore, we must compute N_m first before we can embed message bits according to the number N_g . After N_m is decided, N_g may then be taken to be a number not larger than N_m . The actual value of N_g is decided by the leading secret message bits, say n_m ones. Consequently, we may assume that N_m satisfies the two constraints of 1) $N_m = 2^{n_m}$ and 2) $1 \leq N_m \leq N_c$, where n_m is a positive integer. In addition, in order to embed message bits by selecting a word sequence from a group G_k , the number of elements in G_k should not be smaller than two so as to embed at least one message bit by Huffman coding; hence, each group G_k mentioned previously should be created to include at least two elements of I . Accordingly, the maximum number N_m of word sequences to be changed in D_{i-1} can be figured out to satisfy the following formula:

$$\lfloor [N_I - (N_m - 1)] / N_m \rfloor \geq 2, \quad (11)$$

where N_I is the number of elements in set I and $N_m - 1$ represents the aforementioned number of chosen splitting points. The inequality (11) can be reduced to

$$N_m \leq (N_I + 1) / 3. \quad (12)$$

Accordingly, we can compute N_m by the following rule:

if $(N_I + 1) / 3 > N_c$, set $N_m = N_c$;

$$\text{if } 1 \leq (N_I + 1) / 3 \leq N_c, \text{ set } N_m = 2^{\lfloor \log_2((N_I + 1) / 3) \rfloor}. \quad (13)$$

Furthermore, the content of D_{i-1} might be too little for N_m to be decided by Eq. (13). In that case, we abandon the original cover document D_0 from which D_{i-1} is generated, and use another longer cover document as the input. After the value of N_m is computed, we can then use the leading n_m bits of the message to decide the number N_g of changed word sequences in D_{i-1} by two steps: 1) express the first n_m message bits as a decimal number; and 2) increment the decimal number by one. The second step is

required to handle the case that the first n_m message bits are all zeros, which leads to the undesired result of no word sequence being changed in the current revision. In this way, N_g becomes really a carrier of n_m message bits. For example, the number of elements of the set I for the previously-mentioned example as shown in Figure 4.8 is $N_I = 7$. Let $N_c = 4$. Because $(N_I + 1)/3 = (7 + 1)/3 \approx 2.67 \leq 4 = N_c$, N_m is computed to be $2^{\lceil \log_2(7+1)/3 \rceil} = 2^1$ according to Eq. (13). So, $n_m = \log_2 N_m = 1$. And if the secret message is “101001...,” then the number N_g of changed word sequences should be taken, according to the above two steps, to be $N_g = (\underline{1})_2 + (1)_{10} = 2$ because the first bit of the secret message is “1.”

(3) Encoding the changed word sequences in the current revision for data hiding.

According to the previous discussions, we may assume that we have computed the number N_g of word sequences which should be changed in the current revision D_{i-1} according to the first n_m bits of the secret message, and that we have classified the available word sequences in Q_i into N_g groups, where each group G_k includes at least two word sequences and all word sequences in G_k are encoded by Huffman coding according to their usage frequencies. Specifically, the usage frequency of a word sequence s_j' is taken to be the summation of the correction counts of all the correction pairs in the chosen set of s_j' , which have s_j' as their common new word sequence. Then, starting from G_1 , we may select from each group G_k one word sequence with a Huffman code identical to the leading bits of the message to be embedded, achieving the goal of data hiding via changing word sequences in D_{i-1} .

For example, assume that the usage frequencies of the word sequences in group G_2 as shown in Figure 4.10(b) are: $q_9 = 100$, $q_{10} = 50$, and $q_{11} = 150$; and the message is “10100...” Then, the Huffman codes assigned to q_9 , q_{10} , and q_{11} are “01,” “00,” and “1,” respectively; and so we select q_{11} to hide the first bit “1” of the message because the code of q_{11} is “1.”

(4) Encoding the replacing word sequences in the previous revision for data hiding.

Symmetrically, we may use as well the replacing word sequences in D_i to embed message data, where each replacing word sequence s_j in D_i corresponds to a changed word sequence s_j' in D_{i-1} , forming a correction pair $\langle s_j, s_j' \rangle$. Specifically, recall that for each s_j' , we can find a chosen set of correction pairs from DB_{cw} . From this set, we

can collect all the *original* word sequences of the correction pairs as another set Q_c' , with each word sequence in Q_c' being appropriate for use as the replacing word sequence s_j . Let $Q_c' = \{s_1, s_2, \dots, s_w\}$. Then, to carry out message data hiding, we encode all s_j in Q_c' by Huffman coding according to their usage frequencies as well, and choose the one with its code identical to the leading message bits for use as the word sequence s_j replacing s_j' . Here the usage frequency of each s_j is the correction count of the correction pair $\langle s_j, s_j' \rangle$. For example, Table 4.3 shows the chosen set of the word sequence “such as” with all included original word sequences already assigned Huffman codes according to their usage frequencies. Based on the table, if the message to be embedded currently is “01001001...,” then we change the word sequence “such as” in the current revision D_{i-1} to be the word sequence “for example” in the previous revision D_i because the Huffman code for “for example,” namely, 0100, is the same as the first four bits of the secret message.

(5) Secret message embedding algorithm.

As a summary, we have demonstrated the usability of the aforementioned four characteristics of revisions for data hiding. Therefore, we can generate a stego-document with a forged revision history which looks like a realistic work written by people collaboratively. The details of the proposed message embedding process are described in Algorithm 4.2 below.

Algorithm 4.2. Secret message embedding.

Input: a cover document D_0 , a binary message M of length t , a secret key K , and the collaborative writing database DB_{cw} constructed by Algorithm 4.1.

Output: a stego-document D' with a revision history $\{D_0, D_1, D_2, \dots, D_n\}$.

Steps:

Stage 1 — message preparation and parameter determination.

Step 1. (*Message composition*) Affix an s -bit binary version of the message length t to the beginning of M to compose a new binary message M' , where the number s of bits for representing t is agreed by the sender and the receiver beforehand.

Step 2. (*Message encryption*) Randomize M' to yield a new binary message M'' using the key K .

Step 3. (*Parameter determination*) Use K again to decide randomly *both* an integer N_a for use as the number of authors *and* another integer N_c for use as the limit on the number N_g of word sequences to be changed in every revision.

Step 4. (*Author encoding*) Create N_a authors to form an author list I_a , and assign a unique n_a -bit code to each author in I_a .

Stage 2 — message embedding and revision generation.

Step 5. (*Message embedding and revision generation*) Generate the *previous revision* D_i from the current revision D_{i-1} repeatedly by the following procedure until all bits in M'' are embedded, where $i = 1$ initially.

Stage 2.1 — embedding data via author encoding.

(a) (*Embedding bits by an author code*) Choose for D_i the author a_j from the author list I_a with the n_a -bit code assigned to author a_j being identical to the leading n_a bits of M'' ; and remove these n_a bits from M'' .

Stage 2.2 — embedding data using the number of changed word sequences in the current revision.

(b) (*Finding the candidate word sequences for changes in D_{i-1}*) Create the candidate set Q_r of word sequences for changes in D_{i-1} by the following steps.

(i) Take in order an unprocessed word w in revision D_{i-1} , and set the currently processed word sequence q as w initially.

(ii) Check if q matches some leading words or all of the words in the *new* word sequence of any correction pair in DB_{cw} —

if so, do the following two steps:

(A) if q is identical to the entire new word sequence, then add q to Q_r and continue;

(B) create a new word sequence, still denoted as q , by concatenating the old q with the word q_r which is to the right of q in D_{i-1} , and go to Step 5.b.ii;

else, continue.

(iii) If there still exists any unprocessed word in D_{i-1} , go to Step 5.b.i; otherwise, continue.

(c) (*Finding independent word sequence lists in Q_r*) Decompose Q_r to form a set $I = \{I_1, I_2, \dots, I_u\}$ of word-sequence lists by the following steps, where

each I_i is a list of mutually dependent word sequences and every two lists are independent.

- (i) Take each word sequence in Q_r as a list initially.
- (ii) Check the *ordered* word sequences in Q_r one by one sequentially:
 - if the currently-checked word sequence q_s and its previous one q_t include some common consecutive words, then regard q_s as dependent on q_t , add q_s into the list of q_t in I , and eliminate the list of q_s itself in I ; else, keep the list of q_s in I .
- (d) (*Deciding the number of word sequences to be changed as a message-bit carrier*) Decide the number N_g of groups into which the set Q_r is to be divided by the following steps.
 - (i) Compute the maximum number N_m of word sequences to be changed in revision D_{i-1} by inequalities Eq. (13) described previously, and compute n_m as $\log_2 N_m$.
 - (ii) Decide the number N_g as the decimal value of the first n_m bits of message M'' plus one, and remove these n_m bits from M'' .

Stage 2.3 — embedding data via the word sequences changed in the current revision.

- (e) (*Choosing splitting points*) Choose randomly $N_g - 1$ elements of I as splitting points using the key K .
- (f) (*Classifying the word sequences into independent sets*) Divide the elements of I into N_g groups, G_{11} through G_{1N_g} , by the splitting points, and take out all the word sequences in the lists in each G_{1k} to form a new group of word sequences, denoted as G_k , resulting in N_g groups of word sequences, G_1 through G_{N_g} .
- (g) (*Choosing word sequences to change for message-bit embedding*) For each group G_k with $k = 1$ initially, encode its word sequences by Huffman coding according to their usage frequencies, choose and *mark* the word sequence $s_{j'}$ in G_k with its code *matching* the leading message bits of M'' as the word sequence to be changed to create the previous revision D_i , and remove the *matched* leading bits from M'' .

Stage 2.4 — embedding message data via replacing the word sequences in the previous revision.

- (h) (*Finding the chosen set*) Find the chosen set of the *previously marked* word sequence s_j' from the correction pairs kept in DB_{cw} , and collect all the original word sequences in the chosen set as a set Q_c' .
- (i) (*Choosing the original word sequence for use in replacement*) Encode the word sequences in Q_c' by Huffman coding according to their usage frequencies, choose the word sequence s_j in Q_c' with its code *matching* the leading message bits of M'' , and remove the *matched* leading bits from M'' .
- (j) (*Conducting word sequence correction*) Replace the word sequence s_j' in D_{i-1} with s_j .
- (k) (*End of looping*) Increase k by one and go to Step 5.g until $k > N_g$.
- (l) (*Revision generation*) Take the final revised content of D_{i-1} as the desired *previous revision* D_i .

Step 6. If message M'' is not exhausted, then repeat the above process to generate more revisions until so; collect the final article and the history of all the revisions D_0 through D_n as a stego-document D' ; and take D' as the output for use on a pre-selected collaborative writing platform.

4.3.3 Secret message extraction

We can extract the secret message in the stego-document by a reverse version of the message embedding process described by Algorithm 4.2. The details are described as an algorithm in the following.

Algorithm 4.3. Secret message extraction.

Input: a stego-document D' including revision history $\{D_0, D_1, D_2, \dots, D_n\}$ kept on a pre-selected collaborative writing platform, the secret key K used in Algorithm 4.2, and the database DB_{cw} constructed by Algorithm 4.1.

Output: a binary message M of length t .

Stage 1 — parameter determination.

Step 1. Use key K to decide two parameters N_a and N_c , and construct a list I_a of N_a uniquely encoded authors, in the same ways as described in Step 4 of Algorithm 4.2.

Stage 2 — message data extraction.

Step 2. (*Message extraction*) For each revision D_{i-1} with $i = 1$ initially, extract the embedded message bitstream M'' by the following steps until $i > n$ where M'' is set empty initially.

Stage 2.1 — extracting message data from the author.

(a) (*Extracting message bits from the author code*) Find the author a_j of the previous revision D_i , and append the n_a -bit code, which is assigned to a_j , to the bitstream M'' .

Stage 2.2 — extracting message bits carried by numbers of changed word sequences in current revisions.

(b) (*Finding candidate word sequences for changes*) Perform Step 5.b of Algorithm 4.2 to get the candidate set Q_r of word sequences for changes in D_{i-1} .

(c) (*Finding the independent word sequence lists in Q_r*) Perform Step 5.c of Algorithm 4.2 to get the set I of lists of independent word sequences from Q_r .

(d) (*Finding the correction pairs between D_i and D_{i-1}*) Perform Algorithm 4.1 with the previous revision D_i and the current revision D_{i-1} as the input to get the correction pairs between D_i and D_{i-1} .

(e) (*Collecting the information of correction pairs*) Collect all the correction pairs yielded in the last step as a set CP , where each element $cp_o = \langle w_o, w_o' \rangle$ in CP includes an *original* word sequence w_o in D_i and a *new* word sequence w_o' in D_{i-1} .

(f) (*Extracting the code for the number of changed word sequences*) Conduct the following steps to extract the code for the number of changed word sequences in D_{i-1} :

(i) compute N_m in D_{i-1} by Eq. (13) and n_m as $\log_2 N_m$;

(ii) express the number of elements in set CP , which is also the total number N_g of the changed word sequences in D_{i-1} , as an n_m -bit binary number N_g' ;

(iii) decrement N_g' by one;

(iv) append the n_m bits of N_g' to M'' .

Stage 2.3 — extracting data via changed word sequences in current revisions.

- (g) (*Choosing splitting points*) Choose randomly $N_g - 1$ elements of I as splitting points using the key K in the same ways as described in Step 5.e of Algorithm 4.2.
- (h) (*Classifying word sequences into independent sets*) Perform Step 5.f of Algorithm 4.2 to classify the elements of I into N_g groups.
- (i) (*Choosing changed word sequences for message extraction*) For each group G_k created in the last step with $k = 1$ initially, encode its word sequences by Huffman coding according to their usage frequencies, and for each word sequence element s_j' in G_k , check whether s_j' is *identical* to a *new* word sequence w_o' in CP ; if so, append the code of s_j' to M'' ; else, check the next word sequence in G_k repeatedly.

Stage 2.4 — extracting message data via replacing word sequences in previous revisions.

- (j) (*Finding the chosen set*) Find the chosen set of word sequence s_j' from database DB_{cw} , and collect all the *original* word sequences in the correction pairs of the chosen set as a set Q_c' .
- (k) (*Extracting the code of replacing word sequences*) Encode the word sequences in Q_c' by Huffman coding according to their usage frequencies, and for each word sequence s_j in Q_c' , check whether s_j is *identical* to an *original* word sequence w_o in CP — if so, then append the Huffman code of s_j to M'' and go to Step 2.i with k increased by one until $k > N_g$; else, check the next word sequence in Q_c' repeatedly.

Stage 3 — message decryption and extraction.

- Step 3. (*Message decryption*) Decrypt the bitstream M'' to get M' using the key K .
- Step 4. (*Message extracting*) Express the first s bits of M' in decimal form as t and output the $(s + 1)$ th through $(s + t)$ th message bits of M' as the secret message M .

4.4 Experimental Results

A collaborative writing database DB_{cw} was constructed by mining the huge collaborative writing data in Wikipedia using Algorithm 4.1 described previously. Note that this is a *totally automatic* work and need be performed only once for building the database DB_{cw} using Algorithm 4.1, where 3,446,959 different correction

pairs were mined from 2,214,481 pages with 33,377,776 revisions in English Wikipedia XML dump. The total size of the downloaded Wikipedia data is about 210.3 GB and the size of the mined data is just 888 MB. Moreover, some revisions might suffer from vandalism [57], [59], and by the method proposed by Bronner and Monz [57], such revisions were ignored if they have been reverted due to vandalism. Also, keywords in Wiki markup¹ were ignored as well. Table 4.1 shows the top 20 most frequently used correction pairs, where the one in the first place is the pair <“BCE”, “BC”> with a correction count of 19,430. Table 4.2 shows some correction pairs, each having more than one word either in its original word sequence or in its new word sequence. One of the correction pairs in this table is <“like”, “such as”> with a correction count of 773.

Table 4.1. Top twenty frequently used correction pairs.

Original word sequence	New word sequence	Usage frequency	Original word sequence	New word sequence	Usage frequency
BCE	BC	19,430	the	a	7,009
BC	BCE	17,878	is	are	6,908
color	colour	15,356	a	the	6,278
colour	color	14,852	are	is	5,430
The	the	14,232	colors	colours	5,301
a	an	9,792	colours	colors	5,078
it's	its	9,658	CE	AD	4,833
is	was	9,607	AD	CE	4,262
an	a	8,954	image	Image	4,259
was	is	7,407	was	were	3,924

The constructed database DB_{cw} contains 1,688,732 chosen sets of correction pairs where all the correction pairs in a chosen set have identical new word sequences, meaning that there are 1,688,732 word sequences which can be chosen and changed to other word sequences in the message embedding phase. Figure 4.11 shows an

¹ http://en.wikipedia.org/wiki/Help:Wiki_markup

illustration of the numbers of entries in the chosen sets with sizes from 2 to 40. Table 4.3 shows the content of a chosen set with the new word sequence “such as,” as well as the usage frequency and Huffman code for each original word sequence which may be replaced by “such as” during message embedding. From the table, we can see that the most frequently used original word sequence is “like,” so it has the shortest code “1” and the largest probability to be chosen.

Table 4.2. Some correction pairs each with more than one word either in the original word sequence or in the new word sequence.

Original word sequence	New word sequence	Usage frequency	Original word sequence	New word sequence	Usage frequency
Irish evil	Evil	2,367	due to	because of	933
Evil	Irish evil	2,253	like	such as	773
US	United States	1,094	didn't	did not	665
It's	It is	1,052	passed away	died	374
due to the fact that	because	359	doesn't	does not	489
have been	were	348	WWII	World War II	395
will be	was	903	UK	United Kingdom	599

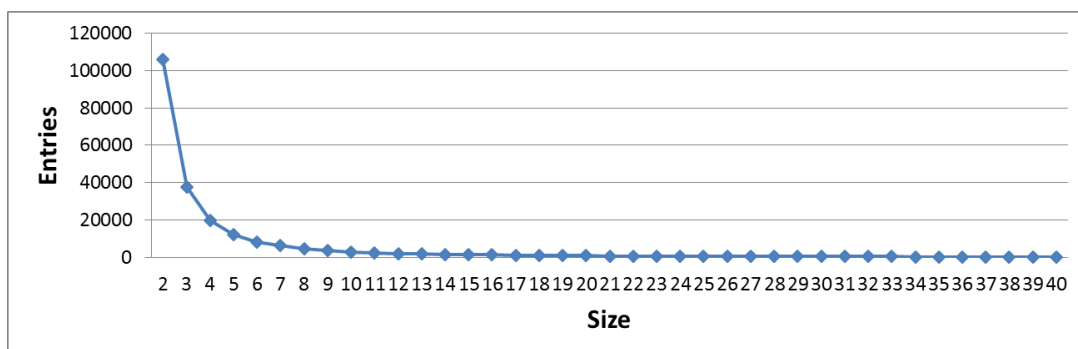


Figure 4.11. The number of entries of chosen sets with the size from 2 to 40.

After the message embedding phase, the proposed system will generate a stego-document to be kept in a collaborative writing platform and a user can later

extract the embedded message from it using a key. Each generated stego-document including its revision history was kept on a Wiki site which was constructed in this study using the free software: MediaWiki². Note that though here the pre-selected collaborative writing platform is the constructed Wiki site, yet the proposed method can be used on *other* collaborative writing platforms as well. As an example, with a cover article as shown in Figure 4.12(a), the message “Art is long, life is short,” and the key “1234” as inputs into Algorithm 4.2, a stego-article as shown in Figure 4.12(c) together with a revision history as shown in Figure 4.12(b) was generated by the proposed method. We can see from Figure Figure 4.12 (b) that five revisions have been created in order to embed the secret message. And Figures 4.12(d) and 4.12 (e) show the extracts of the differences between the two newest revisions, where the words in red in Figure 4.12(d) were corrected to be those in red in Figure 4.12(e) by the author “Natalie.” Figures 4.12(f) and 4.12(g) shows respectively the messages extracted by Algorithm 4.3 using a right key and a wrong one. These results show that when a user uses a wrong key, the system will return a random string as the message extraction result.

Table 4.3. An example of a chosen set with the new word sequence “such as”.

Original word sequence	Usage frequency	Huffman code	Original word sequence	Usage frequency	Huffman code
like	773	1	specifically	12	011001
including	143	00	namely	10	011000
for example	39	0100	particularly	10	0111111
of	29	01110	like the	10	010100
notably	23	01101	most notably	10	010101
especially	20	01011	include	9	0111110
and	16	011110			

A series of experiments with different parameters have also been conducted to quantitatively measure the data embedding capacity of the proposed method using a lot of cover documents as inputs. Since the data embedding capacity is dependent on the secret message content which influences the selections of authors and changed

² <http://www.mediawiki.org/wiki/MediaWiki>.

word sequences for each revision, we have run experiments for each document ten times using different messages as inputs, and recorded the average of the resulting data embedding capacities. The parameters of six different cover documents are shown in Table 4.4. For example, document 1 has 2,419 characters, 641 words, and 80 sentences; document 3 has 10,128 characters, 2,211 words, and so on.

A new tool, called signal-rich-art image, for automatic identification and data capture applications is proposed, which is created from an artistic target image for use as a carrier of a given message. The created image is visually similar to the target image, achieving the effect of so-called signal-rich art. With its function similar to those of barcodes or QR codes, such a type of image is produced by fragmenting the composing characters of the message and injecting them into the target image by a novel image-block luminance modulation scheme. Skillful techniques are also proposed for message extraction from a mobile phone-captured version of the signal-rich-art image printed on paper or displayed on a screen. Good experimental

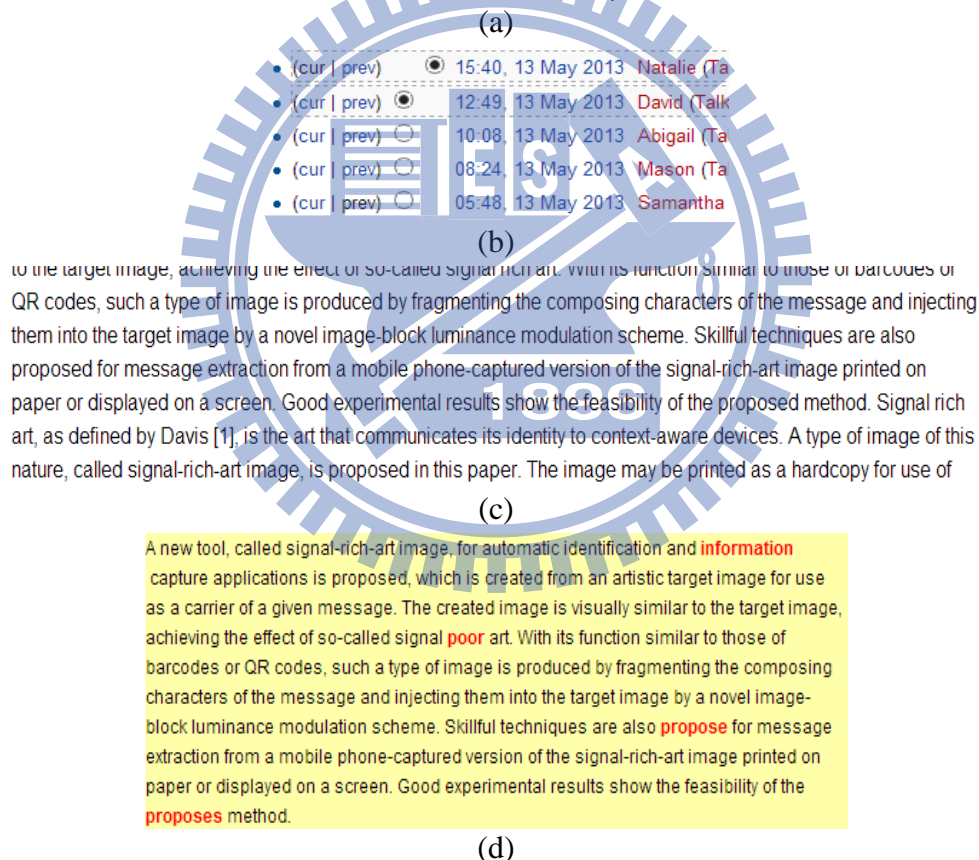


Figure 4.12. An example of generated stego-documents on constructed Wiki site with input secret message “Art is long, life is short.” (a) Cover document. (b) Revision history (c) Stego-document. (d) Previous revision of revision of (e) with words in red being those corrected to be new words in revision of (e) in red. (e) Newest revision of created stego-document. (f) Correct secret message extracted with the right key “1234.” (g) Wrong extracted secret message with a wrong key “123.”

A new tool, called signal-rich-art image, for automatic identification and data capture applications is proposed, which is created from an artistic target image for use as a carrier of a given message. The created image is visually similar to the target image, achieving the effect of so-called signal rich art. With its function similar to those of barcodes or QR codes, such a type of image is produced by fragmenting the composing characters of the message and injecting them into the target image by a novel image-block luminance modulation scheme. Skillful techniques are also proposed for message extraction from a mobile phone-captured version of the signal-rich-art image printed on paper or displayed on a screen. Good experimental results show the feasibility of the proposed method.

(e)

Art is long, life is short

(f)

壇沃裝嶼迥r豐瀚r嫗錚揮鐵r賄統 艱陵r吳r 泯r錯籠r 畎r迨迨佳姪r

(g)

Figure 4.12. An example of generated stego-documents on constructed Wiki site with input secret message “Art is long, life is short.” (a) Cover document. (b) Revision history (c) Stego-document. (d) Previous revision of revision of (e) with words in red being those corrected to be new words in revision of (e) in red. (e) Newest revision of created stego-document. (f) Correct secret message extracted with the right key “1234.” (g) Wrong extracted secret message with a wrong key “123” (continued).

Table 4.4. The information of experimental documents.

Document	Character	Word	Sentence	Document	Character	Word	Sentence
Document 1	2,419	641	80	Document 4	11,215	2,617	86
Document 2	4,762	956	45	Document 5	26,591	6,180	631
Document 3	10,128	2,211	121	Document 6	60,349	14,306	1,603

In these experiments, firstly we selected the replacing word sequences for a revision to be the top n most frequently used ones in the database DB_{cw} , where $n = 2, 4, 8, 16, 32$. Figure 4.13(a) shows the resulting data embedding capacities from which we can see that the more the selected replacing word sequences, the more the embedded message bits. This result comes from the fact that when more replacing word sequences are available, the constructed Huffman codes will become longer.

We have also conducted experiments on using different numbers of revisions (1, 2, 4, 8) in the generated stego-documents to see the resulting data embedding capacities. Figure 4.13(b) shows the results which indicate that when the number of revisions in the stego-document is larger, more message bits can be embedded, as expected. This means that if we want to embed a larger secret message, more

revisions should be generated. Yet, on a Wiki site, each revision will be stored as its original text without any compression. Thus, a larger storage space is required to store more generated revisions when the secret message is longer. However, one can solve this issue by simply comparing the difference between two adjacent revisions and only storing the difference between them where this comparison function may be provided by other collaborative writing platforms if desired. Furthermore, we can see also from Figures 4.13(a) and 4.13(b) that when a cover document has a larger size, the resulting data embedding capacity will be larger as well. Thus, if we want to embed more data, we have to choose a larger cover document.

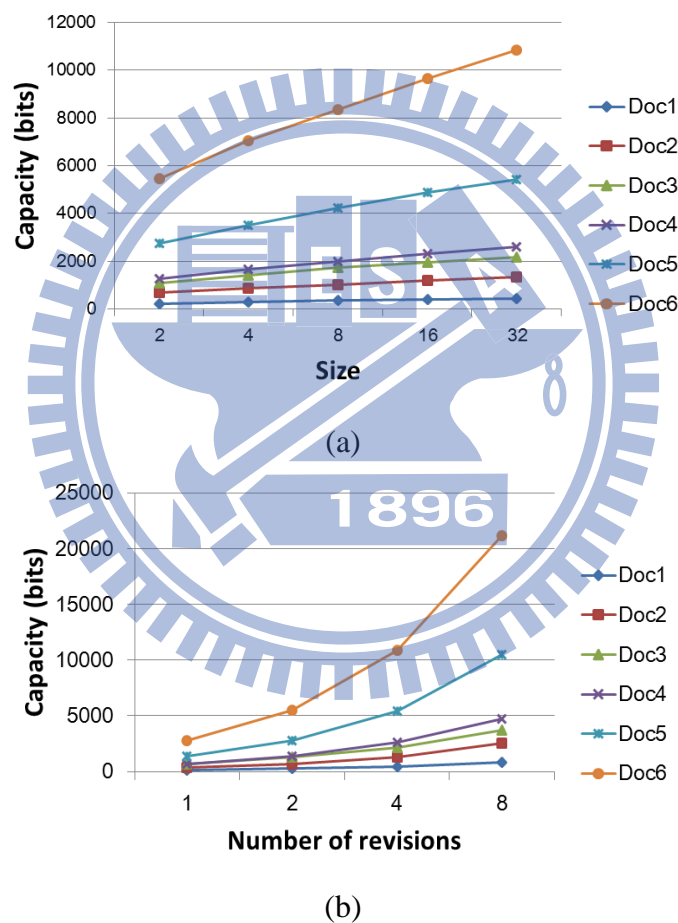


Figure 4.13. The embedding capacities. (a) Embedding capacities of documents with chosen sets of different sizes. (b) Embedding capacities of documents with different number of revisions.

Figure 4.14 shows a comparison of the resulting embedding capacities yielded by the proposed method with those yielded by Liu and Tsai's method [64]. We can see from Figure 4.14 that when the number of revisions of the proposed method is

equal to one, the embedding capacity of the proposed method is very close to that yielded by Liu and Tsai [64]. Note that not every word sequence in the current revision D_{i-1} can be utilized for data embedding in the proposed method, because we limit the maximum number of corrected word sequences in a revision. Thus, when the number of revisions is just *one*, the embedding capacity of the proposed method may not be better than that of Liu and Tsai [64] which allows the use of *every* word for message embedding. However, when the number of revisions is equal to or greater than two, the embedding capacities of the proposed method are instead much larger.

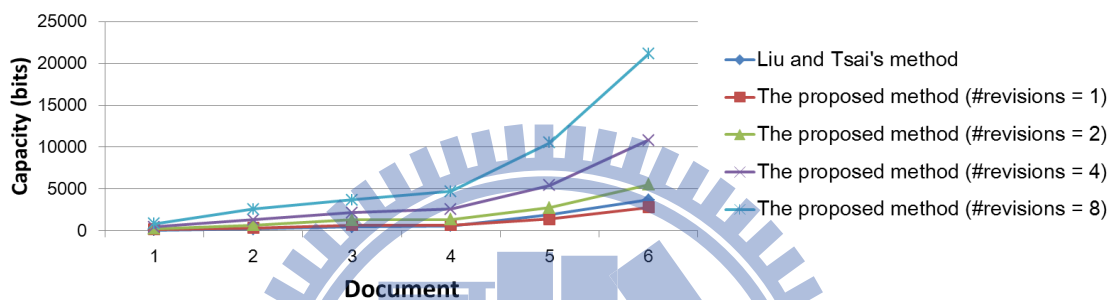


Figure 4.14. Comparison of embedding capacities yielded by Liu and Tsai [64] and proposed method using different numbers of revisions.

Like the methods proposed by [57], [58] which can be utilized for multiple languages, we have tried to apply Algorithm 4.1 to two adjacent revisions of a Chinese document and obtain the correction pairs for them successfully, where the two revisions are shown in Figure 4.15. Note that since Chinese has no explicit word segmentation mark, we cannot use *spaces* to split an article in Chinese into words. Therefore, each character in Chinese was treated as a word directly to solve the issue. Figure 4.15 shows the found correction pairs between the two revisions, in which, e.g., one of the found correction pairs is <做到, 達成>, where both word sequences in the pair mean the same as “achieve” in English.



Figure 4.15. An example to show the interoperability of the proposed method which can be applied on Chinese articles.

Moreover, for the purpose of presenting the contributions made by the proposed method, we have compared it with several other methods for data hiding via texts [35]-[37], [64] as shown in Table 4.5. Firstly, the synonym replacement methods [35]-[37] utilize synonym dictionaries to embed messages, where the synonym dictionaries were usually *manually* built by language experts. And the embedding capacities of these methods are limited, since only those word sequences in the cover document which exist in the synonym dictionary can be utilized for data embedding. Also, since they replace the word sequences in a cover document into their synonyms, the resulting stego-document is usually a worse version of the original cover document due to the possible losses of the original meanings in the replacements. Furthermore, the usage frequencies of the corresponding synonyms of a word sequence are not analyzed in these methods. Secondly, the change tracking method proposed by Liu and Tsai [64] utilizes synonym dictionaries and a small collaborative writing database with only 7,581 chosen sets to embed messages, where the synonym dictionaries were built *manually* as well. Also, the embedding capacities of this method is limited, since only two revisions are generated by two authors and only the

word sequences in the cover document are degenerated for data embedding. Moreover, the usage frequencies of word sequences of this method are just a simulated one created by using the Google SOAP Search API.

Table 4.5. Comparison of methods for data hiding via texts.

Method	Utilized database	Database construction	Embedding capacity	# of revisions	# of authors	Usage frequencies of word sequences
Chapman <i>et al.</i> [35]	Synonym dictionary	Manually	Limited	1	1	–
Bolshakov [36]	Synonym dictionary	Manually	Limited	1	1	–
Shirali-Shahreza and Shirali-Shahreza [37]	Synonym dictionary	Manually	Limited	1	1	–
Liu and Tsai [64]	Synonym dictionary + Small collaborative writing database	Mainly manually	Limited	2	2	Simulated
Proposed method	Large collaborative writing database	Automatically	Unlimited	Unlimited	Many	Real data

As a summary, several merits of the proposed method can now be pointed out, which include: (1) the database of the proposed method is constructed *automatically* from Wikipedia, which is the largest collaborative writing platform on the Internet; therefore, the resulting stego-document generated by the proposed method is more realistic than that generated by the other four methods [35]-[37], [64]; (2) the database constructed by the proposed method is much larger than that by Liu and Tsai [64], with 1,688,732 chosen sets in the former and only 7,581 in the latter; (3) the usage frequency of each correction pair used in the proposed method is a real parameter obtained by mining the collaborative writing data found on Wikipedia, but that of Liu and Tsai [64] is just a simulated one created by using the Google SOAP Search API; and (4) the proposed method can simulate the collaborative writing process conducted by multiple authors and revisions, but Liu and Tsai [64] can only

generate one pre-draft version of a cover text, simulating the work of two authors. Thus, to the best of our knowledge, this is the first work that can simulate the real collaborative writing process with multiple authors and revisions by mining the revision histories on Wikipedia or similar platforms and using the characteristics in the collaborative writing process effectively for message embedding.

Furthermore, to illustrate the *usability* of the proposed method in the real world, it is pointed out that one can build a collaborative writing platform, such as a Wiki site, for uses by a school, company, or government and then implement the proposed method on this platform. For example, for a school, especially with a large size, the teachers may establish a big wiki site with many documents for general teaching, administration, and communication uses, which are accessible by teachers, staff members, students, parents, etc. Sometimes, a teacher might want to communicate with a student's parents in a secret way. Then, the wiki site may be used as a platform for such covert communication of messages. In addition, the teacher may keep secret records of the students on the wiki site using the data embedding schemes provided by the proposed method. That is, a collaborative writing platform can not only let people work collaboratively but also can let people hide message into the documents existing on this platform for applications of covert communication and secret data keeping.

4.5 Security Consideration 1896

4.5.1 Camouflage

In the proposed method, we collected collaborative writing data in Wikipedia written by real people to construct the database DB_{cw} for use in message embedding. Therefore, the stego-document created using DB_{cw} is more robust to attacks by malicious users since the stego-document looks like a realistic work completed by multiple virtual authors on a collaborative writing platform. These authors do not *actually* edit these revisions and so are regarded as *virtual authors*. These virtual authors are created to simulate the real-world authors and used to embed messages to avoid the problem of involving real authors who might leak the secret. Also, to increase the realisticness of the created stego-document, the content of the corrected word sequences in a revision and the word sequences replacing the corrected ones are selected according to the *real* usage frequencies mined from the collaborative writing

data in Wikipedia. Thus, the *statistical property* of simulated corrections in the generated stego-document is close to that of a real one.

Moreover, in order to increase the camouflage effect of the stego-document created by the proposed method, two additional ways can be adopted. The first is to change the time of editing for each generated revision in a stego-document to make it fit the model of revision time in reality, such as the analyzed patterns of revision history mentioned in [63]. This can be achieved by using a key to select randomly a time for each revision in a possible time duration between the related pair of adjacent revisions. The second way is about the selection of authors for data embedding. If an author makes more realistic corrections in his/her revision history of creating a stego-document, then inclusion of him/her as one of the collaborative authors will cause less conspicuousness to adversaries. This idea can be implemented simply by pre-generating some revision data of virtual authors who looks like owning the real collaborative writing work from the collaborative writing platform, as conducted in the study.

In addition, since we assume that only word sequence corrections will occur in the collaborative writing process, the stego-document created by the proposed method contain only such a type of correction. We can remedy this by manipulating additionally the unused portions of the stego-document to include more types of corrections, such as paraphrases and factual edits, to mislead the adversary, where these extra corrections will be ignored during message extraction.

4.5.2 Randomness

According to Kerckhoffs' principle [66], it may be assumed that an adversary, who understands the system but does not have the secret key, can obtain no information about the embedded message. By using the key to enhance the security of the proposed technique, some randomness measures in the phases of secret message embedding and secret message extraction are adopted in the proposed method: (1) randomization of the bits of the secret message to be embedded by encryption; (2) randomization of the parameters and author encoding, including the number of authors, the maximum allowed number of word sequences changed in the revision, the author list, and the author codes; and (3) randomization of the selections of the splitting points for each revision.

More specifically, in the first measure, the secret message is randomized through encryption by using the key, where the encryption method we adopted is AES-256. The Advanced Encryption Standard (AES) is one of the most popular ciphers and provides very high security — the public known attacks up to now have all been shown to be computational infeasible [67]-[68]. In the second measure, the parameters (the number of authors and the maximum allowed number of word sequences changed in the revision) and the author encoding (the author list and the author code for each author) are decided by the key and some pseudo-random number generators. In the third measure, for each revision, $N_g - 1$ lists of the set I for use as the splitting points are selected randomly by the key and a pseudo-random number generator. Let the resulting stego-document D' include revision history $\{D_0, D_1, D_2, \dots, D_n\}$ with N_a authors, the size of the set I of word sequences for selection in each revision D_k be I_k , and the number of word sequences changed in each revision D_k be N_{gk} . Then, for an adversary who does not have the key, he/she needs to execute Algorithm 3 for all possible combinations of word splitting points of the revisions and the author codes, and observe the result to check the correctness of the encrypted secret message. The time complexity for this work is of the order of $(N_a!) \times \left[\prod_{k=0}^{n-1} C(I_k, N_{gk} - 1) \right]$ which is a very big number, where $C(a, b)$ means the combination of a things taken b at a time without repetition. Moreover, it is very hard for an adversary to decide which result yielded by the algorithm is correct because the secret message is encrypted by AES-256 and looks like random noise. Therefore, the proposed method is expected to be secure for secret message hiding.

Additionally, the collaborative writing database may be available to adversaries since they can re-construct the collaborative writing database by using the same Wikipedia data and the same algorithms as those proposed in this study. To increase the security against this type of attack, one additional way to increase the robustness of the proposed method is to use the key to decide the subset of a chosen set and select a word sequence from the subset. Therefore, only authorized users with the key can know the correct subset of the chosen set, and an adversary cannot.

4.5.3 Possible extensions for the proposed method using natural language processing methods

For the ability of constructing the collaborative writing database *automatically* and generalizing the proposed method for *multi-language uses*, four characteristics of collaborative writing as mentioned previously have been analyzed based on the *assumption* that only word sequence corrections will be made in a revision. However, the real collaborative writing process is much more complicated and language-dependent, so data hiding via collaborative writing is still worth intensive researches.

Many possible methods in natural language processing [57]-[59], [69] may be applied to extend the proposed method. For example, some original word sequences in an input cover document may be *polysemous*. Therefore, selecting appropriate word sequences from DB_{cw} by the proposed method to replace such polysemous word sequences might constitute a meaningless context. One possible way out is to analyze the *distributional similarity* of word sequences [69] to find appropriate replacing word sequences that do not cause this problem, where distributional similarity means the similarity in the meanings of those words that have the same contexts in documents. Moreover, we can also build language models [57]-[59], such as dependency trees used in grammatical analysis, to embed messages during revision generations based on the model.

4.6 Summary

A new data hiding method via creations of fake collaboratively-written documents on collaborative writing platforms has been proposed. An input secret message is embedded in the revision history of the resulting stego-document through a simulated collaborative writing process with multiple virtual authors. With this camouflage, people will take the stego-document as a normal collaborative writing work and will not be expected to realize the existence of the hidden message. To generate simulated revisions more realistically, a collaborative writing database was mined from Wikipedia, and the Huffman coding technique was used to encode the mined word sequences in the database according to the statistics of the words. Four characteristics of article revisions were identified, including the author of each revision, the number of corrected word sequences, the content of the corrected word

sequences, and the word sequences replacing the corrected ones. Related problems arising in utilizing these characteristics for data hiding have been solved skillfully, resulting in an effective multi-way method for hiding secret messages into the revision history. Moreover, because the word sequences used in the revisions were collected from a great many of real people's writings on Wikipedia, and because Huffman coding based on usage frequencies is applied to encode the word sequences, the resulting stego-document is more realistic than other text steganography methods, such as word-shift methods [30], non-displayed characters based methods [31], synonym replacement methods [35]-[37], etc. The experimental results have shown the feasibility of the proposed method. Future works may be directed to analyzing more characteristics of collaborative writing works or establishing appropriate language models [57]-[59] for more effective data hiding or other applications.



Chapter 5

A New Data Hiding Technique via Message-rich Character Image for Automatic Identification and Data Capture Applications

5.1 Introduction

With the advance of technology, machines have long been used to read automatically information in the reality for various applications, like optical character recognition (OCR), license plate recognition, supermarket checkout systems, etc. Recently, many more methods have been developed for this purpose, and they are collectively known as AIDC techniques [40]. The processed information is presented in various forms, some being visible (like barcodes) and others invisible (like watermarks hidden behind images). Such forms of multimedia, enabling the vision of pervasive communication, are termed integrally as *message-rich multimedia* in this study as mentioned previously.

One technique that realizes the use of message-rich multimedia for the AIDC purpose is *barcode reading*. Being attached to objects, barcodes represent machine-readable data by patterns of lines, rectangles, dots, etc. The data encoded into such barcodes can be extracted using barcode reading techniques [24]-[27]. But most types of barcodes, such as Code 39 [20], PDF417 [21], QR code [22], and Data Matrix code [23] shown in Figure 5.1, just encode information to yield unsightly images with no aesthetics. If a barcode contains not only the encoded information but also has a visual appearance of an art image, the artistic effect of the barcode will be more attractive than those of conventional ones.

Data hiding is an alternative pervasive communication technique for the AIDC purpose that embeds data into cover media for applications like covert communication, copyright protection, authentication, etc. With the advance of computer technology, many data hiding methods have been applied on *digital* cover media, such as images, videos, audios, text documents, etc. However, these data hiding methods transfer data via *digital* files only. Furthermore, they are mostly insufficient to enable the vision of

pervasive communication when one wants to interact with the surrounding environment. Such methods may be called “*digital*” *data hiding*.

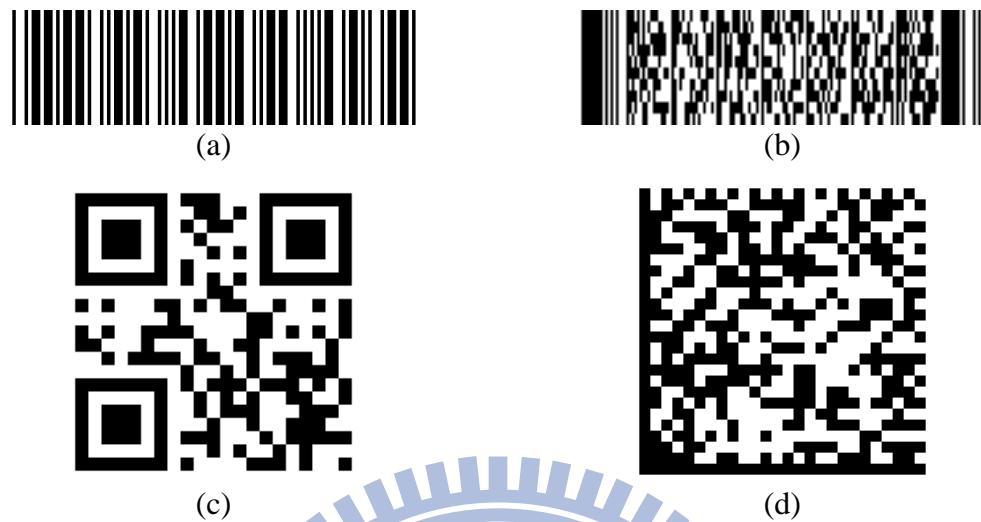


Figure 5.1. Examples of commonly-used barcodes. (a) Code 39. (b) PDF 417. (c) QR code. (d) Data matrix code.

Another type of data hiding, which may be called “*hardcopy*” *data hiding*, can embed information into the so-called *image barcodes* using halftone techniques [17]-[19]. These barcodes have the visual appearances of other images and the encoded information can be decoded from their *hardcopy* versions acquired by scanners. That is, the encoded information can survive *print-and-scan* “*attacks*.” However, if one uses a mobile device to capture images of *hardcopy image barcodes*, the information might not be decoded successfully since the captured image will suffer from more types of distortions than those acquired by scanning, such as geometric deformation, noise addition, blurring, etc. Also, message carriers other than printed papers, such as screens on display devices, cannot be used to encode information since the halftone methods are based on the printing technique. Instead, the method proposed in this study can decode the message which is carried in an image captured from a printed paper or a display screen using a mobile-device camera, achieving the effect of pervasive communication.

Specifically, a new data hiding technique via message-rich character image, which is created from an artistic target image for use as a carrier of a given message, is proposed. The image may be printed as a hardcopy for use of any purpose, which is then “re-imaged” by a mobile-phone camera and “understood” by some *automatic*

identification and data capture (AIDC) techniques [40] proposed in this study. Message-rich character images may be of the forms of documents, labels, posters, etc. Also, such images may have the visual appearances of artistic-flavored photos, pictures, paintings, which are more attractive to humans than those produced by conventional AIDC techniques, like barcodes, QR-codes, etc.

5.2 Idea of Proposed Method

The proposed message-rich character images not only includes the content of a given message, but also has an artistic effect of being visually similar to the pre-selected target image. The proposed method using the message-rich character images for AIDC purposes is illustrated in Figure 5.2, which includes two phases: image generation and message extraction.

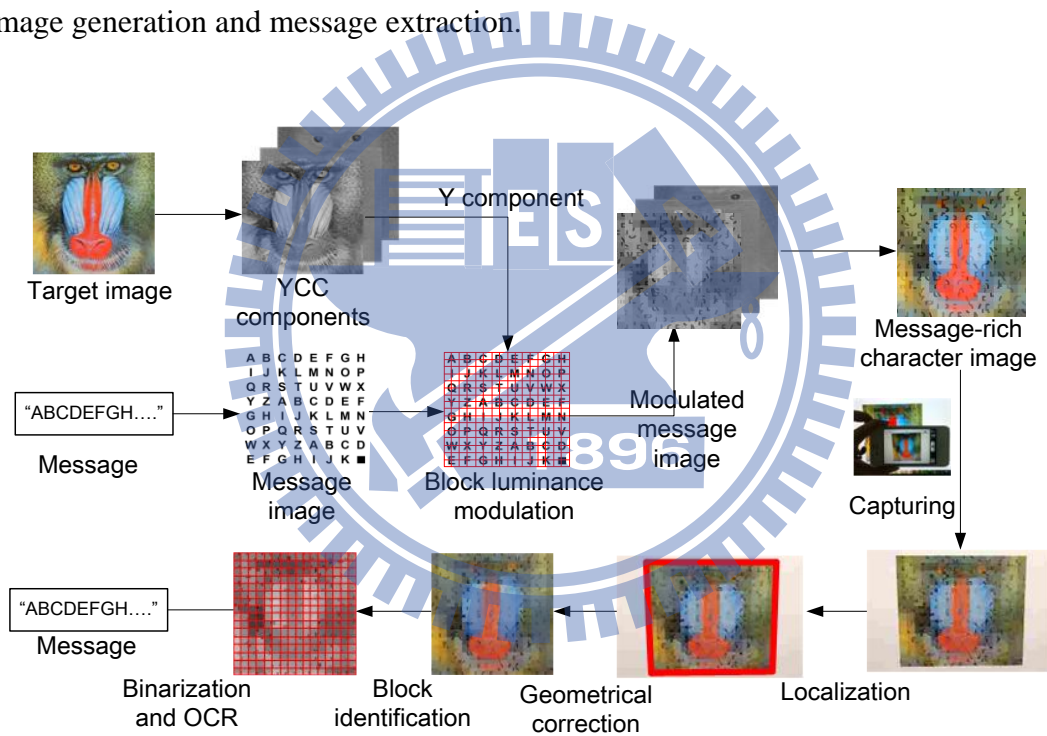


Figure 5.2. Illustration of proposed method.

In the first phase, given a target image I_t and a message M , a message-rich character image I_s is created by three steps: (1) transform M into a *message image* I_m consisting of the characters of the message content; (2) *modulate* the gray values of each *character-fragment* F_i of I_m into two values calculated from the Y-channel values of the corresponding target block B_i of I_t , resulting in a *modulated message image* I_m' ; (3) replace the Y-channel of I_t with I_m' to get the desired I_s .

In the second phase, the message M is extracted from a captured version I_s' of a paper of the printed message-rich character image I_s by three steps: (1) localize and segment out the region I_s'' of the original part of I_s in I_s' ; (2) perform an inverse perspective transform to correct the geometric distortion in I_s'' ; (3) identify the blocks in I_s'' , binarize them, and perform OCR to extract the message M from them.

5.3 Generation of Message-rich Character Image

5.3.1 Message image creation

Unlike most barcode systems that encode message contents by patterns (dots, lines, etc.), the proposed method converts a message M into a set of *binary character shapes* drawn from a database, as illustrated in Figure 5.3(a). Next, a message image I_M of the size of the target image I_T is created by aligning the character shapes plus an *ending pattern* as shown in Figure 5.3(b) in a raster-scan order. For example, with the target image I_T as shown in Figure 5.3(c) and the message $M = \text{“ABCDEFGH...”}$, the resulting message image I_m is as that shown in Figure 5.3(d). If the result cannot fill up I_M , then repetitions of the character shapes are conducted.

5.3.2 Block luminance modulation

After the message image I_M is created, it is “injected” into the target image I_T under the constraint that the resulting image retains the visual appearance of I_T . For this, we utilize a characteristic of the YCbCr color model — the luminance component Y is *independent* of the others [70] — to embed I_M into the Y -channel of I_T . This will solve a problem of illumination variation encountered in the later stage of message extraction. A block luminance modulation technique is proposed here to divide the message image I_M into character-fragments F_i and modulate the *mean* of each F_i to be the same as that of the corresponding target block B_i of I_T . The resulting modulated message image I_M' looks like the Y -component of the target image I_T . For example, Figure 5.3(f) shows the created modulated message image I_M' which looks like the Y -component of the target image shown in Figure 5.3(e), and Figure 5.3(g) shows a zoom-out of part of Figure 5.3(f) (the red portion).

More specifically, firstly the message image I_M and the Y -component of I_T are divided into character-fragments and target blocks, respectively, where the size of each block is $1/4$ times of a character image. The character-fragments F_i then are

fitted into the target blocks B_i in a random way controlled by a key K . Secondly, let N_B and N_W denote the numbers of black and white pixels of the character and non-character parts in F_i , respectively. The pixels of each B_i are sorted according to their Y values in an ascending order to obtain an ordered Y-value set $\{q_1', q_2', \dots, q_m'\}$. Then, two representative Y values r_1 and r_2 are computed for B_i as follows:

$$r_1 = \frac{1}{N_B} \sum_{t=1}^{N_B} q_t', \quad r_2 = \frac{1}{N_W} \sum_{t=N_B+1}^{N_B+N_W} q_t'. \quad (14)$$

Note here that $r_2 \geq r_1$. Finally, the value p_t of each pixel P_t in F_i is modulated to obtain a new pixel value p_t' by the following rule:

$$\text{set } p_t' = r_1 \text{ if } P_t \text{ is black; or } r_2 \text{ if } P_t \text{ is white.} \quad (15)$$

The mean $\mu_{F_i'}$ of the character fragment F_i' so modulated will be equal to the mean μ_{B_i} of the target block B_i because we have:

$$\mu_{B_i} = \frac{1}{m} \sum_{t=1}^m q_t' = \frac{1}{N_B + N_W} \sum_{t=1}^{N_B+N_W} q_t', \quad (16)$$

and

$$\begin{aligned} \mu_{F_i'} &= \frac{1}{m} \sum_{t=1}^m p_t' = \frac{1}{N_B + N_W} \sum_{t=1}^{N_B+N_W} p_t' \\ &= \frac{1}{N_B + N_W} \sum_{t=1}^{N_B} p_t' + \frac{1}{N_B + N_W} \sum_{t=N_B+1}^{N_B+N_W} p_t' \\ &= \frac{1}{N_B + N_W} \sum_{t=1}^{N_B} r_1 + \frac{1}{N_B + N_W} \sum_{t=N_B+1}^{N_B+N_W} r_2 \\ &= \frac{1}{N_B + N_W} N_B r_1 + \frac{1}{N_B + N_W} N_W r_2 \\ &= \frac{1}{N_B + N_W} N_B \left(\frac{1}{N_B} \sum_{t=1}^{N_B} q_t' \right) + \frac{1}{N_B + N_W} N_W \left(\frac{1}{N_W} \sum_{t=N_B+1}^{N_B+N_W} q_t' \right) \\ &= \frac{1}{N_B + N_W} \sum_{t=1}^{N_B+N_W} q_t' = \mu_{B_i}. \end{aligned} \quad (17)$$

This means that the *overall* gray appearances of the modulated message image I_M' and the Y-component of I_T are roughly the same, as already mentioned. Accordingly, we replace the Y-component of I_T with I_M' to generate finally the

desired message-rich character image I_c which has the visual color appearance of I_t , as shown by the example seen in Figure 5.3(h).

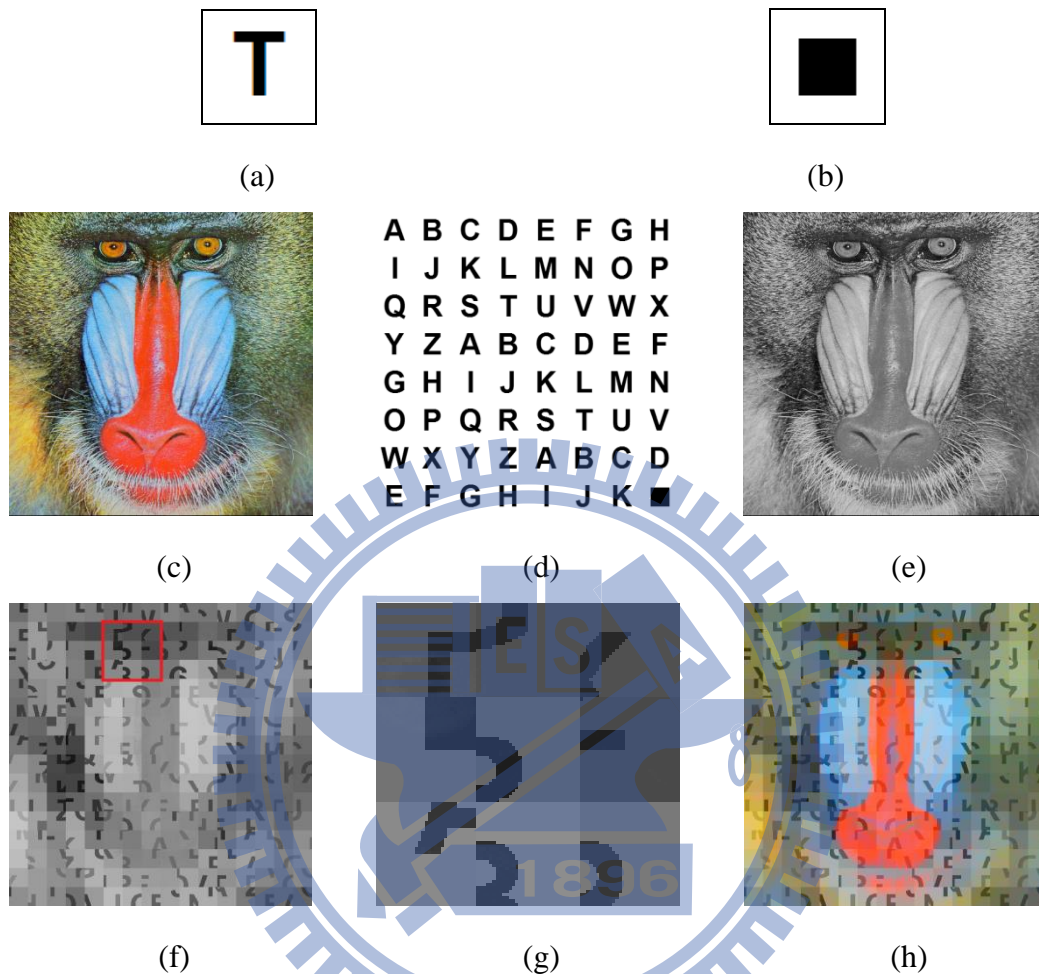


Figure 5.3. Message-rich character image generation. (a) Image of character “T.” (b) Ending pattern. (c) Target image. (d) Message image. (e) Y-channel of (c). (f) Modulated message image. (g) Zoom-out of red square region in (f). (h) Resulting printed message-rich character image.

Accordingly, later while conducting message extraction, the message characters can be extracted from the Y-component of a captured version of I_c by classifying the pixels of each block into two groups according to their Y values, with the two pixel groups representing the character and non-character parts, respectively. Also, an OCR technique is applied to extract these characters. However, if the two representative values r_1 and r_2 are too close, it is hard to separate them in the message extraction phase. Therefore, an adjustment of the representative values r_1 and r_2 is conducted, resulting in r_1' and r_2' , so that the absolute difference between r_1' and r_2' becomes not

smaller than a pre-defined contrast threshold $\delta \geq 0$. For example, Figure 5.4 shows four character-fragments resulting from modulations with different values of δ , from which one can see that the two colors in modulated character-fragments will be more easily distinguished when δ is larger.



Figure 5.4. Modulated character-fragments resulting from uses of different contrast threshold values of δ for the difference between the two representative values r_1 and r_2 . (a) $\delta=0$. (b) $\delta=10$. (c) $\delta=20$. (d) $\delta=30$. (e) $\delta=40$. (f) $\delta=50$.

The detail of the proposed representative-value adjustment scheme is described in the following. Note that, after the adjustment, the absolute difference between r_1' and r_2' must be not smaller than the contrast threshold δ , and that the mean of the modulated character fragment F_i'' based on r_1' and r_2' must be identical to that of the target block B_i . Thus, the values of r_1' and r_2' must satisfy the following two constraints:

$$|r_2' - r_1'| \geq \delta; \quad (18)$$

$$\mu_{F_i''} = \mu_{B_i}. \quad (19)$$

Two possible cases can be identified in the adjustment process: 1) the original absolute difference δ_0 of r_1 and r_2 is already not smaller than δ , i.e., $\delta_0 \geq \delta$; and 2) the reverse, i.e., $\delta_0 < \delta$. In the first case, the values of r_1 and r_2 satisfy (18) and (19) automatically, so that they may be used as r_1' and r_2' , respectively, directly, i.e., we have the rule:

$$\text{if } \delta_0 \geq \delta, \text{ then set } r_1' = r_1 \text{ and } r_2' = r_2. \quad (20)$$

For the second case with $\delta_0 < \delta$, the absolute difference between the two representative values must be increased, after the adjustment, *at least* for the amount of $\delta - \delta_0$ for the resulting values of r_1' and r_2' to satisfy constraint (18). Specifically, let the adjustment value of r_1 be t so that $r_1' = r_1 - t$. Then, the adjustment value of r_2

should be at least $(\delta - \delta_0) - t$ so that $r_2' \geq r_2 + [(\delta - \delta_0) - t]$. Such value adjustments indeed can satisfy constraint (18) because with the fact that $r_2 - r_1 = \delta_0$, we have

$$|r_2' - r_1'| \geq |r_2 - r_1 + (\delta - \delta_0)| \geq \delta.$$

Also, for r_1' and r_2' to satisfy constraint (19), suppose that r_2' is adjusted for *exactly* the amount of $(\delta - \delta_0) - t$. The reason to make this assumption is to reduce the color distortion in the created message-rich character image, as will be discussed later in Section 5.5. Then, the yet-unknown value t may be computed by:

$$\begin{aligned}
\mu_{F''} &= \frac{1}{m} \sum_{t=1}^m p_t'' = \frac{1}{N_B + N_W} \sum_{t=1}^{N_B} p_t'' + \frac{1}{N_B + N_W} \sum_{t=N_B+1}^{N_B+N_W} p_t'' \\
&= \frac{1}{N_B + N_W} \sum_{t=1}^{N_B} r_1' + \frac{1}{N_B + N_W} \sum_{t=N_B+1}^{N_B+N_W} r_2' \\
&= \frac{1}{N_B + N_W} N_B r_1' + \frac{1}{N_B + N_W} N_W r_2' \\
&= \frac{1}{N_B + N_W} N_B (r_1 - t) + \frac{1}{N_B + N_W} N_W (r_2 + [(\delta - \delta_0) - t]) \\
&= \frac{1}{N_B + N_W} N_B r_1 + \frac{1}{N_B + N_W} N_W r_2 - \\
&\quad \frac{1}{N_B + N_W} N_B t + \frac{1}{N_B + N_W} N_W (\delta - \delta_0) - \frac{1}{N_B + N_W} N_W t \\
&= \mu_{B_i} - \frac{1}{N_B + N_W} N_B t + \frac{1}{N_B + N_W} N_W (\delta - \delta_0) - \frac{1}{N_B + N_W} N_W t \\
&= \mu_{B_i}
\end{aligned} \tag{21}$$

where the fact that the new value of p_t'' in F_i'' is set to be r_1' if the color of P_t in F_i is black; or to be r_2' if the color of P_t in F_i is white has been used in the derivations, and the last step is based on the use of (19). Accordingly, we can get:

$$-\frac{1}{N_B + N_W} N_B t + \frac{1}{N_B + N_W} N_W (\delta - \delta_0) - \frac{1}{N_B + N_W} N_W t = 0, \tag{22}$$

from which the desired value t can be derived to be

$$t = [N_W / (N_B + N_W)](\delta - \delta_0).$$

Therefore, the values of r_1' and r_2' can be computed by the rule:

$$\text{if } \delta_0 < \delta, \text{ then set } r_1' = r_1 - t \text{ and } r_2' = r_2 + [(\delta - \delta_0) - t]. \tag{23}$$

5.3.3 Algorithm for message-rich character image creation

Based on the above discussions, a detailed algorithm for message-rich character image creation is described as follows.

Algorithm 5.1. Message-rich character image creation.

Input: a target image I_T , a message M , and a contrast threshold value δ .

Output: a message-rich character image I_C .

Steps:

Stage 1 — transforming the message into a message image.

- Step 1. Convert M into a set of *binary character shapes* drawn from a database.
- Step 2. Create the message image I_M of the size of the target image I_T by aligning the character shapes plus an ending pattern in a raster-scan order.
- Step 3. If the aligning result in Step 2 cannot fill up I_M , then repetitions of the character shapes are conducted.

Stage 2 — modulating the message image.

- Step 4. Divide the Y-component of target image I_T into *target blocks* $\{B_1, B_2, B_3, \dots, B_N\}$ where $N = N_T \times N_T$.
- Step 5. For each fragment block F_i in message image I_M , generate a modulated fragment block F_i'' as follows.
 - (a) Compute two representative values r_1 and r_2 of the corresponding target block B_i according to (14).
 - (b) Compute $\delta_i = |r_2 - r_1|$, and use it and the input contrast threshold δ to obtain two adjusted representative values r_1' and r_2' from r_1 and r_2 according to (20) and (23).
 - (c) For each pixel P_t in F_i , if P_t is black, set the value p_t'' of the corresponding pixel P_t'' in F_i'' as $p_t'' = r_1'$; else, set $p_t'' = r_2'$.
- Step 6. Compose all the resulting F_i'' to get a modulated message image, denoted by I_M' .

Stage 3 — injecting the message image into the target image.

- Step 7. Replace the Y-component of I_T with I_M' to generate the desired message-rich character image I_C as the output.

5.4 Message Extraction

The various techniques proposed for extracting the message embedded in the message-rich character image are described first, with a combination of them described as an algorithm at last.

5.4.1 Message-rich character image localization and inverse perspective transform

Assume that the message-rich character image I_c is printed and posted or displayed against a white background, and that the captured image I_d contains only the original image of I_c and the background. The first assumption here may be removed simply by adding a white surrounding zone to I_c . To extract the message from I_d , we must *localize* the region of I_c in I_d . For this, we apply the Hough transform and polygonal approximation to find the largest non-white quadrangle Q in I_d as shown by the example seen in Figure 5.5(a). Also, image I_d will suffer from perspective distortion if the axis of the camera is not directed perpendicularly toward the plane of the message-rich code image I_c [27] during image acquisition, as seen in Figure 5.5 (a) as well. As a remedy, an *inverse perspective transform* is performed on Q to correct the distortion. The result of conducting this on Figure 5.5(a) is shown in Figure 5.5(b). Finally, the Y-component of the resulting Q is taken as an intermediate result, which we call the *captured modulated message image* and denote it by I_M'' .



Figure 5.5. Localization and correction of perspective distortion in captured message-rich character image. (a) Localized message-rich character image portion (enclosed by red rectangle). (b) Result of perspective distortion correction applied to red portion region in (a).

5.4.2 Block number identification and block segmentation

To identify the blocks in I_M'' in order to binarize them and perform OCR to the contents of them, an idea similar to the Hough transform [71] is adopted: uses the statistics of the pixels' gradient values to *guess* the number N_S of blocks in the horizontal or vertical direction in I_M'' because those pixels on the splitting lines between the blocks usually have larger gradient values.

In more detail, at first the gradient value g_{xy} of each pixel R_{xy} with value r_{xy} at coordinates (x, y) in I_M'' is computed by a Sobel operator [72]:

$$g_{xy} = \left| (r_{x+1,y-1} + 2r_{x+1,y} + r_{x+1,y+1}) - (r_{x-1,y-1} + 2r_{x-1,y} + r_{x-1,y+1}) \right| + \left| (r_{x-1,y+1} + 2r_{x,y+1} + r_{x+1,y+1}) - (r_{x-1,y-1} + 2r_{x,y-1} + r_{x+1,y-1}) \right|. \quad (24)$$

Next, for each *possible* value n_j of N_S , the distance d_j between the splitting lines of every two possible adjacent blocks is computed as $d_j = L/n_j$ where L is the side length of the square-shaped I_M'' . Then, the horizontal or vertical lines separated by the distance of d_j are taken as *candidate* splitting lines, where the positions of these candidate splitting lines described by image coordinates are computed by:

$$x = u \times d_j \text{ and } y = v \times d_j, \quad (25)$$

where $u = 1 \sim \lfloor L/d_j \rfloor$ and $v = 1 \sim \lfloor L/d_j \rfloor$, respectively. Also, the *average gradient value* AG_{n_j} of the pixels on each candidate spitting line is computed as:

$$AG_{n_j} = \frac{1}{\lfloor L/d_j \rfloor \times L} \sum_{u=1}^{\lfloor L/d_j \rfloor} \sum_{y=1}^L g_{(u \times d_j)y} + \frac{1}{\lfloor L/d_j \rfloor \times L} \sum_{v=1}^{\lfloor L/d_j \rfloor} \sum_{x=1}^L g_{x(v \times d_j)}. \quad (26)$$

Finally, the value n_j of N_S with the *largest* average gradient value is taken as the desired number of blocks of I_M'' in the horizontal or vertical direction, and division of I_M'' into blocks is conducted accordingly. For example, Figure 5.6(a) shows a captured modulated message image I_M'' , Figure 5.6 (b) is the image of the computed gradient values, and Figure 5.6(c) illustrates the average gradient values for different N_S , where the n_j with the largest average gradient value is seen to be 16 (indicated by the orange arrow). Therefore, the found value for N_{SF} is 16. The corresponding image division result is shown in Figure 5.6(d).

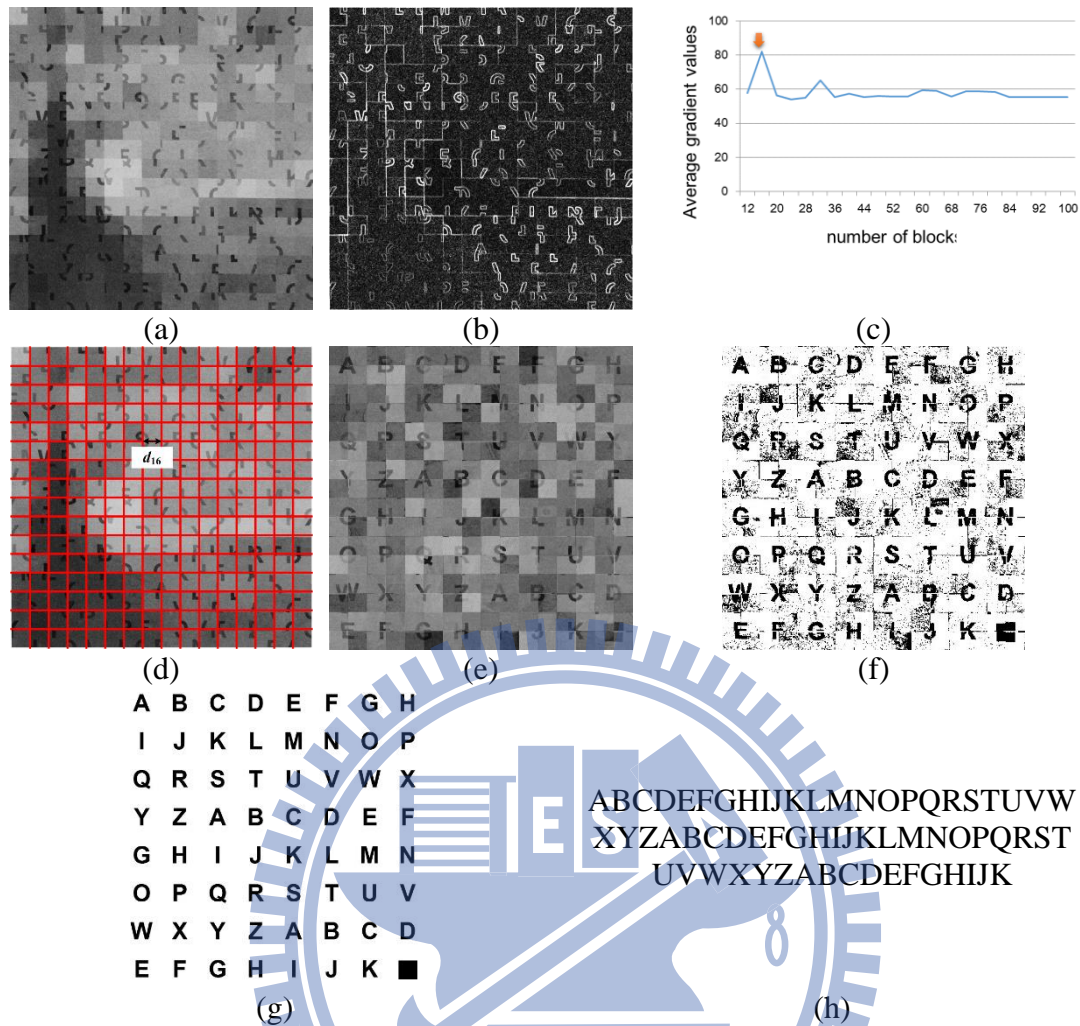


Figure 5.6. Message extraction. (a) Captured modulated message image I_M'' . (b) Gradient values of (a). (c) Average gradient values of pixels on candidate spitting lines for different N_s . (d) Image division result according to determined number of blocks $N_s = 16$. (e) Fragment reordering result of (d). (f) Binarization result of (e). (g) OCR result of (f). (h) Extracted message.

5.4.3 Binarization and optical character recognition

After the blocks of I_M'' are segmented out, the character-fragments of the message image I_M may be recovered from the blocks by using the key K mentioned previously. Denote the resulting image by I_M''' . Then, moment-preserving thresholding [73] is applied to F_i' to binarize it, and every four mutually-connected binarized blocks are grouped up to form a character image I_{c_i}' since a character image is divided into four blocks in the message image generation phase.

Next, a similar degree sd_{ij} between each I_{c_i}' and each reference character image I_{c_j} in the database is computed:

$$sd_{ij} = 1 - \left(\sum_{x=1}^m |p_{ix}' - p_{jx}| \right) / m, \quad (27)$$

where the pixels of I_{c_i}' and I_{c_j} are assumed to be $\{p_{i1}', p_{i2}', \dots, p_{im}'\}$ and $\{p_{j1}, p_{j2}, \dots, p_{jm}\}$, respectively. Finally, each I_{c_j}' is recognized using an OCR scheme according to the computed similarity degree values: the character with the largest similarity is taken as the message delivered by I_{c_j}' . For example, the recovered original message image I_m with its character-fragments reordered using a key is shown in Figure 5.6(e), which, after being binarized, results in Figure 5.6(f). The OCR result of Figure 5.6(f) is shown in Figure 5.6(g), and the final extracted message characters are shown in Figure 5.6(h).

5.4.4 Message extraction algorithm

A detailed message extraction algorithm is as follows.

Algorithm 5.2. Message extraction.

Input: a captured version I_d of a message-rich character image.

Output: the message M embedded originally in I_d .

Steps:

Stage 1 — localizing the message-rich character image.

Step 1. Find the largest non-white quadrangle Q in I_d by the Hough transform and polygonal approximation.

Stage 2 — correcting geometric distortion.

Step 2. Perform an inverse perspective transform on Q to correct the perspective distortion and take the Y-component of Q as the captured modulated message image I_M'' .

Stage 3 — identifying blocks in the message image.

Step 3. Compute the gradient value g_{xy} of each pixel R_{xy} in I_P'' according to (24).

Step 4. For each possible value n_j of N_s , compute the average gradient values AG_{n_j} of the pixels on each candidate spitting line according to (26).

Step 5. Select the value n_j yielding the largest AG_{n_j} for use as the desired number N_s of blocks of I_M'' in the horizontal or vertical direction; and divide I_M'' accordingly into blocks.

Stage 4 — binarizing the blocks to extract the message.

Step 6. Binarize each block F_i' by moment-preserving thresholding [73].

Step 7. Group every four mutually-connected binarized blocks to form character images I_{c_i}' .

Step 8. Extract the corresponding character from each character image I_{c_i}' by the following steps.

- (a) Compute the the similar degree sd_{ij} between each I_{c_i}' and each reference character image I_{c_j} according to (27).
- (b) Select the character with the largest similarity sd_{ij} as the message delivered by I_{c_i}' .

Step 9. Concatenate the extracted characters to get the embedded message M .

5.5 Experimental Results

The proposed system was developed using Microsoft C#.NET and the generated message-rich character images were captured with an iPhone 4S. The character image database includes the printable characters of the ASCII codes. A series of experiments using different parameters have been conducted and corresponding statistics plotted (in Figure 5.8) to show the accuracy rates of extracted message characters, including: (1) the contrast threshold δ of the minimum difference between r_1 and r_2 ; and (2) the number of blocks N_s in the horizontal or vertical direction of the message image.

Figures 5.7(a) through 5.7(c) show three test target images used in the experiments. The corresponding message-rich character images generated with parameters $N_s = 32$ and $\delta = 40$ are shown in Figures 5.7(d) through 5.7(f). These images were all printed to be of the same size of 127×127 mm.

One of the parameters that influence the accuracy of the extracted message is the contrast threshold δ for the minimum difference between r_1 and r_2 . If δ is too small, the two representative values r_1 and r_2 will be too close so that the message extracted might be wrong. For example, Figure 5.8(a) shows the accuracy rate of the extracted messages with $\delta = 0, 20, 40,$ and 60 , from which it can be seen that the larger the

value of δ , the higher the accuracy rate of the extracted message; when δ is larger than 40, an accuracy rate of 100% is yielded.

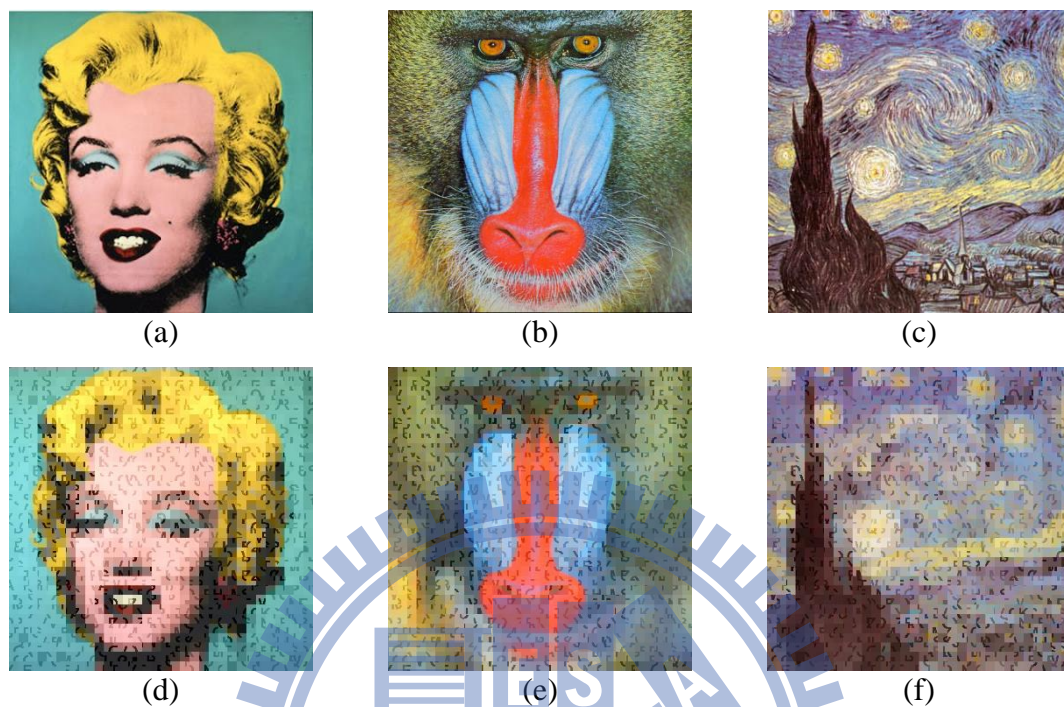


Figure 5.7. Created message-rich character images. (a)-(c) Test target images. (d)-(f) Resulting message-rich character images with $N_s = 32$ and $\delta = 40$.

It can also be seen from Figure 5.8(b) that the larger the value of δ , the larger the RMSE of the resulting message-rich character image with respect to the target image. So there is a tradeoff between achieving higher message extraction accuracy and obtaining a better visual quality in the resulting message-rich character image.

Another parameter that influences the accuracy of the extracted message is the number N_s of blocks in the horizontal or vertical direction in the message image. The larger the value of N_s , the larger the character capacity of the message image; however, the larger the value of N_s , the smaller the size of the block, and so the lower the accuracy of the extracted message, as can be seen from Figure 5.8(c).

Finally, to show the robustness of the proposed method, we have conducted some attacks on the created message-rich character images. For example, Figures 5.9(a) and 5.9(b) show two attacked versions of the message-rich character image shown in Figure 5.7(f) with message “ComputerVisionLab” injected. The experimental results show that the carried message can still be extracted from either of

these two attacked images. In addition, by regarding image taking from a display screen as a type of attack, a third attacked version so acquired is shown in Figure 5.9(c). The resulting message extraction rate is 96.88%, which, though not 100%, means that the proposed method can handle message carriers other than paper copies.

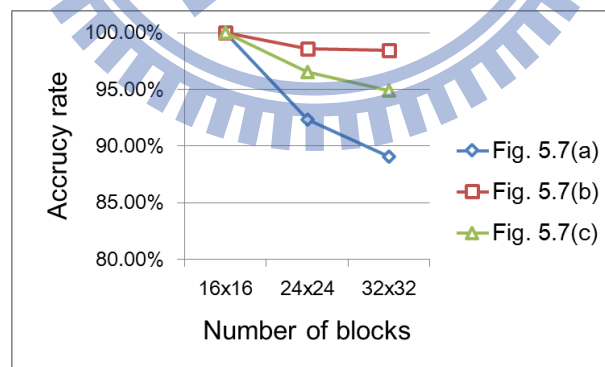
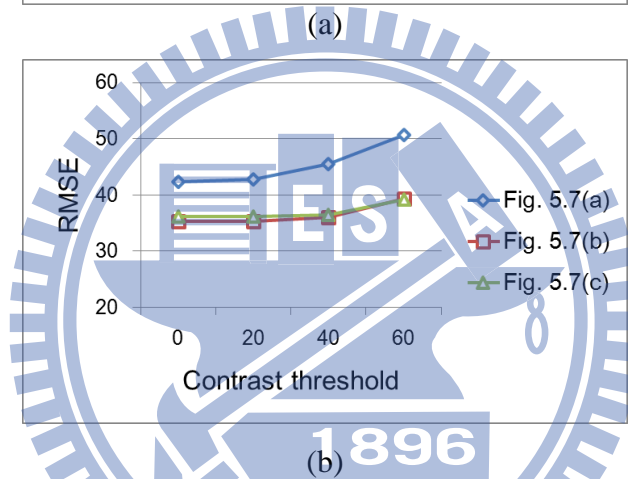
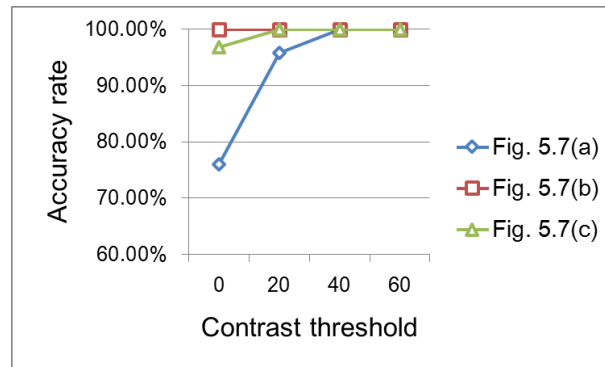


Figure 5.8. Plots of trends of results using various parameters. (a) Accuracy rates of extracted messages with different contrast threshold values δ , with #blocks $N_s = 16$. (b) RMSE values of created message-rich character images with respect to target images for different contrast threshold values of δ , with #blocks $N_s = 16$. (c) Accuracies of extracted messages with different #blocks N_s , where contrast threshold value $\delta = 40$.

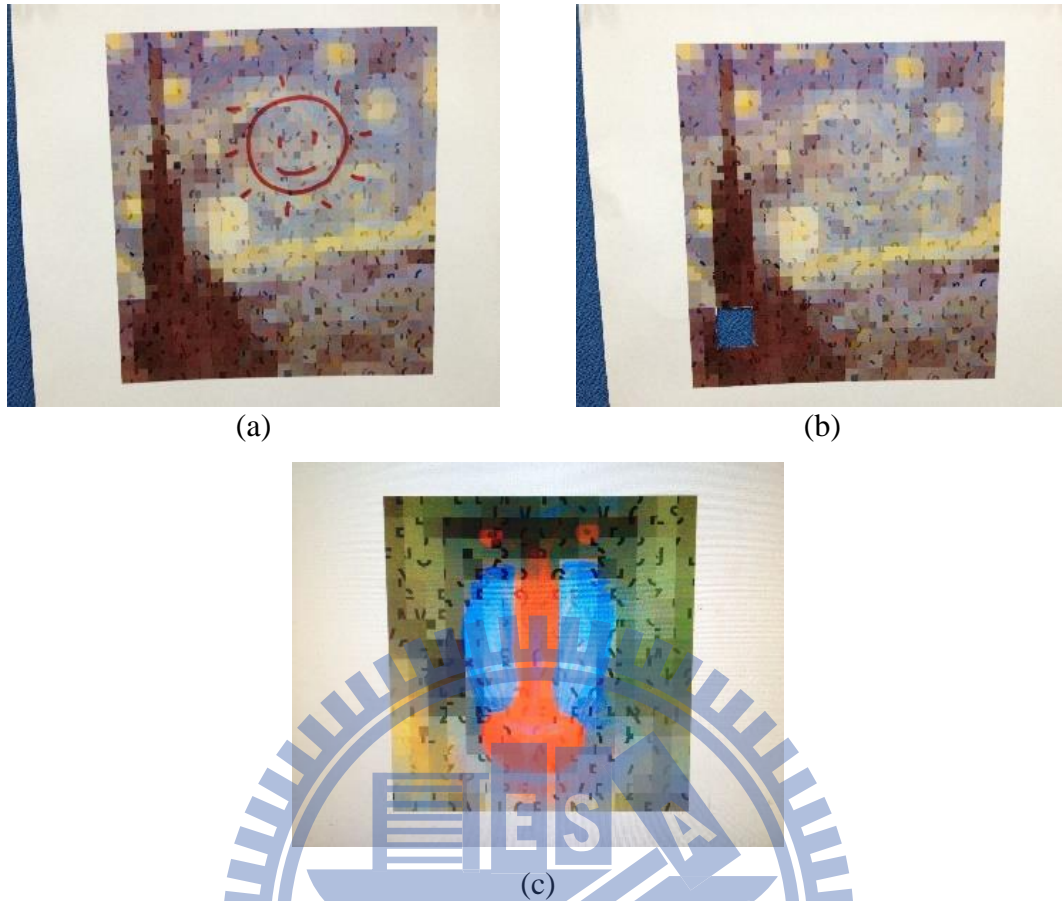


Figure 5.9. Robustness of proposed method. (a) A captured message-rich character image under defacement attack. (b) A captured message-rich character image under another defacement attack. (c) A message-rich character image captured from a monitor screen.

5.6 Summary

A new data hiding technique for AIDC applications via message-rich character image has been proposed, which is created from a target image for use as a carrier of a given message. The artistic favor of the target image is kept in the created image, achieving pervasive communication. Comparing with other AIDC tools like QR codes and hardcopy image barcodes, the proposed message-rich character image has several merits: (1) the image can not only be printed on papers but also be displayed on screens for various uses; (2) the image can endure more distortions like perspective transformation, noise, screen blurring, etc.; (3) the message can be extracted from an image captured by a mobile phone (this is not the case for the hardcopy image barcode [17]-[19]); (4) by utilizing the power of OCR, the image can endure more

serious attacks, such as partial defacement, image taking from screens, etc. (again, this is not the case for the hardcopy image barcode); (5) if message extraction from the message image by machine is not necessary to carry out, humans can still read the information appearing in the extracted message image because it is composed of characters, and so meaningful and readable. Experimental results show the feasibility of the proposed method.



Chapter 6

A New Data Hiding Technique via Message-rich code Image for Automatic Identification and Data Capture Applications

6.1 Introduction

In Chapter 5, we proposed a new data hiding technique via message-rich character image for automatic identification and data capture to realize pervasive communication on hard copies of images. It is created from a target image used as a carrier of a given message by fragmenting the shapes of the composing *characters* of the message and “injecting” the resulting character fragments randomly into the target image by a block luminance modulation scheme. Each message-rich character image so created has the visual appearance of the corresponding pre-selected target image while conventional barcodes [20]-[23] do not. Also, the data embedded by the method presented in Chapter 5 can be extracted from a “camera-captured” version of the created message-rich character image while those embedded by the use of the aforementioned hardcopy data hiding methods using image barcodes cannot. The function may be implemented on a mobile device.

However, as shown in Figure 6.1(b), each message-rich character image generated by the method in Chapter 5 contains many small character fragments with undesired visual effects. Also, it requires an optical character recognition scheme to extract the embedded message, which is usually time-consuming. Also, the size of each block cannot be too small in order to keep the resolution in the captured image sufficiently good for correct extraction of the character shapes in the image. To solve these problems, another new type of message-rich multimedia, called *message-rich code image*, is proposed in this study. Specifically, instead of transforming the given message to be embedded into a character message image, the message is converted, in the sense of *data coding*, into a bit stream of codes first, which is then represented by binary *pattern blocks*, each being composed of 2×2 *unit blocks*. A block luminance modulation scheme is then applied to each pattern block to yield a message-rich code image with the visual appearance of a pre-selected target image. An example of the

resulting message-rich code image is shown in Figure 6.1(c), which is more pleasing than the message-rich character image shown in Figure 6.1(b) generated by the method in Chapter 5 . A more detailed comparison with the method in Chapter 5 by experiments reveals the following additional merits of the proposed method in this chapter: (1) the yielded message-rich code image has a much better visual appearance of the target image; (2) the accuracy rate of message extraction from the generated code image is higher; (3) the message extraction speed is higher.

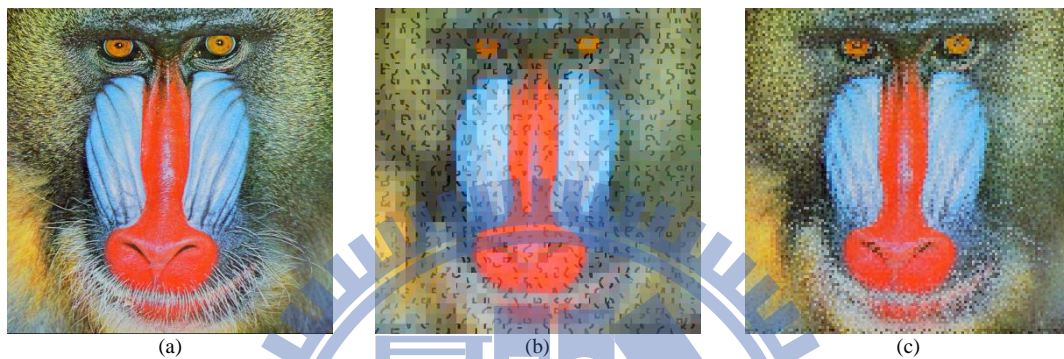


Figure 6.1. Examples of message-rich images yielded by the method in Chapter 5 and proposed method. (a) Target image. (b) Message-rich character image created by the method in Chapter 5. (c) Message-rich code image created by proposed method.

6.2 Idea of Proposed Method

The proposed method includes two main phases of works as illustrated in Figure 6.2: 1) message-rich code image generation; and 2) message extraction. In the first phase, given a target image I_T and a message M , a message-rich code image I_C is created by four major steps.

- Stage 1-1 — transform message M into a bit stream B of codes;
- Stage 1-2 — transform every three bits of B into four bits and represent them by a binary pattern block, resulting in a *pattern image* I_P ;
- Stage 1-3 — modulate each pattern block T_i of I_P by two *representative values* calculated from the Y-channel values of the corresponding block B_i of target image I_T , yielding a *modulated* pattern image I_P' ;
- Stage 1-4 — replace the Y-channel of target image I_T with I_P' to get a message-rich code image I_C as the output.

In the second phase, given a camera-captured version I_c' of a *paper or display copy* of the message-rich code image I_c , the message M is extracted from I_c' by four major steps.

Stage 2-1 — localize the region I_c'' of the original part of the message-rich code image I_c in I_c' ;

Stage 2-2 — correct the geometric distortion in I_c'' incurred in the image acquisition process;

Stage 2-3 — identify the unit blocks in I_c'' automatically and divide I_c'' accordingly into pattern blocks, each with 2×2 unit blocks;

Stage 2-4 — binarize each pattern block of I_c'' , recognize the result to extract the bits embedded in it, compose all the extracted bits to form a bit stream B , and transform B reversely to get the message M .

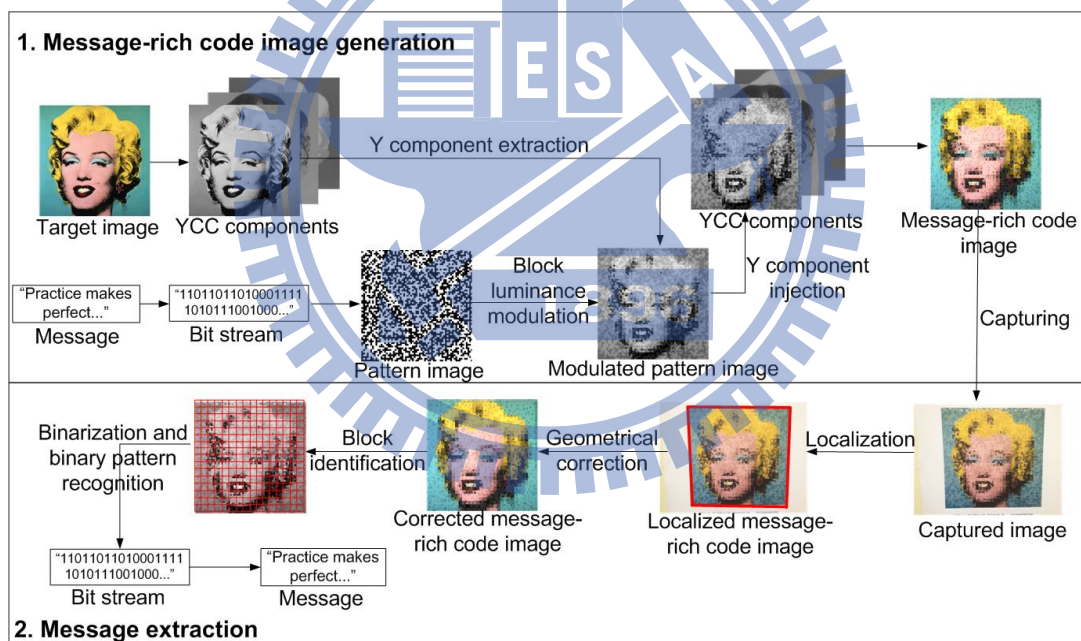


Figure 6.2. Illustration of major steps of two phases of proposed method.

6.3 Generation of Message-rich Code Image

6.3.1 Pattern image creation

Unlike the method in Chapter 5 that transforms a message M into a *character message image*, the proposed method in this chapter transforms M into a bit stream B of codes, uses binary *code patterns* to encode the bits of B , and composes the code

patterns, each in the form a *pattern block*, to form a *pattern image* similar in appearance to a pre-selected target image. Specifically, each pattern block T consists of several unit blocks F_i , with each F_i representing a bit of the code pattern C which T represents. A main issue here is how to design the code patterns so that the corresponding pattern blocks are suitable for use not only in message embedding but in block luminance modulation (see Stage 1-3 in this chapter above). To solve this issue, two characteristics must be provided in the designed code patterns: 1) the number of bits in each code pattern C must be small enough, so that the pattern block T representative of C can keep the *local* color characteristic of the corresponding target image area; and 2) the colors of the unit blocks F_i of the pattern block T representative of each code pattern C should not be all the same, since this will cause the original bits represented by the unit blocks of the code patterns *undistinguishable* during message extraction.

The first characteristic mentioned above is necessary for the resulting message-rich code image to become more similar to the pre-selected target image. And as an illustration of the necessity of the second characteristic, Figure 6.3 shows an example of undistinguishable binary code patterns, where the unit blocks F_i of the pattern block T representative of a code pattern C with bits “0000” are all of an identical color originally and are modulated to be all of another color, but then in the message extraction stage, the bits represented by the modulated pattern block cannot be extracted since only one color exists in this modulated pattern block and the bits corresponding to this color cannot be uniquely determined (more details discussed later).

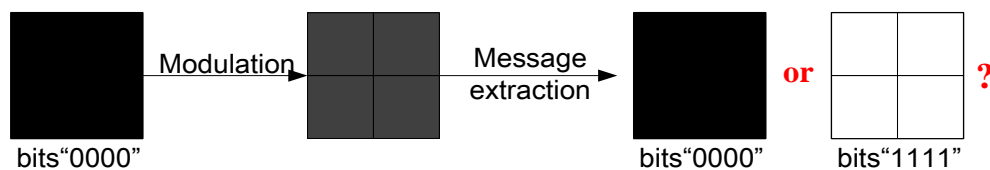


Figure 6.3. An example of undistinguishable binary code patterns.

Therefore, in this study each pattern block representative of a code pattern is set to be of the smallest size of 2×2 unit blocks. Also, a novel *bit expansion* scheme is proposed to expand every three bits of the bit stream B into four ones which are not all

the same in order to satisfy the second required characteristic of the code pattern. In detail, let the bit stream B be denoted as

$$B = b_{11}b_{12}b_{13}b_{21}b_{22}b_{23}b_{31}b_{32}b_{33} \dots b_{n1}b_{n2}b_{n3};$$

and for every three consecutive bits $b_{i1}b_{i2}b_{i3}$ in B , we perform at first a bit expansion operation to get four bits $b_{i1}'b_{i2}'b_{i3}'b_{i4}'$ by the following rule:

$$\text{set } b_{i4}' = \overline{b_{i1} \vee b_{i2} \vee b_{i3}} \quad \text{and } b_{ij}' = b_{ij} \text{ for } j = 1, 2, 3, \quad (28)$$

where \vee and $\overline{\quad}$ denote bitwise “OR” and “complement” operations, respectively. The resulting four bits $b_{i1}'b_{i2}'b_{i3}'b_{i4}'$ will not be all identical, as can be verified by ORing the four bits b_{i1}' through b_{i4}' , leading to the following result:

$$\begin{aligned} b_{i1}' \vee b_{i2}' \vee b_{i3}' \vee b_{i4}' &= (b_{i1} \vee b_{i2} \vee b_{i3}) \vee b_{i4}' \\ &= (b_{i1} \vee b_{i2} \vee b_{i3}) \vee \overline{b_{i1} \vee b_{i2} \vee b_{i3}} = 1, \end{aligned} \quad (29)$$

which means that at least one “1” must exist in $b_{i1}'b_{i2}'b_{i3}'b_{i4}'$, and if all the four bits b_{i1}' through b_{i4}' are “1s,” then according to (28), all the three bits b_{i1} , b_{i2} , and b_{i3} must be “1s” as well, leading to the result $b_{i4}' = \overline{b_{i1} \vee b_{i2} \vee b_{i3}} = 0$, which is a contradiction. Moreover, the total possible number of distinct expanded four bits $b_{i1}'b_{i2}'b_{i3}'b_{i4}'$ for different combinations of $b_{i1}b_{i2}b_{i3}$ is eight, as shown in Figure 6.4. These eight combinations are taken as the code patterns which we mentioned previously.

Message bits $b_{i1}b_{i2}b_{i3}$	Code pattern $b_{i1}'b_{i2}'b_{i3}'b_{i4}'$	Pattern block T_i
000	0001	
001	0010	
010	0100	
011	0111	
100	1000	
101	1011	
110	1101	
111	1110	

Figure 6.4. Performing proposed bit expansion scheme on every three message bits to yield eight binary code patterns represented by pattern blocks.

Next, we create a 2×2 pattern block $T_i = F_{i1}F_{i2}F_{i3}F_{i4}$ with four unit blocks F_{i1} through F_{i4} to represent the non-all-identical bits b_{i1}' through b_{i4}' of each code pattern C_i , where the color of unit block F_{ij} is set to be black if $b_{ij}' = 0$ or white if $b_{ij}' = 1$. Accordingly, as can be seen from Figure 6.4, the colors of the pattern blocks representative of the eight code patterns are all non-identical as well.

Finally, we create a *pattern image* I_P of the size of the target image I_T by arranging all the pattern blocks T_i , say n ones, in a raster-scan order. If the n pattern blocks do not fill up I_P , then we repeat to fill them into I_P again and again until they do. For example, with the target image I_T as shown in Figure 6.5(a) and the bit stream $B = "110110110100011111010111001..."$, the pattern image I_P resulting from such filling operations is shown in Figure 6.5(b).



Figure 6.5 Message-rich code image generation. (a) Target image. (b) Pattern image I_P . (c) Y-channel of (a). (d) Modulated pattern image. (e) Zoom-out of red square region in (d). (f) Resulting message-rich code image.

6.3.2 Block luminance modulation

After the pattern image I_P is created, it is “injected” into the target image I_T under the constraint that the resulting image retains the visual appearance of I_T . For

this, we utilize a characteristic of the YCbCr color model to embed I_P into the Y-channel of I_T . A block luminance modulation technique is used as the same way in Section 5.3.2, which modulates the mean of each pattern block T_i to be the same as that of the corresponding target block B_i of I_T . The resulting modulated pattern image I_P' so has roughly the visual appearance of the Y-component of the target image I_T . For example, Figure 6.5(d) shows a modulated pattern image I_P' so created, which looks like the Y-component of the target image I_T shown in Figure 6.5(c); and Figure 6.5(e) shows a zoom-out version of part of Figure 6.5(d) enclosed by the red rectangle.

The details for block luminance modulation is omitted here, where the detailed steps for block luminance modulation is the same as those in Section 5.3.2. After the pattern image I_P is modulated, the *overall* gray appearance of the modulated pattern image I_P' and that of the Y-component of I_T is roughly the same. Accordingly, we replace the Y-component of I_T with I_P' to generate finally the desired message-rich code image I_C which has the visual color appearance of I_T , as shown by the example seen in Figure 6.5(f).

Later, when conducting message extraction, the message bit stream can be extracted from the Y-component of a captured version of I_C by classifying the pixels of each pattern block into two classes according to their Y values: black and white. However, an issue may occur here as the same one described in Section 5.3.2: if the two representative values r_1 and r_2 are too close, it will be difficult to “separate” them in the classification process. Therefore, an adjustment of the representative values r_1 and r_2 is conducted, resulting in r_1' and r_2' , so that the absolute difference between r_1' and r_2' becomes not smaller than a pre-defined contrast threshold $\delta \geq 0$. For example, Figure 6.6 shows a pattern block resulting from modulations with different values of δ , from which one can see that the two colors in a modulated pattern block will be more easily distinguished when δ is larger. The detail of the proposed representative-value adjustment scheme can be found in Section 5.3.2 so it is omitted here.

6.3.3 Algorithm for message-rich code image creation

Based on the above discussions, a detailed algorithm for message-rich code image creation is described as follows.

Algorithm 6.1. Message-rich code image creation.

Input: a target image I_T , a message M , and a contrast threshold value δ .

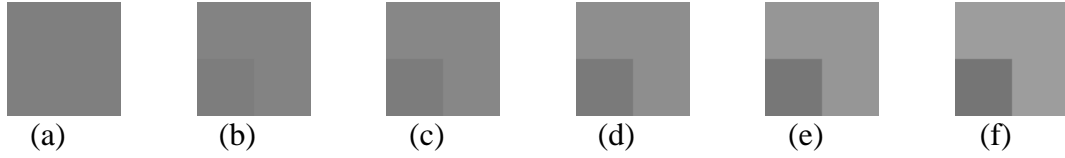


Figure 6.6. Modulated pattern block resulting from uses of different contrast threshold values of δ for the absolute difference between the two adjusted representative values r_1' and r_2' . (a) $\delta = 0$. (b) $\delta = 5$. (c) $\delta = 10$. (d) $\delta = 20$. (e) $\delta = 30$. (f) $\delta = 40$.

Output: a message-rich code image I_c .

Steps:

Stage 1 — transforming the message into a bit stream.

Step 1. Transform message M into a bit stream B .

Stage 2 — generating the pattern image.

Step 2. Split B into n three-bit segments as $b_{11}b_{12}b_{13}b_{21}b_{22}b_{23} \dots b_{n1}b_{n2}b_{n3}$.

Step 3. Expand every three bits $b_{i1}b_{i2}b_{i3}$ in B into four bits $b_{i1}'b_{i2}'b_{i3}'b_{i4}'$ according to (28) and generate the corresponding pattern block T_i according to the rules shown in Figure 6.4.

Step 4. Align all the generated pattern blocks T_i in a raster-scan order to form a pattern image I_P of the size of target image I_T , with each side having N_T patterns; and if the result does not fill up I_P , repeat the filling until it becomes so.

Stage 3 — modulating the pattern image.

Step 5. Divide the Y-component of target image I_T into *target blocks* $\{B_1, B_2, B_3, \dots, B_N\}$ where $N = N_T \times N_T$.

Step 6. For each pattern block T_i in pattern image I_P , generate a modulated pattern block T_i'' as follows.

- (a) Compute two representative values r_1 and r_2 of the corresponding target block B_i according to (14).
- (b) Compute $\delta_0 = |r_2 - r_1|$, and use it and the input contrast threshold δ to obtain two adjusted representative values r_1' and r_2' from r_1 and r_2 according to (20) and (23).
- (c) For each pixel P_t in T_i , if P_t is black, set the value p_t'' of the corresponding pixel P_t'' in T_i'' as $p_t'' = r_1'$; else, set $p_t'' = r_2'$.

Step 7. Compose all the resulting T_i'' to get a modulated pattern image, denoted by I_P' .

Stage 4 — injecting the pattern image into the target image.

Step 8. Replace the Y-component of I_T with I_P' to generate the desired message-rich code image I_C as the output.

6.4 Message Extraction

6.4.1 Localization of message-rich code image and inverse perspective transform

The localization scheme is the same as the one utilized in the previous method, where the description of the localization scheme can be found in Section 5.4.1. In short, we apply the Hough transform and polygonal approximation to find the largest non-white quadrangle Q in I_a as shown by the example seen in Figure 6.7(a). Next, an *inverse perspective transform* is performed on Q to correct the distortion. The result of conducting this on Figure 6.7(a) is shown in Figure 6.7(b). Finally, the Y-component of the resulting Q is taken as an intermediate result, which we call the *captured modulated pattern image* and denote it by I_P'' .

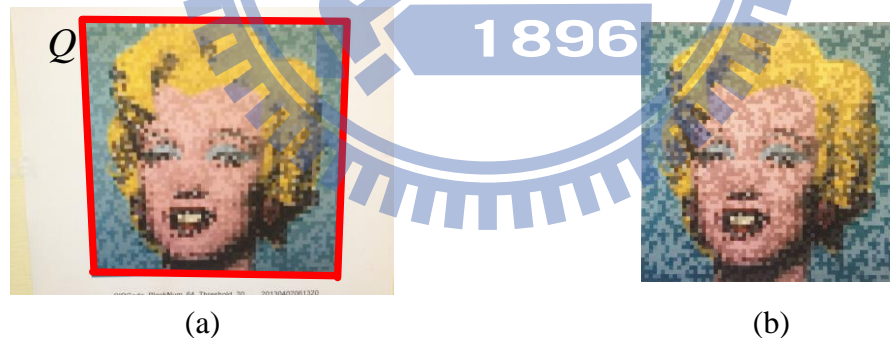


Figure 6.7. Localization and correction of perspective distortion in captured message-rich code image. (a) Localized message-rich code image portion (enclosed by red rectangle). (b) Result of perspective distortion correction applied to red portion region in (a).

6.4.2 Block number identification and block segmentation

To identify the unit blocks in I_P'' in order to apply binarization and pattern recognition to them, an idea similar to the Hough transform is adopted, which uses the

statistics of the pixels' gradient values to *guess* the number N_s of unit blocks in the horizontal or vertical direction in I_P'' because those pixels on the splitting lines between the unit blocks usually have larger gradient values. In more detail, at first the gradient value g_{xy} of each pixel R_{xy} with value r_{xy} at coordinates (x, y) in I_P'' is computed by (24) in Section 5.4.2: Next, for each *possible* value n_j of N_s , the distance d_j between the splitting lines of every two possible adjacent unit blocks is computed as $d_j = L/n_j$ where L is the side length of the square-shaped I_P'' . Then, the horizontal or vertical lines separated by the distance of d_j are taken as *candidate* splitting lines, where the positions of these candidate splitting lines described by image coordinates are computed by (25) in Section 5.4.2. Also, the *average gradient value* AG_{n_j} of the pixels on each candidate spitting line is computed by (26) in Section 5.4.2:

Note that in addition to the actual value N_s which will yield a large average gradient value, the values n_j which are divisors of N_s will yield large average gradient values as well. For this, the value n_j yielding the *largest* average gradient value AG_{n_j} , denoted as AG_o , is selected first and those n_j yielding average gradient values AG_{n_j} *close* to AG_o are selected as well. Then, the largest n_j from these selected values of n_j is taken as the desired number N_s of blocks of I_P'' in the horizontal or vertical direction, and division of I_P'' into unit blocks is conducted accordingly.

For example, Figure 6.8(a) shows a captured modulated pattern image I_P'' , Fig. Figure 6.8(b) is the image of the computed gradient values, and Figure 6.8(c) illustrates the average gradient values for different values of N_s , where the n_j yielding the largest average gradient value is seen to be 16 (indicated by the black arrow). Also, the n_j 's, which yield the average gradient values close to the largest average gradient value 16, are seen to be 32 and 64 (indicated by the green and red arrows, respectively), and the n_j yielding the largest gradient value among the three selected ones is 64. Therefore, the desired N_s is taken to be 64, and the corresponding image division result is shown in Figure 6.8(d).

6.4.3 Binarization and recognition of pattern blocks

After the captured modulated pattern image I_P'' are segmented into unit blocks, we try to recover the pattern blocks in pattern image I_P by grouping every four mutually-connected unit blocks as a pattern block since the size of a pattern block is 2×2 . The number of pattern blocks in the horizontal or vertical direction in I_P'' is so just $N_T = N_s/2$. Subsequently, the moment-preserving thresholding technique [73] is

applied to each pattern block T_i''' to binarize it automatically. And the four *unit blocks* in each resulting pattern block T_i''' are denoted as F_{i1}' , F_{i2}' , F_{i3}' , and F_{i4}' , respectively.

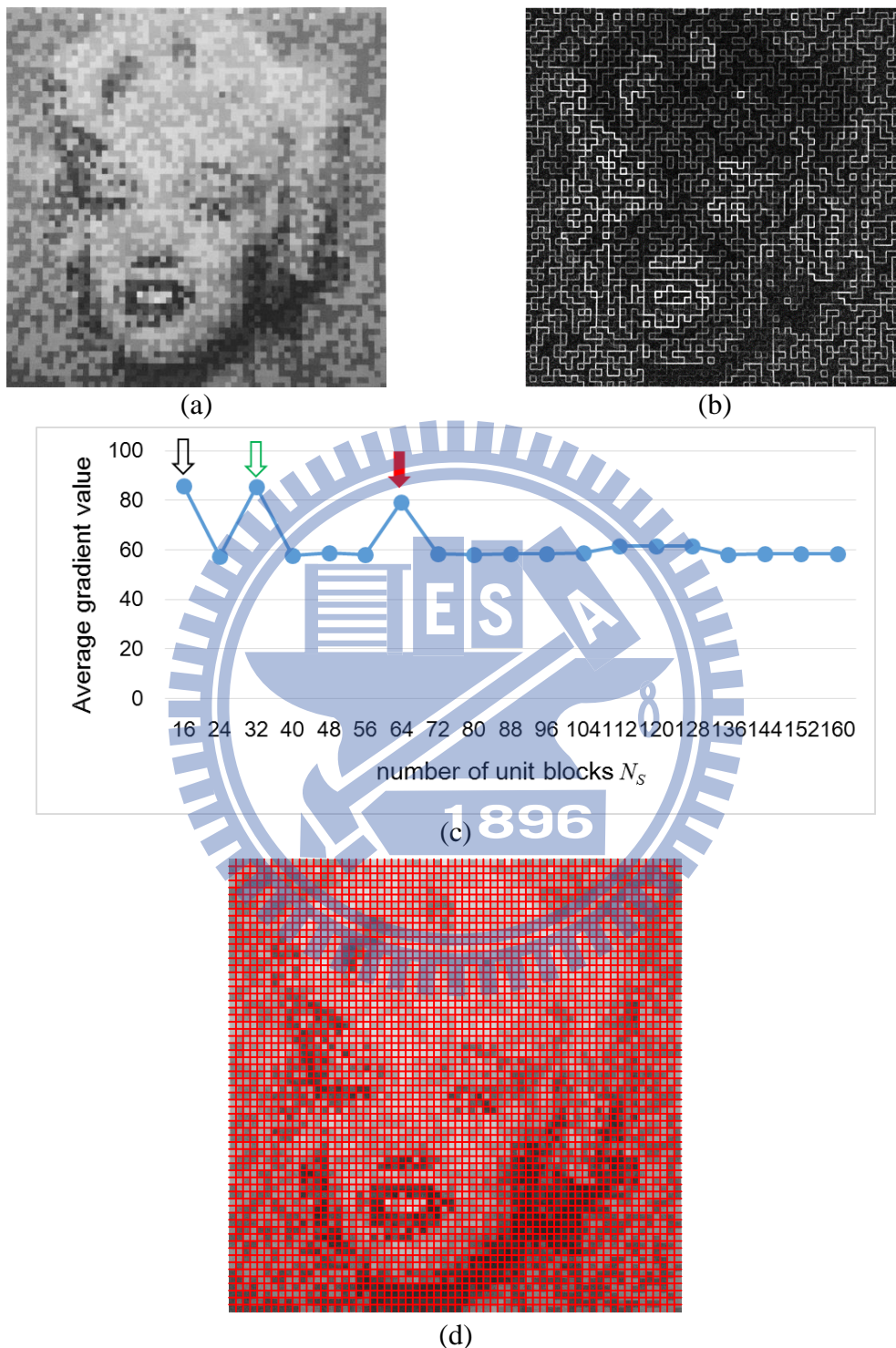


Figure 6.8. Block number identification. (a) Captured modulated pattern image I_p'' . (b) Gradient values of (a). (c) Average gradient values of pixels on candidate spitting lines for different N_s . (d) Image division result according to determined number of unit blocks, $N_s = 64$.

Next, how to classify each T_i''' as one of the eight possible *code patterns*, which we denote as BP_k with $k = 1 \sim 8$, as shown in Figure 6.4 is the issue now. This is an eight-class pattern classification problem. To solve it, we use a *minimum absolute distance classifier*. Specifically, each possible code pattern BP_k has four unit blocks, say denoted as F_{k1}'' through F_{k4}'' , and the color of each F_{kj}'' is either black or white. Hence, we may utilize the *feature of blackness* to describe F_{kj}'' ; that is, if the color of the unit block F_{kj}'' is black, then we take the blackness feature bf_{kj} of F_{kj}'' to be “1”; else, to be “0.” Next, we compute the *real* blackness feature bf_{ij} of each unit block F_{ij}' in T_i''' by:

$$bf_{ij} = NB_{ij}/(NB_{ij} + NW_{ij}), \quad (30)$$









where NB_{ij} and NW_{ij} are the numbers of black and white pixels in unit block F_{ij}' , respectively. Then, the *absolute distance* AD_{ik} of the blackness feature between T_i''' and BP_k can be computed as:

$$AD_{ik} = \frac{1}{4} \sum_{j=1}^4 |bf_{ij} - bf_{kj}|. \quad (31)$$

Subsequently, the code pattern BP_m with the minimum absolute distance AD_{im} is selected as the result of classifying the pattern block T_i''' . For example, let the blackness features of unit blocks F_{i1}' through F_{i4}' of a pattern block T_i''' be $bf_{i1} = 0.9$, $bf_{i2} = 0.13$, $bf_{i3} = 0.22$, $bf_{i4} = 0.12$, respectively. The absolute distances AD_{ik} of T_i''' to all the eight possible code patterns BP_k with $k = 1 \sim 8$ are shown in Table 6.1. And the code pattern with the minimum absolute distance is BP_4 with $AD_{i4} = 0.1425$. So, the corresponding four bits b_{i1}' through b_{i4}' of the pattern block T_i''' are “0111.” Finally, we extract the original three message bits b_{i1} , b_{i2} , and b_{i3} as “011” based on b_{i1}' through b_{i4}' by the rule $b_{ij} = b_{ij}'$ for $j=1, 2$, and 3 according to (28).

An example of results yielded by the above message extraction process is shown in Figure 6.9, where a captured modulated pattern image I_P'' is shown in Figure 6.9(a), which, after being binarized, results in Figure 6.9(b); the result of code-pattern classification is shown in Figure 6.9(c); and the final extracted bit stream are shown in Figure 6.9(d). These results show that the proposed code-pattern classification scheme corresponding to the minimum absolute distance criterion works correctly for the purpose of embedded message extraction. More experimental results will be presented later to prove this statement.

Table 6.1. An example of code pattern recognition.

binary code pattern	Corresponding 4 bits	Absolute distance AD_{ik}
	0001	$AD_{i1} = (0.1+0.87+0.78+0.12)/4 = 0.4675$
	0010	$AD_{i2} = (0.1+0.87+0.22+0.88)/4 = 0.5175$
	0100	$AD_{i3} = (0.1+0.13+0.78+0.88)/4 = 0.4725$
	0111	$AD_{i4} = (0.1+0.13+0.22+0.12)/4 = 0.1425$
	1000	$AD_{i5} = (0.9+0.87+0.78+0.88)/4 = 0.8575$
	1011	$AD_{i6} = (0.9+0.87+0.22+0.12)/4 = 0.5275$
	1101	$AD_{i7} = (0.9+0.13+0.78+0.12)/4 = 0.4825$
	1110	$AD_{i8} = (0.9+0.13+0.22+0.88)/4 = 0.5325$

6.4.4 Message extraction algorithm

Algorithm 6.2. Message extraction.

Input: a captured version I_d of a message-rich code image.

Output: the message M embedded originally in I_d .

Steps:

Stage 1 — localizing the message-rich code image.

Step 1. Find the largest non-white quadrangle Q in I_d by the Hough transform and polygonal approximation.

Stage 2 — correcting geometric distortion.

Step 2. Perform an inverse perspective transform on Q to correct the perspective distortion and take the Y-component of Q as the captured modulated pattern image I_P'' .

Stage 3 — identifying pattern blocks in the code image.

Step 3. Compute the gradient value g_{xy} of each pixel R_{xy} in I_P'' according to (24).

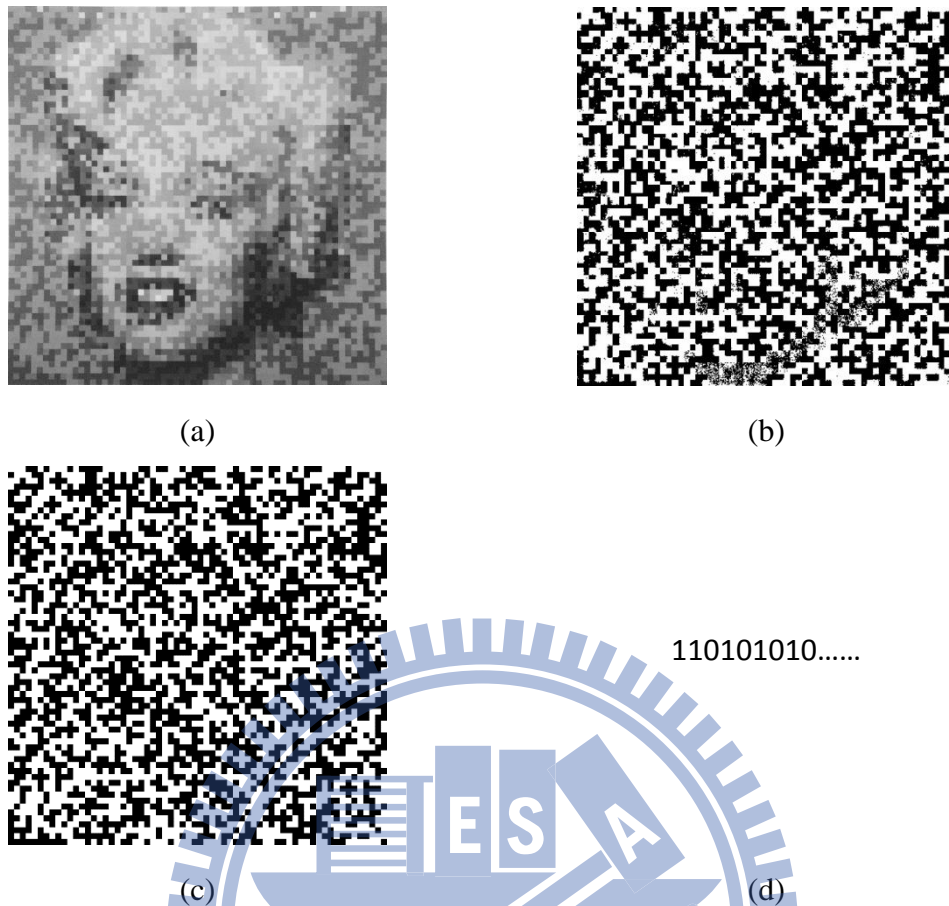


Figure 6.9. Binarization and code-pattern recognition. (a) Captured modulated pattern image. (b) Binarization result of (a). (c) Result of code-pattern recognition of (b). (d) Extracted message.

Step 4. For each possible value n_j of N_s , compute the average gradient values AG_{n_j} of the pixels on each candidate spitting line according to (26).

Step 5. Select the value n_j yielding the largest AG_{n_j} , denoted as AG_o , and those n_j 's yielding AG_{n_j} close to AG_o ; pick the largest n_j from all the selected n_j 's for use as the desired number N_s of blocks of I_P'' in the horizontal or vertical direction; and divide I_P'' accordingly into unit blocks.

Stage 4 — binarizing the pattern blocks to extract the message.

Step 6. Group every four mutually-connected unit blocks and denote them as F_{i1}' through F_{i4}' to form a pattern block T_i''' in I_P'' .

Step 7. Extract three bits from each pattern block T_i''' by the following steps.

- (a) For each unit block F_{ij}' in T_i''' , compute its blackness feature bf_{ij} according to (30).

- (b) Computing the absolute distance AD_{ik} of T_i''' to each of the eight possible code patterns BP_k shown in Figure 6.4 according to (31).
- (c) Select the code pattern BP_m with the minimum absolute distance and take the corresponding four bits of BP_m as the recognized four bits b_{i1} through b_{i4}' of T_i''' .
- (d) Extract the original three message bits b_{i1}, b_{i2}, b_{i3} as $b_{i1}', b_{i2}', b_{i3}'$, respectively, according to (28).

Step 8. Concatenate the extracted bits into a bit stream B and transform reversely B to get the embedded message M .

6.5 Experimental Results

The proposed method was implemented on a 3.0GHz PC with a Core i7 CPU and 8G RAM using the language Microsoft C#.NET, and generated message-rich code images were captured with an iPhone 4S and analyzed to extract the embedded messages in a series of experiments. Corresponding statistics were plotted as well to show the accuracy of the extracted messages using different parameters including: (1) the contrast threshold δ for the minimum difference between the representative values r_1' and r_2' ; and (2) the number of unit blocks N_s used in the horizontal or vertical direction of the created pattern image. Figures 6.10(a), 6.10(c), and 6.10(e) show three test target images used in the experiments. The corresponding message-rich code images generated with parameters $N_s = 128$ and $\delta = 40$ are shown in Figures 6.10(b), 6.10(d), and 6.10(f), respectively. These images were all printed to be of the same size of 127×127 mm.

One of the parameters that influence the accuracy of the extracted message is the contrast threshold value δ for the minimum difference between the two representative values r_1' and r_2' . If δ is too small, r_1' and r_2' will be too close so that the extracted message might be wrong. Figure 6.12(a) illustrates the accuracy rates of message extraction with $\delta = 0, 20, 40,$ and 60 , which shows that the larger the value of δ , the higher the accuracy rate of the extracted message; when $\delta > 40$, an accuracy of 99.8% is reached; and when $\delta > 60$, an accuracy of 100% is reached. Figure 6.12(b) shows that the larger the value of δ , the larger the RMSE of the resulting message-rich code image with respect to the target image. So there is a tradeoff between achieving higher message extraction accuracy and obtaining better visual quality in the generated code

image. Figure 6.11 shows some code images created with different contrast threshold values of δ , where the target image is Figure 6.10(e) and $N_S = 64$. As can be seen, Figure 6.11(a) has the best visual appearance when compared with the others, but has the lowest message extraction accuracy for only 85.60%, because the two representative values are too close (so that the colors of most regions in Figure 6.11(a) look like the same).



Figure 6.10. Created message-rich code images. (a), (c), and (e) Target images. (b), (d) and (f) Resulting message-rich code images with $N_S = 128$ and $\delta = 40$.

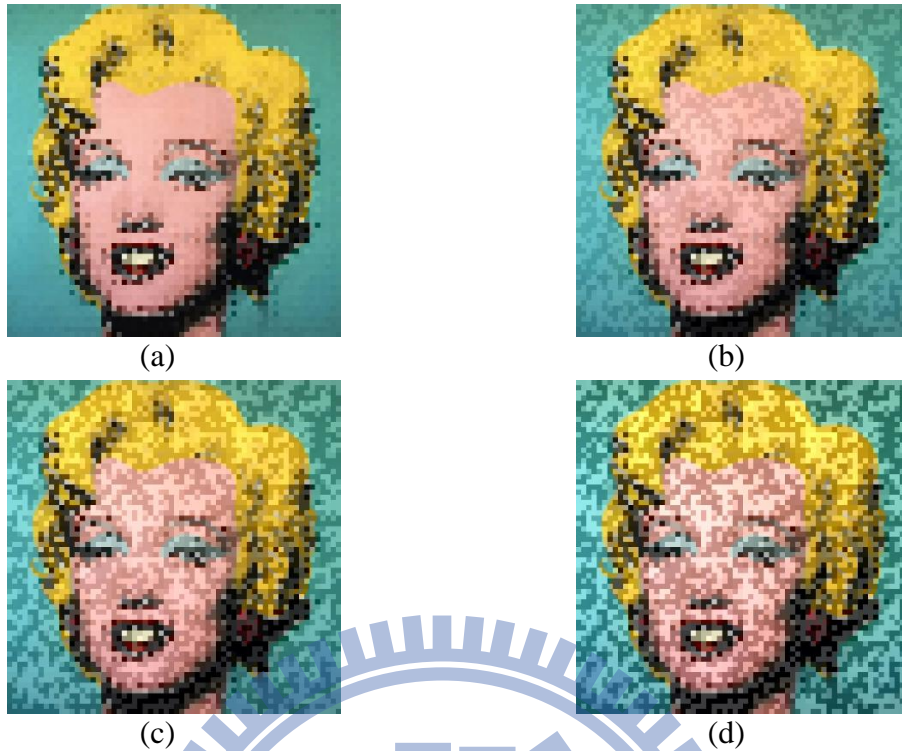


Figure 6.11. Created message-rich code images with different contrast threshold values of δ , where $N_s = 64$. (a) Resulting message-rich code image with RMSE = 66.35 and accuracy rate = 85.60%, where $\delta = 0$. (b) Resulting code image with RMSE = 66.57 and accuracy rate = 98.97%, where $\delta = 20$. (c) Resulting code images with RMSE = 68.47 and accuracy rate = 100%, where $\delta = 40$. (b) Resulting images with RMSE = 72.27 and accuracy rate = 100%, where $\delta = 60$.

Another parameter that influences the message extraction accuracy is the number N_s of unit blocks in the horizontal or vertical direction in the created pattern or code image. The larger the value of N_s , the larger the message embedding capacity of the created code image, yet the smaller the size of the unit block and so the lower the message extraction accuracy. This can be seen from Figure 6.12(c), where when $N_s = 16$, the accuracy of 100% is reached; when $N_s = 64$, the accuracy of 99.76% is reached; and when $N_s = 128$, the lower accuracy of 96.31% is yielded. Figure 6.13 shows some message-rich code images generated with different values of N_s with Figure 6.10(a) as the target image and the contrast threshold $\delta = 40$. As can be seen, when N_s is larger, the visual appearance of the created image is better with a smaller RMSE, but the message extraction accuracy is lower. Specifically, the accuracy of Figure 6.13 (d) is 99.11%, instead of 100% which is reached by the other three cases.

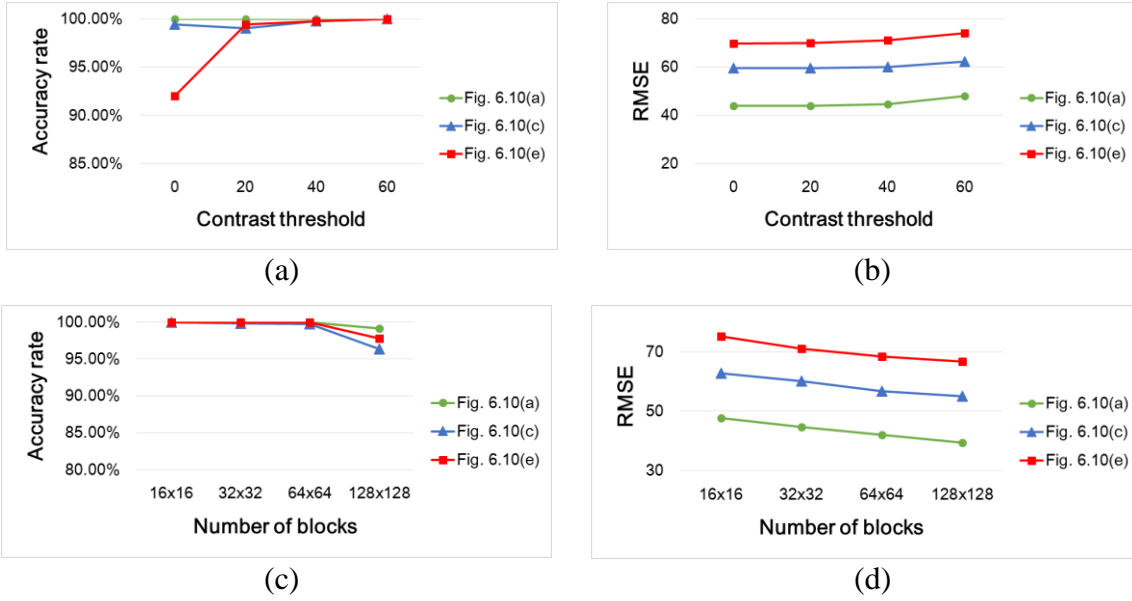


Figure 6.12. Plots of trends of results using various parameters. (a) Accuracy rates of extracted messages with different contrast threshold values δ , with #unit blocks $N_S = 32$. (b) RMSE values of created message-rich code images with respect to target images for different contrast threshold values of δ , with #unit blocks $N_S = 32$. (c) Accuracy rates of extracted messages with different #unit blocks N_S with contrast threshold $\delta = 40$. (d) RMSE values of created message-rich code images with respect to target images with different #unit blocks N_S and contrast threshold $\delta = 40$.

Table 6.2 shows a comparison of the results of the proposed method via message-rich code images in this chapter and those of the method via message-rich character image in Chapter 5 with the target images as shown in Figure 6.10 against different numbers N_S of unit blocks in the horizontal or vertical direction in the created pattern or message images. As can be seen from the table, the proposed method yields higher message extraction accuracy than the method via message-rich character image in Chapter 5, e.g., when $N_S = 32$, the message extraction accuracy yielded by the proposed method reaches 99.80% while that yielded by the method via message-rich character image in Chapter 5 is only 88.28%. Moreover, when $N_S = 64$, the message extraction accuracy of 99.76% yielded by the proposed method is *much higher* than that yielded by the method via message-rich character image in Chapter 5, which is only 34.70%.

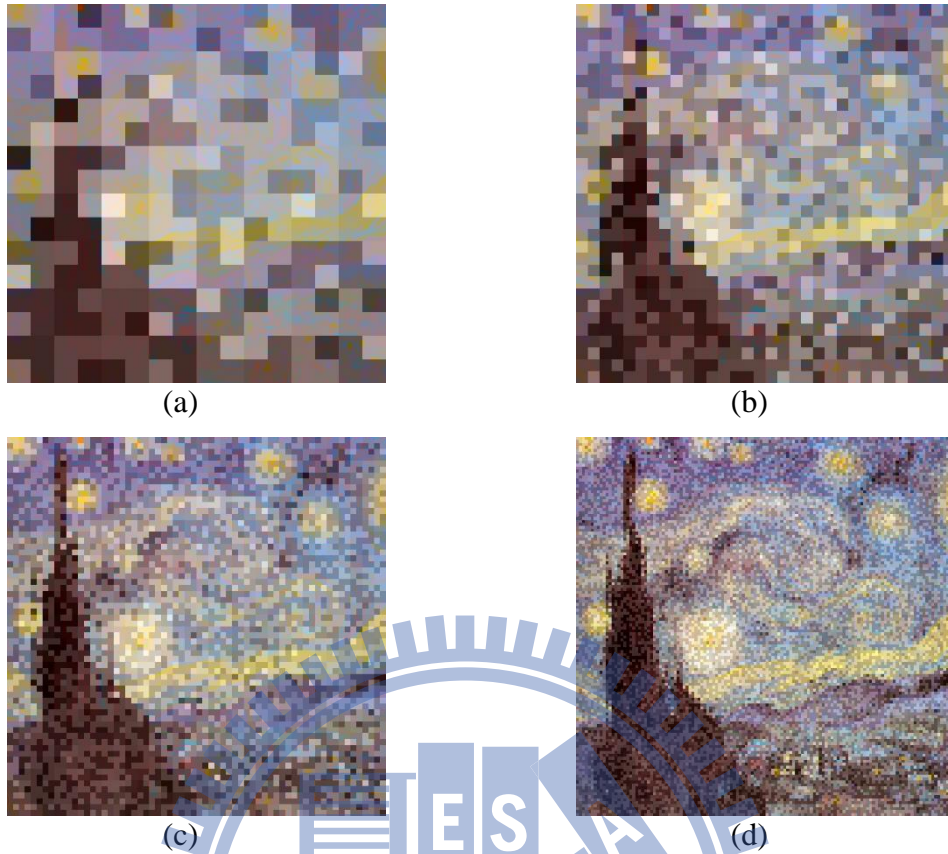


Figure 6.13. Created message-rich code images with different #unit blocks N_s , where contrast threshold value $\delta = 40$. (a) Resulting message-rich code image with $RMSE = 47.66$ and accuracy rate = 100%, where $N_s = 16$. (b) Resulting message-rich code image with $RMSE = 44.63$ and accuracy rate = 100%, where $N_s = 32$. (c) Resulting message-rich code image with $RMSE = 42.05$ and accuracy rate = 100.00%, where $N_s = 64$. (d) Resulting message-rich code image with $RMSE = 39.43$ and accuracy rate = 99.11%, where $N_s = 128$.

Also, Figure 6.14 shows the resulting binarized captured message-rich images of the proposed method via message-rich code image and the method via message-rich character image in Chapter 5, where the value of N_s is 32 in Figures 6.14(a) and 6.14(b) and 64 in Figures 6.14(c) and 6.14(d). As can be seen from Figure 6.14(a), with $N_s = 32$ the characters in the binarized captured message-rich character image created by the method in Chapter 5 are still clear enough so that the message extraction accuracy yielded with Figure 6.14(a) as the input is still high, reaching 98.61%. However, with $N_s = 64$, as seen from Figure 6.14(c), the characters in the binarized captured message-rich character image created by the method in Chapter 5

become undistinguishable so that the message extraction accuracy yielded with Figure 6.14(c) as the input becomes worse, only 41.25%. Furthermore, as seen from Figures 6.14(b) and 6.14(d), the pattern blocks in the binarized captured message-rich code images created by the proposed method are both clear enough so that the message extraction accuracy rates yielded by them are both still high, reaching 99.80% and 99.76%, respectively.

In addition, we compare the times consumed by the code-pattern recognition steps in the proposed method and the method in Chapter 5. As can be seen from Table 6.2, the recognition time used by the proposed method in this chapter is much less than that used by the method in Chapter 5. This is owing to the time-consuming OCR operation conducted by the method in Chapter 5 on every character image, which computes the similarity of the character image with each possible character image in the database and selects the most similar one as the recognition result. In contrast, the proposed method only needs to recognize each pattern block as coming from one of eight possible code-pattern classes by computing the absolute distances of the pattern block to the eight classes and selecting the one with the minimum absolute difference.

Table 6.2. Comparison of results of proposed method in this chapter and method in Chapter 5 with $\delta = 40$.

Target image	N_s	Method	Accuracy rate (%)	Recognition time (ms)
Fig. 6.10(a)	16	Message-rich code images	100	60
		Message-rich character images	100	1186
	32	Message-rich code images	100	62
		Message-rich character images	96.53	1812
	64	Message-rich code images	100	93
		Message-rich character images	40.86	3045
Fig. 6.10(c)	16	Message-rich code images	100	68
		Message-rich character images	100	1590
	32	Message-rich code images	99.80	63
		Message-rich character images	98.61	1696
	64	Message-rich code images	99.76	83
		Message-rich character images	41.25	2263
Fig. 6.10(e)	16	Message-rich code images	100	54
		Message-rich character images	100	1230
	32	Message-rich code images	99.80	55
		Message-rich character images	88.28	1697
	64	Message-rich code images	100	82
		Message-rich character images	34.70	2315

As a summary, the proposed method has the following merits with respect to the method in Chapter 5: (1) the yielded message-rich code image has a better visual appearance since a larger number N_s of unit blocks can be utilized in the proposed method; (2) the message extraction accuracy is higher since much less details are contained in a unit block of the proposed method; (3) the message extraction speed is higher since classification of only eight classes need be conducted to extract the corresponding four bits of each binarized pattern block.

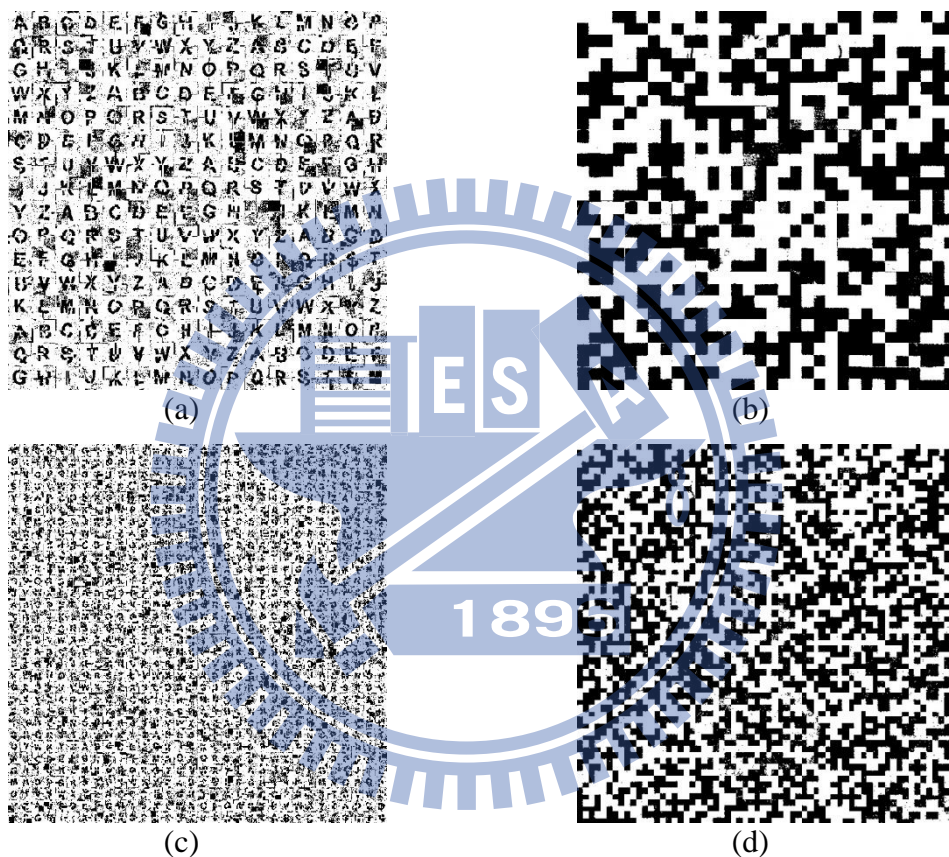


Figure 6.14. Binarized captured message-rich images created by method in Chapter 5 and proposed method in this chapter and respective message extraction accuracy rates, where the target image of these resulting images is Figure 6.10(c). (a) Binarized image by method in Chapter 5 with $N_s = 32$ and accuracy rate = 98.61%. (b) Binarized image by proposed method in this chapter with $N_s = 32$ and accuracy rate = 99.80%. (c) Binarized image by method in Chapter 5 with $N_s = 64$ and accuracy rate = 41.25%. (d) Binarized image by proposed method in this chapter with $N_s = 64$ and accuracy rate = 99.76%.

Furthermore, as shown in Figure 6.4, only eight binary code patterns are utilized by the proposed bit expansion scheme. However, the possible number of all code patterns is 16 since the possible number of combinations of the four expanded bits is $2^4 = 16$. Also, only two of the possible 16 code patterns will cause the undistinguishable problem as illustrated in Figure 6.3. Hence, 14 code patterns can be utilized in the bit expansion scheme. For this, as shown in Figure 6.15, we conducted an experiment to test another bit expansion scheme, in which the first three bits $b_{i1}'b_{i2}'b_{i3}'$ are taken to be the same as the original bits $b_{i1}b_{i2}b_{i3}$ and the fourth bit b_{i4}' is decided by the brightness of the lower right sub-block B_{i4} in its corresponded target block B_i when $b_{i1}'b_{i2}'b_{i3}'$ are not all the same. Specifically, if the mean value of the pixels in B_{i4} is larger than that of the pixels in B_i , then we assign b_{i4}' to be “1”; else, we assign b_{i4}' to be “0.” Figure 6.16 shows the results generated by using the two different bit expansion schemes with Figure 6.10(c) as the target image. Figures 6.16(a) and 6.16(b) show the created pattern images by using the original bit expansion scheme and the new bit expansion scheme just described, respectively. And Figures 6.16(c) and 6.16(d) show the created message-rich code images corresponding to Figures 6.16(a) and 6.16(b), respectively, where Figure 6.16(d) shows that the created message-rich code image by using the new bit expansion scheme has a smaller RMSE value.

Message bits $b_{i1}b_{i2}b_{i3}$	Code pattern $b_{i1}'b_{i2}'b_{i3}'b_{i4}'$		Pattern block T_i
000	0001		
001	0010	0011	
010	0100	0101	
011	0110	0111	
100	1000	1001	
101	1010	1011	
110	1100	1101	
111	1110		

Figure 6.15. Performing another bit expansion scheme on every three message bits to yield 14 binary code patterns represented by pattern blocks.

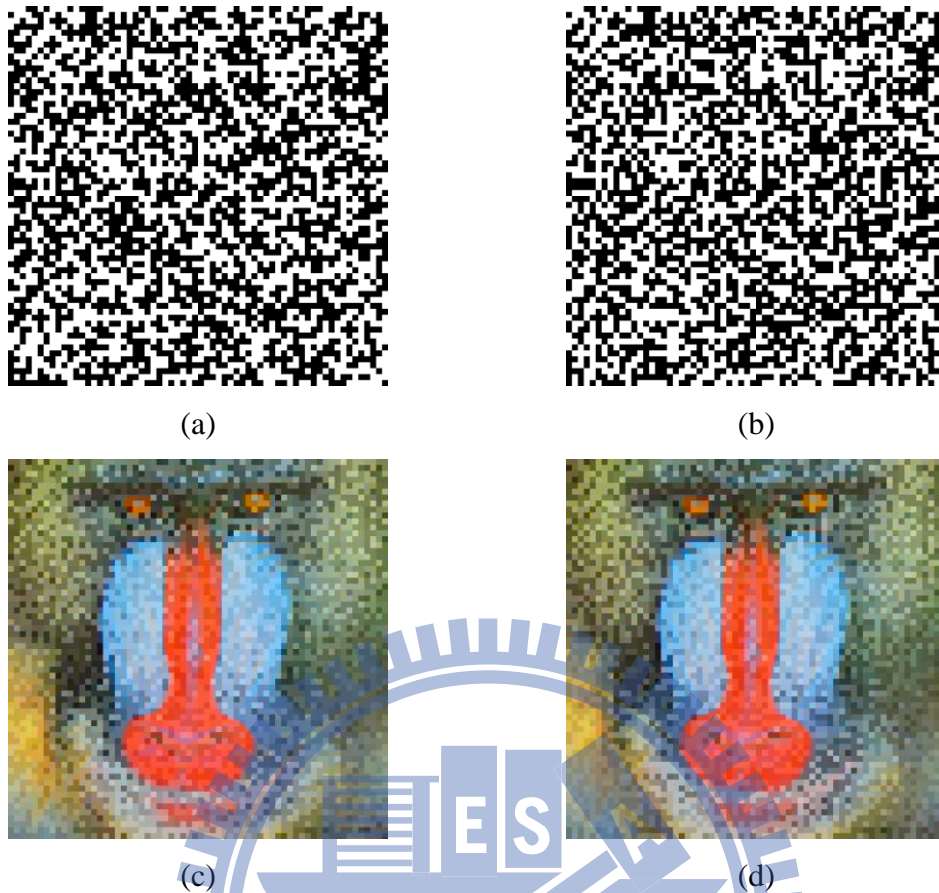
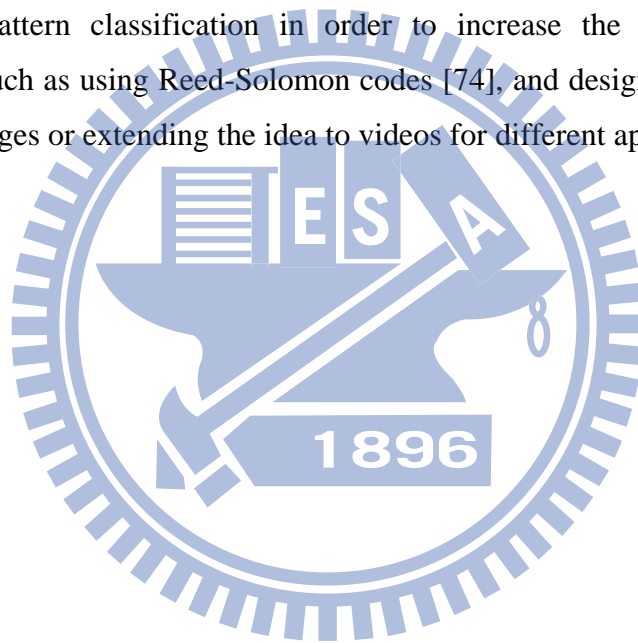


Figure 6.16. Results yielded by using two different bit expansion schemes with $N_s = 64$ and $\delta = 20$. (a) Pattern image yielded by the original bit expansion scheme. (b) Pattern image yielded by the new bit expansion scheme. (c) Message-rich code image yielded by the original bit expansion scheme with $RMSE = 55.97$. (d) Message-rich code image yielded by the new bit expansion scheme with $RMSE = 55.44$.

6.6 Summary

A new data hiding technique via message-rich code image for applications of automatic identification and data capture has been proposed, which is created from a target image for use as a carrier of a given message. The artistic favor of the target image is kept in the created image, achieving pervasive communication. Skillful techniques of code pattern design, unit block segmentation, pattern block classification, etc. have been proposed for message data embedding and extraction. Comparing with other automatic identification and data capture techniques like the use of barcodes and data hiding, automatic identification and data capture using the

proposed message-rich code image has several merits: (1) the image has the visual appearance of any pre-selected target image (this is not the case for the case of using barcodes [17]-[19]); (2) the proposed method can endure more distortions in acquired versions of the code image like perspective transformation, noise, screen blurring, etc. (this is not the case for data hiding [4]-[19]); (3) the message can be extracted from an image captured by a mobile device (this is not the case for data hiding [4]-[19]). Also, the proposed method in this chapter has following additional merits when compared with the method in Chapter 5: (1) the yielded message-rich code image has a better visual appearance; (2) the message data extraction accuracy is higher; (3) the data extraction speed is higher. Experimental results show the feasibility of the proposed method. Further studies may be directed to applying error-correction techniques to the result of code-pattern classification in order to increase the resulting message extraction rate, such as using Reed-Solomon codes [74], and designing more types of message-rich images or extending the idea to videos for different applications.



Chapter 7

Conclusions and Suggestions for Future Studies

In this dissertation study, the new concept of message-rich multimedia is proposed, which enriches human-environment interaction and advances pervasive communication. Various data hiding methods have been designed for creating message-rich multimedia for different pervasive communication applications as described specifically in the following.

- (1) For images with large data volumes, a large-volume data hiding technique to hide these secret images with large data volumes into any target images of the same size has been proposed.
- (2) For encrypted images, also proposed is a data hiding technique based on the techniques of double image encryption and spatial correlation comparison to improve the performance of two previous methods when dealing with flat cover images.
- (3) A text data hiding technique via creations of fake collaboratively-written documents on collaborative writing platforms has been proposed. With this camouflage, an attacker will take the stego-document as a normal collaborative writing work and will not be expected to realize the existence of the hidden message.
- (4) Two data hiding techniques via message-rich character images and message-rich code images for automatic identification and data capture applications have been proposed to realize the innovative idea of pervasive communication on hard copies of images on papers or displays of monitors or TVs. The created image is visually similar to the target image with the function similar to those of barcodes or QR codes, achieving the effect of pervasive communication.

In the following, conclusions of each method and suggestions for future researches are given.

7.1 Conclusions

- (1) A new data hiding method has been proposed, which not only can create meaningful mosaic images but also can transform a secret image into a mosaic

one with the same data size for use as a camouflage of the secret image. By the use of proper pixel color transformations as well as a skillful scheme for handling overflows and underflows in the converted values of the pixels' colors, secret-fragment-visible mosaic images with very high visual similarities to arbitrarily-selected target images can be created with no need of a target image database. Also, the original secret images can be recovered nearly losslessly from the created mosaic images. Future works may be directed to applying the proposed method to images of color models other than the RGB one.

- (2) A new data hiding method based on double image encryptions and refined spatial correlation comparison on encrypted images has been proposed, which solves a problem encountered in the two existing methods [55]-[56] when dealing with flat cover images. This problem comes from the way of flipping the three LSBs of each pixel in part of each block in an encrypted image to embed a message bit. The proposed method improves this by encrypting the four LSBs of each pixel of every block instead of flipping three of them to embed a bit. Also, a refined side-match scheme utilizing the spatial correlations of both recovered and unrecovered blocks has been proposed to decrease the bit-extraction error rate, in contrast with Hong *et al.* [56] which utilizes only those of recovered blocks. Future works may be directed to applying the proposed method for various information hiding purposes.
- (3) A new data hiding method via collaboratively-written articles with simulated revision history records on collaborative writing platforms has been proposed. An input secret message is embedded in the revision history of the resulting stego-document through a simulated collaborative writing process with multiple virtual authors. With this camouflage, people will take the stego-document as a normal collaborative writing work and will not be expected to realize the existence of the hidden message. To generate simulated revisions more realistically, a collaborative writing database was mined from Wikipedia, and the Huffman coding technique was used to encode the mined word sequences in the database according to the statistics of the words. Four characteristics of article revisions were identified, including the author of each revision, the number of corrected word sequences, the content of the corrected word sequences, and the word sequences replacing the corrected ones. Related problems arising in utilizing these characteristics for data hiding have been solved skillfully,

resulting in an effective multi-way method for hiding secret messages into the revision history. Moreover, because the word sequences used in the revisions were collected from a great many of real people's writings on Wikipedia, and because Huffman coding based on usage frequencies is applied to encode the word sequences, the resulting stego-document is more realistic than other text steganography methods, such as word-shift methods [30], non-displayed characters based methods [31], synonym replacement methods [35]-[37], etc.

- (4) A new data hiding technique for automatic identification and data capture applications via message-rich character images has been proposed, which is created from a target image for use as a carrier of a given message. The artistic favor of the target image is kept in the created image, achieving the goal of pervasive communication. Comparing with other AIDC tools like QR codes and hardcopy image barcodes, the proposed message-rich character image has several merits: (1) the image can not only be printed on papers but also be displayed on screens for various uses; (2) the image can endure more distortions like perspective transformation, noise, screen blurring, etc.; (3) the message can be extracted from an image captured by a mobile phone (this is not the case for the hardcopy image barcode [17]-[19]); (4) by utilizing the power of OCR, the image can endure more serious attacks, such as partial defacement, image taking from screens, etc. (again, this is not the case for the hardcopy image barcode); (5) if message extraction from the message image by machine is not necessary to carry out, humans can still read the information appearing in the extracted message image because it is composed of characters, and so meaningful and readable.
- (5) A new data hiding technique via message-rich code image for applications of automatic identification and data capture has been proposed, which is created from a target image for use as a carrier of a given message. The artistic favor of the target image is kept in the created image, achieving pervasive communication. Skillful techniques of code pattern design, unit block segmentation, pattern block classification, etc. have been proposed for message data embedding and extraction. Comparing with other automatic identification and data capture techniques like the use of barcodes and data hiding, automatic identification and data capture using the proposed message-rich code image has several merits: (1) the image has the visual appearance of any pre-selected target image (this is not

the case for the case of using barcodes [17]-[19]); (2) the proposed method can endure more distortions in acquired versions of the code image like perspective transformation, noise, screen blurring, etc. (this is not the case for data hiding [4]-[19]); (3) the message can be extracted from an image captured by a mobile device (this is not the case for data hiding [4]-[19]). Also, the proposed method via message-rich code images has following additional merits when compared with the method via message-rich character images in Chapter 5: (1) the yielded message-rich code image has a better visual appearance; (2) the message data extraction accuracy is higher; (3) the data extraction speed is higher. Experimental results show the feasibility of the proposed method. Further works may be directed to applying error-correction techniques to the result of code-pattern classification in order to increase the resulting message extraction rate, such as using Reed-Solomon codes [74].

7.2 Suggestions for Future Studies

In the future, the following topics may be considered for further studies.

(1) Creating more types of message-rich multimedia —

It is desired to create more types of message-rich multimedia like video and speech to broaden the applications of pervasive communication. For example, we can extend the ideas of message-rich character images and message-rich code images in Chapters 5 and 6, respectively, to videos for different applications. In this way, people can interact with a video, such as to obtain some related information about the video, while they watch it.

(2) Enabling more types of smart devices —

In addition to computers and smart phones, more type of smart devices, such as Google Glasses and smart watches, may be designed to “understand” message-rich multimedia for pervasive communication. For example, people can wear Google Glasses and use the cameras existing on Google Glasses to capture images, create message-rich images, and extract the embedded information in the created images by the techniques proposed in Chapters 5 and 6.

(3) Designing more efficient data hiding methods —

It is desired to design more efficient data hiding methods for uses in the above two topics. For example, for the proposed method in Chapter 3, future works may be directed to analyzing more characteristics of collaborative writing works

or establishing appropriate language models [57]-[59] for more effective data hiding or other applications.

(4) Creating more types of hard copies of message-rich multimedia —

Other than papers or monitor or TV displays, like LED panels, advertisement paintings, etc., may be considered for use in pervasive communication. For example, a company can embed advertisements into its advertisement paintings, and customers can later use mobile devices to obtain the embedded information from the captured versions of the advertisement paintings.



References

- [1] M. Wieser, "The computer for the 21st century," *Scientific American Special Issue on Communications, Computers, and Networks*, pp. 78–89, 1991.
- [2] S. Poslad, "Ubiquitous Computing: Smart Devices, Environments and Interactions," *Wiley*, 2009.
- [3] B. Davis, "Signal rich art: enabling the vision of ubiquitous computing," *Proc. SPIE 7880: Media Watermarking, Security, and Forensics III*, N. D. Memon, J. Dittmann, A. M. Alattar, and E. J. Delp III, Eds., vol. 788002, Feb. 2011.
- [4] F.A.P. Petitcolas, R. J. Anderson, and M. G. Kuhn, "Information hiding—a survey," *Proc. IEEE*, vol. 87, no. 7, pp. 1062–1078, July 1999.
- [5] W. Bender, D. Gruhl, N. Morimoto, and A. Lu, "Techniques for data hiding," *IBM Syst. J.*, vol. 35, no. 3, 4, pp. 313–336, 1996.
- [6] C. K. Chan and L. M. Cheng, "Hiding data in images by simple LSB substitution," *Pattern Recog.*, vol. 37, pp. 469–474, March 2004.
- [7] Z. Ni, Y. Q. Shi, N. Ansari and W. Su, "Reversible Data Hiding," *IEEE Trans. Circuits Syst. & Video Technol.*, vol. 16, no. 3, pp. 354–362, March 2006.
- [8] C. W. Lee and W. H. Tsai, "A lossless large-volume data hiding method based on histogram shifting using an optimal hierarchical block division scheme," *J. of Inform. Sci. & Eng.*, vol. 27, no. 4, pp. 1265–1282, 2011.
- [9] J. Tian, "Reversible data embedding using a difference expansion," *IEEE Trans. Circuits Syst. & Video Technol.*, vol. 13, no. 8, pp. 890–896, Aug. 2003.
- [10] Y. Hu, H. K. Lee, K. Chen, and J. Li, "Difference expansion based reversible data hiding using two embedding directions," *IEEE Trans. Multimedia*, vol. 10, no. 8, pp. 1500–1512, 2008.
- [11] J. Fridrich, M. Goljan, and R. Du, "Invertible authentication," *Proc. SPIE*, vol. 3971, pp. 197–208, 2001.
- [12] C. C. Chang, C. C. Lin, C. S. Tseng, and W. L. Tai, "Reversible hiding in DCT-based compressed images," *Information Sciences*, vol. 177, pp. 2768–2786, 2007.
- [13] S. Lee, C. D. Yoo, and T. Kalker, "Reversible image watermarking based on integer-to-integer wavelet transform," *IEEE Trans. Information Forensics and Security*, vol. 2, no. 3, pp. 321–330, 2007.

- [14] W. H. Lin , S. J. Horng , T. W. Kao , P. Fan , C. L. Lee and Y. Pan, “An efficient watermarking method based on significant difference of wavelet coefficient quantization,” *IEEE Trans. Multimedia*, vol. 10, no. 5, pp.746–757, 2008.
- [15] D. C. Wu and W. H. Tsai, “A steganographic method for images by pixel-value differencing,” *Pattern Recognition Letters*, vol. 24, no. 9-10, pp. 1623–1636, 2003.
- [16] C. H. Tzeng, Z. F. Yang, and W. H. Tsai, “Adaptive data hiding in palette images by color ordering and mapping with security protection,” *IEEE Transactions on Communications*, vol. 52, no. 4, pp. 791–800, 2004.
- [17] O. Bulan, G. Sharma, and V. Monga, “Orientation modulation for data hiding in clustered-dot halftone prints,” *IEEE Trans. Image Processing*, vol. 19, no. 8, pp. 2070–2084, 2010.
- [18] O. Bulan, and G. Sharma, “High capacity color barcodes: per channel data encoding via orientation modulation in elliptical dot arrays,” *IEEE Trans. Image Processing*, vol. 20, no. 5, pp. 1337–1350, 2011.
- [19] N. Damera-Venkata, J. Yen, V. Monga, and B. L. Evans, “Hardcopy image barcodes via block-error diffusion,” *IEEE Trans. Image Processing*, vol. 14, no. 12, pp. 1977–1989, 2005.
- [20] BS ISO/IEC 16388: Information Technology-Automatic Identification and Data Capture Techniques-Code39 Bar Code Symbology Specification, BS ISO/IEC 16388, 2007.
- [21] BS ISO/IEC 15438: Information Technology-Automatic Identification and Data Capture Techniques-PDF417 Barcode Symbology Specification, BS ISO/IEC 15438, 2006.
- [22] BS ISO/IEC 18004: Information Technology-Automatic Identification and Data Capture Techniques-QR Code 2005 Bar Code Symbology Specification, BS ISO/IEC 18004, 2006.
- [23] BS ISO/IEC 16022: Information Technology-Automatic Identification and Data Capture Techniques-Data Matrix Bar Code Symbology Specification, BS ISO/IEC 16022, 2006.
- [24] E. Ouaviani, A. Pavan, M. Bottazzi, E. Brunelli, F. Caselli, and M. Guerrero, “A common image processing framework for 2D barcode reading,” in *7th Int. Conf. on Image Process. and Its Appl.*, vol. 2, no. 465, pp. 652–655, Jul. 1999.

- [25] C. Zhang, J. Wang, S. Han, M. Yi and Z. Zhang, "Automatic real-time barcode localization in complex scenes," in *IEEE Int. Conf. on Image Processing*, pp. 497–500, 2006.
- [26] H. Yang, X. Jiang, and A. C. Kot, "Localization of four extreme corners for barcode images reading using mobile phones," in *IEEE Int. Conf. on Image Processing*, pp. 3897–3900, 2010.
- [27] H. Yang, A. C. Kot, and X. Jiang, "Binarization of low-quality barcode images captured by mobile phones using local window of adaptive location and size," *IEEE Trans. Image Processing*, vol. 21, no. 1, pp. 418–425, 2012.
- [28] K. Bennett, "Linguistic steganography: Survey, analysis, and robustness concerns for hiding information in text," Purdue Univ., West Lafayette, IN, CERIAS Tech. Rep. 2004–13, May 2004.
- [29] A. M. Alattar and O. M. Alattar, "Watermarking electronic text documents containing justified paragraphs and irregular line spacing," *Proc. SPIE*, vol. 5306, Jun. 2004.
- [30] Y. Kim, K. Moon, and I. Oh, "A text watermarking algorithm based on word classification and inter-word space statistics," *Proc. 7th Int. Conf. Document Analysis & Recognition*, Edinburgh, Scotland, UK, pp. 775–779, 2003.
- [31] I. S. Lee and W. H. Tsai, "A new approach to covert communication via PDF files," *Signal Processing*, vol. 90, no. 2, pp. 557–565, 2010.
- [32] P. Wayner, "Mimic functions," *Crypt.*, vol. XVI, no. 3, pp. 193–214, 1992.
- [33] P. Wayner, *Disappearing Cryptography: Information Hiding: Steganography and Watermarking*, 2nd ed. San Mateo, CA: Morgan-Kaufmann, pp. 81–128, 2002.
- [34] *Spam Mimic*, [Online]. Available: <http://www.spammimic.com>.
- [35] M. Chapman, I. D. George, and R. Marc, "A practical and effective approach to large-scale automated linguistic steganography," *Proc. Information Security Conf.*, Malaga, Spain, pp. 156–165, Oct. 2001.
- [36] I. A. Bolshakov, "A method of linguistic steganography based on collocationally-verified synonymy," *Proc. 6th Information Hiding Workshop*, Toronto, ON, Canada, pp. 180–191, May 2004.
- [37] M. H. Shirali-Shahreza and M. Shirali-Shahreza, "A new synonym text steganography," *Proc. Int. Conf. Intelligent Information Hiding and Multimedia Signal Processing*, Harbin, China, pp. 1524–1526, Aug. 2008.

- [38] R. Stutsman, C. Grothoff, M. Attallah, and K. Grothoff, "Lost in just the translation," *Proc. ACM Symp. Applied Computing*, Dijon, France, pp. 338–345, 2006.
- [39] I. J. Lai and W. H. Tsai, "Secret-fragment-visible mosaic image — a new computer art and its application to information hiding," *IEEE Trans. Information Forensics and Security*, vol. 6, no. 3, pp. 936–945, 2011.
- [40] A. Furness, "Machine-readable data carriers - a brief introduction to automatic identification and data capture," *Assembly Automation*, vol. 20, pp. 28–34, 2000.
- [41] X. Hu, W. Zhang, X. Hu, N. Yu, X. Zhao, and F. Li, "Fast estimation of optimal marked-signal distribution for reversible data hiding," *IEEE Trans. Information Forensics and Security*, vol. 8, no. 5, pp. 187–193, May 2013.
- [42] W. B. Pennebaker and J. L. Mitchell, *JPEG: Still Image Data Compression Standard*. New York: Van Nostrand Reinhold, pp. 34–38, 1993.
- [43] E. Reinhard, M. Ashikhmin, B. Gooch, and P. Shirley, "Color transfer between images," *IEEE Computer Graphics and Applications*, vol. 21, no. 5, 2001.
- [44] D. Coltuc and J.-M. Chassery, "Very fast watermarking by reversible contrast mapping," *IEEE Signal Processing Letters*, vol. 14, no. 4, pp. 255–258, 2007.
- [45] R. Z. Wang, C. F. Lin, and J. C. Lin, "Image hiding by optimal LSB substitution and genetic algorithm," *Pattern Recog.*, vol. 34, no. 3, pp. 671–683, 2001.
- [46] C. H. Yang, "Inverted pattern approach to improve image quality of information hiding by LSB substitution," *Pattern Recog.*, vol. 41, no. 8, pp. 2674–2683, 2008.
- [47] Z. Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.
- [48] T. R. Nielsen, P. Drewsen, and K. Hansen, "Solving jigsaw puzzles using image features," *Pattern Recog. Letters*, vol. 29, no. 14, pp. 1924–1933, 2008.
- [49] T. S. Cho, S. Avidan, and W. T. Freeman, "A probabilistic image jigsaw puzzle solver," *Proc. IEEE CVPR*, San Francisco, CA, USA, pp. 183–190, 2010.
- [50] D. Pomeranz, M. Shemesh, and O. Ben-Shahar, "A fully automated greedy square jigsaw puzzle solver," *Proc. IEEE CVPR*, Providence, RI, USA, pp. 9–16, 2011.

- [51] E. Demaine and M. Demaine, “Jigsaw puzzles, edge matching, and polyomino packing: Connections and complexity,” *Graphs and Combinatorics*, vol. 23, pp. 195–208, 2007.
- [52] D. Kundur and K. Karthik, “Video fingerprinting and encryption principles for digital rights management,” *Proc. IEEE*, vol. 92, no. 6, pp. 918–932, 2004.
- [53] S. Lian, Z. Liu, Z. Ren, and H. Wang, “Commutative encryption and watermarking in video compression,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 6, pp. 774–778, 2007.
- [54] M. Cancellaro, F. Battisti, M. Carli, G. Boato, F. G. B. Natale, and A. Neri, “A commutative digital image watermarking and encryption method in the tree structured Haar transform domain,” *Signal Process.: Image Commun.*, vol. 26, no. 1, pp. 1–12, 2011.
- [55] X. Zhang, “Reversible data hiding in encrypted images,” *IEEE Signal Process. Lett.*, vol. 18, no. 4, pp. 255–258, 2011.
- [56] W. Hong, T. S. Chen, and H. Y. Wu, “An improved reversible data hiding in encrypted images using side match,” *IEEE Signal Process. Lett.*, vol. 19, no. 4, pp. 199–202, 2012.
- [57] A. Bronner and C. Monz, “User edits classification using document revision histories,” *Proc. 13th Conf. of the European Chapter of the Association for Computational Linguistics*, Avignon, France, pp. 356–366, 2012.
- [58] A. Bronner, M. Negri, Y. Mehdad, A. Fahrni, and C. Monz, “CoSyne: synchronizing multilingual wiki content,” *Proc. 8th Annual Int. Symp. on Wikis and Open Collaboration (WikiSym)*, Linz, Austria, Article 33, pp. 1–4, 2012.
- [59] C. Dutrey, D. Bernhard, H. Bouamor, and A. Max, “Local modifications and paraphrases in Wikipedia’s revision history,” *Procesamiento de Lenguaje Natural*, vol. 46, pp. 51–58, 2010.
- [60] M. Erdmann, K. Nakayama, T. Hara, and S. Nishio, “Improving the extraction of bilingual terminology from Wikipedia,” *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 5, no. 4, Article 31, 17 pages, 2009.
- [61] A. Max and G. Wisniewski, “Mining naturally-occurring corrections and paraphrases from Wikipedia’s revision history,” *Proc. LREC 2010, Valletta*, Malta, pp. 3143–3148, 2010.

- [62] R. Nelken and E. Yamangil, “Mining Wikipedia’s article revision history for training computational linguistics algorithms,” *Proc. AAAI Workshop on Wikipedia & Artificial Intell.: An Evolving Synergy*, Chicago, Illinois, USA, pp. 31–36, 2008.
- [63] F. B. Viégas, M. Wattenberg, and K. Dave, “Studying cooperation and conflict between authors with history flow visualizations,” *Proc. SIGCHI Conf. on Human Factors in Computing Systems*, Vienna, Austria, pp. 575–582, 2004.
- [64] T. Y. Liu and W. H. Tsai, “A new steganographic method for data hiding in Microsoft word documents by a change tracking technique,” *IEEE Trans. on Information Forensics and Security*, vol. 2, no. 1, pp. 24–30, 2007.
- [65] L. Bergroth, H. Hakonen, and T. Raita, “A survey of longest common subsequence algorithms,” *Proc. 7th Int. Symp. on String Process. Inf. Retrieval (SPIRE)*, A Coruna, Spain, pp. 39–48, 2000.
- [66] A. Kerckhoffs, “La cryptographie militaire,” *J. Sciences Militaires*, vol. 9, pp. 5–38, 1883.
- [67] A. Biryukov and D. Khovratovich, “Related-key cryptanalysis of the full AES-192 and AES-256,” *Proc. 15th Int. Conf. on The Theory and Application of Cryptology and Information Security (ASIACRYPT)*, Tokyo, Japan, pp. 1–18, 2009.
- [68] A. Bogdanov, D. Khovratovich, and C. Rechberger, “Biclique cryptanalysis of the full AES,” *Proc. 17th Int. Conf. on The Theory and Application of Cryptology and Information Security (ASIACRYPT)*, Seoul, Korea, pp. 344–371, 2011.
- [69] N. Madnani B. J. and Dorr, “Generating phrasal and sentential paraphrases: a survey of data-driven methods,” *Computational Linguistics*, vol. 3, no. 3, pp. 341–387, 2010.
- [70] C. Lin, “Face detection in complicated backgrounds and different illumination conditions by using YCbCr color space and neural network,” *Pattern Recognition Letters*, vol. 28, pp.2190–2200, 2007.
- [71] J. Illingworth and J. Kittler, “A survey of the Hough transform,” *Computer Vision, Graphics, and Image Processing*, vol. 44, no. 1, pp. 87–116, 1998.
- [72] R. C. Gonzales and R. E. Woods, *Digital Image Processing*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, pp. 572–580, 2002.

- [73] W. H. Tsai, "Moment-preserving thresholding: a new approach," *Computer Vision, Graphics, and Image Processing*, vol. 29, no. 3, pp. 377–393, 1985.
- [74] V. Guruswami and M. Sudan, "Improved decoding of Reed-Solomon and algebraic-geometric codes," *IEEE Trans. Inform. Theory*, vol. 45, pp.1757–1767, 1999.



Vitae

Ya-Lin Lee was born in Changhua, Taiwan, R.O.C. on February 2, 1987. She received the B.S. degree in computer science from National Chiao Tung University, Taiwan, in 2009, and works toward her Ph.D. degree at the College of Computer Science, National Chiao Tung University. She has been a research assistant at the Computer Vision Laboratory in the Department of Computer Science at National Chiao Tung University from August 2009. Her current research interests include information hiding, image processing, pattern recognition, machine learning, and data mining.



List of Publications of Ya-Lin Lee

Journal Papers and Book Chapters

- (1) **Y. L. Lee** and W. H. Tsai, “A new secure image transmission technique via secret-fragment-visible mosaic images by nearly-reversible color transformations,” *IEEE Transactions on Circuits and Systems for Video Technology*, accepted and to appear.
- (2) **Y. L. Lee** and W. H. Tsai, “A new data hiding method via revision history records on collaborative writing platforms,” *ACM Transactions on Multimedia Computing, Communications and Applications*, accepted and to appear.
- (3) **Y. L. Lee** and W. H. Tsai, “Reversible data hiding by image encryptions and spatial correlation comparisons,” *Journal of Information Science and Engineering*, accepted and to appear.
- (4) **Y. L. Lee** and W. H. Tsai, “New image steganography via secret-fragment-visible mosaic images by nearly-reversible color transformation,” *Advances in Computing - Lecture Notes in Computer Science (LNCS)*, Vol. 6939, G. Bebis, et al. (eds.), Springer, Berlin/Heidelberg, Germany, pp. 64–74, Sep. 2011.
- (5) **Y. L. Lee** and W. H. Tsai, “A new data transfer method via signal-rich-art code images captured by mobile devices,” *IEEE Transactions on Circuits and Systems for Video Technology*, submitted.

Conference Papers

- (1) **Y. L. Lee** and W. H. Tsai, “New image steganography via secret-fragment-visible mosaic images by nearly-reversible color transformation,” *Proceedings of 2011 International Symposium on Visual Computing*, Las Vegas, Nevada, USA, pp. 64–74, Sep. 2011.
- (2) **Y. L. Lee** and W. H. Tsai, “Signal rich art image — a new tool for automatic identification and data capture applications using mobile phones,” *Proceedings of 38th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2013)*, Vancouver, Canada, pp. 1942–1946, May 2013.

Patents

- (1) **Y. L. Lee** and W. H. Tsai, “LM code — a tool for data transfer applications using mobile devices,” *Republic of China Patent* (pending).
- (2) **Y. L. Lee** and W. H. Tsai, “LM code — a tool for data transfer applications using mobile devices,” *USA Patent* (pending).
- (3) **Y. L. Lee** and W. H. Tsai, “A data hiding method via revision history records on collaborative writing platforms,” *Republic of China Patent* (pending).
- (4) **Y. L. Lee** and W. H. Tsai, “A data hiding method via revision history records on collaborative writing platforms,” *USA Patent* (pending).

